Aplicaciones didácticas en VPython y su empleo en entornos de aprendizaje cooperativo

Tutores: Pablo I. Hurtado Fernández
Francisco de los Santos Fernández

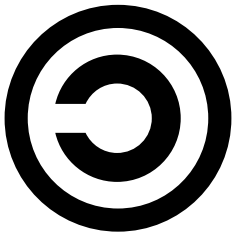
Dpto. de Electromagnetismo y Física de la Materia

Trabajo Fin de Máster

Septiembre 2015

ugr | Universidad de Granada

Pablo Villegas
pvillegas@ugr.es



copyleft

Pablo Villegas, 2015.

©2015 por Pablo Villegas Góngora. Esta obra está sujeta a la licencia Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional de Creative Commons. Para ver una copia de esta licencia, visite http://creativecommons.org/licenses/by-nc-sa/4.0/.

Prefacio

***

El objetivo de este Trabajo de Fin de Máster es el diseño y desarrollo de aplicaciones en Física en un entorno Python, enmarcadas en un ambiente de trabajo cooperativo en el aula, proporcionando propuestas de trabajo, indicaciones didácticas y material educativo para su uso.

Aunque hay diversas aproximaciones al problema que aquí abordaremos, plantean la dificultad de encontrar dichas perspectivas interdisciplinares, bien por la rigidez de la organización interna de los centros, la formación inicial del profesorado, la presión de evaluaciones externas o la falta de materiales interdisciplinares bien diseñados. Los temas propuestos permiten desarrollar no solo cuestiones meramente ceñidas al currículum, sino entender que una manera interdisciplinar de actuación es necesaria en el mundo complejo en el que vivimos. Además, es de esperar que la metodología de trabajo seleccionada permita superar una dinámica tradicional en el entorno de clase y que permita una mejora en la atención a la diversidad de nuestro alumnado.

Más allá de la relevancia curricular, otro objetivo será construir un proyecto que conecte nuestro tema de estudio de manera transversal con el resto de asignaturas. El desarrollo desde una perspectiva transversal e interdisciplinar debe permitir al alumnado desarrollar múltiples competencias, y a su vez potenciar el pensamiento crítico y la capacidad de reflexión.

Por otro lado, las estrategias explicadas en el trabajo facilitan la integración de las TIC (competencia digital) como herramienta de aprendizaje y potencian otras competencias como la lingüística, matemática, el tratamiento de la información, aprender a aprender, la autonomía personal y el trabajo cooperativo con los compañeros.

Así, podemos distinguir los siguientes objetivos como parte fundamental de este trabajo:

- Generar material de trabajo y aplicaciones computacionales en el ámbito de la Física.
- Introducir el aprendizaje cooperativo como herramienta de trabajo en el aula.
- Considerar cómo debe desarrollarse este tipo de aprendizaje de forma pragmática para que sea eficaz.
- Generar herramientas de trabajo en el aula para fomentar un alumnado crítico, reflexivo y capaz de participar activamente en la sociedad.

Índice general

***

Capítulo 1

Aprendizaje cooperativo

***

1.1 Introducción

"La educación auténtica, repetimos, no se hace de A para B o de A sobre B, sino A con B, con la mediación del mundo."

Paulo Freire

El objetivo principal de un docente es generar un proceso de aprendizaje significativo y duradero, acompañado de elementos vitales en el desarrollo personal de nuestro alumnado, entre los que debemos destacar la capacidad de desarrollo de pensamiento crítico y estrategias de colaboración y cooperación con el resto de la sociedad.

Asimismo, la idea de construir una sociedad democrática y solidaria debe basarse en una escuela que se sustente en dichos principios (Dewey [3], Chomsky [2]). Una educación transversal, multidisciplinar y democrática desarrolla intrínsecamente aquellos valores que debemos transmitir al alumnado, sin necesidad de ser materias a impartir específicamente. Además, una educación basada en la enseñanza directa y unidireccional profesor-alumno no fomenta esta idea de escuela democrática. La profunda individualidad y competitividad que, a veces, se fomenta en las aulas y entre el alumnado, tampoco promueve los principios de cooperación, solidaridad y apoyo que el alumnado debe adquirir como parte de la sociedad. De hecho, en *La Escuela Moderna,* Ferrer Guardia [4] ya discute la idoneidad de una enseñanza marcada por exámenes y premios individuales y la necesidad de una escuela racional, laica y científica para construir las bases de una sociedad fraternal, solidaria y justa.

