REPRESENTACION ABSTRACTA DE SISTEMAS GRAFICOS.

TEORIA DE OBJETOS GRAFICOS.

Tesis Doctoral:

Dirigida por el

Dr. D. Buenaventura Clares Rodríguez

Presentada por

D. Juan Carlos Torres Cantero

Vº Bº

El Director

Dpt. de Lenguajes y Sistemas Informáticos
Universidad de Granada

Granada, 13 de Mayo de 1992

PROLOGO

Desde los orígenes de la Informática Gráfica, la mayor parte de las investigaciones se han centrado en el desarrollo de algoritmos de visualización. Era lógico que esta fuese la principal preocupación puesto que la generación de imagen es en sí la razón de ser de la Informática Gráfica.

En este ámbito ha sido necesario desarrollar algoritmos eficientes para resolver problemas importantes, tales como dibujo de elementos geométricos, recortado, rellenado, transformación geométrica, eliminación de partes ocultas, iluminación, etc. La eficiencia ha sido un imperativo crucial, ya que los algoritmos anteriores se utilizan como funciones básicas a bajo nivel.

Por otra parte, la continua evolución del hardware hacia sistemas con autenticas prestaciones gráficas, ha hecho necesario que se revisasen los métodos de visualización para adaptarlos a los nuevos sistemas.

Esto ha hecho que se dedicase menos atención a otros problemas, que consideramos son importantes. Este es el caso de la formalización de las funciones de entrada y salida, la normalización de interfases de usuario, la representación de la información gráfica y la utilización de técnicas adecuadas para el análisis y diseño de aplicaciones gráficas.

A medida que la disciplina ha ido alcanzando un cierto nivel de madurez, se ha despertado el interés por estos aspectos. Consideramos que todos ellos están íntimamente relacionados, y que es posible encontrar una aproximación teórica global a los mismos, que sirva de marco general al desarrollo de cualquier aplicación gráfica.

Este trabajo se ha desarrollado con el propósito de avanzar en el establecimiento de una base teórica uniforme para la Informática Gráfica, con este fin se propone una abstracción matemática de un sistema gráfico. El modelo presentado posee un buen comportamiento matemático. A lo largo del trabajo se introducen algunas aplicaciones de la teoría a casos prácticos, para mostrar la concordancia de la abstracción con la realidad. No obstante somos conscientes de que, validar el modelo conllevará demostrar su aplicación a la mayoría de las aplicaciones prácticas. En este sentido queda aún un largo camino por recorrer.

La memoria consta de siete capítulos. El primero justifica la necesidad del formalismo y discute su utilidad e importancia siguiendo el enfoque de la teoría general de sistemas. El capítulo concluye realizando un breve resumen de los puntos de contacto con desarrollos anteriores.

El capítulo segundo introduce el concepto de objeto gráfico, junto con algunos conceptos auxiliares: aspecto, presencia, transformación. En este capítulo se introducen también los conceptos de instancia gráfica y familia de objetos.

En el capítulo tercero se definen operaciones de modelado. De las distintas operaciones se estudian sus propiedades y se demuestra que el conjunto de objetos posee estructura de espacio vectorial y álgebra de boole.

El capítulo cuarto se dedica a la definición y estudio de funciones sobre el conjunto de objetos y a la visualización. Se muestra como realizar operaciones habituales (recortado y cambio de atributos de visualización) utilizando funciones. La visualización se define en base al formalismo de Fiume, demostrándose que para los objetos gráficos que son abstracción de sistemas físicos, existe un objeto Fiume equivalente.

En el capítulo quinto se muestra como aplicar el formalismo desarrollado a tres problemas concretos: la estructuración de un sistema de dibujo, la obtención de condiciones necesarias para obtener un sólido por barrido de dos elementos, y la formalización de la animación basada en Key-Frame.

El capítulo sexto se dedica a la aplicación del formalismo en la especificación formal de sistemas gráficos. Se estudia la posibilidad de extender un lenguaje de especificación algebraica para especificar sistemas gráficos utilizando la teoría de objetos gráficos y sistemas interactivos. De estos últimos se da una caracterización de especificación correcta. Por último se propone una metodología de especificación para sistemas gráficos.

El capítulo séptimo resume las conclusiones y principales aportaciones del trabajo.

El apéndice A contiene una breve descripción del lenguaje de especificación algebraico ALPLA, que se referencia en capítulo sexto.

CONTENIDO

Símbolos

A	Dominio de aspecto
$\mathbf{C}_{rgb}$	Dominio de aspecto RGB, definido en el ejemplo 2.1.
$\mathbf{D}_{ef}$	Dominio efectivo de una instancia
î	Transformación identidad
n	Dimensión del espacio
N	Conjunto de los números naturales
O	Conjunto de objetos
P	Dominio de presencia
P	Punto de $\mathbf{R}^n$
R_N	Instancia normal
R	Cuerpo de los número reales
Z	Conjunto de los números enteros
V	Volumen de un objeto
α	Función de aspecto
Γ	Subconjunto de $\mathbf{R}^{n+1}$. $\Gamma = \{ P \in \mathbf{R}^{n+1} \mid p_{n+1} = 1\}$
τ	Transformación geométrica
μ	Función de presencia
ν_i	Vector unitario i-ésimo de la base canónica de $\mathbf{R}^n$
ν_0	Punto en el origen de coordenadas
Ω	Conjunto de instancias

Notación

ESCALARES: Los escalares y las coordenadas de puntos y vectores se denotarán por letras minúsculas.

CONJUNTOS: Los conjuntos se representarán, en general, mediante letras mayúsculas en negrita.

PUNTOS y VECTORES: Los puntos y vectores se representarán por letras mayúsculas o como una n-upla de coordenadas, usando la misma letra para denotar el punto y sus coordenadas. Así, por ejemplo, la n-upla

$(x_1,x_2,..,x_n)$ representa al Punto o vector $X \in \mathbf{R}^n$

En algunos se interpretarán los puntos como vectores, delimitados por el punto en cuestión y el origen, para realizar operaciones sobre $\mathbf{R}^n$ considerado como espació Euclideo.

REPRESENTACION BINARIA: La representación binaria de un escalar se representará como la yuxtaposición de bits. Cada bit se denotará utilizando la letra del escalar con el superíndice b y un subíndice para indicar su posición. Así:

$$c_0^b c_1^b .. c_{p-1}^b c_p^b$$

es la representación binaria de $c \in \mathbf{Z}$ con $p+1$ bit.

BASE DE $\mathbf{R}^n$: Denotaremos los vectores de la base canónica de $\mathbf{R}^n$, considerado como espacio Euclideo, por $\nu_1,\nu_2,..,\nu_n$ y por ν_0 el origen de coordenadas

1. INTRODUCCION.

1. INTRODUCCION.

La abstracción es fundamental en la resolución informática de cualquier problema. Cuando hacemos una abstracción necesitamos un modelo conceptual en el que poder desenvolvernos. Una buena parte del presente trabajo se va a dedicar a establecer un marco conceptual útil para el desarrollo de aplicaciones gráficas.

Por este motivo comenzaremos analizando el papel que juega la abstracción en la resolución informática de un problema. De forma general podemos decir, que cuando nos planteamos resolver un problema estamos trabajando con un sistema[1], existente o no, del que conocemos determinada información, relativa tanto a su estructura (**características**) como a su comportamiento (**fenómenos**), y del que deseamos obtener información adicional, que es consecuencia de la información conocida.

En cualquier caso, la naturaleza del problema es en sí irrelevante para el objetivo que perseguimos ahora. Así, en el planteamiento anterior se podrían considerar problemas tan dispares como:

· El estudio de un sistema biológico. P.e. un ecosistema del que se conoce su estructura (la distribución de población de las distintas especies) y su comportamiento (las interacciones entre especie) y del que se desea calcular la estructura después de introducir una nueva especie.

· Un sistema administrativo. P.e. el sistema de gestión de nominas de una empresa.

· Un sistema físico. P.e. un deportista que se desplaza por una pista.

También es irrelevante el que el sistema exista realmente. La diferencia está tan sólo en el significado que tiene la información conocida del sistema, que puede ser datos tomados del sistema existente o requisitos para un sistema a diseñar.

De un modo general, se puede considerar que la aplicación de medios informáticos para resolver un problema implica realizar una representación de las características y los fenómenos relevantes del mismo, para posteriormente, con la representación de las características 'simular' los fenómenos del sistema.

Indudablemente, la representación anterior, puede realizarse de distintas formas, dependiendo de cual sea 'la óptica con que se observe el sistema', en esta 'óptica' influye el

---

[1] Utilizaremos aquí el término sistema para referirnos tanto a tipologías de sistemas como a sistemas concretos, ya que normalmente el nivel al que se utiliza el término está claro en el contexto.

lenguaje que se utiliza para realizar la representación, la finalidad de la misma y la percepción 'subjetiva' de la persona que la realiza. Dicha 'óptica' se conoce como **sistema de representación** [LeMo90]. Siempre que se resuelve un problema se está utilizando un determinado sistema de representación, en base al cual se realiza la representación de las características y fenómenos, que constituye lo que se denomina un **modelo** del sistema (ver fig. 1).

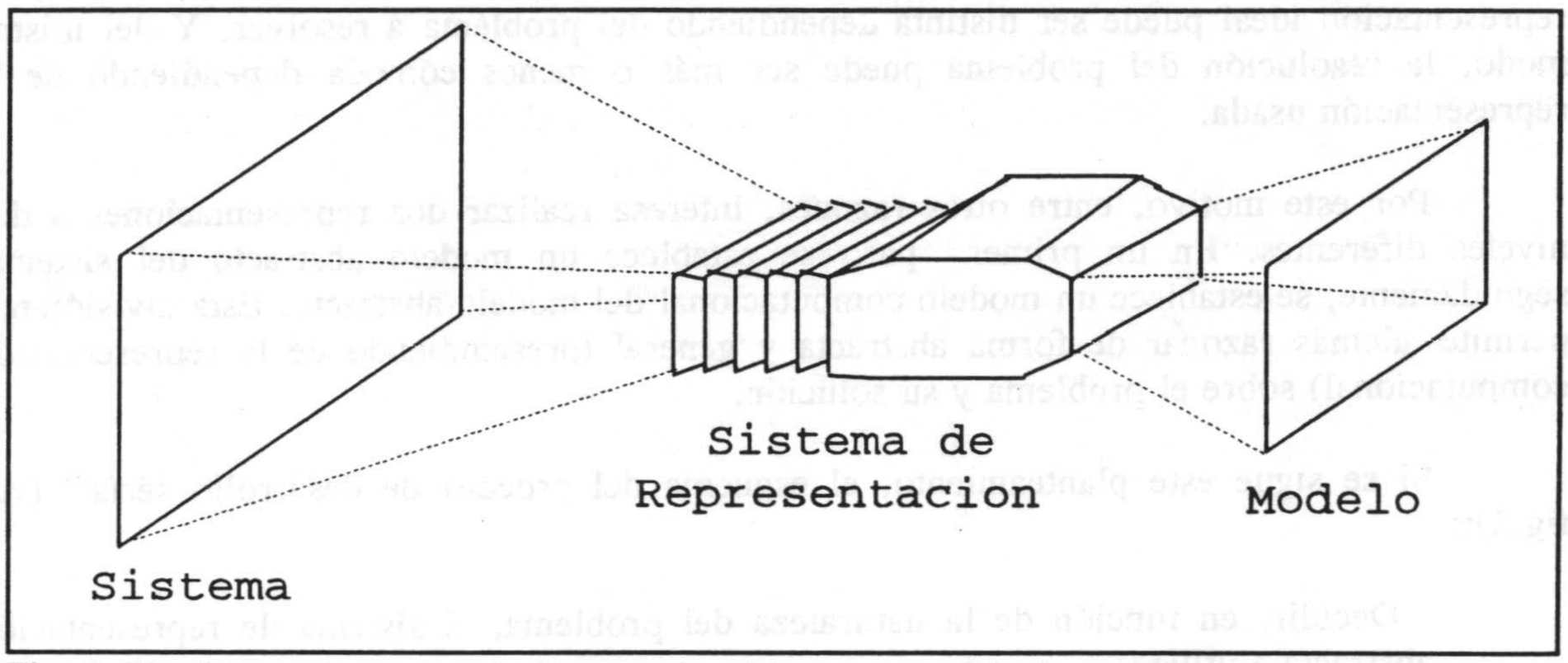


Fig. 1. Un sistema de representación permite obtener un modelo de un sistema.

La realización de la representación constituye, en sí, un proceso de abstracción, en el que, de acuerdo con el sistema de representación utilizado, se genera una representación, que no es más que una abstracción del sistema. El nivel de dicha abstracción estará determinado por el nivel del lenguaje usado en el sistema de representación. En el ámbito informático interesa distinguir dos tipos de modelos: **modelos computacionales** o **informáticos** y **modelos abstractos**. Los primeros se obtienen cuando el sistema de representación utiliza un 'lenguaje' informático, los segundos cuando el lenguaje usado es abstracto. Así, por ejemplo, para resolver problemas de movimiento de sólidos podemos usar un modelo de representación basado en la mecánica de Newton, lo que nos generará, para cada problema un modelo abstracto.

En cualquier caso, el fin del modelo es facilitar el estudio del sistema, y por tanto, al elegir el sistema de representación se deberá de tener en cuenta que de los resultados obtenidos sobre el modelo se deben de poder inferir conclusiones relativas al sistema. Por tanto es deseable que la representación, considerada como aplicación matemática sea biyectiva (cosa que no siempre será factible).

Nos centraremos ahora en el proceso de solución informática del problema, que podemos

subdividir en dos etapas[1]:

- el establecimiento del modelo computacional.

- y la resolución del problema sobre dicho modelo.

Estas dos etapas están íntimamente relacionadas. Para un sistema dado, la representación ideal puede ser distinta dependiendo del problema a resolver. Y del mismo modo, la resolución del problema puede ser más o menos cómoda dependiendo de la representación usada.

Por este motivo, entre otras razones, interesa realizar dos representaciones a dos niveles diferentes. En un primer paso se establece un modelo abstracto del sistema, seguidamente, se establece un modelo computacional del modelo abstracto. Esta división nos permite además razonar de forma abstracta y general (prescindiendo de la representación computacional) sobre el problema y su solución.

Si se sigue este planteamiento, el esquema del proceso de desarrollo sería[2] (ver fig.2)::

· Decidir, en función de la naturaleza del problema, el sistema de representación abstracta a utilizar.

· Con el sistema de representación anterior se establece el **modelo abstracto** del sistema.

· Se estudia dicho modelo abstracto para decidir el sistema de representación computacional a usar.

· Utilizando dicho sistema de representación se deriva el **modelo computacional** a partir modelo abstracto.

A cada uno de estos niveles pueden plantearse distintas alternativas, que condicionan, en un sentido u otro, la solución final.

---

[1] La división se establece en base al papel jugado por la abstracción del proceso de representación, y por tanto es independiente del modelo de ciclo de vida para el posible desarrollo del software.

[2] En todo lo anterior no se ha tenido en consideración el ciclo de vida del software, por que el proceso descrito es independiente de él, puesto que se trata de una abstracción del proceso que no contiene aspectos temporales. Así si, por ejemplo, se utilizase un ciclo de vida clásico, el desarrollo de la aplicación informática, implicaría el análisis del problema a resolver (que se realiza a nivel de sistema), la especificación de requisitos de la aplicación, (que se puede realizar en los niveles de sistema y de modelo abstracto) y el diseño y codificación de la aplicación (trabajando en los niveles de los modelos abstracto y computacional).

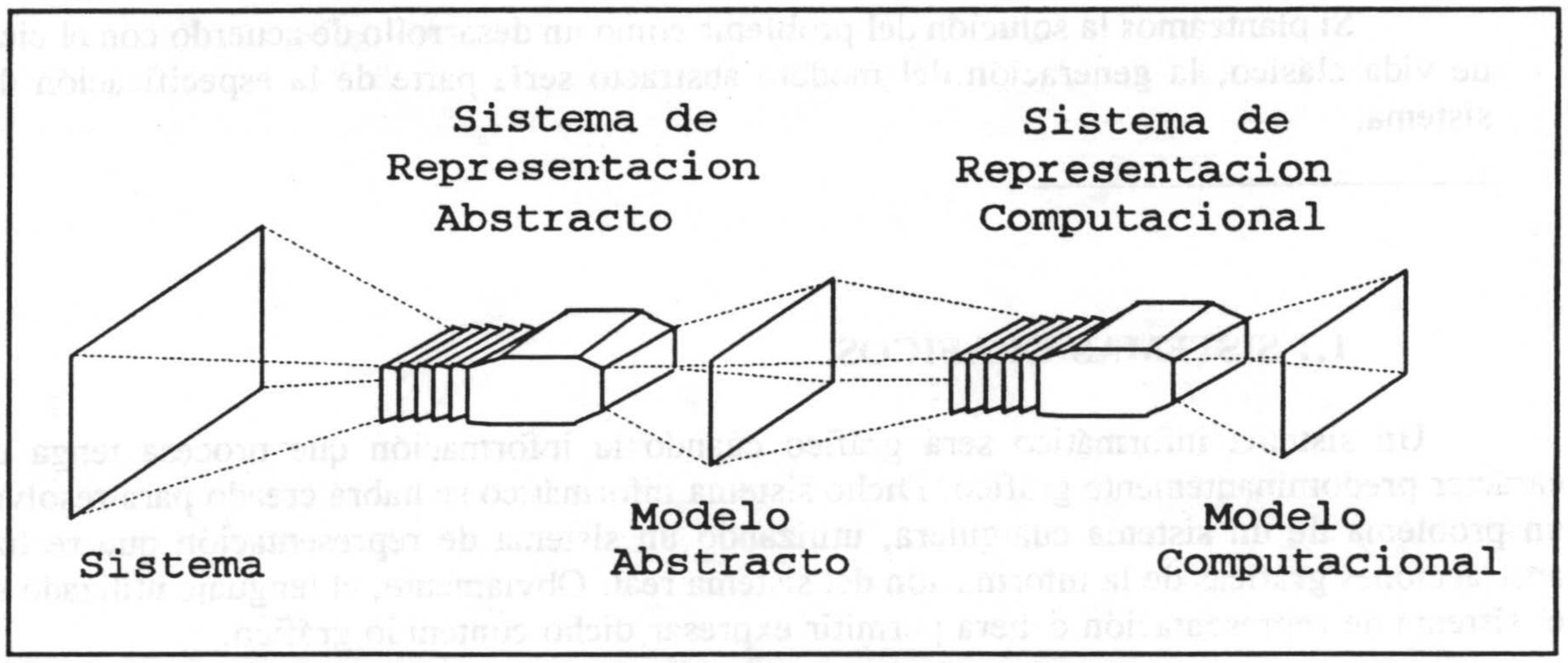


Fig. 2. Jerarquía de modelos en la realización de una solución informática.

El modelo abstracto utilizado podrá o no ser formal, dependiendo del lenguaje en el que este expresado. Entenderemos que el modelo es formal cuando esté expresado matemáticamente mediante reglas precisas, que permitan expresar y demostrar propiedades del sistema. El hecho de que el modelo establecido tenga un carácter formal, es importante en el proceso de desarrollo de la solución informática, ya que, además de permitir una mayor precisión en la descripción, permitirá la utilización de métodos formales de especificación y validación [Wing90].

Ejemplo 1.1:

A modo de ejemplo podemos pensar en dos problemas que impliquen la representación de un cubo en el espacio: el cálculo de la velocidad de caída desde una altura dada, y la generación de una imagen de una escena formada por dados. En cualquiera de los casos el primer paso a realizar es decidir el sistema abstracto de representación a utilizar, que estará lógicamente condicionado por la finalidad de la representación, así para el primer problema podremos usar la mecánica de Newton como sistema de representación abstracto; para el segundo se podrá usar como lenguaje cualquier forma de descripción de la posición, orientación y textura de los dados.

Utilizando el sistema de representación elegido se podrá generar un modelo abstracto para cada problema. En el primer caso, el modelo contendrá información de la posición y orientación de los dados, de su tamaño y de su peso, así como de la geometría del escenario en que se encuentran, y de las fuerzas que se aplican a ellos. En el segundo caso contendrá información de la posición y orientación de los dados, de su tamaño, color y textura, y posiblemente de la posición de fuentes de luz y del observador.

El hecho de disponer de un modelo abstracto formal, simplifica la descripción de cada modelo, ya que parte de la información del comportamiento del modelo se deduce del lenguaje formal que se utiliza en su descripción (como la mecánica de Newton). Además permite razonar sobre el problema y su solución.

Si planteamos la solución del problema como un desarrollo de acuerdo con el ciclo de vida clásico, la generación del modelo abstracto sería parte de la especificación del sistema.

1.1 SISTEMAS GRAFICOS.

Un sistema informático será gráfico cuando la información que procesa tenga un carácter predominantemente gráfico. Dicho sistema informático se habrá creado para resolver un problema de un sistema cualquiera, utilizando un sistema de representación que realiza abstracciones gráficas de la información del sistema real. Obviamente, el lenguaje utilizado en el sistema de representación deberá permitir expresar dicho contenido gráfico.

Al nivel del sistema informático la información gráfica puede aparecer en [ver fig.3]:

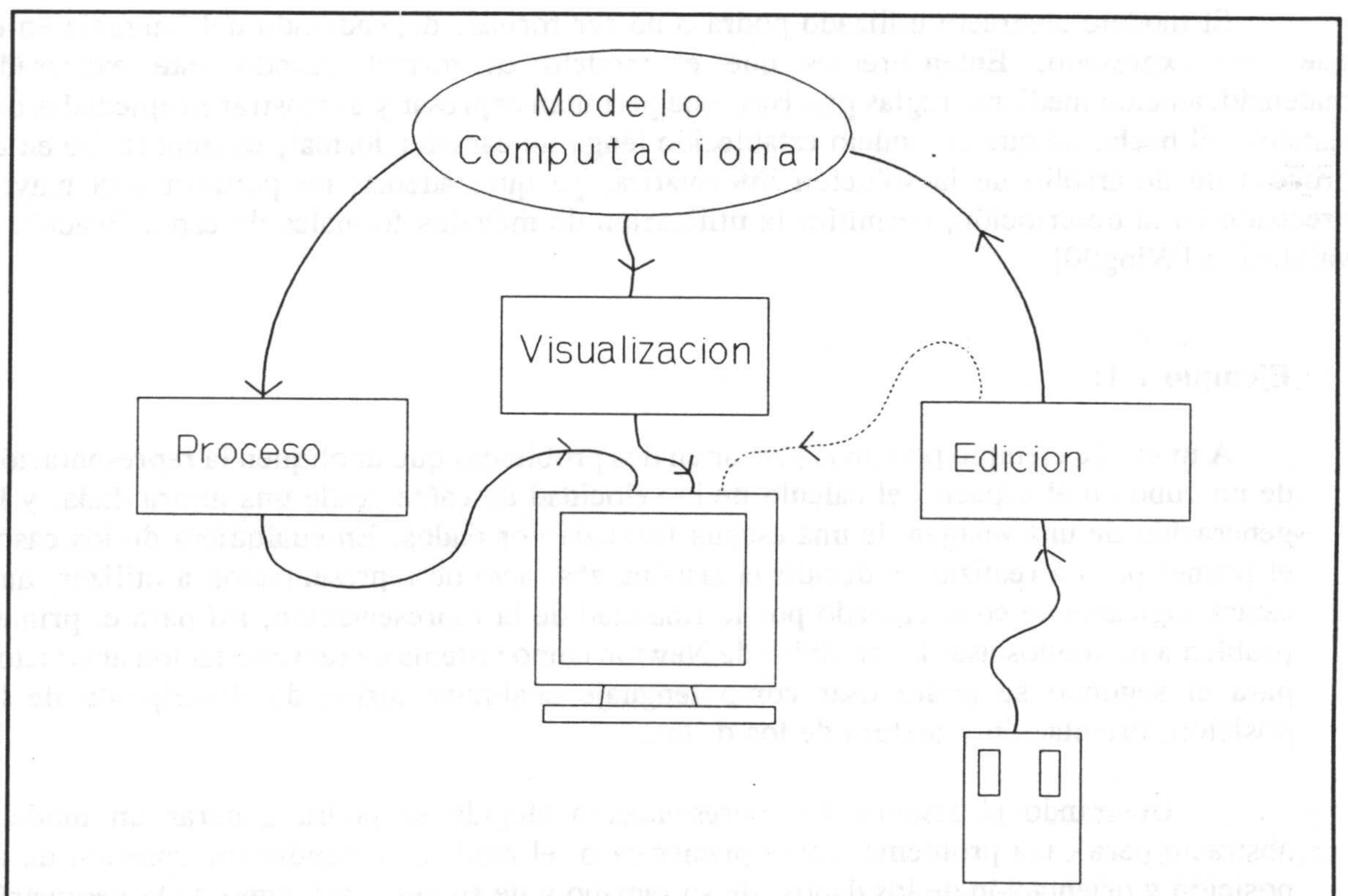


Fig.3: Esquema de un sistema gráfico

- La entrada. Se introduce información de carácter gráfico.
- La salida. Se genera información de carácter gráfico.
- El proceso. El proceso en sí utiliza información de carácter gráfico.

Es decir en cada uno de los subsistemas de un sistema informático: subsistema de entrada, subsistema de salida y tratamiento. Por tanto podremos decir que un sistema informático es gráfico cuando alguno de estos subsistemas gestiona información gráfica.

Este planteamiento es más amplio que el considerado clásicamente. Así por ejemplo, Beatty define la informática gráfica como : "La Informática Gráfica es la ciencia de la generación de imágenes gráficas con ordenador" [Beat82].

Algunos autores (B.Herzog, R.Cabezas), consideran como informática gráfica, tan sólo a la generación de imágenes sintéticas [Cabe90].

No obstante, consideramos que es más apropiado el considerar a la informática gráfica en sentido amplio. Si se quiere entender a la informática gráfica como una disciplina, habrá que caracterizarla por la utilización de un conjunto de técnicas propias (diseño, estructuras de datos, algoritmos, etc.), e indudablemente existen problemas que no implican la realización de imágenes y que se resuelven utilizando el mismo tipo de técnicas. Sirva de ejemplo la determinación de la longitud de una curva de nivel de terreno a partir de las cotas de varios puntos.

Actualmente se tiende a considerar que incluso la síntesis de imágenes y el análisis de imágenes poseen muchos puntos de contacto, y que en algunos aspectos se superponen [Pun_90]. La definición dada anteriormente está más en consonancia con esta última tendencia.

La estructura del sistema informático desarrollado podrá ser distinta dependiendo de cuales de las condiciones anteriores se cumplan. Así, se podrá hablar de problemas de reconocimiento de formas, de tratamiento de imágenes, de visualización, de generación de imágenes, de modelado, etc.

El proceso de solución informática de un problema relativo a un sistema gráfico es, en principio, el mismo al seguido en la solución de problemas sobre cualquier otro tipo de sistemas por medios informáticos. No obstante, aparecen dos diferencias importantes:

- La información a representar tiene una fuerte componente geométrica. Además, la estructura geométrica del sistema suele estar también condicionada, al menos en parte, ya que los sistemas a resolver suelen ser sistemas físicos o tener una representación gráfica ya definida.

- Tanto la creación como la interpretación del modelo conlleva la definición de elementos geométricos o su visualización.

Estas diferencias se deberán de tener en cuenta a la hora de establecer tanto el modelo abstracto como el modelo computacional. Además, para el desarrollo de un modelo abstracto válido para un universo suficientemente grande de problemas se plantean dos problemas fundamentales:

· Hay distintos aspectos, tanto en la visualización como en la definición de geometrías,

en los que intervienen un gran número de disciplinas. En este sentido son fundamentales: la Optica, la Teoría del Color, la Fisiología y Psicología de la Visión, la Teoría de la señal, la Geometría, entre otras[1]. Esto hace que se deban de considerar un gran número de parámetros de naturaleza diversa, y que por tanto son difíciles de tratar con los mismos métodos.

· No existe un formalismo que permita el tratamiento a nivel abstracto de todos los componentes de un sistema gráfico.

A nivel formal, el establecimiento de una base teórica para la informática gráfica conllevará la formalización de cada uno de los aspectos específicos que aparecen en las aplicaciones gráficas. En este sentido se deberá de establecer un modelo teórico para el modelado, la visualización y la interacción.

---

[1] Una revisión completa de las disciplinas que sirven de fundamento se puede encontrar en [Pun_90].

1.2 OBJETIVOS

Este trabajo pretende establecer un marco teórico general, que sirva de soporte para la resolución de problemas gráficos. Concretamente este esfuerzo se centra en:

- Definición de una teoría, que pueda ser usada como soporte del lenguaje de representación de sistemas gráficos.

- Definición de un formalismo que permita la generación de modelos abstractos formales de sistemas gráficas. Dicho formalismo deberá, por lo expuesto anteriormente, contemplar:

  · La formalización de las operaciones de modelado.

  · La formalización de las funciones de entrada.

  · La formalización de funciones de visualización.

- Establecimiento de metodologías y técnicas adecuadas para la especificación de aplicaciones gráficas.

1.3 ANTECEDENTES.

De acuerdo con la propuesta de objetivos establecida anteriormente, realizaremos una valoración de las principales aportaciones efectuadas hasta la fecha, analizando por separado distintos campos, que de uno u otro modo, tocan aspectos relacionados con los objetivos antes mencionados. Concretamente se describe el estado de los siguientes campos: Normalización, Modelado de sólidos y superficies, Formalización de interacción y visualización, y Especificación y diseño de aplicaciones gráficas.

1.3.1 Normalización.

El proceso de normalización ha sido muy importante porque, por un lado ha supuesto una primera definición de las operaciones de entrada y visualización, así como de estructuras de modelado, y, por otro lado, la necesidad de definir y validar los propios estándares y sus implementaciones ha servido de motor para el desarrollo de técnicas formales de especificación y verificación.

La normalización ha venido condicionada por la dinámica del mercado, y por tanto a afectado casi exclusivamente a productos, fundamentalmente a librerías de funciones gráficas, que habitualmente se conocen como estándares gráficos. Las funciones que aparecen en la librería realizan operaciones gráficas de entrada, de salida o de control. La estandarización establece las interfases de la librería. Una de las interfases más importantes de toda librería es la dada por las llamadas a la librería. Los estándares especifican las llamadas que deben de aparecer en la librería, estableciendo su semántica. Para especificar la sintaxis se debe de fijar el lenguaje de programación. Así, para cada estándar existe una descripción de la librería y diversas descripciones de la sintaxis de las llamadas para diferentes lenguajes de programación. La estandarización asegura, por tanto, la portabilidad para un lenguaje dado.

Se han desarrollado varios estándares: GKS [Hopg86b][Bono88], CORE, PHIGS. La inercia del mercado ha hecho que no se rechazasen los estándares más antiguos en favor del PHIGS, sino que se ha tendido a extender el GKS para permitir su utilización en un mayor número de situaciones (GKS-3D).

Los tres estándares más utilizados actualmente son:

GKS: Graphical Kernel System (ISO 7942)

GKS-3D: Graphical Kernel System Extensions for Three Dimensions (ISO 8805)

PHIGS: Programmer's Hierarchical Interactive Graphics Systems (ISO/IEC 9592)

Una valoración de las ventajas e inconvenientes de cada uno, según el tipo de aplicación a desarrollar se puede encontrar en [Bett88], y una revisión de las implementaciones de GKS y PHIGS se puede encontrar en [Wirw89].

GKS-3D mantiene compatibilidad con GKS, añadiendo a éste último funciones de entrada y salida en tres dimensiones. Concretamente, el GKS-3D añade al GKS:

· Primitivas de dibujo 3D

· Transformación de visualización 3D.

· Entrada 3D.

· Acceso a algoritmos de eliminación de líneas y caras ocultas.

El PHIGS se diferencia del GKS (y del GKS-3D) en la estructuración del dibujo. En GKS, las primitivas de salida o bien no se almacenan (modo inmediato), o bien se almacenan en una estructura interna a la librería. La estructura de datos interna está cerrada al usuario, es decir, no existe mecanismo para editar su contenido. Dicha estructura está formada por un conjunto de segmentos, cada uno de los cuales es un conjunto de primitivas de salida. Con los segmentos sólo se pueden realizar operaciones para añadir primitivas, borrar todo el segmento, cambiar atributos, o copiar (indirectamente).

En PHIGS todo la información gráfica generada debe de encontrarse en la estructura de dibujo ('Central Structure Store'). Dicha estructura constituye una jerarquía (más exactamente un grafo acíclico dirigido) de elementos gráficos, editable por el usuario. Esto es, la librería dispone de rutinas para editar los nodos de la jerarquía ('estructura' en PHIGS). Esta característica del PHIGS lo hacen especialmente idóneo cuando se necesite actualizar parte del dibujo, o cuando el dibujo tenga una estructura compleja, ya que normalmente la descomposición de cualquier sistema conducirá a la generación de una jerarquía [Fole90], pero no es valida para cualquier tipo de sistema [Hard87].

Los diferentes estándares coinciden en el establecimiento de un conjunto de funciones de entrada y salida, que por tanto se pueden entender como primitivas mínimas de un sistema gráfico. No obstante, no se ha conseguido que el comportamiento del estándar sea independiente del hardware.

Podemos concluir que los estándares, en sí, no sirven como lenguaje para representar, de forma abstracta y general los sistemas gráficos.

1.3.2 Modelado gráfico.

Podríamos definir el modelado gráfico, citando a Salmon, como: "el proceso según el cual un objeto gráfico complejo se construye utilizando componentes simples" [Salm87]. En esta definición la construcción se puede entender a dos niveles: a nivel de modelo abstracto o a nivel de modelo computacional. En este apartado nos centraremos en el nivel de modelo

abstracto, de acuerdo con los objetivos planteados.

A nivel conceptual, lo que interesa no es la estructura de la información contenida en el modelo, ya que el proceso de composición de un sistema complejo con elementos más simples nos conduce siempre a una estructura jerárquica. El problema es el de establecer unos criterios de descomposición lo suficientemente generales como para que se puedan aplicar en una familia amplia de sistemas. En este sentido, cabe destacar los estudios realizados sobre modelado de sólidos y superficies.

El modelado de sólidos ha sido profusamente estudiado [Baer_79, Casal85, Kunii85B, Manty82, Olive85, Requi80, Requi82, Requi83, Samel85, Schoo83]. Requicha lo definió como: "el conjunto de teorías, técnicas y sistemas orientados a la representación 'completa en cuanto a información' de sólidos. Dicha representación debe permitir (al menos en principio) calcular automáticamente cualquier propiedad bien conocida de cualquier sólido almacenado" [Requi83].

Un primer problema a resolver es la determinación del concepto de sólido. Requicha los define del siguiente modo: "Un sólido es un subconjunto del espacio Euclideo que posee las siguientes propiedades:

(1) Rígido: El sólido tiene una forma fija e invariante que es independiente de su posición y orientación.

(2) Homogéneo: El sólido debe de tener un interior homogéneo, sin que existan zonas aisladas.

(3) Finito: Debe de ocupar una porción finita del espacio.

(4) El conjunto de sólidos es cerrado bajo movimientos rígidos y ciertas operaciones booleanas: Las transformaciones rígidas (giro o traslación) y operaciones que añadan o eliminen materia deben de producir nuevos sólidos.

(5) Describible finitamente: Debe de existir algún aspecto del sólido que asegure el que se pueda representar como una secuencia finita de datos.

(6) Determinado por el contorno: El contorno debe de determinar sin ambigüedad al interior." [Requi80]

Matemáticamente, los solidos se pueden describir como subconjuntos del espacio Euclideo que sean acotados, cerrados, regulares y semianalíticos [Requi80].

Un método de modelado de sólidos, en tanto que sistema de representación, se puede entender como una relación entre sólidos y modelos de éstos. Esto es, si **S** es un conjunto de sólidos, y **A** es un conjunto de abstracciones de sólidos o modelos, un método de modelado **M** es una relación:

M: S --> A

Los métodos de modelado de sólidos pueden tener los siguientes atributos y propiedades [Requi80]:

(1) Dominio: El dominio del método de modelado es el conjunto de sólidos representables por él. Idealmente será el conjunto de todos los sólidos posibles. Obviamente será siempre un subconjunto de **S**, esto es

$$\text{Dominio}(M) \subset \mathbf{S}.$$

(2) Validez: Un modelo es inválido si no corresponde a ningún sólido real. Es importante el que el método de modelado asegure de por sí la validez de todos los modelos que se puedan crear o que existan procedimientos automáticos para comprobar la validez de los modelos. En el primer caso diremos que el método de modelado es válido. Un método de modelado es válido si la aplicación objeto-modelo es suprayectiva, es decir rango(M) = **A**.

(3) Completitud o no ambigüedad: Un método de modelado es no-ambiguo si cada modelo se corresponde, a lo sumo, con un único objeto real. Es deseable que el método no sea ambiguo. La no-ambigüedad es necesaria para poder determinar la igualdad de dos objetos a partir de sus modelos. Matemáticamente la no ambigüedad equivale a que la aplicación objeto-modelo sea inyectiva. Formalmente

$$m(s_1) = m(s_2) \Rightarrow s_1 = s_2$$

(4) Unicidad: Se dice que existe unicidad en el método de modelado si cada objeto real tiene, a lo sumo, un posible modelo. Si el método no asegura la unicidad no es posible comprobar la igualdad de objetos sobre el modelo. Esta condición es indispensable para que el método de modelado se pueda considerar una aplicación matemática. Es decir:

$$\text{si } a_1 = m(s) \text{ y } a_2 = m(s) \Rightarrow a_1 = a_2$$

Además de estas propiedades formales, se suelen considerar las siguientes propiedades informales de los métodos de modelado:

(1) Conciso: La concisión de un método esta relacionada con el contenido en información de los modelos.

(2) Facilidad de creación: Los métodos de modelado deben permitir la construcción de modelos. La facilidad de creación está relacionada con la concisión.

(3) Eficiente en el contexto de la aplicación: Con los modelos se suelen realizar operaciones y cálculos. El método de modelado debe estar orientado a los algoritmos que van a operar con los modelos.

M. Mäntylä enumeró como principales problemas de los métodos de modelado de sólidos [Mänt89] los siguientes:

- Inexactitud numérica.
- No adaptación a modelos grandes.
- Dominio limitado
- Dificultad para representar propiedades no geométricas.
- Deficientes interfases de usuario.

Se han propuesto diversos métodos de modelado de sólidos. La principal diferencia entre los distintos métodos está en sus dominios, aunque no obstante, existe también una gran diversidad en sus propiedades. A continuación se relacionan los principales métodos de modelado de sólidos propuestos [Requi80]:

Alambres: Los objetos se descomponen en una colección de aristas. sólo se puede usar para objetos delimitados por caras planas. Se puede generalizar utilizando perfiles curvos en lugar de líneas. En cualquier caso es ambiguo, y no único.

Muestreo del contorno: El objeto se representa por una colección finita de puntos del contorno. Es un modelo ambiguo y no único. Se utiliza tan sólo como paso intermedio en la entrada de sólidos ya existentes.

Instancias: Se utiliza un determinado repertorio de objetos. Cada uno de ellos está caracterizado por un conjunto de parámetros. No hay mecanismos de combinación de objetos. Si el repertorio de objetos base es independiente, el método de modelado presenta unicidad. No es ambiguo. Su dominio está limitado, y condicionado por el repertorio de objetos base.

Descomposición celular: Los objetos se descomponen en un conjunto de tetraedros que son disjuntos dos a dos (salvo intersecciones en caras o vértices). No es ambiguo.

Barrido: Un objeto generado por barrido se descompone en un objeto más simple (que suele ser una superficie), y una trayectoria (que es una curva en el espacio). El objeto está definido por el volumen barrido por la superficie al desplazarse a través de la trayectoria. El desplazamiento puede ser lineal, es decir sin cambiar de orientación o rotacional. El principal problema de la representación de sólidos por barrido, además de su dominio limitado, es el que no existan procedimientos para determinar si el cuerpo generado es un sólido válido, ni para determinar sus propiedades [Requ80]. El método no es ambiguo.

Fronteras: Se considera al sólido como el volumen encerrado por una superficie cerrada. La descomposición del objeto es por tanto la de su frontera, que se descompone en caras, usualmente planas. Las caras se descomponen en lados y éstos en vértices.

Geometría constructiva de sólidos: Se parte de un conjunto de solidos simples, que se definen geométricamente a partir de parámetros. Sobre estos objetos se definen

operaciones booleanas (unión, intersección, diferencia). Sobre esta base, un solido cualquiera se descompone en solidos primarios según las operaciones anteriores. Es decir, la descomposición del sólido es una expresión matemática en la que aparecen tan sólo solidos primarios. La validez del modelo está por tanto garantizada por el soporte matemático. Posee un buen dominio y un buen comportamiento como método, pero su visualización es complicada.

Enumeración espacial jerárquica (octtree): Se considera al espacio dividido en pequeñas celdas cúbicas (o voxels). Cualquier objeto se descompone en el subconjunto de celdas que ocupa. Usualmente el modelo del objeto se almacena en memoria como un árbol octal (octtree). Dan lugar a algoritmos simples. Es fácil establecer la validez del modelo. El modelo es por tanto una aproximación. Se han realizado diversos intentos para extender el dominio de este método [Ayal88].

Analítico (ASM): El objeto se descompone en un conjunto de pequeños elementos disjuntos. Cada elemento ("hiperpatch") se obtiene por deformación de un cubo unitario, con un formalismo semejante al de modelado de superficies.

Para cada método existe una forma natural de definir y editar el modelo, no obstante se pueden realizar determinadas combinaciones. Así es frecuente utilizar un modelo B-Rep y una definición por barrido.

Cada método tiene ventajas y limitaciones. Por este motivo la mayor parte de los sistemas de modelado utilizan métodos mixtos, en los que se mantienen simultáneamente varios modelos del mismo objeto, cada uno con un método distinto. Cada modelo se utiliza para realizar determinadas operaciones, reconstruyendo uno a partir del otro cuando sea necesario. Una combinación usual es la de CSG con B-rep.

Respecto al **modelado de superficies**, la situación es parecida a la del modelado de sólidos. En la práctica se utilizan un gran número de métodos, cada uno de los cuales funciona bien en determinadas situaciones.

La diferencia fundamental, con el modelado de sólidos, está en la existencia de un soporte matemático común. Las curvas suelen describirse como funciones matemáticas. A nivel de ecuación, las superficies se pueden representar de las siguientes formas:

Explícitas: sólo se puede usar para superficies univaluadas. Se utiliza para representación de datos experimentales.

$$z = f(x,y)$$

Implícitas: Se suele usar para describir cuádricas. Poseen la ventaja de ser orientables, es decir permiten determinar a que lado de la curva se encuentra un punto cualquiera del espacio.

$$f(x,y,z) = 0$$

Paramétricas: Se suele usar con ecuaciones bicuádricas o bicúbicas a trozos. Tiene la ventaja de ser flexible. Al igual que la ecuación explícita no es orientable.

$$x = f_1(u,v),\ y = f_2(u,v),\ z = f_3(u,v)$$

La diferencia entre la descripción de una curva o una superficie está tan sólo en la dimensión del lugar geométrico definido por la ecuación. Así, la ecuación explícita

$$z = f(x,y)$$

describe una curva (posee dos grados de libertad), mientras que el par de ecuaciones

$$z = f_1(x) \quad y = f_2(x)$$

describe una curva en el espacio.

Usualmente se utiliza la descripción paramétrica, salvo que se trate de curvas o superficies univaluadas (obtenidas normalmente como datos experimentales) o se estén describiendo cuádricas. En ambos casos no es frecuente editar la curva, o superficie.

Cuando lo que interesa es diseñar la superficie, se utilizan ecuaciones paramétricas. El problema es por tanto encontrar dos o tres ecuaciones (dependiendo de la dimensión del espacio) que describan la superficie. Concretamente, para una curva en el plano sería:

$$x = X(u)$$
$$y = Y(u)$$

Estas ecuaciones deben diseñarse interactivamente, siendo por tanto necesario disponer de un método para controlar la forma de la superficie a partir de muy poca información. Un método satisfactorio es definirla a partir de un conjunto finito de puntos de control. Esto equivale a poner las expresiones anteriores en la forma:

$$x = \Sigma_{i=1}^{n}\ x_i \cdot F_i(u)$$

$$y = \Sigma_{i=1}^{n}\ y_i \cdot F_i(u)$$

donde x_i es la coordenada x del i-ésimo punto de control, y las F_i son funciones de forma. Como funciones de forma se suelen tomar funciones polinómicas.

La superficie se crea introduciendo los puntos de control, a partir de los cuales se calcula la ecuación analítica.

Para editar la superficie bastará con modificar la posición de alguno de los puntos de control. La forma en que afectan los cambios de posición de cada punto de control depende las funciones de forma F_i. Un hecho importante es que las curvas y superficies construidas de esta forma son invariantes bajo giros y traslaciones, lo que permite que el realizar dichas operaciones sobre la curva se efectúe transformando los puntos de control.

A nivel general, los métodos de diseño de curvas y superficies pueden tener, según las funciones de forma que utilicen, las siguientes características:

Carácter local o global. Un método es local si el cambio de posición de un punto de control afecta sólo a una zona limitada de la superficie. Si el cambio de un punto modifica la forma de toda la superficie se dice que el método es global.

Interpolante: Un método es interpolante si la superficie generada pasa por los puntos de control. Es decir los puntos de control son puntos de paso.

Grado de continuidad: Atendiendo a la continuidad de la superficie. En determinados casos el grado de continuidad no será el mismo en todos los puntos de la superficie. En dichas casos se entenderá como grado de continuidad el menor de entre todos los puntos de la superficie (o curva).

Grado del polinomio: Grado de la función polinómica resultante.

En términos generales interesan los métodos locales con un grado de continuidad alto y un grado de polinomio bajo. El hecho de que un método sea local facilita la edición. El grado de continuidad y el grado del polinomio están relacionados, lo que hace que no sea posible conseguir simultáneamente ambas condiciones.

Nos centraremos, en primer lugar, en los métodos de **diseño de curvas**. De estos, los más utilizados son los métodos de diseño de curvas de Bézier, los Splines, los B-Splines [Böhm84] y los B-Splines racionales [Till83].

El método de Bézier utiliza como funciones de forma los polinomios de Bernstein. Genera una curva descrita utilizando n puntos de control con un polinomio de grado n-1.

Tanto los Splines como los B-Splines y los B-Splines racionales generan la curva con funciones polinómicas a trozos. Las características de continuidad de la curva dependen del grado del polinomio.

El cuadro siguiente resume las propiedades de los métodos anteriores, comparándolos con la interpolación lineal.

	Grado	Interpola	Carácter	Continuidad
Lineal	1	si	L_2	C^0
Bézier	n-1	no	G	C
Spline Local	3	si	L_4	C^1
Spline Global	3	si	G	C^2
B-Spline	k	no	L_{k+1}	C^{k-1}
B-Spline Rac.	k	no	L_k	C^{k-1}

Las **superficies** se pueden diseñar a partir de curvas que se usan como perfiles, o a partir de un conjunto de puntos de control. Esta última posibilidad se realiza generalizando los métodos de diseño de curvas a 3D.

Para construir una superficie a partir de perfiles, se parte de una o varias curvas definidas en el espacio, y se utiliza una ecuación que produzca la superficie en función de las curvas. Entre los métodos usados, destacamos los siguientes:

Superficie cilíndricas. La superficie se forma por la traslación de un segmento de recta sobre una curva cualquiera. O lo que es lo mismo, por el barrido por traslación de la curva. La ecuación de la superficie es:

$$S(u,v) = P(u) + v * L$$

donde S(u,v) es la ecuación de la superficie, P(u) es la curva y L es el vector definido por el segmento de recta.

Superficie de revolución. Se obtiene al girar una curva plana respecto a un eje. Si el giro se realiza respecto al eje z, la ecuación de la superficie es:

$$S(u,v) = (X(u) * \cos(v),\ Y(u) * \mathrm{sen}(v),\ z(u))$$

Superficie reglada. Son una generalización de las cilíndricas. La superficie está definida por la unión mediante segmentos de recta de dos curvas del espacio. Equivale a interpolar linealmente las dos curvas. Las dos curvas deben de tener el mismo rango del parámetro.

$$S(u,v) = (1-v) * P(u) + v * Q(u)$$

Superficie de unión. Dada una familia de curvas con la misma parametrización $S_i(u)$, se puede construir una superficie interpolando los puntos con el mismo valor de parámetro en todas las curvas:

$$S(u,v) = \Sigma_{i=1}^{n}\ S_i(u)\cdot F_i(v)$$

Para generar una superficie de unión partiendo de una familia de perfiles puede ser conveniente generar la familia completa a partir de un número reducido de curvas. Dos métodos usuales de generar la familia de perfiles son:

Revolución. Dada una curva P(u) y un eje, se obtienen perfiles a partir de la curva girándola respecto al eje ángulos Θ, $2\cdot\Theta$, etc.

Perfil dirigido por un eje. Dada una curva plana P(u) y una curva cualquiera Q(u) se pueden obtener perfiles colocando la curva P sobre la Q con una orientación fija respecto a ésta. El proceso será: trasladar P, calcular tangente de Q en el punto de contacto, girar P para que su orientación sea fija respecto a la tangente a Q. El proceso de generación ulterior de una superficies de unión a partir de perfiles dirigidos por un eje es semejante al de construcción de un sólido por barrido.

Para generar una superficie a partir de un conjunto de puntos de control, se pueden utilizar generalizaciones de los métodos de diseño de curvas. En este caso se usará una malla de puntos de control (x_{ij},y_{ij},z_{ij}). La superficie se calculará a trozos, o parches, en la forma:

$$X(u,v) = \Sigma_{i=1}^{n} x_{ij} \cdot F_i(u) \cdot F_j(v)$$

siendo F la función de forma utilizada.

El dominio de todos los métodos está determinado por las funciones de forma, que en la mayoría de ellos son funciones polinómicas. Tan sólo los B-Splines racionales poseen un dominio más amplio. Por otra parte, se puede compensar el hecho de que el dominio sea reducido aumentando el número de trozos utilizados en la construcción de la curva.

En el diseño de sólidos la situación es diferente, se desea poder representar un conjunto amplio de sólidos, y no existe un soporte matemático que lo permita. Los distintos métodos son por tanto incompatibles (total o parcialmente) entre sí. Los problemas de dominio y validez son cruciales, pasando a segundo término los problemas de facilidad de edición.

1.3.3 Especificación de sistemas gráficos.

La especificación es un paso decisivo en el desarrollo de una aplicación informática, independientemente que sea, o no, gráfica. L. Lamport la define informalmente como [Lamp89] :

> "Considero útil ver la especificación como un contrato entre el usuario de un sistema y su constructor. El contrato debe indicar al usuario todos los aspectos que debe de conocer para utilizar el sistema, y debe de indicar al constructor todos los detalles del sistema que debe de conocer para construirlo".

En definitiva, la especificación responde a la pregunta ¿que se va a construir?. Este planteamiento general fue expuesto en una forma más precisa por G. Carson [Cars83], quien enumeró las funciones concretas que debe de cumplir la especificación de un sistema gráfico:

- ser una descripción precisa y no ambigua del sistema.
- servir de referencia a los usuarios.
- permitir el planteamiento y resolución de preguntas precisas sobre el sistema.
- permitir la selección de los detalles de implementación.
- servir de referencia a los diseñadores.
- servir de base para la prueba y validación del sistema.
- conducir a un buen diseño.

Parece existir un acuerdo generalizado respecto a este modo de concebir la especificación y respecto a su importancia en el desarrollo de sistemas informáticos en general y de sistemas gráficos en particular.

La necesidad de huir de la ambigüedad, a la hora de describir un sistema, ha obligado a buscar técnicas formales de especificación que permitiesen más precisión que el lenguaje natural [Cars83]. J.M. Wing [Wing90] define los métodos formales como:

> "Un método de especificación es formal si posee un fundamento matemático. Este fundamento permite definir con precisión nociones como consistencia, completitud, especificación, implementación y corrección. Permite probar que una implementación es correcta, que una especificación es realizable, y probar propiedades del sistema sin necesidad de ejecutarlo para determinar su comportamiento".

La especificación de un sistema necesitará por tanto, para poder ser tratada formalmente, de la utilización de modelos matemáticos para representar el sistema a desarrollar.

Una de las principales ventajas de la utilización de métodos formales de especificación es la posibilidad de verificar formalmente que la implementación se corresponde con la especificación. En general, la validación de un sistema puede realizarse por dos métodos [Gnat83,Hübn86]. El primero está basado en la utilización de técnicas de prueba sobre el programa ejecutable. El segundo método se basa en la utilización de pruebas de corrección y de métodos formales para la construcción de los programas, y por tanto necesita de una especificación formal. A esta segunda alternativa se la suele conocer como verificación.

La verificación del sistema se puede realizar en dos modos [Gnat83]: a posteriori o a priori. La primera opción consiste en demostrar que el sistema, una vez desarrollado, es correcto según una especificación formal. El segundo método consiste en derivar la implementación como una serie de pasos de transformación a partir de la especificación, de tal modo que se garantice que dichos pasos preservan la corrección del sistema. Si la secuencia de derivaciones es correcta la implementación será correcta.

A lo largo de la última década se han propuesto diversos métodos de especificación formal. Estos métodos se pueden dividir en dos grandes grupos [Wing90,Duce88A]:

- **Especificación orientada al modelo**. Una especificación orientada al modelo define un modelo formal del sistema, en base a conceptos matemáticos bien establecidos. En este grupo cabe destacar el **Método de Viena (VDM)**, el lenguaje **Z**[(1)], las **redes de Petri** y el lenguaje **CSP** (estos dos últimos para sistemas concurrentes).

- **Especificación orientada a propiedades**. En ellas la descripción del sistema determina sus propiedades, pero no establece un modelo. En este grupo se destacan los métodos de **especificación algebraica**.

En cualquier caso, la especificación consta de un conjunto de módulos. Cada uno de estos módulos es en sí la especificación de una parte del sistema.

---

(1) El método Z puede ser usado igualmente en especificación orientada a propiedades.

La especificación de un módulo utilizando el método de Viena consta de una descripción del estado del módulo y de un conjunto de operaciones que lo modifican. El estado del módulo esta descrito por un conjunto de variables de estado. Cada operación se especifica dando precondiciones y postcondiciones, que son expresiones lógicas que ligan a las variables que determinan el estado del módulo. Las precondiciones deben de cumplirse para que la operación se pueda realizar. Las postcondiciones se cumplen necesariamente después de realizarse la operación.

En una especificación algebraica cada módulo (que se puede considerar como un tipo abstracto de datos) es un conjunto dotado de una cierta estructura algebraica, la descripción del módulo es una definición formal de dicha estructura algebraica. La especificación de cada módulo consta de un conjunto soporte, de la descripción de las operaciones definidas sobre dicho conjunto soporte y de un conjunto de axiomas que se deben de cumplir. La descripción de las operaciones indica simplemente su funcionalidad. Los axiomas son ecuaciones en las que intervienen las operaciones.

Entre los lenguajes de especificación algebraica cabe destacar los lenguajes de especificación **Clear**, **ASF**, **OBJ** y **CIP**.

