Universidad de Granada

Escuela Técnica Superior de Ingeniería Informática

Departamento de Lenguajes y Sistemas Informáticos

Alhambra: un modelo para la producción de animación bidimensional

Tesis Doctoral

Domingo Martín Perandrés

Granada, 1999

Alhambra: un modelo para la producción de animación bidimensional

Memoria que presenta

Domingo Martín Perandrés

para optar al grado de Doctor en Informática

Director

Juan Carlos Torres Cantero

Departamento de Lenguajes y Sistemas Informáticos

Escuela Técnica Superior de Ingeniería Informática

Universidad de Granada

La memoria titulada ***Alhambra*: un modelo para la producción de animación bidimensional**, que presenta D. Domingo Martín Perandrés para optar al grado de Doctor, ha sido realizada en el Departamento de Lenguajes y Sistemas Informáticos bajo la dirección del doctor D. Juan Carlos Torres Cantero.

Granada, Septiembre de 1999

El Doctorando

El Director

Fdo. Domingo Martín Perandrés

Fdo. Juan Carlos Torres Cantero

Agradecimientos

Quiero expresar mi agradecimiento a todos aquellos que me han aguantado, sufrido, soportado y apoyado todos estos año, algunos de los cuales fueron especialmente duros. Sin su ayuda y energía no creo que pudiera haber llegado hasta aquí. Gracias, muy especialemnte, a Maribel por TODO (espero poder compensarte de alguna manera). Gracias Juan Carlos (tu confianza y guía han sido imprescindibles para salir adelante). Gracias Papá, Mamá, Rosi, Maria, Ernesto, Abuelas, Carlos, Manolo. Gracias Rafa. Gracias Lola. Gracias a Daniela Tost y Pere Brunet por su ánimo. Gracias a todos mis amigos y a todos mis compañeros.

Gracias a los miembros del tribunal, y todos aquellos que sean capaces de llegar hasta el final de esta memoria. Espero que este trabajo les pueda servir para algo.

Gracias a todos que con sus ideas y correcciones han ayudado a hacer que este documento sea algo coherente.

Gracias a los que han desarrollado el software gratuito (Linux, TEX, LATEX, etc) y han tenido la generosidad de donarlo para uso por los demás, entre los que me incluyo.

a Maribel y mis padres

Índice general

Índice de figuras

Prólogo

Desde el comienzo de los gráficos por ordenador ha habido un gran interés por producir imágenes altamente realistas. El objetivo principal ha sido el de reproducir fielmente, desde un punto de vista físico-matemático, el comportamiento de los distintos elementos que influyen en la producción de una imagen. Por una parte, la física de la luz al interactuar con los objetos, lo que ha llevado al desarrollo de métodos como el trazado de rayos y la radiosidad. Pero también se ha buscado una apariencia realista en el movimiento, con el estudio y aplicación de la cinemática y la dinámica. En la actualidad se están estudiando métodos y técnicas cuyo objetivo es poder obtener seres virtuales, con capacidad de sentir su entorno y de reacionar frente a los sucesos que se desarrollen.

Frente a esta línea, que podriamos llamar realista, también en los últimos años, está apareciendo otra, cuyos objetivos son distintos. Frente a modelos matemáticos y físicos muy desarrollados y ajustados a la realidad, se proponen métodos simples y de resultados, visualmente hablando, muy similares. Frente a la imagen similar a la que se obtendría con una máquina de fotos, imágenes que pretenden trasmitir información visual de forma cualitativa, no cuantitativa, expresando, no representando, mostrando que es, y no como es.

Dentro de esta última línea se encuentra el dibujo animado clásico. Considerado por muchos como un arte, necesita para la producción de largometrajes de grandes equipos humanos y de gran cantidad de recursos económicos. A pesar de ello, es una industría con una gran capacidad, indicado por el éxito de sus productos.

Desde su aparición, el ordenador se ha ido convirtiendo en una herramienta que ha permitido la mejora en la producción de muchos tipos de procesos. En en caso de la animación clásica, ésto no ha sido así.

Con estos planteamientos básicos, se definen los objetivos finales de esta memoria: obtener un sistema que permita producir animación lo más parecida posible a la animación clásica, en todos sus aspectos (visual, expresivos, etc). La solución que se aporta es el modelo *Alhambra*.

En el primer capítulo, se hace una presentación histórica de la animación. A continuación se hace una clasificación de la animación, que permite describir las capacidades y limitaciones de cada tipo. Dado que el objetivo de esta tesis es la animación clásica realizada por el ordenador, se hace una descripción de los elementos, visuales y de comportamiento, que definen la animación 2D. Con

dichos elementos se hace el planteamiento de las necesidades del sistema ideal, al cual se deberá aproximar toda posible solución. Por último, se comentan los trabajos relacionados con el tema.

En el capítulo segundo se hace una exposición del modelo propuesto para alcanzar los objetivos descritos. Se describen formalmente los distintos componentes del modelo *Alhambra*, distinguiendo entre los que tienen como objetivo producir la visualización y aquellos que se encargan de la animación.

El capítulo tercero se dedica a la exposición de como se implementa el modelo *Alhambra*, cuando éste es aplicado a modelos poligonales 2-variedad basados en triángulos, las ventajas y limitaciones que aporta. También se hace una descripción de las capacidades de animación, mediante el uso de funciones.

En el capítulo cuarto se exponen los resultados, conclusiones y futuros trabajos.

El apéndice A contiene una tabla que relaciona los distintos términos en castellano y los correspondientes en inglés [1].

[1]No me considero un ortodoxo de la "pureza del idioma", pero tampoco entiendo la necesidad de usar terminos, que en algunos casos tienen traducción y palabras que los designan, o variar la gramática a nuestro antojo (píxel y píxeles, no pixels ??). Creo que se debe buscar una solución intermedia entre los extremos de la traducción literal y del acatamiento de todos los barbarismos, obtenida con el consenso de la mayoría; o por su uso común, pero en tal caso, debemos poder dar ese primer paso que permita exponer nuevos terminos y sus palabras asociadas.

Capítulo 1

Introducción

Animación significa "dotar de vida a algo inanimado"[1]. En el contexto de los gráficos, se puede entender como la creación de la ilusión de que las cosas están vivas, de que cambian. Esta definición, no solo incluye lo que normalmente se ha entendido por animación en el pasado, ésto es, la ilusión de movimiento al crear una serie de imágenes y proyectarlas a cierta velocidad, sino que también incluye otras posibilidades de cambio, como son las metamorfosis [64] o cambios de forma [4] (Figuras 1.1, 1.2 (*Morphing*)), y los cambios de color o intensidad de la luz (Figura 1.3).

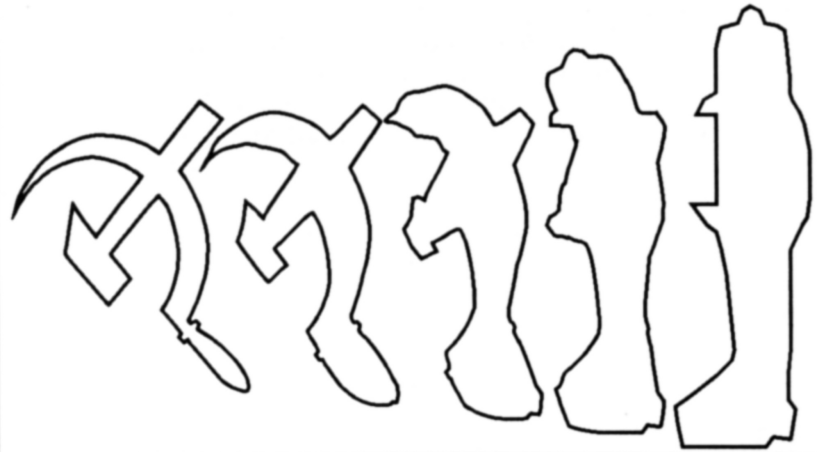


Figura 1.1: Ejemplo de metamorfosis [64].

La ilusión de cambio suave, sin cortes ni brusquedades, se debe a la existencia de un fenómeno fisiológico del sistema visual humano, llamado *persistencia de la visión*. Cuando una serie de imágenes son presentadas en sucesión rápida, el sistema visual mezcla las imágenes. Si las sucesivas imágenes varían de una a otra ligeramente, el efecto final que se produce es la percepción de un continuo, de un cambio suave.

La técnica de la animación se usa también con imágenes creadas a partir

[1] Animación proviene de la palabra latina *animus*, que significa dar vida a algo, la cual a su vez está relacionada con la palabra griega *anemus*, que significa aliento [36].

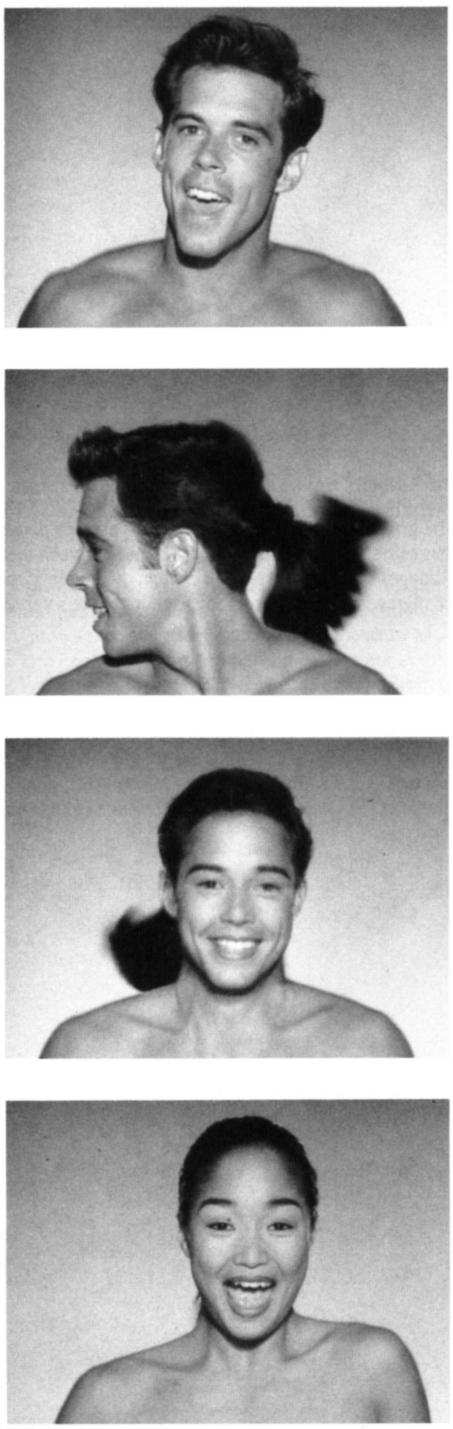


Figura 1.2: Ejemplo de metamorfosis [4].

de modelos de plastilina, marionetas y formas recortadas en papel.

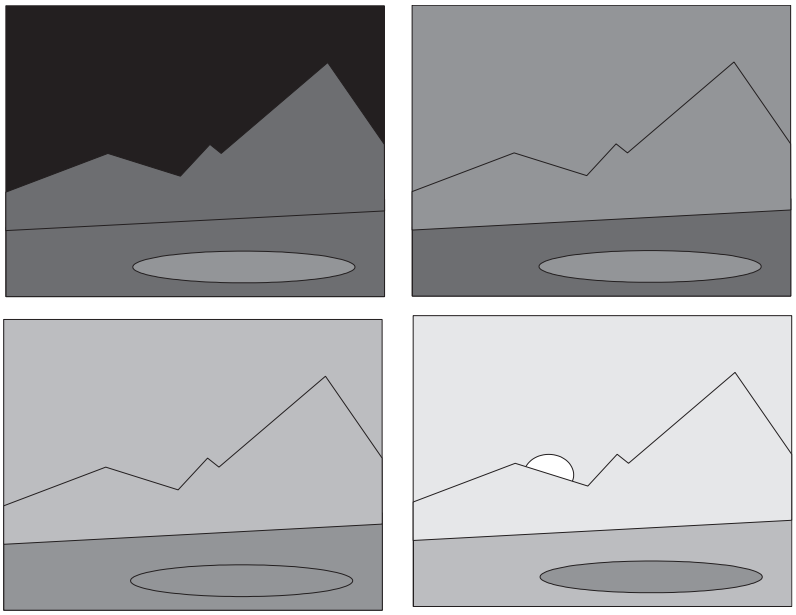


Figura 1.3: Ejemplo de cambio de intensidad de la iluminación.

1.1. Historia

La historia de la animación es relativamente reciente, comenzando los primeros estudios en 1824, año en el que Peter Roget presenta el trabajo *"The persistence of visión with regard to moving objects"* en la British Royal Society. En 1831, Joseph Antoine Plateau y el Dr. Simon Ritter construyen una máquina llamada phenakistoscopio, con la que se produce la ilusión del movimiento. La máquina posee dos tambores, el exterior tiene unas pequeñas ventanas; el interior contiene las imágenes. Cuando se hacían rotar a la velocidad adecuada, se producía la ilusión. Tres años más tarde, Horner desarrolla el zoetrope a partir del phenakistoscopio de Plateau. En 1872, después de un largo paréntesis, Eadweard Muybridge comienza su recopilación de imágenes de animales en movimiento.

Durante los años 1887 a 1889, Thomas Edison se dedica a investigar sobre las imágenes en movimiento, y crea su kinetoscopio, el cual proyecta 13 segundos de película. En 1889, George Eastman comienza la comercialización de película para fotografías, cuya banda está formada por nitro-celulosa. En 1895, se produce un evento muy importante: Louis y Auguste Lumiere solicitan una patente para un dispositivo, llamado cinematógrafo, capaz de proyectar imágenes en movimiento. Un año más tarde, Thomas Armat diseña el vitascopio, el cual será usado como referencia para los siguientes proyectores. A principios del presente siglo, en 1906, J. Stuart Blackton realiza la primera animación, llamada *"Humorous phases of funny faces"*. A partir de este momento, se produce una proliferación de animaciones. Así, en 1908, Emile Cohl produce una película basada en figuras blancas sobre fondo negro, y Winsor McCay produce una secuencia de animación basada en su personaje *"Little Nemo"*. Winsor McCay también produce, en 1906, una animación titulada *"Gertie the Trai-*

ned Dinosaur" formada por 10000 dibujos, que puede considerarse la primera película.

En 1913, Pat Sullivan crea una serie de animación llamada "Félix el gato" (*"Felix the cat"*). En 1915, Earl Hurd crea la animación basada en acetatos (*Cel animation*), técnica en la que se ha basado la animación a partir de ese momento. Cuatro años más tarde, el International Feature Syndicate produce varios títulos: *"Silk Hat Harry"*, *"Bringing up father"* y *"Krazy Kat"*. En 1923, Walt Disney extiende las técnicas de Max Fleischer para combinar acción real con personajes de animación en la película "Alicia en el país de la maravillas" (*"Alice's wonderland"*). Lotte Reiniger produce el primer largometraje de animación titulado *"Prince Achmed"* en 1926. Al año siguiente, la Warner Brothers proyecta "El Cantante de jazz"(*"The Jazz Singer"*), primera película que incorpora sonido. Es 1928, Walt Disney produce la primera película de animación con sonido sincronizado, llamada "El ratón Mickey" (*"Mickey Mouse"*). A partir de estos momentos, y durante una década, se van perfeccionando las técnicas de animación que se seguirán usando hasta nuestros días.

En 1943, John y James Whitney producen *"Five Abstract Film Exercises"*, y en 1945, John Witney crea los gráficos por ordenador analógico. El mismo crea, en 1961, secuencias de títulos para televisión con un mecanismo diferencial. Durante el periodo que va de 1963 a 1967 se produjeron una docena de películas en la Bell Telephone Laboratories. En 1963, E. Zajac crea la primera animación por ordenador: *"Two-gyro gravity gradient attitude control system"*. Un año después, Ken Knowlton, comienza a desarrollar técnicas de ordenador para producir películas de animación. F. Sinden hace *"Force, Mass and Motion"*, una película que demuestra las leyes del movimiento de Newton, usando el lenguaje FORTRAN. Huggins y Weiner hacen la película *"Harmonic phasors"*, la cual muestra la composición de complicadas formas de onda periódicas. McCumber crea una película que muestra el efecto de la oscilación Gunn, la cual se produce en los semiconductores. Julesz y Bosche experimentan en la visión humana y en la percepción. Noll crea películas con estereovisión. En 1964 Ken Knowlton crea el lenguaje BEFLIX en un IBM 7094. Este lenguaje manipula directamente una matriz de 252x184 píxeles con 3 bits, lo cual le permite representar ocho niveles de gris. Otros sistemas semejantes son EXPLOR de K. Knowlton y GENESYS, desarrollado por Ronald Baecker [1969] en el MIT para su tesis. El sistema creado por Computer Image Corporation, SCANIMATE, es un sistema de animación analógico que permite al animador modificar las señales producidas por el sintetizador de video. Con dicho sistema se crean animaciones para las películas "2001: Una Odisea en el Espacio" (*"2001: A Space Odyssey"*) y "Submarino Amarillo" (*"Yellow Submarine"*). CAESAR extiende las capacidades de SCANIMATE permitiendo el control de las partes de personajes de animación, así como la producción del intercalado. Otro trabajo importante es el desarrollado por Burtnyk y Wein en el Consejo Nacional de Investigación de Canadá. Se puede considerar pionero en la animación por imágenes clave, desarrollando los principios del intercalado por ordenador en 1971. Hay que destacar los siguientes programas, desarrollados como ayuda en la producción de animación convencional:

1. TWEEN, creado por Catmull en 1979, es un programa que proporciona los medios para generar y manipular las imágenes de personajes. Su principal objetivo es la producción de intercalado.

2. PAINT, desarrollado por Smith en 1978, permite colorear usando un lápiz y una tableta.

3. SOFTCEL, creado por Stern, es un sistema que usa las memorias de imagen (*Frame buffer*) para reemplazar las operaciones de copiar manualmente los dibujos a los acetatos y luego colorearlos.

Lucasfilm, como casa comercial productora de animación, crea los efectos de las películas *" Star Trek II"* y "El retorno del Jedi" usando la técnica de los sistemas de partículas desarrollada por Reeves. Edwin Catmull y Alvy Ray Smith, tras trabajar en el Instituto Tecnológico de Nueva York, junto con otros miembros de Lucasfilm, crean Pixar en 1986. John Lasseter produce las siguientes películas de Pixar: *"Luxo Jr."* (1986), *"Red Dreams"* (1987), *"Tin Toy"* y *"Knickknack"* (1989).

La película *"Tron"*de Walt Disney es producida en 1982 por MAGI, Robert Abel, Information International Inc. y Digital effects.

Una revisión de las películas de animación generadas por ordenador desde 1961 hasta 1989 se puede encontrar en [44].

1.2. Clasificación de la animación

Para el estudio de la animación, basada en dibujos o imágenes, se van a presentar varios esquemas de clasificación. Esta clasificación es necesaria para entender cual es el área sobre el que trata la presente tesis. La clasificación se realiza en función de distintos elementos. Un primer elemento clasificador es el agente, persona o máquina, que produce la animación; otro elemento es el medio o herramienta que se use, que podrá ser manualmente o automáticamente. Una tercera clasificación tiene como elemento discriminador el que se usen imágenes o parámetros para la generación de la animación; y una última clasificación está relacionada con la apariencia visual. A continuación se exponen las distintas posibilidades con más detalle.

Los dos primeros criterios suelen agruparse, ya que productor y herramienta pueden no ser independientes. La primera clasificación se basa en estos dos criterios: quién produce la animación y con qué herramienta se crea. El productor de la animación puede ser una persona o un ordenador. En el caso de que el productor sea una persona, ésta puede o no utilizar un ordenador como medio para la producción de la misma.

La principal clave de esta división es el productor de la animación. Si el productor es un dibujante hablaremos de animación convencional o clásica. Cada dibujo es realizado manualmente, lo cual es laborioso de producir e involucra a grandes equipos en las películas de animación.

Figura 1.4: Clasificación de la animación en base al generador y la herramienta.

Si el productor es un ordenador, hablaremos de animación por ordenador o modelada. Se crean, mediante el uso de un ordenador, modelos de todos los componentes de la escena (actores, fondos, iluminación, cámaras, etc), y se producen los fotogramas finales.

Otra clasificación se obtiene en función de que la realización de la animación se produzca manualmente o por ordenador. Para el caso de usar el ordenador como herramienta de ayuda en la animación convencional, se hablará de animación asistida por ordenador. En la Figura 1.4 se puede observar los criterios comentados y como se relacionan.

El tercer criterio, comentado anteriormente, se basa en la manera de producir las imágenes: si se basa en los propios dibujos, hablaremos de animación por imágenes clave; si se usan parámetros, se llamará procedural o paramétrica.

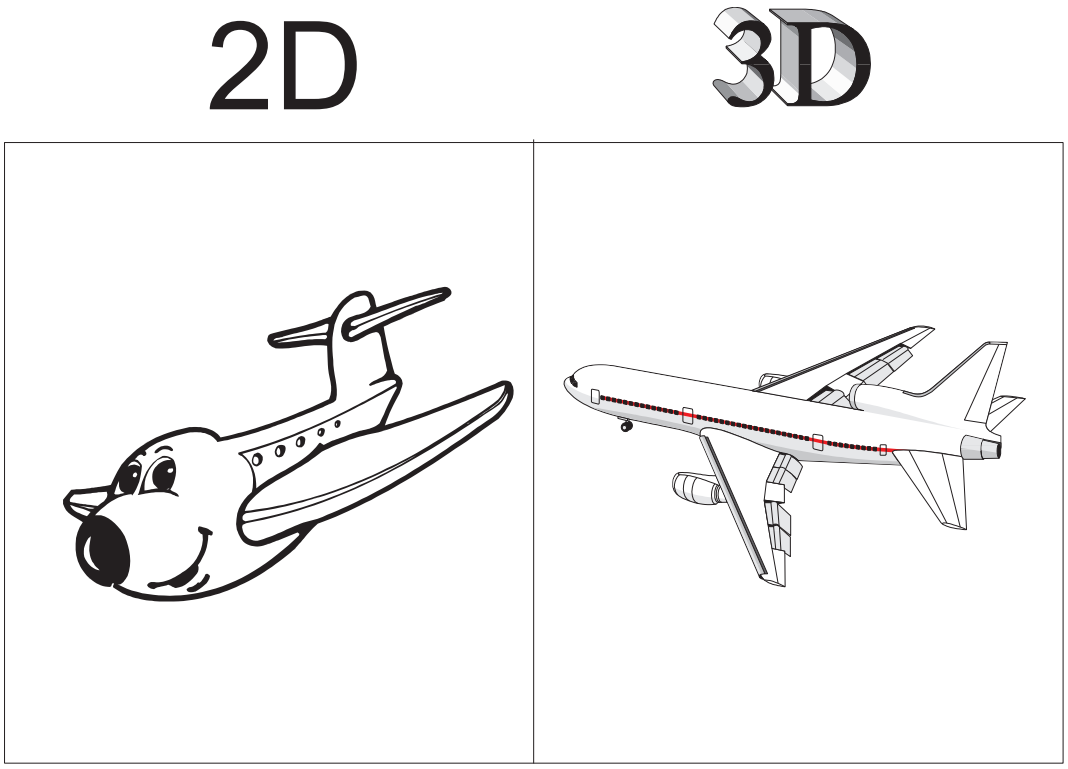


Figura 1.5: Clasificación de la animación en base a su aspecto.

Ambos conceptos serán expuestos más detalladamente en la página 19.

Otro concepto es el de la apariencia visual (Figura 1.5). En función de este criterio, tenemos la animación de aspecto bidimensional y la de aspecto tridimensional. En general, la animación convencional, asistida o no, está orientada hacia la producción de animación en la cual los personajes tienen coloreado plano (ésto está cambiando en las últimas películas animación) y líneas de forma. En cambio, el fin de la animación por ordenador suele ser la producción de animación con apariencia tridimensional.

A continuación se exponen las características principales de cada tipo de animación.

1.3. Animación convencional

En la animación convencional, el dibujante posee un modelo mental de lo que quiere representar mediante dibujos. Cada uno de los dibujos es creado partiendo de las ideas y emociones, de la forma de interpretar el personaje, de las vivencias y habilidades que el dibujante posee. Ésto permite que los personajes tengan formas y conductas extrañas, las reglas de la perspectiva sean flexibles y se puedan manejar las leyes de la física. Esta flexibilidad se plasma en una serie de trazos en el papel, el dibujo. Para producir una animación son necesarios, al menos, 24 fotogramas por segundo. Ésto da una idea de la magnitud del esfuerzo necesario para crear un largometraje.

1.3.1. Proceso de producción de la animación convencional

El proceso de producción de una película de dibujos animados suele ser largo y costoso. Suelen haber equipos con un gran número de componentes entre los que se incluyen: un director, un productor, animadores y asistentes de animación, intercaladores, coloreadores, etc. La producción conlleva una serie de procesos que se describen a continuación [74].

- Guión (*Script*)
  El primer paso es la creación de un guión, el cual, como para otro tipo de producciones, es de mucha importancia. La calidad de la película, no en su parte técnica, depende, en gran manera, del guión. Un buen guión puede dar como resultado una buena película, aunque no está garantizado, pero es muy difícil obtener un buen resultado con un mal guión. Los guiones para animación difieren de los de acción real en que los diálogos no son tan importantes. De hecho, se deben evitar los diálogos complejos. Es la acción visual, en forma y tiempo, la que debe transmitir el mensaje, a modo de mímica.

- Esquema de la historia (*Storyboard*)
  A partir del guión el director crea un esquema genérico de la historia, una serie de dibujos que muestran la acción descrita en el guión. En base a este esquema, se pueden detectar errores e inconsistencias en el guión que deben ser corregidos por el director. Es una especie de resumen gráfico de la historia.

- Banda sonora (*Sound Track*)
  Una vez se han terminado el guión y el esquema de la historia, se graban la banda sonora, los diálogos y otros efectos sonoros. Ésto es así porque en animación debe haber una perfecta sincronización entre imágenes y sonido. Para la animación es más fácil sincronizar las imágenes con el sonido que al revés. Sin la banda sonora y los diálogos, el dibujante no puede temporizar correctamente la acción. La forma de hacerlo es grabando la música de forma provisional, con pocos músicos, indicando lo esencial de la música en cuanto a melodías y ritmo. A esta banda se le añaden marcas sonoras que sirven para llevar a cabo el sincronismo.

- Descomposición de la banda sonora (*Track Breakdown*)
  Esta tarea consiste en analizar los diálogos fonéticamente, por sonidos en vez de deletreando, y documentar la posición de cada sonido con respecto a las imágenes de la película. Por ejemplo, se puede indicar que en la imagen 14 se debe comenzar a pronunciar un sonido, y que el mismo durará 10 imágenes. Esta descomposición es llevada a una hoja tabulada (*Bar sheet*). En esta hoja se indican exactamente la posición de las imágenes y sonidos, permitiendo un fácil análisis e identificación visual por parte del animador.

- Diseño (*Design*)
  Mientras se realiza la descomposición del sonido, los diseñadores producen una interpretación visual de los personajes de la película. Cuando

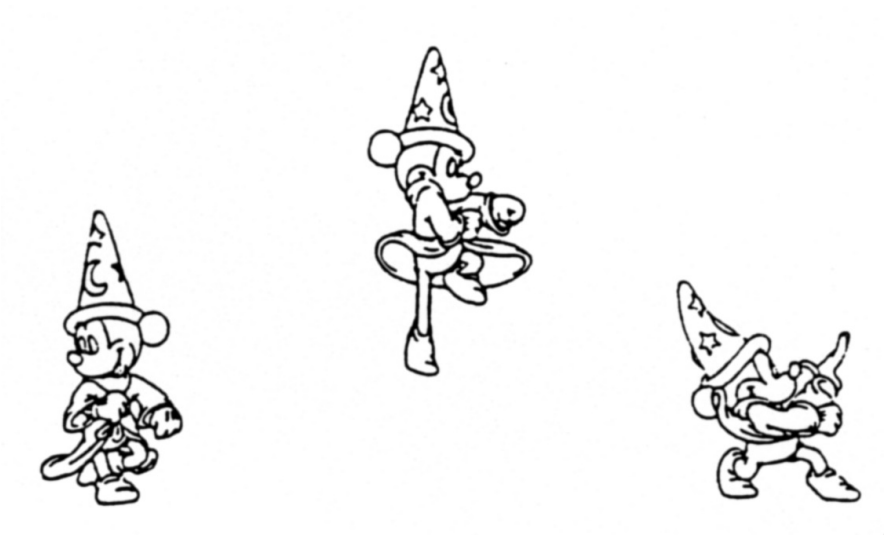


Figura 1.6: Imágenes clave.

estas interpretaciones son aceptadas, se crean diferentes vistas desde diferentes ángulos de cada personaje, colocadas en una sola hoja, llamada hoja del modelo (*Model sheet*), que es usada como referencia por todos los animadores. En este momento, idealmente, se puede estar creando también el estilo de los fondos.

- Bobina Leica (*Leica Reel*)
  La bobina Leica se crea a partir de las hojas tabuladas y el esquema de la historia. La bobina Leica es, en esencia, un esquema filmado de la historia, que se puede proyectar sincronizada con la banda sonora final. Ciertos dibujantes se encargan de mostrar, con los dibujos del esquema de la historia de forma exacta, como debe ser el resultado final, incluyendo

Figura 1.7: Intercalado entre imágenes clave.

el estilo que deben poseer las imágenes, describiendo la acción con más de un dibujo. Cuando todas las escenas se realizan de esta manera, se fotografían, teniendo en cuenta la temporización de las hojas tabuladas. La proyección de la bobina Leica permite al director hacerse una idea general del aspecto de la película. En este momento se pueden hacer cambios en el contenido visual de la película sin que su coste sea demasiado alto. De hecho, suele ser la última etapa en la producción de una película de animación en la que se pueden hacer cambios.

- Prueba de línea (*Line Tests*)
  Una vez la bobina Leica es aceptada por director y productores, es cuando empiezan a actuar los animadores y los intercaladores. El animador, interpretando al personaje, dibuja las imágenes clave (*Keyframe*) (Figura 1.6). Esta fase es la más importante, ya que la expresividad del dibujo depende de la capacidad y habilidad del animador para expresarla. Una vez que el animador ha creado las imágenes clave, deben obtenerse los dibujos intermedios. Esta tarea es lo que se llama intercalar (*Inbetween*). Siguiendo una serie de especificaciones el intercalador crea los dibujos intermedios (Figura 1.7). Una vez se han creado los dibujos, se realiza unas pruebas de línea. La prueba de línea consiste en filmar los dibujos en los momentos precisos indicados por la hoja tabulada. Algunas veces es necesario modificar la animación varias veces en una escena particular si la prueba de línea muestra que la acción no está funcionando. Generalmente, la prueba de línea funciona a la primera, y la escena puede ser cortada y añadida a la bobina Leica sustituyendo los dibujos descriptivos de dicha escena. Gradualmente se van añadiendo todas las escenas y se puede hacer una visualización de la película completa, lo cual permite realizar ajustes finales.

- Ajustado (*Clean Up*)
  En las grandes producciones hay varios animadores trabajando sobre un mismo personaje. Es muy importante que todos los dibujos de un mismo personaje tengan un estilo visual consistente. De esta tarea se encargan los ajustadores. Una vez se realiza esta tarea es conveniente realizar una prueba de línea para comprobar que no aparecen otros errores.

- Trazado y coloreado (*Trace and Paint*)
  Cuando se tienen los dibujos probados y ajustados, son pasados a acetatos. Al principio de la animación ésto era una tarea costosa realizada manualmente. Actualmente este proceso o bien se realiza con una fotocopiadora o bien los dibujantes trabajan directamente sobre los acetatos. Una vez se tienen los dibujos en los acetatos, se colorean los dibujos por la parte de atrás usando colores opacos, permitiendo que los dibujos estén por encima de los colores y no se pierdan.

- Fondos (*Backgrounds*)
  Mientras se hacen los dibujos y se colorean, otro equipo de artistas se encargan de producir los fondos, todo aquello que no se mueve que está detrás de los personajes móviles. Aquí también se debe mantener una continuidad en el estilo.

- Comprobación (*Checking*)
  Conforme los acetatos y los fondos son terminados, son pasados a un comprobador, el cual se encarga de comprobar de que los dibujos, con sus trazos y colores están preparados para ser fotografiados.

- Filmado (*Final Shoot*)
  Es esta etapa se fotografían los fondos y los acetatos conjuntamente. Ésta es la etapa final de producción artística.

- Primeras pruebas (*Rushes*)
  La película con las fotografías es enviada a un laboratorio de revelado que debe devolverla lo más aprisa posible. En cuanto se tiene se proyecta y se observa si existe algún error. En tal caso se rectifica la escena y se vuelve a filmar. En caso contrario, la escena es cortada y sustituye a la escena correspondiente en la prueba de línea.

- Doblaje (*Dubbing*)
  Cuando se posee toda la película en un estado final y el director está satisfecho con el resultado, director y editor pasan a seleccionar los efectos sonoros que serán incluidos en la película. Cuando dichos efectos están perfectamente sincronizados con la acción, el director y el editor se encargan de mezclar las pistas de voz, las de sonido y las de efectos en una sola pista. Con ésto, la película queda descompuesta en dos partes: la banda sonora y la banda de imágenes.

- Copia maestra (*Answer Print*)
  A partir de las dos bandas, se solicita una copia maestra a los laboratorios. Ésto implica la mezcla de las dos bandas en una sola.

1.3.2. Principios de la animación convencional

Desde finales de los años 20 hasta una década después, la animación pasó de ser una novedad, a convertirse en un arte en los estudios de Walt Disney. Durante este período de tiempo, fueron apareciendo una serie de técnicas que hacían que la animación fuera más expresiva, basándose en modelos en movimiento y en el estudio de acciones en vivo. Estas técnicas se fueron aislando, analizando y perfeccionando, dando paso a una serie de reglas, que han pasado a convertirse en los principios fundamentales de la animación. A continuación se exponen dichas técnicas:

1. Aplastamiento y extensión (*Squash and Stretch*)
   Permite definir la rigidez, la masa y la velocidad de un objeto al distorsionar su forma durante la acción (Figura 1.8).

2. Temporización (*Timing*)
   Separación de las acciones para definir el peso y tamaño de los objetos y la personalidad de los personajes.

3. Anticipación (*Anticipation*)
   Consiste en la preparación de la acción (Figura 1.9).

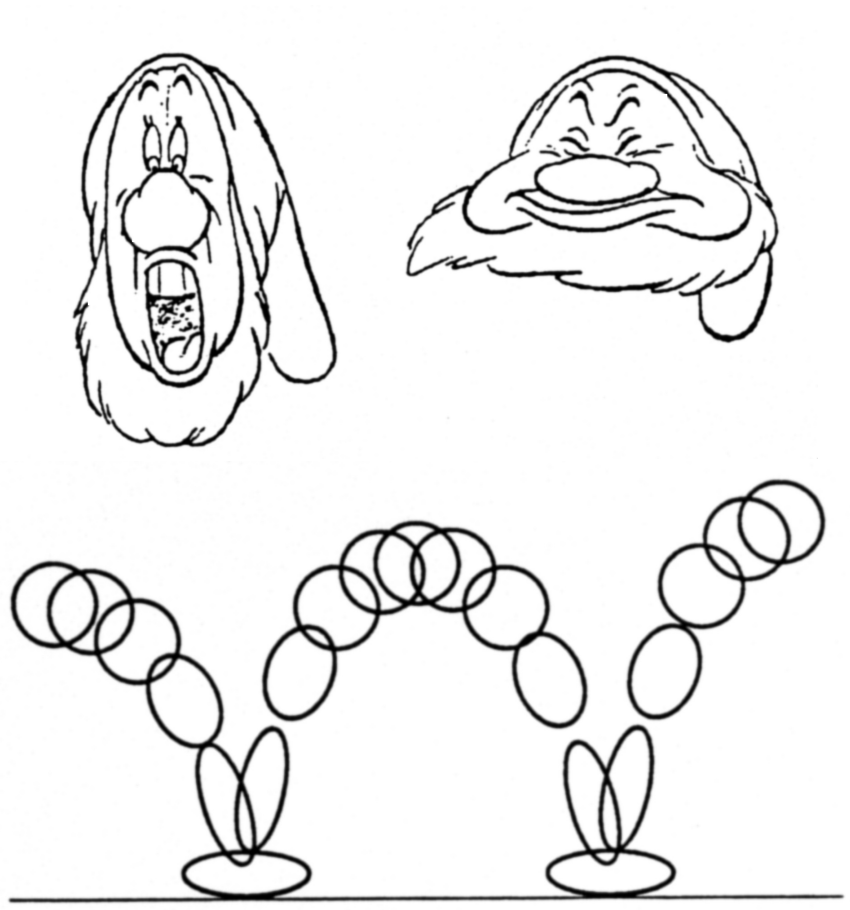


Figura 1.8: Técnica del aplastamiento y estiramiento [55].

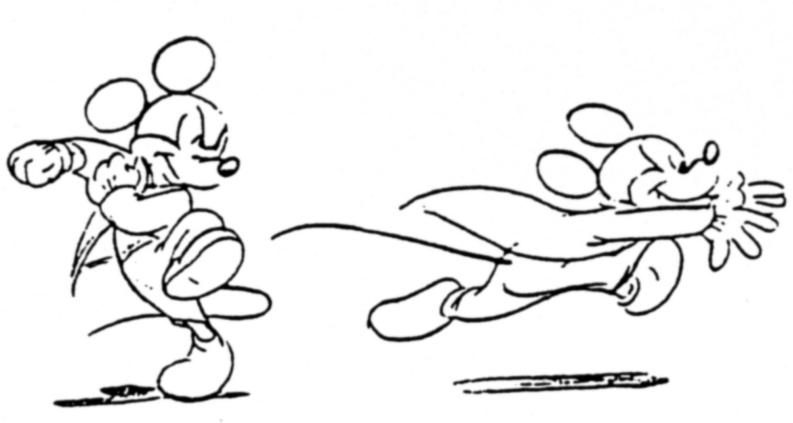


Figura 1.9: Técnica de la anticipación de la acción.

4. Escenificación (*Staging*)
   Presentar una idea de tal manera que esté inconfundiblemente clara.

5. Completar y solapar (*Follow Through and Overlapping Action*)

La terminación de una acción y el establecimiento de su relación con la siguiente.

6. Acción directa y acción postura a postura (*Straigh Ahead and Pose-To-Pose Action*)
   Las dos posibilidades de crear el movimiento para un animador. En el primer caso el dibujante trabaja directamente, de forma espontánea, hasta terminar la escena. En el segundo caso hay una preparación y estudio de los movimientos.

7. Lento al llegar lento al salir (*Slow In Slow Out*)
   Indica que el espaciado entre las imágenes tiene que ser tal que permita definir correctamente la temporización y el movimiento. Los movimientos en los extremos de una acción suelen ser lentos.

8. Arcos (*Arcs*)
   Consiste en seguir trayectorias curvas para obtener movimientos naturales.

9. Exageración (*Exaggeration*)
   Acentuación de lo principal de una idea a través del diseño y la acción.

10. Acción secundaria (*Secondary action*)
    La acción que se debe a la acción de otro objeto o personaje.

11. Atractivo (*Appeal*)
    Crear un diseño o acción que guste al que lo va a ver.

Estas técnicas, creadas para la animación convencional [71], han sido llevadas a la animación por ordenador, aunque en algunos casos su aplicación ha cambiado [38].

Para concluir, se puede decir que la animación convencional está orientada, principalmente, a la producción de personajes o imágenes de apariencia plana, y que su mayor ventaja está en la flexibilidad con que éstas pueden ser producidas. El inconveniente es que para mantener dicha flexibilidad hay que realizar todos los dibujos a mano. La animación por ordenador está orientada hacia la producción de imágenes de apariencia tridimensional, manejando entornos que son más complejos. La flexibilidad de la animación convencional se demuestra en la expresividad que se obtiene en los personajes. Ésto mismo se está intentando realizar, como se verá más adelante, con modelos tridimensionales, pero, en la actualidad, ni su uso es sencillo, ni se obtienen, en todos los casos, los resultados deseados. Pongamos un ejemplo para concluir: una película como "Blancanieves y los siete enanitos" (*"Whitesnow and the seven dwarfs"*) de Walt Disney, con las disposiciones expresivas mostradas, sería muy difícil de realizar, actualmente, mediante animación por ordenador (modelado de los personajes, movimientos, etc).

1.4. Animación asistida por ordenador

Una vez descritos los pasos en la producción de la animación convencional y las técnicas que se suelen aplicar, pasamos a ver que tareas puede desempeñar el ordenador al ser integrado, como herramienta, en la producción de animación convencional. En general, esta integración ha consistido en una asimilación de las técnicas clásicas, para ser usadas mediante el ordenador. Se han desarrollado métodos para poder colorear y tintar, para sincronizar y editar, para introducir los dibujos y para el intercalado, pero es en esta última tarea donde no se ha encontrado una solución que permita automatizar completamente el proceso. Por supuesto, la animación sigue siendo un proceso creativo del dibujante.

Siguiendo las fases de producción, el ordenador puede usarse en:

1. La creación de los dibujos
   a) Los dibujos clave pueden ser digitalizados
   b) Los dibujos clave pueden ser creados con un editor gráfico interactivo.
   c) Objetos complejos pueden crearse mediante programas.
2. La creación de movimiento
   El intercalado y movimientos complejos pueden ser generados por el ordenador (no siempre, ni con la flexibilidad necesaria).
3. El coloreado
   El coloreado se puede hacer de forma automática o con poca intervención humana.
4. El filmado
   Se pueden controlar cámaras físicas o programar, en su totalidad, cámaras virtuales.
5. La postproducción
   La edición y la sincronización pueden ser controlados por el ordenador.

Algunos sistemas desarrollados son [41, 19].

De las tareas que se han expuesto comentaremos más en detalle el intercalado (Pág. 19), ya que es la fase donde aparecen más problemas para su automatización.

1.5. Animación por ordenador

Como ya se ha comentado, en la animación por ordenador o modelada, es donde, verdaderamente, toma el ordenador un papel insustituible. En este tipo de animación no es, sólamente, una herramienta de apoyo, sino que es un elemento básico. La animación por ordenador consiste en la producción de una secuencia de imágenes, partiendo de los distintos modelos y datos que están

definidos. La gran diferencia frente a la animación clásica y a la animación asistida, consiste en que, una vez tiene los modelos y datos que controlan la animación, es el propio ordenador el que gobierna la obtención de las imágenes, en el sentido de que no hay indicación humana.

Es este sentido de automatismo, y la posibilidad de manejar grandes volúmenes de datos a gran velocidad y precisión, los que convierten al ordenador en la herramienta que permite este tipo de animación. Lo que para un hombre pueden ser horas de trabajo, el obtener una perspectiva de un modelo, el ordenador lo hace rápidamente y sin errores. Si además pensamos que, para un solo segundo de animación, se necesitan al menos 24 dibujos, podemos alcanzar a ver lo imposible de la tarea, cuando se tienen que realizar miles de dibujos. En este sentido, el ordenador no reemplaza al dibujante en una tarea que ya se realizaba, sino que es el medio que lo ha permitido.

En comparación con el proceso de producción de un dibujo por parte del animador, el cual se limita a plasmar en el papel su imagen mental, poseyendo una gran inmediatez, el proceso de animación por ordenador suele ser más largo. En primer lugar, se debe crear un modelo 3D de cada uno de los componentes de la escena. Después, los objetos deben posicionarse dentro de un sistema de coordenadas de mundo, incluyendo la cámara sintética, fuentes de luz, etc. Una vez se tienen los objetos situados, hay que animarlos, producirles cambios: en posición, forma, color, etc. Para cada cambio que se produzca, se tiene que obtener una visualización (*Rendering*) de la escena, que puede incluir efectos como sombras, transparencias, texturas, eliminación de partes ocultas, etc. Por último, cada una de estas imágenes es grabada para una posterior reproducción.

La flexibilidad de la animación por ordenador no reside en que sus modelos estén formados por polígonos o parches de superficies, en que sus transformaciones se realicen mediante matrices o cuaterniones, y en que se le aplique un modelo de iluminación, basado en la geometría y en fórmulas, para obtener las imágenes, sino que está en la posibilidad de crear mundos virtuales, con objetos y reglas virtuales que pueden ser modificadas cambiando la definición de los mismos. El ordenador es insustituible en este tipo de animación, debido a la capacidad de producir imágenes a partir de modelos tridimensionales de forma rápida y efectiva. Esta posibilidad está muy limitada en un dibujante por el tiempo que debe dedicarle para alcanzar el mismo nivel de realismo. Por ejemplo, una visión de un edificio, por dentro y por fuera, con características de materiales e iluminación; o la visualización de un flujo de gases, no se podrían realizar usando animación convencional ya que las necesidades humanas, de recursos, de precisión, etc, limitan los su realización.

Las fases para producir una animación por ordenador, una vez creada la historia, que es común para toda película, serían las siguientes:

1. Modelado
   Consiste en la representación o descripción de objetos 3D.

2. Animación
   Esta fase incluye tanto la colocación de los objetos y las cámaras en

coordenadas de mundo, como las técnicas para animarlos.

3. Visualización
   Consiste en obtener la imagen apartir de la escena.

Pasamos a continuación a hacer un breve estudio las técnicas de modelado usadas en animación.

1.5.1. Modelado

De las distintas formas de representar un objeto 3D, las más usadas en animación son las basadas en superficies. Este esquema representa los objetos por la superficie exterior de éstos y no por su contenido interior. Dentro de esta categoría hay tres posibilidades:

- Uso de polígonos
  Un modelo poligonal consiste en un conjunto caras que representan la superficie del objeto. Las caras se componen a su vez de aristas que se definen mediante vértices. La representación puede ser exacta, aunque por lo general suele ser una aproximación. El uso de polígonos para representar superficies es una técnica muy común. De hecho, se ha realizado un gran número de estudios y se han desarrollado gran cantidad de técnicas, basándose en esta representación, tanto para modelado como visualización, eliminación de partes ocultas, etc. En caso de que el objeto sea curvo, éste se aproxima poligonalmente. Para este caso se desarrollaron técnicas de suavizado (Gouraud y Phong [21, Págs. 598-599,738-739]) que permiten recuperar visualmente la curvatura.

- Uso de superficies implícitas o algebraicas
  Este tipo de superficies se definen mediante una ecuación que se iguala a 0. Los puntos de la superficie son aquellos que cumplen la condición expuesta por la ecuación. Algunas de estas superficies son las cuádricas: esferas, conos, cilindros y elipsoides. Existe un conjunto que extiende las posibilidades de las anteriores: las supercuádricas [3].

- Uso de superficies paramétricas
  Este tipo de superficies se definen mediante una ecuación que depende de dos parámetros. Este tipo de representación presenta varias ventajas frente a la descripción explícita y la implícita. Su representación es mucho más compacta que la poligonal y permite la evaluación directa de la superficie del objeto, frente al muestreo de las superficies implícitas o algebraicas. Los tipos de superficies que podemos usar son:

  - De Bézier
  - Splines cúbicos
  - B-splines y otros tipos
  - De Coons
  - De Gordons

1.5.2. Creación de los objetos

Hemos visto la forma de representar los objetos. Veamos ahora como pueden ser creados. Los métodos más usados para crear objetos son los siguientes:

- Digitalización 3D del objeto
  Mediante dispositivos como los escáneres láser, o basados en campos magnéticos o sonido, se obtiene las coordenadas 3D de ciertos puntos del objeto.
- Digitalización 2D y reconstrucción 3D
  Este tipo de proceso suele usar un conjunto de imágenes ortogonales del objeto, al cual se le han hecho ciertas marcas. Una vez que se han digitalizado las imágenes, se crean las mallas de superficies y se relacionan los puntos por parejas.
- Sistemas de modelado de objetos
  En este caso, se dispone de un programa que permite componer y manejar los objetos, usando alguno o algunos de los esquemas comentados anteriormente. Cabe destacar la Geometría Constructiva de Sólidos, que permite generar sólidos mediante operaciones booleanas regularizadas. Otro esquema muy interesante es el basado en superficies implícitas, mediante el cual se pueden obtener objetos blandos (*Soft objects*).
- Usando procedimientos
  Se crean procedimientos que permiten la representación del objeto.

1.5.3. Animación

En este apartado vamos a comentar técnicas que se usan para la animación por ordenador. Antes, se muestran algunos ejemplos de la aplicación de las técnicas clásicas a su uso con el ordenador. En la Figura 1.10 se observan ejemplos de aplastamiento y extensión, y en la Figura 1.11, uno de diseño atractivo, representado por la lámpara pequeña, que no corresponde, exáctamente, con un escalado de la grande.

En general, las figuras que se suelen usar son articuladas, compuestas de varias piezas unidas formando una jerarquía. Ésto hace que el tipo de animación sea, preferentemente, de tipo paramétrico (Pág. 20). También en la animación por ordenador se produce el intercalado. Para realizarlo se siguen dos esquemas: interpolación entre posiciones y aplicación de modelos físicos. En el caso de que se haga interpolación, los principios y métodos para la animación asistida se verán en la sección 1.6. Éstos son similares para dos y tres dimensiones.

Para alcanzar mayor realismo se usa la cinemática y la dinámica. Hay un gran interés en reproducir movimientos de forma realista, siguiendo las leyes de la física. Existen un gran número de trabajos dedicados a este tema [2, 6, 10, 32, 51, 55, 70].