Por ello, y como indica Freire [5], la relación con los alumnos y alumnas ha de ser la de un compañero o compañera más, con un objetivo de reflexión mutua y evitando que las clases sean una mera donación o entrega de conocimientos.

Una propuesta interesante, que puede permitir cambiar la estructura de una clase sin apenas interacción es el aprendizaje cooperativo, realizando y resolviendo tareas de manera colectiva. Por ello, en este capítulo nos centraremos en describir su puesta en funcionamiento a nivel básico en una clase, detallando cómo ha de desarrollarse paso por paso y qué estructuras de trabajo, tareas y evaluaciones permite llevar a cabo.

Su implantación a veces puede ser compleja, pues implica un cambio en la dinámica completa del profesorado de forma transversal. Tanto la crítica por parte de los compañeros, como la dificultad para adaptar el mobiliario de las aulas y el horario necesario, necesita de una implicación del centro educativo que puede ser difícil de conseguir. Pese a todo, pienso que es una estructura que mejora la enseñanza tradicional y junto a la inclusión de nuevas tecnologías o elementos innovadores en nuestras aulas, ayuda a amplificar estrategias de atención a la diversidad, para desarrollar múltiples competencias necesarias en nuestro alumnado y potenciar su desarrollo personal y social.

En cuanto a su fundamentación psicopedagógica, recogeremos los puntos más importantes que justifican este tipo de aprendizaje y se detallan en Torrego y Negro [21]. En primer lugar, la teoría sociohistórica, propuesta por Lev Vigotsky, concede un papel central a la educación, subordinando el desarrollo humano a procesos y contextos que se dan en la estructura de la sociedad, en lugar de a procesos naturales o biológicos. Por ello, es importante la interacción entre iguales en nuestras aulas, y partiendo de la base de que atender de forma individualizada a cada alumno o alumna es difícil, el aprendizaje entre iguales se convierte en una excelente propuesta. De hecho, el lenguaje y explicación puede ser a un nivel mucho más cercano que facilita el aprendizaje, actuando el profesorado como guía o ayuda para asegurar que, en efecto, la interacción comporte aprendizaje. De la misma manera, potencia la zona de desarrollo pró-

ximo del alumnado, pues la construcción de conocimientos compartidos se produce de manera permanente.

Por otro lado, no debemos obviar la teoría de la interdependencia social. La interdependencia se define como la dinámica de ser mutuamente responsable y de compartir un conjunto común de principios con otros. En esencia, y como veremos posteriormente, esta interdependencia puede ser positiva (si el éxito se produce conjuntamente), negativa (en un entorno de competición, pues el objetivo es el fracaso de los demás) o ausente (si no hay interacción con el resto de individuos). Las dinámicas y comportamientos que podemos potenciar en nuestras aulas, en función del tipo de interdependencia empleada, pueden llevar al desaliento y obstrucción del aprendizaje en el alumnado. De hecho, imaginemos un ejemplo claro en el que el profesor o profesora realiza una pregunta en clase, y un solo alumno o alumna levanta la mano y contesta correctamente a la pregunta. ¿Anima este tipo de estructura a la participación del resto del alumnado? ¿Qué sentimientos despierta en aquellos que sabían la respuesta pero no pueden contestar o participar? ¿Fomenta la pérdida de atención en el alumnado?