Las especificaciones algebraicas pueden ser constructivas o no constructivas [Hore90]. En una especificación algebraica constructiva se imponen una serie de requisitos a la estructura de los axiomas, de tal modo que se puedan interpretar como reglas de construcción. La ventaja de la utilización de especificación constructiva es la posibilidad de realizar prototipado directo.

La **especificación formal de sistemas gráficos** ha cobrado un gran interés, que en una buena parte está motivado por el desarrollo de los estándares gráficos. Un estándar se puede considerar en sí como una especificación, de la que diversos fabricantes derivarán distintas implementaciones. En este contexto la no ambigüedad y la verificación son dos problemas cruciales.

En la mayor parte de las aplicaciones se puede tolerar una cierta ambigüedad en la especificación. La tolerancia de ambigüedad se puede interpretar en el sentido de que el aspecto al que afecta dicha ambigüedad no es relevante ni para el usuario ni para el diseñador. Este último 'resolverá' la ambigüedad al desarrollar el sistema, ya que el sistema, lógicamente, no es ambiguo. Téngase en cuenta que cualquier implementación de un sistema se puede considerar como una especificación, que establece un modelo real del mismo.

Cuando lo que se desarrolla es un standard gráfico, la existencia de ambigüedad en la especificación puede conducir a que dos implementaciones distintas no sean equivalentes, por haberse resuelto la ambigüedad de modo diferente en cada una de ellas.

Se han propuesto distintos métodos formales para la especificación de sistemas gráficos. Guttag propone un método de especificación en el que se separa la definición de la estructura funcional del sistema de su interfase [Cars83]. Cada módulo consta de dos partes, en una se define el módulo, usando especificación algebraica, y en la otra se da el conjunto de operaciones visibles exteriormente (la interfase del módulo), especificadas mediante precondiciones y postcondiciones. Estos trabajos fueron el origen del lenguaje de especificación Larch.

Posteriormente, Mallgren utilizó especificación algebraica para especificar sistemas gráficos [Mall82]. Mallgren define formalmente cuatro elementos básicos (región, dibujo, transformación y estructura jerárquica de dibujo) que se usarán como base para la realización de cualquier especificación.

Una región es un conjunto de puntos del plano o del espacio. La región puede ser un continuo (ej. un rectángulo en el plano) o un conjunto discreto de puntos aislados (para modelar dibujos raster). Sobre las regiones están definidas las operaciones booleanas: unión, intersección, diferencia, así como funciones para determinar inclusión y si una región está vacía.

Mallgren define un dibujo como una función parcial que asigna un color a los puntos del plano (o del espacio). Con los dibujos realiza operaciones de combinación (suma). La suma se define punto a punto, dando como resultado para cada punto la suma de los colores de los dos dibujos.

Más recientemente, diversos autores han utilizado métodos formales para especificar sistemas gráficos, fundamentalmente estándares. Entre estos cabe destacar la utilización del método de Viena [Duce88A], y la especificación algebraica [Duce89] [Gnatz83].

La **completitud** y la **consistencia** son dos propiedades esenciales en una especificación algebraica. Una especificación es completa si el resultado de realizar cualquier operación en cualquier estado del módulo y para cualquier valor de los argumentos está determinado por los axiomas. La especificación es consistente si dicho resultado es único. En general no se puede demostrar que una especificación sea completa y consistente.

No obstante se pueden construir los axiomas siguiendo una sistemática que garantice ambas propiedades. W. Mallgren propone un método heurístico para el diseño de los axiomas, basado en trabajos anteriores de Guttag [Mall82]. Dicho método parte de la descripción de las funciones, que clasifica en tres grupos:

· Funciones de **Consulta** (inquiry), que devuelven información de tipos definidos fuera del módulo.

· **Generadores**, que devuelven información del tipo definido por el módulo (es decir elementos del conjunto soporte del álgebra).

· Funciones **Básicas**. Las funciones básicas son un subconjunto de los generadores, que permiten generar cualquier valor del tipo definido por el módulo.

En el conjunto de axiomas aparecerán los necesarios para describir el resultado de aplicar las funciones de consulta después de las funciones básicas y los que permitan expresar el resto de los generadores en función de las funciones básicas.

De esta forma se asegura que la especificación es completa y consistente, ya que describe el resultado de aplicar las funciones de consulta en cualquier estado del módulo. Esta comprobación de la consistencia y completitud de la especificación se puede realizar de forma automática.

En determinados casos puede ser interesante incluir axiomas auxiliares. Esto se hace normalmente para especificar igualdad de elementos, que de otro modo podrían ser distintos. Si se incluyen dichos axiomas se deberá, en principio, comprobar la consistencia de la especificación.

Un problema fundamental en la especificación de un sistema gráfico es poder inferir propiedades de la imagen generada. La especificación garantiza un determinado comportamiento del módulo en cuanto a sus funciones, basado en establecer condiciones sobre los resultados de las funciones, y indirectamente en el estado del módulo. La imagen no forma parte del estado del módulo, es una interfase del módulo. La mayor parte de los métodos que se han propuesto de especificación formal de sistemas gráficos no han abordado este problema, dejándolo como un aspecto de conexión con el dispositivo, o lo han planteado fijando la estructura del dispositivo y asumiendo una determinada relación del estado de este con el del módulo.

Podemos pues concluir que existen métodos formales para especificar sistemas gráficos, y que de entre ellos, la especificación algebraica parece un método aceptable y útil en la práctica. No obstante todos los métodos expuestos se centran en la especificación del sistema como conjunto no gráfico de información, lo que en definitiva impide el que se pueda inferir ninguna propiedad de las imágenes generadas.

1.3.4 Formalización de interacción.

La especificación formal de sistemas interactivos plantea problemas propios, que no pueden ser resueltos con los métodos descritos en el apartado anterior. La interacción ocurre en la interfase entre dos sistemas, uno de los cuales es el sistema software que se está especificando. El otro sistema puede ser software, hardware o el usuario.

La raíz de la dificultad para especificar las interfases está en formalizar las restricciones de sincronización que se producen en la comunicación. En este sentido, un sistema interactivo será un caso particular de sistema concurrente, en el que uno de los procesos es el usuario. El resto de las componentes de la interfase (semántica, sintaxis, etc.) se pueden expresar con métodos convencionales.

Con este planteamiento cabe pensar en la utilización de métodos de especificación de sistemas concurrentes, para la especificación de sistemas gráficos interactivos. Entre las propuestas realizadas en este sentido destacamos la utilización de unidades de E/S, CSP y Tipos de datos compartidos.

P.J.W. ten Hagen y R.van Liere proponen la utilización de **unidades de entrada/salida** [Hage88]. Cada unidad encapsula propiedades del dispositivo y su realimentación. De esta forma se pretende que la concordancia entre cada información de entrada e información de salida quede garantizada por una unidad.

El **CSP** es un lenguaje de especificación de sistemas concurrentes [Hoar78]. Una especificación en CSP constituye un modelo del sistema. El método se basa en una

formalización del concepto de proceso y en operaciones de paso de mensajes entre procesos.

Un proceso está constituido por una secuencia de sucesos. La especificación de un proceso se realiza en la forma:

A = (read -> B)

que indica que el proceso A realiza una lectura y posteriormente se comporta como el proceso B. La ejecución concurrente de varios procesos se expresa por

A | | B

Las operaciones de paso de mensajes se realizan a través de canales. Al indicar una operación de este tipo se hace referencia al canal, al mensaje y al tipo de operación (recepción '?', o envío '!'). Así, las operaciones siguientes

c!v -> c?v

indican del envío de un mensaje a través del canal c y su posterior recepción.

El CSP ha sido utilizado por D.A. Duce para especificar el subsistema de entrada en el GKS [Duce89]. La ventaja de la utilización del CSP es que dicho lenguaje da un soporte formal a la especificación. El principal inconveniente es que la especificación es en sí un modelo, y por tanto no es compatible con especificaciones algebraicas.

Tipos de datos compartidos.

W. Mallgren propuso la utilización de tipos de datos compartidos, para facilitar la integración de la especificación de la interacción con una especificación algebraica del resto del sistema [Mall82].

Mallgren realiza la especificación de un tipo de datos compartido algebraicamente, para ello hace uso de los conceptos de estado e historia, incluyendo axiomas que relacionen las operaciones con el estado del sistema.

Para ello se asocia a cada sistema un estado. Inicialmente el sistema se encuentra en un estado especial, que se denota por $init. Cada llamada o terminación de una operación puede cambiar el estado del sistema. Esto equivale, formalmente, a asociar a cada operación

Op_i t_1 x ... x t_2 -> V

dos funciones de sucesos (o de cambio de estado):

$Op_i$$call	estado x t_1 x ... x t_2 -> estado
$Op_i$$return	estado x t_1 x ... x t_2 -> estado

El estado del sistema estará, por tanto, determinado por la secuencia de sucesos que le han ocurrido.

Para especificar las restricciones funcionales y de sincronización se utilizan dos funciones (que denomina características) asociadas a cada operación:

Op_i:value estado x t_1 x ... x t_2 -> V

Op_i:wait estado x t_1 x ... x t_2 -> boolean

La función value da el valor devuelto por la operación. La función wait se utiliza para especificar la sincronización. La operación Op_i puede terminar sólo cuando Op_i:wait es falso. La sincronización se especifica mediante axiomas. A continuación se muestra la especificación de un tampón finito, de una posición [Mall82].

```
put    item    ->
get            -> item

funciones características:

put: wait     state x item  -> boolean
put: value    state x item  -> null
get: wait     state         -> boolean
get: value    state         -> item

funciones de sucesos:

$init                       -> state
put$call      state x item  -> state
put$return    state x item  -> state
get$call      state         -> state
get$return    state         -> state
axiomas:

put
     wait($init,x) = false
     wait(put$return(S,x1),x2) = true
     wait(get$return(S),x) = false

get
     wait($init) = true
     wait(put$return(S,x)) = false
     wait(get$return(S)) = true
     value($init) = undefined
     value(put$return(S,x)) = x
```

En esta especificación no se encuentran todos los axiomas necesarios para que la especificación sea completa. W. Mallgren reduce el número de axiomas que deben aparecer en la especificación dando una serie de reglas para construir los axiomas ausentes (que denomina axiomas implícitos). La definición de las funciones de sucesos podría también omitirse, pues se puede deducir de la de las operaciones. Aún así se observa que la especificación es compleja.

W. Mallgren define la corrección de una implementación en base a dos criterios:

1- El sistema toma sólo estados correctos.
2- El valor devuelto por las operaciones es correcto.

En base a estos dos criterios realiza un diseño estándar para un tipo de datos compartido, utilizando como mecanismo de sincronización regiones críticas condicionales [Andr83]. La estructura propuesta para una operación Op, definida por

Op$call	state x t_1 x ... x t_2 -> state
Op$return	state x t_1 x ... x t_2 -> state
Op:value	state x t_1 x ... x t_2 -> V
Op:wait	state x t_1 x ... x t_2 -> boolean

es:

```
var S: state;                          { Global declaration for}
initially S <- $init      { all operations of the type}
...
function Op(a1:t1,a2:t2,..,an:tn): V;
begin
  with S when true do
        S <- Op$call(S,a1,..,an);
  with S when ¬Op:wait(S,a1,..,an) do
        begin
          Op <- Op:value(S,a1,..,an);
          S <- Op$return(S,a1,..,an)
        end
end
```

W. Mallgren muestra que la implementación es correcta, de acuerdo con la especificación. No obstante no da ningún método para comprobar que la especificación, en sí, es correcta como sistema concurrente, es decir que esta libre de problemas de bloqueos.

La propuesta de Mallgren presenta la ventaja de ser más fácil de integrar en una especificación algebraica. No obstante, no existen métodos, tal como se ha dicho, que permitan comprobar la corrección de la especificación. Por otra parte el conjunto de axiomas necesario para especificar un sistema real, aún cuando se asuman algunos, puede ser demasiado grande.

1.3.5 Formalización de visualización.

La visualización es uno de los aspectos más significativos de la informática gráfica. En lo referente a su formalización es necesario destacar el trabajo de Fiume.

E. L. Fiume se centra en la visualización de imágenes en dispositivos de puntos (raster) [Fium89]. En ellos, defiende la necesidad de establecer un formalismo único para dichos sistemas.

El formalismo arranca de la definición de escenas. Las escenas se define como una composición de objetos. En la definición de los objetos se contempla sus propiedades volumétricas y visuales. Formalmente un objeto se define como:

"Un objeto estático es un par $(\mathbf{Z_0},\mathbf{I_0})$, en el que $\mathbf{Z_0} \subseteq \mathbf{R}^3$ y $\mathbf{I_0}$: $\mathbf{Z_0}$ -> **C**, siendo **C** un espacio de color definido como un subconjunto de $\mathbf{R}^c$ con $c \geq 1$"

Para componer las escenas, a partir de objetos, se introducen operaciones de unión e intersección. Ambas funciones realizan la unión o intersección de $\mathbf{Z_0}$ y asignan un color arbitrario al objeto resultante.

Los conceptos anteriores se utilizan para formalizar el proceso de visualización en un sistema raster. Concretamente se abordan los siguientes problemas:

· Visibilidad: Define formalmente el concepto de visibilidad como la determinación de la parte visible de cada objeto en una escena. Utiliza este formalismo para calcular límites de complejidad a problemas simples de visibilidad.

· Realismo: Define formalmente el concepto de imagen. Establece un formalismo de generación de imagen basado en teoría de la medida y análisis real. Utiliza el formalismo para describir algunas técnicas de visualización realista.

· Gráficos de imagen[1]: Estudia el problema de discretización de líneas y de transformación de imágenes de puntos, analizando la equivalencia entre transformaciones de imágenes y visualización de escenarios transformados.

· Modelos de iluminación: Formaliza modelos de iluminación local y global. Estudiando la complejidad del último.

Su trabajo se centra, por tanto, en la visualización, obteniendo buenos resultados en estimación de complejidad de problemas de eliminación de partes ocultas y rendering.

Por otra parte, hay que destacar que E. Fiume no dota a su espacio de objetos de una estructura algebraica apropiada, haciendo difícil su utilización tanto como herramienta de especificación de objetos, como de soporte matemático para el modelado. Además las

---

[1] Bit-Mapped Graphics en Inglés.

operaciones definidas no cumplen las propiedades usuales de las operaciones con volúmenes, dado que se asigna un color arbitrario a las zonas en que se solapan dos objetos. Así, A U (A ∩ B) es en general distinto de A.

1.3.6 Diseño de sistemas gráficos.

La Informática Gráfica es un campo natural de aplicación de las técnicas de programación dirigida a objetos (PDO), estando esta última prácticamente aceptada como herramienta de diseño e implementación de aplicaciones gráficas.

Inicialmente la mayoría de los esfuerzos de aplicación de la PDO a la Informática Gráfica se centraron en el desarrollo de modelos de interacción Hombre-Máquina a través de Interfases Gráficas de Usuario (GUIs) [Blac89,Otte90,Wiss88]. No obstante, se ha comenzado a utilizar con éxito en la resolución de problemas de visualización [Glas89,Ureñ91,Fium88], habiéndose planteado incluso la necesidad de desarrollar un nuevo estándar orientado a objetos [Wiis90].

La evolución del desarrollo de software y de los lenguajes de programación se han producido de modo paralelo desde el nacimiento de la Informática. La Ingeniería del Software se ha consolidado durante las décadas 70 y 80 y ha influido de modo decisivo en las metas del diseño de los lenguajes de programación. Por otra parte, los conceptos introducidos por los lenguajes de programación han sido adoptados en la especificación y el diseño de software.

El paradigma de la Programación Dirigida a Objetos (PDO), nace con Simula en 1968, que introduce los conceptos de clase y herencia y se consolida con Smalltalk [Gold89], a principios de la década de los 80. Es este lenguaje-entorno el que introduce los conceptos fundamentales que actualmente se manejan en la PDO y que, junto con Objective-C [Cox_86] constituye la corriente 'más pura' de este paradigma. Con posterioridad han surgido gran número de lenguajes dirigidos a objetos que presentan matices importantes en la concepción de la programación: CLOS, que introduce la orientación a objetos en la programación funcional; Eiffel, C++ y Turbo Pascal 5.5 que, en la más pura tradición del Pascal y ADA, introducen la comprobación fuerte de tipos; ACTORS, FLAVORS y un largo etcétera, en el que algunos autores incluyen lenguajes de programación que no incorporan todos los conceptos de la PDO, pero sí importantes mecanismos de abstracción de datos (ADA, CLU) [Lisk86].

El Desarrollo Dirigido a Objetos (DDO), entendido como el proceso global de obtención de software bajo este enfoque, posee una actualidad palpitante y, como en el caso del desarrollo dirigido a datos o a funciones, ha tenido una evolución iniciada en los lenguajes de programación y seguida por métodos de diseño y especificación. Los primeros métodos de diseño datan de 1986 [Booc87, Booc91] y los de especificación de requerimientos del 89-90 [Bail89, Coad90].

Nos encontramos, por tanto, en plena evolución del DDO, cuando los conceptos de PDO se encuentran relativamente bien establecidos. En este contexto, los sistemas gráficos constituyen, posiblemente por la cercanía entre el espacio de los problemas y de las soluciones, una de las primeras aplicaciones reales del diseño e implementación dirigidos a objetos.

1.4 CONCLUSIONES.

En el desarrollo de aplicaciones gráficas es necesario disponer de métodos que permitan la representación, abstracta y computacional, del sistema real. Un elemento fundamental de dicha representación es la utilización de un lenguaje de representación, que en el caso de la representación abstracta es deseable que sea un lenguaje matemático, ya que esto facilitará un tratamiento formal del problema.

En el ámbito de la Informática Gráfica no se dispone un lenguaje de representación suficientemente general. En distintos campos concretos se utilizan representaciones, más o menos formales, pero en cualquier caso particulares.

El presente trabajo se propone establecer un marco teórico útil en la realización de sistemas de representación aplicables a sistemas gráficos, mostrando su utilización en el desarrollo de técnicas y metodologías de especificación de aplicaciones gráficas. Para ello se debe:

- Establecer un formalismo teórico que sirva de soporte simultaneamnte a conceptos de modelado, visualización e interacción.

- Desarrollar un método homogéneo para especificar tanto aspectos secuenciales, como concurrentes de sistemas gráficos, y que permita establecer de forma precisa la corrección de la especificación.

- Desarrollar una metodología de especificación que permita realizar fácilmente un diseño Orientado a Objetos.

La especificación algebraica, la teoría de visualización de Fiume y el modelado son tres puntos de referencia obligados, en el desarrollo del trabajo. Por tanto la teoría se desarrollará con el propósito de conectar adecuadamente con ellos.

2. CONCEPTO DE OBJETO GRAFICO.

2. CONCEPTO DE OBJETO GRAFICO.

Este capítulo presenta un soporte teórico útil para la manipulación de información gráfica. El elemento fundamental de la teoría son los modelos abstractos de sistemas gráficos, que denominaremos, en adelante, objetos gráficos. Un objeto gráfico es pues un modelo abstracto obtenido a partir de un sistema gráfico cualquiera.

Este capítulo se centra en la definición algebraica de los objetos gráficos. En el capítulo siguiente se definirán algunas operaciones que se pueden realizar sobre los objetos gráficos, permitiendo la construcción de unos objetos gráficos a partir de otros.

Informalmente podemos definir un objeto gráfico como una abstracción de un ente que posee un fuerte contenido de información geométrica (o visual). Así, un objeto gráfico podrá ser una abstracción de un objeto físico (p.e. una silla) o de otra abstracción (p.e. una función matemática). Lo realmente importante es que el objeto gráfico contenga información visual y/o geométrica del ente al que representa. Por tanto, un objeto gráfico no será necesariamente una abstracción de un objeto físico, en consecuencia es lógico admitir que para algunos objetos gráficos no existirá un objeto físico equivalente (en el sentido de que aquel pueda ser abstracción del objeto físico). Esta situación permite, entre otras cosas, que el conjunto de objetos gráficos sea cerrado respecto a las operaciones de modelado que se definirán. Así, por ejemplo, se podrá considerar el objeto gráfico resultante de substraer un determinado objeto gráfico de otro. Aunque, en algunos casos, no exista un objeto físico equivalente al objeto gráfico resultante, podemos manejarlo dicho resultado conceptualmente, y podremos encontrar un objeto físico equivalente, por ejemplo, a la suma de dicho resultado con otros objetos gráficos. No obstante, se dará una interpretación de cada objeto gráfico que permita su visualización. Obviamente se garantiza, que de todos los objetos físicos, se puedan derivar objetos gráficos.

El carácter geométrico y visual de los objetos gráficos se traduce en que contengan información relativa a su aspecto visible y a su ubicación y extensión en el espacio. El primer apartado de este capítulo se dedica a definir de forma precisa los conceptos necesarios para manejar estas nociones.

Al considerar la apariencia visible como parte de la información de los objetos gráficos, necesitaremos poder operar con, por ejemplo, el color de dos objetos que se van a componer, para obtener el color de la composición. Esta filosofía es diferente de la usada habitualmente, consistente en separar la información de aspecto de la información de modelado, y que conduce a que primero se compongan los objetos y posteriormente se les asigne color. Creemos que es más lógico utilizar una sistemática de trabajo que integre la información de modelado y visualización. Dicha sistemática puede parecer, en principio, menos natural que la habitual. Consideramos que esto es cuestión simplemente de uso, ya que en cualquier caso lo importante es que los resultados sean predecibles, que sea posible deducir

la secuencia de operaciones necesarias para generar un determinado objeto, y que los resultados obtenidos habitualmente se puedan obtener con la nueva sistemática; y estas cosas son posibles con la teoría propuesta.

En adelante se utilizará el término objeto para hacer referencia a los objetos gráficos, salvo que se preste a confusión con los objetos físicos, o con estructuras de programación, en cuyo caso se aclarará explícitamente

2.1 CONCEPTOS BASICOS.

Como paso previo se introducen algunos conceptos, que se usarán más adelante como soporte de la definición formal de objeto gráfico. Concretamente se introduce el concepto de aspecto, que hace referencia a la apariencia visible de los objetos y los conceptos de presencia, volumen y transformación geométrica que hacen referencia a la ubicación y extensión de los objetos.

2.1.1 Aspecto.

La forma en que vemos los cuerpos visibles depende de una serie de atributos, como color, opacidad, textura, etc. Para poder trabajar con estos atributos, se define una estructura algebraica que denominaremos 'dominio de aspecto'.

La construcción de objetos gráficos por composición implica el cálculo del aspecto del objeto resultante a partir de los valores de aspecto de los objetos que intervengan en la composición. Por este motivo es necesario definir operaciones de aspecto. Supongamos que se ha definido una operación de suma de objetos gráficos, para calcular la suma de objetos $O_1 + O_2$, se debe determinar en cada punto el valor de aspecto del objeto resultante, parece natural determinar dicho valor como una combinación de los aspectos de ambos objetos:

$$\text{Aspecto}(O_1 + O_2) = \text{Aspecto}(O_1) + \text{Aspecto}(O_2)$$

para lo cual será necesario definir la operación suma sobre los aspectos. Estas operaciones se podrían definir siguiendo dos filosofías:

1· Definiendo explícitamente las operaciones de aspecto a utilizar con cada operación de combinación de objetos. Esto implicaría indirectamente la determinación del conjunto de valores de aspecto. Esto equivaldría, por ejemplo, a decir que la suma se realiza componente a componente, lo cual condiciona no solo el comportamiento algebraico, si no también la forma de los valores de aspecto.

2· Indicando las propiedades que deben de cumplir las operaciones sobre el conjunto de valores de aspecto, de forma que se garanticen las propiedades necesarias en la combinación de objetos. Así, por ejemplo, se podría decir que debe existir una operación de suma que dote al conjunto de valores de aspecto de estructura de grupo abeliano.

Hemos optado por la segunda opción, al entender que es más general, ya que permitirá la definición, en cada caso, del dominio más apropiado, sin que se deba de modificar la teoría.

Definición 2.1: Un dominio de aspecto, **A**, es un conjunto dotado de las siguientes operaciones y propiedades:

1. Está definida una operación interna, +, que denominaremos suma, respecto a la cual **A** es un grupo abeliano.

2. Existe una operación externa, *, que denominaremos producto por escalar, con la cual (**A**, +, *) es un espacio vectorial.

3. Existen dos operaciones binarias internas, $\cup$ e $\cap$, que denominaremos unión e intersección respectivamente, y una operación unaria, $\sim$, que denominaremos complementación, respecto a las cuales **A** es un álgebra de Boole. El elemento vacio de dicha álgebra coincide con el elemento neutro para la estructura de espacio vectorial.

4. Existe una operación interna, $\times$, que denominaremos producto, que satisface las siguientes propiedades:

- Es asociativa
- Es conmutativa
- Posee elemento unidad.
- El elemento neutro del espacio vectorial es elemento absorvente. □

Comentarios:

Cada valor de aspecto representa una configuración de los atributos de visualización de un objeto. Las operaciones incluidas en la definición permiten calcular el aspecto de un objeto que se ha obtenido por composición de otros.

La interpretación de los resultados de cada operación dependerá, lógicamente, de la definición que en cada caso se realice de la operación. En cualquier caso, consideramos que lo realmente importante no es que el color obtenido por suma de dos colores 'coincida' con el obtenido por mezcla de dos pinturas o por superposición de dos focos de luz, si no el que dicho color sea perfectamente predecible, se conozca el comportamiento matemático de la operación de composición, y se puedan generar efectos de combinación concretos, tales como los citados anteriormente.

*E. Fiume da la siguiente definición de espacio de color [Fium89]: 'Un espacio de color **C** es un subconjunto de $\mathbb{R}^c$, con $c \geq 1$'. Desde nuestro punto de vista esta definición tiene el inconveniente de ser más restrictiva que la nuestra, al fijar el conjunto soporte. Además, en ella, no se especifica ninguna estructura algebraica para dicho espacio, lo que impedirá que el conjunto de objetos de Fiume dotado de las operaciones de combinación posea ninguna estructura algebraica (concretamente no es un álgebra de Boole).*

En principio podría parecer que las operaciones y estructuras consideradas restringen bastante los conjuntos que pueden ser dominios de color, pero si se estudian con cierto detalle se verá que las propiedades exigidas son bastante habituales. Por otra parte, debe tenerse en cuenta que la operación $\times$ puede coincidir la intersección.

Manteniendo la notación habitual, denotaremos con el símbolo 0 al elemento neutro del dominio de presencia respecto a la suma, con 1 al elemento unidad respecto a $\times$, con ϕ y ∞ a los elementos vacío (que coincide con 0) y universal del algebra de boole.

Tal como se ha expuesto anteriormente, para cada aplicación concreta se definirá un dominio de aspecto específico que contemple la información visual necesaria. Los dos ejemplos siguientes muestran dos definiciones de dominios de aspecto. En el primero los valores de aspecto son colores RGB, en el segundo son niveles de grises.

Ejemplo 2.1: Un modelo de visualización simple se puede obtener considerando como única característica del objeto su color, prescindiendo de otras características del material. El color se podría representar, por ejemplo en modelo RGB. En dicho modelo los colores se representan por una terna (r,g,b) en la que cada componente representa la intensidad de un color primario. Se puede definir un dominio de aspecto, $\mathbf{C_{rgb}} = (\mathbf{A}, +, *, \cup, \cap, \sim)$, para este modelo del siguiente modo:

$\mathbf{A}$ es $\{0,1,..,2^p-1\}^3 \subset \mathbf{Z}^3$ donde $p \in \mathbf{N}$

$+$ es la suma usual de ternas:

$$C_1+C_2=(r_1+r_2,g_1+g_2,b_1+b_2) \quad C_1,C_2 \in \mathbf{A}$$

realizándose la suma de componentes sobre el espacio de congruencias módulo 2^p. Es decir:

$$a + b = (a + b) \bmod (2^p) \quad a,b \in \{0,1,..,2^p-1\}$$

$*$ es el producto sobre el espacio de congruencias módulo 2^p:

$$k*C = (k*r \bmod 2^p, k*g \bmod 2^p, k*b \bmod 2^p) \quad C \in \mathbf{A}, k \in \{0,..,q-1\} \subset \mathbf{Z}$$

con q primo[1]

Las operaciones booleanas se definen a nivel de bit, en base a la representación en complemento a dos de cada una de las componentes de la terna.

Sea $C \epsilon \mathbf{A}$, $C = (x,y,z)$ con $x,y,z \in \{0,1,..,2^p-1\}$

la representación binaria de c en complemento a dos será:

$$C = x^{bb}_0 x^b_1 .. x^b_{p-1}, y^{bb}_0 y^b_1 .. y^b_{p-1}, z^{bb}_0 z^b_1 .. z^b_{p-1}$$

donde $x^b_i, y^b_i, z^b_i \in \{0,1\}$

Entonces, las operaciones booleanas se definen del siguiente modo:

---

(1) El producto se debe definir sobre un cuerpo externo.

$\sim$ se define como el resultado de la complementación de la representación binaria bit a bit:

$$\sim C = \sim x^{bb}_0 \sim x^{b}_1 .. \sim x^{b}_{p-1}, \sim y^{bb}_0 \sim y^{b}_1 .. \sim y^{b}_{p-1}, \sim z^{bb}_0 .. \sim z^{b}_{p-1}$$

$\cup$ se define como la unión bit a bit, sean $C,D \in \mathbf{A}$ con $C = (x,y,z)$, $D = (u,v,w)$

$$C \cup D = x^{bb}_0 \cup u^{bb}_0 .. x^{b}_{p-1} \cup u^{b}_{p-1}, y^{bb}_0 \cup v^{bb}_0 .. y^{b}_{p-1} \cup v^{b}_{p-1},$$

$$z^{bb}_0 \cup w^{b}_0 .. z^{b}_{p-1} \cup w^{b}_{p-1}$$

$\cap$ se define como la intersección bit a bit:

$$C \cap D = x^{b}_0 \cap u^{b}_0 .. x^{b}_{p-1} \cap u^{b}_{p-1}, y^{b}_0 \cap v^{b}_0 .. y^{b}_{p-1} \cap v^{b}_{p-1},$$

$$z^{b}_0 \cap w^{b}_0 .. z^{b}_{p-1} \cap w^{b}_{p-1}$$

El producto de colores, $\times$, se define como el producto componente a componente en el espacio de congruencias 2^p

$$C_1 \times C_2 = (r_1 \times r_2 \bmod 2^p, g_1 \times g_2 \bmod 2^p, b_1 \times b_2 \bmod 2^p) \quad C_1, C_2 \in \mathbf{A}$$

Es inmediato demostrar que $\mathbf{C_{rgb}}$ satisface la estructura de dominio de aspecto antes definida. En él la suma tendrá el significado de añadir las intensidades de las componentes. El producto equivale a aumentar la intensidad del color. El complemento de un color es el complementario respecto al blanco para la representación RGB usada.

En este dominio la suma de dos colores puede producir un color resultante con menor intensidad, debido a que las componentes del color están definidas sobre un espacio de congruencias. En general esto no supondrá un problema especial. En los casos en que se desee conseguir que la suma de dos colores siempre se realice con un aumento de intensidad se podrá construir un dominio de aspecto distinto, o aplicar filtros o transformaciones de color al objeto resultante.

Ejemplo 2.2: Si consideramos como información de aspecto, en lugar del color RGB, un nivel de intensidad de gris, podemos obtener dominios de aspecto más simples. Un ejemplo de dominio de aspecto permitiendo tres niveles de gris es el dado por $\mathbf{C_{m3}} = (\mathbf{A_{m3}}, +, *, \cup, \cap, \sim)$, siendo:

$\mathbf{A_{m3}}$ es $\{0,1,..,3\} \subset \mathbf{Z}$

$+$ la suma usual sobre el espacio de congruencias módulo 4:

$$C_1 + C_2 = (C_1 + C_2) \bmod 4 \qquad C_1, C_2 \in \mathbf{A_{m3}}$$

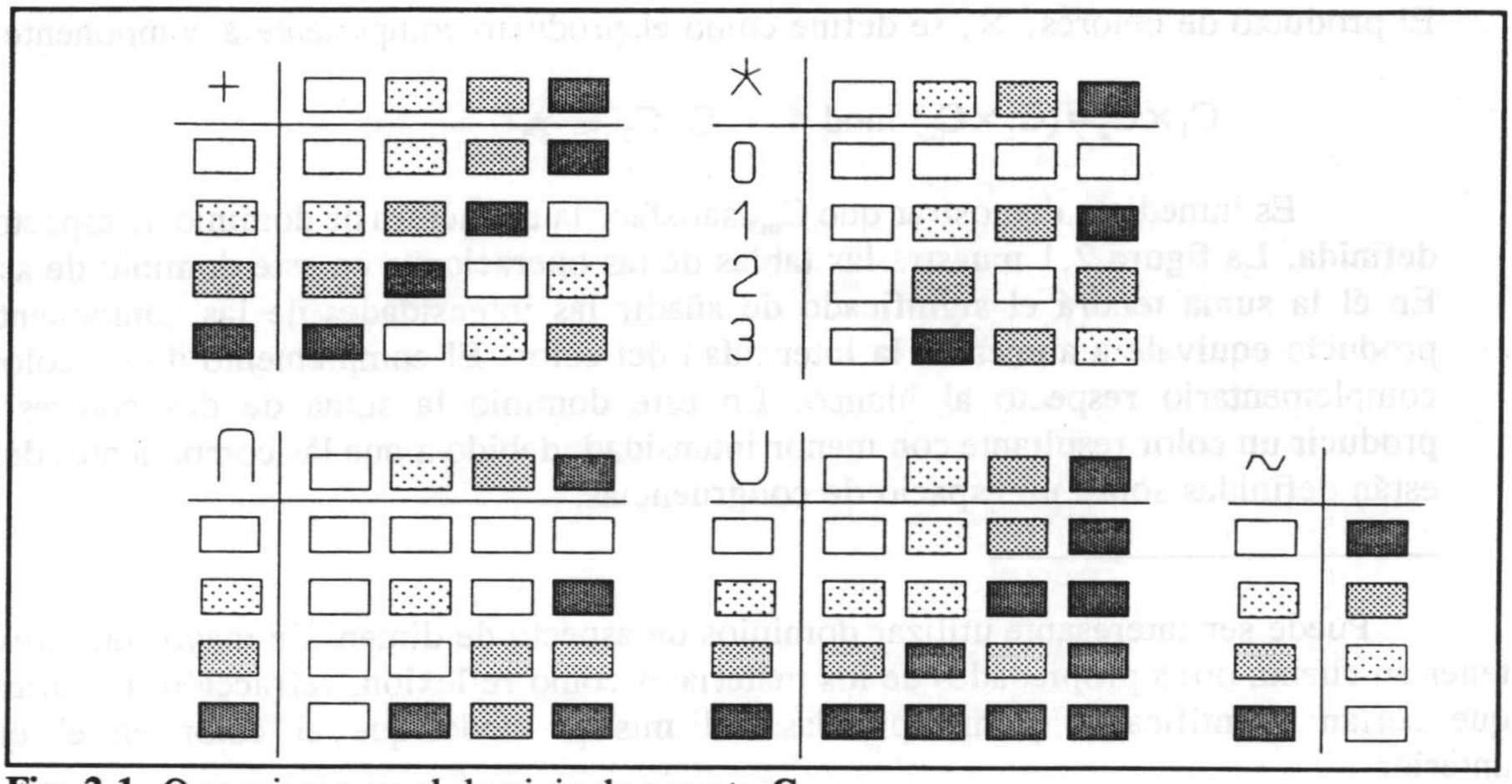


Fig. 2.1. Operaciones en el dominio de aspecto C_{m3}.

* el producto modulo 4:

$$k*C = (k*C) \bmod 4 \qquad C \in \mathbf{A},\ k \in \{0,..,q-1\} \subset \mathbf{Z}$$

con q primo[1]

Las operaciones booleanas se definen a nivel de bit, en base a la representación en complemento a dos

Sea $C \epsilon \mathbf{A}$, con representación binaria $c^{bb}_0 c^{b}_1 .. c^{b}_{p-1}, c^{bb}_0 c^{b}_1 .. c^{b}_{p-1}, c^{bb}_0 C^{b}_1 .. c^{b}_{p-1}$

Entonces, las operaciones booleanas se definen del siguiente modo:

$\sim$ se define como el resultado de la complementación de la representación binaria bit a bit:

$$\sim C = \sim c^{bb}_0 \sim c^{b}_1 .. \sim c^{b}_{p-1}$$

$\cup$ se define como la unión bit a bit, sean $C,D \in \mathbf{A}$

$$C \cup D = c^{bb}_0 \cup d^{bb}_0 .. c^{b}_{p-1} \cup d^{b}_{p-1}$$

$\cap$ se define como la intersección bit a bit:

$$C \cap D = c^{b}_0 \cap d^{b}_0 .. c^{b}_{p-1} \cap d^{b}_{p-1}$$

---

[1] El producto se debe definir sobre un cuerpo externo.

El producto de colores, $\times$, se define como el producto componente a componente

$$C_1 \times C_2 = (C_1 \times C_2) \bmod 4 \qquad C_1, C_2 \in \mathbf{A}$$

Es inmediato demostrar que $\mathbf{C_{m3}}$ satisface la estructura de dominio de aspecto antes definida. La figura 2.1 muestra las tablas de las operaciones en este dominio de aspecto. En él la suma tendrá el significado de añadir las intensidades de las componentes. El producto equivale a aumentar la intensidad del color. El complemento de un color es el complementario respecto al blanco. En este dominio la suma de dos colores puede producir un color resultante con menor intensidad, debido a que las componentes del color están definidas sobre un espacio de congruencias.

Puede ser interesante utilizar dominios de aspecto de dimensión mayor que tres, para tener en cuenta otras propiedades de los materiales como reflexión, refracción, texturas, etc., que serían cuantificadas y manipuladas del mismo modo que el color en el ejemplo anterior.

2.1.2 Volumen.

Una característica fundamental de cualquier objeto gráfico es su extensión en el espacio. A dicha extensión le denominaremos volumen del objeto. Usaremos este término con independencia de la dimensión del espacio.

Definición 2.2: Un volumen V es un subconjunto de $\mathbf{R^n}$, sobre el que consideramos definidas las operaciones habituales de suma y producto de elementos de $\mathbf{R^n}$. □

Es necesario distinguir entre el concepto de volumen como conjunto de puntos, finito o infinito, al que nos referimos en la definición anterior, y la medida, o medidas, de dicho volumen, a las que usualmente se denomina también volúmenes. En el primer caso hacemos referencia a un subconjunto del espacio, que puede tener cualquier dimensión, mientras que en el segundo caso el 'Volumen' es un número. Para evitar confusiones usaremos, en adelante, el término volumen para la acepción dada en la definición anterior, refiriendonos a su medida como 'medida del volumen' para evitar confusiones.

Es importante observar que podemos definir medidas tales que tomen valor cero sobre volúmenes no vacíos. Esto ocurre, por ejemplo, con la medida usual de volumen en $\mathbf{R^3}$ cuando el volumen es un sólo punto.

2.1.3 Transformaciones geométricas.

Es usual, en aplicaciones gráficas, que se cambie la posición de los elementos gráficos. A una función matemática que cambia la posición u orientación de los elementos gráficos se la conoce como una transformación geométrica.

El concepto de transformación geométrica suele introducirse en los textos de Informática Gráfica de un modo informal, a partir de ejemplos de transformaciones usuales, como giros, traslaciones, escalados, deslizamientos etc. [Newm81], [Salm87]. La implementación de algoritmos que realicen transformaciones geométricas requiere de la utilización de un formalismo matemático unificado y compacto. Por este motivo, habitualmente, se utiliza notación matricial para representar las transformaciones geométricas, lo que, matemáticamente, equivale a representar las transformaciones mediante aplicaciones lineales. Así, una transformación en dos dimensiones es:

$$(x',y') = (x,y) \cdot \begin{vmatrix} a & b \\ c & d \end{vmatrix}$$

Obsérvese que, en el formalismo usado, para permitir que las matrices que representan a puntos sean siempre filas, el punto antecede a la transformación que se le aplica, por lo que, cuando se concatenan matrices de transformación se aplican de izquierda a derecha. Mientras que cuando se utiliza notación funcional las transformaciones geométricas se aplican de derecha a izquierda.

La notación anterior no permite expresar las traslaciones. Para poderlas incluir, manteniendo un formalismo matricial, lo que se suele realizar la transformación como una aplicación lineal en el espacio de dimensión inmediata superior.

Formalmente, una transformación geométrica se suele definir como, τ: $\mathbf{R}^n \rightarrow \mathbf{R}^n$ realizándose en tres pasos [Newm81] [Roger76]:

1. El punto a transformar, $P = (x_1,..,x_n)$, se convierte a coordenadas homogéneas. El paso a coordenadas homogéneas se realiza por medio de una aplicación

$$\tau_h : \mathbf{R}^n \rightarrow \mathbf{R}^{n+1}$$

tal que a cada punto $X \in \mathbf{R}^n$, $X = (x_1,..,x_n)$, le asocia un punto de $\mathbf{R}^{n+1}$, que conserva el valor de las n primeras componentes y le asigna valor uno a la última, $X' = (x_1,..,x_n,1)$

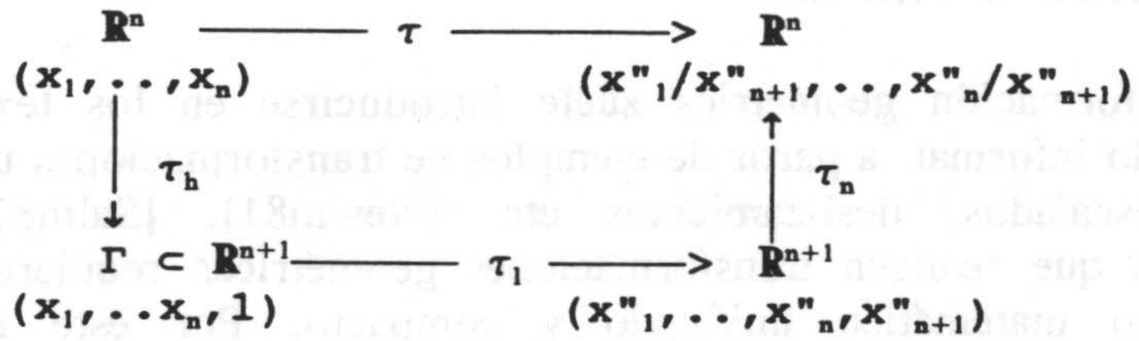
2. Al punto en coordenadas homogéneas se le aplica una transformación lineal, τ_l, en general inyectiva, salvo cuando se consideran como transformaciones geométricas las proyecciones.

3. El Punto resultante se proyecta sobre $\mathbf{R}^n$, dividiendo las n primeras componentes por la última:

$$\tau_n : \mathbf{R}^{n+1} \dashrightarrow \mathbf{R}^n$$

$$X'' = (x''_1,..,x''_{n+1}) \dashrightarrow X''' = (x''_1/x''_{n+1},..,x''_n/x''_{n+1})$$

Gráficamente la construcción de τ a partir de la composición anterior se puede representar por el siguiente grafo de funciones:



Otro enfoque consiste en asumir que las transformaciones geométricas son transformaciones afines [Prep85]. La forma general de una transformación afín es:

$$P' = P \cdot A + C$$

donde A es una aplicación lineal (que puede estar representada por una matriz n $\times$ n) y C un punto constante (que puede representarse por una matriz fila). A partir de este planteamiento se pueden estudiar las propiedades de distintas transformaciones en función de las características de la matriz A.

En determinados casos, ninguna de estas aproximaciones al problema es adecuada, siendo necesario definir las transformaciones de una forma simple y abstracta. Concretamente esto ocurre cuando se intenta establecer un formalismo matemático general que contemple distintos aspectos de los sistemas gráficos, tales como los presentados por Mallgren o Fiume. E. Fiume define las transformaciones geométricas como una biyección α: $\mathbf{R}^3$ -> $\mathbf{R}^3$ [Fium89], y Mallgren como una aplicación de $\mathbf{R}$ en $\mathbf{R}$ [Mall82]. El problema de estas definiciones es que son demasiado poco restrictivas, y por tanto incluyen como transformaciones un conjunto demasiado amplio de funciones, perdiendo por tanto las propiedades usuales y deseables de estas, como por ejemplo, el conservar los segmentos de recta.

Aquí, nos proponemos dar una definición formal de transformación geométrica, con el fin de incorporarla posteriormente al formalismo que se desarrollará. Nuestro propósito es establecer una definición abstracta, pero próxima al concepto habitual de transformación, para poder utilizarla como mecanismo de modelado. Por este motivo, separaremos las proyecciones y las transformaciones de perspectiva, que no estarán incluidas en la definición de transformación geométrica. Estas últimas transformaciones se considerarán más adelante como transformaciones de visualización. Además, de este modo se consigue que la definición de transformación geométrica asegure un buen comportamiento matemático, (biyectividad, existencia de inversa) que de otro modo no sería posible.

Para que el formalismo sea consistente con la experiencia, demostraremos que dicha definición es equivalente a la construcción usualmente utilizada en la práctica para las transformaciones geométricas.

Definición 2.3: Una transformación geométrica, τ: $\mathbf{R}^n$ -> $\mathbf{R}^n$, es una aplicación biyectiva, que cumple la siguiente propiedad:

$$\forall X,Y \in \mathbf{R}^n \; \forall ß \in [0,1] \subset \mathbf{R} \Rightarrow \tau(ß \cdot X + (1-ß) \cdot Y) = ß \cdot \tau(X) + (1-ß) \cdot \tau(Y)$$

□

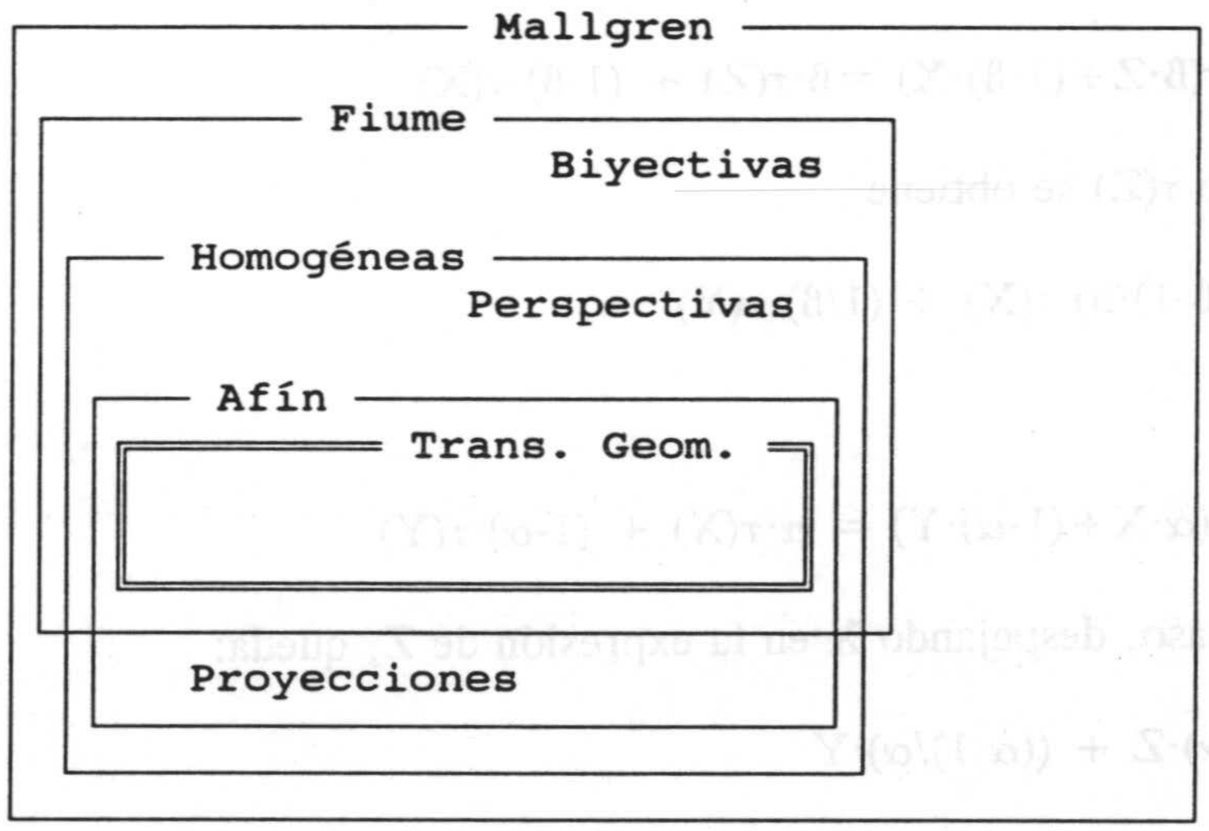
Se demostrará que la definición anterior implica que los puntos obtenidos al transformar todos los puntos de un segmento de recta formen a su vez un segmento de recta. Sin embargo, se verá, que dicha definición no implica el que la transformación geométrica sea una aplicación lineal.

La definición anterior no considera como transformaciones geométricas a las proyecciones por no ser biyectivas.

Se puede extender el concepto de transformación geométrica y considerarlo como una aplicación que transforma volúmenes en volúmenes, entendiendo que la aplicación de la transformación a un volumen consiste en su aplicación a todos sus puntos, siendo el resultado de la transformación el volumen formado por el conjunto de puntos transformados. En este caso tendremos: τ:$\mathcal{P}(\mathbf{R}^n)$ -> $\mathcal{P}(\mathbf{R}^n)$. No obstante se usará la misma notación, salvo que sea necesario distinguir ambas situaciones.

Comentarios:

La definición dada es más restrictiva que las dadas por Mallgren y Fiume, dicha definición considera las transformaciones homogéneas que se utilizan en modelado (es decir excluyendo las perspectivas y las proyecciones. El cuadro 1 muestra las relaciones existentes entre las diferentes definiciones de transformación geométrica



Cuadro 1. Relación entre definiciones de Transformación geométrica

Los siguientes lemas establecen propiedades útiles de las transformaciones geométricas.

Lema 2.1: Para toda transformación geométrica, τ, se cumple que:

$$\forall X,Y \in \mathbf{R}^n \; \forall \alpha \in \mathbf{R} \Rightarrow \tau(\alpha \cdot X + (1-\alpha)\cdot Y) = \alpha \cdot \tau(X) + (1-\alpha)\cdot \tau(Y)$$

Demostración:

Sea $Z = \alpha \cdot X + (1-\alpha)\cdot Y$ con $\alpha \in \mathbf{R}$

Se pueden presentar los tres casos siguientes, en función del valor de α:

a) $\alpha \in [0,1]$, en este caso el lema es obvio.

b) $\alpha < 0$ en este caso, despejando Y en la expresión de Z, queda:

$$Y = (1/(1-\alpha))\cdot Z + (\alpha/(1-\alpha))\cdot X$$

llamemos ß a: $ß = 1/(1-\alpha)$

por ser $\alpha < 0$ será: $ß \in (0,1)$

Quedando la expresión anterior:

$$Y = ß\cdot Z + (1-ß)\cdot X \quad \text{con } ß \in [0,1]$$

con lo que, por ser τ transformación geométrica, se cumple:

$$\tau(Y) = \tau(ß\cdot Z + (1-ß)\cdot X) = ß\cdot \tau(Z) + (1-ß)\cdot \tau(X)$$

donde despejando $\tau(Z)$ se obtiene

$$\tau(Z) = (ß-1)/ß)\cdot \tau(X) + (1/ß)\cdot \tau(Y)$$

y por tanto

$$\tau(Z) = \tau(\alpha\cdot X + (1-\alpha)\cdot Y) = \alpha\cdot \tau(X) + (1-\alpha)\cdot \tau(Y)$$

c) $\alpha > 1$ en este caso, despejando X en la expresión de Z, queda:

$$X = (1/\alpha)\cdot Z + ((\alpha-1)/\alpha)\cdot Y$$

llamemos ß a: $ß = 1/\alpha$

que por ser $\alpha > 1$ será: $ß \in (0,1)$. Y queda:

$$X = ß\cdot Z + (1-ß)\cdot Y \quad \text{con } ß \in [0,1]$$

con lo que, por ser τ transformación geométrica, se cumple:

$$\tau(X) = \tau(ß\cdot Z+(1-ß)\cdot Y) = ß\cdot\tau(Z) + (1-ß)\cdot\tau(Y)$$

donde despejando $\tau(Z)$ se obtiene

$$\tau(Z) = (1/ß)\cdot\tau(X) - ((1-ß)/ß)\cdot\tau(Y)$$

y por tanto

$$\tau(Z) = \tau(\alpha\cdot X+(1-\alpha)\cdot Y) = \alpha\cdot\tau(X) + (1-\alpha)\cdot\tau(Y)$$

□

Lema 2.2: Toda transformación lineal biyectiva $T:\mathbf{R}^n -> \mathbf{R}^n$, es una transformación geométrica.

Demostración:

Sea T: $\mathbf{R}^n -> \mathbf{R}^n$ una transformación lineal biyectiva, demostraremos que:

$\forall X,Y\in\mathbf{R}^n \ \forall ß\in[0,1] =>$ $\quad T(ß\cdot X+(1-ß)\cdot Y) = ß\cdot T(X) + (1-ß)\cdot T(Y)$

con lo que se habrá demostrado que T es una transformación geométrica.

En efecto, sean $X,Y\in\mathbf{R}^n$ dos puntos cualesquiera, por ser T lineal se debe de cumplir:

$$T(ß\cdot X + \alpha\cdot Y) = ß\cdot T(X) + \alpha\cdot T(Y) \quad \forall\alpha,ß \in \mathbf{R}$$

en particular, si $\alpha = 1 - ß$ y $ß\in[0,1]$, se tiene

$$T(ß\cdot X+(1-ß)\cdot Y) = ß\cdot T(X) + (1-ß)\cdot T(Y)$$

□

Lema 2.3: La transformación geométrica del producto de un escalar, $ß\in\mathbf{R}$, por un punto, $X\in\mathbf{R}^n$, es:

$$\tau(ß\cdot X) = ß\cdot\tau(X) + (1-ß)\cdot\tau(\nu_0)$$

donde ν_0 es el punto origen de coordenadas (0,0,...0).

Demostración:

Pongamos el producto $ß\cdot X$ en la forma:

$$ß\cdot X = ß\cdot X + (1-ß)\cdot\nu_0$$

Aplicando a dicha expresión la definición de transformación geométrica, y teniendo en cuenta el lema 2.1, resulta evidente el lema. □

Lema 2.4: Una transformación geométrica puede no ser transformación lineal.

Demostración:

Realizaremos la demostración utilizando un contraejemplo. Supongamos la transformación $T:\mathbf{R}^n \to \mathbf{R}^n$ definida por:

$$T(X) = X + B \quad X,B \in \mathbf{R}^n \text{ siendo B fijo}$$

que obviamente es biyectiva. Veamos que es una transformación geométrica

$$T(ß\cdot X+(1-ß)\cdot Y) = ß\cdot X + (1-ß)\cdot Y + B =$$

$$= ß\cdot(X+B) + (1-ß)\cdot(Y+B) =$$

$$= ß\cdot T(X) + (1-ß)\cdot T(Y)$$

Sin embargo, evidentemente T, no es lineal. □

Lema 2.5: La transformación geométrica de una combinación lineal, $\alpha\cdot X + ß\cdot Y$, de dos puntos, $X,Y \in \mathbf{R}^n$, con $\alpha,ß \in \mathbf{R}$, es:

$$\tau(\alpha\cdot X + ß\cdot Y) = \alpha\cdot\tau(X) + ß\cdot\tau(Y) + (1 - \alpha - ß)\cdot\tau(\nu_0)$$

donde ν_0 es el origen de coordenadas.