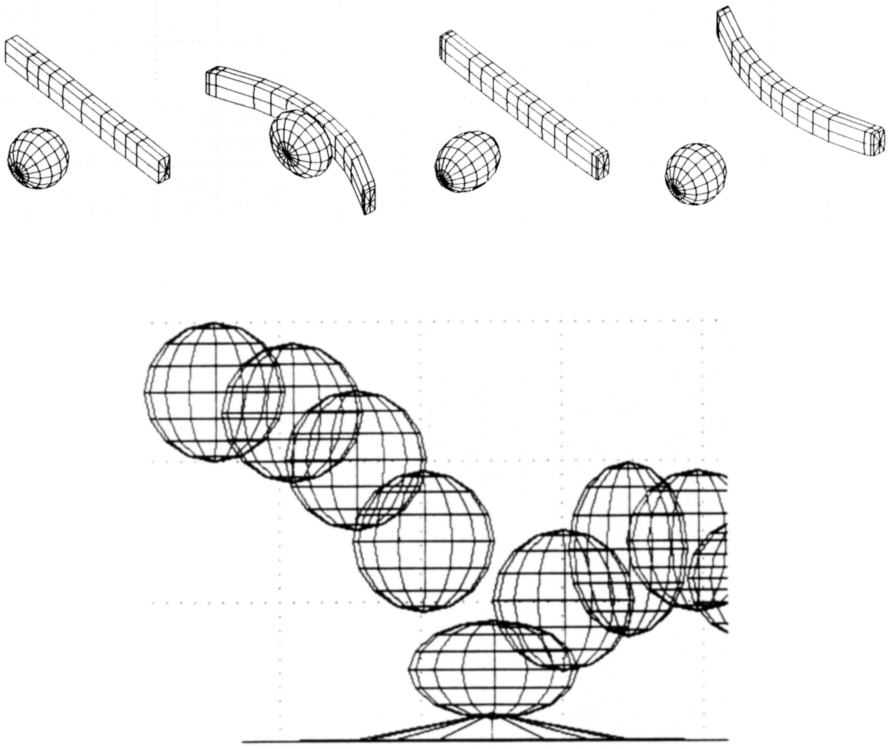


Figura 1.10: Técnica de aplastamiento y estiramiento en 3D [55].

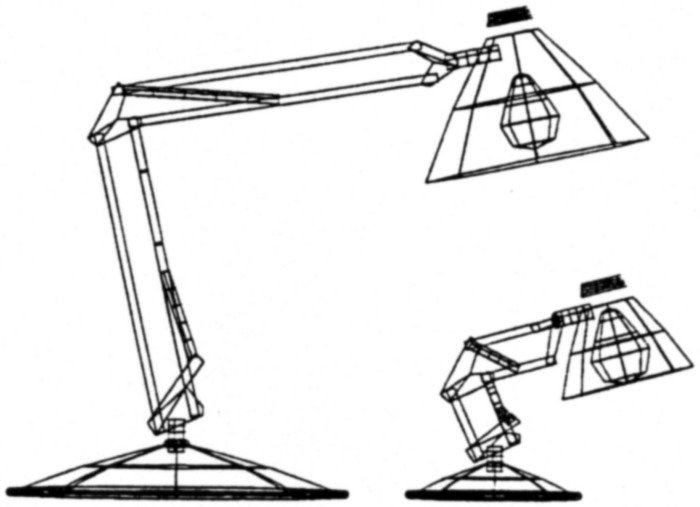


Figura 1.11: Diseño atractivo en 3D [38].

Se han desarrollado técnicas específicas para resolver problemas concretos, como pueden ser la animación de la cara [72, 33, 77] , de objetos hechos de tela [8], pelo [14, 35], gases [16, 68, 69, 73], plantas [56], etc.

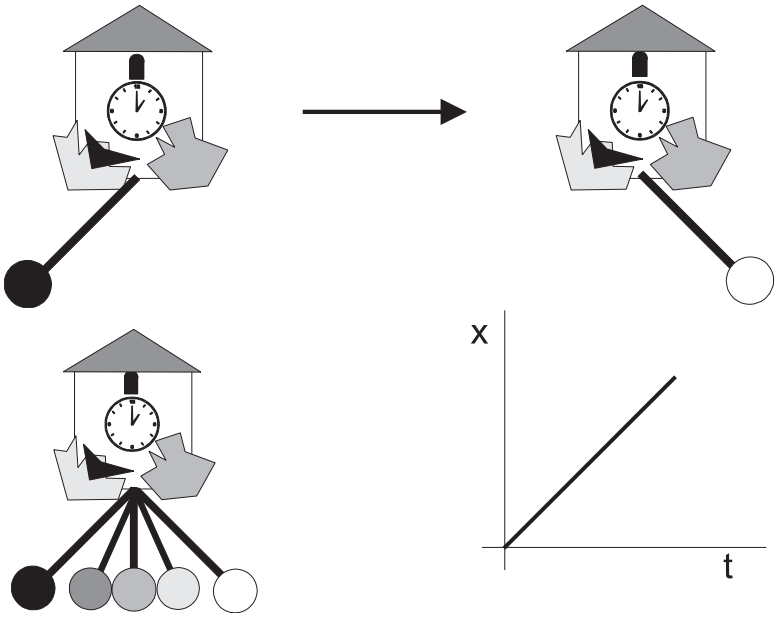


Figura 1.12: La interpolación lineal de imágenes puede producir resultados incorrectos.

1.6. El intercalado

El intercalado consiste en, dada una configuración inicial y otra final, obtener las intermedias. El significado de configuración dependerá del tipo de datos con los que se esté trabajando. Si lo que se tienen son imágenes, se hablará de interpolación por imágenes clave. También se puede hacer una interpolación mediante el uso de variables, en tal caso se hablará de intercalado por parámetros. A continuación se hace una descripción más detallada.

- Intercalado por imágenes clave
  En el intercalado por imágenes clave se parte de una imagen inicial y otra final, y se desean obtener las intermedias. Éste es el tipo de intercalado que se hace en animación clásica. En este tipo de intercalado, se relacionan las partes iguales o similares de los dibujos inicial y final, y se obtienen las posiciones intermedias. En el caso de usar un ordenador, se tienen dos conjuntos de vértices que representan a la imagen inicial y a la final, y entre ambos se realiza una interpolación posicional. Un problema que puede surgir es el de la discordancia entre las imágenes inicial y final. Entre la imagen inicial y la final existe un cierto grado de diferencia, que debe ser "rellenada" con las imágenes intermedias. Si la diferencia es muy grande, la relación entre ambas imágenes no es fácil de obtener. El intercalador humano posee información que le permite relacionar ambas imágenes. Para un ordenador, la situación es mucho más difícil, ya que, no solo debe hacer un ajuste entre el número de puntos de la imagen inicial y la final, sino que además se tiene que establecer la relación entre las partes adecuadas.

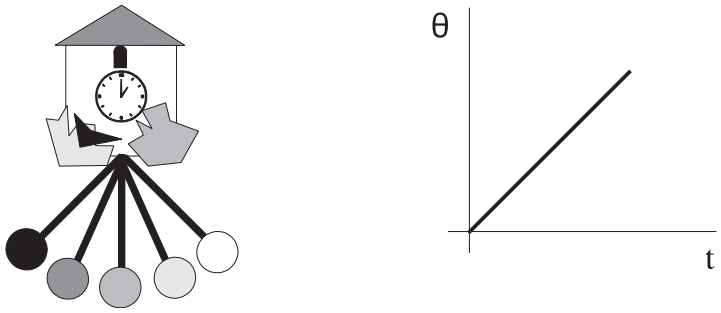


Figura 1.13: La interpolación paramétrica lineal.

Este problema se produce fácilmente en la animación clásica, ya que los dibujos son proyecciones bidimensionales de personajes tridimensionales imaginados (en general), según son visualizados mentalmente por el dibujante. En la proyección hay una perdida de información; por ejemplo cuando partes de un personaje, debido a la posición en que se encuentra, tapan otras partes [9]. El intercalador humano puede inferir la composición y forma original del personaje, lo cual es muy difícil para un ordenador si no posee otra información que no sean los dibujos clave.

Aunque haya una correspondencia exacta entre las imágenes inicial y final, es posible que la interpolación pueda producir resultados incorrectos. En la Figura 1.12 se observa el problema que puede surgir al interpolar linealmente entre dos imágenes clave.

- Intercalado de parámetros
  Este tipo de intercalado se usa principalmente en la animación asistida y la animación por ordenador. Consiste en asignarle a cada elemento de la imagen una variable, la cual controla la forma en la que se va a modificar dicho elemento. Por ejemplo, para el caso del péndulo, se le puede asociar un parámetro que es el ángulo que formaría un eje que pasara por la mitad del péndulo. Con respecto a dicho eje se pueden dar las vértices que representan la imagen del péndulo. Al variar el ángulo, se varía la posición del eje, y, por tanto, el dibujo del mismo. En este método se interpolan los valores de los parámetros. En el caso del péndulo, se interpola entre un ángulo inicial y otro final. En la Figura 1.13 se ve la

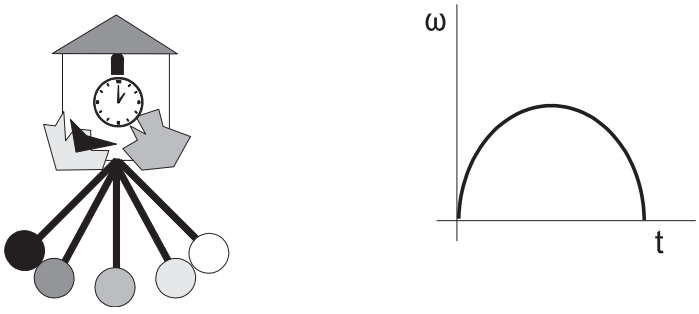


Figura 1.14: La interpolación paramétrica con ley física.

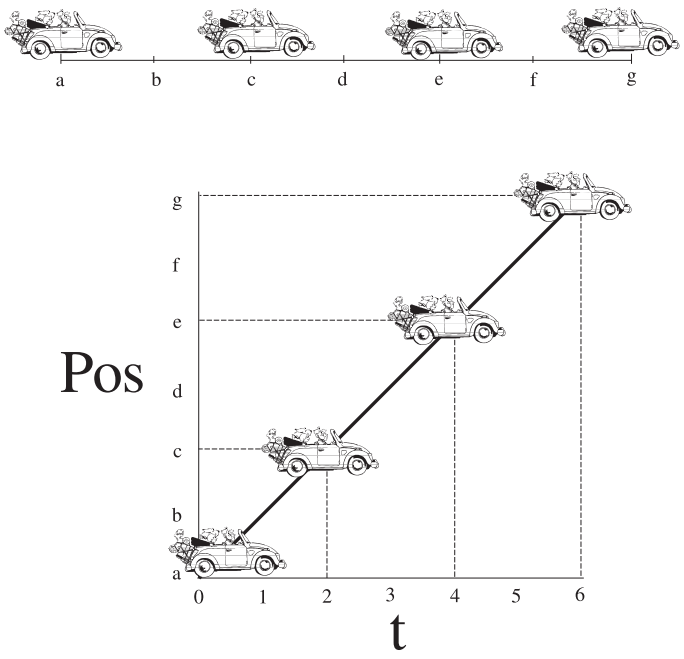


Figura 1.15: Método lineal de interpolación.

forma paramétrica del ejemplo anterior. Obsérvese que el resultado es correcto, en cuanto a la apariencia visual, pero, al ser una interpolación lineal sobre el ángulo, el resultado no se corresponde con el movimiento real del péndulo.

Este tipo de interpolación no tiene por qué ser lineal, ni incluso ser una interpolación, sino que el parámetro puede estar controlado por una ley física. Dependiendo de que la regla de interpolación sea una ley física o no, algunos autores [44] dividen este tipo en animación paramétrica por imágenes clave (sin ley física), y animación algorítmica (con ley física). En la Figura 1.14 se puede ver el resultado al aplicar un modelo físico del mismo ejemplo.

1.6.1. Métodos de interpolación

Independientemente de que se interpolen imágenes, posiciones, o parámetros, la interpolación puede ser lineal o no lineal. Veamos a continuación diferentes soluciones.

- Interpolación lineal
  El uso de una función lineal como interpolante puede causar, también, problemas de continuidad. Veamos un ejemplo. Supongamos que un coche está en la posición A en el tiempo $t = 0$ y en la posición G en el tiempo $t = 6$. Si hacemos una interpolación basada en los dibujos (animación por imágenes clave), se obtendrá el resultado de la Figura 1.15, en el cual se observa que los intervalos están igualmente espaciados. La función que

define el movimiento es lineal. De esta función se deduce que la velocidad del objeto es constante.

Supongamos ahora que el coche no se mueve de la manera vista anteriormente, sino que debe cumplir ciertos requisitos:

Tiempo	**Posición**
0	A
2	B
4	F
6	G

Si usamos una interpolación lineal por tramos, se podrán obtener las imágenes intermedias de la misma manera que en el caso anterior. Si se proyectaran estas imágenes, se comprobaría que se produce un efecto no deseado: el objeto cambia bruscamente de velocidad al pasar de un tramo a otro. Ésto es debido a que la función que determina la posición tiene discontinuidades de la primera derivada (la velocidad) en los puntos de unión (Figura 1.16).

- Interpolantes no lineales
  Para poder solucionar esta problemática, se recurre a interpolantes que no son lineales como pueden ser las curvas de Bézier, los splines cúbicos y otros tipos de curvas. Veamos el ejemplo anterior usando splines cúbicos. La curva resultante es un polinomio cúbico y, por tanto, posee primera derivada. En la Figura 1.17 se observa una posible solución al problema planteado con anterioridad. Veamos ahora algunas leyes que se usan normalmente en la animación:

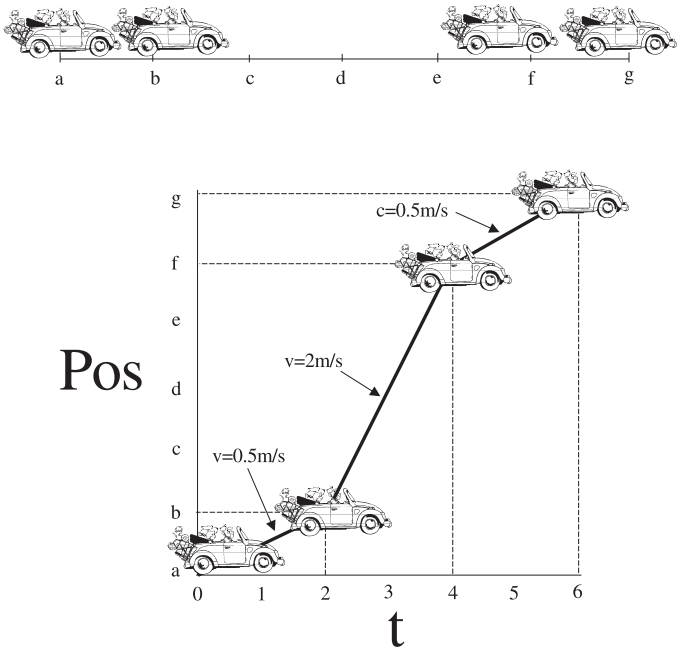


Figura 1.16: Método lineal por secciones de interpolación.

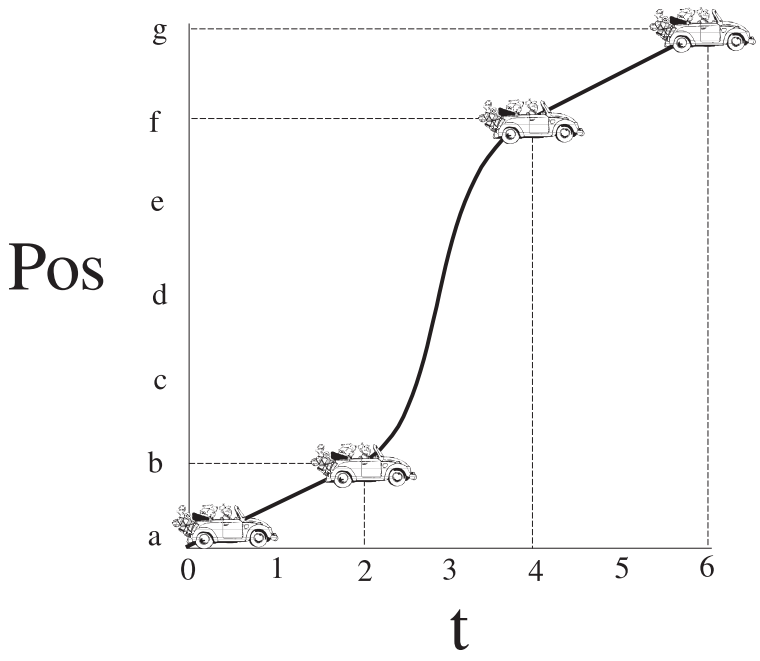


Figura 1.17: Interpolación cúbica mediante spline cúbico.

1. Ley=$K * frac$ constante
2. Ley=$(1 - (cos\theta * frac/2))$ aceleración
3. Ley=$sen(\theta * frac/2)$ desaceleración
4. Ley=$(1 - cos(\theta * frac))/2$ aceleración-desaceleración

Otra solución, consiste en programar una ley o leyes que rijan el proceso. Ésto es, a veces, muy difícil y costoso de expresar.

- P-curvas
  Otra posibilidad, es el uso de las P-curvas. Desarrolladas por R. Baecker (1969) tienen la ventaja de definir a la vez la trayectoria y su localización en el tiempo. Para ello, en la curva que define la trayectoria, se incluyen unas marcas. Estas marcas representan intervalos de tiempo iguales. La mayor o menor concentración de marcas determina la menor o mayor velocidad.

- Restricciones con puntos móviles(*Moving Points Constraints*)
  Esta técnica es parecida a la de las P-curvas. Cada curva, asociada con algunos puntos del objeto animado, varía en el espacio y en el tiempo. Al igual que la P-curvas controla la trayectoria y la dinámica usando las marcas. A cada curva que se forma se le llama punto móvil.

1.6.2. Técnica de esqueleto

Como se ha comentado, el problema del intercalado suele ser la no correspondencia entre las imágenes clave. Una solución es el uso de la técnica basada en esqueletos [7]. En vez de usar la figura completa, para calcular el

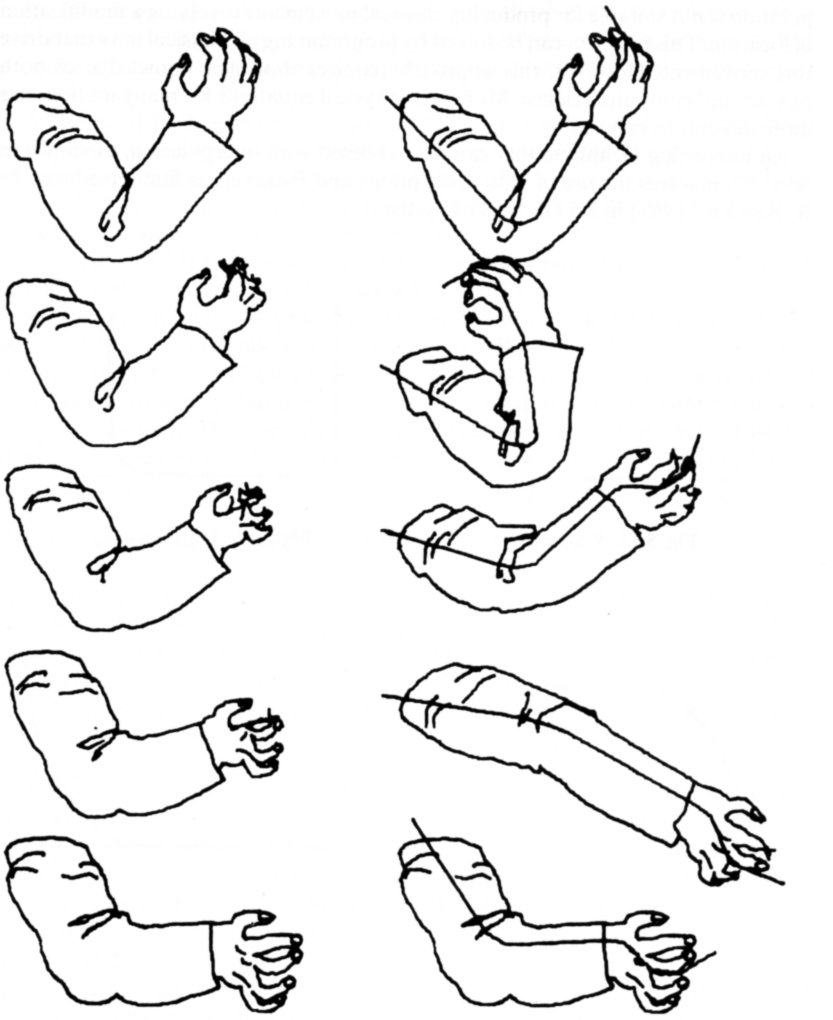


Figura 1.18: Ejemplo de deformación usando la técnica de esqueleto [7].

intercalado, se usa una representación muy esquemática, el esqueleto. Dicho esqueleto, suele ser una figura muy sencilla y, por tanto, no plantea problemas cuando se realiza la correspondencia para la interpolación. Para definir el esqueleto, se usan mallas de polígonos de cuatro lados. Para cada polígono, una vez transformado mediante la interpolación, se pueden calcular, mediante una interpolación lineal, los nuevos puntos intermedios, entre los que se encuentran los que definen el objeto. En la Figura 1.18 , se observa un ejemplo de su aplicación.

1.7. Objetivos: características del sistema ideal

Una vez expuestas las principales características tanto de la animación convencional, asistida o no por ordenador, y la animación por ordenador, se pueden destacar las ventajas e inconvenientes que aporta cada tipo:

- Animación convencional
  Ha sido durante años la animación que ha predominado. Tiene una estética muy particular. Normalmente se ha orientado a la producción de personajes, entendiendo como tales versiones caricaturizadas de personas, animales y objetos. Es producida manualmente lo que aporta una extrema flexibilidad, pero la hace muy costosa en tiempo y dinero. La entrada del ordenador ha permitido facilitar ciertas tareas y reducir costes, y en la actualidad se está intentando la total automatización de la producción de la animación, y del intercalado. En las últimas producciones se aprecia una cada vez mayor utilización de imágenes y efectos generados por ordenador dentro de las películas de animación convencional ("El Rey León", "Alexandra", "El principe de Egipto", etc).

- Animación por ordenador
  Este tipo de animación es cada vez más utilizada. Surgida con el advenimiento de los ordenadores y de la informática gráfica, se ha usado para la producción de imágenes a partir de modelos tridimensionales. Permite la creación virtual de mundos y objetos reales o imaginarios. Una vez definidos los distintos modelos y parámetros que controlan la animación, la producción de la misma es un proceso automático. El coste de una producción por ordenador puede ser también muy alto pero, en general, estos costes no son comparables con los de realizar la misma producción mediante animación convencional, ya que, o bien no es posible, o bien para alcanzar el grado de realismo deseado, el coste sería mucho mayor. Actualmente, las imágenes de síntesis se están utilizando para crear o recrear entornos y cosas, que serían muy caras de producir realmente, o que no pueden crearse, entendiendo que existe una limitación económica o física. En los últimos años están apareciendo películas en las cuales, aunque la estética es tridimensional, se utilizan personajes (*"Toy Story"*, "Bichos" (*"Bugs"*), "Hormiga Z" (*"Antz"*) creandose un tipo de animación intermedia entre la pura 3D y la convencional.

Se puede apreciar que ambos tipos de animación tienen sus puntos fuertes y débiles, y que, en general, dichas características son complementarias. Por tanto se puede hacer una exposicición de las capacidades del sistema de animación ideal:

- Capacidad de producir animación de estética 2D y 3D indistintamente.
- Gran flexibilidad en la creación y modificación de los actores, los cuales pueden ser personajes, objetos reales o inventados.
- Economía de producción mediante la automatización de la misma.

Dados los requisitos del sistema que se podría considerar ideal, se van a establecer las extensiones o añadidos que habría que incluir en los sistemas actuales, 2D y 3D, para alcanzar dichos objetivos.

No se puede partir de un sistema 2D debido a que posee menos información que uno 3D. Éste tipo de sistema no permite usar aquellos objetos y personajes

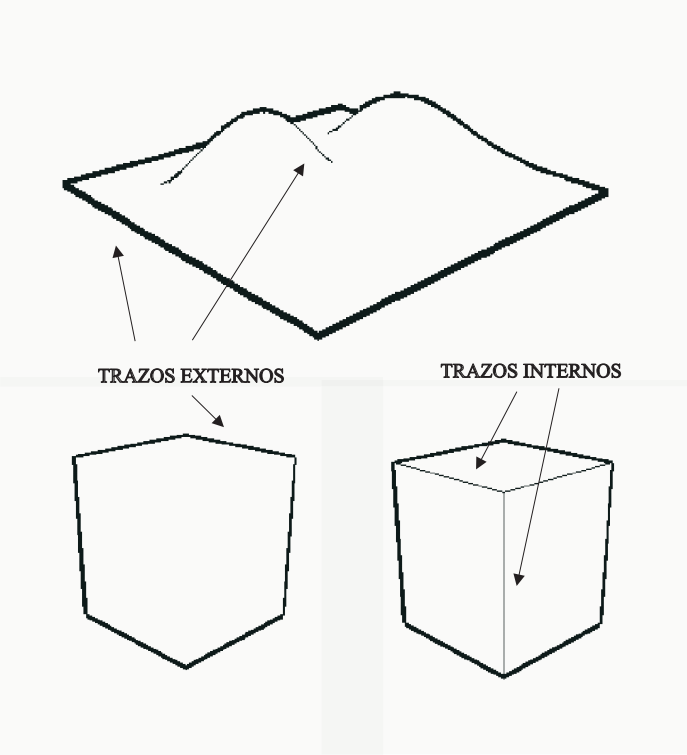


Figura 1.19: Líneas de forma: trazos externos e internos.

que tengan una constitución tridimensional, caso que suele ser el más general. Se plantea la posibilidad de añadir ciertos niveles de inteligencia artificial que permitieran un reconocimiento de los elementos representados, tal como hace un intercalador. Pero en definitiva, esto nos llevaría a la reconstrucción de un modelo 3D. Todas las operaciones que se realizan en el espacio 3D, movimientos de cámara, rotaciones, etc, no se podrían realizar. Lo mismo con operaciones de iluminación, suavizado, etc. Ésto nos lleva a concluir que nuestro sistema ideal no se puede alcanzar partiendo de una extensión de un sistema 2D.

La otra posibilidad es partir de un sistema 3D. Por definición, dicho sistema tendría resueltas todas las capacidades necesarias para producir imágenes con apariencia tridimensional, así como operar con los modelos, también 3D, en el espacio. Las rotaciones, movimientos de cámara, iluminación, etc, se realizan fácilmente en un sistema 3D. De hecho, como se ha comentado, se usan para la producción de imágenes fotorrealistas, entendiendo como tal, aquellas que intentan representar las escenas lo más real posible, ésto es, los efectos de la luz sobre los objetos de la forma más aproximada posible. Obsérvese esta distinción, representación de los efectos de la luz, y no de los contenidos, en el sentido de que las escenas que se estén sintetizando pueden representar objetos reales, pero también pueden ser objetos imaginados o con personajes tipo *"Toy Story"*. De hecho, este último ejemplo, junto con otros que se han comentado, demuestran que la animación 3D está bastante desarrollada. Los resultados que se obtienen actualmente son satisfactorios y la situación sólo puede mejorar con los avances, tanto software como hardware, que se hagan en el futuro.

Con este sistema también se podrá trabajar con animaciones puramente 2D, ya que sería un subconjunto del espacio 3D. Por tanto, la extensión que habría que incluir a este sistema sería el de poder trabajar con modelos 3D, pero que en su expresión bidimensional, tuvieran la apariencia y estética de un dibujo realizado a mano. Ésta es la capacidad que se pretende alcanzar: la *"dibujización"* de los modelos 3D.

Expongamos a continuación cuales son los elementos que permiten reconocer a un dibujo como tal, ya que los mismos deberán ser integrados en las imágenes que se obtengan de nuestro sistema ideal para alcanzar el segundo objetivo. Los elementos estéticos característicos de un dibujo son las líneas de forma, lo que llamamos trazos, y el coloreado plano. La otra componente característica es el alto grado de flexibilidad en la modificación de los modelos. Por tanto, tenemos dos problemas diferentes: uno de visualización y otro de animación. Hay que observar que éstas son características asociadas a la producción de dibujos para la animación. Cuando los dibujos se orientan hacia la ilustración, la componente estética gana un mayor peso, imponiendo en ciertos casos nuevas restricciones, como pueden ser el de mezclar distintos tipos de proyección en un sola imagen, la alta variabilidad y "carácter" de los trazos y la ausencia de color. Antes de continuar, daremos una definición informal de lo que se consideran trazos y del coloreado plano.

Vamos a entender por trazos a las líneas que definen las formas del objeto (Figura 1.19). Son los elementos mediante los cuales se expresa el objeto. Se corresponderían con las líneas y curvas que haría el dibujante para obtener el dibujo. Dentro de este conjunto de líneas, abiertas y cerradas, se pueden distinguir dos tipos: aquellas que representan el limite físico del objeto, la/s silueta/s, y el resto, que normalmente delimitan otro tipo de características. A las siluetas las llamaremos trazos externos y al resto los llamaremos trazos internos. La posibilidad de que aparezca más de una silueta, sólo se da en los objetos cóncavos (Figura 1.20). Los trazos internos, normalmente representan un cambio de color o tono, el cual está asociado a un cambio de la curvatura del objeto. Es importante determinar el por qué surgen los trazos internos. Por un lado tenemos los cambios de color o tono que, como se ha dicho, están asociados a un cambio en la curvatura del objeto, pero también a la existencia de una o varias fuentes de luz que definen dicho cambio de color. Por otro lado tenemos la posibilidad de obtener trazos internos a partir de cambios de la curvatura independientemente de la existencia o no de fuentes de luz. Esta situación es más estática que la comentada anteriormente. Como se verá más adelante, se ha implementado un mecanismo que permite obtener los trazos interiores, de la forma más flexible.

La componente relacionada con la animación es el otro elemento importante. Ya que se desea obtener imágenes lo más parecidas a la producidas manualmente, esta capacidad debe estar muy desarrollada, ya que como se ha comentado, la flexibilidad en cuanto a la forma es inherente a la producción de animación 2D. Como se verá, se han desarrollado métodos que permiten un grado de flexibilidad muy similar al alcanzado en 2D pero aplicado a modelos tridimensionales. Estos métodos se están aplicando con éxito en animaciones 3D.

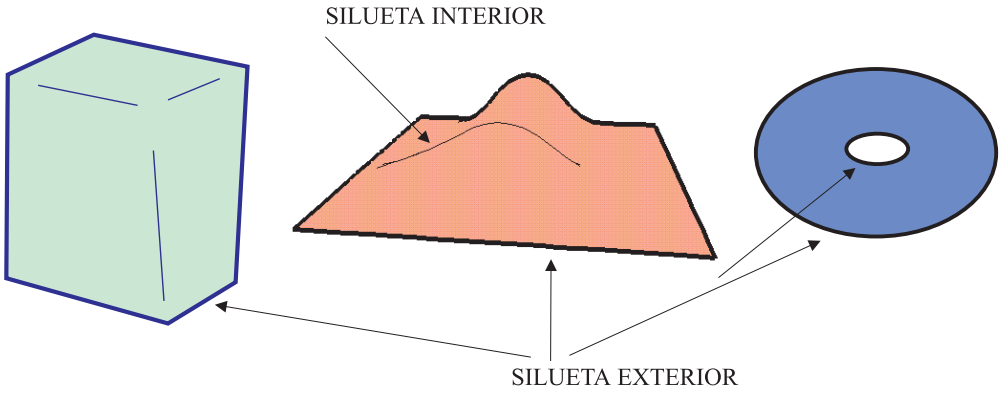


Figura 1.20: Siluetas externas e internas: trazos externos.

Por tanto, se puede concluir que el problema de la obtención de animaciones con apariencia bidimensional a partir de modelos 3D se puede descomponer en dos líneas principales, un problema de visualización y otro de animación. Estos dos problemas serán las referencias principales en el siguiente capítulo.

1.8. Trabajos anteriores

Durante los últimos años ha habido un gran interés en la visualización no fotorrealista, o como también es llamada, visualización expresiva o pictórica [11, 15, 23, 24, 26, 25, 27, 28, 30, 37, 40, 45, 46, 47, 48, 49, 60, 61, 62, 67, 75, 76, 80]. Este tipo de visualización presenta ventajas frente a la aproximación fotorrealista en diversas aplicaciones, en las cuales es más importante describir que reproducir la escena exactamente. Ejemplos de tales aplicaciones son la representación esquemática de objetos, la ilustración y la animación bidimensional.

La mayoría de estos trabajos se han orientado a la producción de imágenes estáticas, ésto es, lo que hemos denominado el problema de la visualización, y son pocos en los que se ha tratado el problema de la animación, entendiendo como tal la modificación flexible de los modelos. Siguiendo esta clasificación se pasa a hacer un repaso de los trabajos presentados en ambos campos.

1.8.1. Visualización

Hay varios trabajos, presentados durante los últimos años, que describen métodos y algoritmos que usan siluetas, líneas de contorno, trazos texturizados, etc, orientados a la producción de ilustración e imágenes de carácter expresivo. Una revisión de lo que se ha venido a llamar como visualización expresiva puede verse en el trabajo de Lansdown [37].

En esta sección realizamos una revisión de los métodos considerando que la obtención de las siluetas se realiza, bien directamente o indirectamente, a partir de un modelo 3D. Por tanto, no vamos a considerar los métodos basados

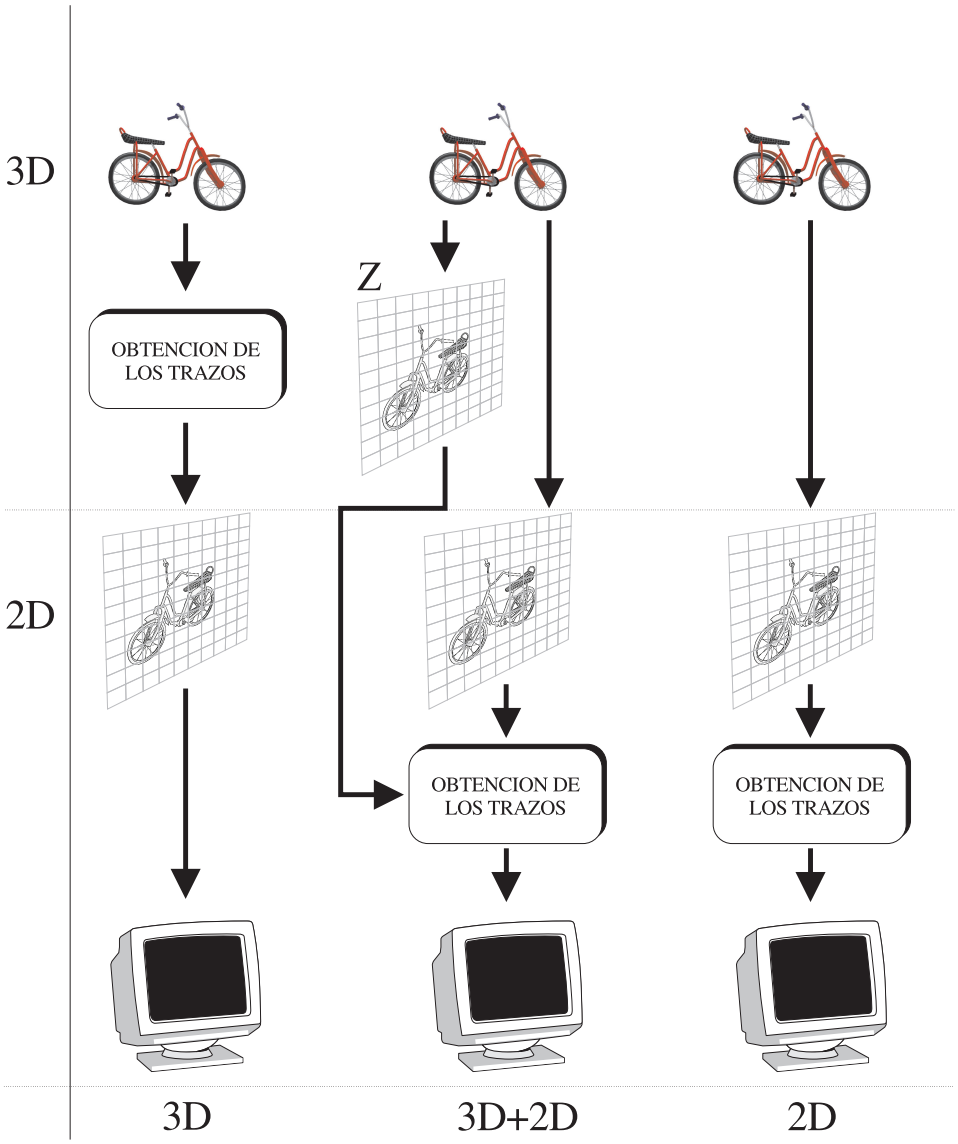


Figura 1.21: Clasificación de los algoritmos según el espacio donde se apliquen.

en filtros, que son usados en el tratamiento de imágenes.

Hay un elemento común que nos permite agrupar a los algoritmos: la dimensión en la cual se realiza el proceso de obtención de las siluetas [46]. A una primera categoría, 3D, pertenecen aquellos algoritmos que se aplican sobre el modelo 3D y los datos que obtienen son 3D. En otra categoría tenemos los métodos mixtos 3D+2D: primero realizan una extracción de información del modelo 3D, almacenándola en forma bidimensional (memoria de profundidad, etc), la cual es usada en una segunda fase como si fuera bidimensional. Por último tenemos aquellos que se realizan en medios bidimensionales como pueden ser la memoria de imagen (*Frame buffer*) y otro tipo de memorias (de estarcido (*Stencil*), etc). La característica que diferencia a estos últimos de los algoritmos de tratamiento de imágenes, es que los primeros operan sobre datos generados a partir de modelos 3D y no sobre datos obtenidos directamente

de la memoria de imagen. En la Figura 1.21 se puede observar gráficamente la clasificación de los tres tipos de algortimos. Dentro de la categoría de los algoritmos que trabajan directamente sobre modelos 3D se puede hacer una subclasificación entre aquellos que trabajan sobre modelos basados en caras planas, normalmente triángulos, y aquellos otros modelos que representan al objeto por una superficie o un conjunto de superficies, normalmente no planas. Dentro de esta categoría se puede obtener otra subcategoría dependiendo de la forma de definir las superficies: paramétricas e implícitas o algebraicas.

Características generales

Los modelos basados en caras planas convexas son ampliamente usados en la visualización. Normalmente, en los modelos poligonales tenemos una discretización de la forma del objeto, así como de las las normales. Las normales son asignadas o bien a los vértices o bien a las caras. Una posibilidad es asignar a cada vértice una normal que corresponde exactamente a la superficie del objeto (si es posible). Otra posibilidad consiste en usar las normales de las caras para obtener las de los vértices, mediante el cálculo de la media de las normales de las caras que circundan el vértice. Al calcular la normal de forma indirecta, cabe la posibilidad de que las normales obtenidas no representen correctamente a las normales verdaderas. En tal caso, aunque los resultados que se obtienen pueden ser incorrectos, no puede haber casos degenerados que interrumpan la ejecución del programa, como sí puede suceder en el caso de superficies no poligonales. Ésto es, si una superficie, sea del tipo que sea, tiene una representación poligonal, para cada vértice es posible obtener una normal. En cambio, en ciertas superficies no es posible obtener, de forma inmediata, el valor de la normal en ciertos puntos (casos degenerados, puntas, etc), por lo que tiene que haber un tratamiento especial. Por otra parte, la desventaja es que para alcanzar el grado de aproximación deseado son necesarias muchas caras lo cual aumenta el número de cálculos y el uso de memoria.

Los algoritmos que utilizan modelos poligonales usan como elemento para formar la/s silueta/s un conjunto de aristas. Suponiendo figuras cerradas bien formadas (sin interpenetraciones, etc), toda arista es compartida por dos caras (objeto 2-variedad (*2-manifold*)). Se parte de un volumen de vista normalizado, con el observador en la parte positiva del eje z (sistema coordenado derecho). Ya que las aristas no poseen directamente normal, se toma como criterio para su selección los valores de las normales de las caras que comparten la arista: formarán parte de la silueta aquellas aristas que compartan una cara visible y otra invisible. La visibilidad o no de una cara viene determinado por el valor de la z de la normal de la cara, si $z > 0$ entonces es visible, si $z < 0$ entonces la cara es invisible. En caso de que la z sea igual a 0, dependiendo de la aplicación, se considera como invisible, ya que la arista de otra cara adyacente será visible y no producirá problemas. En superficies abiertas, las caras del límite de la figura poseen aristas que no son compartidas. Estas aristas también deben ser consideradas como aristas de la silueta. Más formalmente, una arista formará parte de la silueta si y sólo si la misma pertenece a una sola cara o, si es compartida por dos caras, una es visible y la otra no.

Los modelos que se basan en superficies, tanto implícitas como paramétricas, tiene la posibilidad de determinar la normal en cada punto de la superficie de forma analítica (puede que no en todos los puntos). Dado un objeto que es representado por la descripción de su superficie mediante formulación paramétrica $S(u, v)$, se puede obtener la función que define la normal para cada punto derivando la función con respecto a cada parámetro obteniendo las tangentes en cada dirección $T_u(u, v) = \frac{\partial S(u,v)}{\partial u}$ y $T_v(u, v) = \frac{\partial S(u,v)}{\partial v}$. Dados los dos vectores, el valor de la normal para un punto u, v viene dado por el producto vectorial de ambas normales, esto es $T(u, v) = T_u(u, v) \otimes T_v(u, v)$. Como se puede observar, la función que define las normales puede a su vez ser considerada como otra superficie, lo cual será utilizado más adelante. Un problema es la existencia de puntos en los cuales la normal no esté bien definida [18, Págs. 281-284]. Suponiendo que no existe este problema, la función $T(u, v)$ es una función de quinto grado en los dos sentidos, aunque puede ser simplificado a grado 3 [52]. Si se dispone de dicha función la silueta viene definida por la siguiente condición $T_z(u, v) = 0$, ésto es, la componente z de la normal es cero. Las raíces se pueden encontrar mediante métodos numéricos (Newton-Raphson, etc).

Un esquema original de extracción de siluetas basado en la función normal es el siguiente [52, Págs. 533-538]. Dada una figura convexa, el conjunto de las normales normalizadas forma una esfera de radio 1. Una manera de aprovechar esta característica es tomar varias normales y con las mismas aproximar una superficie, que será un parche de la esfera. Para calcular la silueta se calcula la intersección de la superficie con el plano $z = 0$. Ésto se aplicaría a todos los parches para obtener la silueta completa.

- **Métodos 3D**

  - Caras planas
    El primer algoritmo que permitía obtener siluetas fue el algoritmo de Roberts [21, Págs. 665-666], que fue pensado como algoritmo para eliminación de líneas ocultas. Impone la condición de que las aristas pertenezcan a caras de una figura poliédrica convexa. La dificultad aparece cuando en la escena hay más de un objeto. El algoritmo, en primer lugar, elimina todas las caras que miran hacia atrás (*Back face culling*) y luego comprueba el estado de cada arista con el resto de las caras. Al ser los objetos convexos, una arista, en caso de que intersecte con una cara, puede estar total o parcialmente oculta. En el segundo caso pueden aparecer uno o dos subsegmentos. Para hallar dichos subsegmentos habrá que realizar el cálculo de intersecciones entre aristas. La visibilidad de los vértices se determina trazando una línea entre el punto y el observador. Se sabe si el proyector atraviesa un poliedro si el mismo posee algún punto que es interior a alguna cara visible.

    Un algoritmo más sofisticado, que permite objetos cóncavos, es el algoritmo de Appel [21, Págs. 666-667]. Pensado también para realizar una eliminación de partes ocultas puede ser usado para la obtención de siluetas. En este caso la única condición es que las aristas formen parte de polígonos. Hace uso de la coherencia de la siguiente mane-

ra. Cada vez que una arista atraviesa una cara visible se produce un cambio de lo que Appel denominó como invisibilidad cuantitativa (*Quantitative Invisibility*). Así, cuando se produce una intersección en la que se produce un paso por detrás de una cara, aumenta la invisibilidad, mientras que cuando se deja de estar detrás se disminuye. Una arista será visible cuando su invisibilidad cuantitativa sea igual a cero. Si no se producen interpenetraciones, estos cambios de visibilidad sólo se producen en las aristas que forman la silueta, según Appel, en la línea de contorno. Para determinar que se produce dicha intersección, se comprueba si la arista de la silueta atraviesa el polígono formado por el observador y la arista en cuestión. Para determinar el punto de la intersección se considera la arista de la silueta como el lado de una ventana de recorte.

El algoritmo funciona de la siguiente manera: se calcula la invisibilidad cuantitativa de un punto de una de las aristas marcadas como silueta. Ésto se podría hacer lanzando un rayo que pasara por el punto. Partiendo de esta arista se iría cambiando la invisibilidad en los puntos en los que se produjera una intersección. Para que el algoritmo funcione con siluetas interiores, debe mantener una lista o conjunto de marcas de todas las aristas que han sido marcadas como silueta.

Markosian [45] presenta un algoritmo basado en el método de Appel, pero realizando ciertas modificaciones para que pueda ser usado de forma interactiva. Para ello realiza una búsqueda aleatoria de las aristas que pueden formar parte de las siluetas. Ésto puede llevar a la pérdida de ciertas siluetas. En caso de que se quiera una búsqueda exhaustiva, el tiempo de cálculo se multiplica por cinco. El concepto invisibilidad cuantitativa es usado para indicar el número de caras que hay delante de una arista. Si un punto tiene una invisibilidad cuantitativa igual a cero implica que el punto es visible. Los problemas que aparecen en el algoritmo de Appel al producirse los cambios de invisibilidad en los propios vértices, lo que llaman *Cusp vertex*, son resueltos de dos maneras: En una primera forma se crean primero las cadenas y luego se aplica la invisibilidad cuantitativa. A partir de esta información se determina la posición de la cadena de aristas que forman la silueta (*Cluster*). La otra forma es lanzando rayos. El algoritmo está pensado para escenas en las cuales los objetos están totalmente contenidos en el volumen de visión, aunque proponen alternativas en caso contrario. Las intersecciones entre aristas se realizan en coordenadas de imagen utilizando un esquema de línea de barrido. Las cadenas pueden ser visualizadas de diferentes formas.

- Superficies
  Elber y Cohen proponen un método, operando con superficies paramétricas (splines), para la eliminación de partes ocultas, que tiene la capacidad de obtener las siluetas [17]. Para limitar el problema, no permiten que aparezcan siluetas interiores. En tal caso se divide el objeto. Ya que las siluetas son curvas no isoparamétricas dependientes del observador, se recurre a un esquema de subdivisión de

la superficie hasta que se alcanza cierta tolerancia, basándose en propiedades de la subdivisión de lo splines, de tal forma que la componente z de la normal sea cero. Una vez hecho ésto, los segmentos deber ser unidos para forma una curva continua (Figura 1.22).

Sederberg y Zundel presentaron un método para la visualización de superficies algebráicas mediante un algoritmo de línea de barrido [66]. El algortimo trata correctamente las singularidades, y permite obtener la silueta de la figura. Para ello introducen una nueva base polinomial llamada base polinomial piramidal de Bernstein (*Bernstein piramid polinomial, BPP*). Una de las ventajas que aportan las superficies implícitas es que permiten ser mezcladas (*Blended*) de forma controlada, normalmente de forma suave y continua. La principal característica de este algoritmo es que la base desarrollada permite calcular, de forma robusta y estable, para cada línea de barrido, los llamados puntos críticos (*Critical points*). Entre dos puntos críticos, la superficie es continuamente visible, por lo tanto, dichos puntos críticos son equivalentes a los puntos que definen la silueta. Para el cálculo de la silueta realizan la intersección de la representación de la superficie con la superficie polar con respecto al observador de la misma.

- **Métodos 3D+2D**
  Dentro de las técnicas que realizan parte del método en tres dimensiones mientras que otras operaciones se realizan en dos dimensiones, está el método propuesto por Saito y Takahashi [59]. El algortimo se basa en lo que llaman memorias G o báferes Geométricos (*G-buffers*), en los cuales se almacenan ciertas propiedades geométricas del modelo para cada píxel, después de eliminar las partes ocultas y habiendo realizado la proyección. Entre los búferes geométricos están aquellos que contienen la identificación del objeto y del parche, de la coordenada u, de la coordenada v, de las coordenadas x, y y z en coordenadas de mundo, de las coordenadas x, y y z del vector normal y la coordenada z en coordenadas de pantalla. A esta información se le pueden aplicar procesos geométricos (proyección de perspectiva, eliminación de partes ocultas), procesos físicos (suaviza-

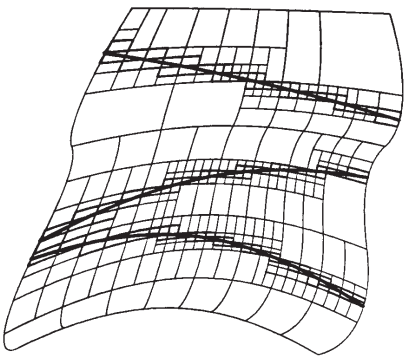


Figura 1.22: Extracción de la silueta en superficies paramétricas [17].