Como ya se ha anticipado hasta ahora, no creo que la respuesta provenga de una enseñanza tradicional donde el profesorado se ve desbordado en muchos casos por la ratio de alumnos y alumnas, la heterogeneidad de sus clases y la atención a la diversidad que a veces es imposible abordar. Esta propuesta, sin resolver completamente todos los problemas, y siendo necesario un esfuerzo importante para su implantación, sí que puede ayudar a mejorar los conocimientos, aptitudes y procedimientos de nuestro alumnado, especialmente de aquellos más necesitados de apoyo en nuestras aulas. No debemos olvidar que en una clase, siempre habrá una pequeña minoría que siga nuestras explicaciones, pero el objetivo es desarrollar nuevas estructuras que no conviertan la educación en un reducto para una pequeña élite que pueda superar las dificultades de la etapa escolar por sus condiciones familiares y sociales.

1.2 Aprendizaje cooperativo

"Nos educan para ser productores y consumidores, no para ser hombres libres"

José Luis Sampedro

Entre las diversas estrategias de aprendizaje nuestro principal objetivo es descartar aquellas que fomentan un ambiente competitivo e individualista en nuestras aulas. Así, emplearemos las estrategias de aprendizaje cooperativo propuestas por diversos autores (Pujolás Maset [13], Pujolás Maset y Doñate [15], Lab! [9], Gavilán Bouzas [7], Torrego y Negro [21]).

Como punto de partida debemos promover que nuestras aulas sean inclusivas, es decir, que independientemente de su origen cultural o cualquier otra diferencia, cualquier alumno o alumna forme parte de ella y pueda ser aceptado. El aprendizaje cooperativo es una manera de que un alumnado diverso pueda aprender en comunidad. Para ello debemos comprender que la cooperación es algo más que colaboración, pues fomenta el respeto entre compañeros y compañeras dentro del aula y permite desarrollar valores fundamentales como la solidaridad y el apoyo mutuo en nuestro alumnado.

De nuevo, siguiendo a Pujolás Maset [14], *"podemos definir el aprendizaje cooperativo como el uso didáctico de equipos reducidos de alumnos, generalmente de composición heterogénea en rendimiento y capacidad, aunque ocasionalmente pueden ser más homogéneos, utilizando una estructura de la actividad tal que asegure al máximo la* ***participación equitativa*** *(para que todos los miembros del equipo tengan las mismas oportunidades de participar) y se potencie al máximo la* ***interacción simultánea*** *entre ellos, con la finalidad de que todos los miembros de un equipo aprendan los contenidos escolares, cada uno hasta el máximo de sus posibilidades y aprendan, además, a trabajar en equipo".*

1.2.1 Sesión de clase

1.2.1.1 Planificación temporal

En primer lugar, abordaremos la temporalización de nuestras sesiones de clase tipo. Es inevitable introducir en ellas una parte de intervención del profesor, como guía de la actividad que se desarrollará. No obstante, es parte clave la participación activa del alumnado y la forma de intervención de éste.

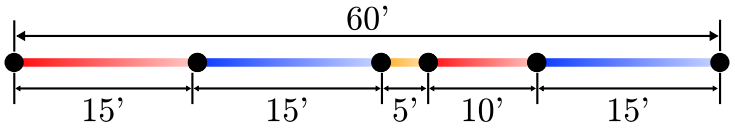


Figura 1.2.1: Estructura de la clase: partes protagonizadas por el profesor (en rojo), por el alumnado (en azul) y descansos (en naranja).

Como hemos indicado anteriormente, hay tres tipos básicos de dinámicas de intervención en el aula:

※ Estructura competitiva Trabajan de forma individual pero compiten entre ellos. No debemos olvidar que esta estructura se fundamenta en la exclusión de gran parte de nuestro alumnado: sólo uno puede alcanzar la máxima recompensa o calificación e independientemente del tipo de meta que establezcamos, el éxito siempre estará unido al fracaso de los demás. Tal y como indica Gavilán Bouzas [7], esta corriente tiene su origen en la década de 1930, debido a la Gran Depresión en Estados Unidos. Refleja el traslado al ámbito de la vida cotidiana de la esencia de una economía capitalista implantada en la sociedad.