Demostración:

Demostraremos en primer lugar dicha igualdad para el caso en que $\alpha \neq 1$. En dicho caso, la expresión $\alpha\cdot X + ß\cdot Y$ se puede poner en la forma:

$$\alpha\cdot X + ß\cdot Y = \alpha\cdot X + (1-\alpha)\cdot[ß / (1-\alpha)\cdot Y]$$

por tanto

$$\tau(\alpha\cdot X + ß\cdot Y) = \alpha\cdot\tau(X) + (1-\alpha)\cdot\tau(ß/(1-\alpha)\cdot Y)$$

Aplicando el lema 2.3 al último factor tenemos

$$\tau(ß/(1-\alpha)\cdot Y) = ß/(1-\alpha)\cdot\tau(Y) + (1 - ß/(1-\alpha))\cdot\tau(\nu_0) =$$

$$= ß/(1-\alpha)\cdot\tau(Y) + ((1 - \alpha - ß)/(1-\alpha))\cdot\tau(\nu_0)$$

con lo que

$$\tau(\alpha\cdot X + ß\cdot Y) = \alpha\cdot\tau(X) + ß\cdot\tau(Y) + (1 - \alpha - ß)\cdot\tau(\nu_0)$$

Queda por demostrar dicha expresión en el caso en que $\alpha = 1$. En dicho caso, y suponiendo que ß es también uno (si no fuese así se pueden permutar los sumandos) tendremos:

$$\tau(X + Y) = \tau(2 \cdot X + (Y-X))$$

expresión a la que se le puede aplicar el resultado anterior, quedando:

$$\tau(X+Y) = 2 \cdot \tau(X) + \tau(-X+Y) - 2 \cdot \tau(\nu_0)$$

aplicando la misma expresión al segundo sumando queda:

$$\tau(X+Y) = 2 \cdot \tau(X) - \tau(X) + \tau(Y) + \tau(\nu_0) - 2 \cdot \tau(\nu_0) =$$

$$= \tau(X) + \tau(Y) - \tau(\nu_0)$$

con lo que queda demostrado el lema. □

Lema 2.6: La imagen mediante una transformación geométrica de un segmento de recta es un segmento de recta.

Demostración:

Demostraremos que la imagen $\tau(X)$ de los puntos $X \in \mathbb{R}^n$ de un segmento de recta por una transformación geométrica τ forman un segmento de recta.

Sea el segmento de recta que une el punto $B \in \mathbb{R}^n$ con el $B+D \in \mathbb{R}^n$. Cualquier punto X del segmento que une ambos puntos se puede expresar en forma paramétrica como:

$$X = B + t \cdot D \quad \text{con } t \in [0,1] \subset \mathbb{R}$$

El transformado, mediante τ, de un punto X del segmento es:

$$\tau(X) = \tau(B + t \cdot D)$$

que por el lema 2.5 es:

$$\tau(X) = \tau(B) + t \cdot \tau(D) - t \cdot \tau(\nu_0)$$

Teniendo en cuenta que

$$\tau(B+D) = \tau(B) + \tau(D) - \tau(\nu_0)$$

podemos reescribir

$$\tau(X) = \tau(B) + t \cdot (\tau(B+D) - \tau(B))$$

que es la ecuación del segmento de recta que une $\tau(B)$ con $\tau(B+D)$. □

Lema 2.7: El transformado X' de un punto cualquiera $X=(x_1,x_2,..x_n) \in \mathbf{R}^n$ mediante una transformación geométrica τ se puede expresar en la forma

$$X' = \sum_{k=1}^{n} [\lambda_{k0} + \sum_{i=1}^{n} x_i \cdot (\lambda_{ki} - \lambda_{k0})] \cdot \nu_k$$

donde $(\nu_1,..,\nu_n)$ es una base de $\mathbf{R}^n$ y los $\lambda_{ki} \in \mathbf{R}$

Demostración:

Sea $X = (x_1,x_2,..x_n) \in \mathbf{R}^n$ y sea τ una transformación geométrica. X se puede expresar en la forma:

$$X = x_1 \cdot \nu_1 + \sum_{i=2}^{n} x_i \cdot \nu_i$$

En virtud del lema 2.5 podemos calcular $\tau(X)$ como:

$$\tau(X) = x_1 \cdot \tau(\nu_1) + \tau(\sum_{i=2}^{n} x_i \cdot \nu_i) - x_1 \cdot \tau(\nu_0)$$

aplicando reiteradamente el mismo lema a la sumatoria queda:

$$\tau(X) = \sum_{i=1}^{n-2} x_i \cdot \tau(\nu_i) + \tau(x_{n-1} \cdot \nu_{n-1} + x_n \cdot \nu_n) -$$

$$\sum_{i=1}^{n-2} x_i \cdot \tau(\nu_0)$$

aplicando una última vez dicho lema queda:

$$\tau(X) = \sum_{i=1}^{n} x_i \cdot \tau(\nu_i) + [1 - \sum_{i=1}^{n} x_i \cdot \tau(\nu_0)]$$

Los transformado de cada elemento de la base y del origen se pueden expresar en la forma

$$\tau(\nu_i) = \sum_{k=1}^{n} \lambda_{ki} \cdot \nu_k \quad i \in \{0,n\}$$

Sustituyendo en la expresión de $\tau(X)$ queda

$$\tau(X) = \sum_{i=1}^{n} \sum_{k=1}^{n} x_i \cdot \lambda_{ki} \cdot \nu_k + [1 - \sum_{i=1}^{n} x_i \sum_{k=1}^{n} \lambda_{k0} \cdot \nu_k]$$

agrupando términos queda

$$\tau(X) = \sum_{k=1}^{n} [\lambda_{k0} + \sum_{i=1}^{n} x_i \cdot (\lambda_{ki} - \lambda_{k0})] \cdot \nu_k$$

con lo que se completa la demostración. □

Veremos, a continuación, que la definición presentada es compatible con la definición habitual basada en la utilización de transformaciones lineales en coordenadas homogéneas. Para ello se demostrará que toda transformación geométrica, en el sentido de la definición 2.3, se puede expresar como transformación lineal en coordenadas homogéneas, y que toda

transformación en coordenadas homogéneas que sea biyectiva es una transformación lineal en el sentido de la definición anterior.

Teorema 2.1: La aplicación $\tau_n\ \tau_l\ \tau_h$ es una transformación geométrica, siempre que τ_l sea una transformación lineal biyectiva que cumpla:

$$x'_{n+1} = w \cdot x_{n+1}$$

siendo x_{n+1} la coordenada n+1 del punto a transformar, x'_{n+1} la coordenada n+1 del punto transformado y $w \neq 0$ una constante real para la transformación geométrica.

Demostración:

1. Demostraremos en primer lugar que dicha composición es biyectiva.

Sea $\tau = \tau_n\ \tau_l\ \tau_h$ y sean $X, Y \in \mathbf{R}^n$ con

$$X = (x_1, x_2, .., x_n)$$

$$Y = (y_1, y_2, .., y_n)$$

entonces

$$\tau_h(X) = X' = (x_1, x_2, .., x_n, 1) \quad X' \in \mathbf{R}^{n+1}$$

$$\tau_l(X') = X'' = (x''_1, x''_2, .., x''_n, w \cdot x_{n+1}) \quad X'' \in \mathbf{R}^{n+1}$$

$$\tau_n(X'') = X''' = (x''_1/x''_{n+1}, x''_2/x''_{n+1}, .., x''_n/x''_{n+1}) \quad X''' \in \mathbf{R}^n$$

Demostraremos que τ es inyectiva. Para ello veamos que, si las imágenes X''' e Y''' de dos puntos X e Y son iguales, ambos puntos son necesariamente iguales:

si $X''' = Y''' \Rightarrow X'' = k \cdot Y''$ con $k \in \mathbf{R}$

si $X'' = k \cdot Y'' \Rightarrow X' = k \cdot Y'$ por ser τ_l biyectiva y lineal

si $X' = k \cdot Y' \Rightarrow X = Y \wedge k = 1$ por la definición de τ_h

con lo queda demostrado que τ es inyectiva.

Veamos ahora que la composición $\tau_n\ \tau_l\ \tau_h$ es suprayectiva.

Sea $\tau = \tau_n\ \tau_l\ \tau_h$ y sea $X''' \in \mathbf{R}^n$ con

$$X''' = (x'''_1, x'''_2, .., x'''_n)$$

demostraremos que $\exists\ X \in \mathbf{R}^n \mid \tau(X) = X'''$

Construyamos el punto X" dado por

$$X" = (w\cdot x"'_1, w\cdot x"'_2, .., w\cdot x"'_n, w) \in \mathbf{R}^{n+1}$$

siendo w el coeficiente de la componente n+1 de τ_1.
La imagen de X" por τ_n es

$$\tau_n(X") = (x"_1, x"_2, .., x"_n) = X"' \in \mathbf{R}^n$$

Por ser τ_1 biyectiva, debe de existir un punto $X' \in \mathbf{R}^{n+1}$, tal que

$$\tau_1(X') = X" \in \mathbf{R}^{n+1}$$

además, por la definición de τ_1 este punto debe ser de la forma:

$$X' = (x'_1, x'_2, .., x'_n, 1)$$

Obviamente, este punto está en la imagen de τ_h, y por tanto

$$\exists\, X \in \mathbf{R}^n \mid \tau_h(X) = X'$$

con lo que queda probado que

$$\forall\, X"' \in \mathbf{R}^n \;\exists\, X \in \mathbf{R}^n \mid \tau_h\, \tau_1\, \tau_n(X) = X"'$$

y por tanto dicha composición es suprayectiva sobre $\mathbf{R}^n$.

2. Demostraremos ahora que la composición $\tau_n\ \tau_1\ \tau_h$ cumple la condición dada en la definición 2.3:

$$\forall X, Y \in \mathbf{R}^n\ \forall ß \in [0,1] \Rightarrow \quad \tau(ß\cdot X + (1-ß)\cdot Y) = ß\cdot\tau(X) + (1-ß)\cdot\tau(Y)$$

En efecto, sean $X, Y \in \mathbf{R}^n$ dos puntos cualesquiera y el número real $ß \in [0,1]$ arbitrario. Construyamos la imagen de

$$ß\cdot X + (1-ß)\cdot Y$$

por $\tau_n\ \tau_1\ \tau_h$:

$$\tau_h(ß\cdot X + (1-ß)\cdot Y) = (ß\cdot x_1 + (1-ß)\cdot y_1, .., ß\cdot x_n + (1-ß)\cdot y_n, 1) =$$

$$= (ß\cdot x_1 + (1-ß)\cdot y_1, .., ß\cdot x_n + (1-ß)\cdot y_n, ß + (1-ß)) =$$

$$= ß\cdot\tau_h(X) + (1-ß)\cdot\tau_h(Y)$$

Por ser τ_1 lineal se cumple que

$$\tau_1(\tau_h(ß\cdot X + (1-ß)\cdot Y)) = ß\cdot\tau_1(\tau_h(X)) + (1-ß)\cdot\tau_1(\tau_h(Y))$$

teniendo en cuenta la definición de τ_l, y llamando

$$X'' = \tau_l(\tau_h(X)) \quad e \quad Y'' = \tau_l(\tau_h(Y))$$

$$\tau_l(\tau_h(\beta \cdot X + (1-\beta) \cdot Y)) =$$

$$= (\beta \cdot x''_1 + (1-\beta) \cdot y''_1, .., \beta \cdot x''_n + (1-\beta) \cdot y''_n, w)$$

y aplicando τ_n al resultado anterior

$$\tau_n(\tau_l(\tau_h(\beta \cdot X + (1-\beta) \cdot Y))) =$$

$$= ([\beta \cdot x''_1 + (1-\beta) \cdot y''_1]/w, .., [\beta \cdot x''_n + (1-\beta) \cdot y''_n]/w) =$$

$$= \beta \cdot (x''_1/w, .., x''_n/w) + (1-\beta) \cdot (y''_1/w + (1-\beta) \cdot y''_n/w) =$$

$$= \beta \cdot \tau(X) + (1-\beta) \cdot \tau(Y)$$

con lo que queda demostrado que la composición $\tau_n\ \tau_l\ \tau_h$ es una transformación geométrica. □

Teorema 2.2: Toda transformación geométrica se puede descomponer en la forma $\tau_n\ \tau_l\ \tau_h$ donde τ_l sea una transformación lineal biyectiva, τ_l: $\mathbf{R}^{n+1}$ -> $\mathbf{R}^{n+1}$, que cumpla: $\tau_l(X) = (x'_1, .., x'_n, w \cdot x_{n+1})$ con $w \neq 0$.

Demostración:

Se hará una demostración constructiva. Dado que las funciones τ_h y τ_n son fijas, construiremos τ_l para una transformación geométrica cualquiera, y demostraremos que la composición de las tres es igual a la transformación original y que τ_l cumple las condiciones del teorema.

Sea una transformación geométrica τ en $\mathbf{R}^n$. En virtud del lema 2.7 podemos expresar la acción de dicha transformación sobre cualquier punto como:

$$\tau(X) = \sum_{k=1}^{n} [\lambda_{k0} + \sum_{i=1}^{n} x_i \cdot (\lambda_{ki} - \lambda_{k0})] \cdot \nu_k$$

Construyamos la aplicación

$$\tau_l: \mathbf{R}^{n+1} \text{ -> } \mathbf{R}^{n+1}$$

definida de forma tal que al ser aplicada sobre un punto $X = (x_1, x_2, .., x_{n+1}) \in \mathbf{R}^{n+1}$, resulta:

$$\tau_l(X) = \sum_{k=1}^{n} [\sum_{i=1}^{n+1} x_i \cdot \sigma_{ki}] \cdot \nu_k + x_{n+1} \cdot \nu_{n+1}$$

estando los coeficientes σ_{ki} definidos como

$$\sigma_{ki} = \lambda_{ki} - \lambda_{k0} \quad \text{si } i \in [1..n]$$

$$\sigma_{k,n+1} = \lambda_{k0}$$

obviamente la aplicación τ_l así definida es lineal y es de la forma exigida en el enunciado del teorema, siendo para ella w=1. Existen infinitas matrices que representa a una transformación geométrica dada, una para cada valor de w. Hemos elegido el valor 1 para w en esta demostración para simplificar los cálculos.

Será necesario, por tanto demostrar que la composición $\tau_n\ \tau_l\ \tau_h$ es igual a τ y que τ_l es biyectiva.

1. La transformación τ es igual a la composición $T=\tau_n\ \tau_l\ \tau_h$

En efecto, teniendo en cuenta la construcción de τ_h y τ_l, la acción de la composición sobre cualquier punto será:

$$\tau_l(\tau_h(X)) = \sum_{k=1}^{n}[\sum_{i=1}^{n+1} x_i \cdot \sigma_{ki}] \cdot \nu_k + \nu_{n+1}$$

$$T(X) = \tau_n(\tau_l(\tau_h(X))) = \sum_{k=1}^{n}[\sum_{i=1}^{n+1} x_i \cdot \sigma_{ki}] \cdot \nu_{k0}$$

teniendo en cuenta la definición de los coeficiente σ_{ki} tendremos:

$$T(X) = \sum_{k=1}^{n} [\lambda_{k0} + \sum_{i=1}^{n} x_i \cdot (\lambda_{ki} - \lambda_{k0})] \cdot \nu_k$$

que coincide con $\tau(X)$

2. Demostraremos a continuación que τ_l es biyectiva.

τ_l es obviamente inyectiva, pues en caso contrario no podría serlo τ.

Para ver que τ_l es suprayectiva basta con considerar que τ si lo es, que τ_n no modifica las proporciones entre las n primeras componentes y que τ_l deja libre la componente n+1.□

En determinados casos se usará sólo la matriz τ_l para denotar la transformación geométrica, sobrentendiendo la aplicación de τ_h y τ_n.

Lema 2.8: La transformación identidad, que denotaremos por î, se construye tomando como τ_l la matriz identidad.

Demostración:

Sea $\tau = \tau_n\ \tau_{\hat{1}}\ \tau_h$ donde $\tau_{\hat{1}}$ es la matriz identidad en $\mathbf{R}^{n+1}$, demostraremos que $\tau(X) = X$

Sea X un punto cualquiera $X \in \mathbf{R}^n$

$$\tau(X) = \tau_n(\tau_{li}(\tau_h(X))) = \tau_n(\tau_{li}(x_1,x_2,..x_n,1)) =$$

$$= \tau_n(x_1,x_2,..x_n,1) = X \qquad \square$$

Teorema 2.3: Toda transformación geométrica τ posee inversa, τ^{-1}, y ésta es igual a:

$$\tau^{-1} = \tau_n \tau_l^{-1} \tau_h$$

Demostración:

Sea $\tau_- = \tau_n \tau_l^{-1} \tau_h$ demostraremos que $\tau \tau_- = \tau_- \tau = \hat{1}$
En efecto:

$$\tau \tau_- = \tau_n \tau_l \tau_h \tau_n \tau_l^{-1} \tau_h$$

La composición $\tau_h \tau_n$ transforma cada punto

$X=(x_1,..,x_n,x_{n+1}) \in \mathbf{R}^{n+1}$ en $\tau_h(\tau_n(X)) = (x_1/x_{n+1},..,x_n/x_{n+1},1) =$

$$= (1/x_{n+1}) \cdot (x_1,..,x_n,x_{n+1})$$

por ser τ_l lineal se cumple que:

$$\tau_l(\tau_h(\tau_n(X))) = (1/x_{n+1}) \cdot \tau_l(x_1,..,x_n,x_{n+1}) = (1/x_{n+1}) \cdot \tau_l(X)$$

y por tanto:

$$\tau_l(\tau_h(\tau_n(\tau_l^{-1}(X')))) = (1/x_{n+1}) \cdot X' \quad \text{siendo } \tau_l(X) = X'$$

y teniendo en cuenta que $\tau_n(K \cdot X) = \tau_n(X)\ \forall X \in \mathbf{R}^n, K \in \mathbf{R}$ se concluye

$$\tau_n(\tau_l(\tau_h(\tau_n(\tau_l^{-1}(X'))))) = \tau_n(X') \quad \text{y por tanto}$$

$$\tau \tau_- = \tau_n \tau_h$$

que obviamente es igual a la identidad en $\mathbf{R}^n$.

De idéntico modo se comprueba que $\tau_- \tau$ es igual a la identidad. $\square$

Teorema 2.4: La composición de dos transformaciones geométricas, $\tau_2(\tau_1)$, es una transformación geométrica, y se puede expresar en la forma:

$$\tau = \tau_2 \tau_1 = \tau_n \tau_{l2} \tau_{l1} \tau_h$$

Demostración:

Veamos que τ es una transformación geométrica, para lo cual se basta con demostrar que

$$\forall X,Y\in\mathbf{R}^n\ \forall ß\in[0,1]\subset\mathbf{R} => \tau(ß\cdot X+(1-ß)\cdot Y) = ß\cdot\tau(X) + (1-ß)\cdot\tau(Y)$$

ya que τ es biyectiva. Por ser τ_1 transformación geométrica se tiene que

$$\tau(ß\cdot X+(1-ß)\cdot Y) = \tau_2(ß\cdot\tau_1(X) + (1-ß)\cdot\tau_1(Y))$$

y por ser τ_2 transformación geométrica

$$\tau(ß\cdot X+(1-ß)\cdot Y) = ß\cdot\tau_2(\tau_1(X)) + (1-ß)\cdot\tau_2(\tau_1(Y)) = ß\cdot\tau(X) + (1-ß)\cdot\tau(Y)$$

La composición de las dos transformaciones se puede expresar en la forma:

$$\tau_2\ \tau_1 = \tau_n\ \tau_{l2}\ \tau_h\ \tau_n\ \tau_{l1}\ \tau_h$$

siguiendo un razonamiento paralelo al de la demostración del teorema 2.3, se encuentra que la composición $\tau_h\ \tau_n$ transforma cada punto

$$X=(x_1,..,x_n,x_{n+1})\in\mathbf{R}^{n+1}$$

en

$$\tau_h(\tau_n(X)) = (x_1/x_{n+1},..,x_n/x_{n+1},1) =$$

$$= (1/x_{n+1})\cdot(x_1,..,x_n,x_{n+1})$$

por ser τ_{l2} lineal se cumple que:

$$\tau_{l2}(\tau_h(\tau_n(X))) = (1/x_{n+1})\cdot\tau_{l2}(X)$$

y por tanto:

$$\tau_{l2}(\tau_h(\tau_n(\tau_{l1}(X')))) = (1/x_{n+1})\cdot\tau_{l2}(\tau_{l1}(X'))\quad \text{siendo } \tau_{l1}(X') = X$$

y teniendo en cuenta que $\tau_n(K\cdot X) = \tau_n(X)\ \forall X\in\mathbf{R}^n, K\in\mathbf{R}$ se concluye

$$\tau_n\ \tau_{l2}\ \tau_h\ \tau_n\ \tau_{l1}\ \tau_h = \tau_n\ \tau_{l2}\ \tau_{l1}\ \tau_h$$

□

La definición formal de transformación geométrica que se ha dado servirá de soporte al desarrollo del concepto de objeto. El hecho de que las transformaciones sean biyectivas permite asegurar la existencia de transformaciones inversas, que se utilizarán frecuentemente a lo largo del formalismo.

2.1.4 Presencia.

Al definir, más adelante, el concepto de objeto gráfico, necesitaremos considerar información sobre su ocupación del espacio. Para ello no nos bastará con conocer el volumen de los objetos, ya que deseamos poder contemplar situaciones en las que varios objetos se superpongan en una misma zona del espacio. Por tanto, necesitaremos, además de conocer los volúmenes, saber cuantos objetos hay superpuestos en cada punto del espacio. Para formalizar este recuento de superposiciones introducimos ahora el concepto de dominio de presencia.

La presencia de un objeto gráfico describe su ocupación del espacio. Podemos pensar en la presencia como una característica del objeto que indica, para cada punto del espacio, si el objeto tiene o no materia en él, y cuanta tiene.

Hablaremos de ocupación o presencia del objeto en el espacio en lugar de extensión, ya que esta última se suele entender como una propiedad cualitativa, es decir nos permite saber si el objeto está o no en un punto del espacio. Entenderemos a la presencia, por el contrario, como una propiedad cuantitativa, es decir podremos preguntar cuantas veces está, en el sentido de presencia, un objeto en un punto del espacio.

Por las razones expuestas al justificar la introducción del concepto de dominio de aspecto, se introduce la estructura algebraica dominio de presencia. Este dominio se usará, por tanto, como soporte matemático para contar los objetos superpuestos en cada punto del espacio. Un valor de presencia del objeto igual a cero en un punto del espacio indicará la no existencia del objeto en dicho punto, un valor uno indicará la existencia del objeto. Para un objeto dado, un valor de presencia negativo en un punto se interpretará diciendo que el objeto se ha formado por sustracción de 'materia' en dicho punto. Una valor de presencia mayor que uno indicará que el objeto se ha construido, en dicho punto, por superposición de varios objetos.

Definición 2.4: Un dominio de presencia, **P**, es un intervalo cerrado de números enteros, cuya cardinalidad es potencia de dos (2^r) y sobre el que están definidas las siguientes operaciones:

1. Está definida una operación interna, $+$, que denominaremos suma, respecto a la cual **P** es un grupo abeliano.

2. Existe una operación externa, $*$, que denominaremos producto por escalar, con la cual $(\mathbf{P}, +, *)$ es un espacio vectorial.

3. Existen dos operaciones binarias internas, $\cup$ $\cap$, que denominaremos unión e intersección, y una operación unaria, $\sim$, que denominaremos complementación, respecto a las cuales **P** es un álgebra de Boole. El elemento vacio de dicha álgebra coincide con el elemento neutro para la estructura de espacio vectorial.

4. Existe una operación interna, $\times$, que denominaremos producto, que satisface las siguientes propiedades:

   · Es asociativa
   · Es conmutativa

· Posee elemento unidad.
· El elemento neutro del espacio vectorial es elemento absorvente. □

Comentarios:

La inclusión en la definición de la restricción sobre la cardinalidad del dominio no es consecuencia del concepto de presencia, si no de la estructura algebraica que debe de tener éste. No obstante dicha restricción, no impone ninguna limitación en la práctica. El dominio de presencia no será necesariamente un anillo con las operaciones, $+$ $\times$, ya que su cardinalidad no es un número primo.

En determinados casos podrá interesar, para reducir el número de operaciones distintas definidas para la presencia, hacer que el producto interno coincida con la intersección.

El que la cardinalidad sea finita permitirá la utilización de una representación binaria, sobre la que se puedan realizar las operaciones booleanas.

Ejemplo 2.3: Un ejemplo simple de dominio de presencia es el dado por el intervalo de enteros [-q,q-1], definiendo las operaciones de un modo análogo a como se definieron para el espacio de color $\mathbf{C}_{rgb}$ [Ejemplo 2.1]. Formalmente se puede definir este dominio como:

$\mathbf{P}_. = (\mathbf{s}, +, *, \cup, \cap, \sim)$ en la que:

$\mathbf{s}$ es $[-2^{r-1},2^{r-1}-1] \subset \mathbf{Z}$, con $q = 2^{r-1}$

$+$ es la suma cerrada sobre el intervalo.
$*$ es el producto sobre el intervalo.
$\times$ es el producto de enteros del intervalo.

Las operaciones booleanas se pueden definir a nivel de bit, en base a la representación en complemento a dos de cada uno de los operandos.

Si denotamos por $x_0^b x_1^b .. x_{r-2}^b x_{r-1}^b$ la representación en complemento a dos de $x \in \mathbf{s}$, las operaciones se definen del siguiente modo:

$\sim$ se define como el resultado de la complementación de la representación binaria bit a bit:

$$\sim x = \sim(x_0^b .. x_{r-1}^b) = \sim x_0^b \sim x_1^b .. \sim x_{r-1}^b \quad x \in \mathbf{s}$$

$\cup$ se define como la unión bit a bit:

$$x \cup y = x_0^b \cup y_0^b .. x_{r-1}^b \cup y_{r-1}^b \quad x,y \in \mathbf{s}$$

$\cap$ se define como la intersección bit a bit:

$$x \cap y = x_0^b \cap y_0^b .. x_{r-1}^b \cap y_{r-1}^b \qquad x,y \in s$$

Es inmediato demostrar que $\mathbf{P}_-$ satisface la estructura de dominio de presencia antes definida.
Este dominio cumple las siguientes propiedades:

$$\sim 0 = -1$$

$$\sim x = -x-1$$

$$\text{si } x<0 \wedge y>0 \Rightarrow x \cup y < 0 \; ; \; x \cap y > 0$$

$$\text{si } x<0 \wedge y<0 \Rightarrow x \cup y < 0 \; ; \; x \cap y < 0$$

$$\text{si } x>0 \wedge y>0 \Rightarrow x \cup y > 0 \; ; \; x \cap y > 0$$

$$\text{si } x > y > 0 \Rightarrow x \cup y \geq x \; ; \; x \cap y \leq y$$

En el ejemplo anterior el complemento del 0 es menos uno. Esta situación ocurre por estar definido como una operación bit a bit sobre la representación en complemento a dos del número. En algunas situaciones este comportamiento de la presencia no será deseable, pues ello implica que el complemento de la no existencia (presencia cero) sea la sustracción de objeto (presencia menos uno). En los casos concretos en que se desee evitar dicho comportamiento se deberá definir el dominio de presencia de forma que el complemento del cero sea el uno. Esto se corresponde con la idea natural de complemento de volúmenes. Es posible definir dominios de presencia que mantengan dicha propiedad, tal como se muestra en el siguiente ejemplo.

Ejemplo 2.4: Se pueden construir dominios de presencia distintos al del ejemplo anterior, en los que el complemento del cero sea el uno. Así, por ejemplo, se puede utilizar, como conjunto soporte, el intervalo $s_+ = [-q,q+1]$. Formalmente se puede definir este dominio como:

$\mathbf{P_n} = (s_+, +, *, \cup, \cap, \sim)$ en la que:

s_+ es $(-2^{r-1}-1, 2^{r-1})$, con $q = 2^{r-1}$

\+ es la suma cerrada sobre el intervalo.
* es el producto sobre el intervalo.

Las operaciones booleanas se pueden definir a nivel de bit, en base a la representación en complemento a dos del opuesto del número (-x). Es decir cada número

x se representa como -x en complemento a dos. Si denotamos por $x^b_0x^b_1..x^b_{r-2}x^b_{r-1}$ la representación interna de x, las operaciones se definen como en el ejemplo 2.2.

Es inmediato demostrar que $\mathbb{P}_n$ satisface la estructura de dominio de presencia antes definida.
Este dominio cumple las siguientes propiedades:

$$\sim 0 = 1$$

$$\sim x = -x+1$$

$$\text{si } x<0 \wedge y>0 \Rightarrow x \cup y > 0 \; ; \; x \cap y < 0$$

$$\text{si } x<0 \wedge y<0 \Rightarrow x \cup y < 0 \; ; \; x \cap y < 0$$

$$\text{si } x>0 \wedge y>0 \Rightarrow x \cup y > 0 \; ; \; x \cap y > 0$$

$$\text{si } x > y > 0 \Rightarrow x \cup y \leq y \; ; \; x \cap y \geq x$$

Este dominio se podría haber definido a partir del dominio $\mathbf{P}_.$ del ejemplo anterior. En efecto, se puede definir una aplicación £ de s_+ en s como:

$$£(x) = -x$$

£ es una aplicación biyectiva, y por tanto tiene inversa, $£^{-1}$. Podemos definir en s_+ las operaciones inducidas por s como:

$$x,y \in s_+ \Rightarrow x + y \equiv £^{-1}(£(x) + £(y))$$

Para hacer referencia a este último tipo de dominio de presencia se introduce la siguiente definición.

Definición 2.5: Se dice que un dominio de presencia es normal si cumple que $\sim 0 = 1$. □

En adelante se hará referencia a dominios normales, salvo que se especifique lo contrario.

2.2 OBJETOS GRAFICOS.

En nuestra conceptualización, un **objeto gráfico** es una abstracción gráfica de un sistema. Hemos utilizado el término abstracción gráfica para hacer énfasis en el hecho de que la información geométrica y visual es la más importante en dicha abstracción.

El hecho de que la información geométrica sea importante fuerza el que se deba de considerar como parte de la abstracción, la posición y orientación, del objeto. Esto conducirá a que cualquier cambio de posición u orientación de un objeto genere un nuevo objeto. Así pues, la información de posición y orientación formará parte del objeto, pudiendo dos objetos diferenciarse tan sólo en su ubicación u orientación en el espacio.

En la práctica, no obstante, puede ser interesante trabajar a distintos niveles de abstracción. Concretamente podrá interesar prescindir de la información de posición y orientación, considerando equivalentes dos objetos que se diferencian tan sólo en dicha información. Esto es, se consideran equivalentes dos objetos cuando se puede obtener el uno a partir del otro, aplicando una transformación geométrica. Este concepto de equivalencia de objetos se formalizará más adelante. Cuando trabajemos a este nivel hablaremos de **familias** de objetos.

De este modo se establecen dos niveles de abstracción, que informalmente se pueden describir como sigue:

1• Familias de objetos. Se corresponden con objetos genéricos (físicos o ficticios), prescindiendo de información de ubicación y orientación. En este sentido, podemos hablar, por ejemplo, de mesas de una determinada familia. La familia se caracteriza por su geometría y aspecto.

2• Objetos. Se corresponden con elementos individuales de las familias. Respecto a la familia, incorpora información de ubicación y orientación. Así, por ejemplo, cada objeto de la familia mesa se encuentra en una posición concreta, con una orientación conocida.

En la práctica interesará introducir niveles de abstracción adicionales, que prescindan de otra información, además de la ubicación, por ejemplo del aspecto del objeto. Estas abstracciones se introducirán como clases de equivalencia de familias de objetos inducidas por funciones (Secc. 4).

La definición de objeto gráfico se realizará de forma que se asegure que la relación sistema-objeto gráfico, para sistemas reales, sea biunívoca, lo que conducirá a que la información de ubicación aparezca en los objetos de forma implícita, es decir, es necesario estudiar la estructura del objeto para deducir su posición, tamaño, etc. En la práctica es cómodo disponer de una representación sobre la cual se pueda manipular directamente, de forma explícita la ubicación. Con este propósito se introducirá, en la sección siguiente el

concepto de **instancia**[1]. Una instancia es una abstracción gráfica de un sistema, en la que la información de ubicación se expresa de forma explícita, lo que equivale a desglosar la información geométrica, distinguiendo entre el sistema de referencia en el que **se define** la instancia, de aquel en el que **se sitúa**. Esto es, indicar por una parte la definición geométrica de la instancia y por otra su colocación en el espacio. Lógicamente cada sistema podrá describirse usando más de una instancia, y será por tanto posible construirlo de diversas formas (cada una tendrá una transformación de instanciación y una definición diferentes).

El resto de esta sección se dedica a dar una definición formal de objeto gráfico. En la definición de un objeto intervienen tres factores: volumen, aspecto, y presencia. El **volumen** del objeto es la región del espacio en la que **está definido** el objeto. Dicho volumen no tiene por qué coincidir con el volumen material del objeto. Si se considera un objeto en movimiento, el volumen del objeto puede ser el conjunto de todos los puntos ocupados por el objeto al recorrer la trayectoria descrita. El **aspecto** da información de la apariencia visual de cada punto del volumen del objeto. La **presencia** indica para qué puntos del volumen del objeto existe realmente 'materia', y, por tanto, se puede definir una representación visual.

Tal como se ha dicho anteriormente, debe entenderse que la información de ubicación en el espacio es parte de la información que contiene el objeto. Por tanto dos objetos podrá diferenciarse tan sólo en su posición. Esto se corresponde con la idea natural de que dos objetos físicos (p.e. dos sillas) pueden ser idénticas en forma, pero no son la misma, ya que cada una está en una posición distinta.

Se pueden definir objetos gráficos en espacios de cualquier dimensión, aunque lo usual será utilizar espacios de dimensión comprendida entre dos, para objetos bidimensionales estáticos, y cuatro, para objetos tridimensionales en movimiento.

Definición 2.6: Un **objeto gráfico** es una terna (V,μ,α), en la que:

V es un volumen de $\mathbf{R}^n$, que denominaremos **volumen** del objeto.
μ es una función que denominaremos **función de presencia**, o simplemente **presencia**, del objeto, y que está definida como: μ: $\mathbf{R}^n$ -> **P**, siendo **P** un dominio de presencia.
α es una función que denominaremos **función de aspecto**, o **aspecto**, del objeto, y que está definida como α: $\mathbf{R}^n$ -> **A**, siendo **A** un dominio de aspecto.

que cumple

$$\forall P \in \mathbf{R}^n,\ P \in V \text{ sii } \alpha(P) \neq 0 \quad \vee \quad \mu(P) \neq 0 \qquad \square$$

---

[1] Esta idea se corresponde con el concepto, habitual en los sistemas gráficos, de instancia. Su inclusión aquí se debe, además de a permitir conectar con el concepto práctico de instancia, a facilitar el desarrollo del formalismo.

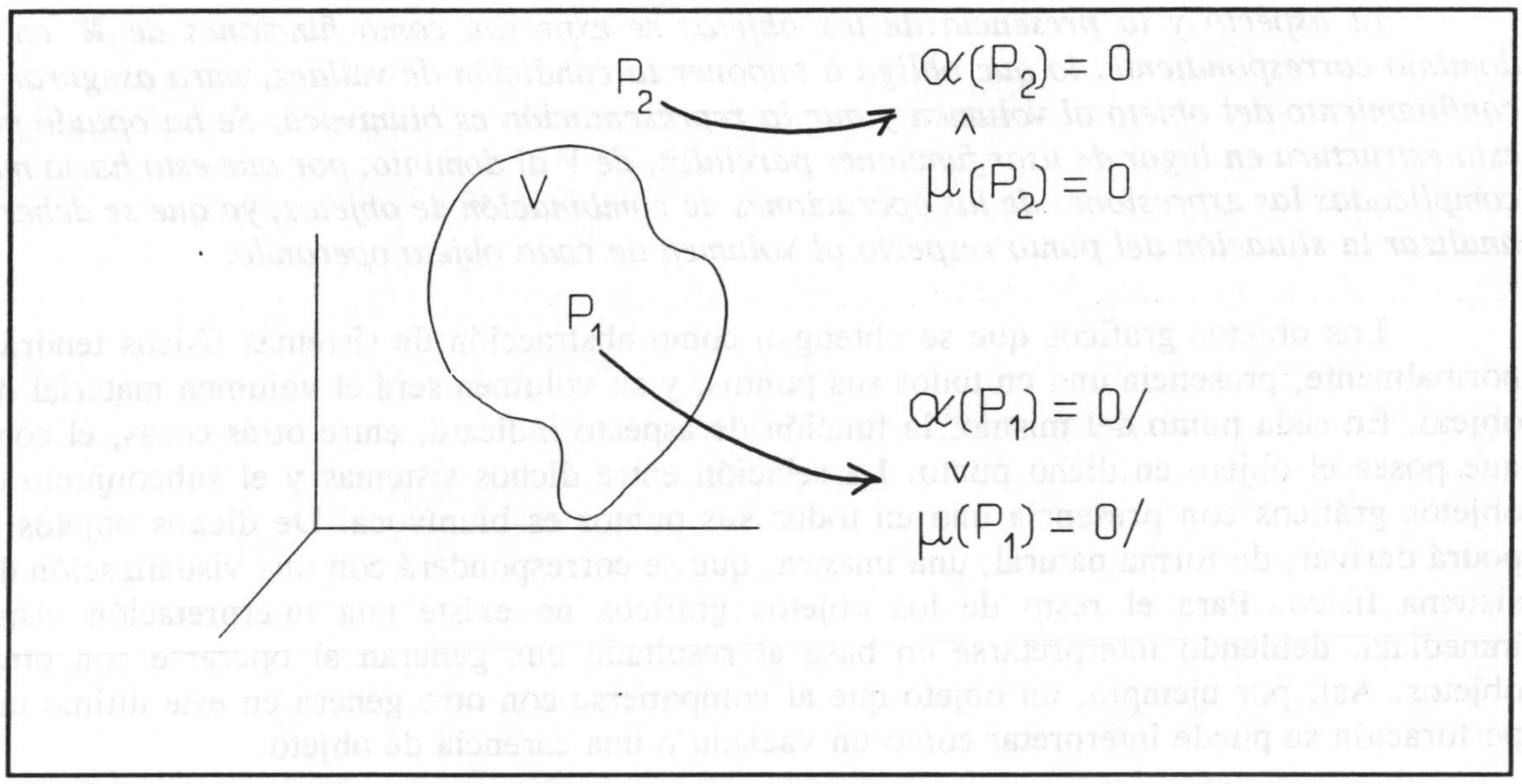


Fig. 2.2. Objeto gráfico. El punto P_1 está dentro del objeto y el punto P_2 está fuera de él.

Comentarios:

La condición impuesta en la definición anterior asegura que toda la información que se posee sobre el objeto gráfico está confinada a su volumen (ver fig.2.2). Por otra parte, el imponer que, en ningún punto del volumen del objeto, μ y α sean simultáneamente nulos, obliga a que todo el volumen del objeto contenga información del mismo, dando, por tanto, significado al volumen del objeto, que podremos entender como la mínima zona del espacio con información del objeto. Por otra parte estas condiciones aseguran que no existan dos objetos gráficos distintos que sean abstracción de un mismo objeto físico.

La función α permite especificar las características de visualización del objeto, color, comportamiento ante la luz, etc. Esta función se define para todos los puntos del volumen.

La función de presencia se utiliza para dar un significado a la suma y substracción de objetos. La función de presencia de un objeto real valdrá 0 ó 1. El que la función de presencia tome en un punto $P \in V$ valor distinto de 0 y 1 se interpretará como:

- *Si $\mu(P) > 1 \Rightarrow$ en P hay ocupación múltiple, esto es, el objeto se ha formado por superposición en dicho punto.*
- *Si $\mu(P) < 0 \Rightarrow$ en P hay ocupación negativa, esto es, el objeto se ha formado por $-\mu(P)$ substracciones de ese punto. Para obtener un objeto con "existencia real" ($\mu(P)=1$) en ese punto se podrá, por ejemplo, construir un objeto añadiendo al actual $-\mu(P)+1$ objetos en dicho punto.*

El aspecto y la presencia de los objetos se expresan como funciones de $\mathbf{R}^n$ *en el dominio correspondiente, lo que obliga a imponer la condición de validez, para asegurar el confinamiento del objeto al volumen y que la representación es biunivoca. Se ha optado por esta estructura en lugar de usar funciones parciales, de V al dominio, por que esto haría más complicadas las expresiones de las operaciones de combinación de objetos, ya que se debería analizar la situación del punto respecto al volumen de cada objeto operando.*

Los objetos gráficos que se obtengan como abstracción de sistemas físicos tendrán, normalmente, presencia uno en todos sus puntos, y su volumen será el volumen material del objeto. En cada punto del mismo, la función de aspecto indicará, entre otras cosas, el color que posee el objeto en dicho punto. La relación entre dichos sistemas y el subconjunto de objetos gráficos con presencia uno en todos sus puntos es biunívoca. De dichos objetos se podrá derivar, de forma natural, una imagen, que se corresponderá con una visualización del sistema físico. Para el resto de los objetos gráficos no existe una interpretación visual inmediata, debiendo interpretarse en base al resultado que generan al operarse con otros objetos. Así, por ejemplo, un objeto que al componerse con otro genera en este último una perforación se puede interpretar como un vaciado o una carencia de objeto.

Denotaremos el **conjunto de objetos** por $\mathbf{O}$.

Un objeto gráfico se puede describir por enumeración espacial, indicando el conjunto de puntos que forman su volumen, y dando el valor de μ y α para cada uno de ellos.

Ejemplo 2.5: Antes de proseguir veamos como se pueden definir objetos gráficos simples. Usaremos para ello el dominio de aspecto $\mathbf{C}_{rgb}$ y el dominio de presencia $\mathbf{P}_n$.

Punto. Un objeto con volumen puntual ubicado en P_0, presencia 1 y aspecto $k \in \mathbf{A}$, se puede definir del siguiente modo:

$$\text{Punto}(P_0) = (\, V, \mu, \alpha \,);$$

donde:

$$V = \{\, P_0 \,\} \subset \mathbf{R}^n$$

$$\mu(P) = \begin{cases} 1 \text{ si } P = P_0 \\ 0 \text{ si } P \neq P_0 \end{cases}$$

$$\alpha(P) = k \cdot \mu(P)$$

Obsérvese que el aspecto se ha definido a partir de la presencia por simplicidad de notación, dando valor de aspecto k al punto P_0 y valor 0 al resto de los puntos.

línea[1]. La línea entre P_1 y P_2, con presencia unidad y aspecto k se puede definir como:

$$\text{línea}(P_1,P_2) = (V, \mu, \alpha);$$

siendo:

$$V = \{ u\cdot P_1 + (u-1)\cdot P_2 \mid u \in [0,1]\ P_1,P_2 \in \mathbf{R}^3\}$$

$$\mu(P) = \begin{cases} 1 \text{ si } P \in V \\ 0 \text{ si } P \notin V \end{cases}$$

$$\alpha(P) = k\cdot\mu(P)$$

Punto moviéndose en una trayectoria P(t). Un punto que se mueve según una trayectoria P(t), en el espacio, se puede representar como un objeto gráfico definido en $\mathbf{R}^4$. Su volumen será la trayectoria, con lo que el corte para cada valor de tiempo dará un punto en el espacio 3D. La figura 2.3 muestra un corte para Y=cte. de la definición de este objeto gráfico. El objeto se puede definir del siguiente modo:

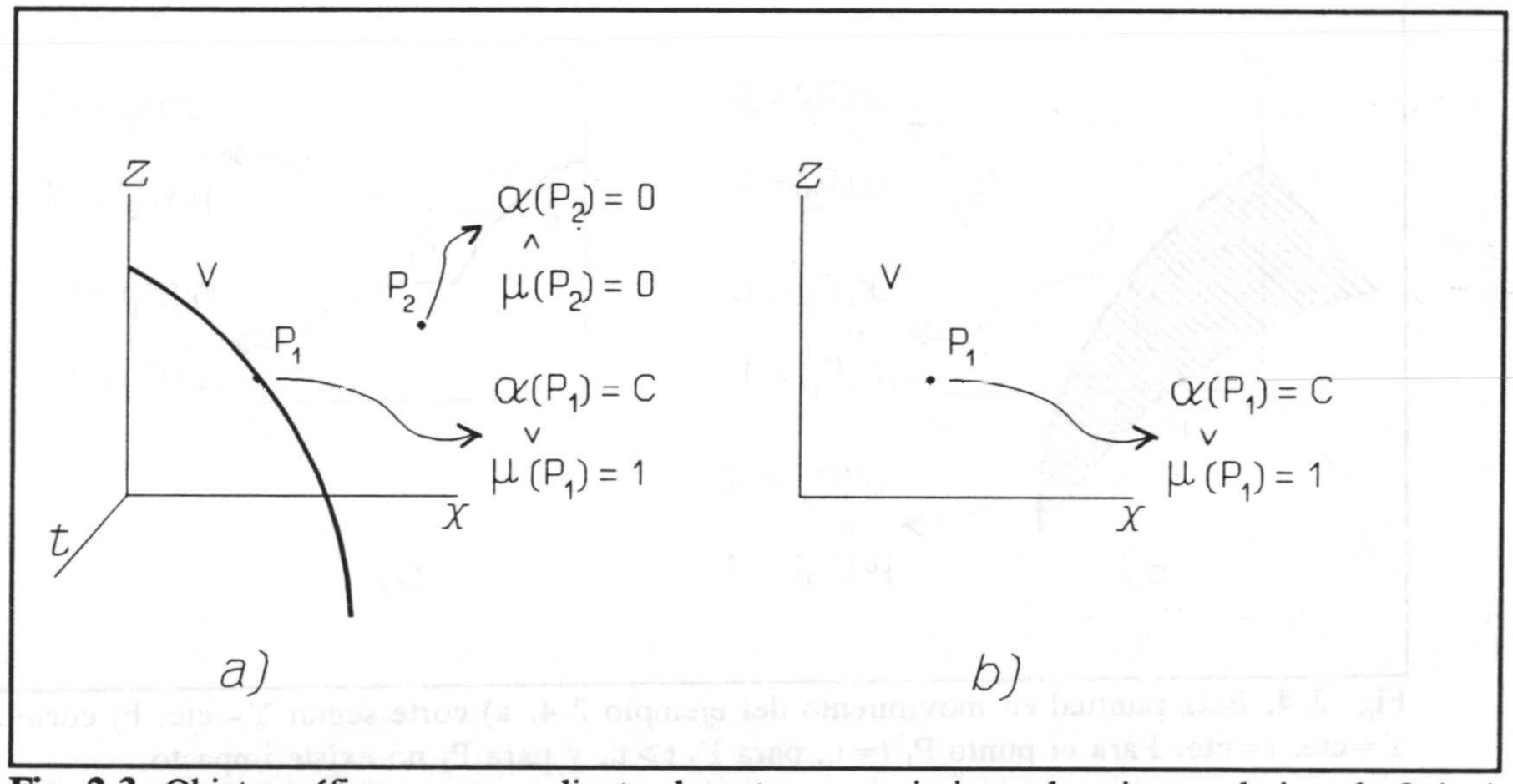


Fig. 2.3. Objeto gráfico correspondiente al punto en movimiento descrito en el ejemplo 2.4. a) Corte según Y=cte. b) Corte según Y=cte. t=cte.

---

[1] De acuerdo con la terminología habitual en Informática Gráfica, denominaremos líneas a los segmentos de recta.

punto(P(t)) = (V, μ, α)

siendo V la trayectoria del punto:

$$V = \{ (X(t),Y(t),Z(t),t) \} \subset \mathbf{R}^4$$

$$\mu(P) = \begin{cases} 1 \text{ si } P \in V \\ 0 \text{ si } P \notin V \end{cases}$$

$$\alpha(P) = k \cdot \mu(P)$$

Bala puntual moviéndose en una trayectoria P(t)

Podemos introducir una variación al objeto anterior para modelar el efecto de un objeto puntual moviéndose en una trayectoria que perfora los objetos que se encuentra a su paso. En este caso lo que se hace es dar valor menos uno a la presencia en los puntos de la trayectoria por los que ya ha pasado el objeto. La figura 2.4 muestra este objeto.

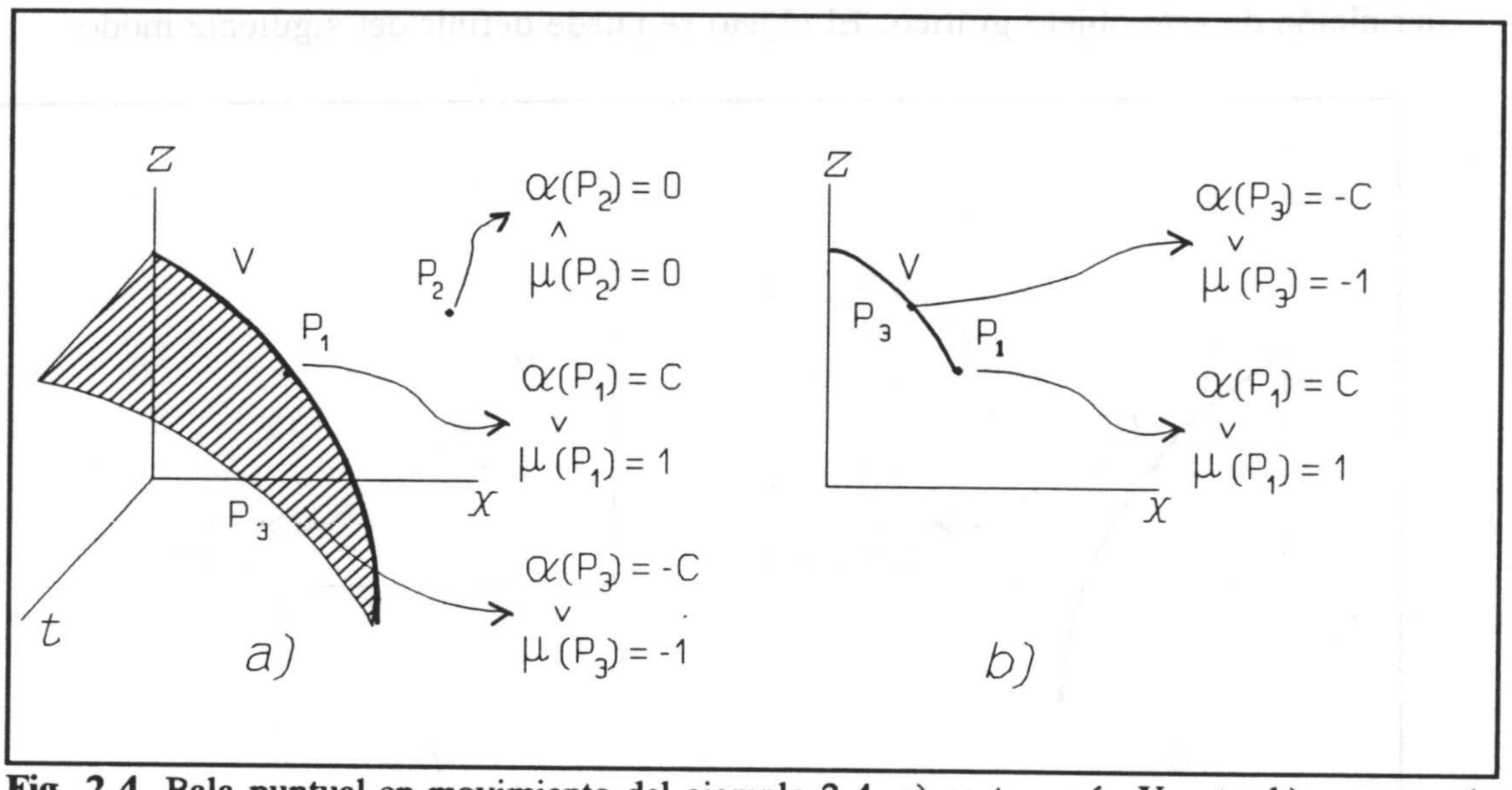


Fig. 2.4. Bala puntual en movimiento del ejemplo 2.4. a) corte según Y=cte. b) corte según Y=cte. t=cte. Para el punto P_1 $t=t_i$, para P_2 $t>t_i$, y para P_3 no existe impacto.

Bala(P(t)) = (V, μ, α)

siendo la trayectoria de la bala

$$T(P,t) = \{ (X(t),Y(t),Z(t),t) \} \subset \mathbf{R}^4$$

siendo P un punto, $P \subset \mathbf{R}^3$

Dado un punto, $P \subset \mathbf{R}^3$, puede, o no, existir un valor de t, tal que el móvil este situado en dicho punto, es decir que el móvil impacte al punto. Esta situación se puede caracterizar formalmente del siguiente modo

$\forall\, P_0=(x,y,z) \in \mathbf{R}^3$ **existe impacto** sii $\exists\, t_i \mid (x,y,z,t_i) \in T(P,t) \subset \mathbf{R}^4$

Utilizando el concepto de impacto las funciones de presencia y aspecto se pueden definir como:

$$\mu(P,t) = \begin{cases} 1 \text{ si } \exists \text{ impacto en P con } t=t_i \\ 0 \text{ si } (\exists \text{ impacto en P con } t<t_i) \vee (\nexists \text{ impacto en P}) \\ -1 \text{ si } \exists \text{ impacto en P con } t>t_i \end{cases}$$

$$\alpha(P,t) = \begin{cases} k \in \mathbf{A} \text{ si } \exists \text{ impacto en P con } t=t_i \\ 0 \text{ si } (\exists \text{ impacto en P con } t\neq t_i) \vee (\nexists \text{ impacto en P}) \end{cases}$$

El volumen se puede definir a partir de las función de presencia como

$$V = \{ (P,t) \in \mathbf{R}^4 \mid \mu(P) \neq 0 \}$$

2.3 TRANSFORMACIÓN GEOMETRICA DE OBJETOS.

Entre las operaciones que se pueden realizar con los objetos cabe destacar las transformaciones geométricas[1]. La aplicación de una transformación geométrica a un objeto supondrá la creación de un nuevo objeto que, en general, diferirá del original en su posición u orientación, pudiendo diferir en tamaño (si se ha realizado un escalado), o incluso en 'forma' (cuando se realice un deslizamiento).