A	B	C
D	X	E
F	G	H

Figura 1.23: Posiciones de los píxeles para aplicar el operador de Sobel.

do, aplicación de texturas) y procesos artificiales (mejora de imágenes). Todas estas características se pueden obtener en un postprocesado. Las siluetas son extraídas mediante operadores diferenciales de primer y segundo orden (desarrollados en el campo del procesado de imágenes), en particular el operador de Sobel: $g = \frac{|A+2B+C-F-2G-H|+|C+2E+H-A-2D-F|}{8}$ (Figura 1.23). Esta operadores, detectores de bordes, se aplican no sobre la información visual de la imagen sino sobre las profundidades almacenadas en la memoria de profundidad. Las siluetas aparecen como la discontinuidad de orden cero de las profundidades, mientras que lo que llaman aristas internas, las aristas que son unión de caras no coplanarias, son detectadas mediante una segunda aplicación del operador. Para la corrección de distintos problemas, se proponen ciertas modificaciones para los operadores y para el cálculo de las profundidades. En la Figura 1.24 se muestra un ejemplo de su aplicación.

Salisbury [60, 61, 62], así como Winkenbach y Salesin [75, 76] han desarrollado varios métodos orientados hacia la ilustración (Figura 1.25). En [75, 76] usan lo que llaman texturas de trazos, tanto para obtener un aspecto, la parte de textura, como para obtener un tono, la parte de suavizado. Usan un mapa planar (*Planar map*) que permite obtener relaciones de adyacencia. Recorriendo dicha estructura y en función del tipo de arista y la diferencia de tono que haya entre las caras adyacentes, se dibujan las siluetas de forma acorde (más o menos guesas, oscuras, etc).

Otro método que trabaja en 3D+2D es el presentado por Leister [40], con la diferencia principal de que la información sobre la que se trabaja en 2D proviene de un trazado de rayos (Figura 1.26).

Richen y Schofield [57] presentan otro algoritmo que trabaja con la aproximación 3D+2D. De hecho el proceso de visualización del sistema realiza esta división. Este sistema está orientado a la producción de imágenes expresivas de forma interactiva, a partir de modelos arquitectónicos, principalmente. El sistema, en la primera fase realiza un cálculo de la imagen en 3D, almacenando, a la manera de Saito y Takahashi, la información de profundidad del báfer-z, así como una memoria en la cual se almacena el material del cual están compuestas las distintas partes del modelo, en coherencia posicional con el báfer-z. Ésto es, para aquellas partes del modelo que son visibles, se dispone de información del material. Se podría hablar de que se crea una memoria de material (*Material-buffer*). Estos dos tipos de información se pueden usar de manera separada o

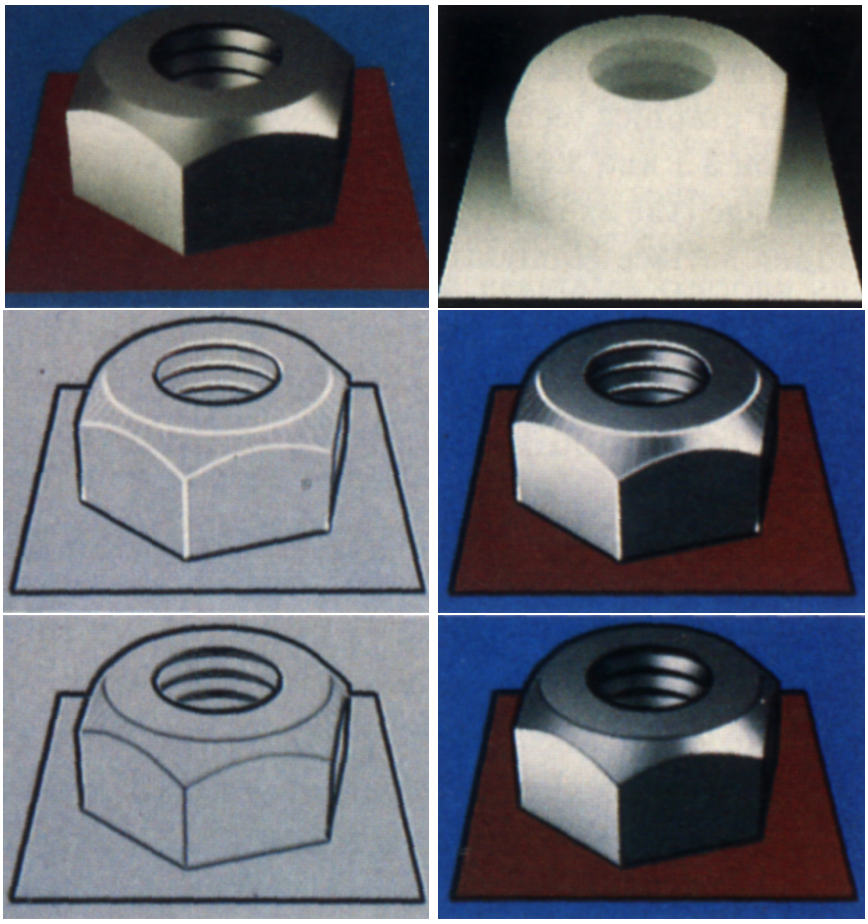


Figura 1.24: Ejemplos de aplicación de los báferes-G [59].

conjunta en una segunda fase para convertir la imagen fotorrealista en imagen pictórica. Por ejemplo, la información del material se puede usar como máscara, de la misma manera que funciona la memoria de estarcido (*Stencil-buffer*), para hacer que solo se afecten las partes de la imagen que se corresponden con cierto material.

- **Métodos 2D**
  P. Rustagi [58] desarrolló un método que trabaja con información bidimensional. Para ello se basa en la llamada memoria de estarcido. La memoria de estarcido sirve para almacenar información que puede ser usada como máscara para otras operaciones (estarcir consiste en dibujar con una máscara). Por ejemplo, dado una representación en la memoria de estarcido, se puede hacer que lo que se dibuje en la memoria de imagen (*Frame buffer*) tenga la misma forma que lo que hay en la memoria de estarcido. Para obtener la silueta de un objeto usando la memoria de estarcido es necesario dibujar el objeto cuatro veces, estando cada vez desplazado en un píxel en las direcciones x e y. Para ello se cambia la transformación de vista. Cada vez que se dibuja el objeto se incrementa el valor del píxel en la memoria de estarcido, la cual está inicializada a cero. Cuando se han realizado las cuatro pasadas, los píxeles del interior

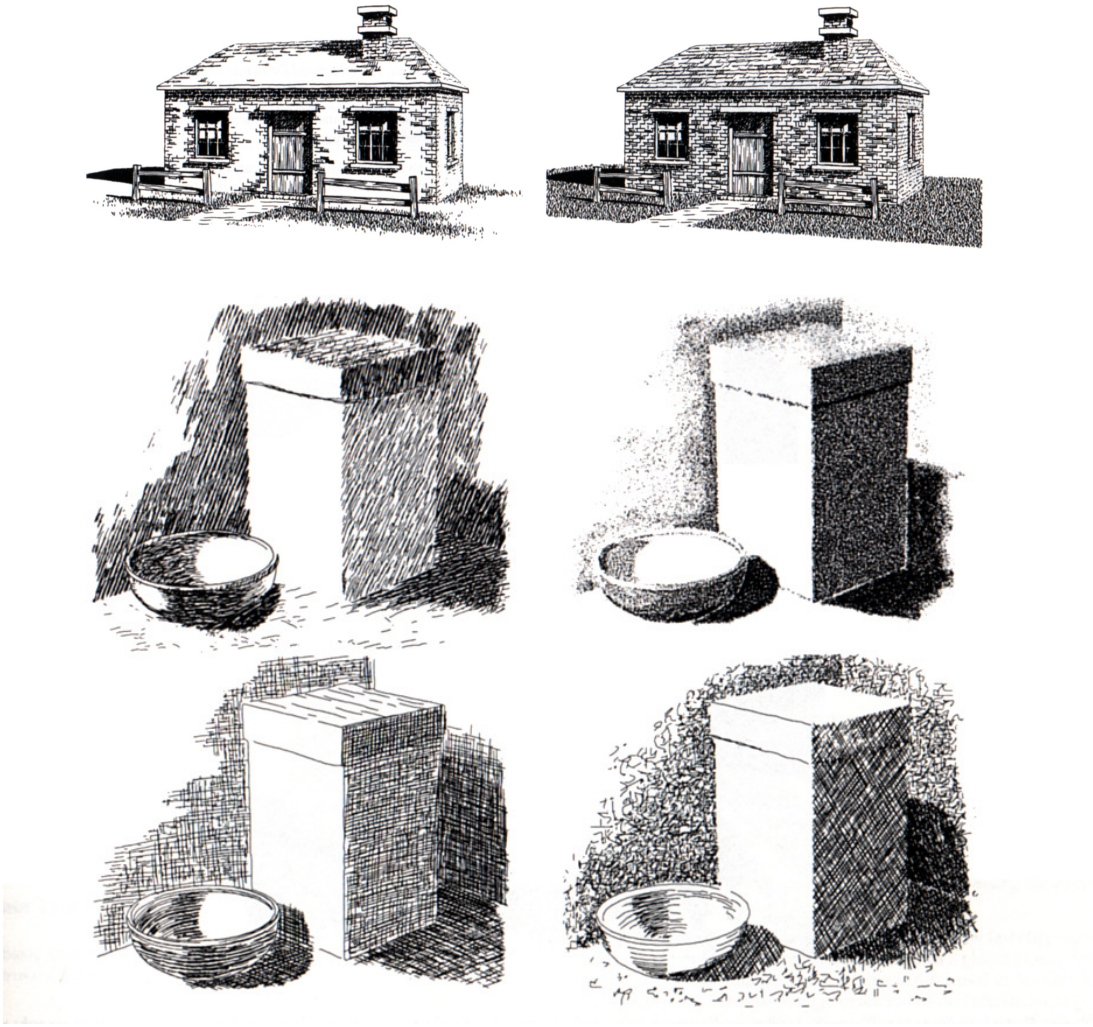


Figura 1.25: Ejemplos de ilustración [75, 61].

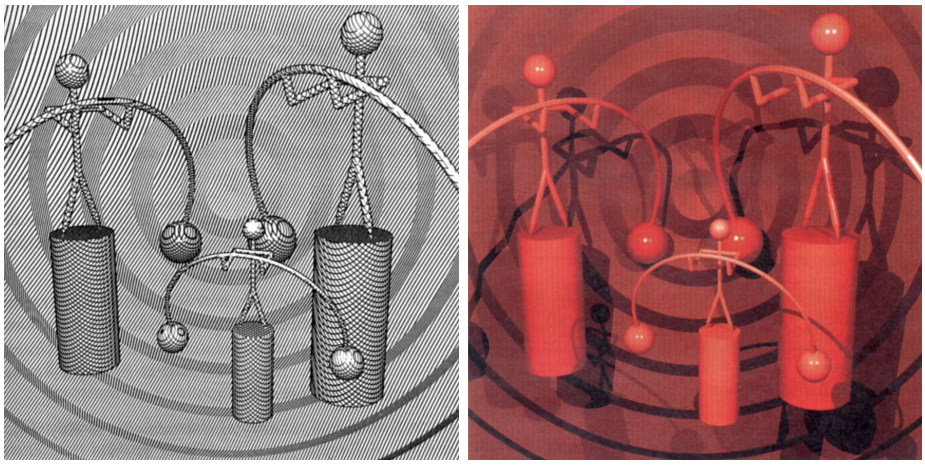


Figura 1.26: Ejemplos de ilustración [40].

tendrán el valor 4, los píxeles de la silueta tendrán valor 2 ó 3 y los del exterior 0 ó 1. Una vez que se tiene esta información se pueden activar las memorias de imagen y de profundidad (*Z-buffer*) y usar la memoria de estarcido como máscara: sólo se permitirá dibujar un píxel si el valor del píxel correspondiente en la memoria de estarcido es igual a 2 ó 3. Dibujando un polígono que cubra el objeto se obtendrá sólo la silueta del color deseado (Figura 1.27).

Un trabajo muy interesante por los resultados estéticos que se obtienen es el de Hsu [30], el cual usa la Deformación Libre de Forma (Pág. 39) para la obtención de trazos con formas arbitrarias en un sistema de dibujo 2D (Figura 1.28).

1.8.2. Animación

A la par que se ha ido desarrollando la informática, lo han hecho las capacidades de mostrar imágenes de síntesis cada vez más complejas y sofisticadas. El problema más difícil que se ha encontrado es el de la reproducción de formas naturales y de seres vivos, y en la aplicación de los principios de la animación tradicional (Pág. 11). Se han desarrollado modelos, fractales, sistemas de partículas, para la representación de fenómenos como el fuego, el agua, nubes, y objetos naturales como montañas, árboles, etc. La representación de las formas curvas y redondeadas, que normalmente se encuentra en los seres vivos, es un problema que se sigue investigando.

Dentro del campo de los métodos desarrollados para obtener modificaciones de los modelos, de tal forma que los mismos presenten un comportamiento asimilable, en cuanto a flexibilidad de movimiento, a los encontrados en la animación clásica son varios, aunque no tan variados como los investigados

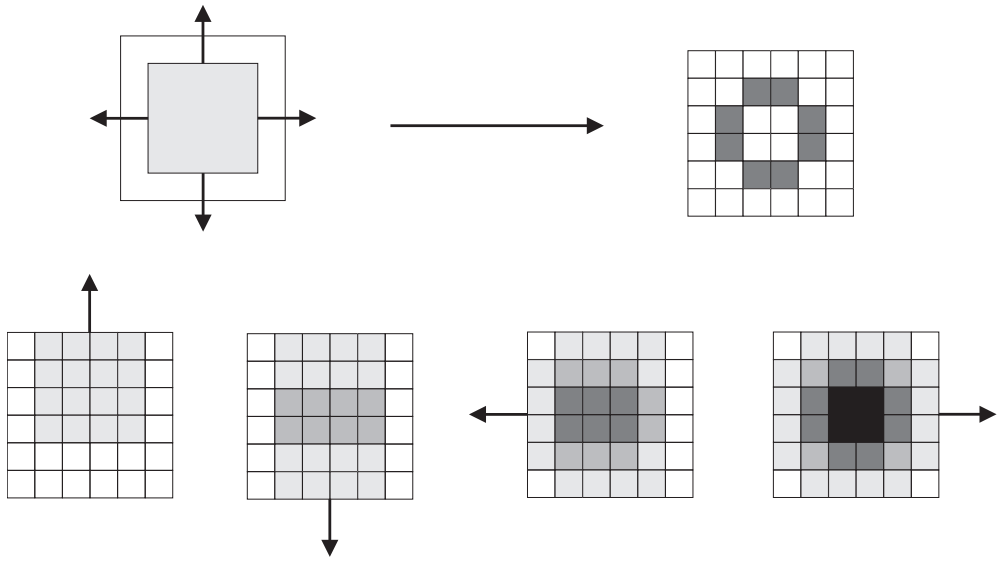


Figura 1.27: Obtención de la silueta con el método de Rustagi.

Figura 1.28: Ejemplo de distintos tipos de líneas usando trazos esqueléticos [30].

en visualización. Hay que observar, que las necesidades de variabilidad en la forma de los objetos, y sus modelos asociados, es una característica propia de los seres vivos, ya que, en general, los seres inertes no cambian (en periodos de tiempo cortos). Las imágenes que se generaban al principio de la Informática Gráfica, usaban modelos sencillos y generalmente de objetos inertes: edificios, maquinaria, etc. Por otro lado, aunque se observa la necesidad de obtener imágenes mas realistas cuando se visualizan seres que cambian, existen también las limitaciones del modelo que se usa, que, como se ha dicho, suelen ser modelos poligonales, así como las restricciones de capacidad en el hardware. El hardware continua su mejora constante, y en el campo del software han aparecido desarrollos y modelos que permiten obtener modelos de seres vivos y modificaciones sobre los modelos, cada vez con más posibilidades. Ello ha permitido la síntesis de imágenes muy realistas de modelos de personas en movimiento. Estas capacidades de modificación y deformación, llevadas a su máxima extensión, están permitiendo, actualmente, la producción de animación de personajes 3D con un nivel de flexibilidad prácticamente igual al que se pueda encontrar en la animación clásica.

Los esquemas para producir deformaciones más conocidos son la Deformación Libre de Forma (*Free Form Deformation*), y la Transformación No-Lineal (*Non-Lineal Transformation*). Estos métodos, además, presentan la ventaja de que son independientes de la representación. Ésto quiere decir que el método se puede aplicar tanto a la geometría que define al objeto (vértices), como a la geometría de las superficies paramétricas que permiten definir la forma del objeto. Estos esquemas permiten definir la modificación del objeto, ya sea directa o indirectamente. En caso de que se desee realizar una animación, es necesario poder crear una secuencia de modificaciones.

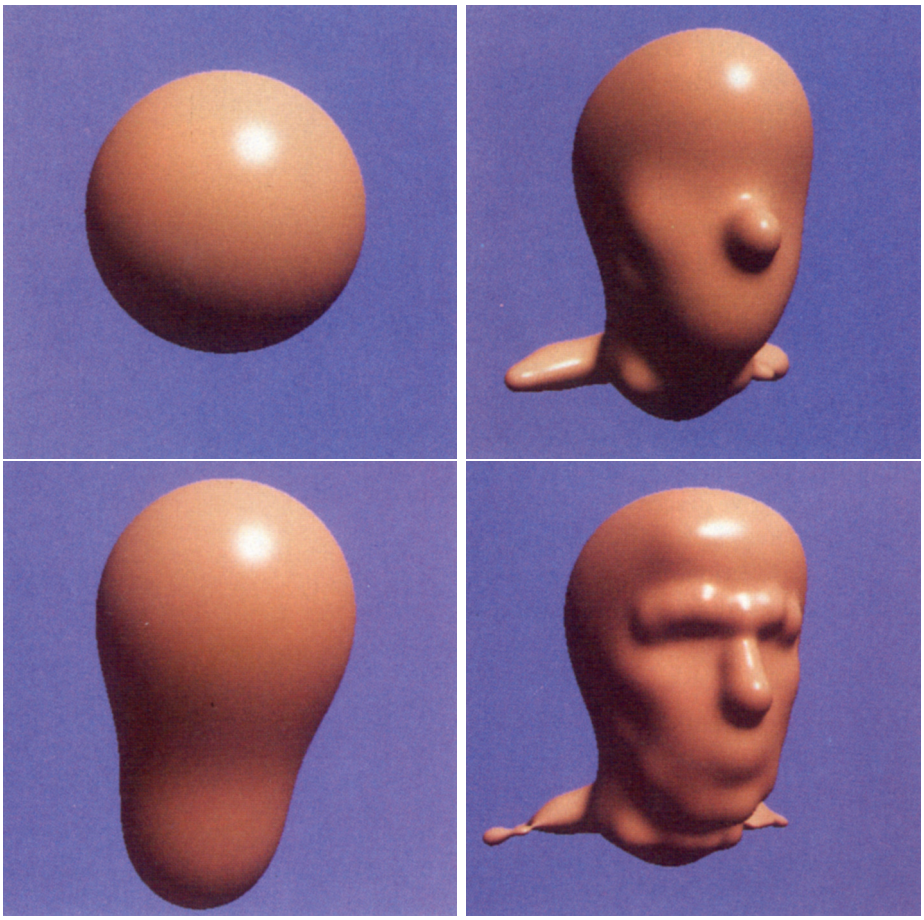


Figura 1.29: Ejemplo de superficies suaves [53].

- **Deformación Libre de Forma** (*Free Form Deformation*)
  El método de la DLF se basa en la siguiente idea: dado un objeto que está incluido dentro de un material que lo recubre totalmente, si se deforma dicho material, el objeto se deforma de la misma manera. Para su implementación se hace uso de hiperparches B-spline. Los pasos que sigue el algoritmo son los siguientes, una vez definida la malla de puntos que envuelve al objeto:

  1. Calcular las coordenadas paramétricas de los puntos.
  2. Modificar los puntos de la malla de control para producir la deformación deseada.
  3. Recalcular las coordenadas cartesianas de los puntos en base a sus coordenadas paramétricas.

  Desde su desarrollo [65], el método de la DLF ha sido ampliamente utilizado para la modificación de modelos. Ésto se debe a la gran capacidad de modificación y al control local que tiene dicho método.

  Esta utilización de superficies de control la podemos encontrar anteriormente en la técnica animación basada en esqueletos [7]. Litvinowickz [41] también hace uso de las DLF en su sistema de animación bidimensional.

Coquillart [12] presenta una variación mediante la cual se pueden usar una mayor variedad de mallas de control, y en [13] realiza una extensión para su utilización como herramienta de animación. Una nueva extensión de los tipos de mallas que se pueden usar se encuentra en [43]. Hsu [30] usa las DLF para la obtención de lo que llaman trazos esqueléticos (*Skeletal Strokes*) (Figura 1.28). En [31] se muestra un método de manipulación directa de las DLFs. Karan [34] hace uso de curvas de control de una forma relacionada con las DLFs.

- **Transformación No-Lineal** (*Non-Lineal Transformation*)
  Éste es el otro método que es independiente de la representación y que permite la deformación de objetos. Desarrollado por Barr [1], es una extensión de las transformaciones geométricas consistente en variar el parámetro que define la transformación. Debido a que es éste el método que se ha elegido para nuestro modelo, una explicación más en profundidad se deja para el siguiente capítulo (Pág. 52).

Otra posibilidad de obtener animación es mediante el uso de las superficies implícitas debido a las características inherentes que tienen en cuanto a que es capaz de producir fácilmente superficies que son contínuas y tienen formas suaves, a diferencia de los modelos poligonales y los basados en superficies paramétricas en las cuales los objetos se forman por composición de partes. El uso de superficies implícitas es a la vez una forma de modelar y de producir animación, al cambiar la forma y posición de los elementos modeladores. Una superficie implícita se define como el conjunto de puntos que cumple una propiedad, en particular el de poseer un isovalor.

$F(p) = isovalor$

F es una función de campo escalar.

El primer trabajo sobre superficies implícitas es el de Blinn [5] (*Blobby molecules*), seguido por Nishimura y colaboradores (metabolas) [54], Wyvill y colaboradores [78, 79] (*Soft objets*), y otros [53] (Figura 1.29).

Capítulo 2

Modelo

El objetivo de esta tesis doctoral es proponer un modelo que permita obtener animación con estética bidimensional partiendo de un modelo tridimensional. Los condicionantes que debe cumplir dicho modelo se han expuesto en el capítulo anterior. En éste se muestran los componentes del modelo que hemos desarrollado y al que hemos llamado *"Alhambra"*.

Una vez se plantearon los objetivos que se deseaban alcanzar, se desarrollaron varios métodos hasta alcanzar resultados positivos. Estos primeros métodos se comentan en la sección de desarrollos previos. Estos trabajos sirvieron para delimitar cuales eran los elementos para alcanzar los objetivos. El problema global queda dividido en dos: un problema de visualización y un problema de animación. La solución del primer problema se alcanza con lo que hemos denominado trazos y el modelo formal de *Luces Virtuales* que le da soporte. Se desarrolla un método que permite determinar la visibilidad de cualquier conjunto de elementos geométricos, característica necesaria para la visualización de los trazos. Por último, se desarrolla una extensión al método de la Transformación No-Lineal como alternativa a la Deformación Libre de Forma.

2.1. Desarrollos previos

Para alcanzar los objetivos se estudiaron varias alternativas. Aunque las soluciones encontradas producían buenos resultados, las mismas eras incompletas en el sentido de que no resolvían todos los casos que se podían presentar, o, al menos, no de forma completamente satisfactoria, o la estética obtenida no era buena en todas las situaciones. Hay que destacar que, aunque en sí no fueron resultados totalmente aceptables, sirvieron para delimitar lo que en posteriores desarrollos se convertiría en la solución que se propone en la presente tesis.

Se hicieron tres desarrollos. En una primera línea, se intentó usar un modelo poligonal a partir del cual se obtenía cierta información que era utilizada para generar los trazos. El segundo método trabajaba en el espacio de la imagen, usando la información discretizada para formar las cadenas que representan la forma del objeto. La tercera línea que se investigó fue la relacionada con el

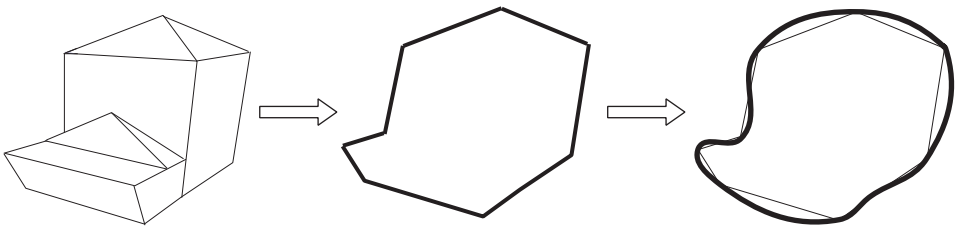


Figura 2.1: Obtención de los trazos.

cálculo de las intersecciones. A continuación se da una explicación más extensa de los tres métodos desarrollados.

El primer método [50] buscaba la producción de los trazos. Ésto es así debido a que, como se ha comentado, los trazos en general tienen una mayor capacidad semántica que el color. Además, las herramientas de visualización con las que se podía contar en aquel momento, no tenían las capacidades que se encuentran actualmente, por lo que se decidió dejar la parte del tratamiento del color para una vez estuviera resuelto el problema de los trazos. Se partía de un modelo poligonal muy sencillo, compuesto de figuras simples como cubos, pirámides, etc. Los objetos se formaban componiendo estas piezas sencillas. Para la obtención de los trazos, primero se marcaban y proyectaban los puntos que formaban parte de las distintas siluetas. A partir de los puntos proyectados se formaban los trazos (Figura 2.1). Una característica muy importante es que la figura final que se obtenía a partir de los puntos no era poligonal (excepto que se quisiera expresamente), ya que dichos puntos eran tomados como puntos de paso de splines cúbicos produciendo formas suaves y haciendo que se recuperara la forma original perdida al aproximar el modelo con un número reducido de caras planas.

Al partir de modelos poligonales muy sencillos, las formas que se obtenían presentaban ciertas limitaciones en cuanto a la posibilidad de ser observados desde cualquier parte (Figura 2.2). También cuando se realizaban ciertos movimientos, giros especialmente, se perdía la coherencia en la forma del objeto. Una manera de resolver ésto era aumentar el número de caras para que la aproximación poligonal fuera mejor. Se observó que para obtener una buena representación, que fuera coherente independientemente del movimiento y de la posición, se tenía que aproximar el objeto con tantas caras que la poligonal formada por los puntos proyectados era visualmente igual a la curva B-spline que se interpolaba con los mismos, con lo cual se perdía la mayor ventaja aportada por el método.

Por otra parte, aparecieron problemas al rellenar las figuras debido al espacio que quedaba entre la curva B-spline y el polígono, y también con las relaciones entre trazos. El primer problema se producía cuando el modelo poligonal era muy sencillo, y por tanto, la curva recuperaba la mayor parte de la forma perdida por ser una aproximación muy burda (p.e. un cubo aproximando una esfera). El problema estaba en que la curva producía un espacio que realmente no estaba ocupado por el modelo. Mientras no hubiera cercanía entre distintos objetos, lo cual se traducía en que no hubiera intersecciones, no había

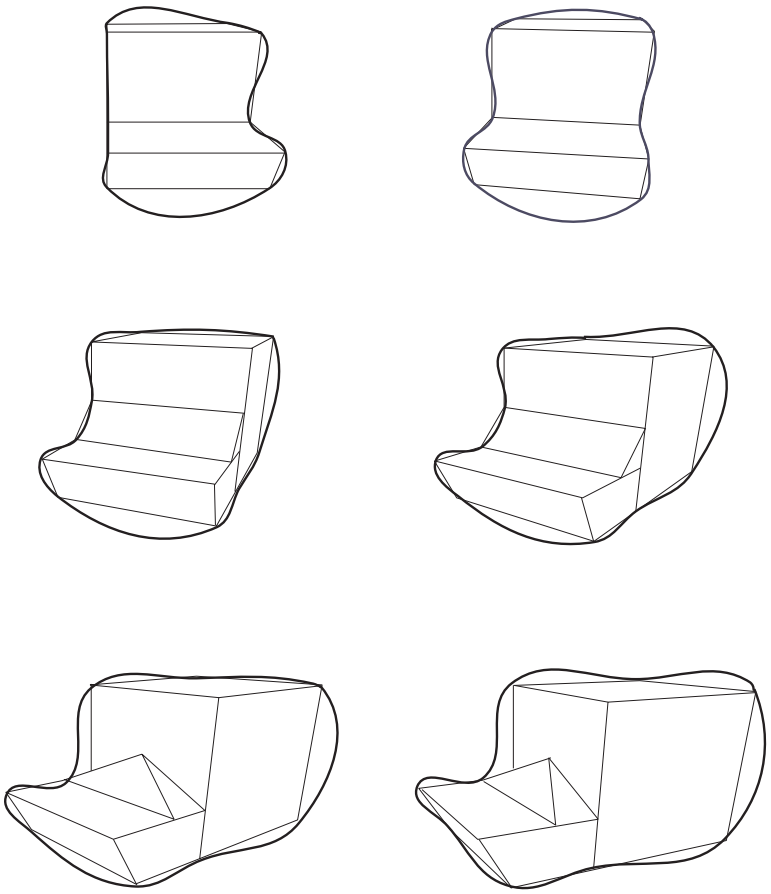


Figura 2.2: Modelo con distintas etapas de refinado de la forma.

problema, pero en caso contrario, los resultados podían ser inesperados ya que en dichas zonas se aplicaba la profundidad de la poligonal que lo controlaba, y la misma podía ser incorrecta. La solución pasaba de nuevo por aumentar la resolución del modelo. El segundo problema se producía cuando había varios trazos y entre los mismos se producían interrelaciones, por ejemplo, dada una posición, se producían dos siluetas que no tenían puntos en común, mientras que desde otra posición, las mismas siluetas se debían fusionar en una sola, perdiendo cada silueta una parte. Para ello se debía controlar las intersecciones en dos dimensiones y a partir de las mismas generar el nuevo polígono de control.

De la implementación de este sistema se obtuvieron dos ideas que se utilizaron en el método que se propone. La primera es que, en caso de usar un modelo poligonal, se debe utilizar un modelo preciso en el sentido de que aproxime suficientemente al objeto como para no necesitar un proceso de suavizado al generar los trazos. Si el modelo es lo suficientemente preciso, la poligonal que se forma se observa como una curva, y los objetos que tiene una forma poligonal se representan exactamente. Además el modelo ocupa realmente el espacio que indican las siluetas.

La otra idea importante es que el espacio 2D debe ser, si ésto es posible, un

espacio en el que se 'resuma" o "condense" información de un espacio superior. La recuperación de información perdida al proyectar puede llevar a resultados no unívocos. Esta idea es pilar para la afirmación que se ha hecho previamente de que el modelo del que se parta, para la generación de animación bidimensional, debe ser tridimensional.

El segundo método que se estuvo estudiando fue el de generar los trazos a partir de la información que se almacenaba en el espacio de la imagen, ésto es, dependiente de la resolución. El método que se exploró estaba orientado a la detección de los vértices que formarían parte de los trazos. De nuevo, el modelo era poligonal. La idea en la que se basaba la detección de las aristas que formarían los trazos, era que dichas aristas, al ser bordes que limitaban la parte visible y la invisible, si se hacía previamente una eliminación de las caras ocultas, y a continuación se intentaban dibujar todos los polígonos, y de cada uno de ellos, todas sus aristas, resultaría que las aristas que forman parte de la silueta sólo son recorridas una vez, mientras que las que no lo son, se recorren dos veces. La característica principal de este método es que la información sobre el número de veces que era recorrida una arista se almacena en un báfer del mismo tamaño que el de la imagen. De hecho, lo que se almacenaban no son las aristas sino que cada vértice guardaba el número de veces que eran recorridas las aristas a las cuales pertenecía. Para cada posición se podía crear una lista de los vértices que ocupaban la misma posición. La otra característica es que el método permite trabajar con conjuntos no relacionados de caras. El modelo se puede representar por un conjunto de caras entre las cuales no existe ninguna relación, excepto la de usar un mismo conjunto de vértices.

Este método se implementó y funciona correctamente, pero es poco eficiente, ya que cuando se tienen que formar las cadenas y se producen colisiones (posiciones donde coinciden más de un vértice) la discriminación de la solución correcta es costosa. De este método hemos aprovechado la idea de que es necesario una estructura de datos que permita una rápida selección de las aristas que forman los trazos. Pensamos que el método se puede usar con nubes de puntos obtenidos con un escáner, a partir de los cuales se formarían las caras.

El tercer problema que se estudió fue el del cálculo de las intersecciones, que se deben realizar cuando los objetos son cóncavos y, por tanto, pueden tener más de una silueta (ver montaña de la Figura 1.20). En este caso, pueden aparecer intersecciones entre las distintas siluetas, que, además, pueden ser abiertas o cerradas. Este problema es similar al de la eliminación de partes ocultas, aunque en este caso, no sólo estamos interesados en saber cuando una parte del objeto queda oculta detrás de otra, sino que también nos interesa saber que partes quedan por delante. Ésto, como se explicará más adelante resulta de mucho interés para obtener imágenes de apariencia bidimensional. El camino que se exploró fue el de realizar las operaciones en el espacio de la imagen, trabajando con polígonos que podían ser entregados al algoritmo de forma independiente. Esta forma de operar producía resultados incorrectos debidos a los problemas de alias (*Aliasing*) y de resolución en el báfer de profundidad.

Todos estos caminos explorados sirvieron, en un caso, para ver que en dicha

dirección no se podía seguir, o que los resultados no eran lo suficientemente buenos pero, sobre todo, sirvieron para delimitar cual debía ser el camino a seguir que permitiera solventar todas las dificultades encontradas.

La solución que presentamos se basa en tres elementos principales:

- Un modelo geométrico 3D
- Los trazos, definidos apartir de *"Luces Virtuales"*
- El modelo de las Transformaciones No-Lineales Extendidas Jerárquicas

Estos elementos se comentan en detalle a continuación.

2.2. Modelo geométrico 3D

El primer elemento a considerar dentro del sistema propuesto es el del modelo que permitirá representar los objetos y personajes. El modelo debe poseer unas características especiales que faciliten el desarrollo de los demás métodos y técnicas. Para que el modelo geométrico pueda se utilizado para la creación de animación debe poseer ciertas propiedades, como son el que se pueda evaluar fácilmente su superfice, poder evaluar la normal en cada punto y que permita ser transformado de forma sencilla e intuitiva.

De los distintos modelos de superficies se elegirá aquel que facilite la implementación de la obtención de los trazos. Como se verá a continuación, la definición formal de los trazos y la forma en que son obtenidos hace que elijamos un modelo poligonal basado en triángulos.

2.3. Trazos

Dentro de los componentes que hacen que una imagen tenga una apariencia bidimensional, los más importantes son las líneas de forma. Como se ha comentado en la introducción, estos elementos junto con el coloreado plano conforman la estética de la animación clásica (Figura 2.3). Pero hay que hacer incapié en que son las líneas de forma las que realmente definen dicha apariencia. Se puede pensar en realizar una animación en la cual los personajes no tengan color (en el sentido de que las formas interiores estén coloreadas), pero no se encontrará el caso en el cual no aparezcan las líneas de forma. Tanto más podemos decir de las líneas de forma cuando se trata de ilustración. Con los trazos, aunque no hay color, se puede producir una animación y una ilustración. De hecho, el proceso de la animación y el intercalado, producen imágenes compuestas sólo de líneas, sin ningún relleno. En la animación clásica y en la ilustración, se han utilizado las líneas de forma para conseguir la expresividad necesaria. El relleno de las áreas delimitadas por las siluetas ha sido de color plano, aunque la ultima tendencia es añadir un mayor grado de tonos, buscando hacer que los personajes adquieran un cierto carácter tridimensional.

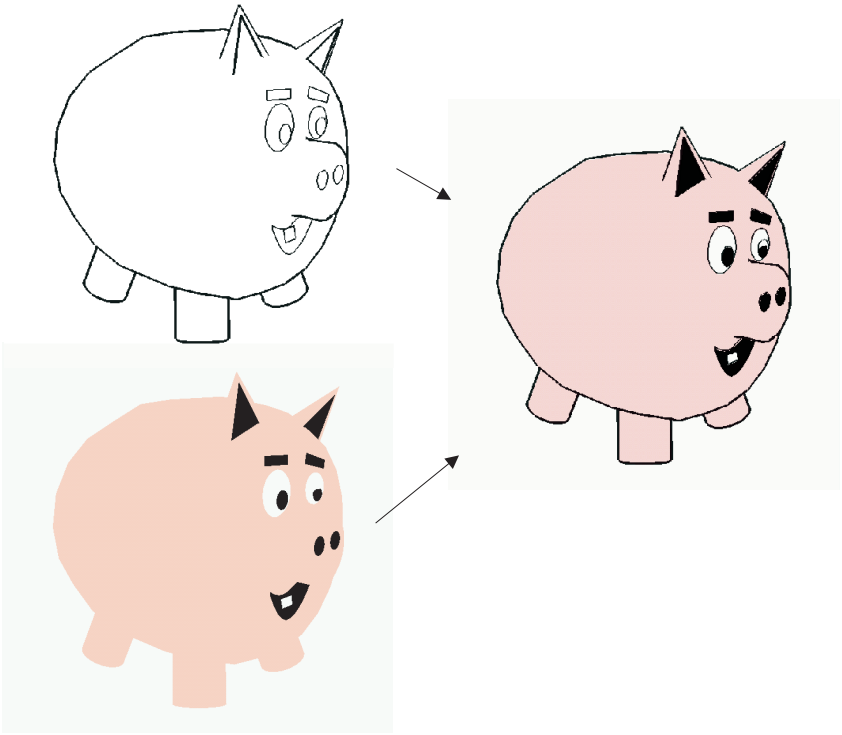


Figura 2.3: Líneas de forma (trazos) y colores planos.

En definitiva, si hay algún elemento pictórico primordial en el dibujo y la ilustración, son las líneas de forma, las cuales hemos llamado trazos (Figuras 1.19 y 2.3). Como se ha dicho anteriormente se puede distinguir entre las líneas de forma que representan las siluetas, interiores y exteriores, que llamaremos trazos externos, y la líneas de forma que permiten distinguir distintas características visuales, que hemos llamado trazos internos. Podemos definir ambos tipos de trazos informalmente como:

- Trazos externos
  Los trazos externos son las siluetas, externa e internas, que posee un objeto cuando es observado desde una posición. Una silueta se puede definir como la curva que representa la frontera entre partes visibles e invisibles de un objeto.
- Trazos internos
  Los trazos internos son aquellas líneas de forma que no son siluetas. En general, los trazos internos distinguen entre partes del objeto que tienen distinta apariencia en algun atributo visual, como por ejemplo el color o el tono.

2.3.1. Algoritmo de visualización

Una vez que se ha hecho una descripción informal de lo que son los trazos, podemos presentar el algoritmo de generación de los trazos:

Para todo Objeto **Hacer**
Obtener los elementos geométricos que formarán los trazos externos e internos
Calcular la intersecciones entre elementos geométricos
Formar cadenas de elementos geométricos
Visualizar las cadenas
FinPara

El algoritmo que se ha expuesto opera sobre elementos geométricos. Un elemento geométrico es un componente de un trazo, y, lógicamente, su naturaleza depende del modelo geométrico elegido para representar el objeto. Puede ser una arista o conjunto de aristas si el modelo es poligonal, o una curva o conjunto de curvas si el modelo está basado en superficies paramétricas.

Para poder obtener los trazos, hay que realizar una búsqueda en el modelo geométrico de los elementos geométricos del mismo que cumplen la característica necesaria para ser clasificados como tales: para los trazos externos se buscan los elementos geométricos que forman el límite entre las partes visibles e invisibles, y para los trazos internos los que delimitan valores de un atributo visual.

El modelo de *"Luces Virtuales"* es utilizado para el tratamiento y obtención de los trazos. Este formalismo permite clasificar, de forma única, a los elementos geométricos que formarán los trazos, tanto externos e internos. El modelo de *"Luces Virtuales"* se convierte en una herramienta formal para la visualización expresiva.

Una vez se han obtenido los elementos geométricos que conforman los trazos, se deben detectar las autointersecciones entre los elementos que forman los trazos. Esta situación puede ocurrir con objetos cóncavos (la montaña de la Figura 1.20). En tal caso, hay que realizar un cálculo que permita determinar que partes son visibles y cuales no. Además, en nuestro caso estamos interesados en poder obtener otros parámetros que permitan visualizar los trazos de una manera más flexible.

En general, e independientemente del tipo de modelo geométrico que se use, una vez son seleccionados los elementos geométricos que definen los trazos, considerados individualmente, hay que realizar una agrupación de dichos elementos para que formen cadenas, que también llamaremos trazos, considerados de forma global. Una cadena es una serie relacionada de elementos geométricos, de tal manera que puede ser tratada como una unidad. La relación que es establece es de continuidad posicional. Las cadenas pueden ser abiertas o cerradas. Se busca el formar cadenas, para poder tratarlas como entidades únicas, con un solo conjunto de atributos, en vez de los distintos atributos de cada uno de sus componentes. Ésto da una mayor capacidad de visualización.

Con la información que se obtiene en las etapas anteriores se pasa a la etapa de visualización.

A continuación se pasa a describir más en profundidad los elementos que se han comentado en los párrafos anteriores, bajo una visión formal, ya que la

implementación de los mismos se verá en el siguiente capítulo.

2.3.2. *Luces Virtuales*

Dado un modelo geométrico se plantea el problema de poder obtener el conjunto de elementos geométricos que representan a los trazos. Mientras que los trazos externos se pueden definir y formalizar fácilmente, no ocurre lo mismo con los trazos internos. Por otra parte, mientras que los trazos externos son elementos que están fijados por las condiciones de la escena, los trazos internos tienen un mayor grado de flexibilidad, ya que su control es definido por el usuario.

El modelo *Luces Virtuales* es un formalismo que nos permite representar las características de los trazos externos e internos de una forma común. Y lo que es más importante, permite al usuario definir cuándo, cómo, dónde y qué elementos geométricos del modelos serán seleccionados como trazos. Esta característica es la que permite cambiar la apariencia de un objeto, pues aunque las siluetas no serán elementos que normalmente sean modificados (aunque se podría hacer para obtener efectos especiales), los trazos internos pueden hacer que un objeto aparezca de una manera o de otra dependiendo de cuando y donde se dibujen. Este formalismo se puede usar para implementar un método para la obtención de los trazos externos e internos, a partir de elementos geométricos obtenidos a partir de superficies paramétricas (curvas paramétricas), modelos poligonales (aristas), etc.

Luces virtuales: selección y clasificación

El modelo de *"Luces Virtuales"* se compone de varios elementos:

- Luces virtuales.
- Modelos de iluminación.
- Un modelo geométrico 3D.

A continuación se definen estos elementos:

Definición 1 *una luz virtual está determinada por una dirección $\vec{v}$, o una posición (x, y, z).*

En el primer caso, la luz virtual está localizada en el infinito, diremos que es *no-local*, mientras que en el segundo está más cerca de la escena, diremos que es *local*. Las luces virtuales son diferentes de las luces normales ya que no contribuyen a la iluminación de la escena. Otra diferencia es que cada objeto en la escena tiene su propio conjunto de luces virtuales, a diferencia de lo que ocurre normalmente, donde las luces son únicas para todos los elementos de la escena. La posición u orientación de las luces virtuales se define en relación con el objeto que las posee. las luces virtuales no tienen intensidad.

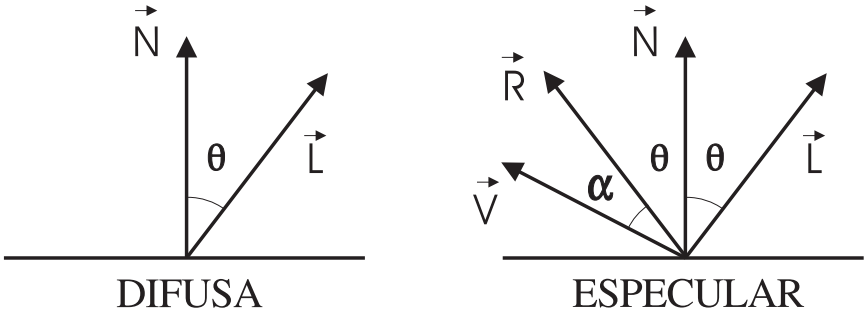


Figura 2.4: Parámetros usados con las luces virtuales.

Las luces virtuales se usan mediante dos modelos de iluminación: difuso y especular.

Definición 2 *un modelo de iluminación difuso asocia un valor real I, entre -1 y 1, a cada punto del objeto para una fuente de luz según la expresión:*

$$I = cos\theta$$

donde θ es el angulo que forma $\vec{N}$, el vector normal a la superficie y $\vec{L}$, el vector dirección de la luz virtual, que es igual al vector dirección para una luz virtual no-local, y calculándose como el vector que van entre la posición de la luz virtual y el punto, para el caso de ser una luz virtual local (Figura 2.4).

Si $\vec{N}$ y $\vec{L}$ están normalizados, el modelo puede ser representado por:

$$I = \vec{N} \cdot \vec{L}$$

Definición 3 *un modelo de iluminación especular asocia un valor real I, entre -1 y 1, a cada punto del objeto para una fuente de luz según la expresión:*

$$I = cos^n\alpha$$

donde α es el ángulo que forma $\vec{R}$, el vector simétrico a $\vec{L}$ respecto a $\vec{N}$, con $\vec{V}$, el vector dirección del observador. El vector $\vec{L}$ se calcula de la misma forma que se ha expuesto para el modelo difuso. El vector $\vec{V}$ se calcula como la diferencia entre la posición del observador y el punto. El coeficiente n modela el brillo del objeto (Figura 2.4).

Cada luz virtual tiene asociada un modelo de iluminación. Por ello, se tienen luces virtuales difusas y luces virtuales especulares. Ésta es otra diferencia respecto a las luces normales. Como se puede ver, hay una descomposición de los elementos del modelo de iluminación simple, que se compone de las reflexiones Lambertiana y de Phong. Debido a que nuestro interés se centra en los cambios de visibilidad o tono del color, la intensidad de la luz virtual y los coeficientes de reflexión pueden ser simplificados. Se considera que valen 1. La principal diferencia entre las luces virtuales difusas y especulares es que el

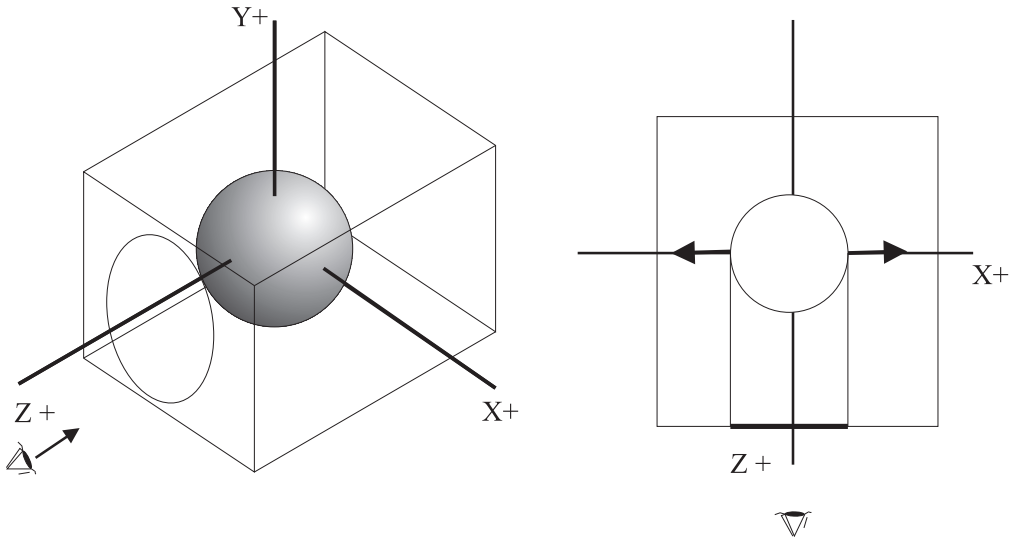


Figura 2.5: Los trazos externos son las siluetas del objeto.

efecto de las luces virtuales difusas es independiente de la posición del observador, mientras que en el caso de las luces virtuales especulares, los trazos son seleccionados dependiendo de la posición de la luz virtual, la orientación del objeto y la posición del observador.

Ahora vamos a definir los trazos:

Definición 4 *dado un modelo geométrico 3D y una luz virtual, un elemento geométrico es un trazo externo si para todo punto de dicho elemento geométrico, el producto escalar del vector normal $\vec{N}$ en dicho punto y el vector dirección $\vec{L}$ de la luz virtual es igual a 0.*

Esta definición se puede ver gráficamente en la Figura 2.5. Se usa una luz virtual difusa colocada en la misma posición del observador. La selección de los trazos internos es más difícil, ya que, no sólo se basa en terminos de la orientación, sino también en la reflexión calculada. En este caso, se puede asociar cualquier modelo de iluminación a cada luz virtual.

Definición 5 *dado un modelo geométrico 3D, una luz virtual, y dos número reales IC y RC, un trazo interno es un elemento geométrico que satisface las siguientes condiciones:*

(a) El ángulo que forman el vector normal y la luz virtual es igual al valor IC.

(b) La reflexión, obtenida usando el modelo de iluminación, es igual al valor RC.