Como hemos comentado previamente, sólo una pequeña parte de nuestro alumnado consigue triunfar, por lo que la sensación de fracaso, incompetencia o desazón, puede aparecer en la mayor parte del alumnado. Estaremos obviando toda socialización y dificultando si cabe competencias básicas como la social y ciudadana (LOE) o social y cívica (LOMCE).

※ Estructura individualista Todos trabajan en solitario sin competir con el resto de compañeros y compañeras. A diferencia de la estructura anterior, aquí el alumnado es totalmente independiente del resto, y su éxito no va ligado al fracaso de los demás. Fomenta la autonomía individual de nuestro alumnado, pero nuevamente, estamos obviando las relaciones interpersonales y no fomentaremos actitudes de solidaridad, cooperación y apoyo en nuestras aulas. Igualmente, dificulta en exceso la atención a la diversidad a nuestros alumnos y alumnas, especialmente en clases numerosas como las actuales.

※ Estructura cooperativa En este caso, alcanzar la meta propuesta depende de un trabajo conjunto y ésta debe ser alcanzada por todo el equipo de trabajo, fomentando el apoyo mutuo y la cooperación. Junto con la autonomía individual y el desarrollo de su propio aprendizaje, deben a su vez ayudar a sus compañeros y compañeras.

Como principal elemento a favor de este tipo de estructuras cabe destacar la función socializadora que puede establecerse en nuestras aulas y el desarrollo de la competencia social, proceso clave en el desarrollo personal de cada individuo. Además, no dificulta el desarrollo del temario a impartir y permite un importante impulso del aprendizaje entre iguales.

El efecto que producen este tipo de estructuras en nuestros estudiantes es considerable. En los dos primeros casos, tanto en el *"cómo"* y *"qué"* obtener, y en la recompensa obtenida, prima una postura individualista de cada alumno y alumna. Parece evidente que, más allá de los contenidos, si queremos desarrollar una estructura de cooperación y enseñar valores sociales y capacidad crítica a nuestro alumnado, las estructuras individualistas y competitivas no son las ideales a plantear. Por esto, dentro de la parte de participación activa de nuestros alumnos y alumnas optaremos por una estructura cooperativa.

1.2.1.2 Disposición del aula

Es indudable que la disposición del aula fomentará cada una de las actividades que pueden desarrollar nuestros alumnos y alumnas en las sesiones de clase. Los miembros de cada equipo de trabajo deben sentarse juntos, viéndose las caras y pudiendo interaccionar entre ellos, pero debemos tener cuidado y dejar suficiente separación entre equipos para que no interfieran en su trabajo. Asimismo, es vital que los equipos sean capaces de tener dinamismo para cambiar la disposición del aula sin excesivo ruido y evitando pérdidas de tiempo en el desarrollo de la clase.

De esta manera, una distribución como la de la figura 1.2.2, presente en multitud de aulas, fomenta el trabajo de manera individual y/o competitiva, pues dificulta el dialogo y la cooperación entre nuestro alumnado. Es vital adoptar distribuciones tales como las planteadas en la figura 1.2.3, que facilitan el diálogo entre los equipos y el desarrollo de las técnicas planteadas en este capítulo.

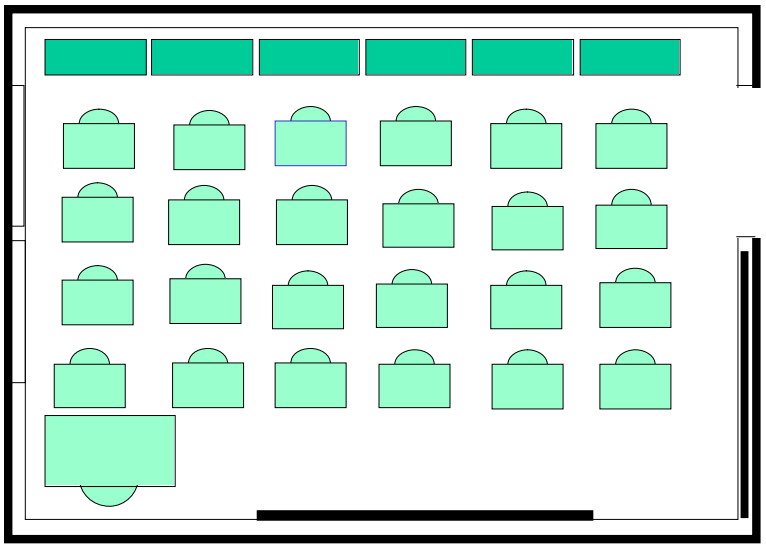


Figura 1.2.2: Distribución de clase para desarrollar una estructura individualista o competitiva. Fuente: Pujolás Maset y Doñate [15].