Definición 2.7: La **transformación geométrica**, T(O), de un objeto, $O = (V,\mu,\alpha)$, es el objeto gráfico dada por la expresión:

$$T(O) = (T(V),\mu\cdot T^{-1},\alpha\cdot T^{-1})$$ □

Algunas transformaciones geométricas, concretamente los giros, escalados, traslaciones, y lo que denominaremos transformación circular, serán necesarias más adelante para definir las operaciones de barrido. Por este motivo se realiza a continuación una definición de estas tres transformaciones, y una caracterización de su magnitud en base a puntos. Esta caracterización permitirá expresar la magnitud de determinadas transformaciones mediante objetos gráficos.

2.4.1 Traslación.

Una traslación es una transformación geométrica en la que todos los puntos del espacio se trasladan una misma distancia. El trasladado, T(P), de un punto dado, $P = (x_1,x_2,..,x_n)$, es:

$$T(P) = (t_1+x_1,t_2+x_2,..,t_n+x_n)$$

En coordenadas homogéneas la traslación se realiza mediante una matriz $(n+1\times n+1)$ que se diferencia de la identidad en los últimos elementos de las n primeras columnas, que contienen los factores de traslación, t_i.

Resulta evidente que la magnitud de la traslación se puede representar por el punto

$$P_t = (t_1,t_2,..,t_n)$$

que es el vector que indica el desplazamiento realizado.

---

[1] Tal como se expuso en la sección 2.1.3, entenderemos las transformaciones geométricas en el sentido de la definición 2.3.

Definición 2.8: Denominaremos **traslación**, $T_t^{[P_t]}(P)$, de un punto $P \in \mathbf{R}^n$, según el punto $P_t \in \mathbf{R}^n$, a la transformación dada por

$$T_t^{[P_t]}(P) = P_t + P \qquad \square$$

La traslación es una transformación geométrica, Este resultado queda establecido por los siguientes lemas, que, además, considerada como operación binaria en $\mathbf{R}^n$, es conmutativa.

Lema 2.9: La traslación $T_t^{[P_t]}$ según $P_t \in \mathbf{R}^n$, es una transformación geométrica en $\mathbf{R}^n$.

Demostración:

Trivial. $\square$

Lema 2.10: Dados dos puntos cualesquiera, $P_1\ P_2 \in \mathbf{R}^n$, el trasladado de P_1 según P_2 coincide con el trasladado de P_2 según P_1, esto es

$$T_t^{[P_1]}(P_2) = T_t^{[P_2]}(P_1)$$

y la traslación según el origen de coordenadas es la transformación identidad

$$T_t^{[P_0]}(P) = P$$

Demostración:

Trivial. $\square$

Como caso particular, en un espacio tridimensional será:

$$T_t^{[a,b,c]}(x,y,z) = (x+a,y+b,z+c)$$

2.4.2 Rotación.

Al rotar un punto respecto al origen se mantiene invariante su distancia al origen. Para expresar matemáticamente la ecuación de una transformación de rotación es conveniente utilizar coordenadas esféricas.

Representemos a un punto, P, en coordenadas esféricas por:

$$P = (r,\Phi_1,\Phi_2,..,\Phi_{n-1})$$

donde r es la distancia al origen y Φ_i es un ángulo, medidos de tal modo que el primer vector de la base en coordenadas cartesianas, ν_1, se represente en coordenadas esféricas por (1,0,..,0).

Para hacer referencia a las componentes radial y angular del punto, se utilizan las funciones 'Rad' y 'Ang' definidas por las siguientes expresiones:

$$\text{Rad}(P) = r$$

$$\text{Ang}(P) = (\Phi_1,\Phi_2,..,\Phi_{n-1}) \qquad \square$$

Teniendo en cuenta que una rotación está especificada por n-1 ángulos de giro, podemos utilizar la componente angular de un punto para indicarla.

Definición 2.9: La rotación $T_r^{[P_r]}(P)$ de un punto, $P= (r,\Phi_1,\Phi_2,..,\Phi_{n-1}) \in \mathbf{R}^n$, según $P_r \in \mathbf{R}^n$ con componente angular $\text{Rad}(P_r) = (\Theta_1,\Theta_2,..,\Theta_{n-1})$, viene dada por

$$T_r^{[P_r]}(P) = (r,\Phi_1+\Theta_1,\Phi_2+\Theta_2,..,\Phi_{n-1}+\Theta_{n-1}) \qquad \square$$

Cualquier punto P_g tal que

$$\text{Ang}(P_g) = (\Theta_1,\Theta_2,..,\Theta_{n-1})$$

produce la misma rotación.

Intuitivamente, se puede interpretar la rotación según un punto dado, como aquella rotación que mueve un punto con componente angular (0,0,..0) sobre la recta que pasa por dicho punto.

De este modo podemos caracterizar las rotaciones mediante puntos, manteniendo el mismo formalismo presentado para las traslaciones. No obstante la utilización de un punto para especificar una rotación tiene limitaciones que no aparecen con las traslaciones: la interpretación de una rotación depende del convenio seguido para expresar los ángulos en coordenadas esféricas, por otra parte para especificar algunas rotaciones será necesario utilizar más de un punto, y además, hay puntos singulares (los polos).

La rotación antes definida es una transformación geométrica.

Lema 2.11: La rotación $T_r^{[P_r]}$ según $P_r \in \mathbf{R}^n$, es una transformación geométrica en $\mathbf{R}^n$.

Demostración:

Trivial. $\square$

Lema 2.12: La rotación $T_r^{[\nu_1]}$ según $\nu_1 \in \mathbf{R}^n$, es la transformación identidad.

Demostración:

Trivial. $\square$

2.4.3 Escalado.

Una transformación de escalado es una transformación geométrica que aplicada a una figura modifica su tamaño. En el escalado todo punto, $P = (x_1,x_2,..,x_n)$, sufre la siguiente transformación en sus coordenadas:

$$T(P) = (s_1 \cdot x_1, s_2 \cdot x_2, .., s_n \cdot x_n)$$

Por tanto, en un escalado, el origen permanece inalterado. El escalado puede introducir una deformación de la figura, ya que los factores de escala de cada coordenada pueden ser distintos.

En notación matricial, un escalado se construye mediante una matriz en la que los elementos de la diagonal principal son los factores de escala s_i, y el resto de los elementos son cero.

La transformación de escalado está perfectamente especificada por los n factores de escala, s_i. Para caracterizar estos factores podemos utilizar al punto cuyas coordenadas coinciden con los factores de escala, es decir:

$$P_s = (s_1,s_2,..,s_n).$$

Los factores de escala de una transformación de escalado, podrán, por tanto, estar especificada mediante un punto.

Definición 2.10: El escalado $T_s^{[P_s]}(P)$ de un punto, $P \in \mathbb{R}^n$, según $P_s \in \mathbb{R}^n$, viene dada por

$$T_s^{[P_s]}(P) = P_s \cdot P$$

interpretándose el producto $P_s \cdot P$ como el punto obtenido por producto componente a componente de los dos puntos. □

El escalado es una transformación geométrica, que, además, considerada como operación binaria en $\mathbb{R}^n$ es conmutativa. Estos resultados quedan establecidos por los siguientes lemas.

Lema 2.13: El escalado $T_s^{[P_s]}$ según $P_s \in \mathbb{R}^n$, es una transformación geométrica en $\mathbb{R}^n$.

Demostración:

Trivial. □

Lema 2.14: Dados dos puntos cualesquiera, P_1 $P_2 \in \mathbf{R}^n$, el escalado de P_1 según P_2 coincide con el escalado de P_2 según P_1, esto es

$$\mathbf{T}_s^{[P_1]}(P_2) = \mathbf{T}_s^{[P_2]}(P_1)$$

y el escalado según el punto $P_u = (1,1,.....,1)$ es la transformación identidad.

Demostración:

Trivial. □

En el caso de que todos los factores de escala sean iguales (punto de especificación del escalado sobre la bisectriz del primer cuadrante), se conservará la componente angular del punto, expresado en coordenadas esféricas, y por tanto sólo se modificará la componente radial. En este caso, y en el supuesto de que se realice un escalado de magnitud s, el punto transformado será

$$T(P) = (s \cdot r, \Phi_1, \Phi_2, .., \Phi_{n-1})$$

2.4.4 Transformación circular.

Una transformación circular se obtiene por composición de una rotación y un escalado. Al igual que el resto de las transformaciones descritas en esta sección, la magnitud de la transformación circular se caracterizará por un punto, lo que permitirá interpretarla como una operación binaria entre puntos.

Definición 2.11: La **trasformación circular** $\mathbf{T}_c^{[P_c]}(P)$ de un punto, $P \in \mathbf{R}^n$, según el punto $P_c \in \mathbf{R}^n$, es la transformación geométrica cuya acción viene dada por la siguiente composición de transformaciones:

$$\mathbf{T}_c^{[P_c]}(P) = \mathbf{T}_s^{[RP_c]}(\mathbf{T}_r^{[P_c]}(P))$$

donde RP_c es el punto cuyas coordenadas son:

$$RP_c = (Rad(P_c), Rad(P_c), .., Rad(P_c))$$ □

Comentarios:

La transformación circular realiza un escalado de forma uniforme en todas direcciones, en magnitud dada por la componente radial del punto de especificación, P_c, y una rotación según P_c.

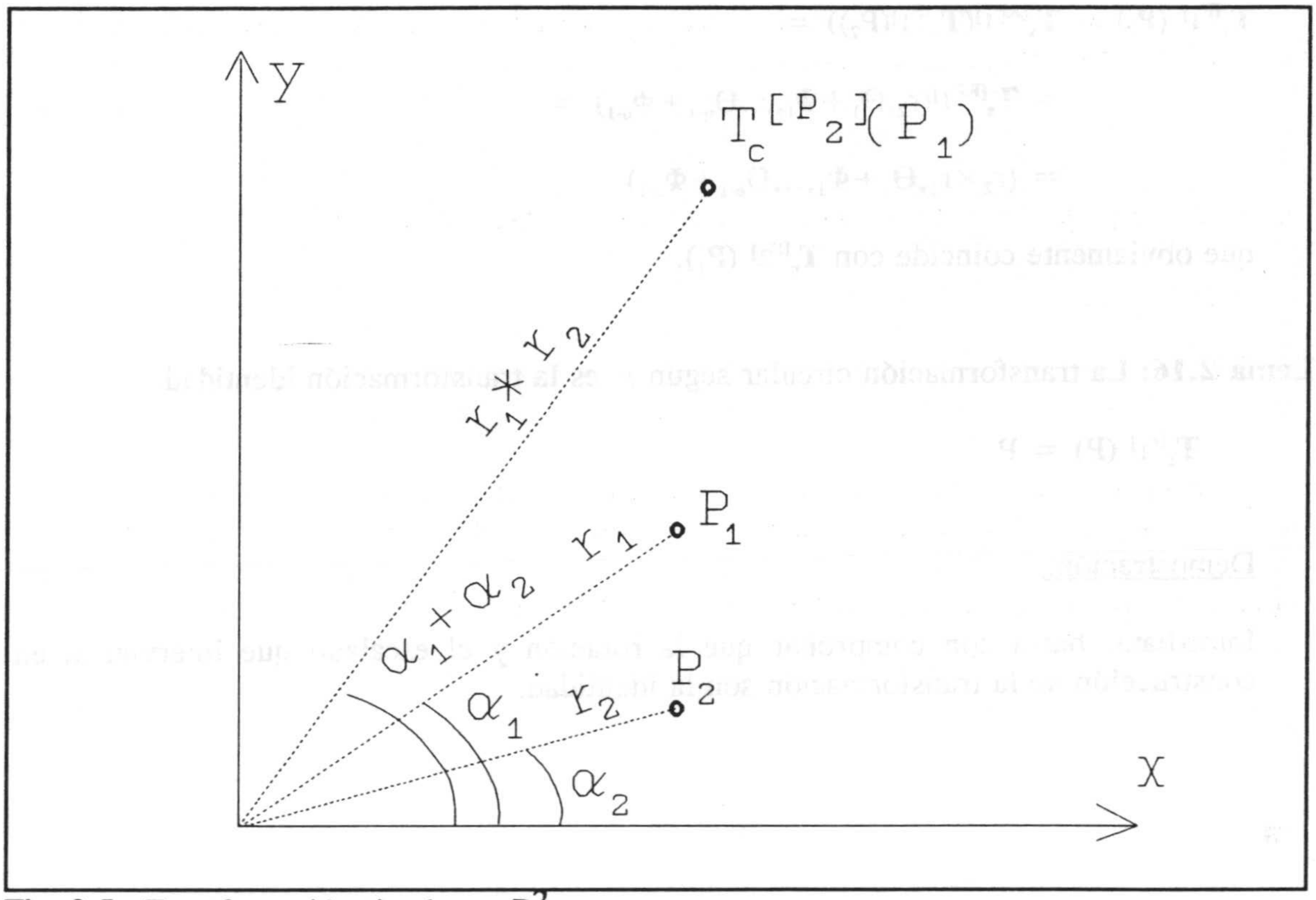


Fig. 2.5. Transformación circular en $\mathbf{R}^2$.

La transformación circular es una transformación geométrica ya que se define como composición de transformaciones geométricas.

Analíticamente la transformación circular de un punto $P_2 = (r_2, \Phi_{21}, \Phi_{22}, .., \Phi_{2,n-1})$, *según el punto* $P_1 = (r_1, \Phi_{11}, \Phi_{12}, .., \Phi_{1,n-1})$, *es el punto dado por (ver fig. 2.5).*

$$T_c^{[P1]}(P_2) = (r_2 * r_1,\ \Phi_{21}+\Phi_{11}, \Phi_{22}+\Phi_{12}, .., \Phi_{2,n-1}+\Phi_{1,n-1})$$

La transformación circular, entendida como operación binaria de dos puntos, es conmutativa.

Lema 2.15: La transformación circular según $P_1 \in \mathbf{R}^n$ de un punto cualquiera, $P_2 \in \mathbf{R}^n$, es igual a la transformación circular de magnitud P_2 del punto P_1

$$\forall\ P_1, P_2 \in \mathbf{R}^n \quad T_c^{[P1]}(P_2) = T_c^{[P2]}(P_1)$$

Demostración:

Sean $P_1=(r_1,\Phi_1,..,\Phi_{n-1})$ y $P_2=(r_2,\Theta_1,..,\Theta_{n-1})$ dos puntos cualesquiera del espacio, expresados en coordenadas esféricas. La transformación circular de P_1 con magnitud P_2 es

$$\mathbf{T}_c^{[P1]}(P_2) = \mathbf{T}_s^{[RP1]}(\mathbf{T}_r^{[P1]}(P_2)) =$$

$$= \mathbf{T}_s^{[RP1]}(r_2, \Theta_1 + \Phi_1, .., \Theta_{n-1} + \Phi_{n-1}) =$$

$$= (r_2 \times r_1, \Theta_1 + \Phi_1, .., \Theta_{n-1} + \Phi_{n-1})$$

que obviamente coincide con $\mathbf{T}_c^{[P2]}(P_1)$. □

Lema 2.16: La transformación circular según ν_1 es la transformación identidad.

$$\mathbf{T}_c^{[\nu 1]}(P) = P$$

Demostración:

Inmediata, basta con comprobar que la rotación y el escalado que intervienen en la construcción de la transformación son la identidad. □

2.4 FAMILIAS DE OBJETOS.

Se ha indicado previamente que, un objeto gráfico se caracteriza, entre otras cosas, por su posición y orientación. Esto es, dos mesas pueden ser objetos distintos sólo por el hecho de estar colocadas en sitios distintos. Sin embargo hay situaciones en las que interesa prescindir de este hecho diferenciador de los objetos, considerando en una misma categoría a todas las mesas que tiene la misma forma, independientemente de la posición en que se encuentren. Esto ocurre, por ejemplo, cuando:

· Interesa comprobar si dos objetos dados se diferencian tan sólo en su colocación en el espacio, es decir se pueden considerar como copia el uno del otro.

· Se desee utilizar distintos ejemplares de un mismo tipo. Es decir, disponer de un catalogo de objetos que después se puedan utilizar reiteradas veces.

Por este motivo se introduce el concepto de familia de objetos, que se corresponde con la idea anteriormente expuesta.

Las familias se definirán en base a una relación de equivalencia definida entre los objetos gráficos, que denominaremos equivalencia geométrica. Dos objetos serán geométricamente equivalentes cuando se pueda obtener el uno a partir del otro aplicando una transformación geométrica. Obsérvese que las transformaciones geométricas pueden realizar deformaciones (p.e. deslizamiento).

Definición 2.12: Diremos que dos objetos son **geométricamente equivalentes**, y lo representaremos como $O_1 \propto O_2$, cuando exista una transformación geométrica, T, tal que $O_1 = T(O_2)$. □

Comentarios:

En la práctica, el hecho de que dos objetos sean geométricamente equivalentes permite la generación de uno a partir del otro. Esto permitirá el almacenar un determinado objeto, como definición de un símbolo, para posteriormente generar objetos gráficos geométricamente equivalentes como instancias de aquel.

Teorema 2.5: La equivalencia geométrica entre objetos es una relación de equivalencia en el conjunto de objetos.

Demostración:

La relación es reflexiva, ya que $\forall O \in \mathbf{O} => O \propto O$, ya que $\hat{1}(O) = O$. Obviamente

la relación es simétrica, dado que las transformaciones geométricas poseen inversa. Para ver que la relación es transitiva basta con considerar que la composición de transformaciones geométricas es una transformación geométrica (§Teorema 2.4). □

La relación de equivalencia geométrica induce una partición en el conjunto de objetos. Cada clase de equivalencia será considerada como una familia de objetos.

Definición 2.13: Denominaremos **familia de objetos** a cada clase de equivalencia inducida por la relación de equivalencia geométrica en el conjunto de objetos. □

Definición 2.14: Denominaremos **conjunto de familias de objetos**, y lo representaremos por **F**, al conjunto cociente del conjunto de objetos respecto a la relación de equivalencia geométrica. □

El concepto de familia es transcendental, pues permite sistematizar la gestión de los objetos gráficos, al posibilitar que los objetos de una familia se generen por transformación geométrica de otros objetos de la misma familia. Por tanto será necesario almacenar tan sólo la definición completa de un sol objeto gráfico de cada familia, junto con la transformación geométrica concreta a aplicar para generar cada objeto usado.

Ejemplo 2.6: Todos los segmentos de recta que tengan el mismo aspecto y presencia pertenecen a la misma familia, ya que es posible obtener un segmento a partir de otro cualquiera aplicando una transformación geométrica.

Por tanto para utilizar una familia de líneas bastaría con almacenar un único objeto gráfico, que podría por ejemplo ser:

$$\text{Línea}(\nu_0,\nu_1) = (\ V,\ \mu,\ \alpha\ );$$

siendo: $V = \{\ P \mid \exists\lambda\in[0,1] \subset \mathbf{R}\ \ P = \lambda\cdot\nu_0 + (1-\lambda)\nu_1\ \} \subset \mathbf{R}^n$

$$\mu(P) = \begin{cases} 1 \text{ si } P\in V \\ 0 \text{ si } P\notin V \end{cases}$$

$$\alpha(P) = c\cdot\mu(P)$$

Las distintas líneas que se definan pueden crearse a partir de esta indicando la anterior, tan solo, en la transformación geométrica, y por tanto sólo será necesario indicar dicha transformación para definir la línea.

Así, por ejemplo, las siguientes líneas se especifican dando las transformaciones:

para Línea(ν_0,$2\cdot\nu_1$) => $\tau = T_s^{[2]}$

para Línea(ν_0,ν_2) => $\tau = T_r^{[r2]}$

Por el momento, para manipular todas las líneas posibles sólo debemos de considerar varias familias de líneas, que se diferenciarán unas de otras en la presencia y el color. Más adelante veremos que es posible obtener todas estas familias, y por tanto todas las posibles líneas, a partir de una única familia utilizando filtros.

Al definir el concepto de familia de objetos hemos prescindido, entre otras, de la información de posición que contenían los objetos. Si queremos seleccionar un objeto como representante privilegiado de cada familia, será necesario prefijar la ubicación de dicho objeto. Una vez seleccionado dicho elemento se construirían todos los objetos de la misma familia realizando transformaciones geométricas.

Sin embargo el problema inverso, es decir determinar si dos objetos pertenecen a la misma familia, no es fácil de resolver en la práctica, aún en el caso de haber trabajado a partir de un representante de la clase, dado que una vez evaluada la transformación geométrica se pierde la posibilidad de identificar al representante de la clase que la ha generado. Para poder resolver de una forma simple dicho problema será necesario introducir formalmente el concepto de instancia.

2.5 INSTANCIAS GRAFICAS.

En esta sección se va a definir formalmente el concepto de instancia gráfica (que usualmente denominaremos simplemente instancia). Se ha visto que el concepto de objeto gráfico es una abstracción de un sistema, que contiene toda la información necesaria para gestionar la información gráfica del mismo. Una instancia gráfica es, asimismo una abstracción gráfica de un sistema. Las instancias poseen la misma estructura que los objetos, pero añaden una transformación geométrica. Dicha transformación se interpretará como una información de colocación, que determina su posición y orientación, esto es, la instancia se define según un determinado sistema de referencia, indicando la presencia y aspecto de cada uno de sus puntos, y la transformación geométrica indica en que parte del espacio se sitúa la instancia. De este modo, en la instancias gráficas, se pretende desglosar la información de la estructura, de la de colocación en el espacio. Indudablemente, para definir la estructura geométrica de la instancia se podrán utilizar distintos sistemas de referencia, haciendo que, en consecuencia, la transformación geométrica cambie. Por tanto, la información contenida en un objeto gráfico y en una instancia gráfica son la misma, la diferencia está tan sólo en la forma de expresarla. Por este motivo, podremos usar, indistintamente instancias u objetos gráficos para representar abstracciones gráficas.

Desde este punto de vista, podremos hablar de las instancias con el mismo contenido de información que un objeto gráfico ('sinónimas'), entendiendo de este modo las instancias que son abstracción del mismo sistema del que lo es el objeto gráfico en cuestión. Así, por ejemplo, al definir la estructura de un cierto segmento de recta, se dará la posición de sus dos extremos. Dependiendo de donde los 'situemos' en dicha definición, la transformación geométrica a aplicar será distinta, supuesto que se desea describir el mismo objeto.

Con esta separación conceptual entre estructura y colocación, se consiguen dos objetivos importantes:

· Se facilita[1] la implementación de sistemas gráficos, ya que la realización de transformaciones geométricas es frecuente y fácil en la práctica.

· Al aislar la estructura de la instancia de su ubicación, se facilita la caracterización de todos los elementos de una misma familia.

Una instancia es una abstracción gráfica de un sistema cualquiera, en cuya definición intervienen cuatro factores: dominio, aspecto, presencia y transformación. Los tres primeros constituyen la definición de la estructura de la instancia. El **dominio** de la instancia es el volumen del espacio en el que **se define** la instancia. Dicho dominio no tiene por qué coincidir con el volumen del sistema al que representa. El **aspecto** da información de la apariencia visual de cada punto del dominio. La **presencia** indica para qué puntos del dominio existe realmente 'materia' y, por tanto, tiene sentido definir una representación visual. En conjunto, el dominio, el aspecto y la presencia definen la estructura del gráfica del sistema representado.

---

[1] De hecho la utilización de instancias como parte de un modelo gráfico es una técnica usada habitualmente.

La **transformación geométrica** indica en qué punto del espacio se debe **ubicar** cada punto del dominio de definición.

Definición 2.15: Una **instancia gráfica** es una cuádrupla (D,τ,μ,α), en la que:

D es un volumen de $\mathbf{R}^n$, que denominaremos volumen de definición de la instancia o simplemente **dominio** de la instancia.
τ es una transformación geométrica.
μ es una función que denominaremos **función de presencia**, (o presencia) definida como: μ: $\mathbf{R}^n \rightarrow \mathbb{P}$, siendo $\mathbb{P}$ un dominio de presencia.
α es una función que denominaremos **función de aspecto**, (o presencia) definida como α: $\mathbf{R}^n \rightarrow \mathbf{A}$, siendo **A** un dominio de aspecto.

que cumple

$$\forall P \in \mathbf{R}^n\text{-}D \Rightarrow \alpha(P) = 0 \quad \wedge \quad \mu(P) = 0$$ □

Comentarios:

Obsérvese que la condición impuesta en esta definición es más débil que la impuesta en la definición de objeto gráfico, ya que aquí no se exige que para $P \in D$ tenga que ser $\alpha(P) \neq 0 \vee \mu(P) \neq 0$. De esta forma la definición de instancias se hace más flexible, a costa de la unicidad de la representación.

*Conceptualmente, usar una transformación τ como parte de la información de la instancia, es equivalente a tener dos sistemas de coordenadas, uno en el que **se describe** la instancia, en el que está su dominio de definición, D, y otro en el que **se sitúa** la instancia, que contiene a $\tau(D)$.*

Como consecuencia de lo expuesto anteriormente, la transformación τ se aplicará a la instancia antes de visualizarla o interpretar sus coordenadas geométricas. Esto es, las funciones que forman la instancia están definidas en D, pero la instancia hace referencia a un objeto que se encuentra en $\tau(D)$, por lo que la visualización de un punto $\tau(P) \in \tau(D)$ habrá de hacerse teniendo en cuenta los valores de $\mu(P)$ y $\alpha(P)$ en $P \in D$. El haber considerado como parte de la definición de la instancia la transformación, facilita la edición y el tratamiento de transformaciones geométricas.

Denotaremos por Ω al conjunto de instancias de objetos gráficos.

Ejemplo 2.7: Antes de proseguir veamos como se pueden definir instancias de algunos de los objetos gráficos simples definidos en el ejemplo anterior. Supongamos para ello el dominio de aspecto $\mathbf{C}_{rgb}$ y el dominio de presencia $\mathbf{P}_n$.

Punto

$$\text{Punto}(P_0) = (\ V,\ \tau,\ \mu,\ \alpha\ );$$

donde:

$$V = \{\ \nu_0\ \} \subset \mathbf{R}^n$$

$$\mu(P) = \begin{cases} 1 \text{ si } P = \nu_0 \\ 0 \text{ si } P \neq \nu_0 \end{cases}$$

$$\tau(P) = P + P_0$$

$$\alpha(P) = k \cdot \mu(P)$$

línea

$$\text{línea}(P_1,P_2) = (\ D,\ \hat{\imath},\ \mu,\ \alpha\ );$$

siendo: $D = \{\ u \cdot P_1 + (u\text{-}1) \cdot P_2 \ \mid\ u \in [0,1]\ \ P_1,P_2 \in \mathbf{R}^3\}$

$$\mu(P) = \begin{cases} 1 \text{ si } P \in D \\ 0 \text{ si } P \notin D \end{cases}$$

$$\alpha(P) = k \cdot \mu(P)$$

Notese que en este caso se ha hecho coincidir la transformación geométrica con la transformación identidad. Por tanto, en este caso, D y τ(D) coincidirán. Podríamos haber optado por utilizar otro sistema de referencia para la instancia, y así, por ejemplo, hacer que P_1 esté en el origen del dominio. En este caso sería:

$$\text{línea}(P_1,P_2) = (\ D,\ \tau,\ \mu,\ \alpha\ );$$

siendo: $D = \{\ (u\text{-}1) \cdot (P_2\text{-}P_1) \ \mid\ u \in [0,1]\ \ P_1,P_2 \in \mathbf{R}^3\}$

$$\tau(P) = P + P_1$$

$$\mu(P) = \begin{cases} 1 \text{ si } P \in D \\ 0 \text{ si } P \notin D \end{cases}$$

$$\alpha(P) = k \cdot \mu(P)$$

Debe observarse el hecho importante de que manteniendo la definición de D, μ y α, y variando tan sólo τ, se puede generar cualquier línea. En general se generarán todos los objetos que se pueden obtener por transformación geométrica.

Al no forzar la definición de instancia a que en todos los puntos del dominio de la instancia sean la presencia o aspectos distintos de los respectivos elementos neutros, pueden existir puntos dentro del dominio que no contengan información relevante (es decir con presencia y aspectos nulos). Para designar a los puntos del dominio de la instancia que contienen información relevante se introduce el concepto de dominio efectivo.

Definición 2.16: Denominaremos **dominio efectivo** de una instancia al conjunto de puntos de $\mathbf{R}^n$ en los que no son simultáneamente nulas μ y α:

$$D_{ef} = \{ P \in D \mid \mu(P) \neq 0 \vee \alpha(P) \neq 0 \}$$ □

Podemos considerar que el dominio efectivo de la instancia es el mínimo volumen que contiene información relevante sobre la instancia.

Hasta aquí, se han establecido dos representaciones distintas de los sistemas gráficos (los objetos y las instancias). Es necesario plantearse que relación existe entre ambas. Con este propósito comenzaremos caracterizando subconjuntos de instancias que representan a un mismo sistema.

Definición 2.17: Dos instancias son **sinónimas** o **efectivamente equivalentes**, $R_1 \cong R_2$, si cumplen que:

· las imágenes mediante sus transformaciones geométricas de sus dominios eficaces coinciden:

$$\tau_1(D_{ef1}) = \tau_2(D_{ef2})$$

· las funciones de aspecto y presencia satisfacen la siguiente condición:

$$\forall P \in \tau_1(D_{ef1}) \; \exists P_1 \in D_1, P_2 \in D_2 \mid \tau_1(P_1) = \tau_2(P_2) = P \wedge$$

$$\alpha_1(P_1) = \alpha_2(P_2) \wedge \mu_1(P_1) = \mu_2(P_2)$$ □

La definición establece que dos instancias son efectivamente equivalentes cuando se sitúan en el mismo volumen del espacio, y para cada punto de ese volumen, ambas asignan el mismo valor al aspecto y a la presencia.

Demostraremos, a continuación, que la equivalencia efectiva entre instancias es una relación de equivalencia en el conjunto de instancias. Esta relación se usará más adelante como puente entre las instancias y los objetos.

Proposición 2.1: La equivalencia efectiva entre instancias es una relación de equivalencia en el conjunto de instancias, Ω.

Demostración:

Para que la equivalencia efectiva sea una relación de equivalencia debe de ser reflexiva, simétrica y transitiva. Demostraremos cada una de estas propiedades por separado:

Es evidente que la relación es reflexiva ya que $\forall R \in \Omega => R \cong R$. Del mismo modo la relación es simétrica dado que su definición está basada en la identidad sobre $\mathbf{R}^n$, **A** y **P**, por tanto $\forall R1,R2 \in \Omega$ $R1 \cong R2 => R2 \cong R1$. Para comprobar que la relación es transitiva basta con encadenar las identidades en las dos definiciones de igualdad, con lo que se prueba que $\forall R1,R2,R3 \in \Omega$ $R1 \cong R2$ y $R2 \cong R3 => R1 \cong R3$. □

No se debe confundir la relación de equivalencia, antes definida, con la identidad en el conjunto de instancias (identidad de los cuatro elementos de la cuádrupla), que se denotará por el símbolo habitual de identidad, $=$.

Pueden existir instancias efectivamente equivalentes con distintas funciones τ, μ y α. En estos casos se cumplirá siempre que las composiciones de las inversas de las transformaciones con las funciones de presencia, por un lado, y con las funciones de intensidad, por otro, sean iguales (en el sentido de asignar los mismos valores a cada punto del volumen) en las dos instancias.

De entre las distintas instancias que pueden ser efectivamente equivalentes destacaremos una, a la que denominaremos normal. Intuitivamente podemos entender a dicha instancia como aquella en la que se utiliza el mismo sistema de referencia para definir la estructura de la instancia y para colocarla.

Definición 2.18: Diremos que una **instancia** es **normal**, si cumple que su Dominio D es igual a su Dominio efectivo y su transformación τ es la identidad. □

Por tanto, en una instancia normal, coinciden el dominio y el dominio efectivo.

Teorema 2.6: En cada clase de equivalencia efectiva existe una instancia normal y es única.

Demostración:

1. Existencia de instancia normal.

Sea $R = (D, \tau, \mu, \alpha)$ un elemento de una clase de equivalencia efectiva cualquiera. Construyamos la instancia:

$R_N = (D_N, \hat{\imath}, \mu_N, \alpha_N)$

con $\mu_N = \mu\, \tau^{-1}$

$$\alpha_N = \alpha\, \tau^{-1}$$

$$D_N = \{ P \in \mathbb{R}^n \mid \mu_N(P) \neq 0 \text{ ó } \alpha_N(P) \neq 0 \} \subseteq \tau(D)$$

que según la definición anterior es una instancia normal. Demostraremos que esta instancia, R_N, obtenida por la expresión anterior es efectivamente equivalente a R, con lo que se habrá demostrado que en la clase de equivalencia de cualquier instancia existe una instancia normal.

Comencemos viendo que las transformaciones de los dominios efectivos son iguales, es decir que:

$$\tau\,(D_{ef}) = \hat{\imath}\,(D_N) = D_N$$

donde D_{ef} es el dominio efectivo de R.

Ambos dominios están definidos del siguiente modo:

$$D_N = \{ P \in \mathbb{R}^n \mid \mu\,\tau^{-1}(P) \neq 0 \text{ ó } \alpha\,\tau^{-1}(P) \neq 0 \}$$

$$\tau(D_{ef}) = \{ P \in \mathbb{R}^n \mid \exists\, P' \in D,\ \tau(P') = P \text{ y } (\mu(P') \neq 0 \text{ ó } \alpha(P') \neq 0) \}$$

Para demostrar la igualdad de ambos conjuntos, mostraremos que cualquier elemento de uno pertenece al otro:

si $Q \in D_N \Rightarrow \mu\,(\tau^{-1}(Q)) \neq 0$ ó $\alpha\,(\tau^{-1}(Q)) \neq 0 \Rightarrow$

$\Rightarrow \exists\, Q' = \tau^{-1}(Q) \in D$, tal que

$\mu(Q') \neq 0$ ó $\alpha(Q') \neq 0$

$\Rightarrow Q' \in D_{ef}$ y $Q \in \tau\,(D_{ef})$

si $Q \in \tau(D_{ef}) \Rightarrow \exists\, Q' \in D$ con $\tau(Q') = Q$

y

$\mu(Q') \neq 0$ ó $\alpha(Q') \neq 0 \Rightarrow \mu\,(\tau^{-1}(Q)) \neq 0$ ó $\alpha\,\tau^{-1}(Q) \neq 0 \Rightarrow$

$\Rightarrow Q \in D_n$ ya que $Q' = \tau^{-1}(Q)$

Así pues, los volúmenes de ambas instancias son iguales.

Veamos ahora que las funciones μ y α satisfacen la segunda condición para que exista equivalencia efectiva. Se ha probado que

$$V = D_N = \tau\,(D_{ef})$$

Sea $P \in V$ un punto arbitrario, construyamos los puntos:

$$P_1 = \tau^{-1}(P) \in D_{ef}$$

$$P_2 = P \in D_N$$

Obviamente se cumple que $\tau(P_1) = \hat{\imath}(P_2) = P$

Los valores de las funciones de aspecto en P_1 y P_2 son:

$$\alpha(P_1) = \alpha(\tau^{-1}(P))$$

$$\alpha_N(P_2) = \alpha(\tau^{-1}(P))$$

y por tanto $\alpha_1(P_1) = \alpha_2(P_2)$

Del mismo modo

$$\mu(P_1) = \mu(\tau^{-1}(P))$$

$$\mu_N(P_2) = \mu(\tau^{-1}(P))$$

y por tanto $\mu_1(P_1) = \mu_2(P_2)$

quedando demostrado que ambas instancias satisfacen la relación de equivalencia efectiva, y que, por lo tanto existe al menos una instancia normal para cada clase de equivalencia efectiva.

2. La instancia normal de cada clase es **única**:

Sean $R_1 = (D_{ef1}, \tau_1, \mu_1, \alpha_1)$ y $R_2 = (D_{ef2}, \tau_2, \mu_2, \alpha_2)$ dos instancias normales cualesquiera de la misma clase de equivalencia efectiva. Vamos a probar que

$$D_{ef1} \equiv D_{ef2}\ ;\ \mu_1 \equiv \mu_2\ ;\ \alpha_1 \equiv \alpha_2$$

con lo que quedará demostrada la unicidad de la instancia normal para clase de equivalencia efectiva.

Por ser ambas instancias normales se cumple que:

$$\tau_1 = \tau_2 = \hat{1}$$

Por ser R_1 y R_2 instancias pertenecientes a una misma clase de equivalencia, y por tener ambas a la identidad como transformación, se deduce que:

$$D_{ef1} \equiv D_{ef2} \equiv V$$

Por otra parte, por pertenecer R_1 y R_2 a la misma clase de equivalencia efectiva, se debe de cumplir que:

$$\forall P \in V \; \exists \; P_1 \in D_{ef1}, P_2 \in D_{ef2} \mid \tau_1(P_1) = \tau_2(P_2) = P \; ; \; \alpha_1(P_1) = \alpha_2(P_2)$$

$$\text{y } \mu_1(P_1) = \mu_2(P_2)$$

Ahora bien, puesto que $\tau_1 = \tau_2 = \hat{1}$, entonces $P_1 = P_2 = P$ y podemos escribir que

$$\forall P \in D_{ef1} \equiv D_{ef2} \equiv V \Rightarrow \alpha_1(P) = \alpha_2(P) \text{ y } \mu_1(P) = \mu_2(P)$$

lo que completa la demostración. □

Colorario 2.2: Para cada instancia, $R=(D,\tau,\mu,\alpha)$, existe una única instancia normal efectivamente equivalente a ella, dado por la expresión:

$$R_N = (D_N,\tau_N,\mu_N,\alpha_N) = (D_{ef}, \hat{1}, \mu\,\tau^{-1}, \alpha\,\tau^{-1})$$

siendo D_{ef} el dominio efectivo de R.

Demostración:

Inmediata una vez demostrada la existencia y unicidad de instancia normal para cualquier clase de equivalencia efectiva. □

Tal y como se ha visto, dos instancias de objetos pueden ser efectivamente equivalentes aunque sus dominios de definición sean diferentes. Del mismo modo, dadas dos instancias R_1 y R_2 distintas, siempre se puede encontrar un par de instancias R_1' y R_2', efectivamente equivalentes a las primeras, y que posean ambas el mismo dominio de definición. Este dominio podría ser, por ejemplo, la unión de los dominios de las dos instancias, y en cualquier caso, deberá contener a los dominios de definición de R_1 y R_2.

Ejemplo 2.7: Consideremos las instancias de objetos del ejemplo anterior. Podemos obtener instancias equivalentes a las allí expuestas modificando D,τ o μ. De este modo se pueden obtener las siguientes instancias, efectivamente equivalentes a las del ejemplo 2.4:

Punto

$$\text{Punto}(P_0) = (D, \hat{\imath}, \mu, \alpha);$$

donde: $D = \{ P_0 \} \subset \mathbf{R}^n$

$$\mu(P) = \begin{cases} 1 \text{ si } P = P_0 \\ 0 \text{ si } P \neq P_0 \end{cases}$$

$\alpha(P) = k \cdot \mu(P)$ con k color del punto

línea

$$\text{línea}(P_1,p_2) = (D, \tau, \mu, \alpha);$$

siendo: $D = \{ P \mid P = u\cdot(0,..,0)+(1-u)\cdot(1,0,..0)\ u\in[0,1] \} \subset \mathbf{R}^n$

$$\mu(P) = \begin{cases} 1 \text{ si } P \in D \\ 0 \text{ si } P \notin D \end{cases}$$

$\alpha(p) = k\cdot\mu(P)$

τ: $\mathbf{R}^n$ -> $\mathbf{R}^n$ | $\tau(0,..0) = P_1$ y $\tau(1,0,..0) = P_2$

Una vez establecido el concepto de objeto gráfico y de instancia de objeto gráfico, es necesario establecer la relación entre ambos. En este sentido se introduce el concepto de instancias sinónimas de un objeto. Entenderemos que una instancia es sinónima de un objeto cuando es abstracción del mismo sistema que el objeto.

Definición 2.19: Diremos que una instancia, $R = (D,\tau,\mu_R,\alpha_R)$, es **sinónima** de un objeto, $O = (V,\mu_o,\alpha_o)$, y lo denotaremos por $R \mathcal{L} O$, si se cumplen las siguientes condiciones:

a) $V = \tau(D_{ef})$

b) $\forall P \in D_{ef}\ \exists P' \in V \mid P' = \tau(P) \wedge \alpha_R(P) = \alpha_o(P') \wedge \mu_R(P) = \mu_o(P')$ □

Veremos que para cada objeto existen instancias que son sinónimas de él, y que todas ellas son efectivamente equivalentes.

Proposición 2.2: Para cada objeto, $O = (V,\mu_o,\alpha_o)$, existe al menos una instancia, R, que es sinónima de él.

Demostración:

Sea $O = (V,\mu_o,\alpha_o)$, construyamos la instancia normal

$$R = (V,\hat{\imath},\mu_o,\alpha_o)$$

su dominio efectivo coincide con V, y por tanto se cumple la condición a). Resulta obvio comprobar que se cumple la segunda condición de la definición. □

Teorema 2.7: Todas las instancias sinónimas de un mismo objeto, $O = (V,\mu_o,\alpha_o)$, son efectivamente equivalentes.

Demostración:

Sea la instancia $R_o = (D_0,\hat{\imath},\mu_o,\alpha_o)$ sinonima del objeto $O = (V,\mu_o,\alpha_o)$, supongamos que la instancia $R_1 = (D_1,\tau,\mu_1,\alpha_1)$ es también sinonima del mismo objeto. Demostraremos que R_o y R_1 son efectivamente equivalentes.

En efecto, el que ambas instancias sean sinónimas de un mismo objeto implica el que $V = \tau(D_{1ef}) = \hat{\imath}(D_{oef})$ que es la primera condición para que ambas instancias sean efectivamente equivalentes. Por otra parte

$$\forall P \in D_{1ef} \;\; \exists P' \in V \mid P' = \tau(P) \; ; \; \alpha_R(P) = \alpha_o(P')$$

$$\text{y} \quad \mu_R(P) = \mu_o(P')$$

que junto con la expresión anterior implica la segunda condición requerida. □

Cada objeto será sinónimo de diversas instancias, siendo todas ellas efectivamente equivalentes. Para representar matemáticamente a un objeto podremos usar cualquiera de dichas instancias. En determinados casos sustituiremos los objetos por instancias, para realizar determinadas operaciones. Siempre, en estos casos, se habrá demostrado que la elección de la instancia no altera el resultado de la operación.

Teorema 2.8: Dado un objeto cualquiera, $O = (V,\mu_o,\alpha_o)$, la instancia normal que es sinónima de él esta dada por la expresión:

$$R = (V,\hat{\imath},\mu_o,\alpha_o)$$

Demostración:

Trivial, a partir de los resultados del teorema 2.7 y proposición 2.2. □

2.5.1 Transformación geométrica de instancias.

En esta sección se definen las transformaciones geométricas de instancias. Dado que las transformaciones geométricas se podrán aplicar a los objetos y a las instancias. Se demostrará que el efecto producido sobre ambas es equivalente.

Definición 2.20: La transformación geométrica, T, de una instancia, R, es una instancia, que denotaremos por T(R), dada por:

$$T(R) = (D, T\cdot\tau, \mu, \alpha)$$

Donde $T\cdot\tau$ indica la aplicación de τ y posteriormente de T, esto es, la composición funcional de ambas transformaciones. □

Para que la definición anterior pueda ser congruente con la definición de transformación geométrica de objetos es necesario que dicha transformación sea estable para la relación de equivalencia efectiva. Este hecho queda establecido en la siguiente proposición.

Proposición 2.3: La transformación geométrica de instancias es estable para la relación de equivalencia efectiva.

Demostración:

Dadas $R_1 \cong R_2$, y una transformación geométrica T, demostraremos que $T(R_1) \cong T(R_2)$
Sea

$$R_1 = (D, \tau, \mu, \alpha)$$

y R_2 la instancia normal efectivamente equivalente a R_1:

$$R_2 = (D_2, \tau_2, \mu_2, \alpha_2) = (D_2, \hat{\imath}, \mu\cdot\tau^{-1}, \alpha\cdot\tau^{-1}) \text{ con } D_2 = D_{ef}$$

La transformación de ambas será:

$$T(R_1) = (D,\ T\cdot\tau,\ \mu,\ \alpha)$$

$$T(R_2) = (D_2,\ T,\ \mu\cdot\tau^{-1},\ \alpha\cdot\tau^{-1})$$

que son efectivamente equivalentes, ya que la instancia normal equivalente a cada una de ellas es:

$$T(R_1) \cong (D_{ef},\ \hat{\imath},\ \mu\cdot\tau^{-1}\cdot T^{-1},\ \alpha\cdot\tau^{-1}\cdot T^{-1}) \cong T(R_2)$$

lo que completa la demostración. □

Teorema 2.9: La transformación geométrica, T(O), de un objeto, O = (V,μ,α), es sinonima de la transformación geométrica de cualquiera de las instancias sinónimas de O.

Demostración:

La instancia normal sinonima del objeto O es R=(V,î,μ,α), y T(O) = (T(V),$\mu\cdot T^{-1}$,$\alpha\cdot T^{-1}$) será sinónimo de la instancia normal

$$R' = (T(V),\hat{\imath},\mu\cdot T^{-1},\alpha\cdot T^{-1})$$

que es efectivamente equivalente a

$$T(R) = (V,T,\mu,\alpha)$$ □

Colorario 2.4: El objeto resultante de la transformación geométrica, T, de un objeto O = (V,μ,α) es sinónimo de la instancia

$$T(O) = (V,T,\mu,\alpha)$$

Demostración:

Trivial. □

Comentarios:

Este resultado permite interpretar la instancia R = (D,T,μ,α) como el resultado de aplicar la transformación geométrica T al objeto O = (D_{ef},μ,α), coincidiendo con la significado habitual de instancia.

Por tanto la diferencia entre instancias y objetos está en la existencia de una transformación geométrica no evaluada en las instancias.

2.5.2 Caracterización de la equivalencia geométrica de objetos.

En la sección 2.4 se expuso la dificultad práctica de determinar si dos objetos pertenecen a la misma familia, aún en el caso de haber trabajado a partir de un representante de la clase. En está sección se realizará una caracterización de la equivalencia geométrica de objetos basada en las instancias, que permitirá resolver esta cuestión.

Lema 2.17: Dos objetos son geométricamente equivalentes si y sólo si existen dos instancias R_1 y R_2, sinónimas respectivamente de O_1 y O_2, y una transformación geométrica T, tales que:

$$R_1 = T(R_2)$$

Demostración:

Sean $O_1 = (V_1,\mu_1,\alpha_1)$ y $O_2 = (V_2,\mu_2,\alpha_2)$

1. Veamos primero que $O_1 \propto O_2 => \exists\, T,R_1 \simeq O_1,R_2 \simeq O_2 \mid R_1 = T(R_2)$

En efecto, si $O_1 \propto O_2$ entonces $\exists T \mid O_1 = T(O_2)$ por tanto

$O_1 = (V_1,\mu_1,\alpha_1) = T(O_2) = (T(V_2),\mu_2 \cdot T^{-1},\alpha_2 \cdot T^{-1})$

con lo que $V_1 = T(V_2)$; $\mu_1 = \mu_2 \cdot T^{-1}$; y $\alpha_1 = \alpha_2 \cdot T^{-1}$

Las instancias normales sinónimas de O_1 y O_2 son

$R_1 = (V_1,\hat{\imath},\mu_1,\alpha_1)$ $\quad R_2 = (V_2,\hat{\imath},\mu_2,\alpha_2)$

y obviamente $T(R_2) = R_1$

2. Demostraremos ahora que si $\exists\, T,R_1 \in O_1,R_2 \in O_2 \mid R_1 = T(R_2) => O_1 \propto O_2$

Sean R_1 y R_2 tales que $R_1 = T(R_2)$, las instancias normales efectivamente equivalentes a ellas son

$R'_1 = (D_1,\hat{\imath},\mu_1,\alpha_1)$ y $R'_2 = (D_2,\hat{\imath},\mu_2,\alpha_2)$

que deben cumplir igualmente que $R'_1 = T(R'_2)$ por ser las transformaciones estables para la equivalencia efectiva. Por tanto

$R'_1 = (D_1,\hat{\imath},\mu_1,\alpha_1) = T(R_2) = (T(D_2),\hat{\imath},\mu_2 \cdot T^{-1},\alpha_2 \cdot T^{-1})$

y como $O_1 = (D_1,\mu_1,\alpha_1)$ y $O_2 = (D_2,\mu_2,\alpha_2)$

se cumple que $O_1 = (D_1,\mu_1,\alpha_1) = T(O_2) = (T(D_2),\mu_2 \cdot T^{-1},\alpha_2 \cdot T^{-1})$

con lo que queda demostrado el lema. □

Como consecuencia inmediata del lema anterior, dos objetos serán geométricamente equivalente, y por tanto pertenecerán a la misma familia, si son sinónimas de instancias que se diferencian solo en la transformación geométrica. Este hecho queda establecido por el siguiente teorema.

Teorema 2.10: Sean O_1 y O_2 dos objetos cualesquiera. O_1 y O_2 pertenecen a la misma familia si y sólo si existe un par de instancias, $R_1 = (D_1,\tau_1,\mu_1,\alpha_1)$ perteneciente a O_1 y $R_2 = (D_2,\tau_2,\mu_2,\alpha_2)$ perteneciente a O_2, tales que

$$D_1 = D_2 \quad \mu_1 = \mu_2 \quad \alpha_1 = \alpha_2$$

Demostración:

Trivial. Basta considerar el resultado del teorema 2.10 y el hecho de que R_1 y R_2 se pueden obtener la una por transformación de la otra ($T=\tau_1\cdot\tau_2^{-1}$). □

2.6 CONCLUSIONES

En este capítulo se han introducido los conceptos de objeto gráfico e instancia gráfica, como dos representaciones alternativas para los sistemas gráficos, y se ha mostrado la equivalencia existente entre ambas.

Como soporte de estas representaciones, se han definido los conceptos de aspecto, presencia y transformación geométrica. El aspecto se ha utilizado para modelar la apariencia visual de los objetos. La presencia se utiliza para modelar la información de ocupación del espacio como una noción cuantitativa. Tanto el aspecto como la presencia se dotan de una estructura algebraica de espacio vectorial y álgebra de boole. La existencia de dicha estructura provee de un mecanismo para calcular valores por composición, y no implica, en el caso concreto del aspecto, una relación con la combinación perceptual.

Como transformaciones geométricas se han considerado solo las que se suelen usar en modelado, es decir excluidas las proyecciones y perspectivas. La definición que se da es simple, pero consistente con la práctica.

A partir de la equivalencia de objetos por transformación geométrica, se ha definido el concepto de familia.

3. OPERACIONES DE MODELADO.

3. OPERACIONES DE MODELADO.

Se definirán operaciones sobre los objetos para poder construir objetos complejos a partir de otros más simples. Estas operaciones se definen de forman que tengan el mejor comportamiento matemático posible, por lo que, siempre que ello tenga sentido, se definirán de forma que sean conmutativas y asociativas. Esto se hará así porque, por un lado, el que las operaciones gráficas de visualización definidas habitualmente no sean conmutativas es una cuestión tecnológica, ya que se han adaptado a los dispositivos de salida más comunes que no son conmutativos[1], y por otro, para facilitar su manipulación matemática. La utilización de estas operaciones permitirá, además de construir objetos complejos, el considerar a escenarios gráficos como objetos gráficos formados por la suma de una serie de componentes.

Nuestro objetivo es definir, para el conjunto de objetos las siguientes operaciones: suma, unión, intersección, producto, barrido circular y producto por escalar. En la práctica interesará poder trabajar tanto a nivel de objeto como a nivel de instancia. Por ello las operaciones se definirán para ambos conjuntos.

Al definir las operaciones sobre las instancias se demostrará que son estables para la relación de equivalencia efectiva, lo que permitirá demostrar que el resultado de las operaciones es igual, tanto si se opera a nivel de objeto como a nivel de instancia. Esto permite utilizar en la práctica cualquier instancia para representar a los objetos operandos, ya que la elección de la instancia no afecta al resultado.

De cada una de estas operaciones se demostrará que, sobre el conjunto de objetos, cumple determinadas propiedades, tales como: poseer inversa, poseer elementos neutro y/o unidad, asociativa, conmutativa o distributiva, lo que hará que el conjunto de objetos, dotado de estas operaciones, posea determinadas estructuras algebraicas. No todas estas propiedades no se mantendrán para las operaciones definidas sobre el conjunto de instancias. Para evitar la duplicación de teoremas relativos a propiedades, para objetos e instancias, se definirán en primer lugar las operaciones para instancias, demostrándose sus propiedades, entre otras la estabilidad para la relación de equivalencia efectiva. Posteriormente se definirá la operación para objetos, que se demostrará coincide con la definida para instancias, con lo que se garantiza el cumplimiento de las propiedades de aquella.

Para interpretar adecuadamente a las operaciones se debe tener presente que el aspecto forma parte de la información del objeto. Por tanto dos objetos pueden diferir tan sólo en su aspecto, esto es, tener la misma forma (volumen y presencia) pero tener, por ejemplo, distinto

---

[1] Existen dispositivos de salida gráfica cuyo comportamiento, respecto a la superposición de información, se aproxima más a una función conmutativa que a una simple sustitución.

color. Esto hará que el opuesto de un objeto no tenga necesariamente como volumen el complementario del volumen del objeto original, ya que el opuesto del objeto tendrá, en cada punto, presencia y aspecto opuestos al del objeto original. En un punto donde la presencia era uno, será cero (si el dominio de presencia es normal) pero su color podrá ser distinto de cero, con lo que dicho punto pertenecerá al volumen. Como, no obstante, deberemos contemplar la posibilidad de obtener un objeto cuyo volumen sea el complementario de uno dado, introduciremos, en combinación con las operaciones que aquí se definen, funciones especiales que operan sobre los objetos. Dichas funciones, o filtros, se estudiarán en la siguiente sección.

3.1 SUMA DE OBJETOS.

Se definirá en primer lugar la suma de objetos. La suma se realiza basándose en la función de presencia, interpretando un objeto como algo que ocupa una parte del espacio, por lo que la suma de dos objetos ocupará el espacio ocupado por los dos sumandos, pudiendo ocurrir que, para valores concretos de α y μ, la suma este estrictamente contenida en la unión de los volúmenes ocupados por los dos objetos. Al sumar las funciones de presencia, y poder tomar éstas valores negativos, pueden aparecer objetos gráficos cuya interpretación física no sea evidente por:

- la ocupación múltiple ($\mu > 1$) de una parte del espacio, que se debe de entender como la superposición, de los dos objetos que intervienen en la suma, en dicha zona. Esto garantiza el que se pueda retirar (sustraer) uno de los objetos, permaneciendo el espacio ocupado por el otro.

- la ocupación de una zona del espacio con presencia negativa ($\mu < 0$), que se debe de entender como la sustracción de un objeto en dicha zona. Esta situación puede ser útil para definir moldes o perforaciones.

En cualquier caso la apariencia visual de estos objetos estará determinada por su volumen visible.