La condición (a) es semejante a la aplicada para obtener los trazos externos. La condición (b) permite seleccionar aquella situación que tiene la configuración deseada por el usuario. Los objetos tienen un conjunto de valores para

cada luz virtual definida. Por ello, no sólo se tienen diferentes tipos de luces sino también diferentes condiciones para cada una. Ésto hace que el método se muy flexible en la obtención de los trazos dónde y cuando el usuario quiera.

Como cada tipo de luz virtual tiene asociada un modelos de iluminación, en ciertos casos es necesario combinar los efectos de las luces virtuales. Ésto se implementa usando una condición final que es aplicada a los resultados individuales. Esta condición es una secuencia de operaciones lógicas que relaciona los resultados de un elemento geométrico para todas sus luces virtuales. Por ejemplo, dadas dos luces virtuales, un elemento geométrico será finalmente un trazo si el mismo es un trazo para la luz virtual_1 **O** para la luz virtual_2, o cuando lo sea para la luz virtual_1 **Y** la luz virtual_2, etc.

2.3.3. Obtención de los trazos

Los trazos son seleccionados usando el modelos de luces virtuales. Debido a que se pueden producir intersecciones entre los trazos seleccionados, es necesario obtener la relación de posición entre los mismos. Cuando se produce una intersección entre trazos, en dicho punto hay un cambio de visibilidad: una parte del objeto oculta otra parte. También pueden haber trazos externos que están totalmente incluidos dentro del objeto. En ilustración y animación clásica la forma, el "carácter" del trazo depende de su disposición y relación con otros trazos: un trazo externo se dibuja normalmente más ancho que otro interno. El poder disponer de esta posibilidad de representación implica el tener que calcular las intersecciones entre trazos.

Los trazos internos son interiores a la silueta del objeto. Si un elemento geométrico es clasificado como trazo externo e interno, se considera que será un trazo externo. Una vez los elementos geométricos son seleccionados, se deben formar cadenas. Aunque es posible visualizar los elementos geométricos individualemente, la formacion de cadenas añade la posibilidad de tratarlos como un elemento único con sus propios atributos. Ésto es, cuando se visualiza una cadena es posible tener en cuenta el estado anterior y el siguiente para un elemento geométrico, y tomar uno u otro dependiendo del contexto, como por ejemplo hacer la línea más ancha, etc.

2.4. Animación

Una vez se han definido el modelo geométrico y los métodos para poder obtener los trazos, componentes que hacen referencia a la visualización, es necesario definir un método o esquema que nos permita crear una animación con dicho modelo. Además, el método de animación debe ser:

- Independiente de la representación.
- Fácil de utilizar.
- Intuitivo.

- Representación simple.
- Expresivo (permitir gran variedad de deformaciones).

De todas estas condiciones, nos resulta de especial interés la última, ya que el campo al cual se orienta el modelo presentado en esta tesis es el de la animación bidimensional clásica. En la misma, es normal el uso de grandes deformaciones, y deformaciones que no son naturales. Como se comentó en la introducción (Pág. 39), la Deformación Libre de Forma es el método que se usa generalmente, aunque presenta el inconveniente de tener una representación costosa. Por este motivo hemos decidido usar la Transformación No-Lineal, debido a que dicho método presenta todas las ventajas enumeradas, modificándola para que tenga control local.

2.4.1. Transformaciones No-Lineales Extendidas

La Transformación No-Lineal es una variación de las transformaciones geométricas, traslaciones, rotaciones, escalados, etc, en la que se modifica la propia transformación, dependiendo de la posición. A la función que relaciona posición y transformación la llamaremos *función de control*. Se puede entender que una transformación geométrica es una transformación no-lineal, cuya función es constante en todo el rango de definición del objeto (de ahora en adelante

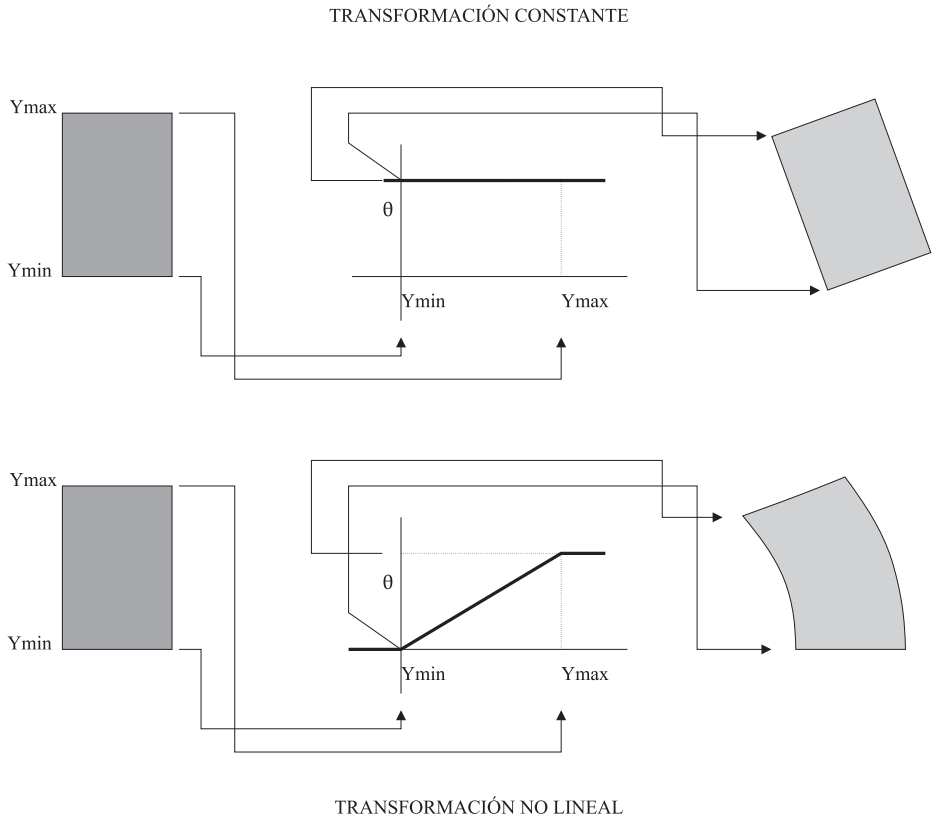


Figura 2.6: Transformación constante y no-lineal.

las llamaremos transformaciones constantes). Es decir, en una transformación geométrica constante, para todo punto (x, y, z) se le aplica la transformación constante T, obteniendo como resultado el punto $(x', y', z') = T(x, y, z)$. En una transformación no-lineal, la función T que se aplica a los puntos es variable y depende del propio punto, ésto es, $(x', y', z') = T_{(x,y,z)}$. Como se puede ver en la Figura 2.6, la función de control de la transformación geométrica es constante, mientras que para la transformación no-lineal es una recta. Para cada transformación se elige un eje de selección. El eje de selección indica que coordenada x, y, o z, del punto, se tomará como variable independiente de la función de control (Figura 2.7).

Barr define las siguientes TNLes:

- Afilar (*Tapering*)
  El afilado consiste en una modificación de un escalado (Figura 2.8, fila 2).
- Torcer (*Twisting*)
  El torcimiento es una derivación de una rotación (Figura 2.8, fila 3).
- Doblar (*Bending*)
  Un doblamiento a lo largo de un eje de selección consiste en una transformación compuesta. En una región se aplica un doblamiento y fuera de dicha región se aplica una rotación y una traslación (Figura 2.8, fila 4).

Mientras que las transformaciones definidas por Barr permiten realizar ciertas modificaciones que resultan interesantes, éstas no poseen suficiente generalidad. Por ello se decide, a partir del esquema presentado, realizar una generalización de las mismas que permita una mayor flexibilidad en la capacidad de aplicación de las deformaciones, así como un tratamiento más completo en la forma de componer dichas transformaciones.

En la generalización que se propone de la TNL, se parte de que se puede aplicar cualquier transformación geométrica afín. Se utiliza también un eje de selección con la misma funcionalidad que se ha comentado hasta ahora.

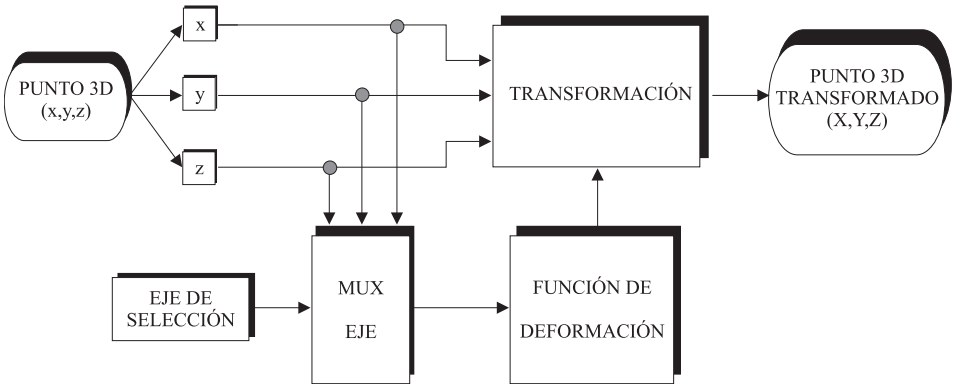


Figura 2.7: Diagrama de una Transformación No-lineal con eje de selección.

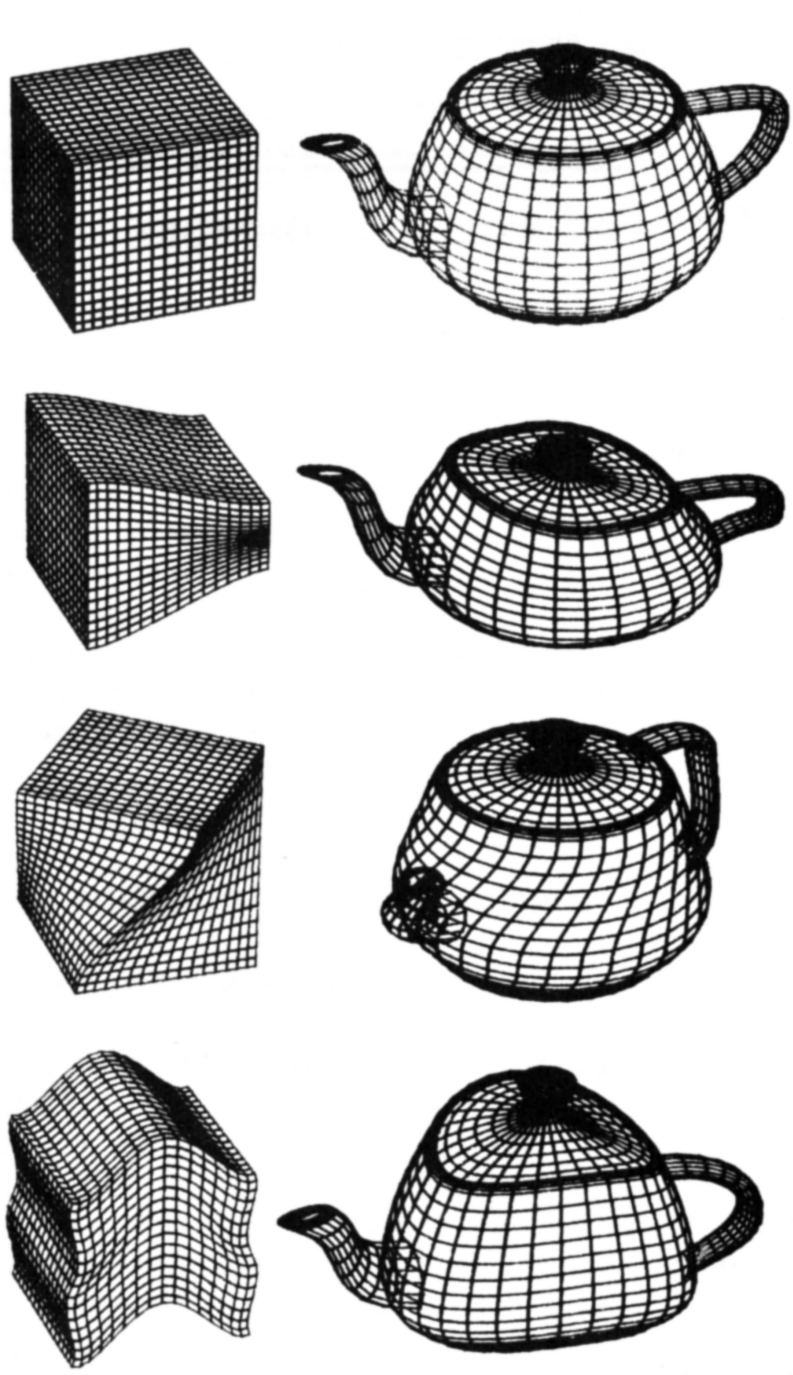


Figura 2.8: Transformaciones No-Lineales de Barr.

Para poder alcanzar un mayor grado de flexibilidad en las deformaciones, se crea también lo que denominamos *eje de aplicación*, cuya función es decidir sobre que coordenada x, y o z, se va a aplicar la transformación (Figura 2.9). En el caso de las transformaciones de Barr, las coordenadas sobre las que se va a aplicar la transformación están fijadas. Como se puede apreciar, se está

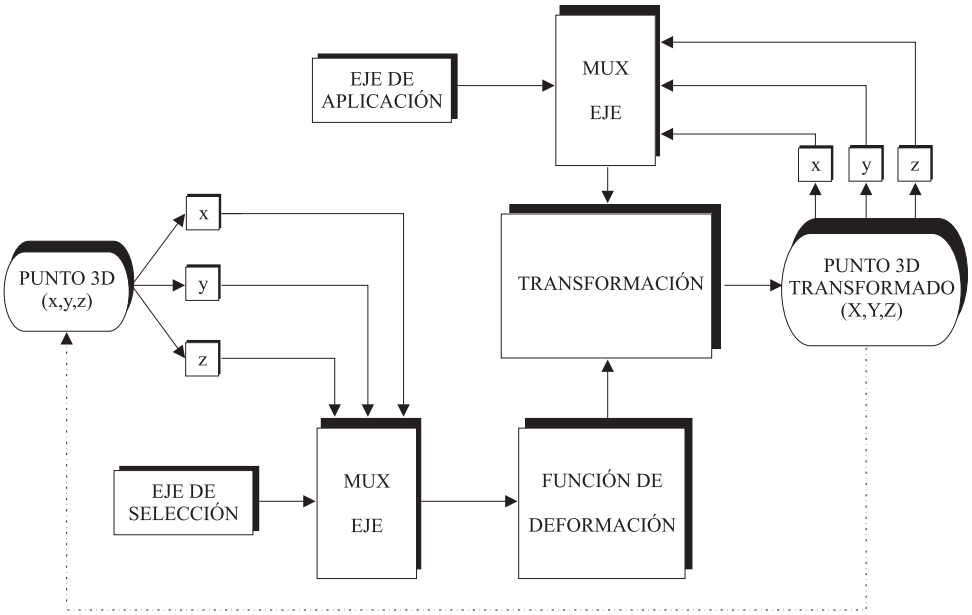


Figura 2.9: Diagrama de una Transformación No-lineal Extendida con eje de selección y eje de aplicación.

generalizando el mecanismo de Barr, en el sentido de que una transformación de Barr puede equivaler a dos transformaciones de las propuestas, aplicándolas con el mismo eje de selección y variando el de aplicación. Por ejemplo, dado el eje de selección z y aplicando un afilado, para Barr quedaría como $(x', y', z') = (rx, ry, z)$, con $r = F(z)$, mientras que con nuestra disposición quedaría como:

Eje de selección z

1)	$(x', y', z') = (rx, y, z)$	$r = f(z)$	Eje de aplicación x
2)	$(x'', y'', z'') = (x', ry', z')$	$r = f(z')$	Eje de aplicación y

El resultado final es el mismo, pero al incluir el eje de aplicación, se pueden crear transformaciones que no son posibles con el esquema de Barr.

Una *función de control* es aquella que define como va a variar el parámetro que controla la transformación no-lineal, en función del valor de una coordenada, que depende del eje de selección (Figura 2.6). Esta función se define en el rango de un valor mínimo y otro máximo que se suelen corresponder con los valores mínimo y máximo del conjunto de vértices para la coordenada indicada por el eje de selección. Lo importante de la función de control es que determina como se va a realizar el cambio de la variable que determina la transformación, por ejemplo, como va a variar el ángulo en una rotación. La forma de la función y los valores que pueden obtenerse son definidos a priori por el usuario, pero el rango en el cual se define puede ser cambiado dinámicamente. Este cambio permite definir la zona en la cual se desea que se aplique la transformación. Fuera de la zona se pueden tomar los valores constantes de los extremos, o permitir que la definición fuera de los extremos sea válida. Un ejemplo de como se usan las funciones de control para una rotación se puede ver en la Figura 2.6.

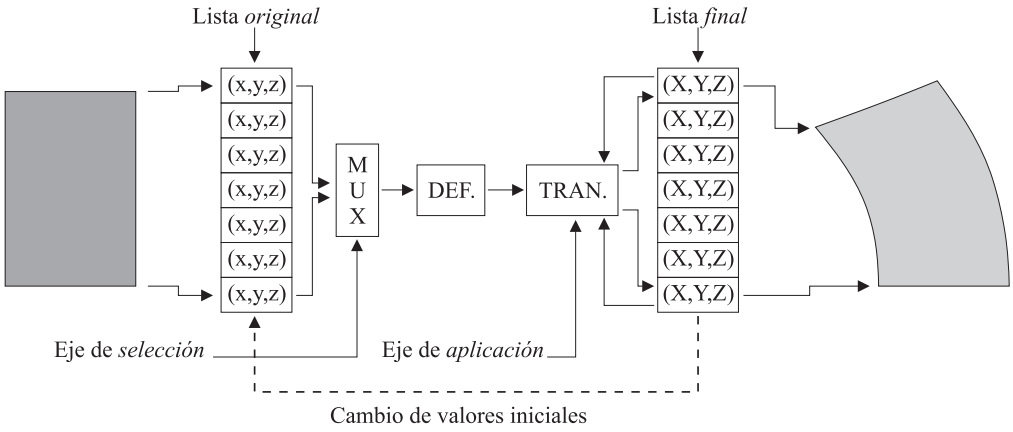


Figura 2.10: Mecanismo de lista inicial y lista final.

Como se puede observar, una vez se define una función de control, los valores que se obtienen de dicha función están relacionados con el valor de la coordenada, dependiendo la misma del eje de selección. Por tanto, se tendrá un rango de valores que vendrá determinado por los valores de la función para el valor mínimo y máximo de la coordenada. En la definición de la función de control, se debe conocer cual va a ser el rango de aplicación de la función, así como la forma en la cual se va a aplicar en función de la posición. Una vez se aplique la transformación, los valores de los vértices cambiarán de forma acorde a las deformación que se le aplica. Si se quiere aplicar una nueva transformación, es necesario que la función de control tenga en cuenta que los valores extremos han cambiado. En general, ésto no se conoce con anterioridad a la aplicación de las transformaciones, lo que nos lleva a buscar otro mecanismo que permita hacer independientes las transformaciones del momento en el cual son aplicadas. Una posible solución es la normalización de las funciones de control y, por tanto, una normalización de la geometría del objeto. Esta posibilidad implica que se tiene que aplicar una renormalización de la geometría de objeto cada vez que se aplica una transformación. Pero incluso esta renormalización puede llevar a resultados no intuitivos en la definición de las funciones de control.

La solución propuesta es utilizar los valores iniciales de los vértices de los objetos como valores para la definición de las transformaciones. Ésto es, el objeto inicial, cuando aún no se ha aplicado ningún tipo de transformación es utilizado para aplicar las transformaciones. Se puede decir que se parte de un objeto maestro a partir del cual se obtienen las sucesivas versiones del objeto transformado. Para implementar este mecanismo se utilizan dos listas de vértices: una lista que llamaremos *inicial* y otra que llamaremos *final* (Figura 2.10). En la primera se introducen los valores iniciales de los vértices, que servirán para obtener el valor, dependiente del eje de selección, que se utilizará con la función de control. En la lista final se encuentran los vértices transformados (la primera vez coincide con el contenido de la lista inicial), a los cuales se aplicará la transformación. Los vértices de la lista final corresponden a las versiones transformadas de los vértices que se encuentran en la misma posición en la

lista inicial. Esta forma de operar permite la aplicación de sucesivas transformaciones, obteniendo resultados que son intuitivos. Ya que las funciones son definidas dependiendo de un valor mínimo y un valor máximo de un parámetro (que se pueden ajustar dinámicamente dependiendo de ciertos atributos de control), mientras la lista inicial no cambie, dichos valores no tienen que ser modificados.

No obstante, existen situaciones en las que es necesario obtener los valores para las funciones de control de los vértices ya transformados, ésto ocurre, por ejemplo, cuando se combinan transformaciones constantes con transformaciones no-lineales. El mecanismo de dos listas permite dar una solución sencilla a este problema: simplemente hay que copiar el contenido de la lista final a la lista inicial, y a apartir de dicho momento utilizar los nuevos valores para definir las transformaciones. En este caso, puesto que se tienen nuevas coordenadas iniciales, habrá que recalcular los valores máximo y mínimo, y reajustar las funciones de control. Este mecanismo permite concatenar transformaciones, ya sean constantes o no-lineales, de forma muy sencilla y bajo control del usuario. También debe estar supervisado por el usuario el efecto final de las transformaciones, controlando que no se produzcan variaciones en la topología del modelo, como autointersecciones, que pueden aparecer en giros muy exagerados.

2.4.2. Transformaciones No-Lineales Extendidas Jerárquicas

La versión de las Transformaciones No-Lineales Extendidas (TNLE) que se ha comentado hasta el momento permite realizar transformaciones a objetos obteniendo resultados muy interesantes. No obstante, el método está limitado a objetos simples, es decir, no se puede usar en objetos compuestos de varias partes, si se desea que cada objeto sufra un tipo de transformación distinto. Si, además, existe una estructura jerárquica, las transformaciones se deben poder aplicar siguiendo dicha jerarquía. Para el caso de transformaciones constantes, no existe ningún problema y se han desarrollado soluciones como por ejemplo el uso de una pila de transformaciones (PHIGS, OpenGL). En el caso de las transformaciones no-lineales, y con la posibilidad de ser combinadas con transformaciones constantes, es necesario resolver dos problemas. Por ello se propone el método de las Transformaciones No-lineales Extendidas Jerárquicas (TNLEJ).

El primer problema consiste en encontrar la forma en la cual se aplica la transformación heredada de un objeto padre a un objeto hijo, ya que las deformaciones, como se ha comentado, se definen según las coordenadas iniciales de los objetos. Por otra parte, está el problema de la extensión de las funciones de control. Se pueden dar diversas situaciones en cuanto a la disposición de las coordenadas del objeto hijo con respecto a la definición de la función de control del objeto padre: que esté totalmente contenido, que esté parcialmente contenido o que esté fuera del rango de aplicación definido para el padre (Figura 2.11).

Para explicar la solución propuesta se parte de un ejemplo sencillo en el

cual se simula una casa (Figura 2.12), en la que existe una jerarquía, en cuanto que el tejado, ojos y boca dependen del cuerpo de la casa (Figura 2.13). Los movimientos de dicho cuerpo se deben transmitir al resto de los componentes, pero, además, cada uno puede tener definidas sus propias transformaciones. En el ejemplo que se muestra en la Figura 2.12, el cuerpo de la casa se modifica de distintas maneras, mientras que el tejado sigue el movimiento del mismo sin sufrir ninguna transformación propia, al igual que la boca. Los ojos se abomban hacia afuera.

El primer problema se resuelve recolocando los objetos dependientes dentro del rango de las funciones de control de los objetos de los cuales dependen. Por ejemplo, ya que la casa se mueve, los ojos deben seguir el movimiento de la casa. En este caso, el rango de definición de los ojos está completamente contenido en el rango de definición de la casa. Para poder incluir un objeto de forma jerárquica dentro de otro se recurre al uso de transformaciones constantes, que al aplicarlas, hacen que los valores mínimo y máximo que definen el rango de aplicación cambien. Estas transformaciones se deben definir en el objeto padre. Para el caso del ejemplo, la transformación que la casa induce en los ojos, es una traslación de los mismos a la posición oportuna mediante una transformación constante, que, además, debe cambiar las coordenadas de la lista inicial de los ojos.

Como se ha comentado, el segundo problema es el rango de validez de la funciones de control. Se ha visto la solución en el caso de que el objeto dependiente esté totalmente contenido dentro del rango del objeto del que depende. Puede ocurrir que el rango de la función de control del objeto hijo esté parcialmente contenida o excluida del rango de la del padre. Ésto sucede, por ejemplo, en el caso del el tejado, el cual está totalmente fuera del rango del cuerpo de la casa, pero se quiere que siga los movimientos de la misma. El mecanismo propuesto para solucionar dicho problema consiste en poder

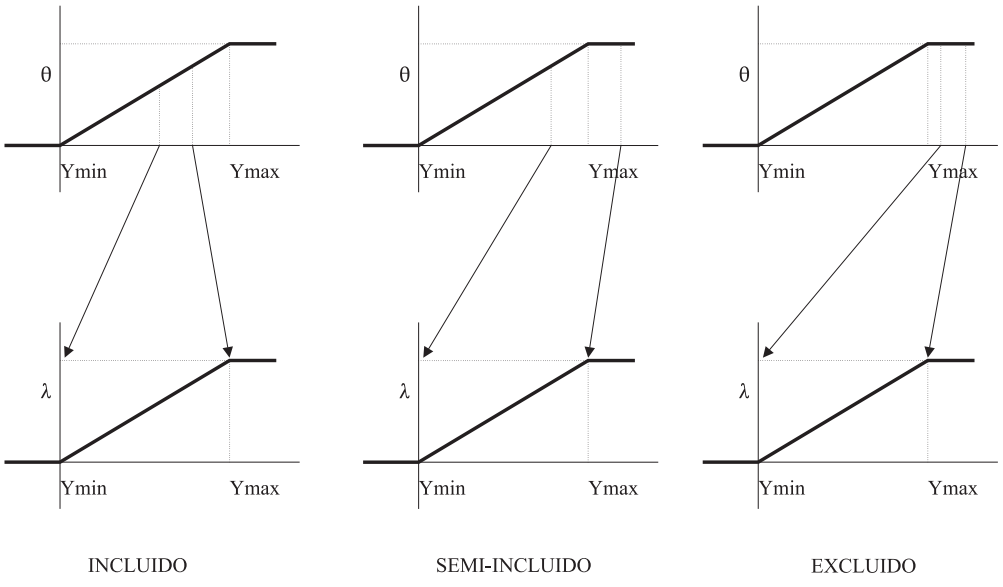


Figura 2.11: Posibilidades de relación entre funciones de control en una jerarquía.

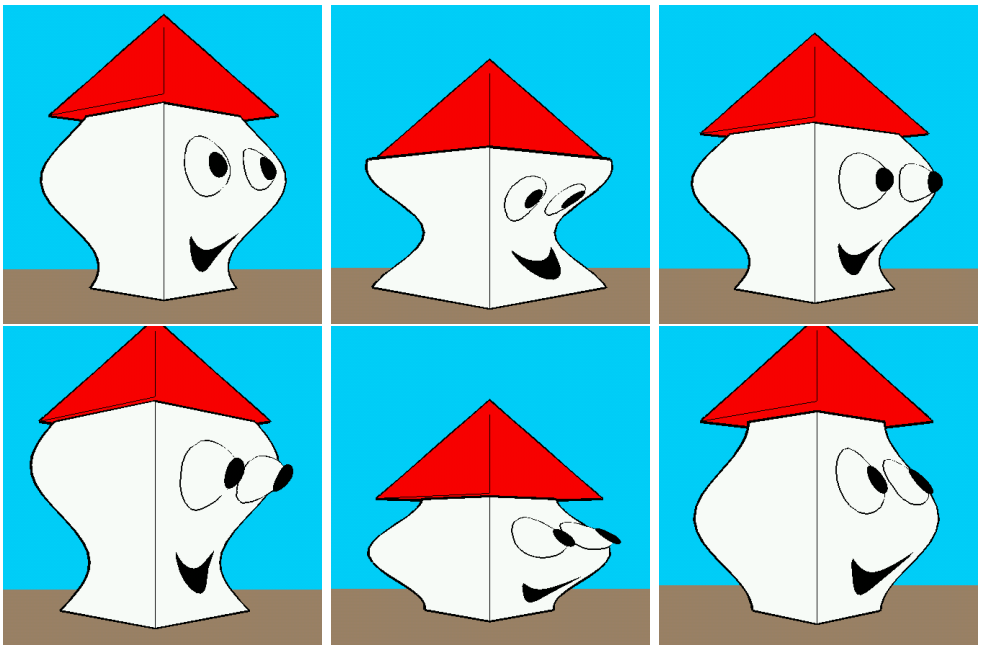


Figura 2.12: Ejemplo de Transformación No-Lineal Extendidad Jerárquica.

controlar la extensión de las funciones de control (Figura 2.14). El control de la forma en que se extiende la función de control del padre, es realizado mediante atributos del objeto hijo.

Las posibilidades que aparecen son:

(a) Si la función define valores constantes en los extremos, calcular la pendiente de la curva en el punto extremo y extender la función de forma

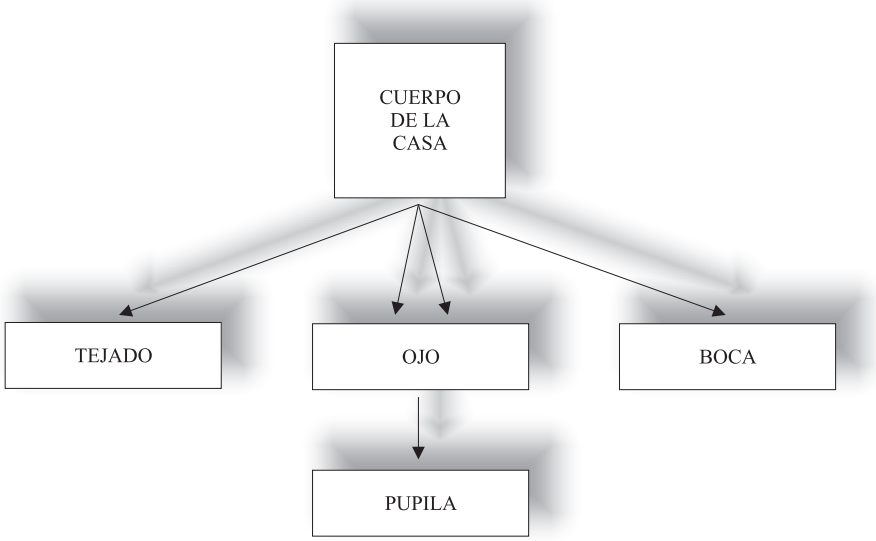


Figura 2.13: Jerarquía de movimientos.

lineal.

(b) Si la función está definida fuera de los límites, realizar una extensión de los mismos.

Esta operación de extensión se puede aplicar individualmente a cada una de las transformaciones del objeto padre, con lo cual se consigue una mayor flexibilidad, o se puede definir de una forma mas global a costa de un menor control. Las jerarquías son definidas por el usuario en función de sus necesidades de diseño.

2.4.3. Animación

Mediante las TNLEJ se pueden obtener deformaciones en los objetos, pero estas deformaciones son estáticas, en el sentido de que para cada configuración de TNLEJ, se obtiene una imagen. Para producir una animación hay que obtener una secuencia de transformaciones. El uso de Transformaciones No-Lineales Extendidas Jerárquicas permite una fácil extensión para poder crear dichas secuencias.

Dado que lo que controla las transformaciones son las funciones de control, el mecanismo desarrollado para obtener una secuencia de deformaciones consiste en hacer que dichas funciones de control no sean estáticas sino que puedan evolucionar en el tiempo (Figura 2.15). Para ello, las funciones de control se definen mediante parámetros que varían en el tiempo. Estas variaciones son representadas por funciones.

Por ejemplo, dada una curva B-spline definida por cuatro puntos, la posición

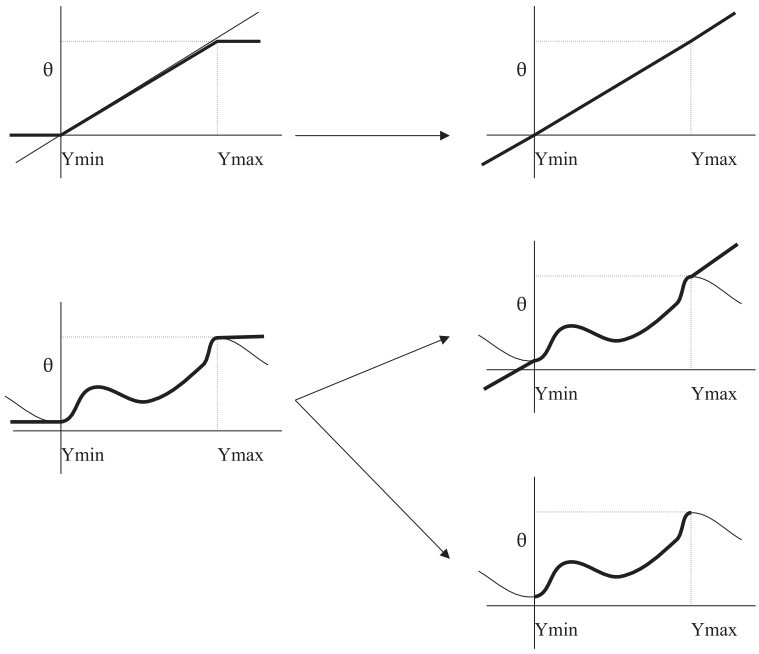


Figura 2.14: Posibilidades de extensión para las funciones de control.

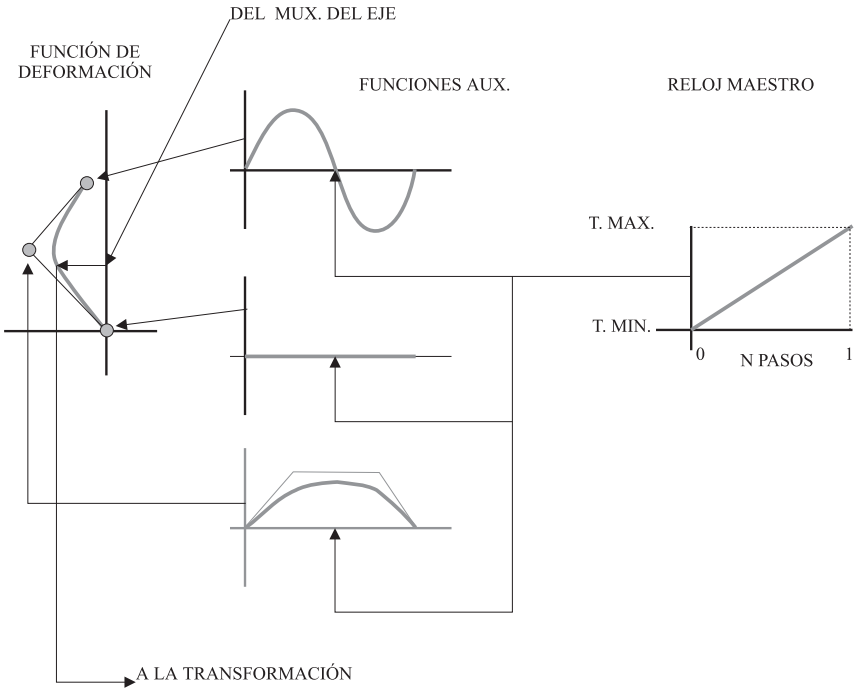


Figura 2.15: Dependencia funcional-temporal de la función de control.

de cada uno de ellos puede ser descrito por otras cuatro funciones, que a su vez pueden ser de cualquier tipo (de hecho, la superficie que representa las posiciones de la función de control no tiene porqué ser una superficie B-spline). La función que controla el primer punto podría ser lineal y la que controla el segundo ser una curva senoidal. Las curvas que permiten obtener la dinámica de los parámetros que definen las funciones de control, dependen de una función que las sincroniza y que hace el papel de reloj maestro.

Las funciones con que se usan son monoevaluadas y bidimensionales, ésto es, del tipo $y = f(x)$. La idea general del desarrollo de las funciones es que las mismas se definen mediante un conjunto de parámetros, los cuales no tienen porqué ser constantes sino variables, y esta variación se puede hacer en relación a otra función, que a su vez pueden tener todos o partes de los parametros de definición dependientes de otras funciones, y así sucesivamente, hasta llegar hasta una función inicial que no depende de otras. Veamos como se definen nuestras funciones.

Definición 6 *una función de control f es una triada ($g(\alpha)$, rango-abcisas, rango-ordenadas), siendo g la función que determina el comportamiento de la deformación, dependiente de α, y rango-abcisas y rango-ordenadas dados por dos valores $Min_{abscisa}$ y $Max_{abscisa}$, y $Min_{ordenada}$ y $Max_{ordenada}$ respectivamente.*

Los rangos vienen determinados por la definición de las TNLEJ, ya que es necesario definir un rango de validez para la variable independiente, así como un rango de aplicación. La referencia aparece en la posibilidad de hacer

que los valores Max y Min del rango de las ordenadas sean dependientes de otras funciones recursivas. Ésto es, $Min_{ordenada} = f_1(s)$ y $Max_{ordenada} = f_2(t)$, siendo f_1 y f_2 funciones recursivas. La otra posibilidad es que los valores de los rangos sean constantes, $Min_{ordenada} = K$ y $Max_{ordenada} = L$, siendo K y L valores contantes. También se permite cuaquier combinación, ésto es, una de los valores del rango es constante y el otro es dependiente. Aunque quizás la característica más importante de las funciones de control es que α pude hacerse que depende de otra función. Por tanto, $\alpha = f(\beta)$, que a su vez es dependiente de otra función, y así sucesivamente. En tal caso, no se definen las abscisas, ya que vendrán impuestas por lar ordenadas de la función de la cual dependa. Se debe imponer una condición de parada la cual se consigue mediante una función básica, que tiene la siguiente definición $f_{basica} = (y = x; 0 - 1, Min_{ordenada} - Max_{ordenada})$. Por defecto $Min_{ordenada} = 0$ y $Max_{ordenada} = 1$ pero se pueden cambiar para ajustarlos a los valores deseados.

Capítulo 3

Implementación con modelos poligonales

En este capítulo se hace una descripción de las soluciones que se han implementado y su relación con la descripción genérica que se ha dado en el capítulo anterior para los distintos elementos del modelo.

3.1. Modelo 3D

Dados los requisitos que se impusieron para el modelo geométrico 3D, el esquema de representación de los objetos que se eligió fue el de una representación de fronteras basado en polígonos planos, más concretamente triángulos, usando la estructura de datos Aristas Aladas (*Winged-Adged*) [21, Págs. 545-546].

En general, los modelos de fronteras basados en polígonos representan al objeto mediante una aproximación, excepto que el propio objeto tenga forma poligonal. La mayor o menor aproximación depende del número de caras, pero se puede aproximar todo lo que se quiera. Normalmente el conjunto de caras se crea a partir de un conjunto de puntos, los cuales se pueden crear a partir de una serie de puntos obtenidos mediante un escáner, o bien trás el muestreo de superficies paramétricas que representen la forma del objeto. A partir de dichos puntos se crea la topología que describe al objeto, ésto es, las aristas y caras. Se dispone la información del modelo mediante su geometría, descrita por los puntos pertenecientes a la superficie del objeto, y la topología, descrita con las aristas y las caras. Mientras que la estructura de datos que almacena los vértices guarda las coordenadas de cada uno de ellos, las estructuras de datos que almacenan las aristas y caras suelen almacenar punteros a vértices y puntero a aristas, respectivamente. Al ser los modelos poligonales una discretización del objeto a representar, permiten que el nivel de aproximación puede ser controlado, dependiendo de las necesidades del usuario: en un momento determinado, con un modelo muy sencillo, con pocos polígonos, permite una rápida visualización y prototipado; una vez encontrada una solución, el modelo se puede refinar para obtener una visualización final con un alto nivel de

detalle. Esta discretización también puede representar una ventaja en cuanto a que, aunque se aplica a un número mayor de elementos, la operación suele ser más sencilla.

Actualmente la representación de los objetos mediante caras triangulares es muy común. De hecho se considera como una representación de "bajo nivel", que es usada como representación objetivo para otro tipo de representaciones de más "alto nivel", como podrían ser la representación basada en superficies paramétricas, en superficies implícitas, etc. Ésto es así, debido a su sencillez, lo cual ha permitido que se desarrollen numerosas técnicas y métodos que trabajan con polígonos, y, especialmente, con triángulos, ya que los mismos presentan, a su vez, ciertas ventajas, como que es la figura poligonal 2D más sencilla que garantiza que es plana. Entre las técnicas que se han desarrollado se encuentran el suavizado de Gouraud y Phong, relleno por lista de aristas activas, báfer-z [21, Págs. 668-672], etc. Para muchos de estos métodos, la condición de ser plano es necesaria para su funcionamiento. Estos métodos han sido implementados en hardware, el cual permite realizar una aceleración de la visualización. Este hardware se encuentra actualmente incluso a nivel de PC. Por último, también se han desarrollado funciontecas *Library* para las cuales los triángulos son considerados como primitivas gráficas básicas, no garantizando un funcionamiento correcto si el polígono no es plano. Un ejemplo es OpenGL.

Otra característica es que la evaluación de las normales es muy sencilla, garantizándose, además, que dicha normal representa exactamente al plano y que no es una aproximación. Esta sencillez contrasta con las de otros tipos de modelos de fronteras, como por ejemplo los basados en superficies paramétricas, en los que hay que recurrir a derivadas, y que en ciertos casos degenerados (Pag. 31), pueden presentar problemas. Esto facilita la evaluación de otros elementos del modelo, ya que los mismos están basados en el uso de la normal de las caras.

El modelo poligonal presenta el inconveniente de que, en general, y excepto que el propio objeto a representar tenga forma poligonal, una buena representación necesita un gran número de caras. Como se ha dicho, este número depende de lo buena que se quiera que sea la aproximación. En la Figura 3.1 se puede ver un ejemplo en el que la figura no está muy bien definida, pero incluso ya el número de caras es grande. Además, un modelo poligonal es una representación discreta de un objeto. Ésto produce efectos de alias *Aliasing* en las distintas operaciones que se aplican. Por ejemplo, si se desea determinar la poligonal que representa la silueta de un objeto, nos podemos encontrar con que haya una cara perpendicular al plano de proyección que une una cara visible y otra invisible, con lo que resulta dificultoso definir claramente a la cara perpendicular como visible o como invisible. En cualquier caso, la solución sólo será una aproximación y, en este caso, bastante mala (Figura 3.2). El problema más grave que se puede presentar con el modelo poligonal, cuando se le aplican transformaciones, es la posibilidad de que se cambie la topología del objeto originalmente representado. Esto puede ocurrir, por ejemplo, si dado un cubo, se le aplica a los vertices de la cara superior un giro de 180 grados. En este caso ocurren autointersecciones, y el cubo deja de ser un cubo. Este

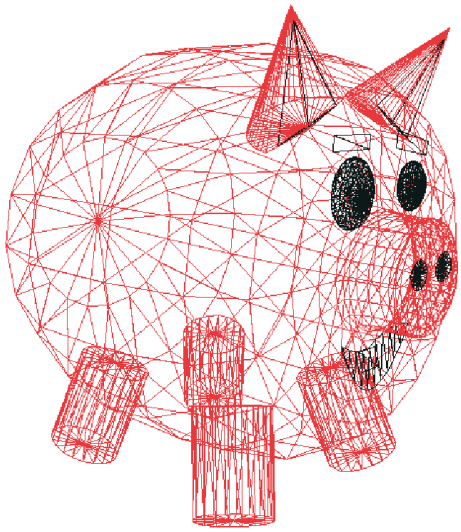


Figura 3.1: Un modelo poligonal con un número bajo de caras.

problema no puede ser resuelto, debiendo ser controlado por el usuario.

El principal motivo que nos han llevado a la elección del modelo poligonal basado en triángulos es que permite una fácil detección y clasificación de los elementos geométricos que conforman los trazos, ya que en este caso, son las aristas. El uso de aristas como elementos definidores de los trazos ha hecho que se tenga que hacer una nueva definición de las *Luces Virtuales*, adaptándola a una representación no contínua del objeto. Para la animación no existe ningún problema ya que el método elegido es independiente de la representación.

Ésto no quiere decir que el modelo esté limitado, sino que en su concretización, impone ciertas limitaciones. Su generalidad permite que en el futuro, frente a las limitaciones que se presenten, se puedan estudiar otros esquemas de representación, como podría ser el uso de superficies paramétricas, que su-

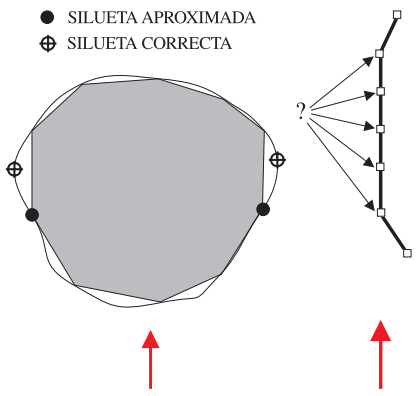


Figura 3.2: Problema en la selección de las aristas que forman los trazos externos.

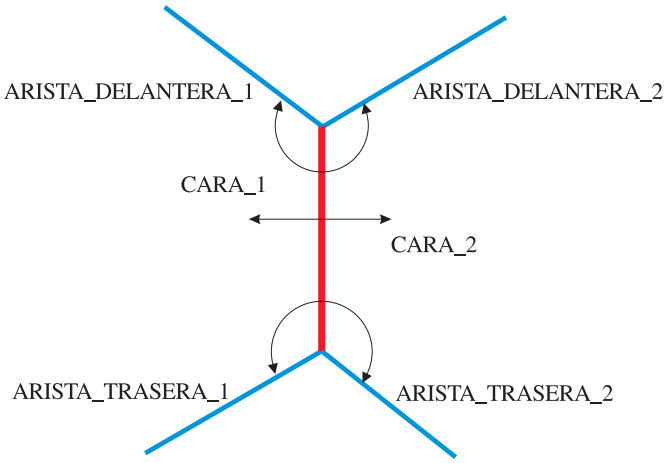


Figura 3.3: Estructura de aristas aladas.

frirían las deformaciones, a partir de las cuales se obtendrían, en un segundo paso, las versiones poligonales, o incluso utilizar las superficies paramétricas para la obtención de los trazos, usando alguno de los métodos desarrollados [17]. También se podrían usar superficies implícitas, aunque esto implicaría un cambio en la forma de aplicar las deformaciones.

3.1.1. Creación de objetos

De las distintas posibilidades que existen para representar un modelo de fronteras basado en polígonos, se seleccionó el de aristas aladas. En la Figura 3.3 se observa cual es la disposición en la que se basa, y el porqué del nombre. En la Figura 3.4 se puede ver la descripción completa de las estructuras de datos utilizadas. Para la implementación se han usado la estructura de objetos y herencia que se muestra en la Figura 3.5.

El motivo de usar el esquema de representación de aristas aladas es que facilita ciertas operaciones de búsqueda. En particular, permite un rápido seguimiento de una serie de aristas, operación que se usa para formar las cadenas de aristas que formarán los trazos. También facilita la clasificación de las aristas, dependiendo del tipo de caras que posea, ya que dispone de punteros a las caras que comparten una arista (como se ha comentado con anterioridad, sólo se permiten objetos 2-variedad (*Two-manifold*)).

Para la generación de los distintos objetos se disponen de distintas posibilidades:

- Crear superficies tipo malla.
  - Introduciendo los puntos manualmente.
  - Obteniendo los puntos aleatoriamente (superficie aleatoria).
  - Introduciendo los puntos de control de una superficie B-spline, a partir de la cual se obtiene, tras ser muestreada, la malla.

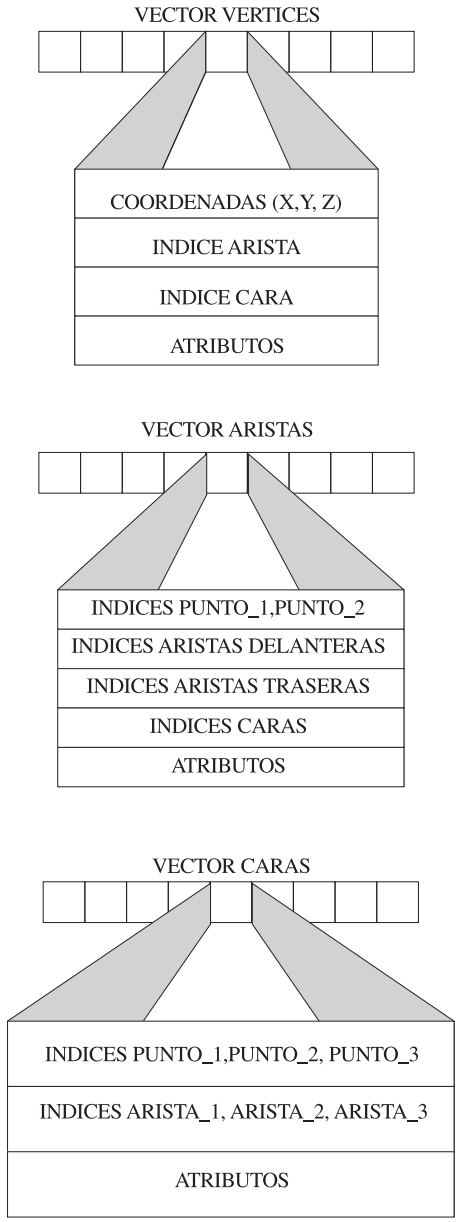


Figura 3.4: Estructura de datos.