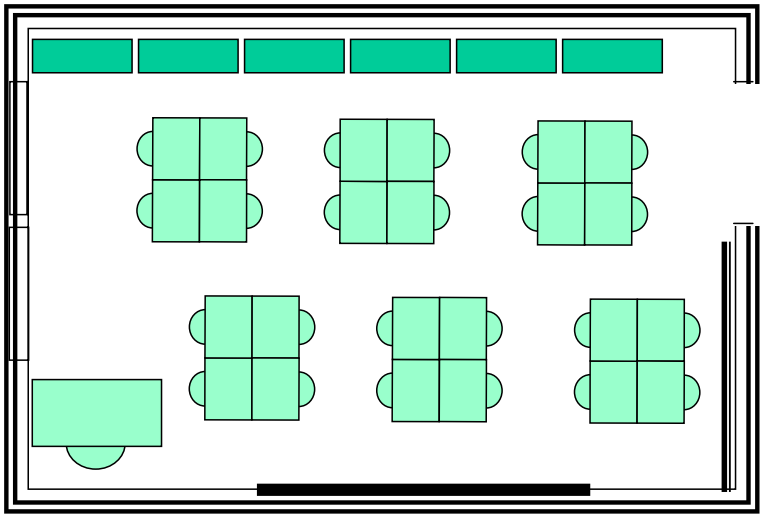


Figura 1.2.3: Distribución de clase para desarrollar una estructura cooperativa. Fuente: Pujolás Maset y Doñate [15].

1.2.2 Formación de equipos

En el aprendizaje cooperativo al hablar de grupos no nos referimos solo a un agrupamiento sin más del alumnado. Por ello, la creación de los equipos es un paso especialmente delicado y debe hacerse previo conocimiento del grupo en que nos encontramos, por lo que es preferible conocer al alumnado al principio de curso antes de desarrollar cualquier tipo de técnica. Para esto, pueden ser bastante útiles las pruebas iniciales o una primera toma de contacto donde podamos identificar las capacidades de cada uno de nuestros alumnos y alumnas y conocer el nivel desde el que parten. Aún así, los equipos de trabajo no deben ser inamovibles y pueden modificarse bajo ciertas circunstancias específicas.

En primer lugar debemos elegir el tamaño de nuestros equipos de trabajo. El tamaño ideal es de 4 personas, distribuidas tal y como se indica en la figura 1.2.4 en la página siguiente. Algunas de las razones para elegir este tamaño de equipos son las siguientes (Lab! [9]):

- Existe diversidad.
- La coordinación no es demasiado compleja, pues no hay demasiados estudiantes.
- Si faltan estudiantes, el equipo puede funcionar con relativa normalidad.
- Pueden distribuirse suficientes roles, como veremos posteriormente.
- Puede subdividirse en parejas para realizar actividades puntuales.

No obstante, puede quedar abierta la opción de realizar equipos impares (tres o cinco personas) para,

- Evitar la segregación en parejas de trabajo.
- Agilizar el acuerdo por mayorías y posibles votaciones internas.

De la misma manera, cabe la posibilidad de realizar grupos tanto homogéneos como heterogéneos. En general los grupos deben ser heterogéneos para que puedan realizar un proceso de aprendizaje entre iguales, así como desarrollar internamente las estrategias de atención a la diversidad esperadas. No obstante, dependiendo de la función del equipo de trabajo, podemos variar la composición de éstos.

※ Grupos homogéneos Esta forma de agrupamiento siempre será esporádica, pero puede servir para diversos objetivos, como potenciar habilidades sociales, reforzar objetivos o enseñar contenidos específicos (grupos de expertos).