Definición 3.1: La **Suma** de dos instancias, $R_1 = (D_1,\tau_1,\mu_1,\alpha_1)$ y $R_2 = (D_2,\tau_2,\mu_2,\alpha_2)$, que denotaremos por $R_1 + R_2$, es una instancia $R = (D,\tau,\mu,\alpha)$ definida como:

$$R = R_1+R_2 = (D, \hat{\imath}, \mu_1 \tau_1^{-1} + \mu_2 \tau_2^{-1}, \alpha_1 \tau_1^{-1} + \alpha_2 \tau_2^{-1})$$

siendo $D = \{ P \mid \mu(P) \neq 0 \text{ ó } \alpha(P) \neq 0 \}$ □

Como se puede observar, el resultado de la suma de dos instancias cualesquiera es una instancia normal.

Como caso particular, la **Suma** de dos instancias normales, $R_1 + R_2$, viene dada por la expresión:

$$R_1+R_2 =(D,\tau,\mu,\alpha) = (D, \hat{\imath}, \mu_1 + \mu_2 , \alpha_1 + \alpha_2)$$

siendo $D = \{ P \mid \mu(P) \neq 0 \text{ ó } \alpha(P) \neq 0 \}$

Los siguientes teoremas establecen las propiedades de la suma de instancias.

Teorema 3.1: La suma de instancias es conmutativa y asociativa.

Demostración:

1. El que la suma es **conmutativa** es trivial. Téngase en cuenta que la suma se define a partir de las sumas en los dominios de presencia **P** y color **C**, siendo ambas conmutativas.

2. Demostremos ahora que la suma es **asociativa**. Sean las instancias $R_1 = (D_1, \tau_1, \mu_1, \alpha_1)$, $R_2 = (D_2, \tau_2, \mu_2, \alpha_2)$ y $R_3 = (D_3, \tau_3, \mu_3, \alpha_3)$. La suma de las dos primeras es:

$$(R_1+R_2) = (D_{12}, \hat{\imath}, \mu_{12}, \alpha_{12}) \text{ con}$$

$$\mu_{12} = \mu_1 \tau_1^{-1} + \mu_2 \tau_2^{-1}$$

$$\alpha_{12} = \alpha_1 \tau_1^{-1} + \alpha_2 \tau_2^{-1}$$

$$D_{12} = \{ P \mid \mu_{12}(P) \neq 0 \text{ ó } \alpha_{12}(P) \neq 0 \}$$

La suma de la instancia anterior con R_3 es:

$$(R_1+R_2)+R_3 = (D, \hat{\imath}, \mu, \alpha) \text{ con}$$

$$\mu = (\mu_1 \tau_1^{-1} + \mu_2 \tau_2^{-1}) \hat{\imath} + \mu_3 \tau_3^{-1} =$$

$$\mu_1 \tau_1^{-1} + \mu_2 \tau_2^{-1} + \mu_3 \tau_3^{-1}$$

$$\alpha = (\alpha_1 \tau_1^{-1} + \alpha_2 \tau_2^{-1}) \hat{\imath} + \alpha_3 \tau_3^{-1} =$$

$$\alpha_1 \tau_1^{-1} + \alpha_2 \tau_2^{-1} + \alpha_3 \tau_3^{-1}$$

$$D = \{ P \mid \mu(P) \neq 0 \text{ ó } \alpha(P) \neq 0 \}$$

que obviamente coincide con $R_1+(R_2+R_3)$. □

Teorema 3.2: La suma de instancias es estable para la relación de equivalencia efectiva.

Demostración:

Dados $R_1 \cong R_1'$ y $R_2 \cong R_2'$ demostraremos que

$$R_1 + R_2 \cong R_1' + R_2'$$

Sean

$$R_1 = (D_1, \tau_1, \mu_1, \alpha_1)$$

$$R_2 = (D_2, \tau_2, \mu_2, \alpha_2)$$

supondremos que R_1' y R_2' son instancias normales, en cuyo caso se cumple:

$$R_1' = R_{1N} = (D_1, \tau_{1N}, \mu_{1N}, \alpha_{1N}) = (D_1, \hat{1}, \mu_1 \tau_1^{-1}, \alpha_1 \tau_1^{-1})$$

$$R_2' = R_{2N} = (D_2, \tau_{2N}, \mu_{2N}, \alpha_{2N}) = (D_2, \hat{1}, \mu_2 \tau_2^{-1}, \alpha_2 \tau_2^{-1})$$

y por tanto:

$$R_1' + R_2' = (D, \hat{1}, \mu_1 \tau_1^{-1} + \mu_2 \tau_2^{-1}, \alpha_1 \tau_1^{-1} + \alpha_2 \tau_2^{-1})$$

que es efectivamente equivalente a $R_1 + R_2$. □

Definiremos ahora la suma de objetos gráficos, que se demostrará que coincide con la operación inducida en el conjunto de objetos por la suma de instancias y la relación de equivalencia efectiva.

Definición 3.2: La suma de dos objetos $O_1 = (V_1, \mu_1, \alpha_1)$ y $O_2 = (V_2, \mu_2, \alpha_2)$ es el objeto dado por la expresión:

$$O_1 + O_2 = (V, \mu_1 + \mu_2, \alpha_1 + \alpha_2)$$

siendo $V = \{ P \in \mathbb{R}^n \mid \mu(P) \neq 0 \text{ ó } \alpha(P) \neq 0 \}$ □

Teorema 3.3: La **Suma** de dos objetos, $O_1 + O_2$, es sinónima a la suma de dos instancias cualesquiera, $R_1 + R_2$, sinónimas de aquellos, $R_1 \simeq O_1$ y $R_2 \simeq O_2$.

Demostración:

Evidente, basta con realizar la suma sobre las dos instancias normales. □

Este resultado nos permitirá operar indistintamente con instancias u objetos, ya que el resultado de la suma no se verá afectado por la elección de las instancias sinónimas de estos.

Definición 3.3: Denominaremos objeto **nulo**, y lo representaremos por ϕ, al objeto:

$$\phi = (\phi, \mu_0, \alpha_0)$$

en el que las funciones de forma y presencia están definidas por

$$\mu_0(P) = 0 \qquad \alpha_0(P) = 0$$ □

Obsérvese que este objeto es sinónimo de infinitas instancias, lo que impedirá que el conjunto de instancias con la operación suma sea un grupo. En concreto la instancia normal sinónima de este objeto es

$$R_{\phi} = (\phi, \hat{1}, \mu_0, \alpha_0)$$

Teorema 3.4: La suma de objetos es conmutativa y asociativa, para ella el objeto nulo actúa como elemento neutro y cada objeto O del conjunto de objetos posee un opuesto, que denotaremos por -O.

Demostración:

1. La suma es **conmutativa** y **asociativa**

Trivial, dado que la suma de instancias es conmutativa y asociativa.

2. Existe un **elemento neutro,**

Sea el objeto nulo, ϕ. Su suma con cualquier objeto $O = (V,\mu,\alpha)$ será:

$$\phi + O = (V', \mu, \alpha)$$

$$\text{con } V' = \{ P \mid \mu(P) \neq 0 \text{ ó } \alpha(P) \neq 0\} = V$$

que es igual a O.

3. Cada elemento posee un **opuesto**

Sea el objeto $O = (V, \mu, \alpha)$

Construyamos el objeto O_o dado por:

$$O_o = (V, -\mu, -\alpha)$$

siendo $(-\mu)(P) = -\mu(P)$ y $(-\alpha)(P) = -\alpha(P)$[1]

La suma de O y O_o viene dada por:

$$O + O_o = (V', \mu - \mu, \alpha - \alpha) = (\phi, \mu_0, \alpha_0) = \phi$$

ya que por la definición de la suma $V' = \phi$ □

---

[1] Nótese que los dominios de presencia y color son grupos abelianos respecto a la suma, y por tanto cada elemento x tendrá un único opuesto, que, siguiendo la notación habitual en grupos aditivos, denotaremos por -x.

Corolario 3.1: El conjunto de objetos dotado de la operación suma es un grupo abeliano. □

Definición 3.4: Llamaremos **diferencia** de dos objetos O_1 y O_2 a la suma del primero con el opuesto del segundo, y la denotaremos por $O_1 - O_2$. □

Obviamente la diferencia de dos objetos, $O_1 - O_2$, representados por $R_1 = (D_1,\tau_1,\mu_1,\alpha_1)$ y $R_2 = (D_2,\tau_2,\mu_2,\alpha_2)$ respectivamente, está representada por:

$$O_1\text{-}O_2 = (D,\tau,\mu,\alpha) =$$

$$= (D, \hat{1}, \mu_1 \tau_1^{-1} - \mu_2 \tau_2^{-1}, \alpha_1 \tau_1^{-1} - \alpha_2 \tau_2^{-1})$$

$$\text{siendo } D = \{ P \mid \mu(P) \neq 0 \text{ ó } \alpha(P) \neq 0)$$

Es fácil comprobar que para dos objetos reales[1], la suma de ambos es un objeto que ocupa la unión de los dos volúmenes, con función de presencia suma. Normalmente no se dará la igualdad del volumen final con las uniones de volúmenes, ya que para un punto puede ser que $\alpha_1 = -\alpha_2$ y $\mu_1 = -\mu_2$, por lo que dicho punto no estaría incluido en el volumen final.

Proposición 3.1: En el conjunto de objetos dotado de la operación suma se cumple:

- El elemento neutro es único.
- El opuesto de cada elemento es único.
- La ecuación $X + O_1 = O_2$ admite solución única dada por
  $X = O_2 - O_1$
- Es valida la ley de simplificación.

Demostración:

Trivial. Por tener dicho conjunto estructura de grupo abeliano. □

Nótese que el conjunto de instancias dotado de la operación suma no es grupo, dado que no existe un único elemento neutro.

Hasta aquí se han definido sólo dos operaciones con los objetos e instancias: sumas y transformaciones geométricas. Veamos que las transformaciones geométricas son distributivas respecto de la suma.

Teorema 3.5: Las transformaciones geométricas de objetos gráficos son distributivas respecto de la suma. □

---

[1] Entenderemos por objetos reales aquellos que pueden ser abstracción de un sistema físico. Para ellos la presencia será cero o uno, y el aspecto será cero en todos los puntos con presencia cero.

Demostración:

Debemos demostrar que dados dos objetos cualesquiera, $O_1=(V_1,\mu_1,\alpha_1)$ y $O_2=(V_2,\mu_2,\alpha_2)$, y una transformación geométrica, T, se cumple:

$$T(O_1+O_2) = T(O_1) + T(O_2)$$

En efecto, por el teorema 3.3, es

$$T(O_1+O_2) = T((V,\mu_1+\mu_2,\alpha_1+\alpha_2))$$

que por el teorema 2.8, es igual a

$$(T(V),(\mu_1+\mu_2)\cdot T^{-1},(\alpha_1+\alpha_2)\cdot T^{-1})$$

siendo $V = \{ P \mid (\mu_1+\mu_2)(P) \neq 0 \text{ ó } (\alpha_1+\alpha_2)(P) \neq 0)$

Por otra parte, según los mismos teoremas

$$T(O_1)+T(O_2)=(T(V_1),\mu_1\cdot T^{-1},\alpha_1\cdot T^{-1}) + (T(V_1),\mu_1\cdot T^{-1},\alpha_1\cdot T^{-1}) =$$

$$= (V',\mu_1\cdot T^{-1}+\mu_2\cdot T^{-1},\alpha_1\cdot T^{-1}+\alpha_2\cdot T^{-1})$$

siendo

$$V' = \{ P \mid (\mu_1\cdot T^{-1}+\mu_2\cdot T^{-1})(P) \neq 0 \text{ ó } (\alpha_1\cdot T^{-1}+\alpha_2\cdot T^{-1})(P) \neq 0)$$

Ambos objetos, obviamente, coinciden. □

3.1.1 Construcción de objetos por suma.

La suma de objetos equivale a la colocación o superposición de dos o más objetos en una zona del espacio. De este modo, podemos definir un objeto complejo mediante la suma de una serie de objetos más simples en los que se puede descomponer. Así, una zona de estudio se puede definir como:

$$Zona\_de\_estudio = Mesa + Silla + Libro$$

La capacidad de la suma para construir objetos es limitada, pues no permite aumentar la dimensión de los objetos, utilizando sumas finitas. Es decir, la suma de dos objetos cuyo volumen tiene dimensión dos tendrá a lo sumo dimensión dos.

No obstante, determinados objetos se pueden describir como suma de una secuencia finita de objetos patrón. Así, una imagen de puntos se podría describir como suma de dichos puntos. En la sección 3.2.1 se volverá a incidir en este aspecto.

3.2 PRODUCTO ADITIVO.

A continuación se define el producto aditivo de objetos por escalares. Hacer el producto de un escalar, k, por un objeto ,O, equivaldrá a la suma k veces del objeto consigo mismo. Se demostrará que con este producto sobre el cuerpo de enteros módulo q, con q primo arbitrariamente grande, el grupo de los objetos tiene estructura de espacio vectorial.

Al igual que se hizo para la suma, se definirá en primer lugar el producto de instancias, para definir a continuación el producto de objetos.

Definición 3.5: Dada una instancia, $R=(D,\tau,\mu,\alpha)$, y un número entero k, se define el producto aditivo k*R como:

$$k * R = (D, \tau, k * \mu, k * \alpha)$$

siendo $(k * \mu)(P) = k * (\mu(P))$

y $(k * \alpha)(P) = k * (\alpha(P))$ □

Teorema 3.6: El producto aditivo de enteros por instancias cumple las siguientes propiedades:

- Distributividad respecto a la suma de instancias.
- Distributividad respecto a la suma de enteros.
- Asociativo para los enteros.
- El número uno actúa como elemento neutro.

Demostración:

1. Distributividad para instancias. Se demostrará que para cualquier entero k, y cualquier par de instancias $R_1= (D_1, \tau_1, \mu_1, \alpha_1)$ y $R_2= (D_2, \tau_2, \mu_2, \alpha_2)$ se cumple que

$$k*(R_1+R_2) = k*R_1 + k*R_2$$

En efecto

$$k*R_1+k*R_2 = (D,\hat{\imath},k*\mu_1\cdot\tau_1^{-1}+k*\mu_2\cdot\tau_2^{-1},k*\alpha_1\cdot\tau_1^{-1}+k*\alpha_2\cdot\tau_2^{-1}) =$$
$$= (D,\hat{\imath},k*(\mu_1\cdot\tau_1^{-1}+\mu_2\cdot\tau_2^{-1}),k*(\alpha_1\cdot\tau_1^{-1}+\alpha_2\cdot\tau_2^{-1})) = k*(R_1+R_2)$$

2. Distributividad para enteros. Se demostrará que para todo par de enteros k_1,k_2, y toda instancia, R, se cumple que $(k_1+k_2)*R = k_1*R + k_2*R$

En efecto

$$k_1*R+k_2*R = (D, \tau, k_1*\mu + k_2*\mu, k_1*\alpha + k_2*\alpha) =$$

$$= (D, \tau, (k_1+k_2)*\mu, (k_1+k_2)*\alpha) = (k_1+k_2)*R$$

3. Asociativo para escalares. Es obvio el demostrar que para todo par de enteros k_1,k_2, y toda instancia, R, se cumple que

$$k_1*(k_2*R) = (k_1*k_2)*R$$

4. El uno es elemento neutro. Su comprobación es trivial. □

Teorema 3.7: El producto aditivo de instancias es estable para la relación de equivalencia efectiva.

Demostración:

Dados dos instancias efectivamente equivalentes, $R_1 \cong R_1'$, y un entero, k, demostraremos que $k*R_1 \cong k*R_1'$
Sea

$$R_1 = (D_1,\tau_1,\mu_1,\alpha_1)$$

supondremos que R_1' es la instancia normal, en cuyo caso se cumple:

$$R_1' = R_{1N} = (D_{1N},\hat{\imath},\mu_{1N},\alpha_{1N}) = (D_{1ef}, \hat{\imath}, \mu_1\ \tau_1^{-1}, \alpha_1\ \tau_1^{-1})$$

y por tanto:

$$k*R_1' = (D_{1ef}, \hat{\imath}, k*\mu_1\ \tau_1^{-1}, k*\alpha_1\ \tau_1^{-1})$$

que es efectivamente equivalente a $k*R_1$; ya que es la instancia normal de la clase de $k*R_1$. □

Ahora definiremos el producto de escalares por objetos gráficos.

Definición 3.6: El producto aditivo de un entero k por un objeto, $O=(V,\mu,\alpha)$, es el objeto gráfico dado por la expresión:

$$k*O = (V', k*\mu, k*\alpha)$$

$$\text{siendo } V' = \{ P \mid k*\mu(P) \neq 0 \ \vee \ k*\alpha(P) \neq 0) \qquad \square$$

Teorema 3.8: El **producto aditivo** de un entero, k, por un objeto, O, es sinónimo al producto del entero por una cualquiera de las instancias sinónimas al objeto.

Demostración:

Dado que el producto de enteros por instancias es estable para la relación de equivalencia efectiva, el resultado no dependerá de la instancia elegida. Por tanto bastará con demostrar que, dada una instancia sinónima del objeto, su producto por el escalar es sinónimo del producto de este por el objeto.

En efecto, sea $R=(V,\hat{\imath},\mu,\alpha)$ la instancia normal sinónima de O, su producto con k es

$$k*R = (V,\hat{\imath},k*\mu,k*\alpha)$$

que es la instancia normal sinónima del objeto $(V, k*\mu, k*\alpha)$. □

Si analizamos la relación existente entre el este producto y la suma antes definida veremos que el conjunto de objetos gráficos con ambas operación tiene estructura de espacio vectorial.

Teorema 3.9: El conjunto de objetos dotado de la operación suma y del producto aditivo sobre el cuerpo de enteros módulo n (con n primo)[1], es un espacio vectorial. □

Demostración:

Es inmediato comprobar que el producto aditivo de enteros por objetos cumple las mismas propiedades que el producto aditivo de enteros por instancias, que junto con el hecho de ser un grupo respecto de la suma le confieren estructura de espacio vectorial. □

Corolario 3.2: Para todo objeto O, y todo entero k, se cumple:

$$0 * O = \phi$$
$$k * \phi = \phi$$
$$k * O = \phi \Rightarrow k = 0 \vee O = \phi$$
$$(-k) * O = -(k * O)$$

Demostración:

Inmediata. □

Con las dos operaciones introducidas hasta aquí, se ha dotado al conjunto de objetos de estructura de espacio vectorial. Esta estructura es útil para describir combinaciones de

---

[1] El espacio vectorial se define sobre el cuerpo de enteros módulo n (con n primo), por requerirse un cuerpo externo para definir el espacio vectorial.

objetos y como soporte del modelado de sólidos por enumeración, que se comentará seguidamente. Sin embargo, dada la complejidad de los objetos gráficos reales, será necesario introducir otras operaciones de modelado.

3.2.1 Representación de objetos por enumeración espacial.

Uno de los posibles métodos de representación de objetos es por enumeración espacial. El método está basado en la realización de una división del espacio en pequeñas celdas, de tal forma que el objeto se describe indicando cuales de dichas celdas ocupa. Esta forma de descripción se suele utilizar en sistemas 2D (imagen raster y quadtrees) y 3D (modelado de sólidos por enumeración espacial u octtres). La principal limitación de este método de representación es su imprecisión.

Esta forma de representación está soportada por el espacio vectorial de objetos. Es más, cualquier objeto que se encuentre en una zona finita del espacio, y que se describa con una precisión finita, se podrá construir como combinación lineal finita de objetos del espacio vectorial de objetos.

Teorema 3.10: La representación de cualquier objeto de $\mathbf{R}^n$, confinado a una zona del espacio de dimensiones $L_1 \times L_2 \times .. \times L_n$, se puede realizar utilizando un vector de $M \cdot (c+1)$ enteros, con un error menor que r, siendo:

$$M = (L_1 \cdot L_2 \cdot .. \cdot L_n)/r^n$$

y c la dimensión del subdominio de aspecto utilizado para los objetos.

Demostración:

Para demostrar el teorema construiremos una base del espacio vectorial de objetos que permita generar cualquier objeto confinado en la zona del espacio indicada con el error establecido. De este modo cualquier objeto se podrá expresar como combinación de elementos de la base, y quedará perfectamente especificado por el vector de enteros formado por los coeficientes de dicha combinación.

La consideración de un error, r, permite dividir la zona del espacio en que se encuentra el objeto en celdas, ocupando cada una un volumen que es un cuadrado (cubo o hipercubo). En total existirán M celdas, en la zona de interés.

Por cada celda del espacio habrá $c+1$ objetos de la base. Estos $c+1$ objetos se definen de forma que combinándolos se pueda generar cualquier color y presencia. Por ejemplo, se puede hacer que todos salvo uno tengan presencia cero, el elemento restante tendrá presencia 1. Los c primeros elementos tendrán como aspectos los de una base del subdominio de aspecto usado en los objetos a describir, el último tendrá como color el neutro de dicho espacio.

Con este esquema, cualquier objeto se construye como combinación lineal de la base definida, o lo que es lo mismo, indicando cuantas veces aparece en cada una de las celdas y con que aspecto y presencia. □

Si los objetos descritos pueden tomar un único valor de aspecto, la dimensión de su subdominio de color será cero, y por tanto, la dimensión de la base será M. Cuando, además, se consideran tan sólo valores de presencia cero y uno, los valores de los coeficientes de las combinaciones lineales de elementos de la base podrán tomar sólo valores binarios, con lo que el objeto se podrá describir con M bits.

3.3 UNION DE OBJETOS.

Esta sección, junto con las siguientes, se dedica a la definición de operaciones booleanas (unión, intersección y complementación) de objetos, haciendo que dichas operaciones coincidan con la idea intuitiva de las operaciones sobre volúmenes visibles. Se demostrará que el conjunto de objetos, con estas operaciones, constituye un álgebra de Boole.

La presente sección define la operación de unión, enunciando sus propiedades. Esta operación al igual que se hizo con las anteriores se definirá sobre instancias y sobre objetos.

Definición 3.7: La **Unión** de dos instancias, $R_1=(D_1,\tau_1,\mu_1,\alpha_1)$ y $R_2=(D_2,\tau_2,\mu_2,\alpha_2)$, es una instancia definida como:

$$R_1 \cup R_2 = (D,\hat{\imath},\mu,\alpha) = (D, \hat{\imath}, (\mu_1\cdot\tau_1^{-1}) \cup (\mu_2\cdot\tau_2^{-1}),(\alpha_1\cdot\tau_1^{-1}) \cup (\alpha_2\cdot\tau_2^{-1}))$$

siendo $D = \{ P \mid \mu(P) \neq 0 \text{ ó } \alpha(P) \neq 0)$ □

Comentarios:

La unión en los dominios de aspecto y presencia se realizará según se haya definido en los dominios concretos que se utilicen en cada caso.

Los ceros en las igualdades con valores de presencia y aspecto denotan al elemento neutro de dichas estructuras.

Comenzaremos demostrando que la operación es estable para la relación de equivalencia efectiva.

Teorema 3.11: La unión de instancias es estable para la relación de equivalencia efectiva.

Demostración:

Dados $R_1 \cong R_1'$ y $R_2 \cong R_2'$ demostraremos que $R_1 \cup R_2 \cong R_1' \cup R_2'$

Sean

$$R_1 = (D_1,\tau_1,\mu_1,\alpha_1)$$

$$R_2 = (D_2,\tau_2,\mu_2,\alpha_2)$$

supondremos que R_1' y R_2' son instancias normales, en cuyo caso se cumple:

$$R_1'=R_{1N} =(D_{1N},\tau_{1N},\mu_{1N},\alpha_{1N}) = (D_{1N}, \hat{\imath}, \mu_1\ \tau_1^{-1}, \alpha_1\ \tau_1^{-1})$$

$R_2' = R_{2N} = (D_{2N}, \tau_{2N}, \mu_{2N}, \alpha_{2N}) = (D_{2N}, \hat{1}, \mu_2\ \tau_2^{-1}, \alpha_2\ \tau_2^{-1})$

y por tanto:

$R_1' \cup R_2' = (D, \hat{1}, \mu_1\ \tau_1^{-1} \cup \mu_2\ \tau_2^{-1}, \alpha_1\ \tau_1^{-1} \cup \alpha_2\ \tau_2^{-1})$

que obviamente coincide con $R_1 \cup R_2$ □

Definición 3.8: La unión de dos objetos $O_1 = (V_1, \mu_1, \alpha_1)$ y $O_2 = (V_2, \mu_2, \alpha_2)$ es el objeto dado por la expresión:

$$O_1 \cup O_2 = (V, \mu_1 \cup \mu_2, \alpha_1 \cup \alpha_2)$$

siendo $V = \{ P \in \mathbf{R}^n \mid (\mu_1 \cup \mu_2)(P) \neq 0 \text{ ó } (\alpha_1 \cup \alpha_2)(P) \neq 0 \}$ □

Teorema 3.12: La unión de dos objetos es sinónima de la instancia resultante de la unión de dos instancias cualesquiera sinónimas de los objetos.

Demostración:

Sean dos objetos, $O_1 = (V_1, \mu_1, \alpha_1)$ y $O_2 = (V_2, \mu_2, \alpha_2)$, y las instancias normales sinónimas de ambos, $R_1 = (D_1, \hat{1}, \mu_1, \alpha_1)$ y $R_2 = (D_2, \hat{1}, \mu_2, \alpha_2)$. La unión de dichas instancias, R_1 y R_2, viene dada por

$$R_1 \cup R_2 = (D, \hat{1}, \mu_1 \cup \mu_2, \alpha_1 \cup \alpha_2)$$

siendo $D = \{ P \mid (\mu_1 \cup \mu_2)(P) \neq 0 \text{ ó } (\alpha_1 \cup \alpha_2)(P) \neq 0 \}$

que es la instancia normal sinónima de

$$(V, \mu_1 \cup \mu_2, \alpha_1 \cup \alpha_2)$$

siendo $V = \{ P \in \mathbf{R}^n \mid (\mu_1 \cup \mu_2)(P) \neq 0 \text{ ó } (\alpha_1 \cup \alpha_2)(P) \neq 0 \}$ □

Las siguientes definiciones establecen elementos singulares del conjunto de objetos.

Definición 3.9: Denominaremos objeto universal, o universo, y lo denotaremos por ∞, al representado por:

$$\infty = (\mathbf{R}^n, \hat{1}, \mu_\infty, \alpha_\infty)$$

con $\mu_\infty(P) = \sim\phi$ y $\alpha_\infty(P) = \sim\phi$ □

Teorema 3.13: La unión de objetos cumple las siguientes propiedades: idempotente, conmutativa, asociativa, y existencia de elementos neutro y absorbente.

Demostración:

1. Idempotente.

Trivial, basta con considerar que la unión de instancias se construye en base a la unión definida en los dominios de aspecto y presencia, y ambos son un álgebra de Boole.

2. Conmutativa.

Trivial.

3. Asociativa.

Trivial.

4. Existencia de elemento neutro.

Consideremos el objeto nulo definido anteriormente (Def. 3.3), teniendo en cuenta que los elementos vacíos de las álgebras de boole de aspecto y presencia coinciden con los elementos neutros de los espacios vectoriales de dichos dominios, por lo que dicha definición es correcta, la unión de dicho objeto con cualquier objeto, $O = (V,\mu,\alpha)$, será:

$$\phi \cup O = O' = (V', \mu, \alpha),$$

$$\text{con } V' = \{ P \mid \mu(P) \neq 0 \text{ ó } \alpha(P) \neq 0) = V$$

que es igual a O.

5. Existencia de elemento absorbente.

Consideremos el objeto universal anteriormente definido. Su unión con cualquier objeto está representada por:

$$\infty \cup O = O' = (V', \mu_\infty \cup \mu, \alpha_\infty \cup \alpha),$$

que es igual a ∞, por ser $\sim\phi$ elemento absorbente en los dominios de presencia y color. □

3.4 INTERSECCION DE OBJETOS.

Definiremos en esta sección la operación de intersección, siguiendo un proceso paralelo al seguido en la definición de unión.

Definición 3.10: La **Intersección** de dos instancias, $R_1=(D_1,\tau_1,\mu_1,\alpha_1)$ y $R_2=(D_2,\tau_2,\mu_2,\alpha_2)$, es la instancia definida como:

$$R_1 \cap R_2 = (D, \hat{\imath}, \mu, \alpha) = (D, \hat{\imath}, (\mu_1 \tau_1^{-1}) \cap (\mu_2 \tau_2^{-1}),(\alpha_1 \tau_1^{-1}) \cap (\alpha_2 \tau_2^{-1}))$$

siendo $D = \{ P \mid \mu(P) \neq 0 \text{ ó } \alpha(P) \neq 0)$ □

Teorema 3.14: La intersección de instancias es estable para la relación de equivalencia efectiva.

Demostración:

Dados $R_1 \cong R_1'$ y $R_2 \cong R_2'$ demostraremos que

$$R_1 \cap R_2 \cong R_1' \cap R_2'$$

Sean

$$R_1 = (D_1,\tau_1,\mu_1,\alpha_1)$$

$$R_2 = (D_2,\tau_2,\mu_2,\alpha_2)$$

supondremos que R_1' y R_2' son instancias normales, en cuyo caso se cumple:

$$R_1'=R_{1N} =(D_{1N},\tau_{1N},\mu_{1N},\alpha_{1N}) = (D_1, \hat{\imath}, \mu_1 \tau_1^{-1}, \alpha_1 \tau_1^{-1})$$

$$R_2'=R_{2N} =(D_{2N},\tau_{2N},\mu_{2N},\alpha_{2N}) = (D_2, \hat{\imath}, \mu_2 \tau_2^{-1}, \alpha_2 \tau_2^{-1})$$

y por tanto:

$$R_1' \cap R_2' = (D, \hat{\imath}, \mu_1 \tau_1^{-1} \cap \mu_2 \tau_2^{-1}, \alpha_1 \tau_1^{-1} \cap \alpha_2 \tau_2^{-1})$$

que obviamente coincide con $R_1 \cap R_2$. □

Definición 3.11: La intersección de dos objetos, $O_1=(V_1,\mu_1,\alpha_1)$ y $O_2=(V_2,\mu_2,\alpha_2)$, es el objeto dado por la expresión:

$$O_1 \cap O_2 = (V, \mu, \alpha) = (V, \mu_1 \cap \mu_2, \alpha_1 \cap \alpha_2)$$

$$\text{siendo } V = \{ P \in \mathbf{R}^n \mid \mu(P) \neq 0 \text{ ó } \alpha(P) \neq 0 \}$$ □

Teorema 3.15: La intersección de dos objetos es sinónima de la instancia resultante de la intersección de dos instancias cualesquiera sinónimas de los objetos.

Demostración:

Sean dos objetos, $O_1=(V_1,\mu_1,\alpha_1)$ y $O_2=(V_2,\mu_2,\alpha_2)$, y las instancias normales sinónimas de ambos, $R_1=(V_1,\hat{\imath},\mu_1,\alpha_1)$ y $R_2=(V_2,,\hat{\imath},\mu_2,\alpha_2)$. El resultado de la intersección de las instancias es

$$R_1 \cap R_2 = (D, \hat{\imath}, \mu, \alpha) = (D, \hat{\imath}, \mu_1 \cap \mu_2, \alpha_1 \cap \alpha_2)$$

$$\text{siendo } D = \{ P \mid \mu(P) \neq 0 \text{ ó } \alpha(P) \neq 0)$$

que es la instancia normal sinónima de

$$(V, \mu, \alpha) = (V, \mu_1 \cap \mu_2, \alpha_1 \cap \alpha_2)$$

$$\text{siendo } V = \{ P \in \mathbf{R}^n \mid \mu(P) \neq 0 \text{ ó } \alpha(P) \neq 0)$$ □

Teorema 3.16: La intersección de objetos cumple las siguientes propiedades: idempotente, conmutativa, asociativa y existencia de elementos neutro y absorbente.

Demostración:

1. Idempotente.

Trivial, basta con considerar que la intersección de instancias se construye en base a la intersección definida en los dominios de color y presencia, y ambos son un álgebra de Boole.

2. Conmutativa.

Trivial.

3. Asociativa.

Trivial.

4. Existencia de elemento absorbente:

Consideremos el objeto nulo. Su intersección con cualquier objeto $O=(V,\mu,\alpha)$ será:

$$\phi \cap O = O' = (V', \mu \cap \mu_\phi, \alpha \cap \alpha_\phi) =$$

$$= (V', \mu_\phi, \alpha_\phi)$$

$$\text{con } V' = \{ P \mid \mu_\phi(P) \neq 0 \text{ ó } \alpha_\phi(P) \neq 0) = \phi$$

5. Existencia de elemento neutro:

Consideremos el objeto universal, su intersección con cualquier objeto está representada por:

$$\infty \cap O = O' = (V', \mu_\infty \cap \mu, \mu_\infty \cap \alpha),$$

que es igual a O, por ser $\sim\phi$ elemento neutro de la intersección en los dominios de presencia y color. □

Se demostrarán seguidamente que la unión e intersección son mutuamente distributivas y que cumplen la ley de absorción. Ambas propiedades son necesarias para que el conjunto de objetos dotado de la unión, intersección y complemento pueda ser un algebra de boole.

Teorema 3.17: La unión e intersección de objetos son mutuamente distributivas.

Demostración:

Sean tres objetos cualesquiera O_1, O_2 y O_3, demostraremos que:

$$O_1 \cup (O_2 \cap O_3) = (O_1 \cup O_2) \cap (O_1 \cup O_3)$$

y

$$O_1 \cap (O_2 \cup O_3) = (O_1 \cap O_2) \cup (O_1 \cap O_3)$$

1. $O_1 \cup (O_2 \cap O_3) = (O_1 \cup O_2) \cap (O_1 \cup O_3)$

Sean

$O_1 = (V_1,\mu_1,\alpha_1)$, $O_2 = (V_2,\mu_2,\alpha_2)$ y $O_3 = (V_3,\mu_3,\alpha_3)$

Se cumple que

$$O_1 \cup (O_2 \cap O_3) = O_1 \cup (V', \mu_2 \cap \mu_3, \alpha_2 \cap \alpha_3) =$$

$$= (V'', \mu_1 \cup (\mu_2 \cap \mu_3), \alpha_1 \cup (\alpha_2 \cap \alpha_3))$$

siendo $V' = \{ P \mid (\mu_2 \cap \mu_3)(P) \neq 0 \text{ ó } (\alpha_2 \cap \alpha_3)(P) \neq 0)$

y

$$V'' = \{ P \mid (\mu_1 \cup (\mu_2 \cap \mu_3))(P) \neq 0 \text{ ó } (\alpha_1 \cup (\alpha_2 \cap \alpha_3))(P) \neq 0)$$

basta tener en cuenta que los dominios de presencia y color son álgebras de Boole para comprobar que dicha expresión coincide con

$(O_1 \cup O_2) \cap (O_1 \cup O_3)$.

2. Para comprobar que

$O_1 \cap (O_2 \cup O_3) = (O_1 \cap O_2) \cup (O_1 \cap O_3)$

basta con seguir el mismo proceso de comprobación del apartado anterior. □

Teorema 3.18: La unión y la intersección cumplen la ley de absorción: $O_1 \cup (O_2 \cap O_1) = O_1 = O_1 \cap (O_2 \cup O_1)$

Demostración:

Por cumplir la unión y la intersección de objetos la leyes distributivas e idempotentes:

$$O_1 \cup (O_2 \cap O_1) = O_1 \cap (O_2 \cup O_1)$$

para comprobar que ambas expresiones son iguales a O_1 basta con operar sobre las instancias, haciendo uso del hecho de que los dominios de presencia y color son álgebras de Boole. □

3.5 COMPLEMENTACION DE OBJETOS.

La complementación de un objeto equivale a generar un molde del objeto. En el caso de objetos reales, el complemento tendrá presencia 1 en el complementario de su volumen.

Definición 3.12: El **complemento** de una instancia, $R = (D,\tau,\mu,\alpha)$, es la instancia, $\sim R$, definida como:

$$\sim R = (D',\hat{1},\mu',\alpha') \quad (D', \hat{1}, \sim(\mu\,\tau^{-1}), \sim(\alpha\,\tau^{-1}))$$

siendo $D' = \{ P \mid \mu'(P) \neq 0 \text{ ó } \alpha'(P) \neq 0)$

donde $\sim(\mu\,\tau^{-1})(P) = \sim(\mu(\tau^{-1}(P)))$ □

El valor de $\sim\mu$ dependerá del dominio de presencia utilizado. En el caso de que se utilice un dominio normal $\sim 0 = 1$ y $\sim 1 = 0$.

Teorema 3.19: La complementación de instancias de objetos es estable para la relación de equivalencia efectiva.

Demostración:

Dadas $R = R'$ demostraremos que $\sim R = \sim R'$
Sea

$$R = (D,\tau,\mu,\alpha)$$

supondremos que R' es una instancia normal, en cuyo caso se cumple:

$$R' = R_N = (D_N,\hat{1},\mu_N,\alpha_N) = (D_N, \hat{1}, \mu\,\tau^{-1}, \alpha\,\tau^{-1})$$

y por tanto:

$$\sim R' = (D'', \hat{1}, \sim(\mu\,\tau^{-1}), \sim(\alpha\,\tau^{-1}))$$

que obviamente coincide con $\sim R$ □

Gracias a la propiedad anterior, se puede definir el complemento de un objeto del siguiente modo.

Definición 3.13: El complemento de un objeto, $O=(V,\mu,\alpha)$, es el objeto dado por la expresión:

$$\sim O = (V', \sim\mu, \sim\alpha)$$

$$\text{siendo } V' = \{ P \in \mathbf{R}^n \mid \sim\mu(P) \neq 0 \text{ ó } \sim\alpha(P) \neq 0 \} \qquad \square$$

Teorema 3.20: El complemento de un objeto es sinónimo del complemento de una instancia cualquiera sinónima de él.

Demostración:

Sea el objeto, $O=(V,\mu,\alpha)$, la instancia normal sinónima de O es $R=(V,\hat{\imath},\mu,\alpha)$, cuyo complemento es

$$\sim R = (D, \hat{\imath}, \sim\mu, \sim\alpha)$$

$$\text{siendo } D = \{ P \mid \sim\mu(P) \neq 0 \text{ ó } \sim\alpha(P) \neq 0)$$

que es la instancia normal sinónima de

$$\sim O = (V', \sim\mu, \sim\alpha)$$

$$\text{con } V' = \{ P \in \mathbf{R}^n \mid \sim\mu(P) \neq 0 \text{ ó } \sim\alpha(P) \neq 0) \qquad \square$$

Teorema 3.13: La complementación cumple las propiedades de involución, dualidad, la unión de cada elemento y su complementario es el objeto Universal, y la intersección de ambos es el objeto nulo.

Demostración:

1. Involución.

Es trivial que $\sim(\sim O) = O$, basta con considerar que la complementación de objetos se construye en base a la complementación definida en los dominios de aspecto y presencia, y ambos constituyen un álgebra de Boole.

2. La unión de cualquier objeto con su complementario es el objeto universal.

En efecto, sea $O = (V,\mu,\alpha)$, un objeto cualquiera. Veamos que $O \cup \sim O = \infty$. El complemento de O es

$$\sim O = (V', \sim\mu, \sim\alpha)$$

$$\text{con } V' = \{ P \in \mathbf{R}^n \mid \sim\mu(P) \neq 0 \text{ ó } \sim\alpha(P) \neq 0 \}$$

La unión de ambos es:

$$O \cup \sim O = (V'', \mu \cup \sim\mu, \alpha \cup \sim\alpha)$$

$$\text{con } V'' = \{ P \in \mathbf{R}^n \mid (\mu \cup \sim\mu)(P) \neq 0 \text{ ó } (\alpha \cup \sim\alpha)(P) \neq 0 \}$$

por ser los dominios de presencia y color álgebras de boole se cumple para ambas que la unión de cada elemento con su complementario es el objeto universal, que es distinto del vacío. Por tanto

$$V'' = \mathbf{R}^n$$

y en consecuencia $\sim O = (\mathbf{R}^n, \mu_\infty, \alpha_\infty)$ que es el objeto universal.

3. La intersección de cada objeto con su complementario es el objeto vacío.

Simple comprobación. Basta con seguir el mismo proceso de la demostración anterior.

4. Dualidad.

La dualidad liga las tres operaciones, unión, intersección y complemento, del siguiente modo:

$$\sim(O_1 \cup O_2) = \sim O_1 \cap \sim O_2 \text{ y } \sim(O_1 \cap O_2) = \sim O_1 \cup \sim O_2$$

Para demostrar que se cumple basta con sustituir en ambos lados de las expresiones para dos objetos arbitrarios. □

Teorema 3.21: El conjunto de objetos dotado de la Unión, Intersección y complementación es un **Algebra de Boole**.

Demostración:

Basta con considerar las propiedades anteriormente demostradas para la unión, intersección y complementación de objetos. □

3.6 BARRIDO O PRODUCTO DE OBJETOS.

En esta sección, se definirá una operación, que denominaremos producto de objetos, o barrido, que permitirá representar la generación de un objeto por barrido.

Usualmente, un objeto generado por barrido es un objeto que se obtiene por desplazamiento de un objeto según una trayectoria. La operación que definiremos generaliza esta noción permitiendo que un objeto se desplace sobre otro objeto, y se denominará producto de objetos. Geométricamente este producto equivale a la realización de una traslación de un objeto a cada uno de los puntos del otro, efectuando posteriormente una suma de los objetos trasladados.

En general, esta operación incrementará la dimensión[1] del objeto. En el caso particular de que uno de los objetos sea un punto, el producto equivaldrá a la realización de una traslación.

En determinados casos este producto no estará definido, dado que habrá situaciones en que la suma a realizar para efectuar el producto de dos objetos tenga infinitos términos.

Antes de definir la operación de barrido, será necesario introducir el concepto auxiliar de conjunto de puntos producto. Dicho conjunto, se definirá para un punto dado y un par de instancias de objetos.

Definición 3.14: Denominaremos **conjunto de pares producto** de dos instancias, $R_1=(D_1,\tau_1,\mu_1,\alpha_1)$ y $R_2=(D_2,\tau_2,\mu_2,\alpha_2)$, sobre el punto P, y lo denotaremos por $\delta(R_1,R_2,P)$, al conjunto de los pares ordenados de puntos (P_1,P_2) que cumplen:

$$P_1 \in D_{ef1}, P_2 \in D_{ef2} \quad y \quad P=\mathbf{T}_t^{[\tau_1(P_1)]}(\tau_2(P_2)) \qquad \square$$

$\tau_1(P_1)$ es un punto del objeto sinónimo de R_1, y $\tau_2(P_2)$ es un punto del objeto sinónimo de R_2. El conjunto de pares producto es por tanto el conjunto de pares de puntos de las dos instancias (P_1,P_2), tales que al trasladar el transformado del segundo punto, $\tau_2(P_2)$, en una distancia dada por el transformado del primer punto, $\tau_1(P_1)$, se obtiene el punto de referencia P.

Lema 3.1: Sean R_1 y R_2 dos instancias cualesquiera y $P \in \mathbf{R}^n$ arbitrario, se verifica que:

$$(P_1,P_2) \in \delta(R_1,R_2,P) \text{ sii } (P_2,P_1) \in \delta(R_2,R_1,P)$$

---

[1] Entendemos por dimensión del objeto la dimensión del volumen del objeto.

Demostración:

Sean $R_1=(D_1,\tau_1,\mu_1,\alpha_1)$ y $R_2=(D_2,\tau_2,\mu_2,\alpha_2)$, dos instancias arbitrarias y P un punto cualquiera de $\mathbf{R}^n$. El conjunto de pares producto $\delta(R_1,R_2,P)$, viene dado por:

$$\delta(R_1,R_2,P)=\{(P_1,P_2) \mid P_1\in D_{ef1},P_2\in D_{ef2} \text{ y } P=T_t^{[\tau 1(P1)]}(\tau_2(P_2))\}$$

dado que $T_t^{[\tau 1(P1)]}(\tau_2(P_2)) = T_t^{[\tau 2(P2)]}(\tau_1(P_1))$

resulta evidente que

$$\forall (P_1,P_2) \in \delta(R_1,R_2,P)$$

se cumple que $P_2\in D_{ef2},P_1\in D_{ef1}$ y $P=T_t^{[\tau 2(P2)]}(\tau_1(P_1))$

y por tanto $(P_2,P_1) \in \delta(R_2,R_1,P)$ □

Una consecuencia inmediata de este lema es el hecho de que $\delta(R_1,R_2,P)$ y $\delta(R_2,R_1,P)$ tengan la misma cardinalidad.

Dadas dos instancias, podemos pensar en el conjunto de puntos del espacio sobre los que el conjunto de pares producto de ambas instancias es no vacío. Este concepto se introduce formalmente en la siguiente definición.

Definición 3.15: Denominaremos **recorrido** de una instancia, $R_1=(D_1,\tau_1,\mu_1,\alpha_1)$, sobre otra instancia, $R_2=(D_2,\tau_2,\mu_2,\alpha_2)$, al volumen formado por el conjunto de puntos P, tales que $\delta(R_1,R_2,P)$ es no vacío.

$$\Pi(R_1,R_2) = \{ P \in \mathbf{R}^n \mid \delta(R_1,R_2,P) \neq \phi \}$$ □

Resulta evidente, por el lema 3.1, que $\Pi(R_1,R_2)$ coincide con $\Pi(R_2,R_1)$.

Considerando la definición de pares de puntos producto, podemos caracterizar al recorrido del siguiente modo.

Lema 3.2: El recorrido de una instancia, $R_1=(D_1,\tau_1,\mu_1,\alpha_1)$, sobre otra instancia, $R_2=(D_2,\tau_2,\mu_2,\alpha_2)$, viene dado por:

$$\Pi(R_1,R_2) = \{ P \in \mathbf{R}^n \mid \exists P_1\in D_{ef1},P_2\in D_{ef2} \text{ con } P=T_t^{[\tau 1(P1)]}(\tau_2(P_2))\}$$

Demostración:

Trivial. □

Para que el concepto anterior sea realmente útil, a la hora de definir operaciones sobre objetos, dicho conjunto debe de ser el mismo para todas las instancias de una misma clase. Este hecho queda establecido por el siguiente lema.

Lema 3.3: Sea $\Pi(R_1,R_2)$ el recorrido de dos instancias, R_1 y R_2. Entonces para cualquier par de instancias R'_1 y R'_2 efectivamente equivalentes a las anteriores, $R'_1 \cong R_1$ y $R'_2 \cong R_2$, se cumple que

$$\Pi(R_1,R_2) = \Pi(R'_1,R'_2)$$

Demostración:

Debemos probar que $\Pi(R_1,R_2) = \Pi(R'_1,R'_2)$, ahora bien, dado que anteriormente se demostró que los puntos involucrados en el conjunto de pares no se ven afectados al conmutar las instancias, para demostrar el lema bastará probar que

$$\Pi(R_1,R_2) = \Pi(R'_1,R_2)$$

ya que $\Pi(R'_1,R_2) = \Pi(R_2,R'_1)$

Sean $R_1=(D_1,\tau_1,\mu_1,\alpha_1)$ y $R_2=(D_2,\tau_2,\mu_2,\alpha_2)$, dos instancias arbitrarias.

Sea $R_{1N}=(D_{1N},\hat{\imath},\mu_{1N},\alpha_{1N})$ la instancia normal de la clase de R_1.

Demostraremos que

$$\forall P \in \mathbb{R}^n,\ P \in \Pi(R_1,R_2) \text{ sii } P \in \Pi(R'_1,R_2)$$

supongamos, en primer lugar que $P \in \Pi(R_1,R_2)$, esto implica que

$$\exists\ P_1 \in D_{ef1}, P_2 \in D_{ef2} \text{ con } P = \mathbf{T}_t^{[\tau_1(P_1)]}(\tau_2(P_2))$$

teniendo en cuenta que $D_{1N} = \tau_1(D_{ef1})$, es evidente que

$$\forall\ P_1 \in D_{ef1}\ \exists\ P'_1 \in D_{1N}\ \mid\ P'_1 = \tau_1(P_1)$$

y por tanto

$$\exists\ P'_1 \in D_{ef1}, P_2 \in D_{ef2} \text{ con } P = \mathbf{T}_t^{[P'_1]}(\tau_2(P_2))$$

con lo que $P \in \Pi(R'_1,R_2)$

De un modo similar se demuestra la implicación inversa. □

En base a los conceptos anteriores definimos el producto de dos instancias. La idea del producto de instancias es la de generar un nueva instancia por barrido de una de las instancias sobre la otra, sin cambiar la orientación de ninguna de ellas. El producto se definirá, por tanto, de forma que el dominio de la instancia resultante sea el recorrido de una de las instancias sobre la otra, y que las funciones de aspecto y presencia sean la suma sobre el conjunto de pares producto de los productos de ambas funciones en las dos instancias.

Definición 3.16: El **Producto**, $R_1 \times R_2$, de dos instancias, $R_1=(D_1,\tau_1,\mu_1,\alpha_1)$ y $R_2=(D_2,\tau_2,\mu_2,\alpha_2)$, viene dado por:

$$R_1 \times R_2 = (D,\hat{\imath},\mu,\alpha)$$

siendo:

$$D = \Pi(R_1,R_2)$$

$$\mu(P) = \Sigma_{\delta(R1,R2,P)}\, \mu_1(P_1) \times \mu_2(P_2)$$

$$\alpha(P) = \Sigma_{\delta(R1,R2,P)}\, \alpha_1(P_1) \times \alpha_2(P_2)$$ □

El cálculo del producto conlleva la realización de sumatorias, por tanto, para poder evaluar dicho producto, en principio, el conjunto $\delta(R_1,R_2,P)$ debe de ser finito $\forall P \in \Pi(R_1,R_2)$. No obstante, se puede aplicar el formalismo si los productos de las funciones de presencia y aspecto son integrables en el conjunto de pares. No entraremos aquí en el estudio de la convergencia de dicha sumatoria, y consideraremos que el producto está definido sólo cuando dichas sumatorias sean finitas.

Para que el producto de instancias pueda ser equivalente al producto de objetos es necesario que el producto de instancias sea estable para la equivalencia efectiva. Este hecho queda establecido por el siguiente teorema.

Teorema 3.22: El producto de instancias es estable para la relación de equivalencia efectiva.

Demostración:

Sean

$$R_1 = (D_1,\tau_1,\mu_1,\alpha_1)$$

$$R_2 = (D_2,\tau_2,\mu_2,\alpha_2)$$

cuyo producto es calculable, y viene dado por

$$R_1 \times R_2 = (\Pi(R_1,R_2),\hat{\imath}\ ,\Sigma_{\delta(R1,R2,P)}(\mu_1(P_1)\times\mu_2(P_2)),$$

$$\Sigma_{\delta(R1,R2,P)}(\alpha_1(P_1)\times\alpha_2(P_2))\ )$$

siendo el conjunto de pares producto de R_1 y R_2

$$\delta(R_1,R_2,P) = \{ (P_1,P_2) \in D_{ef1} \times D_{ef2} \mid P=\mathbf{T}_t^{[\tau_1(P_1)]}(\tau_2(P_2))\}$$

Sean $R_1 \cong R_1'$ y $R_2 \cong R_2'$, demostraremos que $R_1 \times R_2 \cong R_1' \times R_2'$.

El segundo producto es calculable, según el lema 3.3.

Será suficiente demostrar que la igualdad se cumple cuando una de las instancias es normal y la otra es arbitraria. Por tanto supondremos que R'_1 y R'_2 son normales

$$R_1' = R_{1N} = (D_{1N},\hat{1},\mu_{1N},\alpha_{1N}) = (D_{ef1}, \hat{1}, \mu_1\cdot\tau_1^{-1}, \alpha_1\cdot\tau_1^{-1})$$

$$R_2' = R_{2N} = (D_{2N},\hat{1},\mu_{2N},\alpha_{2N}) = (D_{ef2}, \hat{1}, \mu_2\cdot\tau_2^{-1}, \alpha_2\cdot\tau_2^{-1})$$

Su producto será

$$R_1' \times R_2' = (\Pi(R'_1,R'_2), \hat{1}, \Sigma_{\delta(R'1,R'2,P)}(\mu_{1N}(P'_1)\times\mu_{2N}(P'_2)),$$

$$\Sigma_{\delta(R'1,R'2,P)}(\alpha_{1N}(P'_1)\times\alpha_{2N}(P'_2))) =$$

$$= (\Pi(R_1,R_2), \hat{1},$$

$$\Sigma_{\delta(R'1,R'2,P)}(\mu_1\cdot\tau_1^{-1}(P'_1)\times\mu_2\cdot\tau_2^{-1}(P'_2)),$$

$$\Sigma_{\delta(R'1,R'2,P)}(\alpha_1\cdot\tau_1^{-1}(P'_1)\times\alpha_2\cdot\tau_2^{-1}(P'_2)))$$

Analicemos los dos últimos términos. El conjunto de pares producto de R'_1 y R'_2 es

$$\delta(R'_1,R'_2,P) = \{ (P'_1,P'_2) \in D_{ef1} \times D_{ef2} \mid P=\mathbf{T}_t^{[P'1]}(P'_2)\}$$

Ahora bien, para los puntos P'_1 y P'_2, existen sendos puntos, P_1 y P_2 pertenecientes a D_1 y D_2 respectivamente, tales que

$$P'_1 = \tau_1(P_1) \quad y \quad P'_2 = \tau_2(P_2)$$

que teniendo en cuenta la definición de conjunto de pares producto, forman un par perteneciente a $\delta(R_1,R_2,P)$. Por otra parte, para los pares, (P_1,P_2) y (P'_1,P'_2) se cumple que

$$\mu_1\cdot\tau_1^{-1}(P'_1) = \mu_1(P_1) \quad \mu_2\cdot\tau_2^{-1}(P'_2) = \mu_2(P_2)$$

$$\alpha_1\cdot\tau_1^{-1}(P'_1) = \alpha_1(P_1) \quad \alpha_2\cdot\tau_2^{-1}(P'_2) = \alpha_2(P_2)$$

con lo que queda demostrado el teorema. □

Teorema 3.23: El producto de instancias es conmutativo y asociativo.