- Crear objetos por revolución.
  - Introducir la poligonal de control manualmente .
  - Introducir los puntos de control de una curva B-spline a partir de la cual se obtiene la curva poligonal.

3.2. Trazos

Antes de ver la particularización del modelo de *Luces Virtuales*, vamos a exponer con más detalle el algoritmo de generación de trazos para modelos

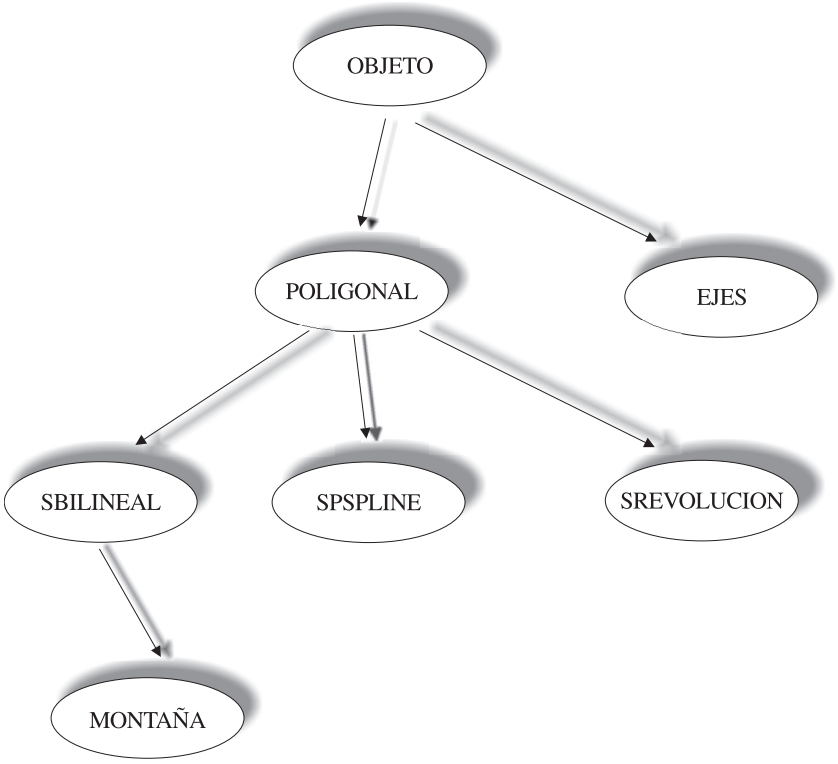


Figura 3.5: Estructura de herencia.

poliédricos. Para estos modelos, los trazos están formados por aristas.

Para todo Objeto **Hacer**
Clasificación y selección de las aristas
Calcular las intersecciones entre aristas

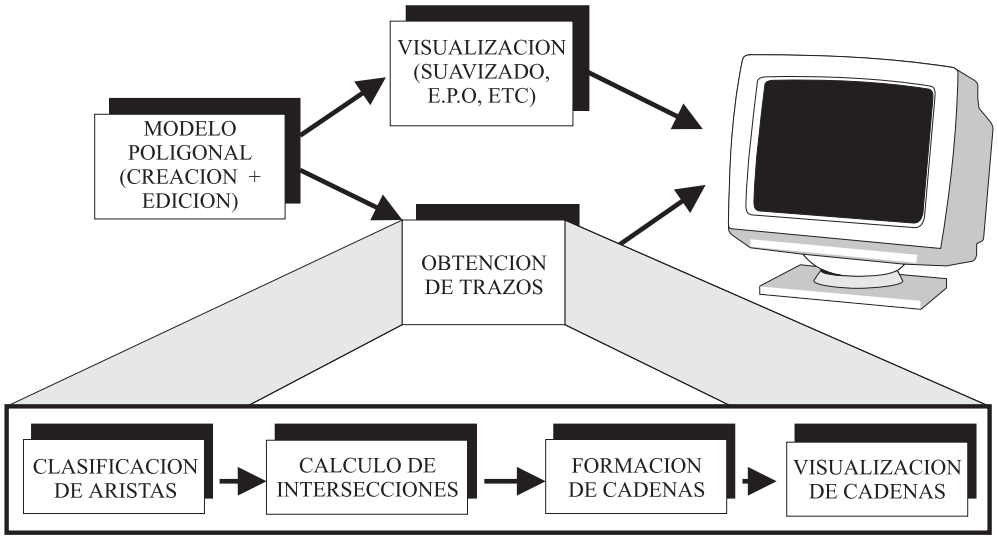


Figura 3.6: Estructura del procedimiento principal.

Formar cadenas de aristas
FinPara
Visualizar las cadenas de aristas

Veamos una descripción más detallada de la implementación de las distintas etapas del algoritmo mostradas en el seudocódigo anterior.

En la primera fase hay que relizar una clasificación y selección de las aristas que serán trazos (de forma indiviual y conjunta cuando formen cadenas). Para relizar esta clasificación los pasos que hay que seguir son los siguientes:

Para todo Luz virtual **Hacer**
Calcular la reflexión de las dos caras
Si cumple las condiciones para la luz virtual **Entonces**
Marcar la arista como trazo
FinSi
FinPara
Operación global

En la Figura 3.6 se puede observar gráficamente las distintas etapas del algortimo.

Dichas etapas se exponen de forma detallada a continuación.

La primera etapa es la encargada de la detección de las aristas que formarán parte de los trazos. Para cada arista del objeto, se calculan los valores de reflexión de cada una de las luces virtuales que tenga asociado el objeto con cada una de las caras de la arista. Las caras pueden clasificarse en *visibles* e *invisibles*, en función de que lo sean o no desde la posición de la luz virtual (la formalización de dicha clasificación se ve en la página 75). Las aristas se clasifican como *normales*, *trazo_visible* y *trazo_invisible*. Una arista se dice que es normal si es compartida por dos caras que o bien son visibles o bien son invisibles. En la implementación actual las aristas normales no pueden formar parte de un trazo. Una arista es parte de un trazo cuando pertenece a la vez a una cara visible y otra invisible. En función de las posiciones relativas que pueden ocupar las dos caras, las aristas denominadas trazo_visible (en adelante sólo trazo), son aquellas cuya cara visible está delante de la invisible; las aristas denominadas trazo_invisible son aquellas en las que la cara invisible está delante de la visible. Las caras también se van a clasificar según posean o no alguna arista trazo o trazo_invisible. Las caras con alguna arista trazo o trazo_invisible se les denomina cara_trazo. Una cara se denomina cara_normal si posee todas las aristas de tipo normal. Para clasificar las aristas y las caras, se van recorriendo las primeras y en función del criterio que se cumpla se les va asignando su correspondiente valor. Una vez que se ha realizado este proceso, se dispone de información sobre qué aristas formarán parte de los trazo y cuales no, y qué caras las aportan.

Una de las luces virtuales será la encargada de permitir la obtención de las siluetas. Normalmente, esta luz virtual es única para todos los objetos, aunque cabe la posibilidad de definir distintas luces virtuales que permitirían obtener otros efectos. Una vez están calculadas las reflexiones, hay que clasificar la

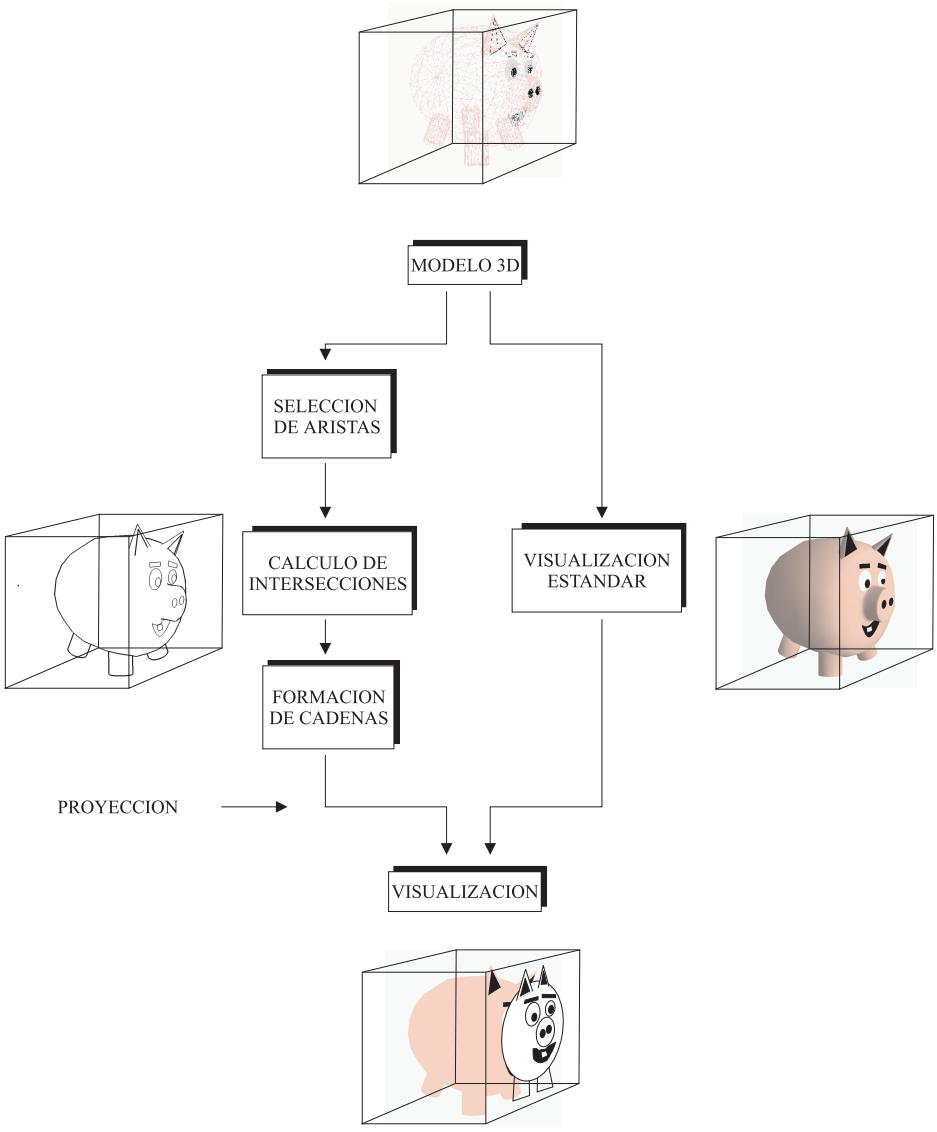


Figura 3.7: Modo 1 de visualización.

arista, la cual podrá ser un trazo o no. Ésto dependerá de que se cumplan las condiciones impuestas para la luz virtual que esté siendo calculada. Al conjunto de resultados parciales obtenidos con todas las luces virtuales se les aplica una operación final, que permite combinarlos usando operaciones lógicas. Una vez se ha aplicado esta etapa, las aristas quedan clasificadas. En la implementación final existen una serie de optimizaciones que permiten acelerar la ejecución del algoritmo. Por ejemplo, se pueden eliminar el tratamiento de las caras traseras. También se ha tenido en cuenta en el cálculo de la expresión lógica que se aplica al final. Aprovechando el recorrido y clasificación de las aristas, se va formando una lista encadenada de aquellas aristas que son clasificadas como trazos. Esta lista permitirá acelerar la formación de cadenas.

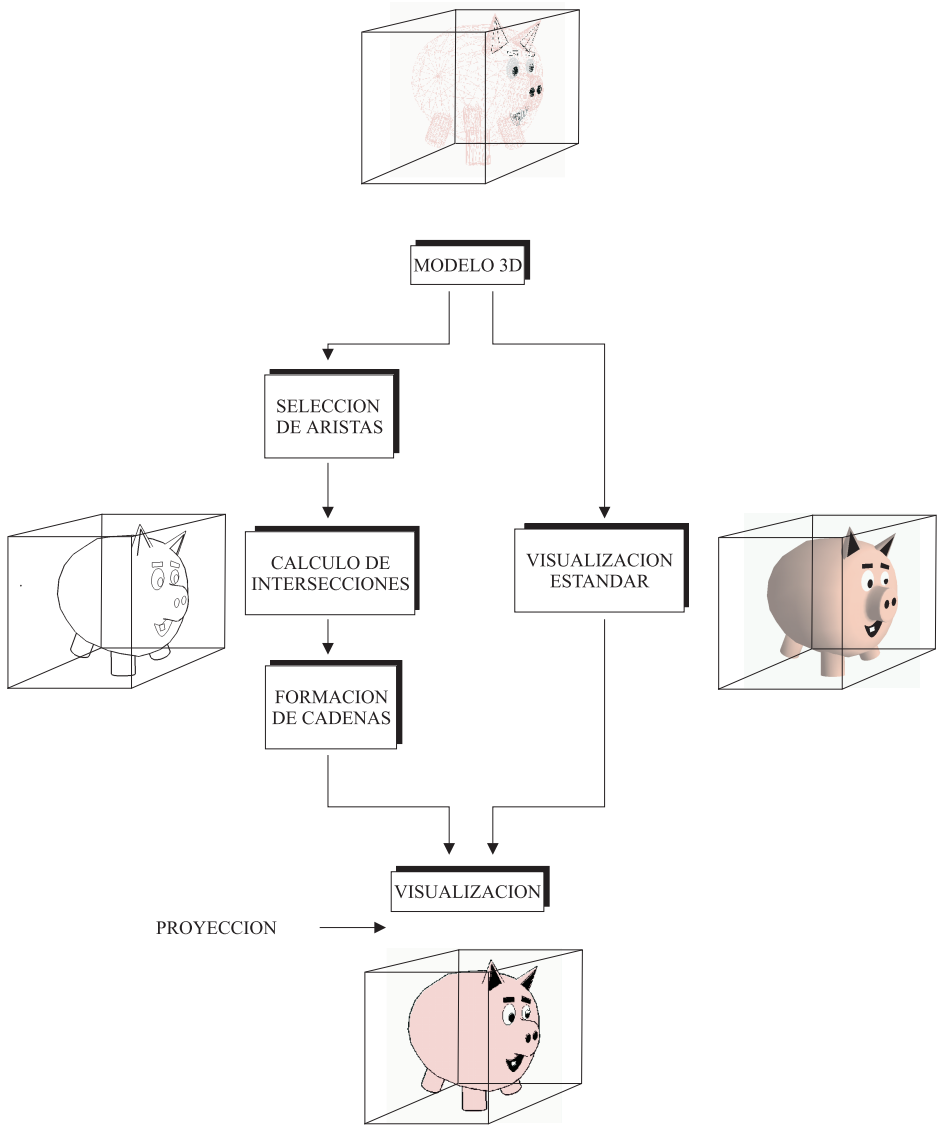


Figura 3.8: Modo 2 de visualización.

En la segunda etapa el algoritmo calcula las intersecciones que se producen en las proyecciones de las caras. Debido a la generalidad, que se impuso al principio del desarrollo de este algoritmo en cuanto las posibilidades de uso, ilustración, animación, dibujo descriptivo, etc, no sólo estábamos interesados en saber si algo quedaba detrás de un objeto, sino en tener información cuantitativa que permitiera una gran flexibilidad en la visualización. Un ejemplo claro de esta necesidad es el hecho de que, en general, tanto en animación como en ilustración, los trazos, tanto internos como externos, que quedan en el interior del objeto, deben ser dibujados con un gosor menor que que los trazos que delimitan la silueta exterior del mismo.

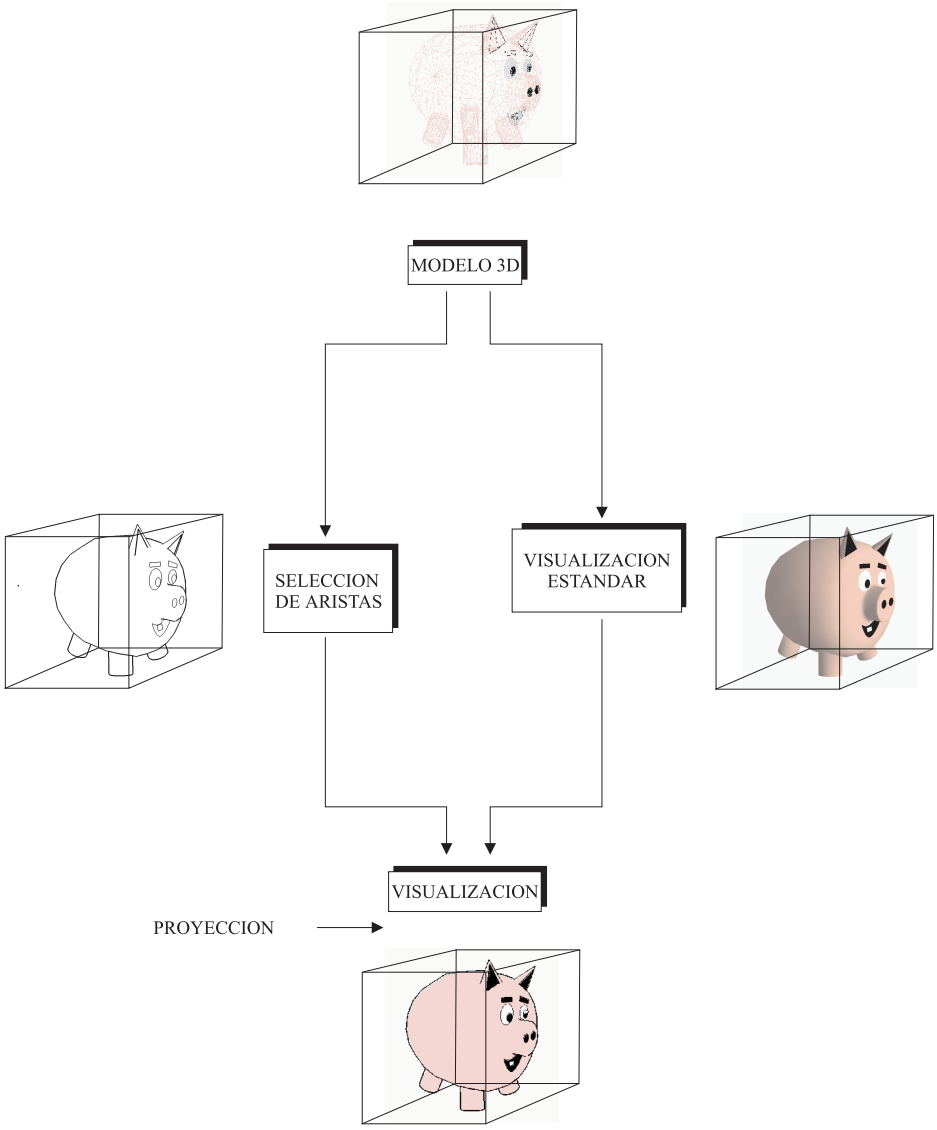


Figura 3.9: Modo 3 de visualización.

La solución aportada se basa en la siguiente idea: encontrar las intersecciones de las aristas es equivalente a encontrar las intersecciones de los bordes de las caras a las que pertenecen. Aunque en principio es algo simple y obvio, la diferencia estriba en el tipo de elemento con el que se opera: en un caso se trabaja con aristas mientras que en el segundo se trabaja con caras. El problema del método, como se verá más adelante, es que es un método de complejidad cuadrática. La gran ventaja que aporta el trabajar con caras reside en que, a diferencia de lo que pasa con las aristas, se puede clasificar la profundidad de un punto aunque no se produzcan intersecciones. Este caso ocurre cuando hay trazos interiores a otros trazos. Usando caras se puede determinar si están o no ocultos los puntos del trazo interior. Ésto se puede resolver también con

trazado de rayos, pero en tal caso es necesario recurrir a algún mecanismo complementario para no producir resultados incorrectos. Otra ventaja que se obtiene al trabajar con caras es que los cálculos se realizan en coordenadas del objeto, lo cual permite obtener cadenas que no estén totalmente dentro del volumen de visión (el modelo debe estar entre los planos delantero y trasero en caso de realizar una proyección de perspectiva; para una proyección paralela no es necesario cumplir estas condiciones).

En la tercera etapa, y con la información que se posee después del cálculo de intersecciones se podrían dibujar las aristas, una a una, sin formar cadenas. En nuestro esquema optamos por la formación de cadenas, cerradas y abiertas. Ésto tiene la ventaja de que al visualizar las cadenas de aristas, los puntos que las componen pueden ser tratados como puntos de control de curvas y trazos texturizadas. Teniendo en cuenta que uno de los campos de interés es la ilustración, y que en la misma la forma local de los trazos puede ser tan importante como la forma global, esta característica resulta de especial interés. También por ésto, se realizan los cálculos en coordenadas del objeto, ya que si no, las cadenas y sus características se verían cortadas en los bordes laterales del volumen de visión. En caso de que coincidan más de un trazo en un vértice, se permite elegir entre escoger primero el trazo exterior o escoger primero el interior.

Para la formación de la cadenas, se parte del primer elemento de la lista encadenada que se ha comentado previamente, y se va formando la cadena obteniendo a partir de dicha arista la siguiente que también haya sido clasificada como trazo. Estas aristas tienen continuidad geométrica, ésto es, comparten un vértice. Aquí se usa de forma intensiva las características de la estructura de datos arista alada. Puede ocurrir que llegados a un vértice, el mismo tenga más de una arista marcada como trazo. En tal caso hay que hacer una selección. Actualmente, la implementación del algoritmo de selección y formación de cadenas es muy sencillo, aunque permite dos estrategias. Dado un punto del cual salen dos o más aristas, se puede optar por elegir el camino más externo o el camino más interno. Aunque simple, ha permitido obtener resultados muy interesantes. La ventaja que presenta el formar cadenas es la de poder tratar a las cadenas como elementos únicos, una curva con sus propios atributos. De otra forma, las aristas serían dibujadas individualmente, cada una con su atributo.

En la última etapa del algoritmo, las cadenas son dibujadas usando los atributos de las aristas de forma global, y teniendo en cuenta que los cambios de visibilidad representan cambios en los atributos. Aquí se va a describir de forma más genérica los distintos modos de visualización que se han implementado, y en un punto posterior, se verán más en detalle, teniendo en cuenta los valores obtenidos por el algoritmo de cálculo de intersecciones. Se han implementado tres modos de visualización, los cuales se diferencian unos de otros en la complejidad y en los objetivos que se desean alcanzar.

El modo más completo busca las aristas que forman la frontera o fronteras entre las partes visibles e invisibles del objeto (Figura 3.7). También detecta la visibilidad entre objetos. El método usado busca las aristas que tienen una

cara visible y otra invisible, o tienen sólo una cara visible. Las operaciones son relizadas en coordenadas de dispositivo normalizado. Las caras son marcadas como visible o invisible dependiendo del signo del vector normal de la misma. Las caras que poseen alguna arista que forma parte de la silueta son a su vez marcadas. Estas caras se usan para obtener parámetros que se utilizan en la visualización de las aristas. Nuestro algoritmo extiende el concepto de invisibilidad cuantitativa (*Quantitative Invisibility*) del método de Appel, en el sentido de que también controla cuantas veces aparece la arista delante de alguna o varias caras. Este concepto puede ser llamado, por similitud, *visibilidad cuantitativa*. El mismo añade un grado más de flexibilidad en la visualización de los trazos. Una vez que se realiza este proceso, las aristas marcadas son unidas formando cadenas. Cuando una cadena es visualizada, es posible tener en cuenta, el estado anterior y posterior de cada arista, y tomar uno u otro dependiendo del contexto, y, por ejemplo, hacer que la línea sea más o menos ancha, etc.

Las cadenas pueden ser visualizadas de diferentes maneras, como puede verse en la Figura 3.10. El método propuesto puede ser utilizado para la eliminación de líneas ocultas. Cuando se usa este método, los objetos, sin contar con los trazos, son visualizados usando OpenGL, activando el test de profundidad. Una vez hecho esto, las siluetas son dibujadas en frente de todos los objetos (con un valor de z igual al del plano delantero). Este método resulta especialmente importante debido a que puede ser la base de los siguientes desarrollos del sistema, como se comentará en la parte de trabajos futuros.

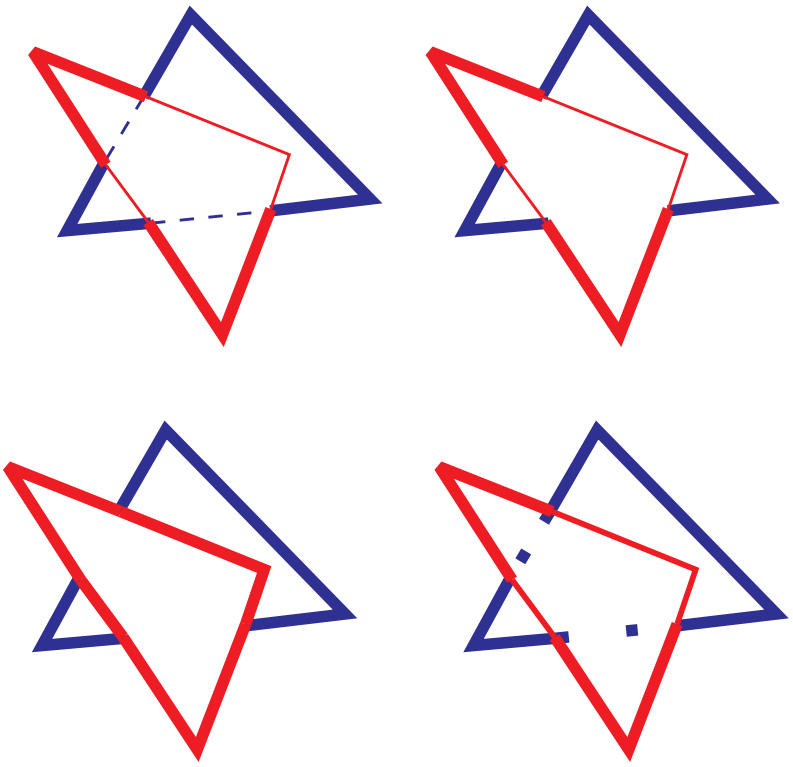


Figura 3.10: Formas de visualizar los trazos.

El segundo modo es una variación del anterior método, en el que las cadenas son transformadas de la misma manera que los objetos (Figura 3.8). El báfer-z se aplica a los objetos y a los trazos. Las caracteristicas visuales de las cadenas son mantenidas mientras que el proceso de visualizacion (*Pipeline*) de OpenGL es usado de forma normal. Ésto permite que cada cadena posea su propio color (en el caso anterior no es posible, debido a que todos los trazos están en un mismo plano).

El modo básico sólo busca aquellas aristas que son parte de la silueta (Figura 3.9). Las aristas son tratadas como elementos individuales. OpenGL realiza la eliminación de las partes invisibles. Es un modo muy rápido, pero no tiene el mismo nivel de flexibilidad de los anteriores modos.

Una vez descritas las distintas fases del funcionamiento del algoritmo, se expone la implementación del formalismo luces virtuales y el método de cálculo de intersecciones, cuando los mismo tienen que ser soportados por un modelo poligonal basado en triángulos.

3.2.1. Luces virtuales

La formalización que se expuso en el capítulo anterior era una descripción general del mecanismo de las luces virtuales, sin estar asociado a ningún modelo geométrico en concreto. En este apartado se realiza una descripción del mismo bajo el enfoque de su aplicación a un modelo geométrico concreto, el poligonal basado en triángulos 2-variedad (*2-manifold*).

Para ello hay que realizar una particularización de las definiciones genéricas para el modelo poligonal. Debido a que se usa un modelo poligonal, las definiciones de las luces virtuales y de los trazos tienen que adoptar una expresión que pasa de continua a discreta.

La definición de los trazos, tanto internos como externos, para un modelo poligonal es la que se expone a continuación, aunque previamente se debe establecer el significado de visibilidad e invisibilidad de una cara:

Definición 7 *una cara es visible si el producto escalar del vector normal de la cara y el vector dirección de la luz virtual es mayor que 0. En otro caso la cara es invisible.*

Ésta es la definición normal de visibilidad. En caso de que la luz virtual esté en el infinito, la dirección es dada directamente por su definición. En caso de que la luz virtual sea local, el vector dirección debe ser calculado a partir de la diferencia entre su posición y un punto en la cara. Una vez establecido el significado de visibilidad, se pueden definir los trazos.

Definición 8 *dado un modelo poligonal 2-variedad y una luz virtual, un trazo externo es una arista que tiene una cara iluminada (visible) y otra no iluminada (invisible).*

Obsérvese que para obtener la misma silueta que se ve desde la posición

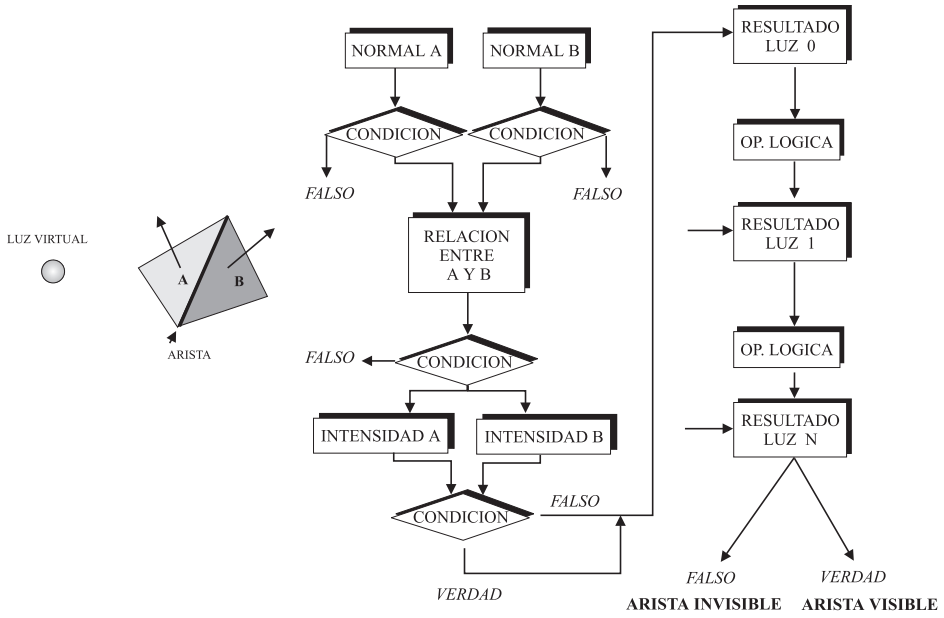


Figura 3.11: Procedimiento de cálculo de las reflexiones usando luces virtuales.

del observador, la luz virtual debe estar ocupando la misma posición que el observador. Éste es el procedimiento que permite tratar de la misma manera a los trazos externos y a los internos, ésto es, que los trazos son determinados por las luces virtuales.

Definición 9 *dado un modelo poligonal 2-variedad, una luz virtual y tres intervalos de números reales* $IC(ic_1, ic_2)$, $DC(dc_1, dc_2)$, $RC(rc_1, rc_2)$, *un trazo interno es una arista que satisface las siguientes condiciones:*

(a) El ángulo que forman el vector normal y la luz virtual está en el intervalo IC.

(b) El ángulo entre los vectores normales de sus dos caras está en el intervalo DC

(c) La diferencia calculada de reflexiones, que se obtiene usando el modelo de iluminación, está incluida en el intervalo RC.

Las condiciones (a) y (b) pueden ser aplicadas para obtener los trazos externos. La condición (b) permite seleccionar aquella configuración en la cual la cara tiene una posición más cóncava o mas convexa. Los valores de las distintas luces virtuales hacen que el método sea muy flexible en la obtención de los trazos dónde, cómo y cuándo el usuario quiera. El proceso es mostrado en la Figura 3.11 . La operación de combinación de los resultados finales ya ha sido explicada en el capítulo anterior, aunque en este caso, los elementos geométricos son las aristas. En la Figura 3.12 hay un ejemplo de donde se pueden ver la posición de las luces virtuales y como afectan a las aristas.

Las aristas de los objetos tiene un atributo para cada luz virtual definida. Este atributo puede tener valores, que indican si la arista es siempre un trazo,

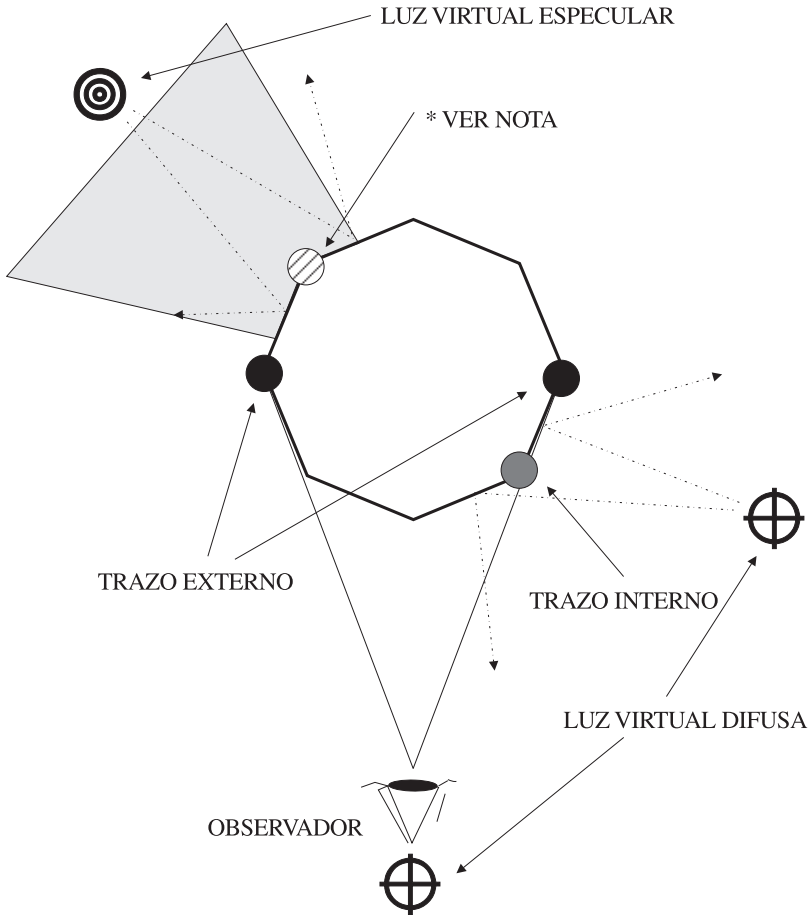


Figura 3.12: Esquema de la aplicación de las luces virtuales. * Esta arista será un trazo interno cuando el observador entre dentro de la zona gris.

no lo es nunca o depende de las operaciones con las luces virtuales. Este mecanismo permite, por ejemplo, que cuando dos objetos son unidos, parte de sus respectivos trazos externos sean eliminados. Para ello, se indica a las aristas seleccionadas, los bordes de contactacto de los dos objetos, que no pueden ser seleccionadas para ser trazos.

El formato de las luces virtuales se expone con detalle en el Apéndice A, sección A.2

3.2.2. Obtención de las intersecciones

Como se expuso en el algoritmo general (Pág. 69), una vez se han detectado las aristas que formarán los trazos hay que calcular las intersecciones, que se pueden producir entre las mismas y que afectan a su visibilidad, pero también hay que detectar el caso de trazos en los que no se producen intersecciones y que pueden ser o no ser visibles, para el caso en el que se use el modo 1 de visualización (Pág. 73). El problema del cálculo de las intersecciones entre aristas se resuelve calculando las intersecciones de las caras. Hay que calcular las intersecciones de cada cara del tipo trazo con todas las demás,

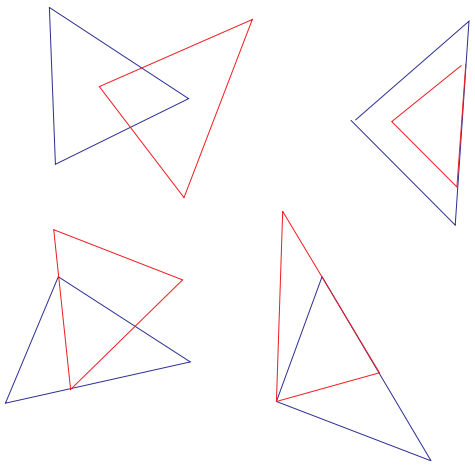


Figura 3.13: Coincidencia de vértices.

ésto es, el resto de las tipo trazo y las tipo normal. Ésto hace que el cálculo de intersecciones presente una complejidad cuadrática. Este cálculo se puede descomponer en un proceso más sencillo de evaluación de las intersecciones entre dos caras. Mediante el uso de dicho método con todas las caras afectas, se obtiene el resultado final.

A continuación se describe la solución que se ha dado al problema del cálculo de las intersecciones entre dos triángulos. Para poder llevar a cabo dicho cálculo se imponen ciertas condiciones:

1. Las caras deben ser convexas
2. Las caras deben ser planas
3. No se permite la interpenetración de caras
4. Las caras presentan el mismo sentido en su definición (el sentido elegido es indiferente)
5. Dos caras no pueden a la vez ser coplanarias e intersectar en sus proyecciones. Ciertos casos degenerados se resuelven mediante convenios
6. Dos aristas de un mismo polígono no pueden ser colineales. Esto se garantiza con triángulos no degenerados

Formalmente, los elementos geométricos con los que opera el algoritmo deben ser considerados como polígonos de forma triangular, ya que al operar sobre los triángulos originales, éstos pueden convertirse en polígonos con más de tres vértices. Las condiciones 1, 2, 4, 5 y 6 van orientadas a permitir una solución más sencilla del problema. En el caso de la condición 3, se debe a motivos de eficiencia.

El incluir las caras que no aportan arista, las denominadas normales, permite poder tratar correctamente los trazos que no intersectan con otros. No es necesario calcular las intersecciones entre caras del tipo normal. Ésto reduce

el número de cálculos, ya que el número de caras que están marcadas como cara_trazo suele ser bastante menor que el número total de polígonos, siendo una función monótona decreciente (excepto en objetos aleatorios). Para reducir el número de polígonos que deben ser tratados se usan cajas englobantes. Otra posibilidad que se ha explotado, es optimizar el número de cálculos a nivel de intersecciones entre caras. Cuando se detecta un estado o intersección para una cara que tiene un estado o intersección simétrico con respecto a la otra cara, se aplica el mismo resultado a las dos. El algoritmo debe poder operar con cualquier configuración de dos triángulos. Así, debe permitir casos normales y casos degenerados (Figura 3.13), también varios triángulos que compartan una arista tipo trazo (Figura 3.14), y el caso en el que puede haber coincidencia en la proyeccion de varias aristas, pudiendo ser éstas de distinto tipo. Este último es el caso genérico que se presenta con objetos 3D abiertos y cerrados (Figura 3.15). Aunque las caras que comparten una arista comparten sus puntos extremos, no ocurre lo mismo con los atributos de dichos puntos, según sean tomados como pertenecientes a una cara u otra.

A continuación se definen unos conceptos necesarios para entender la forma de operar del algoritmo. El concepto de cambio de visibilidad es asimilable al de invisibilidad cuantitativa de Appel, pero con cierto matiz que los distingue. El esquema de cambio de visibilidad que se ha implementado informa de cuántas veces un vértice está detrás de polígonos y cuántas veces delante. Cada punto posee un atributo con dos valores que cuenta el número de veces que el punto está delante de un polígono y el número de veces que está detrás. Esta información también se puede obtener a partir de un trazado de rayos u otros métodos que trabajen a nivel de puntos o aristas, pero existen ciertos casos que no se tratarían correctamente, como es el de la coincidencia de una arista entre varias caras. El interés por poder disponer de esta información se debe a que permite añadir un grado más en el control de la visualización de las siluetas. En animación, pero sobre todo en ilustración, el aspecto de los trazos cambia en función de su posición. Así, por ejemplo, nos encontramos que las siluetas exteriores suelen ser dibujadas con un grosor mayor que el de

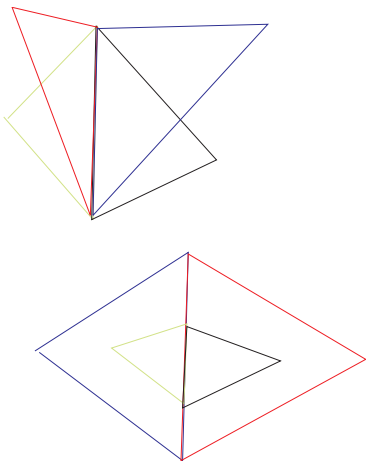


Figura 3.14: Compartición de una arista.

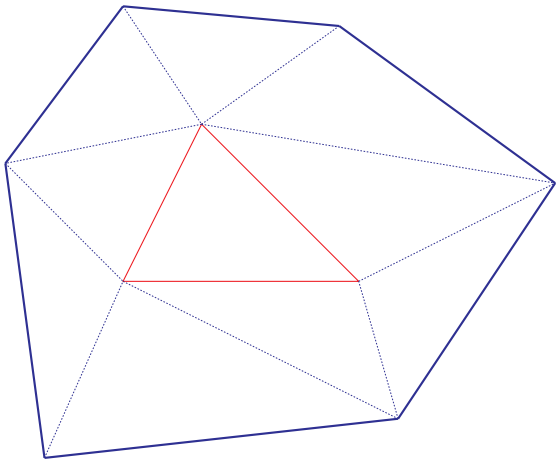


Figura 3.15: Coincidencia en la proyección de varias aristas.

las siluetas interiores. Éste es el tipo de característica visual que se pueden obtener al disponer de la información de cambio de visibilidad indicando la posición de delante y de detrás.

Los puntos y sus atributos de cambio de visibilidad son los elementos que definen las aristas. La geometría de una arista se define por su dos puntos extremos, pero los atributos de la misma son los del primer punto, siguiendo el sentido en el que se recorren las caras a las que pertenece la arista. Se puede decir, más formalmente, que dados dos vértices, V1 y V2, la arista se define como el intervalo semiabierto [V1,V2). Por ésto, una misma arista compartida puede ser dibujada de diferente forma, ya que para cada cara existe un atributo que indica como debe ser dibujada.

Un concepto importante es el de vértices constituyentes y el de vértices añadidos. Los vértices constituyentes son aquellos que forman inicialmente el triángulo; su número no varía. Los vértices añadidos son vértices nuevos que se añaden a la geometría original cuando se produce una intersección normal (este concepto se ve más adelante). Una característica de este algoritmo es que cada vez que se calcula la interacción de un triángulo con respecto a otro, siempre se toma su geometría original, es decir, los vértices constituyentes. Los posibles vértices añadidos que poseen las caras no se usan para los cálculos, pero sí en la fase de actualización de datos. Otra característica importante es el concepto de referencia absoluta y referencia relativa. Este concepto se relaciona con el de cambio de visibilidad de la siguiente manera. Los vértices mantienen un atributo que indica el cambio de visibilidad que depende del tipo de vértice. Un vértice constituyente es siempre una referencia absoluta porque los valores de cambio de visibilidad que mantiene son absolutos. Los valores absolutos se pueden utilizar directamente. En cambio, los vértices añadidos se dice que son referencias relativas, porque los valores de cambio de visibilidad que mantienen son relativos. Un valor relativo debe referenciarse con respecto a un valor absoluto para que sea correcto. Los vértices constituyentes se inicializan de tal forma que indican que están en una posición neutra, ni delante ni detrás. Un vértice constituyente se dice que es interior si el mismo está dentro de un

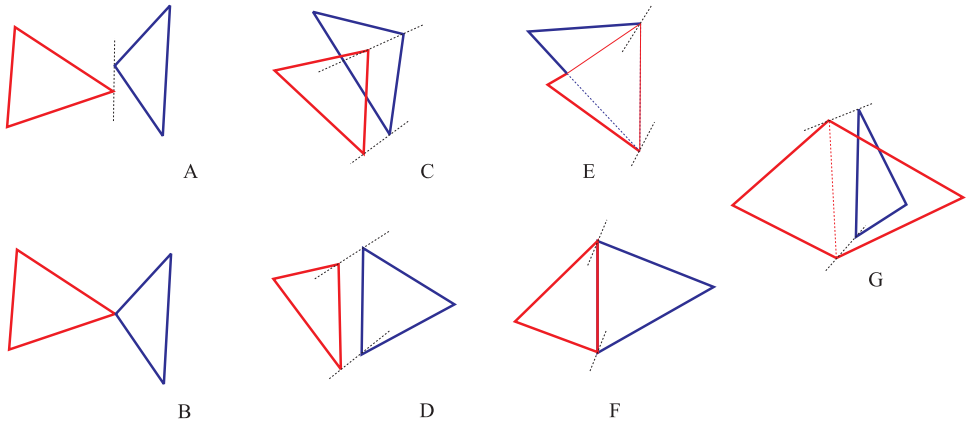


Figura 3.16: Distintas configuraciones de triángulos.

polígono (el interior de un polígono incluye su frontera).

Para que el algoritmo funcione de forma correcta, es necesario establecer dos convenios que permiten resolver ambigüedades y hacer que sea más sencillo y eficiente. Por ejemplo, entre los triángulos del caso A de la Figura 3.16 no existen intersecciones entre aristas, ni uno está dentro del otro, pero existe una coincidencia a nivel de proyección en un vértice: el vértice que está detrás, y hay que considerar si está dentro o fuera del otro triángulo. El convenio elegido es considerar que está fuera, ya que si se indicara que está dentro, al depender el atributo de visibilidad de la arista del atributo del vértice, se entendería que toda la arista está dentro. Además, también resuelve el caso en el cual la coincidencia del vértice es total (los dos puntos ocupan la misma posición en el espacio 3D, caso B, Figura 3.16).

Convenio 1 *si la única intersección entre dos caras se produce en un vértice, dicho vértice se considera externo, ésto es, se considera que no hay intersección.*

Otro caso es la coincidencia de una arista entre dos o más triángulos. Dicha coincidencia puede ser porque: a) los dos vértices de la aristas posean las mismas coordenadas, b) porque uno de ellos coincida totalmente y el otro coincida en la proyección, o c) porque los dos coincidan en la proyección. En todos los casos se deben dibujar las aristas en la misma posición. El problema está en decidir cuando la arista de una cara cubre la arista de la otra cara. Si se puede establecer una ordenación espacial el problema parece que está resuelto: simplemente se aplica la posición calculada a cada cara, quedando una delante y otra detrás, ésto es, las aristas se dibujan en el mismo sitio pero se ha establecido un orden (casos C y D de la Figura 3.16). El caso E, que se distingue porque existen intersecciones, además de la coincidencia de arista, se puede resolver de la misma manera, de forma artificial, aplicando un convenio, ya que las dos aristas están en la misma posición y por tanto ninguna tiene preferencia sobre la otra. Esta ambigüedad se puede eliminar observando no el posicionamiento de las aristas sino el de las caras a las que pertenecen. En tal caso, también se puede decidir una ordenación.

Para el caso F de la Figura 3.16, existe coincidencia total y no existen intersecciones, no se puede establecer una ordenación espacial, lo cual implica que el criterio de posición espacial no puede resolver este tipo de casos. Una posible solución es la siguiente: no puede haber coincidencia total aunque sí en las proyecciones. Ésto implica que no se pueden tener modelos que no sean 2-variedad. Con esta condición, siempre se puede establecer una ordenación espacial, con lo cual se puede establecer un orden en las aristas. Aparece otro problema a nivel de complejidad del algoritmo. Como se ve en el caso G, existe un coincidencia de arista pero se puede establecer una ordenación espacial. El problema está en que al considerar el triángulo A con respeto a B va a indicar que A está detrás de B, y cuando se haga la misma operación entre A y C también indicará que está A detrás de C: el efecto acumulativo es que A está detrás de dos objetos, lo cual no es cierto. Ésto se puede resolver a costa de complicar el algoritmo detectando este tipo de situación; se ha implementado y es plenamente funcional, siendo una posibilidad más. La opción que se ha elegido, se basa en establecer un convenio parecido al que se ha comentado para el caso de la coincidencia de un vértice.

Convenio 2 *cuando existe coincidencia de arista entre dos triángulos, habrá interacción si los dos están al mismo lado de la arista compartida, sino no habrá interacción y se establece un orden predeterminado.*

El convenio elegido considera que en caso de que haya interacción, el orden lo establece la ordenación espacial de caras, la cual está asegurada. En caso de que no haya interacción, hay que establecer una ordenación: el triángulo que esté más a la izquierda está delante. Si se produce una igualdad con el criterio anterior, está delante el triángulo que esté por arriba. Este criterio hace que el algoritmo sea mucho más sencillo en el tratamiento de ciertos casos, se obtienen resultados que muestran un grado de coherencia visual alto y además permite trabajar con modelos que no sean 2-variedad[1].

Otra característica que posee el procedimiento es la de categorizar las siluetas. En un primer momento, y tal como se ha definido el problema, se podría plantear la solución como una ordenación de las caras y un dibujo de atrás hacia adelante de las mismas. El objetivo no es sólo obtener los trazos, sino que los mismos puedan ser representados de diversas formas, y no sólo con la posibilidad de la visibilidad o no que plantea la solución anterior. Ésta característica es considerada como esencial para su utilización, ya que se desea dotar de "carácter" a las siluetas.