※ Grupos heterogéneos Son la base de trabajo de este tipo de aprendizaje, pues permiten desarrollar y poner de manifiesto multitud de aspectos esenciales (conflictos cognitivos, aprendizaje entre iguales, aprendizaje significativo, etc.).

Para la formación de grupos heterogéneos no debemos tener en cuenta la preferencia del alumnado, si bien sí que puede sernos útil su opinión para conocer sus prioridades y elecciones de grupo dentro la dinámica del aula. Así, separaremos en primer lugar la clase en tres subbloques. El primer bloque, que agrupa a una cuarta parte de la clase, engloba a los alumnos y alumnas más capaces de dar ayuda, que resuelven con éxito las tareas y son capaces de cooperar con el resto de compañeros y compañeras. El segundo bloque, que usualmente agrupa a otra cuarta parte de la clase está compuesto por el alumnado que necesita más ayuda. Por último, aproximadamente la mitad de la clase restante constituye la *"clase media"* de nuestro alumnado.

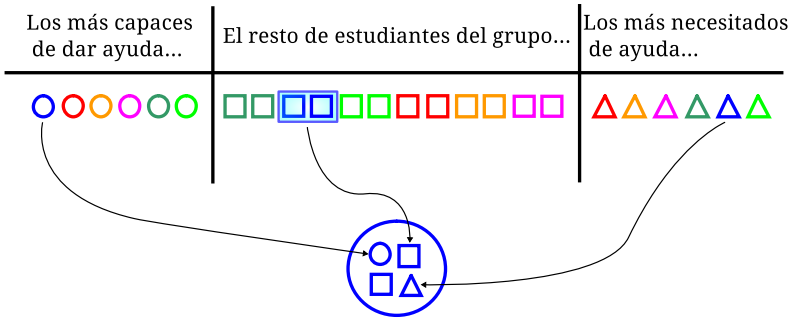


Figura 1.2.4: Formación de equipos

De nuevo, de acuerdo a la figura 1.2.4, formamos los equipos de trabajo, pudiendo hacer pequeñas correcciones para asegurar heterogeneidades en función de otros posibles criterios (etnia, religión, grupos paritarios, etc.). Finalmente, pueden ser útiles también pequeñas correcciones en base a la conflictividad del alumnado, para evitar grupos de amigos, etc.

1.2.2.1 Asignación de roles

Es importante asignar a cada alumno o alumna un rol dentro del equipo, que por otro lado será rotatorio y se realizará solo de manera temporal (al diseñar un plan de equipo, como veremos posteriormente). Los distintos roles que aquí proponemos son los siguientes:

- Coordinador: anima al equipo a completar el trabajo y dirige las revisiones de las tareas realizadas. Además, determina quién debe asumir las tareas si algún miembro del equipo se encuentra ausente. En caso de que los equipos sean de 4 alumnos puede ejercer también el papel de portavoz.
- Portavoz: habla en nombre del equipo cuando han de exponer una opinión o el profesor o profesora requiera una intervención.
- Secretario: se encarga de realizar las anotaciones en el cuaderno de equipo y es el responsable de su cuidado. Además, ha de tener control y conocimiento sobre el plan del equipo.
- Moderador: procura que el equipo no pierda el tiempo, controla el tono de voz y es el responsable de asegurar que cada miembro cumple con sus responsabilidades.
- Responsable de material: es el responsable del cuidado del material y de la zona de trabajo.

1.2.2.2 Cuaderno de equipo

El cuaderno de equipo es un documento esencial para desarrollar este tipo de aprendizaje, pues incluye todo lo referente al funcionamiento interno del alumnado y la evolución de las clases. Asimismo, sirve como herramienta de evaluación y autoevalución, y recoge todas las metas, objetivos y compromisos de nuestros alumnos y alumnas. Se divide en varias partes principales que detallamos a continuación, pero que pueden variar en función de diversas características: edad, grado de cooperatividad en las clases, mecanismos de evaluación...