Demostración:

Veremos en primer lugar que el producto de instancias es conmutativo. Sean dos instancias cualesquiera

$$R_1 = (D_1,\tau_1,\mu_1,\alpha_1)$$

$$R_2 = (D_2,\tau_2,\mu_2,\alpha_2)$$

su producto es

$$R_1 \times R_2 = (\Pi(R_1,R_2),\ \hat{\imath},\ \Sigma_{\delta(R1,R2,P)}(\mu_1(P_1)\times\mu_2(P_2)),$$

$$\Sigma_{\delta(R1,R2,P)}(\alpha_1(P_1)\times\alpha_2(P_2))\ )$$

Teniendo en cuenta que $\Pi(R_1,R_2) = \Pi(R_2,R_1)$, que los pares de puntos contenidos $\delta(R_1,R_2,P)$ son los simétricos de los contenidos en $\delta(R_2,R_1,P)$, y que el producto en los espacios de presencia y aspecto es conmutativo, resulta obvio que

$$R_1\times R_2 = R_2\times R_1$$

Veamos que el producto es asociativo. Consideremos una tercera instancia, $R_3 = (D_3,\tau_3,\mu_3,\alpha_3)$, el producto de esta por las dos anteriores, $R'=R_1\times R_2=(D',\tau',\mu',\alpha')$, es:

$$R_3\times(R_1\times R_2) = (\Pi(R_3,R'),\ \hat{\imath},\ \Sigma_{\delta(R3,R',P)}(\mu_3(P_3)\times\mu'(P')),$$

$$\Sigma_{\delta(R3,R',P)}(\alpha_3(P_3)\times\alpha'(P'))\ )$$

Analicemos cada uno de los componentes por separado.

$$\Pi(R_3,R') = \{\ P \in \mathbb{R}^n\ \mid\ \exists\ P_3\in D_{ef3},P'\in D'_{ef}\ \text{con}\ P=\mathbf{T}_t^{[\tau_3(P_3)]}(P')\}$$

con lo que $P = P' + \tau_3(P_3)$

Teniendo en cuenta que

$$D'_{ef} = \Pi(R_1,R_2) = \{\ P'\in\mathbb{R}^n\ \mid\ \exists\ P_1\in D_{ef1},P_2\in D_{ef2}\ \text{con}\ P'=\mathbf{T}_t^{[\tau_1(P_1)]}(\tau_2(P_2))\}$$

se deduce que $P = \tau_1(P_1) + \tau_2(P_2) + \tau_3(P_3)$

por lo que

$$\Pi(R_3,\Pi(R_1,R_2)) = \Pi(\Pi(R_3,R_1),R_2)$$

Por otra parte, para la función de presencia se cumple

$$\Sigma_{\delta(R3,R',P)}(\mu_3(P_3)\times\mu'(P')) = \Sigma_{\delta(R3,R',P)}(\mu_3(P_3)\times[\Sigma_{\delta(R1,R2,P')}(\mu_1(P_1)\times\mu_2(P_2))])$$

$$= \Sigma_{\delta(R3,R',P)}[\Sigma_{\delta(R1,R2,P')}\mu_3(P_3)\times(\mu_1(P_1)\times\mu_2(P_2))]$$

y teniendo en cuenta la igualdad de los recorridos anteriormente establecida queda

$$\Sigma_{\delta(R3,R',P)}(\mu_3(P_3)\times\mu'(P')) =$$

$$= \Sigma_{\{(P1,P2,P3)\ |\ \tau1(P1)+\tau2(P2)+\tau3(P3)=P\}}\ \mu_3(P_3)\times\mu_1(P_1)\times\mu_2(P_2)$$

Del mismo modo se demuestra que las funciones de aspecto coinciden. □

Definiremos, a continuación, el barrido de objetos, para lo que se deberá definir el conjunto de puntos producto de dos objetos.

Definición 3.17: Denominaremos **conjunto de pares producto** de dos objetos, $O_1=(V_1,\mu_1,\alpha_1)$ y $O_2=(V_2,\mu_2,\alpha_2)$, sobre el punto P, y lo denotaremos por $\delta(O_1,O_2,P)$, al conjunto de los pares ordenados de puntos (P_1,P_2) que cumplen:

$$P_1\in V_1, P_2\in V_2 \ \text{ y } \ P=\mathbf{T}_t^{[P1]}(P_2)$$ □

Nótese que la definición anterior hace que dicho conjunto coincida con el obtenido para las instancias normales.

Definición 3.18: El **producto** de dos objetos, $O_1=(V_1,\mu_1,\alpha_1)$ y $O_2=(V_2,\mu_2,\alpha_2)$, está dada por la expresión:

$$O_1 \times O_2 = (V, \mu, \alpha)$$

siendo:

$$V = \{ P \in \mathbf{R}^n \mid \mu(P) \neq 0 \vee \alpha(P) \neq 0 \}$$

$$\mu(P) = \Sigma_{\delta(O1,O2,P)}\ \mu_1(P_1) \times \mu_2(P_2)$$

$$\alpha(P) = \Sigma_{\delta(O1,O2,P)}\ \alpha_1(P_1) \times \alpha_2(P_2)$$ □

La figura 3.1 muestra el resultado del producto de dos líneas. Se aprecia que el volumen del objeto resultante es el mismo que el que se obtendría por barrido de una línea sobre la otra. En la figura $\delta(L_1,L_2,P) = (P_1,P_2)$.

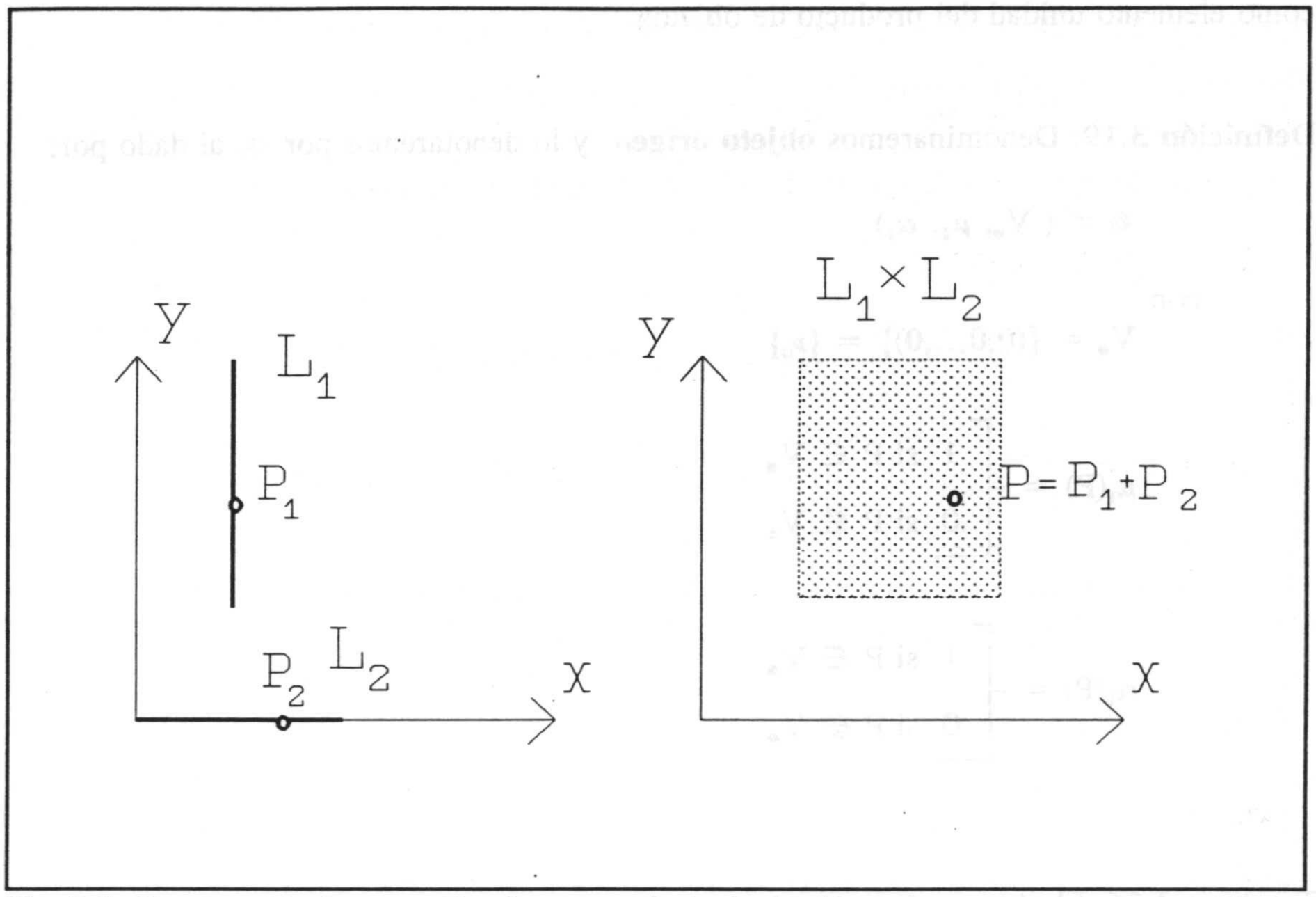


Fig. 3.1. Construcción de un rectángulo por producto de dos líneas.

Para poder operar indistintamente con instancias u objetos, se probará que el resultado del producto de objetos es sinónimo del obtenido por producto de instancias.

Teorema 3.24: El producto de dos objetos es sinónimo del producto de dos instancias cualesquiera sinónimas de ambos objetos.

Demostración:

Trivial. Basta con realizar el producto con las instancias normales de ambos objetos. □

Teorema 3.25: El producto de objetos es conmutativo y asociativo.

Demostración:

Trivial. Una vez demostrado que el producto de instancias satisface dichas propiedades. □

El objeto constituido por el origen de coordenadas con presencia y color uno, actúa

como elemento unidad del producto de objetos.

Definición 3.19: Denominaremos **objeto origen**, y lo denotaremos por ⊚, al dado por:

$$\circledcirc = (V_{\bullet}, \mu_1, \alpha_1)$$

con

$$V_{\bullet} = \{(0,0,..,0)\} = \{\nu_0\}$$

$$\mu_1(P) = \begin{cases} 1 & \text{si } P \in V_{\bullet} \\ 0 & \text{si } P \notin V_{\bullet} \end{cases}$$

$$\alpha_1(P) = \begin{cases} 1 & \text{si } P \in V_{\bullet} \\ 0 & \text{si } P \notin V_{\bullet} \end{cases}$$

□

Teorema 3.26: El objeto origen actúa como elemento neutro respecto al producto de objetos.

Demostración:

Realicemos el producto del origen con un objeto cualquiera $O_2 = (V_2,\mu_2,\alpha_2)$

$$O_2 \times \circledcirc = (V, \Sigma_{\delta(O2,\bullet,P)} \mu_2(P_2) \times \mu_1(P_{\bullet}),$$

$$\Sigma_{\delta(O2,\bullet,P)} \alpha_2(P_2) \times \alpha_1(P_{\bullet})$$

con $V = \{ P \in \mathbb{R}^n \mid \mu(P) \neq 0 \vee \alpha(P) \neq 0)$

Es fácil comprobar que el conjunto de pares producto de ambos objetos es:

$$\delta(O_2, \circledcirc, P) = \{ (P_2,\nu_0) \mid P_2 \in V_2 \text{ y } P_2 = P \}$$

por tanto las funciones de aspecto y presencia son las de O_2, quedando probado que ⊚ actúa como elemento neutro del producto. □

El producto de un objeto cualquiera por un objeto puntual, con presencia y aspecto unidad, equivale a la traslación del objeto. Un objeto puntual, con dichos valores de presencia y aspecto, se puede obtener por traslación del objeto origen. Este hecho queda establecido por el siguiente teorema.

Teorema 3.27: Para un objeto, $O = (V, \mu, \alpha)$, y un punto, $P \in \mathbf{R}^n$, cualesquiera, se cumple que:

$$\mathbf{T}_t^{[P]}(\circledcirc) \times O = \mathbf{T}_t^{[P]}(O)$$

Demostración:

Calculemos ambas expresiones sobre las instancias normales sinónimas a los dos objetos:

$$R = (D, \hat{\imath}, \mu, \alpha) \quad \text{con } D = V$$

y

$$R_\bullet = (\{v_0\}, \hat{\imath}, \mu_1, \alpha_1)$$

El término $\mathbf{T}_t^{[P]}(\circledcirc) \times O$ es el objeto sinónimo a la instancia $\mathbf{T}_t^{[P]}(R_\bullet) \times R$. Para calcularla debemos determinar previamente el barrido y conjunto de puntos producto de ambas instancias, que son:

$$\delta(\mathbf{T}_t^{[P]}(R_\bullet), R, Q) = \{(P, Q-P)\} \text{ si } Q-P \in D_{efR} \text{ y } \{\phi\} \text{ si } Q-P \notin D_{efR}$$

y por tanto

$$\Pi(\mathbf{T}_t^{[P]}(R_\bullet), R) = D_{efR} + P$$

Con lo que la instancia a calcular es:

$$\mathbf{T}_t^{[P]}(R_\bullet) \times R = (D_{efR} + P, \hat{\imath}, \mu \cdot \mathbf{T}^{-1}{}_t^{[P]}, \alpha \cdot \mathbf{T}^{-1}{}_t^{[P]})$$

ya que las funciones de presencia y aspecto vienen dadas por la expresión:

$$\mu'(Q) = \Sigma_{\delta(R\bullet, R, Q)}\, \mu_1(P) \times \mu(Q-P) = \mu(Q-P) = \mu(\mathbf{T}^{-1}{}_t^{[P]}(Q))$$

Que coincide con la expresión para el segundo término. □

Este resultado permite realizar las traslaciones de objetos como productos de objetos.

Definición 3.20: Dados dos objetos cualesquiera, diremos que son divisibles si existe un tercer objeto que multiplicado por el primero genera el segundo. □

En general dos objetos no serán divisibles. Concretamente el origen sólo es divisible por objetos puntuales, por lo que, en general, los objetos no poseen inverso respecto del producto.

3.7 PRODUCTO CIRCULAR.

En esta sección se definirá otra operación, que denominaremos producto circular de objetos, o barrido circular, que permitirá representar la generación de objetos por barrido.

El objeto generado por producto circular es el objeto que se obtiene por giro del primer objeto según el segundo realizando previamente un escalado dado por la posición del primer objeto. El objeto generado no coincide, necesariamente, con el que se generaría por barrido circular, no obstante eligiendo el objeto trayectoria adecuado se pueden generar los mismos objetos. Por tanto, está operación no coincide con el barrido rotacional habitualmente utilizado [Requ82], pero con una elección adecuada de los objetos puede generar los mismos resultados.

En general, esta operación incrementará la dimensión del objeto. En el caso particular de que uno de los objetos sea un punto el producto equivaldrá a la realización de un giro y un escalado.

Antes de definir la operación de barrido circular de instancias, será necesario, al igual que se hizo para el producto, introducir conceptos auxiliares. Concretamente se definirá el conjunto de pares circulares y el recorrido circular.

Definición 3.21: Denominaremos **conjunto circular de pares producto** de dos instancias, $R_1=(D_1,\tau_1,\mu_1,\alpha_1)$ y $R_2=(D_2,\tau_2,\mu_2,\alpha_2)$, sobre el punto P, y lo denotaremos por $\delta_c(R_1,R_2,P)$, al conjunto de los pares ordenados de puntos (P_1,P_2) que cumplen:

$$P_1 \in D_{ef1}, P_2 \in D_{ef2} \text{ y } P=\mathbf{T}_c^{[\tau_1(P_1)]}(\tau_2(P_2)) \qquad \square$$

$\tau_1(P_1)$ es un punto del objeto sinónimo de R_1, y $\tau_2(P_2)$ es un punto del objeto sinónimo de R_2. El conjunto circular de pares producto es por tanto el conjunto de pares ordenados de puntos de las dos instancias (P_1,P_2), tales que al realizar una transformación circular de magnitud $\tau_1(P_1)$, del punto $\tau_2(P_2)$, se obtiene el punto de referencia P.

Lema 3.4: Sean R_1 y R_2 dos instancias cualesquiera y $P \in \mathbf{R}^n$ arbitrario, se verifica que:

$$(P_1,P_2) \in \delta_c(R_1,R_2,P) \text{ sii } (P_2,P_1) \in \delta_c(R_2,R_1,P)$$

Demostración:

Sean $R_1=(D_1,\tau_1,\mu_1,\alpha_1)$ y $R_2=(D_2,\tau_2,\mu_2,\alpha_2)$, dos instancias arbitrarias y P un punto cualquiera de $\mathbf{R}^n$. El conjunto circular de pares producto $\delta_c(R_1,R_2,P)$, viene dado por:

$\delta_c(R_1,R_2,P)=\{(P_1,P_2) \mid P_1 \in D_{ef1}, P_2 \in D_{ef2}$ y $P=T_c^{[\tau 1(P1)]}(\tau_2(P_2))\}$

dado que $T_c^{[\tau 1(P1)]}(\tau_2(P_2)) = T_c^{[\tau 2(P2)]}(\tau_1(P_1))$

resulta evidente que

$\forall (P_1,P_2) \in \delta_c(R_1,R_2,P)$ se cumple que

$P_2 \in D_{ef2}, P_1 \in D_{ef1}$ y $P=T_c^{[\tau 2(P2)]}(\tau_1(P_1))$

y por tanto $(P_2,P_1) \in \delta_c(R_2,R_1,P)$

La implicación inversa es obvia. □

Una consecuencia inmediata de este lema es el hecho de que $\delta_c(R_1,R_2,P)$ y $\delta_c(R_2,R_1,P)$ tengan la misma cardinalidad.

Dadas dos instancias, podemos pensar en el conjunto de puntos del espacio sobre los que el conjunto circular de pares producto de ambas instancias es no vacío. Este concepto se introduce formalmente en la siguiente definición.

Definición 3.22: Denominaremos **recorrido circular** de una instancia, $R_1=(D_1,\tau_1,\mu_1,\alpha_1)$, sobre otra instancia, $R_2=(D_2,\tau_2,\mu_2,\alpha_2)$, al volumen formado por el conjunto de puntos P, tales que $\delta_c(R_1,R_2,P)$ es no vacío.

$$\Pi_c(R_1,R_2) = \{ P \in \mathbb{R}^n \mid \delta_c(R_1,R_2,P) \neq \phi \}$$ □

Resulta evidente, en virtud del lema 3.4, que $\Pi_c(R_1,R_2)$ coincide con $\Pi_c(R_2,R_1)$.

Teniendo en consideración la definición de conjunto circular de pares de puntos producto, podemos caracterizar al recorrido circular del siguiente modo.

Lema 3.5: El recorrido circular de una instancia, $R_1=(D_1,\tau_1,\mu_1,\alpha_1)$, sobre otra instancia, $R_2=(D_2,\tau_2,\mu_2,\alpha_2)$, viene dado por:

$\Pi_c(R_1,R_2)= \{ P \in \mathbb{R}^n \mid \exists P_1 \in D_{ef1}, P_2 \in D_{ef2}$ con $P=T_c^{[\tau 1(P1)]}(\tau_2(P_2))\}$

Demostración:

Trivial. □

Para que el concepto anterior sea realmente útil, para definir operaciones sobre objetos, dicho conjunto debe de ser el mismo para todas las instancias sinónimas de un mismo

objeto. Este hecho queda establecido por el siguiente lema.

Lema 3.6: Sea $\Pi_c(R_1,R_2)$ el recorrido circular de dos instancias, R_1 y R_2. Para cualquier par de instancias R'_1 y R'_2 efectivamente equivalentes a las anteriores, $R'_1 \cong R_1$ y $R'_2 \cong R_2$, se cumple que

$$\Pi_c(R_1,R_2)=\Pi_c(R'_1,R'_2)$$

Demostración:

Dado que anteriormente se demostró que los puntos involucrados en el conjunto circular de pares no se ven afectados al conmutar las instancias, para demostrar, el lema bastará con demostrar que

$$\Pi_c(R_1,R_2) = \Pi_c(R'_1,R_2)$$

Sean $R_1=(D_1,\tau_1,\mu_1,\alpha_1)$ y $R_2=(D_2,\tau_2,\mu_2,\alpha_2)$, dos instancias de objetos arbitrarias.

Sea $R_{1N}=(D_{1N},\hat{I},\mu_{1N},\alpha_{1N})$ la instancia normal de la clase de R_1. Demostraremos que

$$\forall P \in \mathbb{R}^n \quad P \in \Pi_c(R_1,R_2) \text{ sii } P \in \Pi_c(R'_1,R_2)$$

supongamos, en primer lugar que $P \in \Pi_c(R_1,R_2)$, esto implica que

$$\exists\, P_1 \in D_{ef1}, P_2 \in D_{ef2} \text{ con } P=\mathbf{T}_c^{[\tau_1(P_1)]}(\tau_2(P_2))$$

teniendo en cuenta que $D_{1N} = \tau_1(D_{ef1})$, es evidente que

$$\forall\, P_1 \in D_{ef1} \;\; \exists\, P'_1 \in D_{1N} \mid P'_1 = \tau_1(P_1)$$

y por tanto

$$\exists\, P'_1 \in D_{1N}, P_2 \in D_{ef2} \text{ con } P=\mathbf{T}_c^{[P'_1]}(\tau_2(P_2))$$

con lo que $P \in \Pi_c(R'_1,R_2)$

De un modo similar se demuestra la implicación inversa. □

En base a los conceptos anteriores definiremos el producto circular de dos instancias. El producto se definirá de forma que el dominio de la instancia resultante sea el recorrido circular de una de las instancias sobre la otra, y que las funciones de aspecto y presencia sean la suma sobre el conjunto de pares producto de los productos de ambas funciones en las dos instancias.

Comentarios:

A diferencia del producto simple, el producto circular se definirá de tal modo que se eviten determinadas singularidades, producidas por sumas infinitas, dado que, al actuar el origen como elemento absorvente para la transformación circular, cualquier objeto que contenga dicho punto producirá factores con sumas infinitas. La singularidad se evitará tan solo en el caso de que los distintos términos que intervengan en la sumatoria infinita sean iguales, sustituyéndose la sumatoria por un término. De este modo se mantienen las propiedades de asociatividad, conmutatividad y existencia de elemento unidad.

Definición 3.23: El **Producto circular** de dos instancias, $R_1=(D_1,\tau_1,\mu_1,\alpha_1)$ y $R_2=(D_2,\tau_2,\mu_2,\alpha_2)$, viene dado por:

$$R_1 \otimes R_2 = (D,\hat{\imath},\mu,\alpha)$$

siendo:

$$D = \Pi_c(R_1,R_2)$$

y las funciones de aspecto y presencia

$$\mu(P) = \Sigma_{\delta c(R1,R2,P)}\, \mu_1(P_1) \times \mu_2(P_2)$$

$$\alpha(P) = \Sigma_{\delta c(R1,R2,P)}\, \alpha_1(P_1) \times \alpha_2(P_2)$$

si $\delta_c(O_1,O_2,P)$ es finito en todos los puntos.

Si existe algún punto, P, en que dicho conjunto es no finito, el producto es calculable solamente si para todos los puntos con el conjunto circular de puntos producto no finito, $\mu_1(P_1)$, $\mu_2(P_2)$, $\alpha_1(P_1)$ y $\alpha_2(P_2)$ tienen el mismo valor, para todos los pares en $\delta_c(O_1,O_2,P)$. Es decir

$$\forall(P_1,P_2)\in\delta_c(O_1,O_2,P) => \mu_1(P_1)=\mu_{o1};\ \mu_2(P_2)=\mu_{o2};\alpha_1(P_1)=\alpha_{o1};\ \alpha_2(P_2)=\alpha_{o2}$$

En este caso el valor de las funciones en P es:

$$\mu(P) = \mu_1(P_1) \times \mu_2(P_2)$$

$$\alpha(P) = \alpha_1(P_1) \times \alpha_2(P_2) \text{ para cualquier } (P_1,P_2) \in \delta_c(O_1,O_2,P)$$ □

El cálculo del producto conlleva la realización de una suma, por tanto, para poder evaluar dicho producto, en principio, el conjunto de pares debe de ser finito, en este caso, no obstante la definición anterior contempla la posibilidad de que el producto sea infinito solo en el caso de que el valor de aspecto y presencia sea el mismo en todos los puntos involucrados, dentro de cada instancia.

Para que el resultado de la operación no dependa de la instancia elegida, interesa que el producto circular de instancias sea estable para la equivalencia efectiva. Este hecho queda establecido por el siguiente teorema.

Teorema 3.28: El producto circular de instancias es estable para la relación de equivalencia efectiva.

Demostración:

Sean

$$R_1 = (D_1,\tau_1,\mu_1,\alpha_1)$$

$$R_2 = (D_2,\tau_2,\mu_2,\alpha_2)$$

cuyo producto circular es

$$R_1 \otimes R_2 = (\Pi(R_1,R_2),\hat{\imath}, \Sigma_{\delta c(R1,R2,P)}(\mu_1(P_1)\times\mu_2(P_2)),$$

$$\Sigma_{\delta c(R1,R2,P)}(\alpha_1(P_1)\times\alpha_2(P_2)))$$

siendo el conjunto de pares producto de R_1 y R_2 sobre P

$$\delta_c(R_1,R_2,P) = \{ (P_1,P_2) \in D_{ef1} \times D_{ef2} \mid P=T_c^{[\tau_1(P_1)]}(\tau_2(P_2))\}$$

Sean $R_1 \cong R_1'$ y $R_2 \cong R_2'$, demostraremos que $R_1 \otimes R_2 \cong R_1' \otimes R_2'$.

Será suficiente demostrar que la propiedad se cumple cuando, en cada par equivalente, una de las instancias es normal y la otra arbitraria. Por tanto supondremos que R'_1 y R'_2 son normales, por tanto iguales a

$$R_1' = R_{1N} = (D_{1N},\hat{\imath},\mu_{1N},\alpha_{1N}) = (D_{ef1}, \hat{\imath}, \mu_1\cdot\tau_1^{-1}, \alpha_1\cdot\tau_1^{-1})$$

$$R_2' = R_{2N} = (D_{2N},\hat{\imath},\mu_{2N},\alpha_{2N}) = (D_{ef2}, \hat{\imath}, \mu_2\cdot\tau_2^{-1}, \alpha_2\cdot\tau_2^{-1})$$

Su producto circular será

$$R_1' \otimes R_2' = (\Pi(R'_1,R'_2),\hat{\imath}, \Sigma_{\delta c(R'1,R'2,P)}(\mu_{1N}(P'_1)\times\mu_{2N}(P'_2)),\Sigma_{\delta c(R'1,R'2,P)}(\alpha_{1N}(P'_1)\times\alpha_{2N}(P'_2))) =$$

$$= (\Pi(R_1,R_2), \hat{\imath}, \Sigma_{\delta c(R'1,R'2,P)}(\mu_1\cdot\tau_1^{-1}(P'_1)\times\mu_2\cdot\tau_2^{-1}(P'_2)),$$

$$\Sigma_{\delta c(R'1,R'2,P)}(\alpha_1\cdot\tau_1^{-1}(P'_1)\times\alpha_2\cdot\tau_2^{-1}(P'_2)))$$

Analicemos los dos último términos. El conjunto circular de pares producto de R'_1 y R'_2 es

$$\delta_c(R'_1,R'_2,P) = \{ (P'_1,P'_2) \in D_{ef1} \text{ X } D_{ef2} \mid P=T_c^{[P'1]}(P'_2)\}$$

Ahora bien, para los puntos P'_1 y P'_2, existen sendos puntos, P_1 y P_2 pertenecientes a D_1 y D_2 respectivamente, tales que

$$P'_1 = \tau_1(P_1) \quad y \quad P'_2 = \tau_2(P_2)$$

que teniendo en cuenta la definición de conjunto circular de pares producto forman un par perteneciente a $\delta_c(R_1,R_2,P)$. Por otra parte, para dichos pares, (P_1,P_2) y (P'_1,P'_2) se cumple que

$$\mu_1 \cdot \tau_1^{-1}(P'_1) = \mu_1(P_1) \quad \mu_2 \cdot \tau_2^{-1}(P'_2) = \mu_2(P_2)$$

$$\alpha_1 \cdot \tau_1^{-1}(P'_1) = \alpha_1(P_1) \quad \alpha_2 \cdot \tau_2^{-1}(P'_2) = \alpha_2(P_2)$$

con lo que queda demostrado el teorema. □

Teorema 3.29: El producto circular de instancias es conmutativo y asociativo.

Demostración:

Veremos en primer lugar que el producto circular de instancias es conmutativo. Sean dos instancias cualesquiera

$$R_1 = (D_1,\tau_1,\mu_1,\alpha_1)$$

$$R_2 = (D_2,\tau_2,\mu_2,\alpha_2)$$

su producto circular es

$$R_1 \otimes R_2 = (\Pi(R_1,R_2),\ \hat{\imath},\ \Sigma_{\delta c(R1,R2,P)}(\mu_1(P_1)\times\mu_2(P_2)),$$
$$\Sigma_{\delta c(R1,R2,P)}(\alpha_1(P_1)\times\alpha_2(P_2))\ )$$

Teniendo en cuenta que $\Pi_c(R_1,R_2) = \Pi_c(R_2,R_1)$, que los pares de puntos contenidos $\delta_c(R_1,R_2,P)$ son los simétricos de los contenidos en $\delta_c(R_2,R_1,P)$, y que el producto en los espacios de presencia y aspecto es conmutativo, resulta obvio que

$$R_1 \otimes R_2 = R_2 \otimes R_1$$

Veamos que el producto es asociativo. Consideremos una tercera instancia, $R_3 = (D_3,\tau_3,\mu_3,\alpha_3)$, el producto de esta por las dos anteriores, $R' = R_1 \otimes R_2 = (D',\tau',\mu',\alpha')$, es:

$$R_3 \otimes (R_1 \otimes R_3) = (\Pi_c(R_3,R'),\ \hat{\imath},\ \Sigma_{\delta c(R3,R',P)}(\mu_3(P_3)\times\mu'(P')),$$
$$\Sigma_{\delta c(R3,R',P)}(\alpha_3(P_3)\times\alpha'(P'))\ )$$

Analicemos cada uno de los componentes por separado.

$\Pi_c(R_3,R') = \{ P \in \mathbf{R}^n \mid \exists P_3 \in D_{ef3}, P' \in D'_{ef} \text{ con } P = T_c^{[\tau 3(P3)]}(P') \}$

teniendo en cuenta que

$D'_{ef} = \Pi_c(R_1,R_2) = \{ P' \in \mathbf{R}^n \mid \exists P_1 \in D_{ef1}, P_2 \in D_{ef2} \text{ con } P' = T_c^{[\tau 1(P1)]}(\tau_2(P_2)) \}$

se deduce que $P = T_c^{[\tau 3(P3)]}(T_c^{[\tau 1(P1)]}(\tau_2(P_2))$

y dado que la composición de transformaciones geométricas es asociativa se cumple que

$\Pi_c(R_3,R') = \Pi_c(\Pi_c(R_3,R_1),R_2)$

Por otra parte, para la función de presencia se cumple

$$\Sigma_{\delta c(R3,R',P)}(\mu_3(P_3) \times \mu'(P')) = \Sigma_{\delta c(R3,R',P)}(\mu_3(P_3) \times [\Sigma_{\delta c(R1,R2,P)}(\mu_1(P_1) \times \mu_2(P_2))])$$

$$= \Sigma_{\delta c(R3,R',P)}[\Sigma_{\delta c(R1,R2,P')}\mu_3(P_3) \times (\mu_1(P_1) \times \mu_2(P_2))]$$

y teniendo en cuenta la igualdad de los recorridos anteriormente establecida queda

$$\Sigma_{\delta c(R3,R',P)}(\mu_3(P_3) \times \mu'(P')) =$$

$$= \Sigma_{\{(P1,P2,P3) \mid P = Tc[\tau 3(P3)](Tc[\tau 1(P1)](\tau 2(P2))\}} \mu_3(P_3) \times \mu_1(P_1) \times \mu_2(P_2)$$

En el caso de que exista alguna singularidad, de las contempladas en la definición, la igualdad sigue siendo valida, suprimiendo la sumatoria correspondiente. Del mismo modo se demuestra que las funciones de aspecto coinciden. □

Definiremos ahora el producto circular de objetos. Para ello se extenderá el concepto de conjunto circular de pares de puntos producto.

Definición 3.24: Denominaremos **conjunto circular de pares producto** de dos objetos, $O_1=(V_1,\mu_1,\alpha_1)$ y $O_2=(V_2,\mu_2,\alpha_2)$, sobre el punto P, y lo denotaremos por $\delta_c(O_1,O_2,P)$, al conjunto de los pares ordenados de puntos (P_1,P_2) que cumplen:

$$P_1 \in V_1, P_2 \in V_2 \text{ y } P = T_c^{[P1]}(P_2)$$ □

Nótese que la definición anterior hace que dicho conjunto coincida con el obtenido para las instancias normales.

Definición 3.25: El producto circular de dos objetos $O_1=(V_1,\mu_1,\alpha_1)$ y $O_2=(V_2,\mu_2,\alpha_2)$ está dada por la expresión:

$$O_1 \otimes O_2 = (V, \mu, \alpha)$$

siendo:

$$V = \{ P \in \mathbb{R}^n \mid \mu(P) \neq 0 \vee \alpha(P) \neq 0 \}$$

y

$$\mu(P) = \Sigma_{\delta c(O1,O2,P)} \mu_1(P_1) \times \mu_2(P_2)$$

$$\alpha(P) = \Sigma_{\delta c(O1,O2,P)} \alpha_1(P_1) \times \alpha_2(P_2)$$

si $\delta_c(O_1,O_2,P)$ es finito en todos los puntos.

Si existe algún punto, P, en que dicho conjunto es no finito, el producto es calculable solamente si para todos los puntos con el conjunto circular de puntos producto no finito, $\mu_1(P_1)$, $\mu_2(P_2)$, $\alpha_1(P_1)$ y $\alpha_2(P_2)$ tienen el mismo valor, para todos los pares en $\delta_c(O_1,O_2,P)$. Es decir

$$\forall (P_1,P_2) \in \delta_c(O_1,O_2,P) \Rightarrow \mu_1(P_1)=\mu_{o1};\ \mu_2(P_2)=\mu_{o2};\alpha_1(P_1)=\alpha_{o1};\ \alpha_2(P_2)=\alpha_{o2}$$

En este caso el valor de las funciones en P es:

$$\mu(P) = \mu_1(P_1) \times \mu_2(P_2)$$

$$\alpha(P) = \alpha_1(P_1) \times \alpha_2(P_2) \text{ para cualquier } (P_1,P_2) \in \delta_c(O_1,O_2,P)$$ □

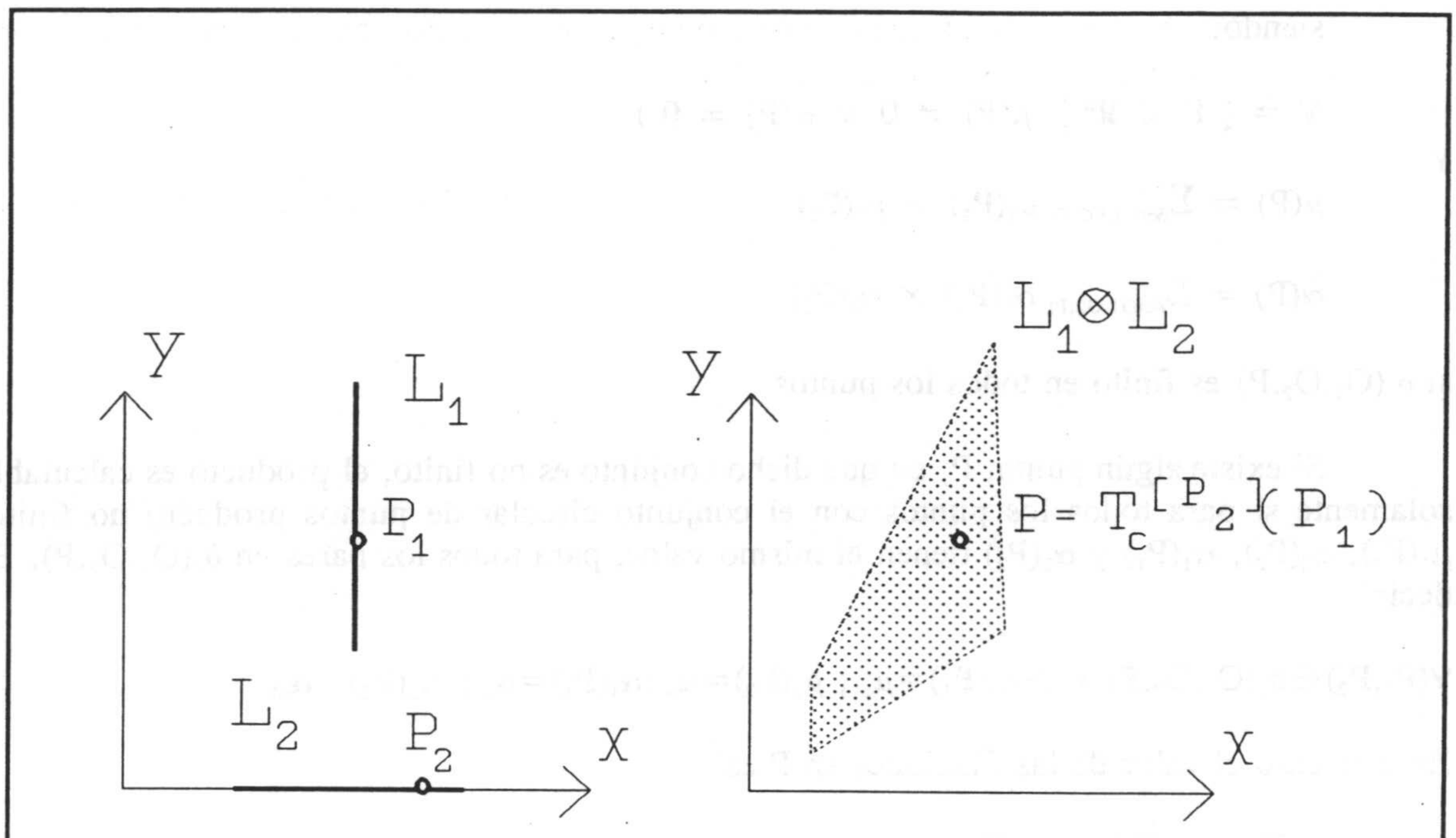


Fig. 3.2. Producto circular de dos líneas.

La figura 3.2 muestra el producto circular de dos líneas. Se observa que los lados superior e inferior del polígono son radiales.

El siguiente teorema permite el cálculo del producto circular de dos objetos usando dos instancias sinónimas.

Teorema 3.30: El producto circular de dos objetos es sinónimo de la instancia resultante de realizar el producto circular de dos instancias cualesquiera sinónimas de los objetos.

Demostración:

Trivial. Basta con realizar el producto con las instancias normales de ambos objetos. □

Teorema 3.31: El producto circular de objetos es conmutativo y asociativo.

Demostración:

Trivial. Una vez demostrado que el producto de instancias satisface dichas propiedades. □

Definiremos, a continuación, un objeto singular, el punto unidad, que se demostrará actúa como elemento neutro del producto circular.

Definición 3.25: Denominaremos **objeto punto unidad**, y lo denotaremos por $\mathbf{O}_\blacktriangleright$, al dado por:

$$\mathbf{O}_\blacktriangleright = (V_\blacktriangleright, \mu_1, \alpha_1)$$

con

$$V_\blacktriangleright = \{\nu_1\} = \{ (1,0,..,0) \} \text{ en coordenadas esféricas}$$

$$\mu_1(P) = \begin{cases} 1 & \text{si } P \in V_\blacktriangleright \\ 0 & \text{si } P \notin V_\blacktriangleright \end{cases}$$

$$\alpha_1(P) = \begin{cases} 1 & \text{si } P \in V_\blacktriangleright \\ 0 & \text{si } P \notin V_\blacktriangleright \end{cases}$$

□

Teorema 3.32: El objeto punto unidad actúa como elemento neutro respecto al producto circular de objetos.

Demostración:

Realicemos el producto del punto unidad por un objeto cualquiera $O_2 = (V,\mu,\alpha)$

$$O_2 \otimes \mathbf{O}_\blacktriangleright = (V, \Sigma_{\delta c(O2,\blacktriangleright,P)}\, \mu_2(P_2) \times \mu_1(P_\blacktriangleright),$$

$$\Sigma_{\delta c(O2,\blacktriangleright,P)}\, \alpha_2(P_2) \times \alpha_1(P_\blacktriangleright)$$

con $V = \{ P \in \mathbf{R}^n \mid \mu(P) \neq 0 \vee \alpha(P) \neq 0)$

Teniendo en cuenta que el único punto del dominio de $V_\blacktriangleright$ es el dado, en coordenadas esféricas por

$$P_\blacktriangleright = (1,0,0,..,0)$$

es fácil comprobar que el conjunto circular de pares producto de ambos objetos es:

$$\delta_c(O_2,\mathbf{O}_\blacktriangleright,P) = \{ (P_2,\nu_1) \mid P_2 \in V_2 \text{ y } P_2 = P \}$$

por tanto las funciones de aspecto y presencia son las de O_2, quedando probado que $\mathbf{O}_\blacktriangleright$ actúa como elemento neutro del producto. □

El producto circular de un objeto cualquiera por otro puntual, con presencia y aspecto unidad, equivale a realizar una transformación circular del objeto según las coordenadas del objeto puntual. Un objeto puntual situado en la posición P se puede obtener por transformación circular del objeto punto unidad según P.

Teorema 3.33: Para un objeto cualquiera, $O = (V, \mu, \alpha)$, y un punto, $P \in \mathbf{R}^n$, cualesquiera, se cumple que:

$$\mathbf{T}_c^{[P]}(O_\blacktriangleright) \otimes O = \mathbf{T}_c^{[P]}(O)$$

Demostración:

Calculemos ambas expresiones sobre las instancias normales sinónimas de los dos objetos:

$$R = (D, \hat{\imath}, \mu, \alpha) \quad \text{con } D = V$$

y

$$R_\blacktriangleright = (\{\nu_1\}, \hat{\imath}, \mu_1, \alpha_1)$$

El término $\mathbf{T}_c^{[P]}(O_\blacktriangleright) \otimes O$ es el objeto sinónimo de la instancia $\mathbf{T}_c^{[P]}(R_\blacktriangleright) \otimes R$. Para calcularla, debemos determinar previamente el barrido circular y conjunto circular de puntos producto de ambas instancias, que son:

$$\delta_c(\mathbf{T}_c^{[P]}(R_\blacktriangleright), R, Q) = \{(P, \mathbf{T}^{-1}{}_c^{[P]}(Q)\} \text{ si } \mathbf{T}^{-1}{}_c^{[P]}(Q) \in D_{efR} \text{ y } \{\phi\} \text{ si } \mathbf{T}^{-1}{}_c^{[P]}(Q) \notin D_{efR}$$

y por tanto

$$\Pi_c(\mathbf{T}_c^{[P]}(R_\blacktriangleright), R) = \mathbf{T}^{-1}{}_c^{[P]}(D_{efR})$$

Con lo que la instancia a calcular es:

$$\mathbf{T}_c^{[P]}(R_\blacktriangleright) \otimes R = (\mathbf{T}^{-1}{}_c^{[P]}(D_{efR}), \hat{\imath}, \mu \cdot \mathbf{T}^{-1}{}_c^{[P]}, \alpha \cdot \mathbf{T}^{-1}{}_c^{[P]})$$

ya que las funciones de presencia y aspecto vienen dadas por la expresión:

$$\mu'(Q) = \Sigma_{\delta c(R\blacktriangleright, R, Q)}\, \mu_1(P) \otimes \mu(\mathbf{T}^{-1}{}_c^{[P]}(Q)) = \mu(\mathbf{T}^{-1}{}_c^{[P]}(Q))$$

Que obviamente coincide con la expresión para el segundo término. □

Corolario 3.3: El producto de un objeto cualquiera por un objeto puntual situado sobre la esfera unidad equivale al giro del objeto. Si el objeto puntual está sobre el primer eje de coordenadas el producto circular equivaldrá a un escalado del objeto.

Demostración:

Trivial. Basta con utilizar el resultado del teorema 3.33, teniendo en cuenta la definición 3.23, de transformación circular.

Definición 3.26: Dados dos objetos cualesquiera, diremos que son divisibles circularmente si existe un tercer objeto cuyo producto circular con el primero genera el segundo. □

En general dos objetos no serán divisibles. Concretamente el objeto punto unidad sólo es divisible por objetos puntuales, por lo que, en general, los objetos no poseen inverso respecto del producto.

3.9 OTRAS PROPIEDADES DEL ALGEBRA DE OBJETOS.

Con las operaciones definidas anteriormente se ha dotado al conjunto de las siguientes estructuras algebraicas:

(O, +) es un grupo abeliano

(O, +, *) es un espacio vectorial

(O, ∪, ∩, ~) es un álgebra de boole

(O, ×) posee elemento neutro, y la ley es asociativa y conmutativa

(O, ⊗) posee elemento neutro, y la ley es asociativa y conmutativa

Los dos productos internos no son distributivos respecto de la suma. Esta propiedad se cumple, en principio, sólo cuando los objetos no se solapan. No obstante, cuando los productos en los dominios de aspecto y presencia sean distributivos respecto de la suma, los productos de objetos también lo serán.

Se han introducido algunos objetos importantes, por su comportamiento antes estas operaciones: objeto vacío, objeto nulo, punto origen, objeto universo y punto unidad. Nos ocuparemos seguidamente de otros objetos notables cuyo comportamiento, es importante geométricamente: arcos y segmentos.

Definición 3.27: Denominaremos **línea**[1] a todo objeto definido por:

$$\text{línea}(P_1,P_2) = (V, \mu, \alpha);$$

siendo:

$$V = \{ u\cdot P_1 + (u-1)\cdot P_2 \mid u \in [0,1] \cdot P_1,P_2 \in \mathbf{R}^n\}$$

$$\mu(P) = \begin{cases} 1 \text{ si } P \in V \\ 0 \text{ si } P \notin V \end{cases}$$

□

El producto de dos líneas, cuando es calculable, genera una superficie: paralelogramo o triángulo.

---

[1] De acuerdo con la terminología habitualmente utilizada en Informática Gráfica, denominaremos líneas a los segmentos de recta.

Teorema 3.34: El producto de dos líneas, cuando es calculable, genera un paralelogramo.

Demostración:

Sean L_1 y L_2 dos líneas definidas como:

L_1 = Línea(Q_{11},Q_{12}), L_2 = Línea(Q_{21},Q_{22})

1. Su producto lineal viene dado por

$$L_1 \times L_2 = (V, \mu, \alpha)$$

siendo:

$$V = \{ P \in \mathbb{R}^n \mid \mu(P) \neq 0 \vee \alpha(P) \neq 0 \}$$

$$\mu(P) = \Sigma_{\delta(L1,L2,P)}\, \mu_1(P_1) \times \mu_2(P_2)$$

$$\alpha(P) = \Sigma_{\delta(L1,L2,P)}\, \alpha_1(P_1) \times \alpha_2(P_2)$$

Calculemos su conjunto de puntos producto:

$$\delta(L_1,L_2,P) = \{(P_1,P_2) \mid P_1 \in V_1, P_2 \in V_2 \text{ y } P = T_t^{[P\,1]}(P_2)\}$$

teniendo en cuenta la definición de línea tenemos

$$P_1 = \{ u \cdot Q_{11} + (u\text{-}1) \cdot Q_{12} \mid u \in [0,1] \}$$

$$P_2 = \{ v \cdot Q_{21} + (v\text{-}1) \cdot Q_{22} \mid v \in [0,1] \}$$

queda

$$\delta(L_1,L_2,P) = \{(P_1,P_2) \mid P_1 \in V_1, P_2 \in V_2,$$

$$P = u \cdot Q_{11} + (u\text{-}1) \cdot Q_{12} + v \cdot Q_{21} + (v\text{-}1) \cdot Q_{22},\ u,v \in [0,1]\}$$

Cuando las dos líneas no sean paralelas, dicho conjunto tendrá un único elemento para cada valor de P, y por tanto será finito, dado que en dicho caso la ecuación anterior tendrá solución única en u y v.

Y dado que μ_1 y μ_2 son constantes e iguales a uno, el volumen será

$$V = \{ u \cdot Q_{11} + (u\text{-}1) \cdot Q_{12} + v \cdot Q_{21} + (v\text{-}1) \cdot Q_{22},\ u,v \in [0,1]\}$$

con lo que es inmediato comprobar que el objeto es un paralelogramo, con valores de presencia y aspecto dados por

$$\mu(P) = \begin{cases} 1 \text{ si } P \in V \\ 0 \text{ si } P \notin V \end{cases}$$

$$\alpha(P) = \alpha_1(P)\cdot\alpha_2(P)\cdot\mu(P)$$

Es inmediato comprobar que los extremos del paralelogramo están en las coordenadas obtenidas por suma de los vértices de las líneas.

2. Calculemos ahora el producto circular:

$$L_1 \otimes L_2 = (V, \mu, \alpha)$$

siendo:

$$V = \{ P \in \mathbb{R}^n \mid \mu(P) \neq 0 \vee \alpha(P) \neq 0 \}$$

$$\mu(P) = \Sigma_{\delta c(L1,L2,P)}\, \mu_1(P_1) \times \mu_2(P_2)$$

$$\alpha(P) = \Sigma_{\delta c(L1,L2,P)}\, \alpha_1(P_1) \times \alpha_2(P_2)$$

Siguiendo el mismo proceso que para el producto lineal obtenemos:

$$\delta_c(L_1,L_2,P) = \{(P_1,P_2) \mid P_1 \in V_1, P_2 \in V_2 \text{ y } P = T_c^{[P_1 1]}(P_2) \}$$

$$= \{(P_1,P_2) \mid P_1 \in V_1, P_2 \in V_2,$$

$$P = T_c^{[u\cdot Q11 + (u-1)\cdot Q12]}(v\cdot Q_{21} + (v-1)\cdot Q_{22}),\ u,v \in [0,1]\}$$

Y el volumen será

$$V = T_c^{[u\cdot Q11 + (u-1)\cdot Q12]}(v\cdot Q_{21} + (v-1)\cdot Q_{22}),\ u,v \in [0,1]\}$$

es inmediato comprobar, teniendo en cuenta los lemas 2.6 y 3.4, que el objeto, cuando el conjunto de pares es finito, es un paralelogramo cuyos valores de presencia y aspecto están dados por

$$\mu(P) = \begin{cases} 1 \text{ si } P \in \Pi(L_1,L_2) \\ 0 \text{ si } P \notin \Pi(L_1,L_2) \end{cases}$$

$$\alpha(P) = \alpha_1(P)\cdot\alpha_2(P)\cdot\mu(P)$$

Corolario 3.4: El producto circular de dos líneas una de las cuales posee un extremo en el origen, genera un triángulo.

Demostración:

Basta con tener en cuenta una transformación circular está definida como

$$\mathbf{T}_c^{[P_c]}(P) = \mathbf{T}_s^{[RP_c]}(\mathbf{T}_g^{[P_c]}(P))$$

y por tanto, al ser un extremo el origen, dos de los extremos del paralelogramo estarán en el origen de coordenadas. □

Definición 3.28: Denominaremos **arco unidad** a todo objeto definido por:

$$\text{arco}(\varphi_1,\varphi_2) = (V, \mu, \alpha);$$

con $\varphi_1, \varphi_2 \in \mathbf{R}^{n-1}$

siendo:

$$V=\{ P \mid P=(1, u\cdot\varphi_{11}+(1-u)\cdot\varphi_{21},..,u\cdot\varphi_{1,n-1}+(1-u)\cdot\varphi_{2,n-1}),\ u\in [0,1]\}$$

$$\mu(P) = \begin{cases} 1 \text{ si } P\in V \\ 0 \text{ si } P\notin V \end{cases}$$

□

Teorema 3.35: El producto circular de un arco unidad por una línea, cuando es calculable, genera una sector circular.

Demostración:

Sean F y L un arco y una línea definidos como:

$$F = \text{Arco}(\varphi_1,\varphi_2),\ L = \text{Línea}(Q_1,Q_2)$$

Calculemos su producto circular:

$$F \otimes L = (V, \mu, \alpha)$$

siendo:

$$V = \{ P \in \mathbf{R}^n \mid \mu(P) \neq 0 \vee \alpha(P) \neq 0)$$

$$\mu(P) = \Sigma_{\delta c(F,L,P)}\, \mu_1(P_1) \times \mu_2(P_2)$$

$$\alpha(P) = \Sigma_{\delta c(F,L,P)}\, \alpha_1(P_1) \times \alpha_2(P_2)$$

Teniendo en cuenta la definición de arco y de transformación circular, sus conjuntos de puntos productos circular son:

$$\delta_c(F,L,P) = \{(P_1,P_2) \mid P_1 \in V_1, P_2 \in V_2 \text{ y } P = T_c^{[P1]}(P_2)\}$$

$$= \{(P_1,P_2) \mid P_1 \in V_1, P_2 \in V_2,$$

$$P = T_g^{[(1,u\cdot\varphi 11+(1-u)\cdot\varphi 21,..,u\cdot\varphi 1,n-1+(1-u)\cdot\varphi 2,n-1)} \quad (v\cdot Q_1+(v-1)\cdot Q_2),\ u,v \in [0,1]\}$$

Y, teniendo en cuenta que la presencia de ambos objetos es uno, el volumen será

$$V = \{T_g^{[(1,\ u\cdot\varphi 11+(1-u)\cdot\varphi 21,..,u\cdot\varphi 1,n-1+(1-u)\cdot\varphi 2,n-1)} \quad (V\cdot Q_1+(v-1)\cdot Q_{22})$$

que es una rotación de la línea respecto al origen. Los valores de presencia y aspecto están dados por

$$\mu(P) = \begin{cases} 1 \text{ si } P \in \Pi(L_1,L_2) \\ 0 \text{ si } P \notin \Pi(L_1,L_2) \end{cases}$$

$$\alpha(P) = k_1\cdot k_2\cdot \mu(P) \quad k \in A$$

□

Corolario 3.5: El producto circular de un arco de 360º con una línea colocada sobre el primer eje de coordenadas y con un extremo en el origen es un círculo.

Demostración:

Trivial. □

3.10 CONCLUSIONES.

En este capítulo se han definido funciones de modelado que permiten combinar objetos. El conjunto de objetos, junto con las operaciones de suma y producto por escalar posee estructura de espacio vectorial, junto con las operaciones booleanas posee estructura de álgebra de boole. Además se han definidos dos productos internos, que son asociativos y conmutativos.

Los productos internos se ha usado para construir objetos simples: triángulos, sectores circulares, rectángulos y paralelogramos.

El concepto de objeto gráfico, tal como se ha presentado, puede servir de nexo teórico a varios métodos de modelado de sólidos y superficies. Además permite hacer un tratamiento homogéneo de sólidos y superficies.

Concretamente, se pueden expresar como expresiones de objetos gráficos los sólidos formados por operaciones booleanas (CSG), por enumeración y por barrido. Queda abierto el problema de construir, mediante operaciones de objetos gráficos, sólidos mediante la instancia de su frontera.

4. FUNCIONES DE OBJETOS Y VISUALIZACION.

4. FUNCIONES DE OBJETOS Y VISUALIZACION.

Una función de objetos es una función que actúa sobre los objetos, modificando alguna de sus componentes. Se pueden considerar funciones para modificar el color, la presencia o la extensión de los objetos. Otro aspecto esencial en cualquier sistema gráfico es la visualización, es decir, la generación de una imagen a partir de un objeto. En este capítulo se aborda el estudio de las funciones de objetos y la formalización de la visualización.

A partir de las funciones de objetos se definirán relaciones de equivalencia en el conjunto de objetos, que permitirán completar la jerarquía de objetos.

4.1 FUNCIONES DE OBJETOS.

En principio, las funciones de objetos son funciones de **O** en **O**, que actúan sobre una, o varias, de las componentes del objeto. Esto es:

$$\mathbf{F}: \mathbf{O} \dashrightarrow \mathbf{O}$$

Esta caracterización de función es demasiado amplia. Un caso particular de funciones de objetos, tal y como las acabamos de considerar, serían las transformaciones geométricas, que han sido tratadas previamente (secc. 2.1.3). No obstante, la definición que se dará de función de objeto es más restrictiva, ya que fija la forma en que actúa sobre las componentes del objeto, y por ello las transformaciones geométricas no se podrán considerar como funciones de objetos, en el sentido de dicha definición.