Este "carácter" se implementará permitiendo que los trazos presenten diversas formas al comienzo, parte intermedia y final, también variación en el grosor y en el estilo, etc (Figura 1.28). Esta variación podrá depender de la posición relativa que ocupe la silueta con respecto a otras siluetas. Por ejemplo, se podrá visualizar la parte de silueta que esté detrás con un tipo de líneas, o si la silueta pasa por delante de otra cambiar de grosor, etc. Esta flexibilidad

[1]El método de cálculo de interseciones permite modelos 2-variedad, pero por simplificación del resto de elementos del modelo, no se permite

no se puede obtener con un método que se limite a la ordenación y visualización de dicha lista ordenada. Es necesario conocer dónde y de qué forma ocurren las intersecciones entre siluetas, y, en caso de que ocurran, determinar la visibilidad y ocultación de las siluetas.

Solución

Para la solución, y partiendo de la necesidad de que hay que calcular las intersecciones entre los triángulos, se ha tomado un esquema de funcionamiento parecido al usado en el recorte de polígonos de Sutherland-Hodgman [21, Págs. 124-129].

El cálculo de las intersecciones es la tarea que plantea mayores problemas. El algoritmo implementado opta por una estrategia incremental, de tal manera que se intentan seleccionar antes los casos más sencillos, dejando los más difíciles y costosos de calcular para el final. Los pasos que sigue el algoritmo son los siguientes:

1. Clasificar los posibles casos que se pueden presentar entre dos triángulos.
2. Calcular las intersecciones.
3. Actualizar los valores tanto de geometría como de atributos.

En el primer paso se realiza una clasificación de la interrelación de los triángulos. Dado el triángulo A y el triángulo B, se van a distinguir los siguientes casos:

$$\begin{aligned} & A \cap B = 0 \\ & A \cap B \neq 0 \quad \Rightarrow \quad A \subset B \vee B \subset A \\ & \qquad\qquad\qquad\quad A \not\subset B \wedge B \not\subset A \end{aligned}$$

Primero se intenta determinar si entre los dos triángulos no hay ningún tipo de interacción (teniendo en cuenta los convenios). En tal caso no hay que actualizar ningún dato. Obsérvese que esta comprobación es necesaria ya que la prueba de las cajas fronteras no es determinante. La siguiente comprobación es si uno de los triángulos está contenido dentro del otro. Si se cumple, el otro está fuera y, por tanto, sólo hay que actualizar los datos del triángulo que está contenido. Puede estar contenido por delante o por detrás del otro triángulo, siendo el resultado visual muy distinto. Si se pasan estas dos pruebas implica que entre los dos triángulos existen intersecciones y se pasa a la siguiente fase.

Obsérvese que los procesos son simétricos en el sentido de que primero son aplicados a un triángulo con respecto al otro y luego al revés. La actualización se produce debido a que pueden aparecer nuevas intersecciones, que deben ser incluidas, pues en ellas es en donde se producen los cambios de visibilidad.

Antes de exponer en detalle el cálculo de intersecciones se definen lo que son las intersecciones normales y las degeneradas.

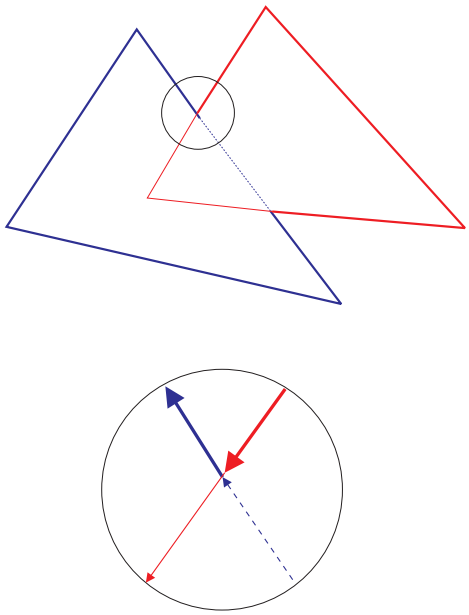


Figura 3.17: Intersección normal.

Una intersección normal es aquella que se produce en el interior de dos aristas (Figura 3.17). Formalmente, dadas dos aristas, A1 con extremos en los vérticea P1 y P2, y A2 con extremos en los vértices Q1 y Q2, se dirá que se produce una intersección normal si: se produce una intersección, las aristas no son colineales y la intersección I cumple que $P1 < I < P2$ y $Q1 < I < Q2$. Cuando se cumplen las tres condiciones, la intersección para ambas aristas es la misma, al menos en las coordenadas proyectadas, pudiendo diferir en la profundidad.

En las intersecciones se producen cambios de visibilidad para las aristas. Estos posibles cambios son: que la arista entre o se produzca una entrada (pasar de fuera a dentro) en el polígono o que la arista salga o se produzca una salida (pasar de dentro a fuera) del polígono. Además hay que incluir un atributo que señale que las entradas y/o salidas pueden ser por delante o detrás del polígono (Figura 3.17). También aquí se produce una situación simétrica, ya que si se determina que una arista entra, la otra sale. Ésto se usa para reducir el número de cálculos. La actualización de los cambios de visibilidad para las intersecciones normales se aplica sobre los vértices nuevos, los cuales son vértices añadidos a la geometría del triángulo. El cambio de visibilidad en un vértice añadido se define como una referencia relativa. Cuando se produce una entrada, el valor correspondiente debe ser incrementado. Cuando se produce una salida, el valor correspondiente debe ser decrementado. Para que estos valores sean correctos deben referenciarse a un valor absoluto.

Una intersección especial o degenerada [39, Pág. 154] es aquella que se produce en uno o dos vértices, dándose el primer caso cuando hay una intersección entre el interior de una arista y un vértice y el segundo cuando la intersección se produce en dos vértices (Figura 3.18). Este tipo de intersección es el más difícil de resolver. Existen dos posibilidades con respecto al tratamiento de las

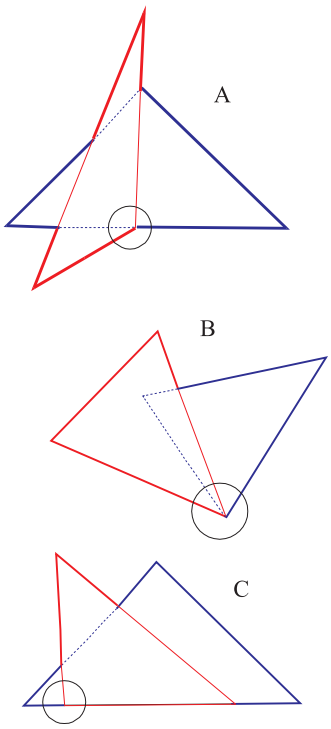


Figura 3.18: Intersecciones degeneradas.

intersecciones degeneradas: una es considerar que aunque sea degenerada, se crea un nuevo vértice con la misma posición, e intentar tratar esta situación en la que aparece un mismo punto repetido. Esta opción lleva a una implementación dificultosa. La otra opción es la de no repetir la geometría pero tener en cuenta los posibles cambios de visibilidad que se produzcan. Veamos como se tratan las intersecciones degeneras para tener en cuenta los cambios de visibilidad y, ya que se producen en vértices constituyentes, sigan siendo referencias absolutas. Cuando se produce una intersección degenerada, en la misma se pueden dar los mismos cambios que se dan en una intersección normal, ésto es, puede entrar o salir, y aunque lo normal es que se pueda aplicar la simetría antes comentada (Figura 3.18, casos A y B), también es posible que no exista dicha simetría, pudiéndonos encontrar con casos en los cuales para una cara la intersección representa una entrada y para la otra cara significa que el vértice está dentro (Figura 3.18, caso C). Si se produce una intersección degenerada y se determina que se produce una entrada, se trata de tal manera que se aumenta en uno la variable correspondiente, ésto es, la de delante o de detrás. La diferencia con respecto a las intersecciones normales, está en que las salidas no producen ningún cambio en las variables que indican la visibilidad. 'Esto es así debido a que mientras que en las entradas se produce un cambio de visibilidad para la arista en la cual se produce dicha entrada, para las salidas se vuelve a recuperar el estado previo al de la entrada, lo cual es equivalente a dejar el estado como estaba. Mediante este esquema, aunque se produzcan intersecciones degeneradas, los vértices constituyentes siguen siendo referencias absolutas.

3.2.3. Descripción detallada del cálculo de intersecciones

Una vez se han expuesto de forma general el algoritmo de cálculo de las intersecciones. en esta sección se hace un estudio más detallado del mismo.

La estructuturación de los datos con la que se trabaja es la siguiente. Un polígono convexo, en este caso un triángulo, se va a describir mediante su geometría y su topología. Ésto es, se dispone de una lista de tres puntos y una lista de tres aristas para cada cara. Cada arista indica los dos puntos extremos que la conforman. Esta estructura permite el que, al producirse una intersección, ésta pueda ser incluida como un nuevo punto. Además, tanto las nuevas intersecciones como los vértices deben poseer unos atributos, que son los que permitirán modificar la forma en que serán visualizadas las siluetas. Todo punto, ya sea como vértice constituyente (inicial) o como nueva intersección, poseerá al menos dos atributos: su posición relativa con respecto al otro polígono, y la posición en profundidad con respecto al mismo. Con respecto al primer atributo podrá tomar los siguientes valores:

DENTRO
FUERA
ALINEACION_DENTRO
ALINEACION_FUERA
ALINEACION_VERTICE
ENTRAR
SALIR

Si un punto tiene como atributo uno que indique que está alineado, este estado será temporal, y al final tendrá que tener uno de los siguientes valores: DENTRO, FUERA, ENTRAR y SALIR. Con respecto al segundo atributo podrá tomar los valores DELANTE y DETRAS. Ambos atributos, así como su uso, serán explicados en siguientes apartados.

Clasificación de la posición de los puntos de un triángulo con respecto al otro

Lo primero que se hace es determinar la posición relativa de cada vértice del triángulo tratado (que llamaremos T1) con respecto al segundo triángulo (que llamaremos T2). Esta primera clasificación de los vértices de T1 se implementa comprobando qué posición, dentro o fuera, ocupa el vértice con respecto a cada arista del triángulo T2. Con el producto vectorial se determina si el punto está dentro o fuera con respecto a la arista estudiada de T2. Al aplicarlo con respecto a las tres aristas del triángulo T2, podemos obtener dos soluciones:

- Todas las posiciones son interiores.
- Alguna o todas las posiciones son exteriores.

En el primer caso indica que el punto está contenido en el triángulo T2. La segunda posibilidad sucede cuando el punto es exterior a T2.

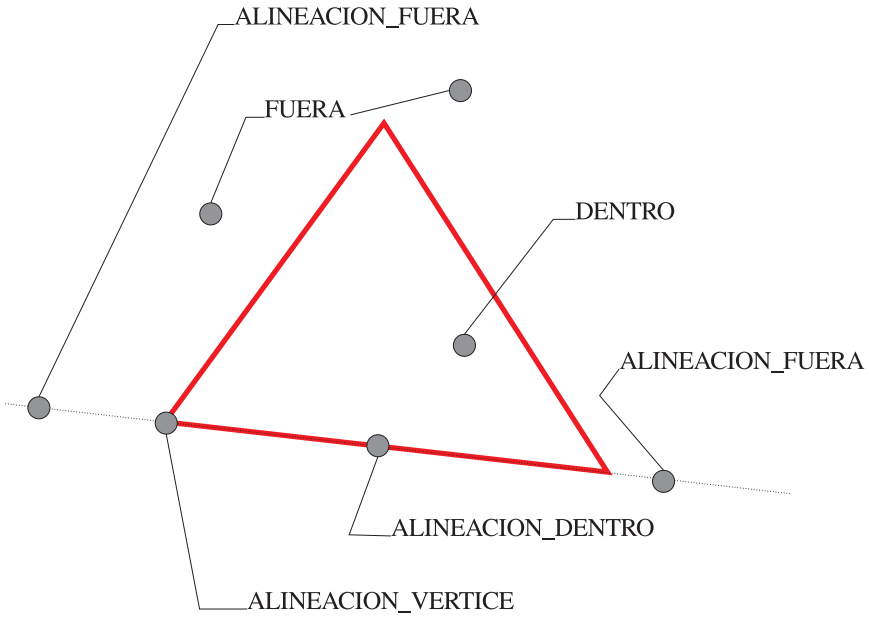


Figura 3.19: Posiciones que puede tener un punto respecto a un polígono.

Además de la categorización de los puntos entre interiores y exteriores, se realiza una subclasificación que es necesaria para el funcionamiento del algoritmo propuesto. Esta subclasificación se hace para distinguir aquellos vértices que coinciden con otro, o que estén incluidos en el interior de la arista, o que estén alineados por la parte no válida de la arista (Figura 3.19).

Esta clasificación se puede ver como un refinamiento de la clasificación general:

Posición general	**Posición concreta**
DENTRO	DENTRO
FUERA	FUERA
FUERA	ALINEACION_FUERA
DENTRO	ALINEACION_VERTICE
DENTRO	ALINEACION_DENTRO

La clasificación anterior hay que aplicarla con cada uno de los tres vértices del triángulo. Los posibles resultados que se pueden obtener con el conjunto de los vértices son los siguientes:

- Todos los puntos son interiores → DENTRO
- Todos los puntos son exteriores → FUERA
- Algunos puntos son interiores → INDETERMINADA

A la primera posibilidad que se le ha denominado DENTRO, a la segunda se le ha denominado FUERA y a la tercera INDETERMINADA. Esto nos da

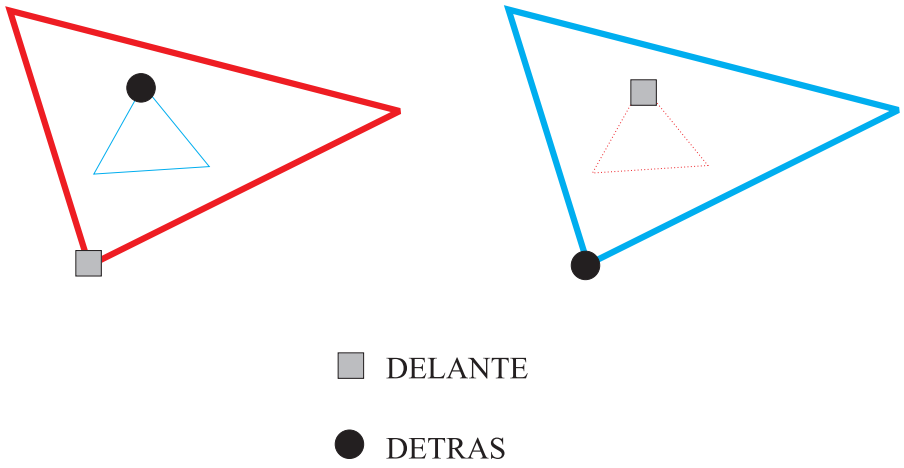


Figura 3.20: Condiciones DENTRO-FUERA y FUERA-DENTRO.

9 posibilidades con las que nos podemos encontrar que son las que se muestran en la tabla siguiente:

Triángulo T1	Triángulo T2	Caso
DENTRO	DENTRO	1
DENTRO	FUERA	2
DENTRO	INDETERMINADO	3
FUERA	DENTRO	4
FUERA	FUERA	5
FUERA	INDETERMINADO	6
INDETERMINADO	DENTRO	7
INDETERMINADO	FUERA	8
INDETERMINADO	INDETERMINADO	9

El caso 1 indica que se ha producido una coincidencia entre los tres vértices de los dos triángulos (esta coincidencia debe ser, al menos en un punto, en la proyección, ya que si no los dos triángulos serían coplanarios). En tal caso hay que determinar la posición relativa, en profundidad, de uno con respecto al otro. Los casos 2 y 4 se resuelven igual que el anterior (Figura 3.20). Obsérvese que para los casos 2 y 4 (también para el 1 según el algoritmo propuesto), no existen intersecciones. En estos casos todos los vértices de un triángulo están marcados como DENTRO y los del otro triángulo FUERA, y para el segundo caso al revés. Pero es necesario determinar su visibilidad para lo cual es necesario obtener las posiciones relativas en profundidad de ambos triángulos. Esta información vendrá dada por los valores DELANTE y DETRAS. Por tanto, cada uno de estos puntos indicará su posición respecto al otro triángulo así como su posición en profundidad con respecto al mismo.

El caso FUERA-FUERA no implica que no haya interacción entre los triángulos, como se puede ver en la Figura 3.21, por lo cual habrá que continuar comprobando si existen intersecciones. En caso contrario los triángulos serían disjuntos.

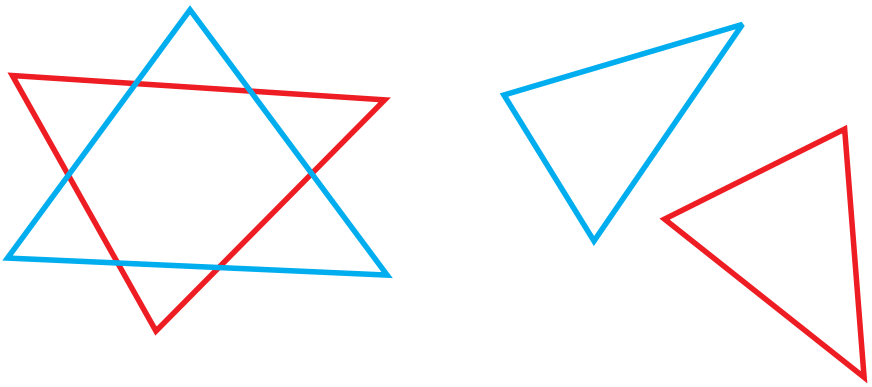


Figura 3.21: Casos FUERA-FUERA.

Los casos 3 y 7 no se pueden producir. En el resto de los casos se producen nuevas intersecciones que habrá que calcular, proceso que se ve en el siguiente apartado.

Detección y cálculo de intersecciones

Una vez se ha determinado la posición de los puntos de un polígono con respecto al otro polígono, se pasa a la fase en la cual se detectan las intersecciones. Debido a que dicho proceso es relativamente complejo, si no se producen intersecciones entre los polígonos se termina; en caso contrario hay que actualizar los datos de los mismos, tanto en cuanto a añadir las nuevas intersecciones como cambiar los atributos de visibilidad de los vértices, si fuera necesario, todo lo cual se realiza en la fase final del algoritmo.

En la detección de intersecciones es donde este algoritmo presenta un parecido mayor con el algoritmo de recorte de polígonos de Sutherland-Hodgman. Dado un triángulo T1, se va a estudiar si ocurren intersecciones en cada una de sus aristas, de forma secuencial. Para ello, lo primero que se hace es clasificar los puntos del triángulo T2 con respecto a la arista A1 que está siendo estudiada. Dicha arista hace el papel de lado de la ventana. No se opera sobre el conjunto de puntos sino sobre la arista. Esta arista, dada una orientación, divide el plano de dos zonas: interior y exterior. Pero como ya se ha comentado, hay una subclasificación, por lo cual se indicará si existe algún tipo de alineación por parte de los vértices del triángulo T2.

Hay que hacer la aclaración de que ambas clasificaciones, la de los puntos con respecto al polígono y la de los puntos con respecto a cada arista del polígono, tienen distintos objetivos. El primero se hace para intentar poder determinar la necesidad de hacer más cálculos en etapas posteriores, además de para detectar la coincidencia de vértices, lo cual implicará, un estudio de dicho vértice acerca de su visibilidad. En el segundo caso, la clasificación se hace para detectar intersecciones. Además es cambiante con respecto a cada arista, por lo que sus datos individuales no informan de la posición con respecto al polígono como totalidad.

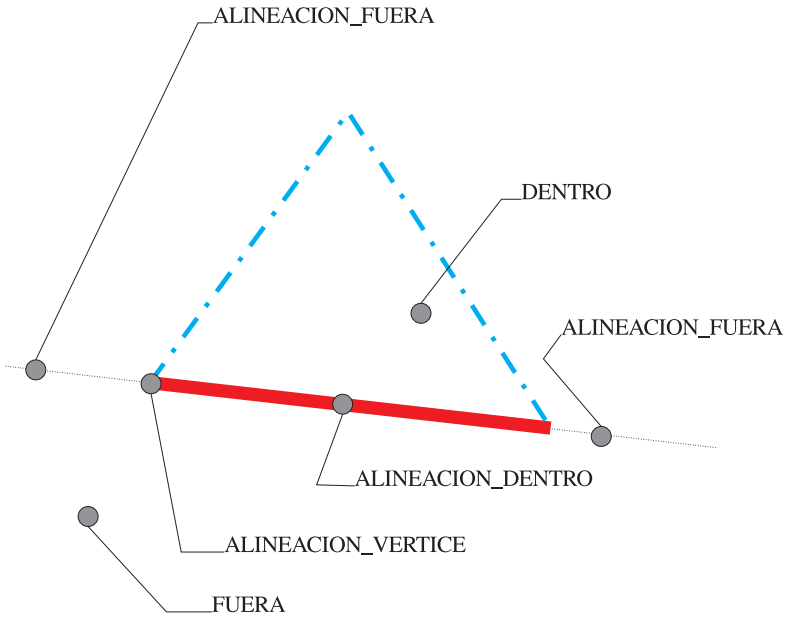


Figura 3.22: Posiciones que puede tener un punto con respecto a una arista.

En la clasificación con respecto a cada arista, indicar que la clasificación sólo se hace con respecto a los vértices constituyentes del polígono, sin tener en cuenta el que hayan podido aparecer nuevos puntos debido a intersecciones. Los posibles casos que se pueden dar para un punto respecto a una arista, son los siguientes (Figura 3.22).

- DENTRO
  El punto está en la parte interior con respecto al lado (depende de la orientación).
- FUERA
  El punto está en la parte exterior.
- ALINEACION_VERTICE
  El punto coincide con otro punto del segundo triángulo. Sólo puede referirse al primer vértice de la arista. Si fuera el segundo no se trataría y se dejaría que fuera del mismo tipo pero para el primer punto de la segunda arista.
- ALINEACION_DENTRO
  El punto está alineado con la arista del segundo triángulo en el interior de la misma, y por tanto no se consideran los vértices.
- ALINEACION_FUERA
  El punto está alineado con la arista del segundo triángulo en la parte no válida de la misma.

Una vez se tienen clasificados los puntos con respecto a la arista, éstos empiezan a ser recorridos de uno en uno, y en función de su posición y la del

Para todo lado del triangulo T1 **Hacer**
Obtener la posición relativa de cada punto del triángulo T2
Para todo todo punto del triángulo T2 **Hacer**
Dependiendo de la posición del punto **Hacer**
DENTRO:
Dependiendo de la posición del punto siguiente **Hacer**
FUERA:
Si hay intersección **Entonces**
Calcular la intersección
FinSi
FinDependiendo
FUERA:
Dependiendo de la posición del punto siguiente **Hacer**
DENTRO:
Si hay intersección **Entonces**
Calcular la intersección
FinSi
FinDependiendo
ALINEADO:
Si el punto esta alineado dentro **Entonces**
Dependiendo de la posición del punto siguiente **Hacer**
ALINEADO: Calcular la intersección
DENTRO:
Dependiendo de la posición del punto anterior **Hacer**
ALINEADO | FUERA: Calcular la intersección
FinDependiendo
FUERA:
Dependiendo de la posición del punto anterior **Hacer**
ALINEADO |DENTRO: Calcular la intersección
FinDependiendo
FinDependiendo
FinSi
FinDependiendo
FinPara
FinPara

Figura 3.23: Algoritmo para la obtención de intersecciones.

siguiente, se determinará si se produce o no intersección. También puede ocurrir que también necesite saber la posición del punto anterior para poder discernir si ocurre o no la intersección. En el primer caso hablaremos de intersecciones simples o en aristas. En el segundo caso hablaremos de intersecciones complejas o en vértices.

En la Figura 3.23 se describe el algoritmo para la obtención de intersecciones.

Intersecciones en aristas

En este apartado se van a estudiar las intersecciones que se pueden producir sólo cuando un punto está clasificado como DENTRO y el siguiente como FUERA o al revés, el primero FUERA y el siguiente DENTRO.

Cuando se presenta este caso, se produce una intersección entre la arista formada por los dos puntos que están siendo tratados y la recta en la cual está incluída la arista del triángulo T1, sobre la que se desea saber si hay o no intersección. Observar que esta condición garantiza que se produce intersección, pero hay que comprobar que la misma se produce en el rango válido de la arista.

Las intersecciones que se buscan son aquellas que se producen en el intervalo interior de ambas aristas. Más formalmente, dada la arista A1 definida como [V1,V2) y la arista A2 definida como [V3, V4), las intersecciones que se exponen son aquellas que ocurren entre (V1,V2) y (V3, V4) (Figura 3.24).

La detección de una intersección válida se realiza comprobando que, para cada una de las aristas que se están estudiando, los dos puntos extremos de la otra arista tienen también las posiciones relativas de DENTRO-FUERA o FUERA-DENTRO. Ésto elimina el que haya algún vértice que esté alineado.

Hay que hacer notar que siempre que se produce una intersección, se produce un cambio en el estado de la visibilidad. Este cambio de visibilidad, en nuestro caso, no va a indicar necesariamente que sea o no visible, ya que, como se ha comentado, se desea que las siluetas tengan otros atributos. Por ello, en vez de indicar su visibilidad de una forma absoluta, se indica si en la intersección se produce una entrada de la arista en la otra cara o una salida, que son las dos posibilidades que existen. A la primera se le ha denominado ENTRAR

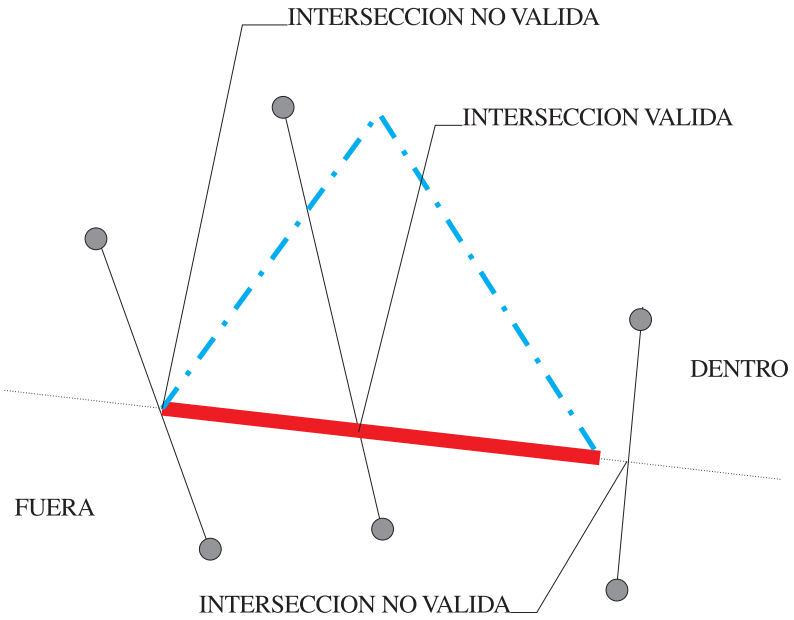


Figura 3.24: Intersecciones válidas y no válidas en el interior de la arista.

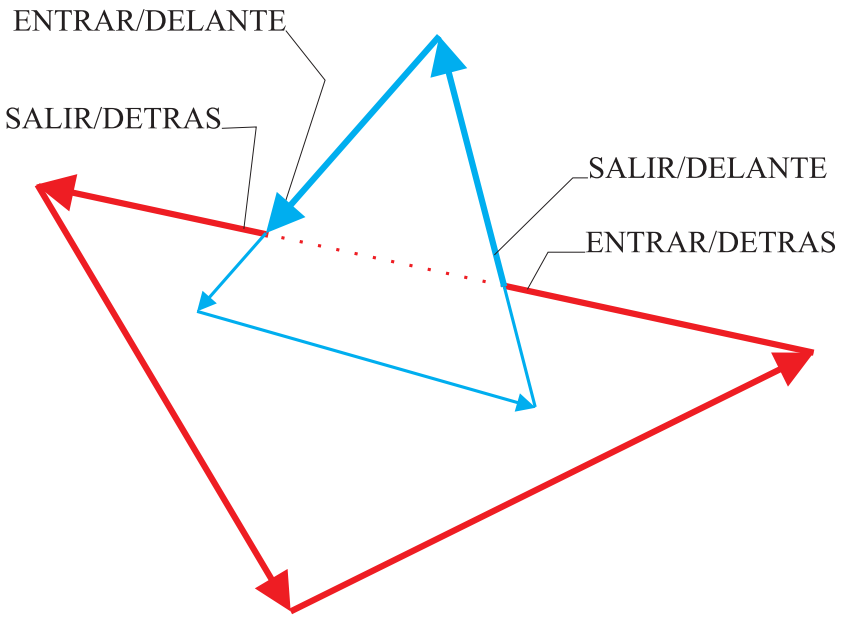


Figura 3.25: Atributos en las intersecciones.

y a la segunda SALIR (Figura 3.25).

Además del atributo que indica en que forma se produce el paso en la intersección, hay que incluir otro atributo que indique la posición relativa en profundidad en dicha intersección. Este atributo tendrá como posibles valores DELANTE y DETRAS. Por tanto, se va a tener para cada intersección que se produzca cuatro posibilidades: ENTRAR/DELANTE, ENTRAR/DETRAS, SALIR/DELANTE y SALIR/DETRAS, que son las que se muestran en la Figura 3.25. Gracias a esta información, es posible implementar los cambios en la visualización de las siluetas.

En la Figura 3.26 se muestran los distintos casos que se pueden producir, indicando si existe o no nueva intersección

Intersecciones en vértices

Los motivos por los que hay que distinguir el caso de cuando se produce una intersección en el interior de la arista de la que se produce en un vértice y la necesidad de subclasificar la posición de los puntos y distinguir el que se produzca una alineación se expone a continuación.

Hay que hacer una primera observación, y es que cuando ocurre una intersección en un vértice no es necesario añadir un nuevo punto, ya que éste coincide con el propio vértice. En caso de optar por no duplicar puntos, se llega a la conclusión de que es necesario realizar la subclasificación de las posiciones para poder detectar dichos casos y poder actuar en consecuencia. La coincidencia de vértices y el alineamiento de aristas hacen que el algoritmo sea más complicado, ya que debe distinguir dichos casos.

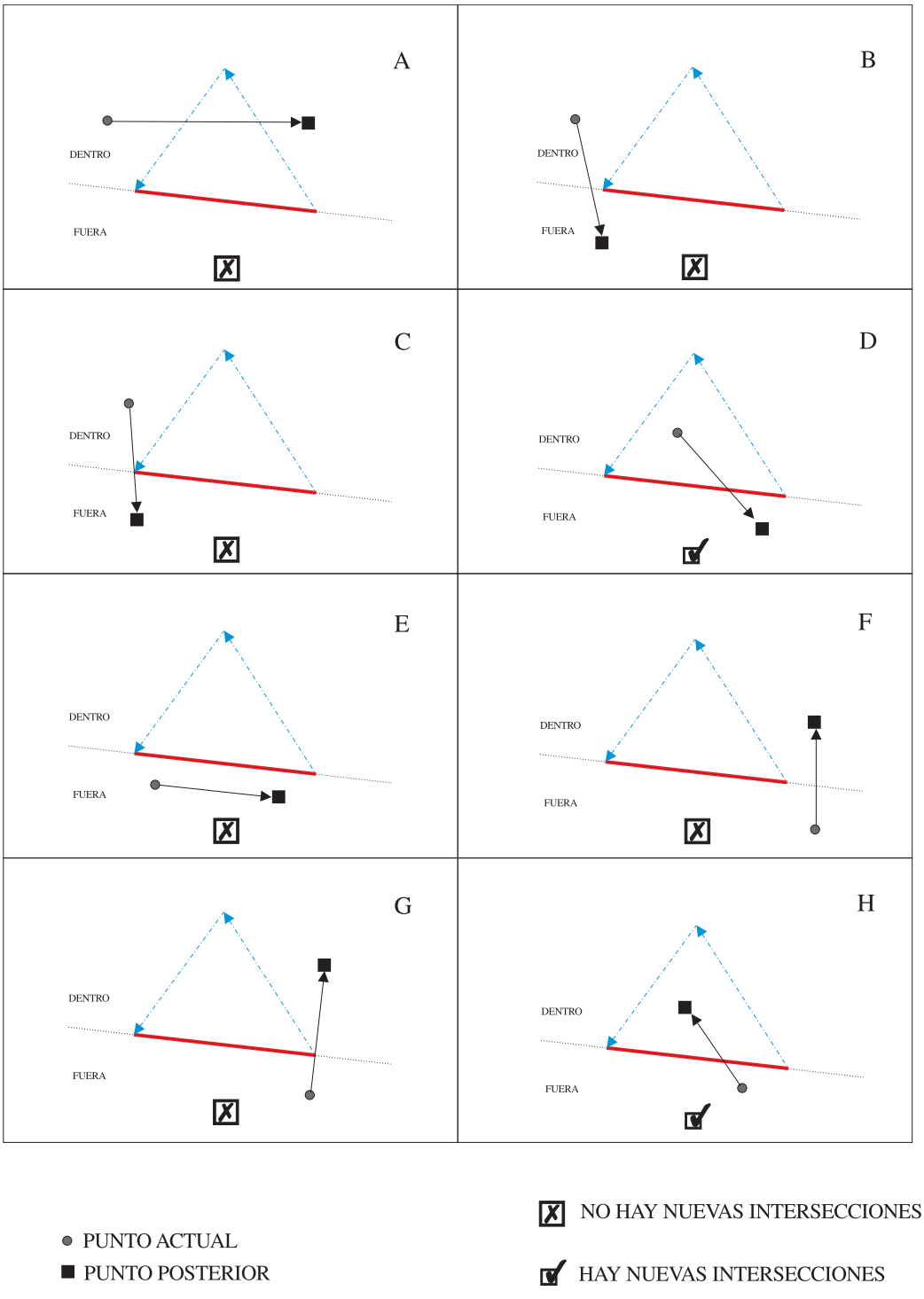


Figura 3.26: Casos posibles para las intersecciones en arista.

Dadas las tres posibilidades de alineación de un punto, se observa que sólo se van a producir nuevas intersecciones si ocurre que el punto tenga como atributo ALINEACION_ DENTRO. Si el punto tienen como atributo ALINEACION_VERTICE no se producirá ninguna nueva intersección pero habrá que indicar si se produce cambio en la visibilidad del vértice afectado. En caso

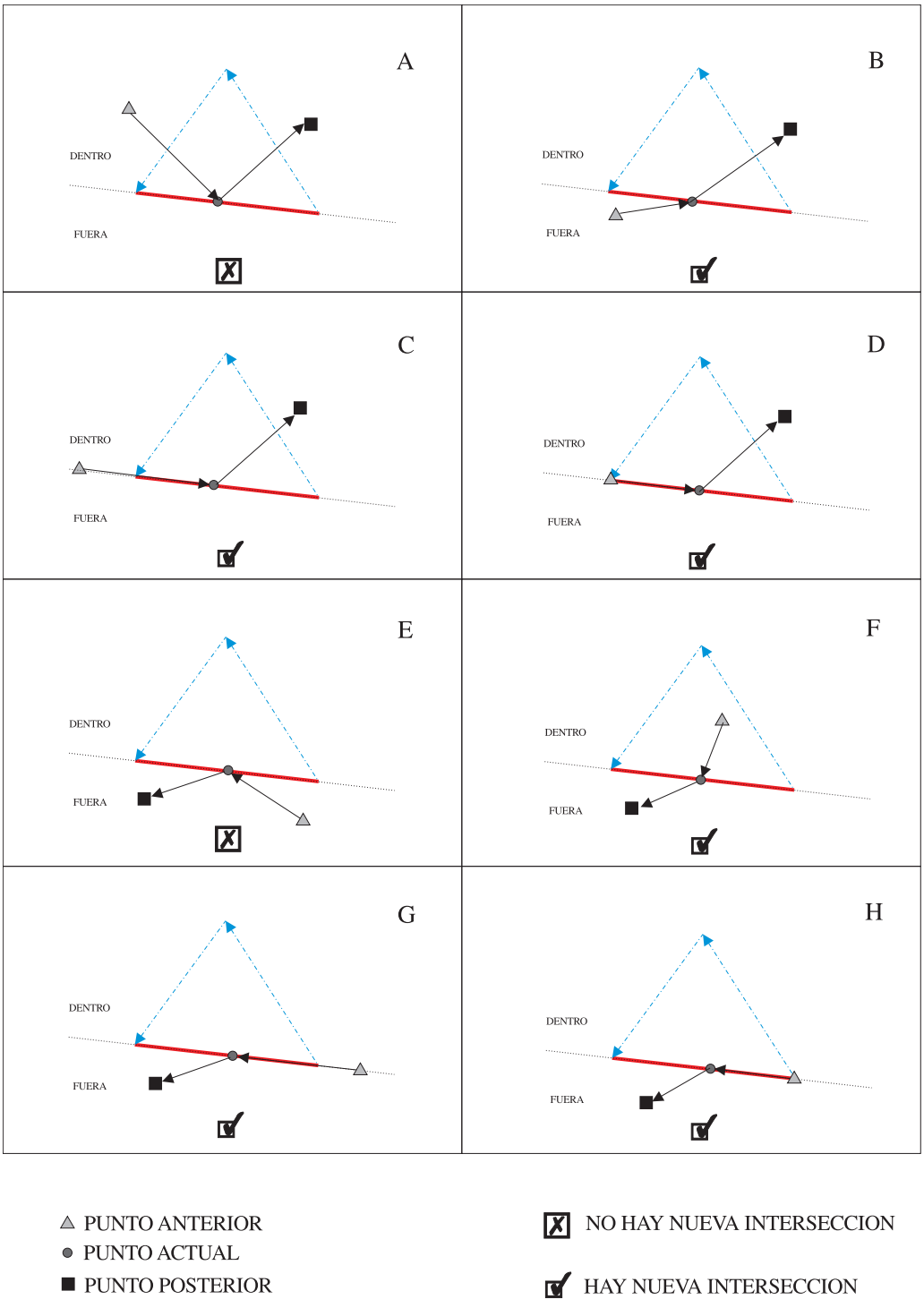


Figura 3.27: Casos en los que se produce intersección.

de que el punto tenga como atributo ALINEACION_FUERA, no se produce ningún cambio directo. Teniendo los puntos clasificados, los posibles casos que se presentan son los que se muestran en la Figura 3.27, en la que también se indica si se produce intersección o no.

Las nuevas intersecciones, como se ve en la Figura 3.27, sólo se producirán

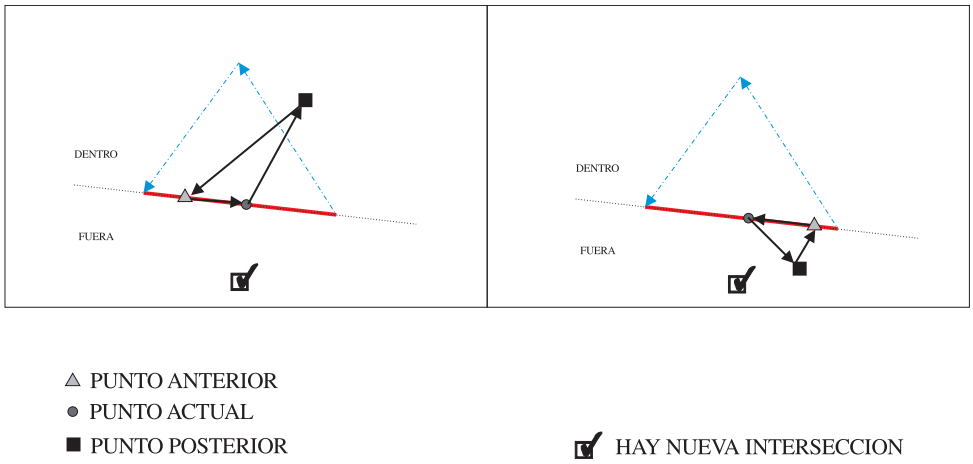


Figura 3.28: Alineación de dos aristas.

en ciertas condiciones. Si el siguiente punto está también alineado. Si el siguiente punto está DENTRO, obsérvese que hay que determinar el estado del punto anterior: si está FUERA o ALINEADO entonces hay una nueva intersección. Si el siguiente punto está FUERA, se producirá una nueva intersección si el punto anterior está DENTRO o ALINEADO.

En estos dos últimos casos hay un detalle importante. La clasificación de un punto como ALINEADO_FUERA podría entenderse como algo superfluo, ya que simplemente podrían entenderse como que el punto está FUERA. Pero como se puede comprobar en la descripción del funcionamiento del algoritmo, hay casos en que, para determinar si se produce una nueva intersección, se comprueba que o bien el punto siguiente o el punto anterior está ALINEADO. Si quitamos la categoría de la ALINEACION_FUERA el algoritmo no funcionaría correctamente como se puede observar en los apartados E y G de la Figura 3.27. El caso en el que se producen dos alineaciones (Figura 3.28) es tratado de forma correcta, calculando para una de las aristas dos inter-

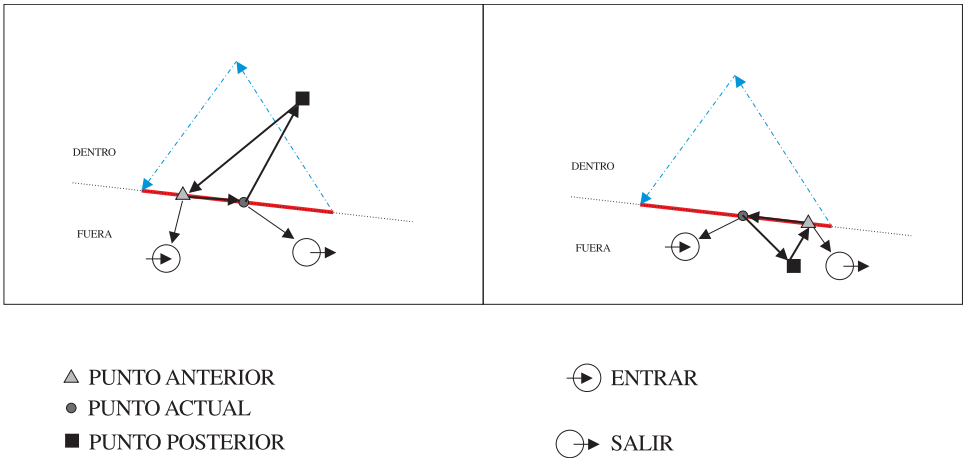


Figura 3.29: Cambio de estado en aristas alineadas.

secciones y para la otra ninguna. Obsérvese que si los puntos en vez de ser clasificados como ALINEADO_FUERA lo hubieran sido como FUERA no se hubiera podido obtener un resultado correcto.

Además de detectar que se produzcan o no intersecciones, la información sobre la posición de los puntos permite determinar cual va a ser el cambio de visibilidad que se va a producir en las nuevas intersecciones. La tabla que controla el cambio de visibilidad en las nuevas intersecciones es la siguiente (suponiendo que la orientación es contra el reloj):

Punto anterior	**Punto actual**	**Punto siguiente**	**Estado**
DENTRO	ALIN_DENTRO	DENTRO	
DENTRO	ALIN_DENTRO	FUERA	ENTRAR
DENTRO	ALIN_DENTRO	ALIN_DENTO o ALIN_FUERA o ALIN_VERTICE	ENTRAR
FUERA	ALIN_DENTRO	DENTRO	SALIR
FUERA	ALIN_DENTRO	FUERA	
FUERA	ALIN_DENTRO	ALIN_DENTRO o ALIN_FUERA o ALIN_VERTICE	SALIR
ALIN_DENTRO o ALIN_FUERA o ALIN_VERTICE	ALIN_DENTRO	DENTRO	SALIR
ALIN_DENTRO o ALIN_FUERA o ALIN_VERTICE	ALIN_DENTRO	FUERA	ENTRAR

En las Figuras 3.29 y 3.30 se pueden observar los cambios de visibilidad que representan la tabla.

Para comprobar que es necesaria dicha clasificación se van a exponer otras posibilidades en las cuales no se aplica la misma y se comprobará que los resultados que se obtienen no son correctos, o llevan a aplicar un mecanismo similar al utilizado por el algoritmo que aquí se propone. Se van a comentar otras dos posibilidades que se han considerado no válidas. En las dos se niega la necesidad de distinguir entre las intersecciones que se producen en el interior de las aristas y aquellas que ocurren en los vértices, lo cual implica que se producirán duplicaciones. La primera posibilidad que se va a estudiar es el caso en el cual se considere una arista como el intervalo cerrado entre los dos puntos extremos [V1,V2], mientras en el segundo se usa el mismo criterio que el usado por el algoritmo que se presenta, ésto es, el intervalo es semiabierto por la derecha [V1,V2):

- En el primer caso, [V1,V2), supongamos que se está estudiando la arista A1 del triángulo T1 y que los puntos que están siendo estudiados del triángulo T2, que llamaremos W1 y W2, el primero está DENTRO y el segundo está alineado con la arista A1 (caso A de la Figura 3.31); como partimos de que no se desea hacer subclasificación, indicará que está DENTRO. Al coger ahora como primer punto W2 y como siguiente punto W3, y ya que se suponen intervalos cerrados, W2 indicará que está

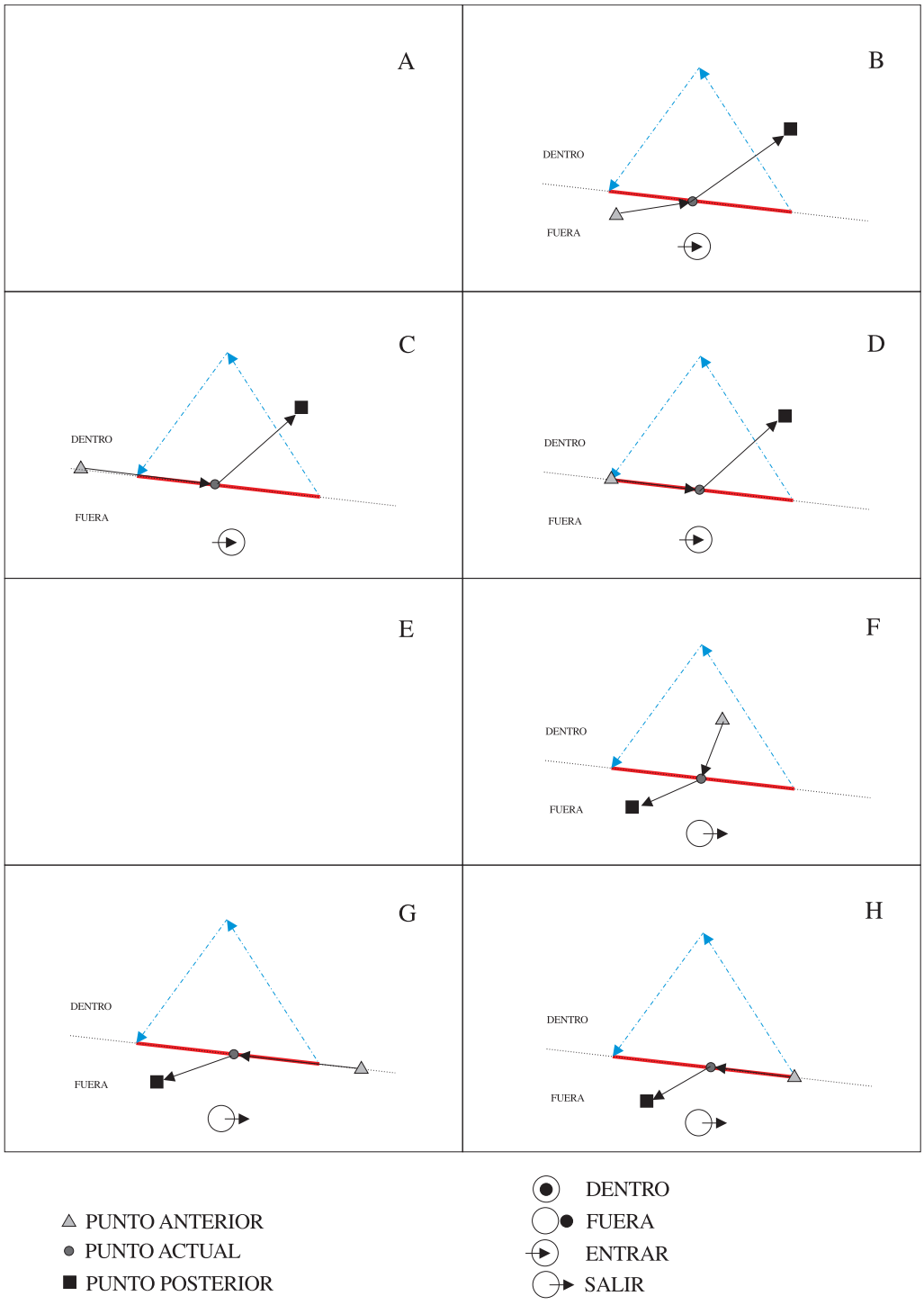


Figura 3.30: Cambios de estado en nuevas intersecciones. Los caso A y E no producen cambios.