En general, los puntos principales son los siguientes:

※ Nombre del equipo Cada grupo debe escoger un nombre y símbolo que den identidad al equipo de trabajo. De hecho, esta identidad de equipo potencia el sentido de pertenencia y cabe esperar que facilite la dinámica de trabajo. Puede emplearse alguna sesión de clase para que nuestros alumnos y alumnas elaboren el nombre, logo u otras identidades propias del equipo.

※ Componentes del equipo Se recoge el nombre de cada alumno y alumna perteneciente al equipo base que desarrollara su trabajo durante un período amplio de tiempo.

※ Normas de funcionamiento De forma negociada y en un debate con el alumnado, deben establecerse unas pocas normas, comprensibles y enunciadas en positivo (Lab! [9]). Ejemplos de posibles normas son las siguientes:

1. Respetaremos los cargos y a los compañeros y compañeras en nuestro equipo.
2. Bajaremos el tono de voz al trabajar en equipo.
3. Limpiaremos y cuidaremos el material de trabajo.
4. Apoyaremos a los compañeros y compañeras que necesiten ayuda.
5. Pediremos ayuda si la necesitamos.

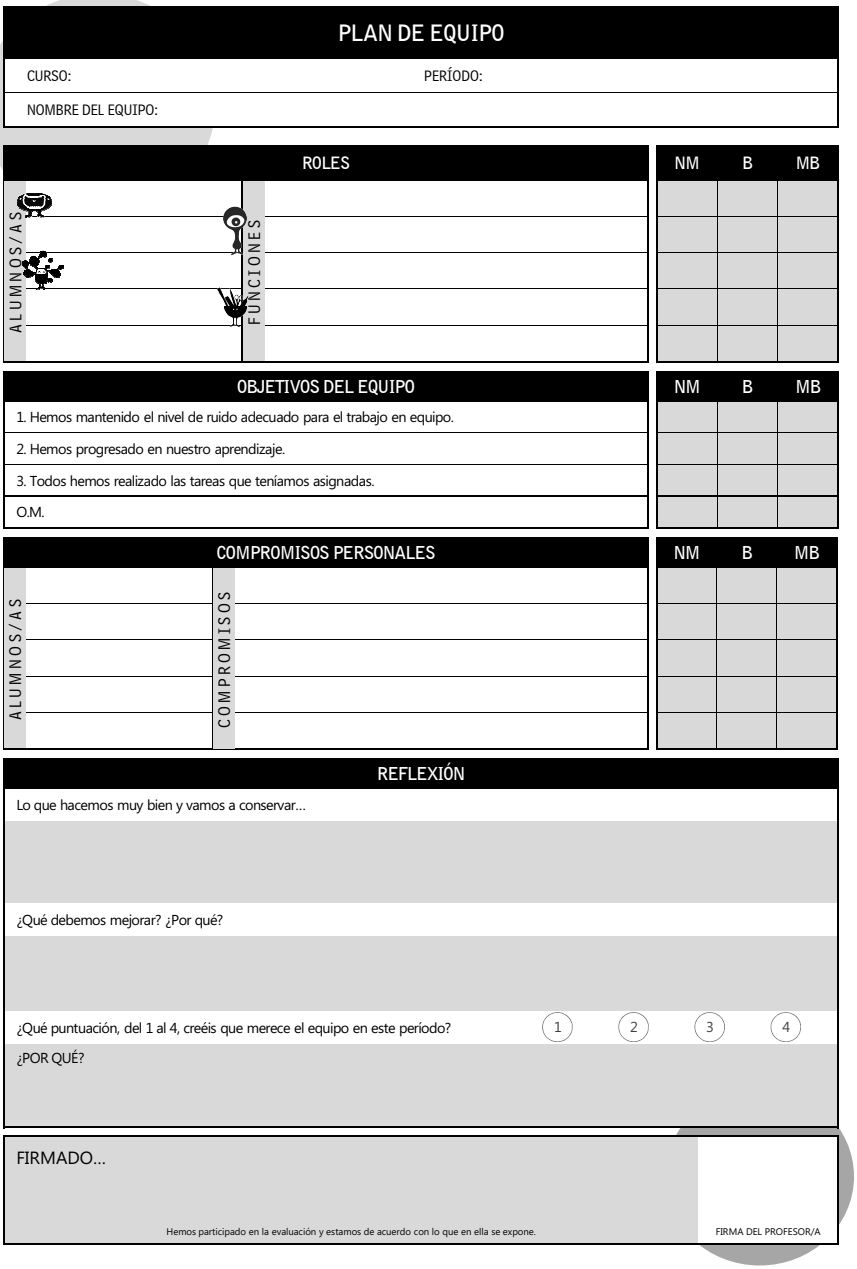
Una vez establecidas deben explicarse y asentarse en la dinámica de la clase. Además, al ser un conjunto de normas consensuadas con el alumnado no puede permitirse su infracción, debiendo estar muy pendientes de su cumplimiento, especialmente mientras se normaliza e implanta la estrategia de trabajo. No merece la pena fijar una cantidad excesiva de normas al comienzo de las sesiones de trabajo, sino ir llegando a su implantación por la propia dinámica de las clases. Por ejemplo, podemos esperar a que los equipos comiencen a trabajar y sea difícil incluso hablar entre los integrantes del equipo base para acordar con toda la clase la creación de una norma como la número 2.