En determinados casos, las funciones se construirán a partir de uno o varios parámetros, que podrán o no ser objetos gráficos. Denotaremos la aplicación de una función **F**, con parámetro ß, sobre un objeto O, con la notación, **F**[ß](O), siempre que no induzca a confusión. Si fuese necesario distinguirla de un producto se usará notación funcional **F**(ß;O).

Según las circunstancias, podrá interesar que las funciones actúen sobre instancias u objetos. Por dicho motivo se dan dos definiciones, y posteriormente se estudiará la equivalencia entre ambos tipos de funciones.

Definición 4.1: Una función de objetos, **F**[ß], con parámetro ß, es una función, **F**: **O** --> **O**, cuya acción sobre un objeto cualquiera, O, es:

$$\mathbf{F}[ß](O) = (V', \mathbf{f}_p[ß](\mu), \mathbf{f}_a[ß](\alpha))$$

siendo

$$V' = \{ P \in \mathbf{R}^n \mid \mu'(P) \neq 0 \text{ ó } \alpha'(P) \neq 0 \}$$

y

$$\mathbf{f}_p[ß](\mu) = \mu' : \mathbf{R}^n \dashrightarrow \mathbf{P}$$

$$\mathbf{f}_a[ß](\alpha) = \alpha' : \mathbf{R}^n \dashrightarrow \mathbf{A}$$

dos funcionales que denominaremos componentes de presencia y aspecto de la función de objetos. □

Es decir, las funciones de objetos modifican la función de aspecto y, o, presencia del objeto, sustituyéndolas por nuevas funciones que para cada punto establecen valores que dependen de los parámetros de la función de objeto y de las funciones de presencia, o aspecto, del objeto al que se aplican.

Definición 4.2: Una función de instancias, **F**[ß], es una función, **F**: **Ω** --> **Ω**, cuya acción sobre una instancia cualquiera, R, es:

$$\mathbf{F}[ß](R) = (D', \hat{\imath}, \mathbf{f}_p[ß](\mu \cdot \tau^{-1}), \mathbf{f}_a[ß](\alpha \cdot \tau^{-1}))$$

siendo

$$D' = \{ P \in \mathbf{R}^n \mid \mu'(P) \neq 0 \text{ ó } \alpha'(P) \neq 0 \}$$

y

$$\mathbf{f}_p[ß](\mu \cdot \tau^{-1}) = \mu' : \mathbf{R}^n \dashrightarrow \mathbf{P}$$

$$\mathbf{f}_a[ß](\alpha \cdot \tau^{-1}) = \alpha' : \mathbf{R}^n \dashrightarrow \mathbf{A}$$

dos funcionales que denominaremos componentes de presencia y aspecto de la función de instancias. □

Ejemplo 4.1

Como casos particulares se pueden dar, entre otros, los siguientes:

- Se mantiene el valor de la presencia (o aspecto) del objeto:

$$\mathbf{f}_p[ß](\mu) = \mu$$

Esta situación aparecerá siempre que la función afecte sólo a la otra componente del objeto (aspecto o presencia).

- La función establece una transformación del aspecto, o presencia, del objeto. Es decir, existe una función, $\alpha_f[ß]$: **A** --> **A**, tal que:

$$\mathbf{f}_a[ß](\alpha) = \alpha' = \alpha_f[ß] \circ \alpha$$

con lo que $\alpha'(P) = \alpha_f[ß](\alpha(P))$

Este tipo de funciones se pueden usar, por ejemplo, para transformar los colores de un objeto. Estas funciones, que denominaremos filtros, se estudiarán en detalle en sección 4.1.2.

- La función es independiente del aspecto, o presencia, del objeto. Es decir, existe una función, $\alpha_s[ß]$: $\mathbf{R}^n$ --> **A**, tal que:

$$\mathbf{f}_a[ß](\alpha) = \alpha' = \alpha_s[ß]$$

con lo que $\alpha'(P) = \alpha_s[ß](P)$

De esta forma se puede, por ejemplo, cambiar el color de un objeto aplicando una textura.

Es importante comprobar que ambas definiciones son consistentes. Para ello demostraremos, que las funciones sobre instancias son estables para la relación de equivalencia efectiva, y que la instancia imagen pertenece al objeto imagen según la misma función.

Lema 4.1: Las funciones de instancias son estables para la relación de equivalencia efectiva.

Demostración:

Sea la función **F**[ß] y la instancia

$R = (D, \tau, \mu, \alpha)$

su instancia normal efectivamente equivalente es

$R_N = (D_N, \hat{\imath}, \mu_N, \alpha_N) = (D_{ef}, \hat{\imath}, \mu \cdot \tau^{-1}, \alpha \cdot \tau^{-1})$

El resultado de aplicar la función a ambas instancias es:

$$\mathbf{F}[ß](R) = (D', \hat{\imath}, \mu', \alpha') = (D', \hat{\imath}, \mathbf{f}_p[ß](\mu \cdot \tau^{-1}), \mathbf{f}_a[ß](\alpha \cdot \tau^{-1}))$$

siendo $D' = \{ P \in \mathbf{R}^n \mid \mu'(P) \neq 0 \text{ ó } \alpha'(P) \neq 0 \}$

$$\mathbf{F}[\beta](R_N) = (D'', \hat{\imath}, \mu'', \alpha'') = (D'', \hat{\imath}, \mathbf{f}_p[\beta](\mu\cdot\tau^{-1}), \mathbf{f}_a[\beta](\alpha\cdot\tau^{-1}))$$

siendo $D'' = \{ P \in \mathbb{R}^n \mid \mu''(P) \neq 0 \text{ ó } \alpha''(P) \neq 0 \}$

que obviamente coinciden. □

Teorema 4.1 El resultado de la aplicación de una función a una instancia de un objeto es una instancia perteneciente al objeto resultante de aplicar la función al objeto.

Demostración:

Trivial. Basta con considerar el resultado del lema 4.1 y el hecho de que la acción de una función sobre un objeto coincide con la aplicación sobre su instancia normal. □

Para poder aplicar el concepto de función sobre las familias de objetos es necesario comprobar que las funciones sobre objetos son estables para la relación de equivalencia formal.

Lema 4.2: Las funciones de objetos, $\mathbf{F}[\beta]$, cuyos funcionales de aspecto y presencia cumplan, que para toda transformación geométrica, T, y cualesquiera funciones de presencia y aspecto, μ y α:

$$\forall T, \mu,\alpha \Rightarrow \mathbf{f}_p[\beta](\mu\cdot T) = \mathbf{f}_p[\beta](\mu)\cdot T \text{ y } \mathbf{f}_a[\beta](\alpha\cdot T) = \mathbf{f}_a[\beta](\alpha)\cdot T$$

son estables para la relación de equivalencia formal.

Demostración:

Sean O_1 y O_2 dos objetos formalmente equivalentes, $O_1 \propto O_2$. Por la definición de equivalencia formal:

$$\exists T \mid T(O_1) = O_2$$

Para cualquier función, $\mathbf{F}[\beta]$, se cumple

$$\mathbf{F}[\beta](O_1) = (V'_1, \mathbf{f}_p[\beta](\mu_1), \mathbf{f}_a[\beta](\alpha_1))$$

siendo $V'_1 = \{ P \in \mathbb{R}^n \mid \mu_1'(P) \neq 0 \text{ ó } \alpha_1'(P) \neq 0 \}$

y

$$\mathbf{F}[\beta](O_2) = (V'_2, \mathbf{f}_p[\beta](\mu_2), \mathbf{f}_a[\beta](\alpha_2))$$

con $V'_2 = \{ P \in \mathbb{R}^n \mid \mu_2'(P) \neq 0 \text{ ó } \alpha_2'(P) \neq 0 \}$

que por el teorema 2.5 es

$$\mathbf{F}[\beta](O_2) = (V_2', \mathbf{f}_p[\beta](\mu_1 \cdot T^{-1}), \mathbf{f}_a[\beta](\alpha_2 \cdot T^{-1}))$$

que teniendo en cuenta la hipótesis del teorema es obviamente igual a

$$\mathbf{F}[\beta](T(O_1)) \qquad \square$$

Existen funciones de objetos que cumplen la hipótesis del teorema. Esta se cumple, en general por todas aquellas funciones que realizan un cambio en la presencia o el aspecto independiente de las coordenadas, como por ejemplo: permutar dos colores, asignar a todo punto con presencia no nula presencia uno.

Esto nos permite definir la acción de determinadas funciones de objetos sobre una familia de objetos.

Definición 4.3: La aplicación de una función de objetos, estable para la relación de equivalencia formal, a una familia es la familia representada por el objeto resultante de aplicar dicha función a un objeto cualquiera de la familia. □

Además de la importancia que tienen en sí estas funciones de objetos, poseen una importancia adicional, en tanto que permiten completar la jerarquía de objetos. En este sentido, se podrán establecer particiones en clases de equivalencia de los conjuntos de objetos y de familias de objetos utilizando funciones. Así, por ejemplo, se podría hablar de todos los objetos que se diferencian tan sólo en su color.

Definición 4.4: Sea $\mathbf{F}[\beta]$ una función, cuyo parámetro β toma valores en un determinado conjunto **C**, definimos la **relación inducida** por la función en el conjunto de objetos, del siguiente modo

$$O_1 \; R \; O_2 \text{ sii } \exists\, \beta_1, \beta_2 \in \mathbf{C} \mid \mathbf{F}[\beta_1](O_1) = \mathbf{F}[\beta_2](O_2) \qquad \square$$

Ejemplo 4.2

Sea la función de cambio de color, definida por los siguientes funcionales componentes:

$$\mathbf{f}_p[\beta](\mu) = \mu$$

$$\mathbf{f}_a[\beta](\alpha) = \alpha'$$

$$\text{con } \alpha'(P) = \begin{cases} 0 & \text{si } \alpha(P) = 0 \\ \beta \in \mathbf{A} & \text{si } \alpha(P) \neq 0 \end{cases}$$

Por la relación inducida por esta función, todas las líneas definidas como:

línea(P_1,P_2,c)

para valores de P_1 y P_2 fijos y cualquier valor de aspecto, c, se relacionan.

De mayor interés es la definición de una relación inducida sobre el conjunto de familias de objetos. Dicha relación permitirá, por ejemplo, relacionar todas las líneas, independientemente de su color,

Definición 4.5: Sea **F**[ß] una función, cuyo parámetro ß toma valores en un determinado conjunto **C**, definimos la **relación inducida** por la función en el conjunto de familias de objetos del siguiente modo

$\psi_1 \, R \, \psi_2$ sii $\exists \, ß_1, ß_2 \in \mathbf{C} \mid \forall \, O_1 \in \psi_1 \, \exists \, O_2 \in \psi_2, \quad \mathbf{F}[ß_1](O_1) = \mathbf{F}[ß_2](O_2)$ □

Se verá que, bajo determinadas condiciones, esta relación es de equivalencia, lo que permitirá establecer particiones en clases de las familias de objetos.

Teorema 4.2: La relación inducida por una función, **F**[ß], en el conjunto de familias de objetos es una relación de equivalencia siempre que la función cumpla una de las condiciones siguientes:

a) ß puede tomar un único valor.

o bien

b) $\forall \, O \in \mathbf{O} \quad \forall \, ß_1, ß_2 \quad \exists \, ß_3 \mid \quad \mathbf{F}[ß_1](\mathbf{F}[ß_2](O)) = \mathbf{F}[ß_2](\mathbf{F}[ß_1](O)) = \mathbf{F}[ß_3](O)$

Demostración:

Es evidente que, independientemente de las condiciones del teorema, las relaciones inducidas por funciones son reflexivas y simétricas. Para que la relación sea transitiva debe cumplirse que

$\forall \, \psi_1, \psi_2, \psi_3 \mid \psi_1 \, R \, \psi_2 \;$ y $\; \psi_2 \, R \, \psi_3 \; => \; \psi_1 \, R \, \psi_3$

que de acuerdo con la definición 4.5 es equivalente a

$\forall \, \psi_1, \psi_2, \psi_3 \mid \exists \, ß_{12}, ß_{21}, ß_{23}, ß_{32} \in \mathbf{C}$,

$\forall \, O_1 \in \psi_1 \, \exists \, O_2 \in \psi_2$ con $\mathbf{F}[ß_{12}](O_1) = \mathbf{F}[ß_{21}](O_2)$

y

$$\forall\, O_2 \in \psi_2\ \exists\, O_3 \in \psi_3 \text{ con } \mathbf{F}[\beta_{23}](O_2) = \mathbf{F}[\beta_{32}](O_3)$$

$$=> \exists\, \beta_{13},\ \beta_{31} \in \mathbf{C} \mid \forall\, O_1 \in \psi_1\ \exists\, O_3 \in \psi_3 \text{ con } \mathbf{F}[\beta_{13}](O_1) = \mathbf{F}[\beta_{31}](O_3)$$

Si se cumple la primera condición del teorema, la transitividad es evidente, ya que $\beta_{12}=\beta_{21}=\beta_{23}=\beta_{32}$.

Supongamos ahora que se cumple la segunda condición del teorema. Consideremos las igualdades

$$\mathbf{F}[\beta_{12}](O_1) = \mathbf{F}[\beta_{21}](O_2)$$

y

$$\mathbf{F}[\beta_{23}](O_2) = \mathbf{F}[\beta_{32}](O_3)$$

Apliquemos $\mathbf{F}[\beta_{23}]$ a la primera y $\mathbf{F}[\beta_{21}]$ a la segunda

$$\mathbf{F}[\beta_{23}](\mathbf{F}[\beta_{12}](O_1)) = \mathbf{F}[\beta_{23}](\mathbf{F}[\beta_{21}](O_2))$$

$$\mathbf{F}[\beta_{21}](\mathbf{F}[\beta_{23}](O_2)) = \mathbf{F}[\beta_{21}](\mathbf{F}[\beta_{32}](O_3))$$

y teniendo en cuenta la condición del teorema tendremos

$$\mathbf{F}[\beta_{23}](\mathbf{F}[\beta_{21}](O_2)) = \mathbf{F}[\beta_{21}](\mathbf{F}[\beta_{23}](O_2))$$

$$\exists\, \beta_{13} \mid \mathbf{F}[\beta_{23}](\mathbf{F}[\beta_{12}](O_1)) = \mathbf{F}[\beta_{13}](O_1)$$

$$\exists\, \beta_{31} \mid \mathbf{F}[\beta_{21}](\mathbf{F}[\beta_{32}](O_3)) = \mathbf{F}[\beta_{31}](O_3)$$

con lo que la implicación es evidente. □

4.1.1 Funciones de transferencia de objeto.

Hasta aquí, hemos considerado las funciones en su forma más general, esto es, aquellas en las que la presencia y aspecto se obtienen como:

$\mathbf{f_p}[\beta](\mu)$: $\mathbf{R^n}$ ----> **P**

$\mathbf{f_a}[\beta](\alpha)$: $\mathbf{R^n}$ ----> **A**

Un caso particularmente interesante de funciones son aquellas en las que el parámetro, ß, es otro objeto. Este tipo de funciones permitirá, por ejemplo, copiar la textura de un objeto en otro.

Para formalizar el concepto de función de transferencia se fijará la dependencia de la función con el objeto parámetro.

Definición 4.6: Diremos que una función, **F**[ß], es una **función de transferencia** de objeto cuando su parámetro sea un par, ß=(S,g), $S=(V_s,\mu_s,\alpha_s)\in\mathbf{O}$, g: $\mathbf{R}^n$ --> $\mathbf{R}^n$, y sus funcionales componentes

$$\mathbf{f}_p[ß](\mu) = \mu'$$

$$\mathbf{f}_a[ß](\alpha) = \alpha'$$

sean de una de las formas siguientes

a) Transferencia de aspecto

$$\alpha'(P) = \alpha_s(g(P))\times\alpha(P)$$

$$\mu'(P) = \mu(P)$$

b) Transferencia de presencia

$$\alpha'(P) = \alpha(P)$$

$$\mu'(P) = \mu_s(g(P))\times\mu(P)$$

c) Transferencia de aspecto y presencia

$$\alpha'(P) = \alpha_s(g(P))\times\alpha(P)$$

$$\mu'(P) = \mu_s(g(P))\times\mu(P)$$ □

La función g se ha incluido como parte de la definición de la función por razones prácticas, ya que de este modo, sin que el objeto S ocupe todo el espacio, puede transferir su presencia, y/o aspecto, a cualquier punto de mismo. Por esta razón, g no será normalmente una aplicación inyectiva, ni sobreyectiva, ya que aplicará todo $\mathbf{R}^n$ en el volumen del objeto S.

Una aplicación notable de las funciones de transferencia de objeto es la incorporación de patrones de color o textura a los objetos. Un determinado estampado estará definido por un objeto. La aplicación del estampado a otro objeto se realizará utilizando una función de color inducida por el estampado. Obsérvese, que la función realiza en realidad una modulación del valor del objeto al que se aplica con el objeto que define la función. Para obtener una asignación del aspecto o presencia, será necesario asignar previamente valor uno al aspecto o presencia del objeto al que se aplica la función.

Teorema 4.3: La relación inducida por una función de transferencia, **F**[(S,g)], con $g\in\mathbf{G}$ $S\in\mathbf{S}\subset\mathbf{O}$, en el conjunto de familias de objetos es una relación de equivalencia, si se cumple alguna de las siguientes condiciones:

a) Los conjunto **G** y **S** contienen un único elemento

o

b) $\hat{\imath} \in \mathbf{G}$ y

$\forall\, g_1,g_2 \in \mathbf{G},\ S_1,S_2 \in \mathbf{S} \Rightarrow \exists\, S_3 \in \mathbf{S} \mid \mathbf{F}[(g_1,S_1)](\mathbf{F}[(g_2,S_2)](S')) = S_3$

estando S' definido del siguiente modo

$$S' = (\infty,1,1)$$

Demostración:

De acuerdo con el teorema 4.2, si el parámetro esta formado por un único par objeto-g el teorema es cierto. En caso contrario será necesario probar que

$$\forall\, O \in \mathbf{O}\ \ \forall\, \beta_1,\beta_2\ \ \exists\, \beta_3 \mid\ \mathbf{F}[\beta_1](\mathbf{F}[\beta_2](O)) = \mathbf{F}[\beta_2](\mathbf{F}[\beta_1](O)) = \mathbf{F}[\beta_3](O)$$

Supongamos que la función de transferencia afecta al aspecto del objeto. En este caso, el primer término de la primera igualdad, $\mathbf{F}[\beta_1](\mathbf{F}[\beta_2](O)) = \mathbf{F}[\beta_2](\mathbf{F}[\beta_1](O))$, equivale, para las funciones de aspecto a

$$\alpha_{s1}(g_1(P)) \times \alpha_{s2}(g_2(P)) \times \alpha(P)$$

que obviamente es igual a

$$\alpha_{s2}(g_2(P)) \times \alpha_{s1}(g_1(P)) \times \alpha(P)$$

Para comprobar la segunda igualdad construyamos S_3 como

$$S_3 = \mathbf{F}[\beta_1](\mathbf{F}[\beta_2](S'))$$

estando S' definido del siguiente modo

$$S' = (\infty,1,1)$$

Según la hipótesis del teorema, S_3, pertenece a **S**, y podemos construir su función de transferencia de objeto. Resulta inmediato comprobar que la función de transferencia definida por S_3 y la identidad en $\mathbf{R}^n$, $\mathbf{F}[\hat{\imath},S_3]$, es igual a $\mathbf{F}[\beta_1] \circ \mathbf{F}[\beta_2]$, y por tanto

$$\mathbf{F}[\hat{\imath},S_3](O) = \mathbf{F}[\beta_1](\mathbf{F}[\beta_2](O))$$

Para la función de presencia se puede realizar la misma comprobación. Queda por comprobar que la condición se cumple para las funciones de transferencia que no afectan al aspecto o presencia, que es evidente. □

De este modo, por tanto, se pueden introducir relaciones de equivalencia en el conjunto de familias, y podremos hablar, por ejemplo, de la clase de todas las familias de objetos que se diferencia tan sólo en su estampado.

4.1.2 Filtros.

Otra clase interesante de funciones es aquella en que el parámetro es una aplicación en el dominio de aspecto, $ß_a$: **A** --> **A**, o en el de presencia. Este tipo de funciones realizan modificaciones del aspecto, o presencia, de los objetos, independientes de las coordenadas.

Definición 4.7: Un **filtro**, $F[ß_p,ß_a]$ es una función cuya acción sobre un objeto, $O = (V,\mu,\alpha)$, esta determinada por la acción de dos funciones, $ß_a$: **A** --> **A** y $ß_p$: **P** --> **P**, según:

$$F[ß_p,ß_a](O) = (V',ß_p \circ \mu, ß_a \circ \alpha)$$ □

El valor del aspecto, o presencia, del objeto imagen, será en cada punto:

$$\alpha'(P) = ß_a(\alpha(P))$$

La función $ß_a$, o $ß_p$, especifica completamente la acción del filtro, y por tanto podrá ser usada para hacer referencia al mismo.

Como casos particulares, se pueden definir filtros que modifiquen tan sólo el aspecto o la presencia, haciendo que una de las funciones sea la identidad. Así, podemos hablar de **filtros de aspectos**, $F[ß_a]$, que será una función cuya acción sobre un objeto, O, esta dada por

$$F[ß_a](O) = (V',\mu,ß_a \circ \alpha) \text{ con } ß_a: \mathbf{A} \dashrightarrow \mathbf{A}$$

Y de **filtros de presencias**, $F[ß_p]$, cuya acción estará dada por:

$$F[ß_p](O) = (V',ß_p \circ \mu,\alpha) \text{ con } ß_p: \mathbf{P} \dashrightarrow \mathbf{P}$$

Los tipos de filtros que se puedan definir dependerá de las propiedades del espacio de color usado, pudiendo definirse filtros para añadir un color, substraerlo, dejar sólo una componente de color, sustituirlo, etc.

Ejemplo 4.3:

Los siguientes son ejemplos de filtros de presencia útiles para operar con objetos.

Revelado $ß_{pr}(x) = Abs(x)$

Hace que tenga presencia positiva, y por tanto material, todo el objeto.

$$\text{Unitario } \beta_{pu}(x) = \begin{cases} 1 \text{ si } x > 0 \\ 0 \text{ si } x = 0 \\ -1 \text{ si } x < 0 \end{cases}$$

Elimina las presencias múltiples del objeto, sin cambiar su apariencia.

Molde $\beta_{pm}(x) = -x$

Construye un molde de la presencia del objeto.

Los siguientes son ejemplos de filtros de aspecto para el dominio de aspecto C_{rgb}

Saturar $\beta_{as}[k](c) = (k \cdot c_R, k \cdot c_G, k \cdot c_B)$

siendo (c_R, c_G, c_B) **las componentes RGB del color.**

Molde $\beta_{am}(x) = -x$

$$\text{Blanco/Negro } \beta_{a2}(x) = \begin{cases} 1 \text{ si } x \neq 0 \\ 0 \text{ si } x = 0 \end{cases}$$

Teorema 4.4: La relación inducida por un filtro aspecto, o presencia, $\mathbf{F}[\beta]$ $\beta \in \mathbf{C}$, en el conjunto de familias de objetos es una relación de equivalencia si se cumple la siguiente condición

$$\forall\, \beta_1, \beta_2 \in \mathbf{C} \Rightarrow \beta_1 \circ \beta_2 = \beta_2 \circ \beta_1 \in \mathbf{C}$$ □

Demostración:

Basta con aplicar el teorema 4.2, teniendo en cuenta que el parámetro es cualquier función de **C.** □

El conjunto de filtros de saturación definidos en el ejemplo 4.3 inducen una relación de equivalencia. Del mismo modo, el conjunto de filtros formado por el filtro molde y el filtro identidad inducen una relación de equivalencia en el conjunto de familias de objetos.

4.2 APLICACIONES.

En la sección anterior se han definidos dos tipos de funciones de objetos, las funciones de transferencia y los filtros. Nos proponemos ahora mostrar como estas funciones se pueden utilizar para representar operaciones habituales en informática gráfica. Concretamente se mostrará como asignar un valor constante de aspecto, o presencia a un objeto, y como realizar un recortado. El hecho de que las funciones se puedan utilizar para realizar dichas operación hará innecesaria la definición de tipos adicionales de funciones.

4.2.1 Asignación de aspecto o presencia a un objeto.

Frecuentemente interesará asignar un color concreto, o una presencia, a todo un objeto. El resultado de esta operación debe de ser un objeto con el mismo volumen que el objeto original, en el que la función de aspecto (o presencia) es constante dentro del volumen.

Esta operación se puede caracterizar como un filtro, cuyo parámetro determine el valor del aspecto, o presencia a asignar. Concretamente podría pensarse en definir dichas funciones de forma que el aspecto, o presencia, del objeto se modifique según:

$\mathbf{F}[ß_p](O) = (V, ß_p \circ \mu, \alpha)$ con $ß_p$: **P** --> **P** definida como

$$ß_p(v) = \begin{cases} k \text{ si } v \neq 0 \\ 0 \text{ si } v = 0 \end{cases}$$

siendo k el valor de presencia a asignar.

Obsérvese que este filtro no modifica el volumen del objeto, ya que no modifica el valor de presencia de los puntos que tenían valor cero.

4.2.2 Recortado.

El recortado es una función esencial en cualquier sistema gráfico, siendo indispensable para delimitar el campo de generación de una imagen. El recortado de un objeto se puede entender como una operación que reduce la zona del espacio en que su μ y su α son no nulas, y por tanto el volumen del objeto.

Para denotar a una función de recortado, independientemente de cual sea la forma en que se realiza, se utilizará la siguiente notación:

$\square_e(O)$

donde e puede ser un volumen o un objeto, indicando este último caso, que se recorta a su volumen el objeto O.

Existen varios formas de realizar un recortado utilizando las funciones y operaciones de la teoría de objetos. Una primera posibilidad, es la definición de la operación de recortado como una intersección entre el objeto en cuestión y un objeto que determine al volumen de recortado. Con este propósito se deberá construir, como objeto determinante del recortado, un objeto cuyo volumen sea el volumen de recortado, y cuyas funciones de aspecto y presencia sean constantes en dicho volumen e iguales a los universos del algebra de boole de aspecto y presencia, respectivamente.

Para realizar el recortado como una función de transferencia de objeto, se utilizará un objeto cuyo volumen sea el de recortado, con presencia y aspecto igual al elemento unidad de los dominios correspondientes. En este caso se utilizará como aplicación, g, la función identidad.

En cualquier caso se consigue que el volumen de recortado quede especificado por un objeto.

4.3 VISUALIZACION.

Un problema fundamental en la mayor parte de las aplicaciones gráficas es el de generación de una imagen a partir de una escena. Es fundamental entender la imagen y la escena como dos entidades distintas, con un contenido de información diferente.

El proceso de visualización se puede abordar utilizando el formalismo de Fiume [Fium89]. Para ello, a efectos de visualización, se puede derivar el concepto de objeto de Fiume a partir del concepto de objeto gráfico, que hemos introducido. Fiume define los objetos gráficos como un par

$$(\mathbf{Z}_0,\mathbf{I}_0) \text{ con } \mathbf{Z}_0 \subseteq \mathbf{R}^3, \quad \mathbf{I}_0: \mathbf{Z}_0 \longrightarrow \mathbf{C} \subseteq \mathbf{R}^c \quad c \geq 1$$

en el que $\mathbf{Z}_0$ representa el volumen del objeto e $\mathbf{I}_0$ es un función que asigna color a cada punto del objeto.

El concepto de Objeto Gráfico presentado en este documento es más general que este último, al que engloba. Por tanto, para cualquier objeto gráfico, no será posible derivar un objeto Fiume que contenga la misma cantidad de información. Tan sólo será posible, para algunos objetos gráficos $O=(V,\mu,\alpha)$ de $\mathbf{R}^3$, definir un objeto de Fiume, O_f, que genere la misma imagen que él.

Para ello definiremos, para cada objeto Fiume, un objeto gráfico equivalente, para lo que será necesario aplicar el espacio de color de Fiume en un dominio de color.

Definición 4.7: Para cada objeto Fiume, O_f,

$$O_f = (\mathbf{Z}_0,\mathbf{I}_0) \text{ con } \mathbf{Z}_0 \subseteq \mathbf{R}^3, \quad \mathbf{I}_0: \mathbf{Z}_0 \longrightarrow \mathbf{C} \subseteq \mathbf{R}^c \quad c \geq 1$$

y dada una función inyectiva, G: $\mathbf{C} \longrightarrow \mathbf{A}$, donde $\mathbf{A}$ es un dominio de aspecto, definimos su objeto gráfico equivalente, O, como

$$O = (V,\mu,\alpha)$$

siendo

$$V = \mathbf{Z}_0$$

$$\mu(P) = \begin{cases} 1 \text{ si } P \in \mathbf{Z}_0 \\ 0 \text{ si } P \notin \mathbf{Z}_0 \end{cases}$$

$$\alpha(P) = \begin{cases} G(\mathbf{I}_0(P)) \text{ si } P \in \mathbf{Z}_0 \\ 0 \quad \text{si } P \notin \mathbf{Z}_0 \end{cases}$$

□

Esto permitirá utilizar la teoría de visualización de Fiume, dentro del formalismo presentado, para los objetos gráficos que se corresponden con objetos Fiume. Para poder utilizar el formalismo de visualización de Fiume con el resto de los objetos es necesario establecer su imagen en base a algún objeto que se corresponda con un objeto Fiume.

Definición 4.8: Dos objetos, $O_1=(V_1,\mu_1,\alpha_1)$ y $O_2=(V_2,\mu_2,\alpha_2)$, definidos en $\mathbf{R}^n$, son **visualmente equivalentes**, y lo representaremos por $\approx$, si sus volúmenes son idénticos, $V_1 = V_2$, y sus funciones de aspecto y presencia cumplen:

$$\forall P \in \mathbf{R}^n \Rightarrow \alpha_1(P_1)=\alpha_2(P_2)$$

y

$$\mu_1(P) > 0 \text{ sii } \mu_2(P) > 0$$

□

Resulta obvio que equivalencia visual es una relación de equivalencia.

Proposición 4.1: La equivalencia visual entre objetos es una relación de equivalencia.

Demostración:

Trivial. □

De este modo, podemos considerar que la imagen de un objeto cualquiera es igual a la de un objeto visualmente equivalente a él. Si en cada clase de equivalencia existe un objeto que se corresponde con un objeto Fiume, podemos utilizar, como imagen, la del objeto Fiume.

Proposición 4.2: En cada clase de equivalencia visual existe un, y sólo uno, objeto gráfico que se corresponde con un objeto Fiume.

Demostración:

Sea el objeto, $O_1=(V_1,\mu_1,\alpha_1)$, definidos en $\mathbf{R}^3$. Construyamos el objeto, $O_2=(V_2,\mu_2,\alpha_2)$, visualmente equivalente a O_1, dado por

$$\forall P \in \mathbf{R}^n \Rightarrow \alpha_1(P_1)=\alpha_2(P_2)$$

y

$$\mu_2(P)=\begin{cases} 0 \text{ si } \mu_1(P) \leq 0 \\ 1 \text{ si } \mu_1(P) > 0 \end{cases}$$

Es inmediato comprobar que el objeto Fiume,

$$O_f = (V_2,\alpha) \text{ con } V_2 \subseteq \mathbf{R}^3, \quad \alpha: V_2 \longrightarrow \mathbf{A} \subseteq \mathbf{R}^c \quad c \geq 1,$$

es equivalente a O_2, y que O_2 es el único objeto gráfico visualmente equivalente a O_1 y equivalente a un objeto Fiume. □

Esta forma de abordar la visualización posee el inconveniente de forzar la interpretación visual de los distintos valores de presencia de los objetos gráficos, ya que de hecho se está prescindiendo de parte de la información de los objetos al realizar la visualización.

Otra alternativa, que aquí no desarrollaremos totalmente, es elaborar una formalización del proceso de visualización basado directamente en el concepto de objeto gráfico. El resto de está sección se dedica a esbozar dicho desarrollo.

Supongamos que generamos una imagen de un objeto bidimensional O sobre un dispositivo de puntos. La imagen puede ser considerada, de forma natural, un objeto bidimensional. El proceso de visualización podrá por tanto entenderse como un proceso según el cual se utiliza un objeto (escena) para asignar el color a otro (pantalla). Formalizaremos el concepto de pantalla de puntos, y a partir de él definiremos el proceso de visualización.

Definición 4.9: Un **Punto de imagen** o **Pixel** es un objeto bidimensional que cumple:

V es un subconjunto plano y convexo.

μ toma valores en {0,1}

α es constante sobre V □

En esta definición se contempla sólo el aspecto material del pixel. Hay un segundo aspecto relevante que es la forma en que se asigna color al pixel a partir de la escena. Usualmente se proyecta la escena sobre el plano que contiene al pixel y posteriormente se determina el color del pixel a partir de las proyecciones que aparecen en una determinada área. Fiume asocia al pixel un área de influencia, que denomina prototipo del pixel, pensamos sin embargo que dicha información no debe de estar asociada al pixel sino al proceso de discretización.

Ejemplo 4.4:

Una posible definición de pixel es la siguiente:

Puntual: $\text{Pixel} = \{ V, \mu, \alpha\}$ con $V = \{(0,0)\}$

$\mu = 1$ en V

$\alpha = K \in A$

Definición 4.10: Una **imagen raster** o **monitor** es un objeto bidimensional, formado por la suma de una distribución regular de puntos de imagen, cuyos volúmenes no se solapan.

$$\text{Imagen} = \Sigma\ \text{Pixel}_{i,j}$$ □

El hecho de que la imagen raster sea un objeto bidimensional implica que su volumen sea un área del plano, y que por tanto esté implícito un sistema de coordenadas en el mismo. Típicamente un monitor será rectangular.

El proceso de visualización se puede entender como la aplicación de una **función de discretización** que para cada par monitor-objeto asocia un color a cada pixel del monitor.

Una forma habitual de construir las funciones de discretización es calculando la proyección del objeto sobre el plano del monitor y promediando, para cada pixel, el valor del color de los puntos del objeto proyectados en su vecindad.

No obstante, desde un punto de vista conceptual, para generar la imagen no tiene por qué realizarse necesariamente una proyección de los puntos del objeto en el monitor. Bastará con establecer una relación entre los puntos del objeto y los pixels, para posteriormente poder calcular el color del pixel a partir de los colores de los objetos con los que está relacionado.

Definición 4.11: Una **relación de visualización** es una aplicación que para un par, monitor-objeto dado, asocia a cada pixel una porción del objeto, que denominaremos fracción visible del objeto en el pixel,

$$\pi_v(O,\text{Imagen},\text{Pixel}_{i,j}) = O_{i,j}$$ □

La forma de la relación de visualización puede ser cualquiera. Hay que destacar el hecho de que los $O_{i,j}$ de los distintos pixel no tienen por qué ser disjuntos.

Ejemplo 4.5:

Las siguientes son posibles definiciones de relaciones de visualización:

· En un sistema 2D se puede asociar un área eficaz a cada pixel, y tomar como π_v el objeto resultante de recortar la escena contra el área eficaz del pixel.

· En un sistema 3D se puede tomar como π_v todos los puntos de la escena que sean tocados por rayos de luz que acaben en el pixel.

Definición 4.12: Denominaremos **función de combinación** o mezcla, a aquella que, dado un pixel de una imagen y una fracción visible en él de un objeto, asocia un color al pixel.

F_b: $Pixel_{i,j} \times O_{i,j} \rightarrow C$ □

El proceso completo de visualización se descompondrá en la proyección de los objetos y en aplicar una función de combinación.

Ejemplo 5.3:

Las siguientes son posibles definiciones de funciones de combinación:

· Minimización de función. Supongamos una función $f:\mathbf{R}_n \rightarrow \mathbf{R}$:

$F_b(Pixel_{i,j},O_{i,j}) = \alpha(P)$ con $P \in O_{i,j}$ y $P=\min(f)$

Si como función de selección, f, se toma una medida de la distancia al pixel se estará visualizando con eliminación de partes ocultas.

· Color en punto medio.

$F_b(Pixel_{i,j},O_{i,j}) = \alpha(P)$ con $P \in O_{i,j}$ y $P=centro(O_{i,j})$

· Media con color anterior. Sea una función de selección f:

$F_b(Pixel_{i,j},O_{i,j}) = \alpha_{pixelij} \cap \alpha(P))$ con $P \in O_{i,j}$ y $P=\min(f)$

De este modo la imagen no pierde el contenido previo, sino que lo utiliza para operar con la información nueva que se añada.

Con este planteamiento se pueden formalizar métodos de visualización más a menos simples, que se utilizan en la práctica. Para poder trabajar con modelos de iluminación más complejos se debe de incluir información del modelo de iluminación en el objeto, ya que esta última no puede restringirse, en casos complejos, a la imagen. La inclusión del modelo de iluminación en el objeto puede realizarse definiendo un dominio de aspecto concreto que contemple dicha información.

4.4 CONCLUSIONES.

En el presente capítulo se ha definido el concepto de función de objeto, que se ha mostrado útil en la generación de objetos por cambio de aspecto o presencia, y concretamente para la definición de texturas.

El concepto de función, por otra parte, permite completar la jerarquía de abstracciones gráficas, en base a las relaciones inducidas por las funciones en el conjunto de familias de objetos. Se ha mostrado que bajo determinadas condiciones dicha relación inducida es una relación de equivalencia, lo que nos faculta a hablar de clases de equivalencia en el conjunto de familias de objetos.

Se ha mostrado que el formalismo de visualización de Fiume puede aplicarse a los objetos gráficos aquí definidos, ya que el concepto de objeto de Fiume es menos general que el de objeto gráfico. Por otra parte el formalismo aquí presentado puede ser extendido para contemplar la visualización como una función de objetos, garantizando un tratamiento uniforme de todos los conceptos.

5.APLICACIONES.

5. APLICACIONES.

Este capítulo se dedica a mostrar algunas posibilidades de aplicación de formalismo expuesto en los capítulos precedentes. El álgebra de objetos gráficos, como conceptualización teórica, puede usarse como herramienta, o soporte, en cualquier desarrollo (teórico o práctico) en el ámbito de las informática gráfica. Sin ánimo de hacer una clasificación exhaustiva, consideramos que puede ser provechosa con los siguientes fines:

· Para servir como soporte a métodos de especificación formal de sistemas gráficos.

· Para permitir la estructuración y formalización de la información manejada por un sistema gráfico.

· Como herramienta teórica para estudiar problemas gráficos.

· Como base teórica para el desarrollo de conceptualizaciones específicas.

En este capítulo se muestran tres ejemplos de aplicación del formalismo dentro de las tres últimas categorías, con el único propósito de servir de ilustración al mismo. No pretendemos, por tanto, explorar todas las posibles aplicaciones, si no más bien, mostrar como el álgebra de objetos gráficos es útil en la formalización de sistemas gráficos.

En el capítulo siguiente se estudiará la utilización del álgebra de objetos gráficos como soporte de métodos de especificación formal de sistemas gráficos.

5.1 APLICACION A UN SISTEMA DE DIBUJO.

En esta sección se realiza la formalización de la información gestionada por un sistema de dibujo como ilustración de la posibilidad de usar el álgebra de objetos gráficos en la formalización de la información manejada por un sistema gráfico.

Nuestro objetivo no es el desarrollo de métodos generales, que en cierto sentido se obtienen como consecuencia del estudio de la especificación, sino, tan sólo, el mostrar como la utilización de los conceptos introducidos hasta aquí puede ser útil para la sistematización de la información gráfica gestionada por un sistema concreto.

Supongamos, a tal efecto, que se desea construir un sistema de dibujo artístico. El sistema debe de permitir la creación de imágenes a partir de un determinado repertorio de primitivas.

Para utilizar el álgebra de objetos en este caso, bastará con trabajar sobre $\mathbf{R}^2$, utilizando como dominio de aspecto el dominio RGB $\mathbf{C}_{rgb}$ (secc. 2.1).

Podemos caracterizar una primitiva como un objeto simple, cuya función de presencia es constante e igual a uno. La definición de una primitiva puede realizarse a partir de una lista de puntos y una función de aspecto. Esto es:

Primitiva: Lista(Punto), Aspecto -> Objeto

La imagen en sí, se puede considerar como un objeto gráfico, obtenido a partir de una expresión de objetos, es decir, como una formula matemática en la que se construye un objeto a partir de otros combinandolos con distintos operadores.

Para un sistema de dibujo, al menos habría que considerar las siguientes primitivas:

· Línea
· Punto
· Arco
· Curva

A partir de estas se pueden generar, realizando operaciones de objetos: polígonos, rectángulos, triángulos, círculos, áreas.

Para generar el dibujo se pueden seguir dos procedimientos. Se puede identificar el dibujo con la imagen. Es decir el programa permitirá la creación de determinados objetos, simples o complejos, que se visualizan, de forma independiente, añadiéndose a la imagen. De este modo no es necesario almacenar el dibujo como tal.

Otra alternativa, es considerar al dibujo completo como distinto de la imagen, de tal forma que todo el dibujo fuese un único objeto gráfico, que se almacenaría como una expresión de objetos.

Para generar la imagen se pueden seguir el proceso de visualización expuesto en la sección 4.3 (o el formalismo de Fiume), para lo que habrían de definirse la relación de visualización y la función de combinación.

Si el dibujo se ha identificado con la imagen se podrá realizar la visualización de la forma usual, es decir, que los objetos dibujados más recientemente tapen a los más 'antiguos'. En caso contrario, en los puntos de intersección de primitivas se combinarán las funciones de aspecto, si se desea evitar este efecto, será necesario realizar la combinación utilizando funciones.

5.2 GENERACION DE SOLIDOS POR BARRIDO.

Esta sección muestra como el álgebra de objetos se puede usar como herramienta para resolver problemas gráficos teóricos. Consideraremos, concretamente, el problema de determinar si el objeto generado por barrido de dos elementos es o no un sólido correcto [Requ80].

Informalmente el problema a resolver es el siguiente: si se realiza el barrido de una curva y una superficie, el elemento generado será, en general, un sólido. Sin embargo hay situaciones en las que dicho elemento no es un sólido correcto. Enunciaremos un teorema que caracteriza estas situaciones.

Como paso previo es necesario caracterizar lo que entenderemos por sólido.

Definición 5.1: La dimensión, d, de un objeto, O de $\mathbf{R}^n$, esta dada por la dimensión de la mínima base $(u_1,..,u_d)$ de $\mathbf{R}^n$ necesaria para generar su volumen. □

Definición 5.2: Un objeto, O de $\mathbf{R}^3$, es un cuerpo correcto de dimensión $n \leq 3$, si posee dimensión n, y $\forall ß \in \mathbf{R}$ arbitrariamente pequeño se cumple

$$\forall P \in V \;\; \exists P_1,..,P_n \in V \;\; | \;\; P_i = P + ß \cdot \mathbf{u}_i \;\vee\; P_i = P - ß \cdot \mathbf{u}_i$$ □

La condición contenida en la definición garantiza el que el sólido sea homogéneo. Cuando la dimensión del cuerpo es 3, diremos que es un sólido.

Teorema 5.1: El objeto, O, generado por barrido de dos objetos, O_1 y O_2, es un sólido correcto de dimensión 3 si

- O_1 y O_2 son cuerpos correctos.
- dimensión(O_1) + dimensión(O_2) = 3
- El conjunto de puntos producto de O_1 y O_2 es finito para cualquier punto.

Demostración:

Uno de los dos cuerpos que interviene en el producto debe de tener dimensión dos y el otro dimensión uno, supongamos que son el O_2 y el O_1 respectivamente.

El volumen de O_2 se puede describir usando una base (u_1,u_2). A dicha base se le puede añadir un tercer vector, u_3, ortogonal a los dos primeros para formar una base de $\mathbf{R}^3$.

Del mismo modo, de O_1, podemos obtener una base de $\mathbf{R}^3$, (v_1,v_2,v_3), de la que el primero permite generar el volumen del objeto.

El objeto generado por barrido de $O_1=(V_1,\mu_1,\alpha_1)$ y $O_2=(V_2,\mu_2,\alpha_2)$ es

$$O_1 \times O_2 = (V, \mu, \alpha)$$

siendo:

$$V = \Pi(O_1,O_2) = \{ P \in \mathbf{R}^n \mid \delta(O_1,O_2,P) \neq \phi \}$$

$$\mu(P) = \Sigma_{\delta(O1,O2,P)} \mu_1(P_1) \times \mu_2(P_2)$$

$$\alpha(P) = \Sigma_{\delta(O1,O2,P)} \alpha_1(P_1) \times \alpha_2(P_2)$$

el conjunto de puntos producto es

$$\delta(O_1,O_2,P) = \{ (P_1,P_2) \mid P_1 \in V_1, P_2 \in V_2 \text{ y } P = P_1+P_2 \}$$

queremos demostrar que

$$\forall P \in V \ \exists Q_1,Q_2,Q_3 \in V \mid Q_i = P + ß \cdot \mathbf{u}_i \vee Q_i = P - ß \cdot \mathbf{u}_i$$

siendo $\mathbf{u}_i$ el vector i-esimo de la base en P.

tomemos como base, en P, la correspondiente al objeto de dimensión dos, O_2. Se cumple que:

$$\forall P \in V \ \exists P_1 \in V_1, P_2 \in V_2 \mid P = P_1 + P_2$$

y dado que O_2 es un cuerpo correcto

$$\forall P_2 \in V_2 \ \exists Q_{21},Q_{22} \in V_2 \mid Q_{2i} = P_2 + ß \cdot \mathbf{u}_i \vee Q_{2i} = P_2 - ß \cdot \mathbf{u}_i$$

se tiene

$$\forall P \in V \ \exists Q_1,Q_2, \in V \mid Q_i = P \pm ß \cdot \mathbf{u}_i = Q_{2i} + P_1$$

Para comprobar que el objeto es homogéneo en P en la tercera dirección se debe de tener en cuenta que O_1 es un cuerpo de dimensión 1, y que por tanto es homogéneo en una dirección, $\mathbf{v}_1$

$$\forall P_1 \in V_1 \ \exists Q_{11} \in V_1 \mid Q_{11} = P_1 + ß \cdot \mathbf{v}_1 \vee Q_{11} = P_1 - ß \cdot \mathbf{v}_i$$

y que necesariamente $\mathbf{v}_1$ debe de ser perpendicular a $\mathbf{u}_1$ y $\mathbf{u}_2$. Probemos este último hecho por reducción al absurdo. Supongamos $\mathbf{v}_1$ es combinación lineal de $\mathbf{u}_1$ y $\mathbf{u}_2$, entonces

$$\exists \lambda_1,\lambda_2 \in \mathbf{R} \mid \mathbf{v}_1 = \lambda_1 \cdot \mathbf{u}_1 + \lambda_2 \cdot \mathbf{u}_2$$

y por tanto, $\exists\ ß \in \mathbf{R}$ suficientemente pequeño para que Q_{11} se pueda poner en la forma

$$Q_{11} = P_1 + ß \cdot \lambda_1 \cdot \mathbf{u}_1 + ß \cdot \lambda_2 \cdot \mathbf{u}_2$$

este punto de O_1 contribuirá a la generación del punto P'

$$P' = Q_{11} + P_2$$

pero dicho punto también se generará a partir del par

$$P' = P_1 + (P_2 + ß \cdot \lambda_1 \cdot \mathbf{u}_1 + ß \cdot \lambda_2 \cdot \mathbf{u}_2)$$

y por continuidad de la transformación, todos los pares de puntos intermedios también generarán P', por lo que su conjunto de puntos producto sería infinito, que esta en contra de la hipótesis del teorema.

Por tanto hay que concluir que $\mathbf{v}_1$ es ortogonal a $\mathbf{u}_1$ y $\mathbf{u}_2$ en P, y por tanto el objeto O es homogéneo en P. □

5.3 ANIMACION.

En esta sección se expone modelo formal de animación, desarrollado a partir del álgebra de objetos gráficos, con el propósito de que sirva de ilustración de la utilización del álgebra en el desarrollo de conceptualizaciones específicas.

Una animación, o película, es una secuencia de vistas tomadas de un escenario que cambia con el tiempo. El estado inicial del escenario se puede considerar un objeto en $\mathbf{R}^3$, formado por la combinación de una serie de actores, colocados en posturas determinadas[(1)]. Los sucesivos cambios que ocurran en el escenario se podrán considerar como funciones que se aplican a los actores.

En conjunto, la animación, se puede considerar como un objeto en $\mathbf{R}^4$, si el escenario es tridimensional, siendo el tiempo la cuarta dimensión. Para cualquier valor del tiempo se tendrá un objeto tridimensional, que será la configuración del escenario en ese instante. En una animación bidimensional, las dimensiones de los espacios serán de un orden inferior.

El escenario se podrá descomponer en un conjunto finito de objetos, que serán los actores. La combinación de estos en el escenario estará dada por una expresión de objetos. Si no se permite que los objetos se solapen, esta expresión puede ser simplemente una suma de los objetos actores. Podremos, así mismo, considerar los actores en movimiento como objetos de $\mathbf{R}^4$ que se combinan para formar la animación.

El problema que nos ocupa es el de la definición formal de la animación. Dicha formalización debe permitir detallar cada una de las imágenes que intervienen en la animación. Este número puede ser demasiado grande para realizar la especificación imagen a imagen. Por este motivo se suelen especificar tan sólo una secuencia de imágenes clave ('KeyFrame') [Thal85] que determinan univocamente al resto de las imágenes de la película. Las imágenes clave, que llamaremos aquí viñetas, son en sí una subsecuencia de la animación. Entre cada dos imágenes clave se generan por interpolación una serie de imágenes intermedias ('in-between'), que junto con las viñetas forman el conjunto de la animación. Esta forma de trabajo coincide básicamente con la utilizada clásicamente en la realización de animaciones.

Para poder generar las imágenes intermedias de forma automática es necesario haber definido cada viñeta a partir de la anterior como un cambio de los objetos que intervenían en aquella. Podemos considerar dos tipos de cambios:

1· Los que no afectan a la estructura del escenario: Cambios de posición, aspecto u otras propiedades de los objetos.

2· Los que modifican la estructura del escenario: cambio en las operaciones de combinación de objetos, incorporación o retirada de actores.

---

(1) Nótese que los actores se consideran parte integrante del escenario, por tanto nuestro concepto de escenario no coincide con el de decorado.

El tratamiento de ambos tipos de cambios debe de realizarse de un modo diferente. Los cambios que no modifiquen la estructura del escenario se pueden parametrizar, permitiendo la generación de las imágenes intermedias asignando distintos valores al parámetro.

Estos cambios se pueden modelar mediante funciones de objetos. Cada viñeta se genera a partir de la anterior aplicándole una secuencia de filtros parametrizados en el intervalo [0,1], cada uno de los cuales afecta a un subconjunto de objetos del escenario. La acción del filtro para valor 0 del parámetro debe de corresponder con la identidad y la acción para valor 1 se corresponde con el movimiento a realizar para pasar a la viñeta siguiente. Las imágenes intermedias se generan asignando valores al parámetro en el intervalo [0,1].

Las acciones que modifican la estructura del escenario no se podrán parametrizar, y por tanto deberán de realizarse de forma instantánea en una viñeta. Para que su ejecución no produzca discontinuidades en la secuencia de imágenes, se debe de asegurar que su acción no modifique la apariencia visible de la viñeta en que se aplican.

A continuación se formalizarán estos conceptos.

Formalmente podremos definir un actor como:

Definición 5.3: Un actor es una terna $\{\mathbf{S},O_o,\mathbf{F}\}$ en la que **S** es un conjunto de objetos en $\mathbf{R}_3$, que denominaremos conjunto de posturas del actor, O_o es un objeto de **S**, que denominaremos postura inicial, y **F** es un conjunto de funciones de la forma:

$$f \in \mathbf{F}: [0,1] \times \mathbf{S} \longrightarrow \mathbf{O}$$

que denominaremos acciones, cumpliéndose:

$$\forall f \in \mathbf{F}, s \in \mathbf{S} \quad f(0,s) = s \quad y \quad f(1,s) \in \mathbf{S} \subset \mathbf{O}$$

$$\forall O \in \mathbf{S} \quad \exists f_1,f_2,..,f_k \in \mathbf{F} \mid \quad O = f_k(1,f_{k-1}(1,..,(1,f_1(O_o))..)$$

$$\forall f \in \mathbf{F} \ \exists g \in \mathbf{F} \mid \forall s \in \mathbf{S}, \forall t \in [0,1] => f(t,s) = g(1-t,s)$$

$$\hat{1} \in \mathbf{F}$$

□

La primera condición garantiza que para todas las aplicación de cualquier acción a cualquier postura, no la modifica para valor del parámetro 0, y genera una nueva postura para valor 1. Para valores intermedios del parámetro se puede generar un objeto cualquiera.

La segunda condición garantiza el que todas las posturas sean alcanzables. La tercera asegura que todas las acciones tengan inversa.

Obsérvese que el conjunto de posturas del actor puede ser finito independientemente del número de imágenes intermedias que se generen, gracias a que las imágenes de las funciones de **F** están en **O** en lugar de en **S**.

A partir de un actor podemos construir un objeto en cuatro dimensiones, es decir un objeto en movimiento. Para ello se especificarán una postura s y una acción f del actor y un intervalo de tiempo inicial $[t_0,t_1]$ en el que se realizará el movimiento. Se le aplicará a dicho actor la acción f que lo colocará en una nueva postura s_1, realizando dicha acción en un tiempo t_1-t_0. La aplicación de la acción definirá un objeto con extensión temporal entre t_0 y t_1.

Un escenario se puede definir como un conjunto de actores que se encuentran en determinadas posturas.

Definición 5.4: Un escenario de orden k es un par $\xi=(\mathbf{A},\omega)$ en el que **A** es un conjunto ordenado de k actores $\mathbf{A} = \{A_1,A_2,..,A_k\}$, ω es un conjunto ordenado de k posturas $\omega = \{ s_i \mid \forall i\in[1..k]\ s_i \in A_i\}$. □

Existe la posibilidad de que los actores se superpongan en el espacio, por ello será necesario determinar la forma en que se componen. La forma de composición puede indicarse como una expresión de objetos, que operará sobre la configuración actual de los actores.

Definición 5.5: Una función de composición ζ sobre un escenario de orden k es una expresión de objetos ζ: $\mathbf{O}^k$ -> **O**, que determina la forma en que se combinan los k objetos en el escenario. □

Para obtener una viñeta, a partir de la anterior, se debe de modificar el escenario. Esta modificación puede modelarse como una función que actúe sobre escenario.

Definición 5.6: Una función de animación $\mathcal{f}$ sobre un escenario de orden k, $\xi = (\{A_1,A_2,..,A_k\},\{s_1,s_2,..,s_k\},\zeta\}$, es una terna $(\mathcal{f},\Delta t,\zeta)$ en el que $\mathcal{f}$ es una k-upla de acciones $\{f_1,..,f_n\}$ que cumplen $f_i\in A_i$, Δt es número real positivo, que denominaremos tiempo de la animación, y ζ es una función de composición.