DENTRO y W3 FUERA. Por ello se calculará una nueva intersección I1, que coincidirá con W2. Observar que la intersección se calcula para A1. Además a esta nueva intersección habrá que incluirle el atributo de cómo se varía la visibilidad, ésto es, si entra o sale del otro polígono. Ésto

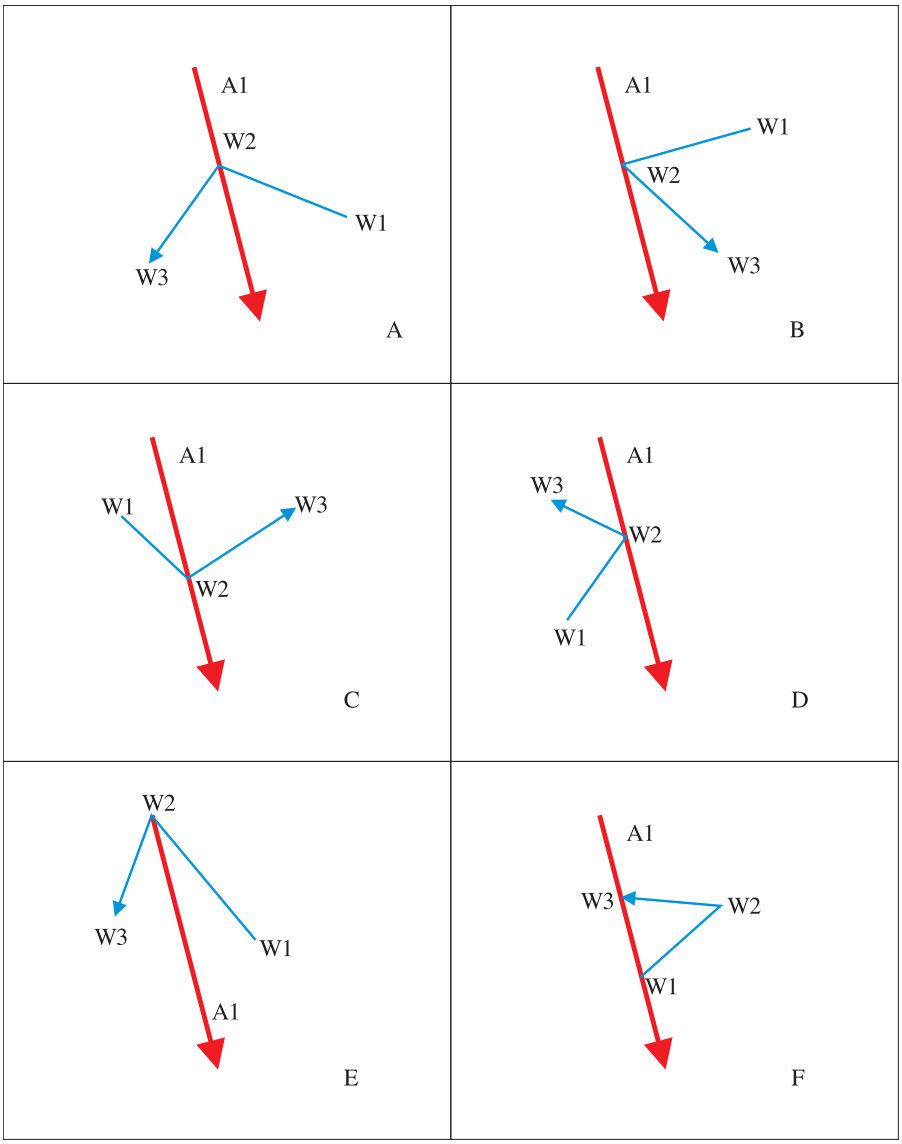


Figura 3.31: Supuesto en el que no existe subclasificación.

no puede ser resuelto si no se recurre al mismo método que el algoritmo propuesto, que consiste en mirar también la posición del punto anterior. Pero hay que incidir en que el punto anterior hace referencia a un vértice constituyente y no a ninguna nueva interseción que se hubiera calculado en pasos anteriores, ya que ésto podría producir una mayor duplicación. En tal caso, si no se distingue que la nueva intersección está alineada, supondría que la clasificación de las intersecciones siempre tendría que recurrir a tres puntos, fuera o no fuera necesario lo cual implica un esfuerzo superfluo. En el caso de que se distinguiera si está alineado, se estaría llegando al camino elegido por el algorimo que se presenta, pero con el añadido del cálculo de una intersección que incluso puede ser inútil. Supongamos que se clasifica correctamente el que el punto sea de entrada o de salida. Si se plantean los casos B y C de la Figura 3.31, a

pesar de la duplicación, el resultado es correcto, pero no así en el caso D. Se ha calculado una intersección y en la mejor de las implementaciones se determina que es inútil. En el algoritmo que se propone, además de no duplicar, primero detecta la posibilidad o no de introducir una nueva intersección y luego se calcula.

En caso de que la intersección se produzca en un vértice, como el caso E de la Figura 3.31, siempre se introduce un punto que es inútil y que no es posible detectar a menos que se determine que hay una coincidencia, lo cual, de nuevo lleva a tener que clasificar los puntos con más posibilidades además de DENTRO y FUERA. Pero el problema más grave surge en el caso F de la Figura 3.31. W1, W2 y W3 indican que están DENTRO lo cual implica que no se calculará ninguna nueva intersección, cuando en realidad se tendrían que introducir dos.

- En el segundo caso [V1, V2) se producen problemas similares, que pueden ser deducidos de lo expuesto en el caso anterior.

Por tanto, se puede deducir que es necesrio poder determinar si un punto está alineado y que tipo de alineación tiene, ya que facilita tanto la detección de nuevas intersecciones como obtener el atributo de cambio de visibilidad. Además no introduce puntos repetidos y el esfuerzo computacional añadido es mínimo: simplemente la detección de distintos estados.

Actualización de datos

Una vez se ha determinado que debido a que existen nuevas intersecciones o existe algún tipo de contacto entre los triángulos, lo primero que se hace es determinar la posición en profundidad de un triángulo con respecto al otro, lo cual dará como resultado que uno estará DELANTE y el otro DETRAS. Con esta información se pasa a la fase de actualización, propiamente dicha, que conlleva las siguientes acciones:

1. Determinar el cambio de visibilidad que se puede producir en los vértices clasificados como ALINEACION_VERTICE.

2. Introducir en la estructura de datos las nuevas intersecciones, incluyendo la información referente a la profundidad.

3. Actualizar los valores, si es necesario, de todos los vértices constituyentes, incluidos los alineados.

Para el cambio en las intersecciones nuevas se usa la información de los puntos referentes a la arista que se está estudiando, mientras que la clasificación que aquí se comenta usa la información obtenida en la primera fase, cuando se obtuvo la posición de los vértices respecto al polígono completo, y además necesita la información que se ha calculado en la fase anterior respecto a la existencia o no de nuevas intersecciones.

También en este caso es necesaria la utilización de la información de posición de los vértices anterior y posterior, pero se añade la obligación de saber si se han producido intersecciones en la arista anterior, la formada por el punto anterior y el actual, y la arista actual, formada por el punto actual y el punto siguiente. Sólo es necesario saber si se ha producido intersección, sirviendo sólo el caso en el que hay una intersección, ya que en el caso de que se produzcan dos, es equivalente, en cuanto a cambios de visibilidad, a que no se hubieran producido. Para esta clasificación, los estados de ALINEACION_DENTRO y ALINEACION_VERTICE se consideran como DENTRO, mientras que el estado ALINEACION_FUERA se considera como FUERA.

La tabla que controla el proceso es la siguiente (V=vértice, I=intersección):

V. anterior	I. anterior	V. actual	I. siguiente	V. siguiente	Estado
FUERA	NO	ALIN_VERTICE	NO	FUERA	DENTRO-FUERA
FUERA	NO	ALIN_VERTICE	SI	FUERA	ENTRAR
FUERA	SI	ALIN_VERTICE	NO	FUERA	SALIR
FUERA	SI	ALIN_VERTICE	SI	FUERA	DENTRO
FUERA	NO	ALIN_VERTICE	NO	DENTRO	ENTRAR
FUERA	SI	ALIN_VERTICE	NO	DENTRO	DENTRO
DENTRO	NO	ALIN_VERTICE	NO	FUERA	SALIR
DENTRO	NO	ALIN_VERTICE	NO	DEMTRP	DENTRO

Se comprueba que de las 32 posibles combinaciones, sólo son válidas las 8 que muestra la tabla. En la Figura 3.32 se pueden ver gráficamente los distintos casos.

Dentro de los posibles resultados que puede entregar dicha tabla observar que hay uno que es DENTRO-FUERA. Esto significa que la posición del vértice o bien será DENTRO o FUERA en función de la posición relativa de un triángulo con respecto al otro. Si el triángulo está delante el estado será FUERA y DENTRO cuando el triángulo esté detrás (Figura 3.33).

Una vez se ha determinado el posible cambio de estado de los vértices coincidentes, se introducen las nuevas intersecciones. Este proceso también se podría haber realizado en el momento que se calculan las intersecciones, pero supondría el conocimiento de la posición en profundidad de un triángulo con respecto al otro. Existen dos motivos para retrasar dicho cálculo a una fase posterior a la detección de las intersecciones, y las dos están relacionadas. La primera es que en el caso de que no hay interacción entre los triángulos se habrá realizado un cálculo, que es inútil. El segundo, y más importante, es que sin poder asegurar que entre ambos triángulos hay algún tipo de interacción, el cálculo de la profundidad, sin un algoritmo bastante más elaborado que el usado, puede entregar resultados incorrectos.

En la fase de actualización de los vértices, sólo se aplica a aquellos en los que se produce un cambio de visibilidad (ENTRAR, SALIR), o en aquellos que están en el interior (DENTRO). Los que están clasificados como FUERA no sufren ningún cambio. El objetivo en el primer caso está claro: indicar que se produce un cambio y de que forma. El segundo caso tiene que ver más con el proceso de visualización, que se expone a continuación.

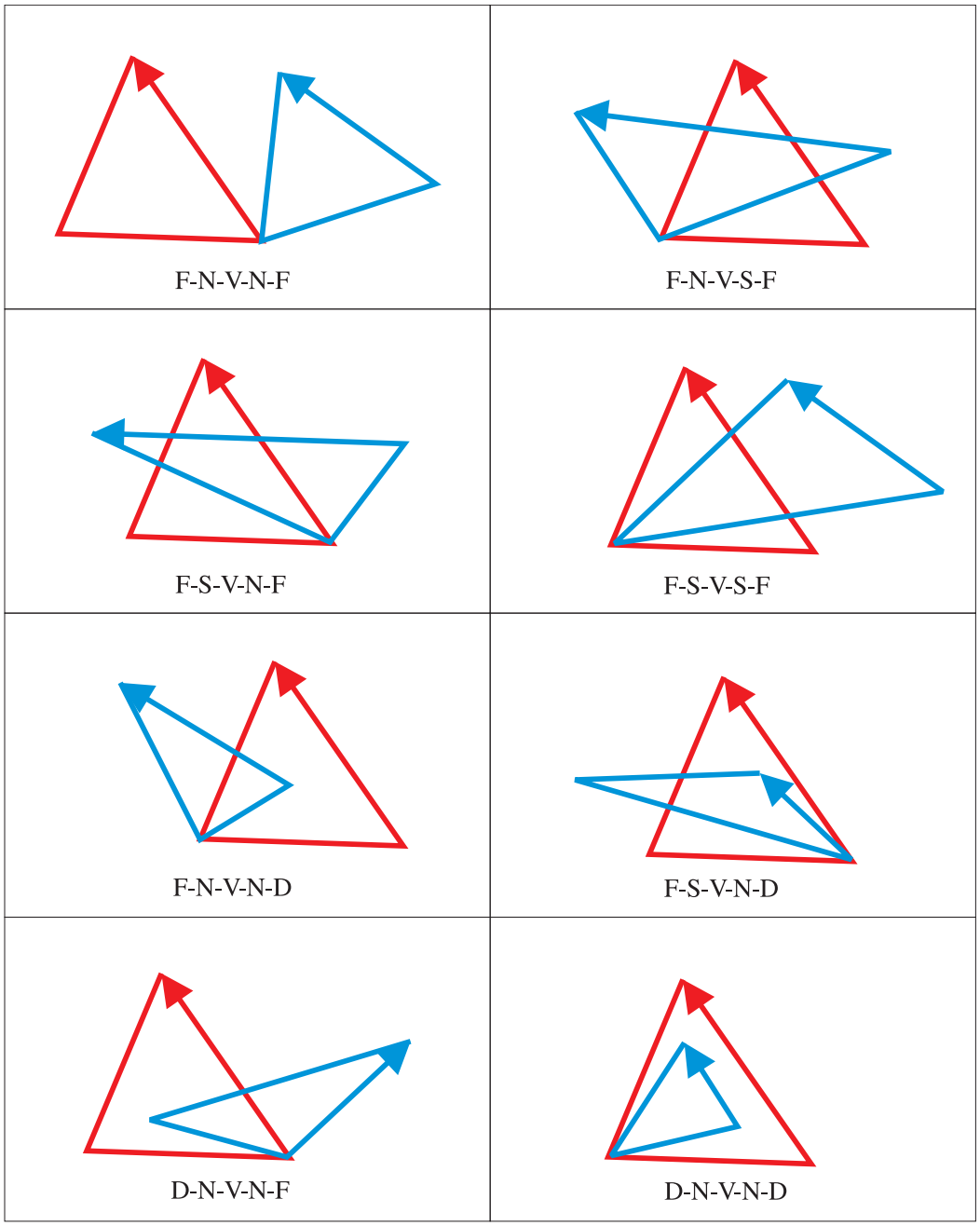


Figura 3.32: Cambios de visibilidad en los vértices coincidentes.

Visualización

Una vez los triángulos se han actualizado pueden ser mostrados. El proceso de visualización se basa en el principio de que dados dos polígonos, con las condiciones que se han exigido en apartados anteriores, si existe algún tipo de

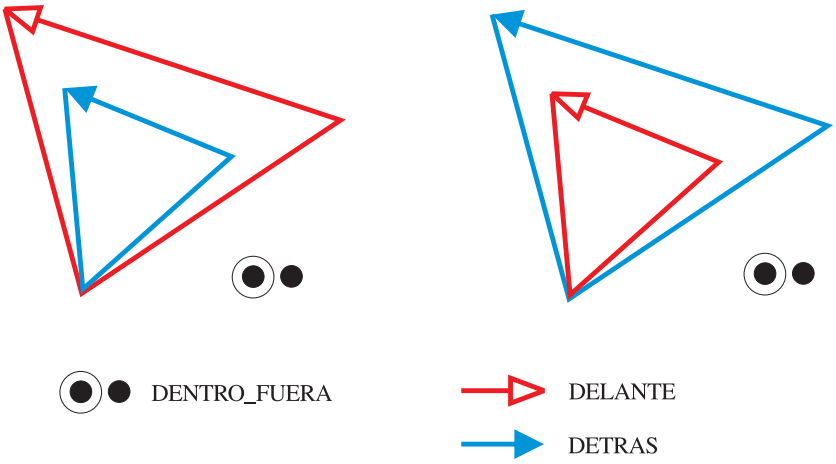


Figura 3.33: Caso en los que el vértice tiene el atributo DENTRO-FUERA.

interacción entre ambos, los cambios de visibilidad de las aristas sólo se pueden producir en las intersecciones, o en caso de que no haya intersecciones, porque los vértices de un triángulo estén contenidos en el otro.

Se parte de una situación en la que ambos triángulos son visibles y por ello los vértices estan marcados como FUERA/DELANTE. Al termino del procesado de ambos triángulos, podrán haber aparecido nuevas intersecciones que indicarán si entran o salen de delante o de detrás, podrán haber vértices coincidentes a los cuales se les aplica lo mismo que a las intersecciones nuevas y podrán haber vértices constituyentes que están en el interior del otro triángulo. Estos vértices estarán marcados como DENTRO, y aunque en ellos no se producen cambios de visibilidad, resulta interesante a la hora de visualizar la silueta, la información de profundidad que puedan aportar. Es decir, el algoritmo de visualización escoge un vértice inicial a partir del cual se va a obtener y dibujar la silueta. Este vértice inicial tiene que ser del tipo de los que no producen cambio de visibilidad, que esté marcado como DENTRO o FUERA. Hay casos en los que siempre se puede encontrar un vértice marcado como FUERA para iniciar la cadena, pero hay otros en los que ésto no ocurre. Por tanto, el algoritmo debe estar preparado para iniciar la cadena tanto con un punto marcado como FUERA, como con un punto que esté marcado como DENTRO. Lo que hace necesario la actualización de los puntos DENTRO es poder saber su posición referente a la profundidad: es necesario saber si dicho punto está DELANTE o DETRAS. Ésto es así porque el algoritmo de visualización usa un esquema de cambio de visibilidad que es relativo, en el sentido de que los vértices indican cambios relativos y no posiciones absolutas. Mientras que en el caso de visualización de dos triángulos tanto el posicionamiento relativo como el absoluto son lo mismo, en un caso en el que existen múltiples niveles de solapamiento, la actualización tanto de vértices constituyentes como de nuevas intersecciones complicaría el procesamiento de los triángulos, cada vez que fueran tratados, mientras que si el posicionamiento es relativo, se deja dicha complejidad al algoritmo de visualización, el cual se aplica una sola vez.

3.3. Animación

La implementación de las Transformaciones No-Lineales Jerárquicas cuando las mismas deben ser aplicadas a un modelo poligonal basado en triángulos no ha representado ningún problema, ya que en sí, el método de las transformaciones no-lineales, y la extensión que proponemos, son independientes de tipo de representación. La única diferencia es que en nuestra implementación se aplica directamente sobre la geometría del objeto en vez de sobre la malla de control de las superficies que definen a dicho objeto. Del conjunto de tipos de funciones que se podrían implementar, por sencillez, pero si perder potencialidad, se decidió implementar los B-splines no racionales y no uniformes, así como el seno y coseno. Los B-splines proporcionan suficiente potencia para la definición de las funciones, de manera sencilla, económica e intuitiva. El hecho de haber implemantado los B-splines y los senos, también muestra como son tratados de manera diferente los parámetros de definición, cuando los mismos se hacen depender de otras funciones.

Las funciones y formatos se exponen con detalle en el Apéndice A, sección A.3

3.4. Arquitectura del sistema

El sistema dispone de dos trayectorias que pueden seguir los elementos geométricos para que sean visualizados. Por una parte el camino normal, y por el otro el que serviría para añadir los trazos.

El sistema ha sido implementado usando C++, y OpenGL para la generación de las imágenes. La estructura de nuestro modelo es mostrada en la figura (Figura 3.34). La línea normal de visualización se usa para mostrar los objetos con coloreado plano o con suavizado de Gouraud. En este caso los objetos se dibujan sin sufrir ningún tratamiento. La segunda línea de visualización es la encargada de producir los trazos. Como se ha visto en los ejemplos previos, los parámetros son controlados mediante ficheros de ordenes (*Scripts*).

OpenGL presenta problemas con las líneas anchas y muy anchas, incluso aunque esté activada el modo *Offset*, ya que el ancho de las líneas es dependiente de la orientación de las caras a las cuales pertenezcan. Debido a que los trazos se calculan en coordenadas normalizadas, y el bafer-z no es lineal en z, hemos tenido que corregir los resultados de nuestra transformación para adaptarla a la de OpenGL.

3.5. Resultados

Para mostrar las posibilidades de las distintas ideas que se han comentado, se han desarrollado varios modelos. Aunque son modelos sencillos, permiten mostrar las posibilidades que tiene la herramienta para la producción de animación bidimensional y de ilustración. Esta sencillez en los modelos se debe,

principalmente a que en el desarrollo de la aplicación se ha primado el poder observar resultados feacientes ha tener un buen interfaz de usuario. Como se ha comentado anteriormente, las definiciones de objetos, luces virtuales y funciones se realiza mediante ficheros, los cuales son creados manualmente, sin ayuda de ninguna herramienta gráfica.

A pesar de la sencillez de los modelos, nos han servido para comprobar que, una vez implementadas, nuestras ideas son correctas. En la Figura 3.36 se pueden ver los distintos modos en que se puede visualizar un modelo (A, alambre; B, trazos; C, color plano; D, bitono; E, suavizado de Gouraud; F, suavizado más brillos especulares). Se han producido animaciones incluyendo los distintos elementos: trazos, luces virtuales y transformaciones no-lineales jerárquicas. En la Figura 3.37 se puede ver un ejemplo de TNLEJ aplicada a el personaje de la casa, relizando un estiramiento y aplastamiento a la vez que una oscilación hacia los lados (no hay aplicadas luces virtuales). En la Figura 3.38 se puede ver al personaje "Brad Pig" corriendo (tiene definidas una luz virtual difusa y otra especular). En las Figura 3.39 y 3.40 se pueden ver fotogramas de una animación, la primera con coloreado plano y la segunda con suavizado de Gouraud. Hay que destacar que usando un Pentium II a 300 Mhz, con 64 Mb de ram, sobre Linux y con Mesa, pero sin aceleración hardware y sin ningún tipo de optimización, con un conjunto de 22000 caras, como el de los fotogramas de las Figura 3.39 y 3.40, se produce una generación de imágenes que permite interactuar con el sistema. En la tabla siguiente se muestran dichos tiempos, para dos resoluciones y distinto número de imágenes, con coloreado plano. Se muestran los tiempos, en segundos, y las imágenes por segundo.

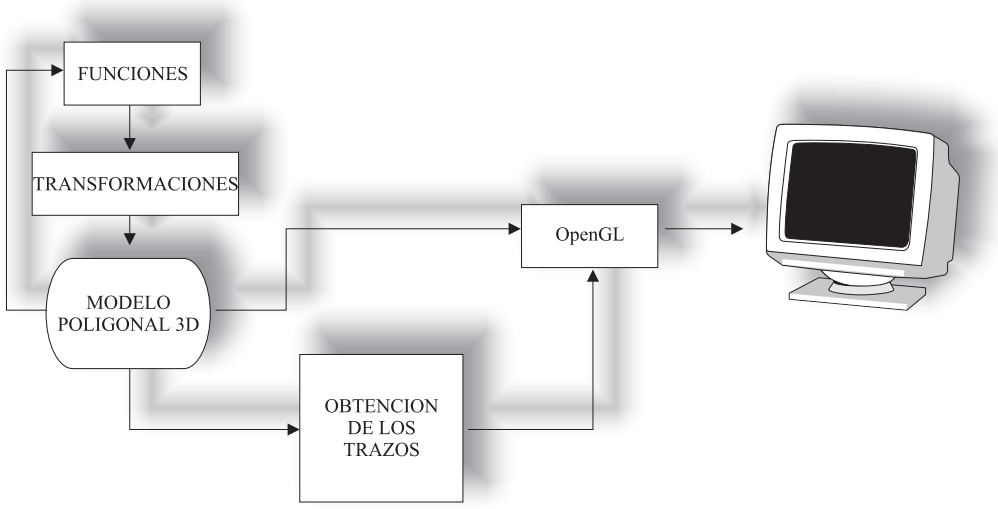


Figura 3.34: Estructura del sistema.

Color plano			
Res \ Ima	**50**	**100**	**200**
400x400	68/0.73	135/0.74	271/0.73
800x800	107/0.46	213/0.46	428/0.46

En la siguiente tabla se muestran los mismos datos pero para el modo con suavizado.

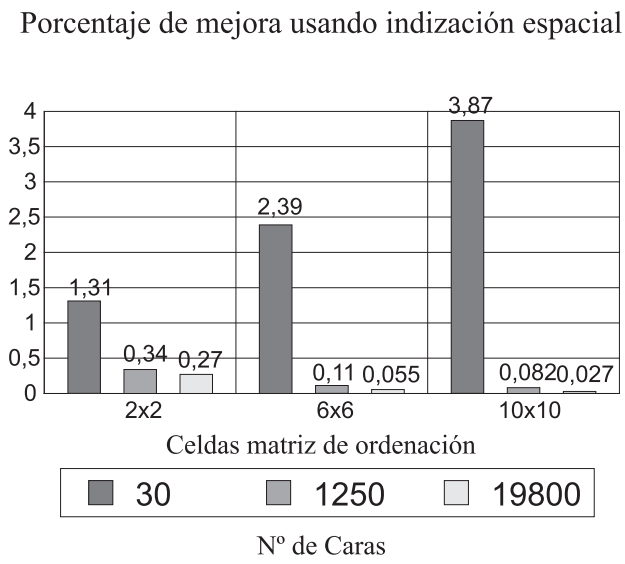
Color suavizado			
Res \ Ima	**50**	**100**	**200**
400x400	71/0.70	143/0.69	286/0.69
800x800	111/0.45	222/0.45	444/0.45

Estos tiempos engloban el procesado de los trazos y las luces virtuales, así como la visualización que es realizada, en su mayoría, por software. Para la misma escena considerada anteriomente se ha tomado un tiempo de 62 segundos para el cálculo de 100 imágenes, excluyendo la visualización. Ésto da una razón de 1.61 imágenes por segundo. Debido a la interrelación de los distintos componentes del programa no se ha podido aislar completamente la parte gráfica de la de cálculo. Por tanto, los tiempos tomado pueden considerarse como cotas superiores.

En la Figura 3.41 se puede ver un ejemplo de las deformaciones que se le pueden aplicar a un cilindro, que simula una pata del cerdo. En la Figura 3.42 se pueden ver fotogramas de una animación en la cual se han definido varios objetos, cada uno con sus luces virtuales, para obtener los trazos cuando el usuario lo ha definido. Por último, en las Figuras 3.43 y 3.44 se pueden ver fotogramas de animaciones con dos personajes, "Poly el marciano" y "Roket E.". Se pueden apreciar las capacidades de modelado y de expresividad.

3.5.1. Estudio de la optimización con indización espacial

Dado que se trabaja con caras triangulares que están en coordenadas de dispositivo normalizado, el proceso de cálculo de intersecciones mira cada cara con el resto de las caras para comprobar si hay alguna intersección entre las aristas. El problema, como se ha comentado, es de complejidad cuadrática. Una manera de intentar reducir la misma es buscar un esquema de indización espacial, de partición del espacio. La idea se basa en que, en general, las intersecciones de una cara que aporta trazos con otras, se producen de forma localizada, lo cual lleva a la búsqueda de esquemas que intente clasificar los elementos de forma que se pueda determinar su posición y las relaciones entre las distintas zonas. La posibilidad más inmediata es la de particionar el espacio del dispositivo normalizado en una malla de tal forma que los triángulos se posicionen en las distintas celdas de la malla. Un triángulo podrá ocupar una o más celdas, todas aquellas sobre las que haya parte de la cara. Una vez



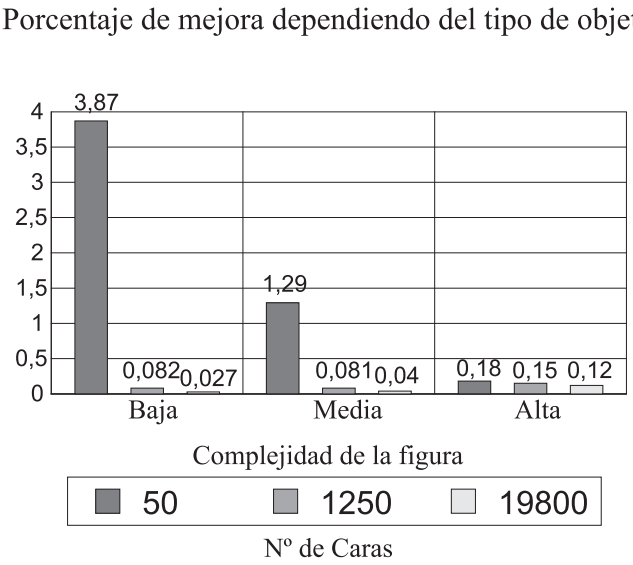


Figura 3.35: Resultados de la aplicación de la partición espacial.

que se ha realizado esta clasificación previa, dada una cara, sólo se tiene que comparar con las caras que ocupen las mismas celdas que la propia cara que está siendo estudiada. Ésto puede hacer que se intente realizar el cálculo entre dos caras varias veces porque las mismas estén en varias celdillas a la vez. Para evitar ésto, cada vez que se haya realizado la intersección entre dos caras, dicha operación deberá ser marcada para que no se vuelva a repetir, o buscar algún mecanismo para que, en caso de que se aplique el cálculo de las intersecciones, los resultados no se produzcan por duplicado.

Tenemos que hacer un estudio referente al tamaño más eficiente de la malla con respecto a un tamaño dado de modelo. Este estudio es parecido a lo que existe sobre el báfer-z, en el sentido de que cuanto más caras, las mismas

son más pequeñas, por tanto, ocupan menos celdas, y, por tanto, se reduce el número de celdas entre las cuales se tiene que realizar la búsqueda. Se puede establecer que debe haber una relación del tipo aproximadamente lineal que relaciona el tamaño de la malla y el tamaño del modelo, el número de caras del modelo.

Esta posibilidad de optimización ha sido parcialmente implementada, y los resultados han sido muy esperanzadores. En la Figura 3.35 se muestran los resultados obtenidos. Los datos que se muestran son relaciones entre el número de comaparaciones que se harían normalmente y las que se hacen aplicando la partición del espacio (un valor 1 indica que no hay mejora, menor que 1 que la hay y mayor que 1 que no la hay). En la primera gráfica se muestran los datos para un mismo objeto a distintas niveles de detalle (más o menos caras) relacionados con el tamaño de la malla de división del espacio. En la segunda gráfica se muestra los datos para distintos tipos de objetos (más o menos complejos), para una malla de 10x10, siendo el de dificultad alta una superficie abierta generada aleatoriamente. Como se puede observar, se han alcanzado, con los modelos de los ejemplos y en situaciones relativamente simples, mejoras de un 98 % en el tiempo con mallas de 10x10.

Capítulo 4

Conclusiones

En esta memoria se ha presentado un formalismo, el de las luces virtuales, cuya implementación permite obtener una de las características más importantes de las imágenes de apariencia bidimensional: los trazos. Se ha implementado un sistema que usa las *Luces Virtuales* para definir y obtener los trazos externos e internos. El método es muy sencillo y se puede aplicar no sólo a modelos poligonales sino a otro tipo de modelos geométricos.

Se ha dado una definición de los trazos internos y externos, los elementos más importantes en la creación de dibujos de aspecto bidimensional. Los trazos han sido definidos formalmente tanto para un modelo geométrico genérico como para el poligonal, el que se ha seleccionado para la implementación del sistema. Descrito lo que son, se han expuesto las características que permiten obtenerlos. Aunque ambos timpos de trazos tienen ciertas características que los diferencian, se ha desarrollado un formalismo que permite englobarlos: *Luces Virtuales*. La detección y clasificación de los trazos se basa en la aparición de diferencias entre características visuales del objeto, las cuales dependen de las luces virtuales. Este mecanismo replica el usado por los dibujantes humanos. La implementación de este formalismo no sólo permite la detección de los trazos sino que permite su definición por parte del usuario de forma fácil e intuitiva.

Ya que se pueden producir intersecciones entre los trazos, las cuales afectan en la forma en la que serán visualizados, es necesario calcularlas. Para el cálculo de las intersecciones entre los trazos se ha propuesto un nuevo algoritmo con amplias posibilidades entre las que se encuentra la eliminación de líneas ocultas, extendiendo el concepto de *invisibilidad cuantitativa* de Appel, en el sentido de que también controla cuantas veces una arista aparece delante de las caras. Actualmente, aunque nuestro método no es tan rápido como el de Markosian, presenta ventajas interesantes: es completo, siempre encuentra los trazos externos, permite formar cadenas que no están totalmente incluidas en el volumen de visión y mantiene los atributos a lo largo de las cadenas de trazos, sin que haya ruptura; la cantidad de visibilidad añade flexibilidad en la visualización de los trazos, al poseer más información que únicamente si la arista está ocluida.

Se ha expuesto un algoritmo que permite calcular la silueta de dos trián-

gulos, sean cuales sean sus posiciones. Partiendo de que este algoritmo se considera un elemento básico para la implementación de procesos más complejos, se ha visto como, a pesar de que en principio la solución parecía muy simple, aparecen muchos casos que lo complican. En particular, se ha mostrado la especial dificultad que entraña los casos en los cuales se producen coincidencia de vértices o de vértices con aristas.

La solución que se ha aportado al problema no está optimizada, admitiendo diversas modificaciones para mejorar su rendimiento tanto en velocidad como en espacio. Otro punto que permite ser mejorado o modificado es el del tipo de polígonos con los cuales puede ser usado este algoritmo.

La necesidad de poder transformar los modelos sólidos de tal forma que los mismos adquieran las características propias de la animación y de la simulación de entidades naturales, lleva a la búsqueda de métodos que permitan dichas transformaciones. Frente al uso extendido, debido a su potencia, de la Deformación Libre de Forma, se proponen las Transformaciones No-Lineales Extendidas Jerárquicas en la mayoría de las situaciones en las que la transformación es menos localizada, con una gran economía en la forma de representación de dichos cambios. Se ha desarrollado un sistema para la producción de animación en el cual se ha incluido el mecanismo de las transformaciones locales jerárquicas para la deformación de los modelos. Los resultados obtenidos son óptimos y consideramos que frente al control más localizado que permiten las DLF, las TNLEJ permite definir deformaciones de una forma sencilla y económica. La utilización de un eje de aplicación en conjunción con la posibilidad de jerarquizar las transformaciones, combinando transformaciones constantes y no-lineales, abre nuevas posibilidades. El método no sólo puede ser usado como herramienta de modelado, debido a las capacidades de modificación de la geometría

El método de las TNLEJ permite obtener deformaciones estáticos. En base al mecanismo que se ha usado para su implementación se ha desarrollado un método basado en el uso de funciones autorreferentes que permiten hacer variar en el tiempo la función de control de la TNLEJ, la que define la deformación. El mecanismo ha resultado a la vez sencillo y extremadamente potente a la hora de definir movimientos, bajo un control total del usario.

Todos los elementos comentados han sido implementados en un sistema que permita mostrar las capacidades y limitaciones de las ideas propuestas. Los resultados que se han obtenido han sido muy satisfactorios, como se puede comprobar en las imágenes en color. La herramienta permite ser usada para la producción de animación bidimensional con personajes, pero también para la ilustración y el dibujo técnico descriptivo.

Las aportaciones más relevantes son:

- Definición de los trazos (Capítulo 2, Pág. 45).
- Implementación de un algoritmo de generación de trazos para modelos poligonales (Capítulo 3, Pág. 67).
- Creación y definición del modelo de *Luces Virtuales* (Capítulo 2, Pág.

48).

- Implementación del modelo de *Luces Virtuales* para modelos poligonales (Capítulo 3, Pág. 75).
- Un nuevo algoritmo de cálculo de intersecciones entre triángulos, basándose en la *visibilidad cuantitativa* (Capítulo 3, Pág. 77).
- Definición de las Transformaciones No-Lineales Extendidas Jerárquicas (Capítulo 2, Pág. 52), extensión del método de las Transformaciones No-Lineales, generalizándolo para el uso de jerarquías.
- Caracterización de la animación mediante el uso de funciones aplicadas a las funciones de control de las Trasformaciones No-Lineales Extendidas Jerárquicas (Capítulo 2, Pág. 60).
- Implementación de un sistema que permite crear animaciones de aspecto bidimensional.

4.1. Futuros desarrollos

El sistema que se ha desarrollado para comprobar las capacidades de las ideas planteadas, ha demostrado que los objetivo iniciales se han alcanzado. Pero ésto no significa el final de la línea de investigación, todo lo contrario, ha planteado nuevas ideas y posibles desarrollos, así como mejoras del mismo.

Entre las posibilidades de mejora están las siguientes:

- Partición del espacio
  Terminar de desarrollar lo expuesto en el capítulo anterior.
- Coherencia temporal-espacial
  Otra de las posibilidades de optimización que se nos plantea es la de la reutilización de información en el tiempo. Para ello, nos basaremos en trabajos previos [29, 42].
- Reducción del número de caras
  En conjunción con la posibilidad anterior, la de establecer distintos niveles de detalle (*LoD*), presenta aspectos muy interesantes. Algunos trabajos importantes son [20, 22].
- Otras mejoras
  Otra posibilidad de mejora sería la de optimizar los datos que se entregan a OpenGL. OpenGL trabaja de forma más rápida cuando se le entregan conjuntos de triángulos en vez de triángulos sueltos. Ejemplo de trabajo que ha tratado el tema es [63].

  Mejora de la interfaz de usuario del sistema desarrollado para los distintos modulos: modelador, luces virtuales, funciones.

Una vez comentadas las posibles mejoras a nivel de utilización, se exponen las líneas que se van a seguir.

Una primera línea es el desarrollo de un espacio mixto 3D+2D. El problema de la producción de líneas con "carácter" está muy limitado con OpenGL, Un espacio mixto que mezcle las capacidades del espacio 3D, para trabajar con el modelo, escenarios, iluminación, luces virtuales, etc, y un espacio 2D que permitiera trabajar con los trazos sin tener que preocuparse de las profundidades ni de las orientaciones, puede ser la solución.

Otra línea apunta al desarrollo de las TNLEJ. Las TNLEJ admiten nuevas posibilidades de desarrollo. Actualmente se está investigando la posibilidad de alcanzar un control más local mediante el uso de jerarquías espaciales o división espacial.

Apéndice A

Implementación de los formatos de datos

En este apéndice se van a exponer los formatos de datos de los distintos elementos que se han usado en la implementación del sistema. Además, para aclarar conceptos, se mostrará un ejemplo de una Trasnformación No-lineal Extendida Jerárquica.

La información necesaria para el funcionamiento del programa se ha estructurado en tres ficheros: fichero de los modelos, fichero de las funciones globales y fichero de la cámara. A continuación se exponen los formatos de cada uno de ellos, explicando sus contenidos

A.1. Formato de los modelos

En el fichero de los modelos, se almacena la información de cada actor o personaje que va a formar parte de la escena. Para cada personaje se tienen que definir sus distintas componentes y para cada componente hay que definir su forma y atributos.

El formato del fichero de modelos es el siguiente:

Elemento_1
Elemento_2
⋮
Elemento_N

Para cada elemento se tiene el siguiente formato:

[
TIPO
NOMBRE
Definición de la geometría

Atributos
Transformaciones constantes
Definición de luces virtuales
Definición de funciones locales
Definición de TNLEJ
]

Para separar cada elementos se usan corchetes. El formato de cada componente es el siguiente:

- *TIPO:* SMANUAL | SBIPARAMETRICA | SBILINEAL | MONTANIA_FRACTAL | SBSPLINE | SREVOLUCION
  Estos son los distintos tipos de onjetos que se pueden modelar.
- *NOMBRE:*
  El nombre del objeto
- *Definición de la geometría:*
  Aquí se define los parámetros que permitirán definir la geometría del objeto, los cuales variarán dependiendo del tipo que sea. El formato para cada tipo se verá a continuación.
- *Atributos*
  El comienzo de la definición de los atributos se indica con: * ATRIBUTOS. En la siguiente línea se definen los parámetros:
  - ILUMINACION: SI | NO
    Indica si se va a aplicar o no el modelo de iluminación.
  - MODO_SUAVIZADO: SUAVIZADO | PLANO
    Indica si se va a relizar suavisado (de Gouraud) o no.
  - COLOR_RELLENO: (x, y, z) siendo $x, y, z \in R$ y $0 \leq x, y, z \leq 1$
    Indica el color de relleno cuado no se aplica el modelo de iluminación. Se define el color mediante el modelo RGB.
  - MATERIAL: *ambiental, difusa, especular, exponente*
    Se indican las componentes ambiental, difusa, especular y el exponente para el modelo especular, para el caso de que se aplique el modelo de iluminación. Los colores se definen mediante el modelo RGB.
    ambiental, difusa, especular: (x, y, z) siendo $x, y, z \in R$ y $0 \leq x, y, z \leq 1$
    exponente: $\forall x \in R$ tal que $0 \leq x \leq \infty$
  - OPACIDAD: x tal que $x \in R$ y $0 \leq x \leq 1$
    Indica el grado de opacidad o trasparencia. Un objeto es totalmente transparente si el coeficiente es 0 y totalmente opaco si es 1.
  - GROSOR_LINEA: *grosor_fino, grosor_grueso*
    Indica los grosores que tendrán las líneas cuando se dibujen con el atributo fino y el atributo grueso. En la actual implementaci'on sólo

se disponen de dos niveles de grosor.
grosor_fino, grosor_grueso: x tal que $x \in N$ y $0 \leq x \leq MaxOGL$

- *Transformaciones constantes*
  En este apartado se definen transformaciones que se desea aplican antes que ninguna otra y sin usar el mecanismo de las TNLEJ. Las trasnformaciones se aplican en el orden en el cual son definidas. Las posibilidades son las siguientes:

  - Traslaciones
    TRASLACION:(δx, δy, δz)
  - Escalados
    ESCALADO:(S_x, S_y, S_z)
  - Rotaciones
    ROTACION_EJE_X: ángulo (en grados)
    ROTACION_EJE_Y: ángulo (en grados)
    ROTACION_EJE_Z: ángulo (en grados)

- *Definición de luces virtuales*
  El comienzo de la definición de las luces virtuales se indica con: * LUCES. El formato de las luces virtuales se expone con detalle en la sección A.2.

- *Definición de funciones locales*
  El comienzo de la definición de las funciones se indica con: * FUNCIONES. El formato de las funciones, tanto locales como globales, se expone con detalle en la sección A.3.

- *Definición de TNLEJ*
  El comienzo de la definición de las TNLEJ se indica con: * TNLEJ. El formato de las Transformaciones No-Lineales Extendidad Jerárquicas se expone con detalle en la sección A.4.

A continución se exponen en detalle los formatos de defincicón de los distintos tipos de objetos que se pueden crear en la implementación actual. Aunque las capacidades son limitadas, han permitido crear ciertos personajes que sirven de muestra de las posibilidades del método.

- SMANUAL
  Una superficie manual es una malla de mxn puntos, dispuestos en forma matricial. A diferencia de las siguientes superficies los vértices son definidos manualmente.

  El formato es el siguiente:

  - NUM_PUNTOS_U: número de puntos
  - NUM_PUNTOS_V: número de puntos
  - PUNTOS_DE_CONTROL: (x, y, z), ...

- SBILINEAL
  Este descripción define una superfice bilineal mediante cuatro vértices. Se puede indicar el número de divisiones que se desea para la superficie.

  El formato es el siguiente:

  - NUM_DIVISIONES_U: número de divisiones
  - NUM_DIVISIONES_V: número de divisiones
  - PUNTOS_DE_CONTROL: $(x, y, z), \ldots$

- SBIPARAMETRICA
  Esta superficie es una superficie reglada, se define dos curvas entre las cuales se interpola linealmente. Las curvas se definen como B-splines, por lo que hay que definir el orden y el tipo, si es uniform o no. Las curvas se dan en el paraámetro u.

  El formato es el siguiente:

  - NUM_PUNTOS_U: número de puntos
  - NUM_PUNTOS_V: número de puntos
  - NUM_DIVISIONES_U: número de divisiones
  - ORDEN_U: orden de la curva
  - TIPO_CURVA_EJE_U: UNIFORME | NO_UNIFORME
    Indica el tipo de curva.
  - PUNTOS_DE_CONTROL: $(x, y, z), \ldots$

- MONTANIA_FRACTAL
  Ésta es una superficie bilineal en la cual las alturas se calculan de forma aleatoria.

  El formato es el siguiente:

  - NUM_DIVISIONES_U: número de divisiones
  - NUM_DIVISIONES_V: número de divisiones
  - ALTURA: altura máxima
  - PUNTOS_DE_CONTROL: $(x, y, z), \ldots$

- SBSPLINE
  En una superficie B-spline, la forma de la misma se define mediante una malla de control.

  El formato es el siguiente:

  - NUM_PUNTOS_U: número de puntos
  - NUM_PUNTOS_V: número de puntos
  - NUM_DIVISIONES_U: número de divisiones
  - NUM_DIVISIONES_V: número de divisiones
  - ORDEN_U: orden de la curva en u
  - ORDEN_V: orden de la curva en v

- TIPO_CURVA_EJE_U: UNIFORME | NO_UNIFORME
  Tipo de la curva en u.
- TIPO_CURVA_EJE_V: UNIFORME | NO_UNIFORME
  Tipo de la curva en v.
- PUNTOS_DE_CONTROL: $(x, y, z), \ldots$

- SREVOLUCION
  Esta superfice se genera por la rotación de un perfil sobre un eje. Tiene dos posibilidades: que la poligonal que define el perfil, se tome como curva generadora, o que la poligonal se tome como poligono de control de una curva B-spline que define la curva generadora. Veamos los dos casos. Para indicar que la poligonal se toma como tal o como puntos de control se hace mediante la variable que indica el número de puntos en el parámetro u: si es 0 se toma como poligonal y si es distinto de 0 indica que hay que tomalo como la poligonal de control de un B-spline.

  El formato es el siguiente:

  - NUM_PUNTOS_U: 0 | $x \neq 0$
  - NUM_PUNTOS_V: número de puntos
  - NUM_DIVISIONES_U: número de divisiones
  - NUM_DIVISIONES_V: número de divisiones
  - ORDEN_U: orden de la curva en u
  - ORDEN_V: orden de la curva en v
  - TIPO_CURVA_EJE_U: UNIFORME | NO_UNIFORME
    Tipo de la curva en u.
  - TIPO_CURVA_EJE_V: UNIFORME | NO_UNIFORME
    Tipo de la curva en u.
  - EJE_REVOLUCION: EJE_X | EJE_Y | EJE_Z
    Indica sobre que eje se va a revolucionar el perfil.
  - PUNTOS_DE_CONTROL: $(x, y, z), \ldots$

A.2. Formato de las luces virtuales

Veamos a continuación el formato de los datos que son necesarios para codificar las distintas luces virtuales de un objeto. Dada la siguiente información que pertenece a un objeto, la parte que define las luces virtuales se define como:

El formato es:

tipo_de_luz, coordenadas, tipo_de_posición, movilidad

con:

- *tipo_de_luz*: DIFUSA | ESPECULAR
  La luz virtual puede tener asociada un modelo de iluminación difuso o especular.

- *coordenadas*: (x, y, z)
- *tipo_de_posición*: POSICIONAL | DIRECCIONAL
  Esta variable indica la localización de luz virtual, pudiendo estar en el infinito, y en tal caso las coordenadas representan el vector dirección de la luz (el otro vértice es el origen). Si no está en el infinito, las coordenas indican la posición en coordenadas de mundo.
- *movilidad*: SI | NO
  Esta variable indica si la luz virtual puede moverse o no con respecto al objeto al que se asocia. En general son estáticas.

Una vez están definidas las luces, hay que definir las condiciones que controlan su aplicación. Para ello hay que dar las condiciones individuales de cada cara, y la condición conjunta de ambas, tanto a nivel de normales como de reflexión. Un ejemplo de las condiciones para la primera luz sería:

El formato es:

cara: valor_1, valor_2, condición, exponente

con:

- *cara*: CARA1 | CARA2
  Para cada una de las dos caras que son compartidas por una arista se definen las condiciones deseadas.
- *valor_1, valor_2*: $\forall x \in R$
  Valores númericos de las condiciones.
- *condición*:
  LI=limite inferior
  LS=limite superior
  operaciones: >, >=, <, <=, !=, =, NINGUNA
  Variable que indica la operación que se va aplicar que debe cumplir la normal de la cara.
- *exponente*: $\forall x \geq 0 \in R$

La operación conjunta se define con:

El formato es:

CONJUNTA: valorN_1, valorN_2, condición_normales, relación, operación_reflexión, valorR_1, valorR_2, condición_reflexión

con:

- *valorN_1, valorN_2, valorR_1, valorR_2*: $\forall x \in R$
- *condición_normales, condición_reflexión*:
  LI=limite inferior
  LS=limite superior
  operaciones: >| >= | <| <= | != | =

- *relación*: CARA1_MENOR_CARA2 |
  CARA1_MAYOR_CARA2 |
  CARA1_IGUAL_CARA2 |
  INDISTINTA
- *operación_reflexión*: ABS(ABS(CARA1)+ABS(CARA2)) |
  ABS(ABS(CARA1)-ABS(CARA2)) |
  ABS(ABS(CARA2)-ABS(CARA1)) |
  ABS(CARA1+CARA2) | ABS(CARA1-CARA2) |
  ABS(CARA2-CARA1) | CARA1+CARA2 |
  CARA1-CARA2 | CARA2-CARA1

Primero se opera con las normales, mediante la relación y la condición con sus valores. Después se usa la operación con las reflexiones, cuyo resultado debe cumplir la condición, fijada por los límites. Se ven aquí reflejadas las tres operaciones que se pueden ver en la Figura 3.11.

Por último se realiza la operación global. La operaciones se relacionan posicionalmente con cada luz virtual definida.

El formato es:

condición OPL condición OPL ...

con:

- *condición*: REFLEXION_SI | REFLEXION_NO
  Para cada luz virtual, por posición, se indica si la arista ha sido seleccionada (REFLEXION_SI) o no (REFLEXION_NO).
- *OPL*: O | Y
  Las operaciones permitidas son un O y un Y lógicos.

Esta definición de las luces virtuales se debe aplicar a cada uno de los objetos, y es importante definir la posición, el tipo y las condiciones de las mismas, pues dependiendo de todos estos parámetros las aristas de objeto serán o no trazos.

A.3. Formato de las funciones

Una vez que se ha descrito el funcionamiento de las Transformaciones No-Lineales Extendidas Jerárquicas en el capítulo anterior, la implementación es bastante directa. Como se expuso, todo el mecanismo de las TNLEJ se basa en funciones. Además, en la definición de las funciones se puede hacer referencia a otras funciones. Por ello, se tuvo que implementar un mecanismo de referencia de las funciones y sus parámetros, el cual se explicó. Pasamos a ver la implementación concreta. El formato para la definición de la funciones es la siguiente:

número_función, clase_función: tipo_función, abscisas, ordenadas, parámetros

con:

- *número_función*: el identificador de la función.
- *clase_función*: FUNCION | FUNCION_D
  Esta variable indica el tipo de función, si es normal (FUNCION), o sus abcisas son dependientes de otra función (FUNCION_D).
- *tipo_funcion*: SENO | COSENO | BSPLINE | BSPLINE_X. Estos son los tipos de funciones que se han implementado.
- *abscisas*: número de función | $Min_{abscisa}$, $Max_{abscisa}$
  si el tipo de función es FUNCION_D, las abcisas vienen indicadas por un número de función, el cual es descrito de la forma FUN_X, siendo X el número de la función que proporcionará el valor de la variable independiente, y también proporciona los valores límites de las abscisas, las cuales pasarán a ser las ordenadas de la función con la que se establece la relación. En caso de que el tipo FUNCION, se tendrán que definir los dos valores para las abscisas, los cuales podrán ser valores constantes o alguna de las siguientes definiciones (MIN_X, MAX_X, MIN_Y, MAX_Y, MIN_Z, MAX_Z), las cuales son alias de los valores de los valores de la caja englobante del objeto.
- *ordenadas*: $Min_{ordenada}$, $Max_{ordenada}$
  Pueden tomar valores constantes, los valores de la caja englobante.
- *parámetros*: Para cada tipo de función, los parámetros que permiten definir su comportamiento.

Veamos como es la definición más concreta de los distinto tipo de funciones:

- **SENO**
  Los parámetros que definen un seno son los siguientes:

  amplitud, frecuencia, desfase y desplazamiento.

  La función coseno se implementa como un seno con un desfase de 90 grados. Es importante observar que todas las funciones se definen dando un rango para las abcisas y otro para las ordenadas. La función seno está definida por defecto entre 0 y 2π. Si se deciden poner unas abcisas diferentes a dichos valores, se indica mediante una bandera (*Flag*) que habrá que realizar un ajuste lineal para relacionar los dos pares de valores. Ésto es, si se define que las abscisas irán entre 0 y 1, se realiza un ajuste lineal que relacione el valor 0 con 0 y 1 con 2π. Para las ordenadas no se realiza ningún ajuste, ya que el mismo se alcanza mediante los párametros de defínción del seno, más particularmente con la amplitud.

  Animación
  La posibilidad de variar en el tiempo la función seno se alcanza haciendo que alguno, varios o todos los parámetros que lo definen no sean constantes sino que a su vez dependan de otras funciones que los hagan evolucionar. Por ejemplo, se puede hacer que la amplitud varie en el tiempo siguiendo una curva cuadrática.

- **BSPLINE**
  Los parámetros que definen un B-spline son los siguientes:

  orden, número de puntos, lista de puntos 3D (x, y, z).

  Dado que hay que definir unas abscisa y unas ordenadas, hay que buscar que valores del B-spline se pueden relacionar con las mismas. En un B-spline, definido el rango del parámetro, al dar distintos valores al mismo se obtienen puntos de la curva, en este caso tres coordenadas, (x, y, z). Nosotros estamos interesados en curvas planas, por lo cual la coordenada z de los distintos puntos de control es 0. Nos queda, por tanto, dos coordenadas para elegir, x e y. Por la forma en que se definen las funciones se ha decidido tomar como valor el de la coordenada y. Por tanto, cuando se da un valor a un B-spline, el mismo se evalua y devuelve la coordenada del punto obtenido. El problema de la relación entre los rangos se resuelve considerado como las abscisas de un B-spline, el rango del parámetro. Si las abcisas que se definen no coinciden con el rango del parámetro, se resuelve con un ajuste lineal, como se ha visto para el seno. El caso de las ordenadas es un poco más complejo. En función del orden del B-spline, la curva tiene una forma u otra. Para el caso de que el orden sea 2, grado 1, el B-spline coincide con el polígono de control. En tal caso, se pueden establecer unos valores mínimo y máximo que puede tomar la curva, que vienen definidos por los valores de la caja englobante del polígono de control (los valores máximo y mínimo de las y). Si estos valores no coinciden con los definidos, se recurre a un ajuste lineal. En caso de que el orden es superior o si a diferencia de lo que se ha considerado hasta ahora, el B-spline es uniforme la solución pasaría por evaluar el B-spline, encontrando sus valores máximo y mínimo y aplicar el mecanismo de ajuste lineal. En la actual implementación se ha simplificado haciendo que, independientemente del orden de la curva, se tomen los valores de la caja englobante del polígono de control.

  Animación
  La forma de hacer variar los B-splines en el tiempo es hacer que las ordenadas puedan depender de otras funciones que hagan variar el rango de aplicación (Figura 2.15). Por ejemplo, el extremo mínimo podía seguir una función constante, mientras que el rango máximo podría variar siguiendo un seno. Aunque es un mecanismo sencillo, y restrictivo, se ha demostrado que, para ciertas transformaciones, da la suficiente capacidad expresiva. De todas formas, se observó que para obtener toda la potencialidad necesaria en las deformaciones variando en el tiempo, no era suficiente, por lo que se relizó una extensión: los BSPLINE_X.

- **BSPLINE_X**
  Los parámetros que definen un bspline_x son los siguientes:

  orden, número de puntos, lista de puntos 3D.

  La gran diferencia con respecto a la curva anterior, es que los puntos que definen el polígono de control del B-spline no son estáticos, sino que se definen de la siguiente manera:

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

Ésto quiere decir que el valor de la coordenada y depende a una función recursiva. Esto hace que el punto se desplace perpendicularmente al eje de las x según viene determinado por la función. Esta limitación es una simplicación en la implementación, no del método. Obsérvese que de esta mánera la forma del spline viene determinada por los puntos de control cuyas coordenas varían dependiendo de otras funciones. Una característica de este método es que la superficie que se define no es una superficie B-spline, ya que las funciones que controlan a los puntos de control no tienen por qué ser del mismo tipo. Ésto da una gran flexibilidad en el diseño de la modificación de la forma de la curva, manteniendo una economía de representación. En el resto de los elementos de un BSPLINE_X son iguales que un BSPLINE.

Veamos un ejemplo. Queremos obtener una función que dado un valor entre 0 y 1 nos entregue un valor de ángulo entre 0 y 360 grados. Puesto que sólo tenemos definidas las funciones seno y coseno, se puede hacer la siguiente dependencia:

$\alpha = f(seno(\beta);;0,360)$ y $\beta = f_{basica}(y = x; 0,1; 0,2\pi)$

Las dependencias recurrentes en el caso de las Transformaciones No-Lineales Extendidas Jerárquicas tiene una estructura definida: los parámetros de las funciones de deformación dependen de otras funciones recurrentes, las cuales, en general y excepto que haya algún reajuste intermedio, dependen de una función final que se considera como un reloj-maestro. Las relaciones de dependencia son: Función_deformación → Funciones_animación → Función_reloj-maestro.

En realidad todas las funciones pueden ser definidas como finales, pero se ha concretado para su uso en funciones temporales. Para hacer funcionar el sistema existe un procedimiento que se llama reloj-maestro al cual se le indica el número de tic de reloj que se desean obtener (equivalente al número de imágenes). Este procedimiento es el encargado de llevar la cuenta de lo que podriamos llamar el "tiempo global". El valor de este "tiempo global" es el usado como variable independiente de la función reloj-maestro.

Para este caso, el procedimiento principal tiene la siguiente forma:

```
void Principal()
{
while (relojMaestro.tic())
  {
  ObtenerImagenes
  }
}
```

Como se puede observar, la actividad que controla todo el programa es el "tic" del reloj, ya que el mismo es el que actualiza el valor del tiempo global que es luego usado por todas las funciones.

A.4. Formato de las TNLEJ

Una vez se han definido los formatos de las funciones, se pueden exponer los de las TNLEJ, ya que las mismas se basan en las funciones.

El formato es:

clase_ transformación, actualización, transformación, parámetros.

con:

- *clase_ transformación*: CONSTANTE | VARIABLE
  Si el valor es CONSTANTE estaremos hablando de aplicar una transformación constante, aquella cuyo valor de la transformación no varía. Aunque la transformación sea clasificada como CONSTANTE, ésto quiere decir que la trasformación es la misma para toda la geometría a la que se aplique, pero no implica que tenga que ser constante en el tiempo. Se puede definir una transformación CONSTANTE en la cual el parámetro que define la transformación pueda variar en el tiempo dependiendo de una función. Si el valor es VARIABLE nos escontramos realmente con una TNLEJ.
- *actualización*: SI | NO
  Indica si los valores de la lista inicial deber ser actualizados con los de la lista final.
- *transformación*: TRASLACION | ESCALADO | ROTACION
- *parametros*:
  Dependen de la clase de transformación y de la trasformación. Veamos cuales son las distintas posibilidades:
  - TRASLACION:
    - CONSTANTE:
      δx | FUN_X, δy | FUN_X, δz | FUN_X;
      Los parámetros de la traslación son los desplazamientos que se desean aplicar en las tres coordenadas o un número de función local.
    - VARIABLE:
      Eje de selección, *Eje de aplicación*, *número de función local*;
      Eje de selección, *Eje de aplicación*: EJE_X | EJE_Y | EJE_Z
  - ESCALADO:
    - CONSTANTE:
      S_x | FUN_X, S_y | FUN_X, S_z | FUN_X;
      Los parámetros del escalado son los factores de escala, que pueden expresarse mediante valores fijos o mediante funciones locales.
    - VARIABLE:
      Eje de selección, *Eje de aplicación*, *número de función local*;
      Eje de selección, *Eje de aplicación*: EJE_X | EJE_Y | EJE_Z

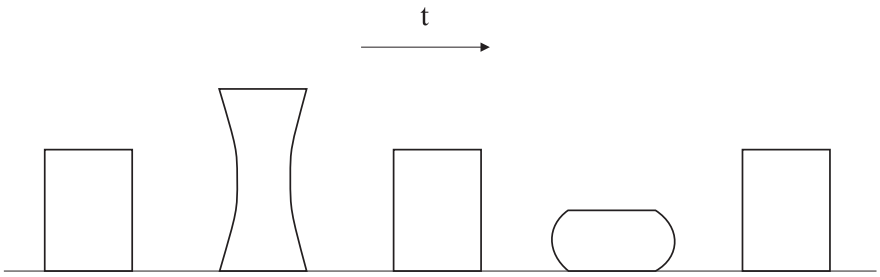


Figura A.1: Ejemplo de estiramiento y aplastamiento.

- ROTACION:
  - CONSTANTE:
    Eje de aplicación, *ángulo* | FUN_X; Se indica el ángulo de rotación (en grados) mediante un valor fijo o mediante una función local.
  - VARIABLE:
    Eje de selección, *Eje de aplicación*, *número de función local*;
    Eje de selección, *Eje de aplicación*: EJE_X | EJE_Y | EJE_Z

A.4.1. Ejemplo de Transformación No-lineal Extendida Jerárquica

Según se expuso anteriormente, la composición de transformaciones se realiza de una manera lógica, ya que en realidad al ser las transformaciones variables, no existe una forma trivial de obtener una sola transformación que combine el efecto de varias. Por tanto, la composición se produce por combinación acumulativa de resultados parciales. Veamos un ejemplo. Se van a definir las deformaciones que permiten implementar el principio clásico de la animación del aplastamiento y estiramiento (*Squash and Stretch*) aplicados a un cuadrado (Figura A.1). Para ello, primero se define una función que haga que el cubo se alarge y se encoja de forma lineal, por ejemplo, sobre su eje y. La variación en la extensión y contracción la vamos a hacer lineal, por simplicidad. Dado que el el cubo tiene una altura 100 y deseamos que el estiramiento lo alarge hasta 150 y la contracción llegue a 50, planteamos las trasformaciones en los puntos inicial, final e intermedios (Figura A.2). Hemos decidido implementar la deformación como una traslación. Para el caso del estiramiento tendremos que sumar 50, mientras que para la contración deberemo restar 50. Por último, queda definir cómo se va a pasar de una posición temporal a otra. Se prodría escoger un esquema lineal (Figura A.3, A), pero en nuestro caso hemos elegido una función senoidal, cuya amplitud es 50 (Figura A.3, b). El otro punto permanece constante. Veamos el código que define esta transformación (el reloj maestro está definido, y es la función global 0).

Definición del seno:

```
1:FUNCION_D:SENO,FUN_0,50,1,0,0;
```

Ésta podría ser una función global o local. Ahora definimos la función que

va a controlar la deformación. Lo vamos a hacer con un BSPLINE. La función quedaría definida como (numeración local):

```
0:FUNCION:BSPLINE,MIN_Y,MAX_Y,0,FUN_1,2,2,(0,0,0),(1,1,0);
```

y por ultimo la TNL:

```
VARIABLE,NO,TRASLACION,EJE_Y,EJE_Y,0;
```

Si ahora se desea añadir la parte que simula el mantenimiento del volumen, hay que introducir una transformación que haga que cuando el objeto se estira se entreche por la zona central (ésta es una posibilidad), y que cuando se aplaste se hinche también por la zona central. La forma de las curvas dada la figura original sería la siguiente (Figura A.4). Observar lo intuitivo que resulta su definición. La forma de trasformar el objeto va a ser mediante un escalado: cuando se quiera que se hinche se escalará por un factor mayor que 1 y cuando se desee estrechar se multiplicará por un factor menor que 1. Para esta transformación se va a hacer uso del mecanismo de variación de los límites en las ordenadas, mientras que se mantiene la forma de curva (Figura A.5).

La función global que va a definir:

```
2:FUNCION_D:SENO,FUN_0,-.4,10,0,1;
```

Como se observa hay un deplazamiento que hace que en realidad los valores mínimos y máximos que se pueden obtener sean $1 + ,4 = 1{,}4$ y $1 - ,4 = ,6$. La función local que define la deformación es la siguiente:

```
1:FUNCION:BSPLINE,MIN\_Y,MAX\_Y,1,FUN\_2,3,3,(0,0,0),
                                    (0.5,1,0),(1,0,0);
```

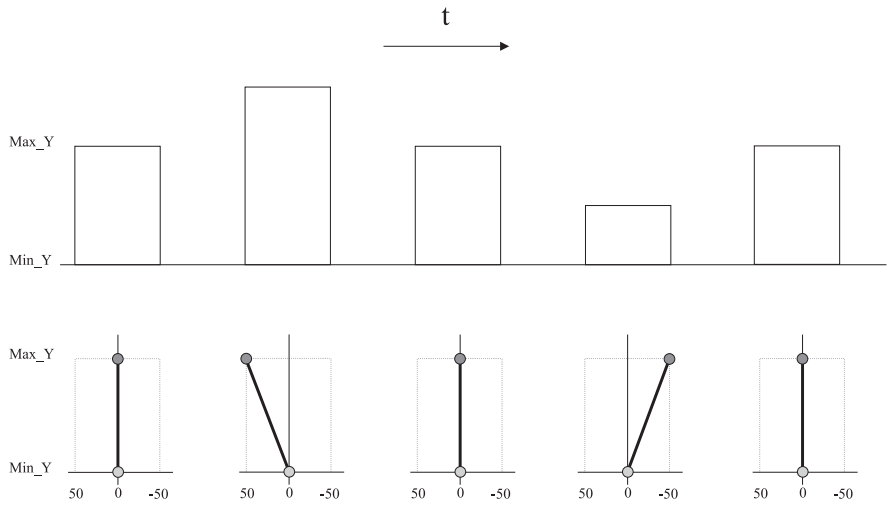


Figura A.2: Estiramiento y aplastamiento mediante una traslación.

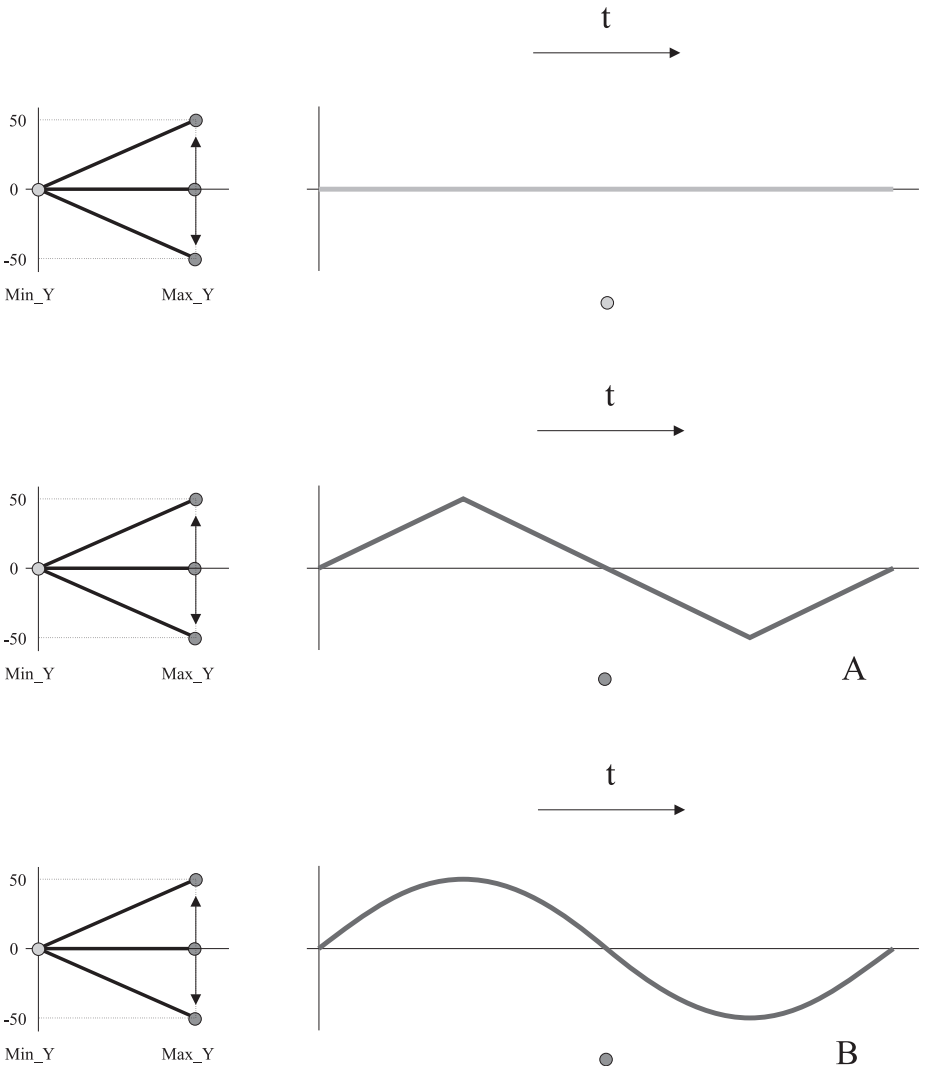


Figura A.3: Dependencia temporal de la función de control de la traslación.

Obsérvese como la ordenada se mantiene constante a 1 mientras que la ordenada máxima varía según la función definida anteriormente. La forma de la curva es fija. Por tanto se cambia el rango (Figura A.5). Ésto produce el resultado deseado. El método permite varias soluciones para un mismo objetivo. Por último, se define la trasformación no-lineal. Como se desea que la trasformación afecte a todo el perímetro del objeto, hay que aplicar la deformación dos veces, con el mismo eje de selección pero distinto eje de aplicación (aquí se ven las ventajas de crear el eje de aplicación, ya que se podría hacer la trasformación en un sólo eje). La transformación queda:

```
VARIABLE,NO,TRASLACION,EJE_Y,EJE_Y,0;
VARIABLE,NO,ESCALADO,EJE_Y,EJE_Z,1;
VARIABLE,NO,ESCALADO,EJE_Y,EJE_X,1;
```

Por último, añadamos a este movimiento el que el cubo oscile de un lado a otro. La definición sería la siguiente:

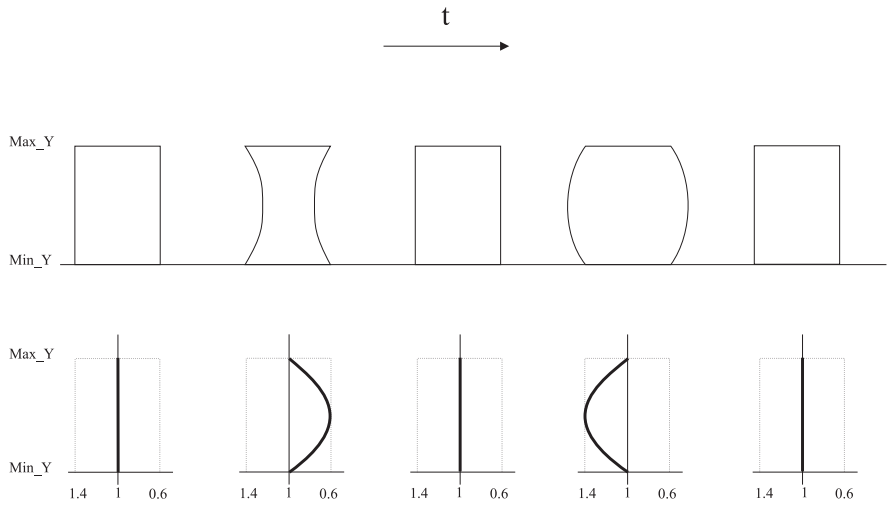


Figura A.4: Definición la función de control en el escalado.

Funciones globales:

```
1:FUNCION_D:SENO,FUN_0,50,1,0,0;
2:FUNCION_D:SENO,FUN_0,-.4,10,0,1;
3:FUNCION_D:SENO,FUN_0,30,5,0,0;
```

Funciones locales:

```
0: FUNCION: BSPLINE, MIN_Y, MAX_Y, 0, FUN_1, 2, 2,
                                    (0,0,0), (1,1,0);
1: FUNCION: BSPLINE, MIN_Y, MAX_Y, 1, FUN_2, 3, 3,
                        (0,0,0), (0.5,1,0), (1,0,0);
2: FUNCION: BSPLINE, MIN_Y, MAX_Y, 0, FUN_3, 2, 2,
                                    (0,0,0), (1,1,0);
```

TNLEJ:

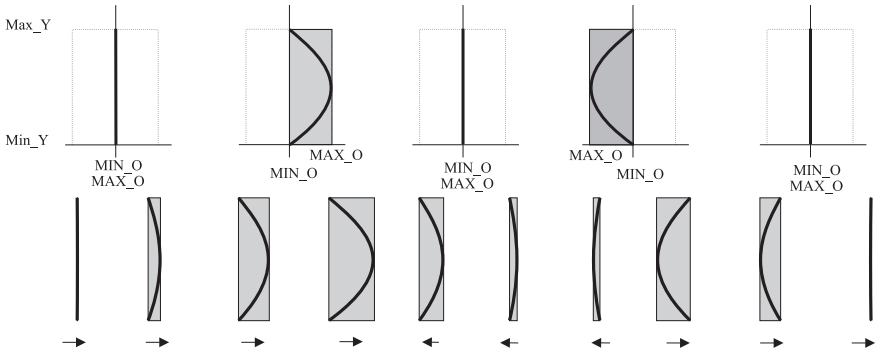


Figura A.5: Variación de las ordenadas.

```
VARIABLE,NO,TRASLACION,EJE_Y,EJE_Y,0;
VARIABLE,NO,ESCALADO,EJE_Y,EJE_Z,1;
VARIABLE,NO,ESCALADO,EJE_Y,EJE_X,1;
VARIABLE,NO,ROTACION,EJE_Y,EJE_X,2;
```

A.5. Formato de la cámara

Además de los objetos y las funciones que los controlan, es necesario definir una cámara que los observe. En esta sección se muestra el formato y las capacidades de la camara.

El formato genérico es el siguiente:

Imágenes a obtener
Parámetros de posicionamiento del observador (nomenclatura PHIGS)
Parámetros de proyección y visualización
Atributos de visión
Atributos de los ejes
Funciones de control de movimiento de la cámara
Definición de luces reales y sus atributos

El formato concreto es el siguiente:

- *Imágenes a obtener*
  - NUM_PASOS:
    Indica el número de imágenes que se quieren obtener.
  - INICIO:
    Indica en que posición se desea empezar.
  - FINAL:
    Indica en que posición se desea termina
  - REPETICION: SI | NO
    Indica se desea repetir o no el ciclo.
- *Parámetros de posicionamiento del observador*
  - VRP: x, y, z
  - VPN: x, y, z
  - VUP: x, y, z
  - PRP: x, y, z
- *Parámetros de proyección y visualización*
  - VENTANA_MUNDO: u_{min}, v_{min}, u_{max}, v_{max}
  - TIPO_PROYECCION: PARALELA | PERSPECTIVA

  - PLANOS_DE_CORTE: plano delantero, plano trasero
  - PUERTO_VISION: x_{min}, y_{min}, x_{max}, y_{max}
    Sistema coordenado izquierdo.
  - POS_VENTANA: x_{orig}, y_{orig}
    Posición de la ventana en Xwindows.
  - TAM_VENTANA:x_{tam}, y_{tam}
    Tamño de la ventana en Xwindows.

- *Atributos de visión*

  - DOBLE_BUFER: SI | NO
  - COLOR_FONDO: x, y, z siendo $x, y, z \in R$ y $0 \leq x, y, z \leq 1$
    Indica el color del fondo. Se define el color mediante el modelo RGB.
  - MODELO_ILUMINACION: ambiental, difuso, especular, exponente especular
    Define los coeficientes correspondientes del modelo de iluminación. Esto coeficientes su usa para la visualización sólida mediante OpenGL.
  - SUAVIZADO: SUAVIZADO | PLANO
    Indica si se va a aplicar suavizado en la visualización sólida mediante OpenGL.
  - PARAR: SI | NO
    Indica si en cada paso se va a parar para intercatuar con el usuario o no.
  - LUCES: SI | NO
    Indica se se va a visualizar una marca con mostrando la posición de las luces reales.
  - BITONO: SI |NO
    Indica si se va a aplicar o no el modo bitono.

- *Atributos de los ejes*

  - EJES: SI | NO
    Indica se se visualizan o no los ejes de coordenadas.
  - TAM_EJES:
    Indica el tamaño de los ejes.

- *Funciones de control de movimiento de la cámara*
  Para cada coordenada se define una función de control que indique como va a cambiar. La combinación de las trea coordenas da la trayectoria de la cámara. Las funciones que se usan son las mismas que ya se han comentado.

- *Definición de luces reales y sus atributos*

  - NUM_LUCES:
    Indica el número de luces reales que se van a definir. Se indican secuencialmente, dando sus parámetros de control, los cuales se incican a continuación.

- POSICION_LUZ: x, y, z
- COLOR_LUZ: x, y, z siendo $x, y, z \in R$ y $0 \leq x, y, z \leq 1$
  Indica el color de la luz. Se define el color mediante el modelo RGB.
- TIPO_LUZ: POSICIONAL | DIRECCIONAL
- LUZ_MOVIL: SI | NO
  Indica si la luz se moverá o no con la cámara.

A.6. Ejemplo de código

A continuación se muestra el código que permite crear la animación mostrada en la Figura 3.39. Hay que definir un fichero con los objetos de las escena. Estos fichero se nombran teerminando en la extensión *.obj*. Además se define un fichero pero con extesión *.vis* que indica el modo en el cual va a ser visualizado el modelo (ALAMBRE, AJEDREZ, SOLIDO, TRAZO (en alguno de las tres modos comentados). El fichero de las funciones globales termina en la extensión *.fun* y la cámara en *.cam*.

Del fichero de objetos sólo se muestra una parte (la montaña y el cuerpo del cerdo).

Fichero de objetos.

```
[
#comentario
TIPO: SBSPLINE
NOMBRE: montania1
NUM_PUNTOS_U:5
NUM_PUNTOS_V:5
NUM_DIVISIONES_U:30
NUM_DIVISIONES_V:30
ORDEN_U:3
ORDEN_V:3
TIPO_CURVA_EJE_U:NO_UNIFORME
TIPO_CURVA_EJE_V:NO_UNIFORME
PUNTOS_DE_CONTROL:
(-1000,0,-1000), (-500,0,-1000), (0,0,-1000), (500,0,-1000), (1000,0,-1000),
(-1500,0,-500), (-500 1000,-500), (0,0,500), (500,800,-500), (1000,0,-500),
(-1500,0,-250), (-500,50, 250), (0,0,0),(500,100,-250), (1000,0,-250),
(-1500,0,500), (-500,0,500), (0,0,500), (500,0,500), (1000,0,500),
(-1000,-200,1000), (-500,-200,1000), (0,-200,1000), (500,-200,1000),
(1000,-200,1000)
#comentario
*  ATRIBUTOS
ILUMINACION:NO
MODO_SUAVIZADO:SUAVIZADO
COLOR_RELLENO:(0.2,.6,0.2)
MATERIAL:(.2,.8,.2),(.2,.8,.2),(1,1,1),10
OPACIDAD:0.2
GROSOR_LINEA:2,5
* TGNL
# tipo, eje de eleccion, eje aplicacion,numero de funcion
NUM_TGNL:3;
CONSTANTE,NO,TRASLACION,0,-80,-500;
CONSTANTE,NO,ROTACION,EJE_Y,45;
```

```
CONSTANTE,NO,ESCALADO,3,3,3;
]
[
TIPO:SREVOLUCION
NOMBRE:cuerpo1
# si numpuntosU=0 no se ajusta bspline
NUM_PUNTOS_U:1
NUM_PUNTOS_V:8
NUM_DIVISIONES_U:50
NUM_DIVISIONES_V:50
ORDEN_U:3
ORDEN_V:3
TIPO_CURVA_EJE_U:UNIFORME
TIPO_CURVA_EJE_V:NO_UNIFORME
EJE_REVOLUCION:EJE_X
PUNTOS_DE_CONTROL:(-800,0,0),(-800,0,800),(800,0,800),(800,0,230),(830,0,200),
(1000,0,200),(1000,0,200),(1000,0,0)
* ATRIBUTOS
ILUMINACION:NO
MODO_SUAVIZADO:SUAVIZADO
COLOR_RELLENO:(1,.8,.8)
MATERIAL:(1,.8,.8),(1,.8,.8),(0,0,0),10
OPACIDAD:0.2
GROSOR_LINEA:1,3
* LUCES
NUM_LUCES_VIRTUALES:2
# tipo (DIF/ESP),pos (x,y,z),clase (DIR/POS),intensidad,movil
DIFUSA,1500,0,0,DIRECCIONAL,1,NO
ESPECULAR,1500,0,0,DIRECCIONAL,1,NO
* REFLEXIONES
# lim. inferior,lim. superior, operacion, condicion, exponente
CARA1:-1,1,>=LI_<=LS,1
CARA2:-1,1,>=LI_<=LS,1
CONJUNTA:0.5,1,>=LI_<=LS,CARA1_MAYOR_CARA2,ABS(CARA1-CARA2),0.5,1,>=LI_<=LS
CARA1:0.7,1,>=LI_<=LS,1
CARA2:0.7,1,>=LI_<=LS,1
CONJUNTA:0,0.95,>=LI_<=LS,CARA1_MENOR_CARA2,ABS(CARA1-CARA2),0.2,0.5,>=LI_<=LS
* OPERACION_LUCES
REFLEXION_SI O REFLEXION_SI
* FUNCIONES
# numero,tipo,absMin,absMax,ordMin (fun para funciones),ordMax,parametros
NUM_FUNCIONES:2
FUNCION_D:BSPLINE,FUN_0,0,720,2,2,(0,0,0),(1,1,0);
FUNCION_D:SENO,FUN_0,50,20,0,50;
* TGNL
#tipo con o var, tipo, eje de eleccion, eje aplicacion,numero de funcion PROPIA
NUM_TGNL:4;
CONSTANTE,NO,ESCALADO,0.3,0.3,0.3;
CONSTANTE,NO,TRASLACION,0,275,900;
CONSTANTE,NO,ROTACION,EJE_Y,FUN_0;
CONSTANTE,NO,TRASLACION,0,FUN_1,0;
]
```

Fichero de funciones globales.

```
#numero de funcion,tipo, parametros
# funciones basicas constantes,
# numero,tipo,ordenadas,parametros ; se numeran apartir de 1 (0=reloj)
NUM_FUNCIONES:10;
FUNCION_D:BSPLINE,FUN_0,0,0,2,2,(0,0,0),(1,1,0);
```

```
FUNCION_D:BSPLINE,FUN_0,1,1,2,2,(0,0,0),(1,1,0);
FUNCION_D:BSPLINE,FUN_0,0,45,2,2,(0,0,0),(1,1,0);
FUNCION_D:SENO,FUN_0,30,5,0,0;
FUNCION_D:BSPLINE,FUN_0,0,50,3,3,(0,0,0),(0.5,1,0),(1,0,0);
FUNCION_D:SENO,FUN_0,200,10,0,0;
FUNCION_D:SENO,FUN_0,-.4,10,0,1;
FUNCION_D:SENO,FUN_0,200,10,0,800;
# para inflar el ojo
FUNCION_D:BSPLINE,FUN_0,0,200,2,3,(0,0,0),(0.5,1,0),(1,0,0);
# para inflar la pupila
FUNCION_D:BSPLINE,FUN_0,0,100,2,3,(0,0,0),(0.5,1,0),(1,0,0);
```

Fichero de la camara.

```
# reloj maestro,numero pasos,inicio,final,repeticion
NUM_PASOS:50
INICIO:0
FINAL:1
REPETICION:NO
# parametros de la camara
VRP:0,0,2000
VPN:0,0,1
VUP:0,1,0
PRP:0,0,1000
# umin,vmin,umax,vmax
VENTANA_MUNDO:-500,-300,500,700
TIPO_PROYECCION:PERSPECTIVA
# delantero, trasero
PLANOS_DE_CORTE:0,-8000
# xmin,ymin,xmax,ymax
PUERTO_VISION:0,0,800,800
POS_VENTANA:50,50
TAM_VENTANA:800,800
DOBLE_BUFER:SI
COLOR_FONDO:.8,.9,1
MODELO_ILUMINACION:.5,.5,.5,1
SUAVIZADO:SUAVIZADO
PARAR:NO
LUCES:NO
BITONO:NO
EJES:NO
TAM_EJES:2000
# se meten funciones-> ordmin,ordmax,orden,numpuntos,puntos
FUNCION_D:BSPLINE,FUN_0,2300,1300,2,2,(0,0,0),(1,1,0);
FUNCION_D:BSPLINE,FUN_0,700,300,2,2,(0,0,0),(1,1,0);
FUNCION_D:BSPLINE,FUN_0,2300,1300,2,2,(0,0,0),(1,1,0);
#FUNCION_D:COSENO,FUN_0,1500,1,0,0;
#FUNCION_D:BSPLINE,FUN_0,300,300,2,3,(0,0,0),(.5,1,0),(1,0,0);
#FUNCION_D:SENO,FUN_0,1500,1,0,0;
# luces reales
NUM_LUCES:1
POSICION_LUZ:1500,500,0
COLOR_LUZ:.5,0.5,0.5
TIPO_LUZ:DIRECCIONAL
LUZ_MOVIL:SI
```

Apéndice B

Términos

2-variedad *2-manifold*
Acción directa ..... *Straigh Ahead Action*
Acción postura a postura ..... *Pose-to-Pose Action*
Acción secundaria ..... *Secondary action*
Afilar ..... *Tapering*
Ajustado ..... *Clean Up*
Alias ..... *Aliasing*
Animación basada en acetatos ..... *Cel animation*
Anticipación ..... *Anticipation*
Árbol octal ..... *Octree*
Arcos ..... *Arcs*
Aristas Aladas, estructura de datos ..... *Winged-edged data estructure*
Atractivo ..... *Appeal*
Báfer ..... *Buffer*
Banda sonora ..... *Sound Track*
Bandera ..... *Flag*
Bobina Leica ..... *Leica Reel*
Caras que miran hacia atrás ..... *Back face culling*
Completar y solapar ..... *Follow Through and Overlapping Action*
Comprobación ..... *Checking*
Conjunto ..... *Cluster*
Copia maestra ..... *Answer Print*
Deformación Libre de Forma ..... *Free Form Deformation*
Descomposición de la banda sonora ..... *Track Breakdown*
Diseño ..... *Design*
Doblaje ..... *Dubbing*
Doblar ..... *Bending*
Escenificación ..... *Staging*
Esquema de la historia ..... *Storyboard*
Exageración ..... *Exaggeration*
Fichero de ordenes, guión ..... *Script*
Filmado ..... *Final Shoot*
Fondos ..... *Backgrounds*
Funcionteca ..... *Library*

Guión ..... *Script*
Hoja del modelo ..... *Model sheet*
Hoja tabulada ..... *Bar sheet*
Imagen clave ..... *Keyframe*
Intercalado ..... *Inbetween*
Lento al llegar lento al salir ..... *Slow In Slow Out*
Mapa planar ..... *Planar map*
Memoria de estarcido ..... *Stencil buffer*
Memoria de imagen, bafer de imagen ..... *Frame buffer*
Memoria de material ..... *Material-buffer*
Mezclado ..... *Blended*
Nivel de Detalle ..... *LoD, Level of Detail*
Objetos blandos ..... *Soft objects*
Primeras pruebas ..... *Rushes*
Proceso de visualizacion ..... *Pipeline*
Prueba de línea ..... *Line Tests*
Puntos críticos ..... *Critical points*
Suavizado ..... *Shading*
Temporización ..... *Timing*
Torcer ..... *Twisting*
Transformación No-Lineal ..... *Non Lineal Transformation*
Trazado y coloreado ..... *Trace and Paint*
Trazos esqueléticos ..... *Skeletal Strokes*
Visualización ..... *Rendering*

Bibliografía

[1] A. Barr. Global and local deformations of solid primitives. *Proceedings of SIGGRAPH*, 18(3):21–30, 1984.

[2] A. Barr, B. Currin, S. Gabriel, and J. Hughes. Smooth interpolation of orientations with angular velocity constraints. *ACM Computer Graphics*, 26(2):313–320, July 1992.

[3] A. M. Barr. Supercuadrics and angle preserving transformations. *IEEE Computer Graphics And Applications*, 1(1):11–23, 1981.

[4] T. Beier and S. Neely. Feature-based image metamorphosis. *ACM Computer Graphics*, 26(2):35–42, July 1992.

[5] J. F. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on graphics*, 1(3):235–256, 1982.

[6] L. Brotman and A. Netravali. Motion interpolation by optimal control. *ACM Computer Graphics*, 22(4):309–316, August 1988.

[7] N. Burntnyk and M. Wein. Interactive squeleton techniques for enhancing motion dynamics in keyframe animation. *Communications of the ACM*, 19(10):564–569, October 1976.

[8] M. Carignan, Y. Yang, N. Magnenat-Thalmann, and D. Thalmann. Dressing animated synthetic actors with complex deformable clothes. *ACM Computer Graphics*, 26(2):99–104, July 1992.

[9] E. Catmull. The problems of computer-assisted animation. *ACM Computer Graphics*, pages 348–353, 1978.

[10] D. Chen and D. Zeltzer. Pump it up: Computer animation of a biomechanically based model of muscle using the finite element method. *ACM Computer Graphics*, 26(2):89–98, July 1992.

[11] P. H. Christensen. Contour rendering. *Computer Graphics*, 33(1):58–60, February 1999.

[12] S. Coquillart. Extended free form deformation: A sculpturing tool for 3D geometric modeling. *ACM Computer Graphics*, 24(4):187–196, August 1990.

[13] S. Coquillart and P. Jancene. Animated free form deformation: An interactive animation technique. *ACM Computer Graphics*, 25(4):23–26, July 1991.

[14] A. Daldegan, N. Magnenat-Thalmann, T. Kurihara, and D. Thalmann. An integrated system for modeling, animating, and rendering hair. *Computer Graphics Forum*, 12(3):211–221, September 1993.

[15] S. Donikiam and G. Hegron. A declarative design method for 3D scene sketch modeling. *Computer Graphics Forum*, 12(3):223–236, September 1993.

[16] D. Ebert and R. Parent. Rendering and animation of gaseous phenomena by commbining fast volume and scan-line a-buffer techniques. *ACM Computer Graphics*, 24(4):357–366, August 1990.

[17] G. Elber and E. Cohen. Hidden curve removal from free form surfaces. *ACM Computer Graphics*, 24(4):95–104, August 1990.

[18] G. Farin. *Curves And Surfaces For CAGD: A Practical Guide. 3 Edition.* Academic Press, 1993.

[19] J. D. Fekete et al. Tictactoon: A paperless system for professional 2D animation. *Proceedings of SIGGRAPH*, pages 79–90, August 1995.

[20] A. Finkelstein and D. H. Salesin. Multiresolution curves. *Proceedings of SIGGRAPH*, pages 261–268, July 1994.

[21] J. Foley, A. van Dam, S. Feiner, and J. F. Hughes. *Computer Graphics: Principles And Practice, 2 Edition.* Addison-Wesley, 1992.

[22] D. R. Forsey and R. H. Bartels. Hierarchical b-spline refinement. *ACM Computer Graphics*, 22(4):205–212, August 1988.

[23] A. Gooch, B. Gooch, P. Shirley, and E. Cohen. A non-photorealistic lighting model for automatic technical illustration. *Proceedings of SIGGRAPH*, pages 447–452, July 1998.

[24] J. Gouret, N. Magnenat-Thalmann, and D. Thalmann. A non-photorealistic lighting model for automatic technical illustration. *Proceedings of SIGGRAPH*, pages 447–452, July 1998.

[25] S. Green. Beyond photorealism. *Workshop on Rendering*, July 1999.

[26] S. Greenberg. Why non-photorealistic rendering ? *Computer Graphics Forum*, 33(1):56,57, February 1999.

[27] P. Haeberli. Paint by numbers: Abstract image representation. *ACM Computer Graphics*, 24(4):207–214, August 1990.

[28] P. Hall. Non-photorealistic rendering by q-mapping. *Computer Graphics Forum*, 18(1):41–56, March 1999.

[29] H. Hoppe. View-dependent refinement of progressive meshes. *Proceedings of SIGGRAPH*, pages 189–198, August 1997.

[30] S. C. Hsu and I. H. Lee. Drawing and animation using skeletal strokes. *Proceedings of SIGGRAPH*, pages 109–118, July 1994.

[31] W. Hsu, J. Hughes, and H. Kaufman. Direct manipulation of free form deformations. *ACM Computer Graphics*, 26(2):177–184, July 1992.

[32] P. Isaacs and M. Cohen. Controling dynamics simulation with kinematics constraints, behavior functions and inverse dynamics. *ACM Computer Graphics*, 21(4):215–224, July 1987.

[33] P. Kalra, A. Mangili, N. Magnenat-Thalmann, and D. Thalmann. Simulation of facial muscle actions based on rational free form deformations. *Computer Graphics Forum*, 11(3):59–69, September 1992.

[34] S. Karan and E. Fiume. Wires: A geometric deformation technique. *Proceedings of SIGGRAPH*, pages 405–414, 1998.

[35] W. Kong and M. Nakajima. Visible volume buffer for efficient hair expression and shadow generation. *Computer Animation 99*, pages 58–65, 1999.

[36] D. Kromker and G. R. Hofmann. Interaction, integration, visualisation. Technical report, Eurographics, 1989.

[37] J. Lansdown and S. Schofield. Expressive rendering: A review of non-photorealistic techniques. *IEEE Computer Graphics and Applications*, pages 29–37, May 1995.

[38] J. Lasseter. Principles of traditional animation applied to 3D computer animation. *Proceedings of SIGGRAPH*, 21(4):35–44, July 1987.

[39] M. J. Lazlo. *Computational Geometry And Computer Graphics In C++*. Prentice-Hall, 1996.

[40] W. Leister. Computer generated copper plates. *Computer Graphics Forum*, 13(1):69–77, 1994.

[41] P. C. Litwinowicz. Inkwell: A $2\frac{1}{2}$d animation system. *Proceedings of SIGGRAPH*, 25(4):113–122, July 1991.

[42] D. Luebke and C. Erikson. View-dependent simplification of arbitray polygonal enviroments. *Proceedings of SIGGRAPH*, pages 199–208, August 1997.

[43] R. MacCracken and K. I. Joy. Free-form deformations with lattices of arbitrary topology. *Proceedings of SIGGRAPH*, pages 181–188, August 1996.

[44] N. Magnenat-Thalmann and D. Thalmann. *Computer Animation: Theory And Practice. Second Edition.* Springer Verlag, 1990.

[45] L. Markosian et al. Real-time non-photorealistic rendering. *Proceedings of SIGGRAPH*, pages 415–419, 1997.

[46] D. Martín. Revisión de métodos para obtener siluetas. *Jornadas de Informática Gráfica*, pages 61–70, September 1998.

[47] D. Martín and J. C. Torres. Obtención y visualización rápida de trazos. *8 Congreso Español de Informática Gráfica CEIG98*, pages 95–108, 1998.

[48] D. Martín and J. C. Torres. *Alhambra*: A system for producing 2D animation. *Computer Animation 99*, pages 38–47, 1999.

[49] D. Martín and J. C. Torres. *Virtual Lights*: A method for expressive visualisation. *EG99-SP*, pages 67–70, September 1999.

[50] D. Martín, J. C. Torres, and V. del Sol. A model for production of two dimensional animation. *Fourth Eurographics Animation and Simulation Workshop*, pages 11–22, September 1993.

[51] D. Metaxas and D. Terzopoulos. Dynamic deformation of solid primitives with constraints. *ACM Computer Graphics*, 26(2):309–312, July 1992.

[52] M. E. Mortenson. *Geometric Modeling*. John Wiley & Sons, 1985.

[53] S. Muraki. Volumetric shape description of range data using "blobby model". *ACM Computer Graphics*, 25(4):227–235, July 1991.

[54] H. Nishimura et al. Object modelling by distribution function and a method of image generation. *The Transactions of the Institute of Electronics and Communication Engineers of Japan*, J68-D(4):718–725, July 1985.

[55] A. Pentland and J. Williams. Good vibrations: Modal dynamics for graphics and animation. *ACM Computer Graphics*, 23(3):215–222, July 1989.

[56] P. Prusinkiewicz, M. S. Hammel, and E. Mjolsness. Animation of plant development. *Proceedings of SIGGRAPH*, 27(4):351–360, July 1993.

[57] R. Richens and S. Schofied. Interactive computer rendering. *Architectural Research Quarterly*, 1(1), 1995.

[58] P. Rustagi. Silhouette line display from shaded models. *Iris Universe*, pages 42–44, Fall 1989.

[59] T. Saito and T. Takahashi. Comprehensible rendering of 3D shapes. *ACM Computer Graphics*, 24(4):197–206, August 1990.

[60] M. P. Salisbury, C. Anderson, D. Lischinski, and D. H. Salesin. Scale-dependent reproduction of pen-and-ink illustration. *Proceedings of SIGGRAPH*, pages 461–468, August 1996.

[61] M. P. Salisbury, S. E. Anderson, R. Barze, and D. H. Salesin. Interactive pen and ink illustration. *Proceedings of SIGGRAPH*, pages 101–108, July 1994.

[62] M. P. Salisbury, M. T. Wong, J. F. Hughes, and D. H. Salesin. Orientable textures for image-based pen and ink illustration. *Proceedings of SIGGRAPH*, pages 401–406, August 1997.

[63] J. Santoja and S. Bayarri. Resolución del problema de extracción de tiras triangulares mediante algoritmos genéticos. *8 Congreso Español de Informática Gráfica CEIG98*, pages 149–162, 1998.

[64] T. Sederberg and E. Greenwood. A physically based approach to 2-d shape blending. *ACM Computer Graphics*, 26(2):25–34, July 1992.

[65] T. Sederberg and S. R. Parry. Free form deformation of solid geometric models. *Proceedings of SIGGRAPH 86*, 20(4):151–160, 1986.

[66] T. W. Sederberg and A. K. Zundel. Scan line display of algebraic surfaces. *ACM Computer Graphics*, 23(3):147–156, July 1989.

[67] D. D. Seligman and S. Feiner. Automated generation of intent-based 3D illustration. *ACM Computer Graphics*, 25(4):123–132, July 1991.

[68] M. Shinya and A. Fourier. Stochastic motion-motion under the influence of wind. *Computer Graphics Forum*, 11(3):119–128, September 1992.

[69] J. Stam and E. Fiume. Turbulent wind fields for gaseous phenomena. *Proceedings of SIGGRAPH*, 27(4):279–280, July 1993.

[70] D. Terzopoulos and K. Fleischer. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. *ACM Computer Graphics*, 22(4):269–278, August 1988.

[71] F. Thomas and O. Johnston. *The Illusion Of Life: Disney Animation*. Hiperion, 1995.

[72] K. Waters. A muscle model for animating three dimensional facial expressions. *ACM Computer Graphics*, 19(3):17–24, July 1985.

[73] J. Wejchert and D. Haumann. Animation aerodynamics. *ACM Computer Graphics*, 25(4):19–22, July 1991.

[74] T. White. *The Animator's Workbook. 3 Edition*. Phaidon-Press Limited, 1992.

[75] G. Winkenbach and D. H. Salesin. Computer generated pen and ink illustration. *Proceedings of SIGGRAPH*, pages 469–476, July 1994.

[76] G. Winkenbach and D. H. Salesin. Rendering parametric surfaces in pen and ink. *Proceedings of SIGGRAPH*, pages 469–476, August 1996.

[77] Y. Wu, P. Beylot, and N. Magnenat-Thalmann. Skin agin estimation by facial simulation. *Computer Animation 99*, pages 210–219, 1999.

[78] G. Wyvill, C. McPheeters, and B. Wywill. Data structure for soft objects. *Visual Computer*, 2(4):227–234, August 1986.

[79] B. Wywill and G. Wyvill. Field functions for implicit surfaces. *Visual Computer*, 5:75–82, 1989.

[80] R. C. Zeleznik, K. P. Herndon, and J. F. Hughes. Sketch: An interface for sketching 3D scenes. *Proceedings of SIGGRAPH*, pages 163–170, August 1996.