El paso de una viñeta a otra se realiza en un tiempo Δt, durante el que cada actor cambia de la postura s_i a la postura $s_{i+1} = f_i(1,s_i)$. Durante el cambio de viñeta se utiliza la función de composición especificada.

Una animación puede definirse como una secuencia de funciones de animación aplicadas a un escenario.

Definición 5.7: Una **animación** de orden k es un par $(\xi,\mathcal{F})$ en el que ξ es un escenario de orden k y $\mathcal{F}$ es una secuencia finita de funciones de animación sobre el escenario ξ. □

5.4 CONCLUSION.

Este capítulo ha mostrado tres aplicaciones de la teoría expuesta en los capítulos precedentes, con el propósito de ilustrar las posibles áreas de aplicación del formalismo. Se ha esbozado la utilización de la teoría en la formalización de un sistema de dibujo, se ha dado una condición necesaria para que un objeto generado por barrido sea un sólido, y se ha presentado un formalización de sistemas de animación que utiliza como soporte la teoría de objetos.

6. ESPECIFICACION Y DESARROLLO DE SISTEMAS GRAFICOS.

6. ESPECIFICACION Y DESARROLLO DE SISTEMAS GRAFICOS.

Hasta aquí nos hemos ocupado de establecer un formalismo teórico, para representar objetos gráficos, y de obtener algunas de las propiedades que de él se derivan. El presente capítulo se dedica a estudiar una implantación de esta teoría en técnicas y metodologías de especificación.

En este capítulo se utilizará como técnica de especificación la especificación algebraica. Para dicha técnica se estudia la incorporación del álgebra de objetos como soporte de la especificación, su adaptación para sistemas gráficos y su extensión para sistemas interactivos.

Las metodologías se utilizan para derivar, de forma sistemática, la especificación de un sistema. En la sección 6.3 se propone una metodología de especificación de sistemas gráficos.

6.1 ESPECIFICACION FORMAL DE SISTEMAS GRAFICOS.

De acuerdo con lo expuesto en la sección 1.3.3, es aconsejable utilizar métodos formales para describir la especificación de cualquier sistema informático, y en particular de sistemas gráficos. Esto facilita el realizar una implementación correcta y que se pueda verificar tanto la especificación como la implementación.

Por otra parte, dado que el formalismo teórico propuesto es un álgebra, parece natural describir la especificación usando especificación algebraica.

La utilización de esta técnica de especificación, en combinación con el álgebra de objetos gráficos, posee las ventajas de permitir incorporar las operaciones del álgebra de objetos en los módulos de especificación, y de utilizar un formalismo homogéneo.

No obstante, nos encontramos con dos problemas fundamentales:

· Desarrollar mecanismos, en el lenguaje de especificación, que permitan inferir el resultado, a nivel de imagen, de las funciones que se definan.

· Incorporar de un modo efectivo el álgebra de objetos al lenguaje de especificación usado.

En el resto de esta sección se abordarán estos dos problemas.

La especificación de un sistema fija su comportamiento funcional, en base a las acciones que se pueden realizar con él, y a su estado interno. Cuando el sistema es un sistema gráfico interesa, no sólo especificar sus acciones, sino también el resultado gráfico, en el dispositivo de salida, de dichas acciones. Es decir, no nos basta con que al preguntar, desde el programa, el color de un pixel que se acaba de asignar a 'verde' se obtenga valor 'verde', sino que además necesitamos que el usuario realmente 'vea' el pixel de color 'verde'. La dificultad en la especificación de la imagen que se genera, radica en el hecho de que la imagen es una interfase del sistema y no una parte de su estado interno.

Por tanto, no es posible especificar la imagen imponiendo condiciones que se refieren a lo que ocurre sólo a un lado de la interfase, hay que considerar necesariamente los dos lados: el del sistema y el del usuario. Consideramos que esto sólo es posible de dos modos:

· Prefijando la estructura de la interfase, es decir de la imagen, lo que permitirá inferir su comportamiento en función de sus estados. Esta opción tiene el grave inconveniente de prefijar el tipo de dispositivo de salida. Usualmente se asume que se utiliza un dispositivo raster.

· Permitiendo que el lenguaje de especificación usado pueda expresar la descomposición de los distintos elementos de dibujo en elementos más simples, que

sean tratables por el dispositivo de salida. De este modo, si se cambia de tipo de dispositivo solo será necesario modificar esta descomposición.

La última opción presenta la ventaja de permitir que la especificación sea más independiente del dispositivo. Hay que tener en cuenta que, en este caso la especificación algebraica no contendrá información sobre la visualización de los elementos básicos (directamente tratables por el dispositivo), de los que se deberá especificar su apariencia en términos de imagen (p.e. utilizando una implementación de referencia, dando una imagen patrón, o usando lenguaje natural). De hecho se está asumiendo un determinado comportamiento del dispositivo ante dichos elementos básicos (p.e. punto o línea), y expresando el resultado gráfico del resto de las operaciones en función de estas (p.e. una poligonal se expresará a partir de imágenes de líneas).

Para comprobar que una implementación es correcta se deberá asegurar, por comprobación visual, que la imagen de los elementos básicos es correcta, pudiendo, a partir de esto, verificarse que el resto de las imágenes generadas son correctas.

En la práctica la especificación de un elemento será un módulo de especificación, estando los aspectos de visualización confinados en un módulo especial. Más adelante se expondrá una forma de extender un lenguaje de especificación algebraica para incorporar estas ideas.

Para poder hacer uso del álgebra de objetos gráficos en el lenguaje de especificación, este último debe permitir la definición de elementos indicando que son objetos gráficos. De dichos elementos, la especificación deberá determinar su volumen y sus funciones de aspecto y presencia. Indicar que un elemento es un objeto gráfico supondrá que con dicho elemento se puedan realizar todas las operaciones definidas en el álgebra de objetos gráficos.

La relación existente entre objetos gráficos e instancias permitirá en la práctica que en la implementación se utilicen, indistintamente unos u otras para representar los objetos.

En el apéndice A se muestra la sintaxis del lenguaje de especificación algebraica ALPLA [Torr91], que contempla los dos aspectos descritos. El resto de esta sección se dedicará a comentar la incorporación de los dos aspectos comentados previamente en dicho lenguaje de especificación. Nuestro propósito no es discutir el lenguaje de especificación en sí, ni su implementación, si no, tan solo utilizarlo como ilustración de como las ideas expuestas se pueden incorporar a un lenguaje de especificación real.

Una especificación algebraica se compone de un conjunto de módulos, cada uno de los cuales especifica un tipo abstracto de datos. En la definición de un módulo, en ALPLA, se puede indicar que dicho módulo representa a un objeto gráfico. Para los módulos que se declaren como objetos gráficos se deberán incluir las funciones consultoras presence y aspect, que, para cada punto, determinarán la presencia, y el aspecto del objeto. En el caso de que el objeto se construya a partir de otros objetos gráficos, usando funciones de modelado, las funciones de aspecto y presencia quedarán automáticamente especificadas.

En el lenguaje se incluyen, como tipos de datos elementales, puntos y líneas, que se pueden construir con las siguientes funciones:

_point: Int × Int × Colour -> Point

_line: Point × Point × Colour -> Line

El volumen del objeto se puede determinar a partir de la información anterior de este modo se garantiza que la definición del objeto sea correcta.

Ejemplo 6.1:

El siguiente módulo es una especificación funcional de un rectángulo.

Graphic Object Rectángulo

Constructors

_rectángulo: Point × Point × Colour -> Rectángulo

Functions

aspect: Point × Rectángulo -> Colour
presence: Point × Rectángulo -> Int
is_in: Point × Rectángulo -> Bool

Axioms

var P, P_1, P_2: Point; C: Colour; R: Rectángulo

is_in(P,_rectángulo(P_1,P_2,C)) = if ((P > P_1) and (P < P_2)) true
else false
aspect(P,_rectángulo(P_1,P_2,C)) = if ((P > P_1) and (P < P_2)) C
else 0
presence(P,R) = if (is_in(P,R)) 1
else 0

La utilización de operaciones de modelado se hace necesaria cuando se construyen objetos complejos, a partir de otros más simples. En este caso, no será necesario especificar explícitamente las funciones de aspecto y presencia. El rectángulo del ejemplo anterior se podría definir como el producto de dos líneas, tal como se muestra en el ejemplo siguiente.

Ejemplo 6.2:

El siguiente módulo es una especificación funcional de un rectángulo definido por producto de dos líneas.

Graphic Object Rectángulo

Constructors

_rectángulo: Line × Line -> Rectángulo

Functions

aspect: Point × Rectángulo -> Colour
presence: Point × Rectángulo -> Int
superficie: Rectángulo -> Real

Axioms

var L_1,L_2: Line;

_rectángulo(L_1, L_2) = if (dot(L_1,L_2)=0) $L_1 \times L_2$
else 0
superficie(_rectángulo(L_1,L_2)) = if(_rectángulo(L_1,L_2)=0) 0.
else length(L_1)*length(L_2)

La función dot es el producto escalar de las dos líneas consideradas como vectores, se utiliza para asegurar que las dos líneas son perpendiculares.

El 0 en el axioma hace referencia al objeto nulo.

Para especificar la visualización de los elementos se debe de especificar la correspondencia que se establece entre los objetos gráficos y la imagen. El establecimiento de esta correspondencia se puede realizar fijando la relación de visualización (def. 4.11) y la función de combinación (def. 4.12).

En el lenguaje de especificación propuesto la visualización se especifica utilizando un módulo especial, correspondiente al objeto 'display'. Este módulo contiene al menos las funciones:

draw : Graphic_Object x Display -> Display

y

get: ? x Display -> Colour

La primera indica se utiliza para dibujar objetos gráficos en el dispositivo. Los axiomas relativos a esta función deben especificar la forma en que se realiza la visualización.

La función get devuelve el color asignado a un elemento de la imagen en función de algún parámetro, y por tanto permite especificar el valor de color de elementos de la imagen. Si el dispositivo es raster los elementos serán pixels y se seleccionarán con un par de coordenadas enteras.

Además el módulo de visualización puede contener otras funciones que permitan controlar parámetros de visualización (tales como ventanas etc.).

Ejemplo 6.3:

El módulo de visualización para un sistema 2D que utilice un dispositivo de salida raster podría ser:

Graphic Object Display

Constructors

_Display: -> Display

Functions

draw : Graphic_Object x Display -> Display
clear: Display -> Display
get: Int x Int x Display -> Colour
setViewTransform: Point x Point x Display -> Display
getViewTransform: Display -> Geometric_Transformation

Axioms

var O: Graphic_Object; I:Display; x,y: Int

clear(I) = _Display
draw(O,I) = draw(Clip(getWindow,Transform(getViewTransform(I),O)),I)
get(x,y,draw(O,I)) = if (is_in(InvTransform(getViewTransform(I),_2DPoint(x,y)),O)) aspect(O) else get(x,y,I)
get(x,y,_Display) = 0
get(x,y,setViewTransform(P_1, P_2,I)) = get(x,y,I)
getViewTransform(setViewTransform(P_1,P_2,I)) = {{1023/(P_2.x-P_1.x),0,0},{0,1023/(P_2.y-P_1.y),0},{-1023*P_1.x/(P_2.x-P_1.x),-1023*P_1.y/(P_2.x-P_1.x),1}}
getViewTransform(_Display) = {{1,0,0},{0,1,0},{0,0,1}}

En este ejemplo se ha considerado la posibilidad de definir ventanas. El dibujo se realiza sobre todo el dispositivo. Se ha asumido que el dispositivo utiliza coordenadas entre 0 y 1023.

La función transform transforma un objeto según una matriz de transformación geométrica.

6.2 INTERACCION.

Nos ocuparemos a continuación de la especificación de la interacción. Para ello será necesario disponer de métodos que permitan especificar restricciones de sincronización. Nuestro propósito es utilizar un formalismo compatible con la especificación algebraica, utilizada para el resto del sistema. La conveniencia de utilizar un formalismo común fue defendida por W. Mallgren [Mall82]. Sin embargo, el formalismo propuesto por W. Mallgren, basado en la inclusión de axiomas sobre el estado del sistema, da lugar a especificaciones complejas, y no permite la comprobación de la vivacidad del sistema (ver Secc.1.3.4).

Por el contrario, proponemos la utilización de mecanismos explícitos de sincronización. Antes de exponer la estructura de una especificación, analizaremos las especificaciones algebraicas de sistemas secuenciales, para determinar con precisión lo que se les debe añadir para usarlas en sistemas concurrentes. Para ello estudiaremos un caso particular, el de una memoria tampón de un sólo elemento. El tipo abstracto de datos podría ser:

Object Tampon

Constructors
_tampon: -> Tampon

Functions
Get <v,R>: Tampon -> Int × Tampon
Put: Tampon × Int -> Tampon

Axioms
var T:Tampon; x: Int;

Get(Put(T,x)).v = x
Get(Put(T,x)).R = _Tampon

Esta especificación establece que el resultado de obtener el valor del tampón (Get) después de haber colocado un valor cualquier es que el tampón está en el mismo estado en que se encontraba al crearlo (vacío) y que se devuelve el último valor introducido.

El resto de las condiciones que se deben de imponer son condiciones de sincronización que deben prohibir el que:

· se extraiga información de un tampón nuevo o vacío.

· se coloque un nuevo dato en el tampón, cuando está ya completo.

Para poder contemplar estas operaciones es necesario disponer de un modelo de operación que permita el acceso concurrente y contemple información sobre el estado del módulo, en base a la cual se puedan establecer las restricciones de sincronización.

Respecto al modelo de operación, supondremos, que las operaciones se realizan de forma indivisible, y que pueden ser llamadas concurrentemente. Cada operación se podrá ejecutar sólo bajo determinadas circunstancias. Si éstas no se cumplen, el proceso que solicito la operación queda bloqueado antes de realizar la operación. Cuando cambie el estado del sistema de forma que la operación se pueda realizar el proceso es activado y la operación se realiza de forma indivisible. A nivel de esquema cada operación equivale a la siguiente secuencia de código, utilizando regiones críticas condicionales:

```
Operación Op

when Op_permitida do
    Hacer_Op
```

Este esquema implica el que el estado del módulo no se altere por el hecho de llamar a la operación, diferenciandose en esto del esquema de Mallgren. La condición de acceso ('Op_permitida') se debe establecer en base a información del módulo. Esta puede ser:

· El estado del módulo [Mall82]

· Variables del módulo.

· Operaciones.

La primera opción implica el añadir a la especificación funciones para trabajar con el estado, haciendo más difícil de manejar la especificación [Ver Secc. 1.3.4]. Para completar la especificación anterior, y siguiendo la terminología semejante a la usada por Mallgren, se incluiría:

Operaciones

$init	-> estado
Put:wait	estado -> booleano
Get:wait	estado -> booleano
Put$return	estado × entero -> estado
Get$return	estado -> estado

Axiomas
∀ S∈estado x,y∈entero

Put:wait($init,x) = falso
Put:wait(Put$return(S,x),y) = verdadero
Put:wait(Get$return(S),x) = falso

Get:wait($init) = verdadero
Get:wait(Put$return(S,x)) = falso
Get:wait(Get$return(S)) = verdadero

En la segunda opción, al utilizar variables en el módulo, se está, de algún modo, concretando el estado del módulo. Esto implica indudablemente añadir más operaciones y axiomas, pero sobre todo, conlleva el pasar de una especificación operacional a un modelo. En el ejemplo anterior se podría incluir una variable booleana que indicase si el tampón esta lleno. Por otro lado sería necesario especificar el valor de las variables de estado después de cada operación, lo que puede ser complejo dentro del formalismo.

La inclusión de operaciones adicionales es más general, y engloba, en cierto modo, a los dos soluciones anteriores, puesto que, ambas soluciones implican la inclusión de operaciones y la definición de variables de estado es formalmente equivalente a definir operaciones de consulta de dicho estado. En este caso resulta simple el especificar axiomáticamente las operaciones de consulta añadidas. Volviendo al ejemplo se puede añadir una operación de consulta que devolviese el estado de ocupación del tampón, quedando la especificación:

Object Tampon

Constructors
_tampon: -> Tampon

Functions
Get <v,R>: Tampon -> Int × Tampon
Put: Tampon × Int -> Tampon
Lleno: Tampon -> Bool

Axioms
var T:Tampon; x: Int;

Get(Put(T,x)).v = x
Get(Put(T,x)).R = _Tampon
Lleno(Put(T,x)) = true
Lleno(Get(T).R) = false
Lleno(_Tampon) = false

Las restricciones de sincronización se pueden expresar fácilmente a partir de la operación 'Lleno':

Get debe de esperar a que Lleno sea verdadero

Put debe de esperar a que Lleno sea falso

Estas condiciones se podrían especificar formalmente utilizando las funciones wait de Mallgren:

Get:wait(T) = ¬Lleno(T)

Put:wait(T,x) = Lleno(T)

No obstante, consideramos que la inclusión de estas operaciones, wait, sobrecarga la especificación, al mezclar las restricciones operacionales con las restricciones de sincronización. Por el contrario, proponemos la inclusión en la especificación de una sección adicional, para agrupar las restricciones de sincronización. En esta sección aparecerá a lo sumo una expresión para cada operación. Dicha expresión puede ser una sentencia condicional, con un significado semejante al de la guarda de una región crítica condicional. Con este formalismo el ejemplo del tampón queda finalmente del siguiente modo:

Object Tampon

Constructors

_tampon: -> Tampon

Functions

Get <v,R>: Tampon -> Int × Tampon
Put: Tampon × Int -> Tampon
Lleno: Tampon -> Bool

Axioms

var T:Tampon; x: Int;

Get(Put(T,x)).v = x
Get(Put(T,x)).R = _Tampon
Lleno(Put(T,x)) = true
Lleno(Get(T).R) = false
Lleno(_Tampon) = false

Syncronization

do Get(T) when Lleno(T)
do Put(T,x) when not Lleno(T)

Denominaremos guarda de una operación a la condición que aparece en la expresión 'when' de dicha operación. Así, la guarda de Get es Lleno y la guarda de Put es ¬Lleno.

La separación de los dos tipos de restricciones permite establecer la corrección de la especificación de ambos componentes por separado. Respecto a las restricciones operaciones la especificación debe de ser completa y consistente (ver secc. 5.1). Es fácil comprobar que la especificación anterior cumple ambas propiedades.

La corrección de las restricciones de sincronización, debe de implicar al menos la vivacidad del módulo, es decir que no existan situaciones en las que alguna operación quede por siempre prohibida.

Esta idea de corrección es bastante natural. Supongamos una máquina de calcular mecánica, que es operada con un conjunto de manivelas. Si la maquina puede alcanzar un estado en el que una de las manivelas queda bloqueada, sin que se pueda desbloquear accionando el resto de las manivelas, diremos que la máquina no funciona correctamente.

A nivel de implementación esta idea equivale a que el módulo esté libre de interbloqueos, lo cual no significa que el sistema que utiliza al módulo lo esté.

6.2.1 Corrección de sistemas interactivos.

En esta sección se formalizará el concepto de corrección de las restricciones de sincronización de una especificación. A partir de esta formalización se establecerá una condición necesaria y suficiente para que la especificación sea correcta.

Definición 6.1: Una **operación libre** es una operación que puede modificar el estado del módulo y que o bien no aparece en la sección de sincronización o bien aparece, pero su condición de espera es verdadera. □

En una especificación dada podrá haber operaciones que siempre sean libres y operaciones que estén libres en determinados estados del módulo. Así, en la especificación del tampón dada anteriormente no hay ninguna operación que sea siempre libre, pues Lleno no modifica el estado del módulo, Get está libre sólo cuando Lleno es verdadero y Put está libre sólo cuando LLeno es falso.

Definición 6.2: Un **interbloqueo** es un estado del módulo en el que al menos una operación esta bloqueada por acción de una restricción de sincronización, cuya condición no puede ser modificada por ninguna operación libre. □

Un interbloqueo es, por tanto, un estado en el que una operación está esperando un suceso que no puede ocurrir nunca, coincidiendo con la acepción usual [Holt83].

Tal como se ha indicado anteriormente, la definición de interbloqueo es interna a la especificación del módulo. Un determinado programa puede, por utilizar inapropiadamente un módulo correcto llegar a una situación de bloqueo. Supóngase, por ejemplo un programa formado por dos procesos que escriben en el tampón anterior.

La definición anterior incluye como interbloqueos, estados en los que puede que alguna operación se pueda ejecutar, es decir esté libre. Por tanto, esta definición de interbloqueo es más fuerte que la dada por Kessel [Kess77].

Definición 6.3: Una especificación es **formalmente correcta** si su especificación operacional es completa y consistente, y su especificación de sincronización asegura que el módulo no pueda alcanzar estados de interbloqueo. □

Para caracterizar la corrección de la especificación introducimos dos conceptos auxiliares:

Definición 6.4: Un conjunto de **funciones sincronizadas**, f_s, es un subconjunto no vacío de funciones que aparecen en la sección de sincronización. □

En el ejemplo anterior existen los siguientes f_s: {Get,Put},{Get},{Put}.

Definición 6.5: El conjunto de **funciones de control**, f_c, de un conjunto de funciones sincronizadas, f_s, para un estado del sistema, es el conjunto de funciones que pueden hacer que la guarda de alguna operación de f_s se haga verdadera.

En el ejemplo anterior, los f_c, asociados a cada f_s, en cualquier estado salvo el inicial, son:

$f_s = \{Get,Put\} \Rightarrow f_c = \{Get,Put\}$

$f_s = \{Get\} \Rightarrow f_c = \{Put\}$

$f_s = \{Put\} \Rightarrow f_c = \{Get\}$

Con estos conceptos podemos dar una condición necesaria y suficiente para que un especificación este libre de interbloqueos, y por tanto sea correcta en cuanto a sincronización en el sentido de la definición anterior.

Lema 6.1: Para una especificación dada, aparece un interbloqueo si y sólo si para algún estado del módulo hay un conjunto de funciones sincronizadas f_s, cuyo conjunto de funciones de control asociado f_c, está contenido en él, $f_c \subset f_s$, siendo todas las guardas de las funciones de f_s simultáneamente falsas.

Demostración:

Demostremos en primer lugar que si se produce un interbloqueo se cumple necesariamente la condición del lema.

En efecto, supongamos que se ha producido en interbloqueo, entonces hay un conjunto de funciones que se encuentran suspendidas. Llamemos f_s a dicho conjunto. f_s es un conjunto de funciones sincronizadas, y las guardas de estas funciones deben ser todas falsas, ya que están suspendidas. Por estar el sistema en un interbloqueo no debe de ser posible que ninguna operación libre haga que las guardas de las funciones en f_s tomen valor verdadero, luego el conjunto de funciones de control asociado a f_s debe de estar necesariamente contenido en él.

Demostraremos seguidamente que el recíproco también es cierto. Supongamos que se cumplen las condiciones del teorema. Esto es: el módulo puede alcanzar un estado en el que hay un conjunto de funciones f_s, cuyas guardas son todas falsas y cuyo conjunto de funciones de control está contenido en él. Entonces no es posible que ninguna operación libre modifique las guardas de las funciones de f_s, y por tanto el módulo alcanzará un interbloqueo si llega a dicho estado. □

Teorema 6.1: Una especificación está libre de interbloqueos si y sólo si no es posible, para ningún estado del módulo, que exista un conjunto de funciones sincronizadas cuyas guardas sean simultáneamente falsas y cuyo conjunto de funciones de control esté contenido en él.

Demostración:

Trivial. □

Es fácil comprobar que la especificación del tampón dada en el epígrafe anterior está libre de interbloqueos. Para ello basta con estudiar las relaciones entre los f_s y los f_c. En efecto, una vez inicializado el sistema, y cualquiera que sea , su estado, existen tres f_s, cuyos correspondientes f_c son, tal como se vio anteriormente:

$f_s = \{Get,Put\} \quad => \quad f_c = \{Get,Put\}$

$f_s = \{Get\} \quad => \quad f_c = \{Put\}$

$f_s = \{Put\} \quad => \quad f_c = \{Get\}$

para el primero no es posible que sean simultáneamente falsas las guardas de las dos funciones y en los dos restantes no se cumple la condición de inclusión, no cumpliéndose, por tanto, en ningún caso la condición del teorema, por lo que la especificación es correcta.

6.3 METODOLOGIA DE ESPECIFICACION.

La técnica descrita en la sección anterior se puede utilizar para describir la especificación de sistemas gráficos. En esta sección nos plantearemos otro problema fundamental en el desarrollo de sistemas gráficos: como generar una especificación correcta.

El problema de especificación es complejo. Se ha abordado desde distintos puntos de vista, adaptándose cada uno a un tipo de problemas y a determinadas filosofías de diseño. Obtener una especificación correcta es, en esencia, generar una descripción lo suficientemente precisa, completa y consistente, y a ser posible formal, de un sistema.

Este problema, tal como lo hemos planteado, no es propio de la informática. En un sentido amplio, un mapa puede ser considerado una especificación de un territorio, ya que es una descripción correcta, precisa y formal del mismo. Las diferencias principales que aparecen entre el proceso de generación de un mapa y la especificación de un sistema informático son dos:

· Para los sistemas informáticos no existe un método de contraste. El mapa es un reflejo de algo existente (el terreno) con el que se puede comparar para comprobar su corrección. Por el contrario, la especificación de un sistema informático suele formar parte de un proyecto de realización, no existiendo, por tanto, una realidad con la que compararla, antes de realizar la implementación.

· No existe un método unificado y universalmente aceptado para describir el software. Esto es valido a cualquier nivel de descripción, desde especificación hasta implementación. Esta carencia, muy posiblemente, no sea un problema de falta de desarrollo de la disciplina, sino más bien una consecuencia inmediata de su complejidad.

Para paliar estas deficiencias es necesario:

· Utilizar una sistemática en la generación de la especificación, que en sí garantice la corrección de la misma.

· Utilizar un método de representación apto, al menos, para el campo propio de la aplicación.

Como técnica de representación de la especificación se puede usar, tal como se ha expuesto anteriormente, la especificación algebraica, con las variantes descritas para sistemas gráficos interactivos.

La metodología de especificación debe de aprovechar la naturaleza gráfica de los problemas que nos ocupan. Consideramos que es posible obtener una especificación de un modo sistemático, aplicando los principios del diseño orientado a Objetos y del diseño de interfases de usuarios. Concretamente, se propone la siguiente secuencia de etapas para la generación de la especificación:

1· Identificar los objetos gráficos a utilizar.
De cada objeto se debe de establecer su naturaleza, para lo cual se puede estudiar el proceso de abstracción seguido para obtenerlo del mundo real, indicando lo que representa (es decir de que es abstracción) y como se debe realizar su visualización. Este primer paso se puede llevar a cabo estudiando la información de entrada y salida que se desea manejar.

2· Estudiar los atributos relevantes de cada objeto gráfico.
Se realizará estudiando las características de los objetos reales, determinando cuales son importantes en el contexto del problema a resolver.

3· Estudiar la estructura estática de los objetos.
Se deberá encontrar las dependencias existentes entre objetos, debidas a su definición. Las dependencias más frecuentes son relaciones de inclusión o composición.

4· Estudiar las acciones a realizar con cada objeto.
Las acciones podrán afectar a todo el objeto o sólo a determinados atributos. Estas estarán ligadas al problema a resolver o frecuentemente al proceso de edición. Estas acciones deben permitir enunciar una solución abstracta del problema.

5· Estudiar la estructura dinámica.
Se deberán encontrar las dependencias existentes entre objetos debidas a las acciones que realizan. Las más frecuentes son debidas a uso o similitud.

6· Establecer la jerarquía de objetos gráficos.
En la jerarquía deben quedar recogidas las distintas dependencias entre objetos.

7· Realizar una especificación algebraica del sistema.
La especificación algebraica podría realizarse utilizándose el lenguaje descrito en la secc 6.1.

El proceso descrito puede utilizarse haciendo uso de la teoría de objetos, lo que permitiría expresar determinadas dependencias entre objetos formalmente, o haciendo un uso de los objetos gráficos solo a nivel conceptual.

Al especificar un sistema puede ser necesario iterar sobre etapas anteriores, esto ocurrirá siempre que se detecte la ausencia de algún elemento de una etapa ya realizada.

Al realizar la implementación, la jerarquía de objetos gráficos dará lugar a una jerarquía de clases (en el sentido de la PDO), en la que cada objeto gráfico se habrá implementado como un objeto de programa.

Ejemplo 6.4:

A modo de ejemplo realizaremos las seis primeras etapas de la especificación de un sistema de visualización de escenas usando Ray Tracing. El sistema gestionará una escena formada por elementos geométricos y modelizará el proceso de rendering. Brevemente, el proceso de especificación sería:

1· Identificar los objetos gráficos a utilizar. Los objetos aparecen en negrita.

Elementos: componentes geométricos a dibujar. Pueden representarse como **elementos simples** o como árboles **CSG** de elementos.
Escena: conjunto de elementos.
Cámara: dispositivo que genera la imagen.
Focos: fuentes puntuales de luz.
Superficies: superficies que delimitan los elementos, y que poseen un comportamiento conocido ante la luz.
Textura: textura de los elementos.

2· Estudiar los atributos relevantes de cada objeto gráfico.

Elementos:
· Colocación en el espacio (matriz de transformación.
· Superficie.
· textura.

Escena:
· Elementos.
· Focos.

Cámara:
· Posición del observador.
· Centro de atención.
· Plano de visión.

Focos:
· Posición.
· Color.
· Intensidad.

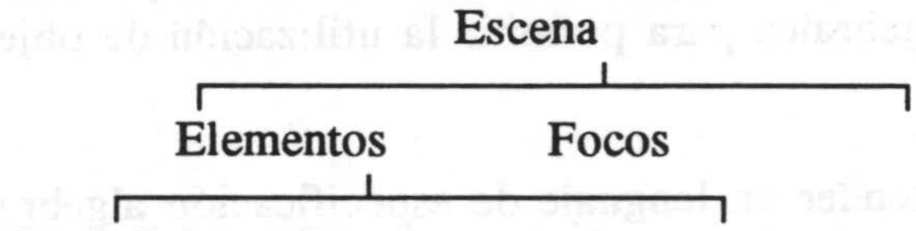
Superficie:
· Coeficientes de reflexión.
· Coeficientes de refracción.

Texturas:

· Color (especificado para cada punto).

3· Estudiar la estructura estática de los objetos.

De la descripción anterior se pueden derivar las siguientes jerarquía



4· Estudiar las acciones a realizar con cada objeto.

Escena:

· Sintetizar la imagen.
· Calcular color en una dirección.
· Añadir elemento.
· Añadir focos.

Elementos:

· Test de intersección con rayo.
· Cálculo del color emitido por un punto de la superficie según un dirección.
· Cálculo de color de un punto de superficie.

Cámara:

· Generar un rayo partiendo del observador y que pase por un pixel.
· Cambiar posición del observador.
· Cambiar centro de atención.
· Cambiar plano de visión.

Superficie:

· Calculo del color de luz emitido por superficie.

Texturas:

· Obtención del color de un punto.

Una discusión detallada de la estructura dinámica, y de la implementación usando PDO, se puede encontrar en [Ureñ92].

6.4 CONCLUSIONES.

Una de las posibles aplicaciones de la teoría de objetos gráficos es su utilización como base en la especificación de sistemas gráficos. En este capítulo se ha mostrado la posibilidad de extender un lenguaje de especificación algebraica para permitir la utilización de objetos gráficos.

Así mismo, se ha mostrado como extender un lenguaje de especificación algebraica para permitir la especificación de sistemas gráficos interactivos, utilizando axiomas de sincronización. Para una especificación de este tipo se han dado una condición necesaria y suficiente para que el sistema esté libre de interbloqueos.

Por último se ha propuesto una metodología de especificación, inspirada en la teoría de objetos, que permite derivar de un modo sistemático una especificación de cualquier sistema gráfico.

7. CONCLUSIONES Y PRINCIPALES APORTACIONES.

7. CONCLUSIONES Y PRINCIPALES APORTACIONES.

Se ha presentado una teoría matemática que puede utilizada, como lenguaje de representación, en el desarrollo de representaciones abstractas de sistemas gráficos, permitiendo expreasar tanto aspectos de visualización como la estructura de modelado.

La teoría se basa en el concepto de objeto gráfico. Las características esenciales de los objetos gráficos, que los distinguen de otras abstracciones anteriores, son:

· La consideración de el aspecto del objeto (color, textura, etc.) como parte de la información propia del objeto, y por tanto como información que se utiliza para operar en el momento de combinar objetos gráficos.

· La consideración de la ocupación del espacio como una propiedad cuantitativa, dando origen al concepto de presencia. Este concepto permite la definición de operaciones aditivas, que sobre el volumen carecerían de sentido.

Sobre el conjunto de objetos se han definido las siguientes operaciones: suma, producto por escalar, unión, intersección, complementación, producto de objetos, producto circular de objetos. El conjunto de objetos dotado de estas operaciones posee la siguiente estructura:

$(\mathbf{O}, +)$ es un grupo abeliano

$(\mathbf{O}, +, *)$ es un espacio vectorial

$(\mathbf{O}, \cup, \cap, \sim)$ es un álgebra de boole

$(\mathbf{O}, \times)$ posee elemento neutro, y la ley es asociativa y conmutativa

$(\mathbf{O}, \otimes)$ posee elemento neutro, y la ley es asociativa y conmutativa

Se ha definido formalmente el concepto de transformación geométrica, y en base a este se han definidos las instancias gráficas, como otra representación abstracta de los sistemas gráficos, equivalente a la transformación geométrica de un objeto gráfico. Se ha estudiado la equivalencia entre instancias y objetos, y se han definido sobre las instancias operaciones de modelado equivalentes a las de los objetos. Sobre los objetos gráficos se han definido relaciones de equivalencia, que permiten generar una estructuración de los objetos en familias.

Sobre el conjunto de objetos se han definido funciones, que permiten realizar cambios de atributos de los objetos, caracterizándose dos tipos concretos de funciones: funciones de transferencia de objetos (que aplican los atributos de un objeto a otro) y filtros. Se ha mostrado que estos dos tipos de funciones permiten realizar procesos muy frecuentes en Informática gráfica, tales como: aplicación de textura, asignación de color y recortado. Se ha mostrado la utilización de las operaciones de modelado para construir objetos simples.

El concepto de función, por otra parte, permite completar la jerarquía de abstracciones gráficas, en base a las relaciones inducidas por las funciones en el conjunto de familias de objetos. Se ha mostrado que bajo determinadas condiciones dicha relación inducida es una relación de equivalencia, lo que nos faculta a hablar de clases de equivalencia en el conjunto de familias de objetos.

Se ha mostrado que el formalismo de visualización de Fiume puede aplicarse a los objetos gráficos aquí definidos, ya que el concepto de objeto de Fiume es menos general que el de objeto gráfico. Por otra parte el formalismo aquí presentado puede ser extendido para contemplar la visualización como una función de objetos, garantizando un tratamiento uniforme de todos los conceptos.

La teoría de objetos gráficos puede servir de nexo teórico a varios métodos de modelado de sólidos y superficies. Además permite hacer un tratamiento homogéneo de ambos. Concretamente, se pueden expresar como expresiones de objetos gráficos los sólidos formados por operaciones booleanas (CSG), por enumeración y por barrido. Queda abierto el problema de construir, mediante operaciones de objetos gráficos, sólidos mediante la instancia de su frontera.

Se ha mostrado la posibilidad de extender un lenguaje de especificación algebraica para permitir la utilización de objetos gráficos.

Así mismo, se ha mostrado como extender un lenguaje de especificación algebraica para permitir la especificación de sistemas gráficos interactivos, utilizando axiomas de sincronización. Para una especificación de este tipo se han dado una condición necesaria y suficiente para que el sistema esté libre de interbloqueos.

Por último se ha propuesto una metodología de especificación, inspirada en la teoría de objetos, que permite derivar de un modo sistemático una especificación de cualquier sistema gráfico.

APENDICE.

APENDICE A:

Descripción del lenguaje de especificación ALPLA.

ALPLA es un lenguaje de especificación algebraica, diseñado para permitir la realización de prototipado. El lenguaje contempla la posibilidad de definir elementos como objetos gráficos, de especificar la forma en que se visualizan los elementos y de especificar concurrencia.

Este apendice contiene un resumen de la sintaxis del lenguaje y una breve descripción de su semántica.

La especificación de un sistema en ALPLA consiste en una colección de módulos. La especificación de cada módulo tiene siete partes, algunas de las cuales son opcionales: header, dependencies, constructors, functions, axioms y syncronization.

La sección de cabecera contiene el nombre del módulo, que es también el del conjunto soporte, y opcionalmente una lista de parámetros formales, usados para construir módulos genéricos. La lista de parámetros es una lista de identificadores que se usan como tipos en módulos genéricos. El tipo actual de cada identificador se determina al usar el módulo.

La sección de dependencias contiene una lista de módulos que se usan en el módulo actual. Object_id es un identificador de módulo que se usa en el módulo actual, es decir el módulo actual puede usar su conjunto soporte y todas sus funciones. La lista de tipos de aparecer cuando el módulo usado tiene parámetros.

Los constructores permiten crear nuevos objetos del tipo definido en el módulo. Cada constructor devuelve un solo resultado, del tipo del módulo. La lista de tipos se utiliza cuando el constructor necesita parámetros para crear el objeto.

La sección functions contiene la signatura de las funciones. Las funciones pueden devolver más de un valor, en cuyo caso se incluirán selectores para hacer referencia a cada uno de ellos.

Los axiomas determinan el comportamiento del módulo. Cada axioma es una ecuación cuyo término izquierdo es una función, y su término derecho es una expresión. El axioma indica que la función del termino izquierdo se puede construir usando la expresión del término derecho.

El bloque de sincronización indica que condición debe de cumplirse para que una función se pueda realizar.

```
<Specification_Module> ::= Header [Dependencies] Constructor
                           Functions  Axioms [Syncronization]

<Header>::= [Graphic] Object object_id [ (parameter_id {,parameter_id})
          { Required parameter_id
            {function_id [<selector_id,{selector_id}>] :
              type_id { x type_id} -> type_id { x type_id} } }]

<Dependencies>::= Import { object_id [( type_id {,type_id})]}

<Constructors> ::= Constructors {function_id :
                    [type_id { x type_id}] -> object_id }

<Functions> ::= Functions { function_id
                  [< selector_id, {selector_id} > ] :
                  type_id { x type_id} -> type_id { x type_id} }

<Axioms> ::= Axioms [ var { var_id {, var_id} : type_id ;} ]
       {function_id (Term {,Term}) [.selector_id] = Expression}

<Expression>::= Term  |  if ( Condition ) Term else Term

<Condition>::= ( function_id [( Term {, Term }) [.selector_id]] |
             var_id ) = Term  |  Boolean-Expression

<Term>::= function_id [ (Term {,Term}) [.selector_id] ]  |
          var_id  |  Constant  |  Boolean-Expression  |
               Integer-Expression  |  ERROR

<Syncronization>::= Syncronization
          {do function_id (var_id{,var_id}) when Condtion}
```

Fig. 1: Resumen de la sintaxis de ALPLA.

BIBLIOGRAFIA.

BIBLIOGRAFIA.

BIBLIOGRAFIA.

Baer79 Baer A.; Eastman C.; Henrion M.: "Geometric modelling: a survey". Computer-Aided Design. Sept. 1979. Vol.11, N.5, pp. 253.

Bail89 Bailin,S.C. "An Object-Oriented Requirements Specification Method". (1989) CACM 32.5.608-623.

Beat82 Beatty J.C.; Booth K.S.: "Tutorial: Computer Graphics". IEEE CS. 1982.

Berg89 Bergstra J.A.; Heering J.; Klint P.: "Algebraic Specification". ACM Press Books. Addison-Wesley, 1989.

Bett88 Bettels J.; Bono P.R.; McGinnis E.; Rix J.: "Guidelines for determining when to use GKS and when to use PHIGS". Computer Graphics Forum. Vol.7, N.4. 1988. pp.347-354.

Blac89 Blacke, E.H.; Wisskirchen,P.: "Object-Oriented Graphics". (1989) Tutorial Notes Eurographics'89.

Booc86 Booch G.: "Object-Oriented Development". IEEE Transation on Software Engineering. Feb. 1986. pp. 211-221. (Ed. Peterson G.E.: "Tutorial Object-Oriented Computing. Vol.2: Implementations". IEEE Computer Society Press. 1987. pp. 5.)

Booc87 Booch,G. "Software Engineering with Ada". Benjamin Cummings Pub.Co.Inc. 1987.

Booc91 Booch,G. "Object Oriented Design with Applications". Benjamin/Cummings. 1991.

Bos_88 Van den Bos, J.: "Abstract interaction Tools: A language for User Interface Management Systems". ACM Trans. on Programming Languages and Systems. Apr. 1988. Vol.10, N.2, pp.267.

Böhm84 Böhm W.; Farin G.; Kahmann J.: "A survey of curve and surface methods in CAGD". Computer Aided Geometric Design". Vol.1, N.1, 1984. pp. 1-60.

Bono88 Bono, F.R.; Herman, I.(ed.):"GKS Theory and Practice". Cumming Publis. Comp. 1988.

Bree89 Breen D. E.: "Message-Based Object-Oriented Interaction Modelling". Proceeding of Eurographics'89. North-Holland 1989.

Cabe90 Cabezas R.: "Informática gráfica: la realidad de lo virtual". Informática Gráfica y Comunicación. Inforarte-90. Valencia 1990.

Cars83 Carson, G.S.: "The specification of Computer Graphics Systems". IEEE Computer Graphics & Applications. Sept. 1983. Vol.3, N.6, pp.27.

Casa85 Casale M.S.; Staton E.L.: "An overview of analytic solid modeling". IEEE Computer Graphics & App. 1985. pp. 45.

Coad90 Coad,P.; Yourdon,E. "Object-Oriented Analysis". Yourdon Press/Prentice-Hall. 1990.

Cox_84 Cox, B.J.: "Change in Programming Technology". IEEE Software. Jan. 1984. Vol. , N. , pp. 50-61. (Ed. Peterson G.E.: "Tutorial Object-Oriented Computing. Vol.1: Concepts". IEEE C.S. Press.1987,pp.9

Cox_86 Cox, B. "Object-Oriented Programming. An Evolutionary Approach". Addison-Wesley. 1986.

Dobk88 Dobkin D.P.: "Computacional Geometry -- Then and Now". Ed. Earnshaw R.A.: "Theoretical Foundations of Computer Graphics and CAD". Springer-Verlag 1988. pp.71-109.

Duce86 Duce D.A.; Fielding E.V.C.: "Towards a Formal specification of the GKS output primitives". Eurographics Conference. 1986. (Ed. Bono P.; Herman I.: "GKS Theory and Practice". Springer-Verlag 1988. pp.239)

Duce88 Duce D.A.; Fielding E.V.C.; Marshall L.S.: "Formal specification of a small example based on GKS". ACM Tramsation on Graphics. Jul. 1988. Vol.7, N.3, pp.180-197.

Duce88a Duce D.A.; Jancene P. (edtors): "Eurographics'88". North-Holland, 1988.

Duce88b Duce D.A.: "Formal specification of Graphics Software". Ed. Earnshaw R.A.: "Theoretical Foundation of Computer Graphics and CAD". Springer-Verlag 1988. pp.543/574.

Duce89 Duce D.A.: "GKS, Structures and Formal Specification". Proceeding of Eurographics'89. North-Holland, 1989.

Duce89a Duce D.A.; ten Hagen P.J.W.; van Liere R.: "Componentes, Framaworks and GKS Inpunt". Proceeding of Eurographics'89. North-Holland, 1989.

Duce90 Duce D.A.; van Liere R.; ten Hagen P.J.W.: "An Approach to Hierachical Input Devices". Computer Graphics Forum. Mar. 1990. Vol.9, N.1, pp. 15-26.

Dufo88 Dufourd J.F.: "Construction of interactive programs in computer graphics". Computer Graphics Forum. Sept. 1988. Vol.7, N.3, pp.161.

Dufo89 Dufourd J.F.: "A Topological Map-Based Kernel for Polyhedron Modelers: Algebraic Specification for Logic Prototyping". Proceeding of Eurographics'89. North Holland 1989.

Earn88 Earnshaw R.A. (Ed.); Bresenham J.E.; Forrest A.R.; Lansdown R.J.; Pitteway M.L.V.: "Theoretical Foundations of Computer Graphics and CAD". Springer Verlag 1988. NATO ASI Series F. Vol.40.

Earn89 Earnshaw R.A.: "New mathematics for computer graphics". Ed. Earnshaw R.A.: "Theoretical Foundation of Computer Graphics and CAD". Springer-Verlag 1988. pp.435/448.

Ehri86 Eheich,R.W.; Williges, R.C.: "Human-Computer Dialogue design". Elseiver Science. 1986.

Fium88 Fiune E.: "The Visible Surface Problem Under Abstract Graphic Models". Ed. Earnshaw R.A.: "Theoretical Foundation of Computer Graphics and CAD". Springer-Verlag 1988. pp.575/586.

Fium89 Fiume, E.,L.: "The Mathematical Structure of Raster Graphics". Academic Press. 1989.

Fium89a Fiume E.: "Towards Realistic Formal Specification For Non-Trivial Graphical Objects". Proceeding of Eurographics'89. North-Holland,1989.

Fole88 Foley J.: "Models and Tools for the Designers of User-Computer Interfaces". Ed. Earnshaw R.A.: "Theoretical Foundation of Computer Graphics and CAD". Springer-Verlag 1988. pp.1121/1151.

Fole90 Foley J.D.; van Dam A.; Feiner S.K.; Hughes J.F.: "Computer Graphics. Theory and Practice". Addison Wesley. 1990.

Glas89 Glassner,A.S.: "An introduction to Ray Tracing". Academic Press. 1989.

Gnat83 Gnatz R.: "An algebraic approach to the standarization and the certification of Graphics Software". Computer Graphics Forum. 1983. Ed. Bono: "GKS Theory and Practice". Springer verlag. 1988.

GNAT86 Gnatz R.: "Specification of interfaces: A Caes Study of Data Exchange Languages". Ed. Encarnaçao: "Product Data Interfaces in CAD/CAM Applications. Design Implementation and Experiences". Springer Verlag 1986.

Gold89 Goldberg,A.; Robson,D. "Smalltalk-80.The Language". Reading.MA-Addison-Wesley. 1989.

Hage88 ten Hagen P.J.W.; van Liere R.: "A model for Graphical Interaction". Ed. Earnshaw R.A.: "Theoretical Foundation of Computer Graphics and CAD". Springer-Verlag 1988. pp.517/541.

Hard87 Hardwick M.; Spooner D.L.: "Comparation of some data models for engineering objets". IEEE Computer Graphics & App. Mar. 1987. pp. 56.

Hoar78 Hoare C.A.R.: "Communicating sequential Processes". Communications ACM, Vol. 21, N.8. 1978. pp.666-677.

Holt83 Holt R.C.: "Concurrent Euclid, the UNIX system, and TUNIS". Addison Wesley,1983.

Hopg86 Hopgood,F.;Duce,D.;Fielding,E.; Robinson; Williams: "Methodology of window management". Springer Verlag. 1986.

Hopg86a Hopgood,F.; Hubbold R.J.; Duce,D.: "Advances in Computer Graphics II". Springer Verlag. 1986.

Hopg86b Hopgood,F.R.A.;Duce,D.A.;Gallop,J.R.;Sutcliffe,D.C: "Introduction to the Graphical Kernel System (GKS)". Academic Press. 1986.

Hore89 Van Horebeek I.; Lewi J.: "Algebraic Specification in Software Engineering. An introduction". Springer Verlag, 1989.

Hore90 Van Horebeek I.; Lewi J.: "Are constructive formal specification less abstract?". Sigplan Notices. May 1990. Vol.25, N.5. pp. 60.

Hübn86 Hübner W.: "Validation of Graphics Systems". Ed. Encarnaçao : "Product Data Interfaces in CAD/CAM Applications. Design Implementation and Experiences". Springer Verlag 1986.

Hübn89 Hübner W.; Lancastre M.: "Towards an Object-Oriented Interaction Model for Graphics Usrer Interfaces". Computer Graphics Forum, Sep. 1989. Vol.8, N.3, pp.207-218.

Hübn89A Hübner W.: "Two Object-Oriented Models to Design Grphical User Interfaces". Proceeding of Eurographics'89. North Holland, 1989.

Jaco83 Jacob R.J.K.: "Using Formal Specifications in the Design of Human- Computer Interface". Communications ACM. Vol.26, N.4, Apr.1983. pp.259

Jone86 Jones, C.B.: "Systematic software development usign VDM". Prentice Hall 1986.

Kess77 Kessels J.L.W. "An alternative to Event Queues for Syncronization in Monitors". Communications ACM,Vol.20,N.7,1977.

Kuni85 Kunii T. L.: "Computer Graphics. Visual Technology and Art". Springer Verlag 1985.

Kuni85a Kunii T. L.: "Frontiers in Computing Graphics. Proceeding of Computer Graphics Tokyo'84". Springer Verlag 1985.

Kuni85b Kunii T.L.; Satoh T.; Yamaguchi K.: "Generation of topological boundary representations from octree encoding". IEEE Computer Graphics & App. Mar. 1985. pp.29.

Laks89 Lakshminarasimhan A.L.; Srivas M.: "A Framework for functional Specification and Transformation of Hidden Surface Elimination Algorithms". Computer Graphics Forum, Jun. 1989. Vol.8, N.2, pp.75-98.

Lamp89 Lamport L.: "A simple approach to specifying concurrent systems". Comm. ACM. Vol.32,N.1, Jan. 1989. pp.32-45.

LeMo90 Le Moigne, J.L.: "La Théorie du système genéral. Théorie de la modèlisation". P.U.F. Paris. 1990. (3ª Ed.).

Lisk86 Liskov,B.; Guttag,J. "Abstraction and Specification in Program Development". The MIT Press - McGraw-Hill. 1986.

Mac_85 Mac an Airchiningh M.: "Report on the User's Conceptual Model". Ed. Pfaff G.E.: "User interface Management Systems". Springer Verlag 1985

Thal85 Magnenat-Thalmann, N.; Thalmann, D.: "Computer annimation theory and practice". Springer. 1985.

Mall82 Mallgren, W.R.: "Formal specification of interactive graphics programming languages". MIT Press. 1982.

Mant82 Mantyla M.; Sulonen R.: "GWB: A solid modeler with Euler operators". IEEE Computer Graphics & App. Sept. 1982. pp.18.

Muus87 Muuss M.J.: "Understanding the preparartion and analysis of solid models". Ed. Roger: "Techniques for Computer Graphics". Springer Verlag, 1987. pp. 109.

Newm81 Newman W.M.; Sproull R.F.: "Principles of interactive computer Graphics". McGraw-Hill, 1981.

Onod85 Onodera T.; Kawai S.: "A formalization for the specification and systematic generation of computer graphics systems". Ed. Kunii T.L.: "Computer Graphics. Visual Tech. & Art". Springer Verlag, 1985. pp.69.

Oliv85 Oliver M.A.: "Display algorithms for quadtrees and octtress and their hardware realisation". Ed. Kessener et al.: "Data Sttructures for Raster Graphics". Springer Verlag, 1985. pp.

Otte90 Otten D.B.M.: "Report on the First Eurographics Workshop on Object-Oriented Graphics" (1990) Computer Graphics Forum. Vol.9, N.3, pp.285-286

Over85 Overveld K. van: "Applications of the method of invariants in Computer Graphics". Ed. Kessener et al.: "Data Structures for Raster Graphics". Springer Verlag, 1985. pp. 173

Pun_90 Pun T.; Blake E.: "Relationships between image Synthesis and Analysis: Towards Unification". Computer Graphics Forum, Jun. 1990. Vol.9, N.2, pp. 149-165.

Prep85 Preparata F.P.; Shamos M.I.: "Computational Geometry. An introduction". Springer-Verlag 1985.

Requ80 Requicha A.A.G.: "Representation of rigid solid: Theory, Method, and Systems". Computing Surveys. Dec.1980. Vol.12, N.4, pp.437.

Requ82 Requicha A.A.G.; Voelcker H.B.: Solid modeling: A historical summary and contemporary assesment". IEEE Computer Graphics & App. Mar. 1982. pp.9

Requ83 Requicha A.A.G.; Voelcker H.B.: "Solid modeling: Current status and research directions". IEEE Computer Graphics & App. Oct. 1983. pp.25.

Roge76 Rogers, D.F.; Adams J.A.: "Mathematical elements for Computer Graphics". McGraw-Hill. 1976.

Roge87 Rogers, D.F.; Earnshaw, R. A.: "Techniques for Computer Graphics". Springer Verlag, 1987.

Roge88 Rogers, D.F.: "Procedural elements for computer graphics". McGraw-Hill, 1988.

Salm87 Salmon, R.; Slater, M.: "Computer Graphics: Systems and Concepts". Addison Wesley, 1987.

Same85 Samel H.: "Bibliography on quadtrees and related hierachical data structures". Ed. Kessener et al.:"Data Structures for Raster Graphics" Springer Verlag, 1985. pp.181

Scho83 Schoosmith J.N.; Fulton R.E.: "Computer-Aided Geometry Modeling: A key to effective use of computers in science and enginneering". IEEE Computer Graphics & App. 1983, pp.6

Seid86 Seidewitz E.; Stark M.: "Towards a general Object-Oriented Software Development Methodology". First inter. Conf. on Ada Prog. Language Applic. for the Nasa. (Ed. Peterson G.E.: "Tutorial Object-Oriented Computing. Vol.2: Implementations". IEEE C.S. Press. 1987. pp. 16)

Teun85 Teunissen W.J.M.; Van des Bos J.: "A model for raster graphics languages primitives". Ed. Kessener et al.: "Data Structures for Raster Graphics". Springer Verlag, 1985. pp.125.

Till83 Tiller W.: "Rational B-Splines for curve and surface representation". IEEE CG&A. Vol.3, N.5. 1983. pp.61-69.

Torr91 Torres J.C.: "ALPLA. Una herramienta de prototipado a partir de especificación algebraica". Report G-91.2: Dpt. Lenguajes y Sistemas Informáticos, Universidad de Granada. 1991.

Torr91a Torres J.C.; del Sol V., Clares B.; Martín D.: "Definición algebraica de transformación geométrica". CEIG'91. Madrid. 1991.

Ureñ91 Ureña C.; Parets J.; Torres J.C.; del Sol V.: "Implementación de algoritmos de síntesis de imágenes por Ray Tracing usando programación orientada a objetos". CEIG'91. Madrid 1991.

Ureñ92 Ureña C.; Parets J.; Torres J.C.; del Sol V.: "An Object-Oriented approach to ray tracing image synthesis algorithm implementatio". Computer & Graphics. Vol.16, N.4. 1992.

Wing90 Wing J.M.: "A Specifier's Introduction to Formal Methods". Computer. Vol.23, N.9, Sept.1990. pp.8-24.

Wyrw89 Wyrwas K.M.; Hewitt W.T.: "A Survey of GKS and PHIGS Implementations". Computer Graphics Forum, Mar. 1989. Vol.8, N.1, pp.49-59.

Wiss88 Wisskirchen P.: "Object-Oriented Graphics". Ed. Ruiter: "Advances in Computer Graphics III". Springer Verlag, 1988. pp.133.