※ Cargos y funciones Este apartado indica las funciones de cada cargo en el equipo para que puedan recordarlos con facilidad. Por tanto, no es una indicación de las rotaciones en los diversos cargos (que se incluirá en el plan de equipo), sino una descripción de sus funciones. Un posible ejemplo es el siguiente:

Roles	Funciones
Coordinador	▪ Anima al equipo a trabajar ▪ Revisa que el trabajo se realice correctamente ▪ Realiza el trabajo correspondiente si alguien falta
Portavoz	▪ Habla por el equipo cuando sea necesario
Secretario	▪ Rellena el cuaderno de equipo ▪ Cuida el cuaderno de equipo
Moderador	▪ Procura no perder el tiempo ▪ Evita perder el tiempo ▪ Controla el tono de voz
Responsable de material	▪ Cuida el material de trabajo ▪ Mantiene limpia la zona de trabajo

Cuadro 1.1: Ejemplo de tabla de cargos

※ Planes de equipo En ellos se recogen, al comienzo de cada tema a impartir, los objetivos y cargos que asume cada alumno o alumna dentro del equipo, compromisos personales concretos que asumen y firma de cada integrante. Un ejemplo de diseño para el plan de equipo es el siguiente:


PLAN DE EQUIPO

CURSO: PERÍODO:

NOMBRE DEL EQUIPO:

ROLES		NM	B	MB
ALUMNOS/AS	FUNCIONES			

OBJETIVOS DEL EQUIPO	NM	B	MB
1. Hemos mantenido el nivel de ruido adecuado para el trabajo en equipo.			
2. Hemos progresado en nuestro aprendizaje.			
3. Todos hemos realizado las tareas que teníamos asignadas.			
O.M.			

COMPROMISOS PERSONALES		NM	B	MB
ALUMNOS/AS	COMPROMISOS			

REFLEXIÓN

Lo que hacemos muy bien y vamos a conservar...

¿Qué debemos mejorar? ¿Por qué?

¿Qué puntuación, del 1 al 4, creéis que merece el equipo en este período? 1 2 3 4

¿POR QUÉ?

FIRMADO...

Hemos participado en la evaluación y estamos de acuerdo con lo que en ella se expone.

FIRMA DEL PROFESOR/A

Figura 1.2.5: Plan de equipo. Fuente: Lab! [9].

En él, junto con la especificación de las funciones de cada alumno o alumna se incluye una valoración de los objetivos que se rellenarán al final del tema. Además, los compromisos personales que adquieren al comienzo del tema también se autoevalúan y deben reflexionar al final del tema qué aspectos del equipo funcionan bien y cuáles no.

※ Diario de sesiones Es un documento donde el secretario del grupo realiza las anotaciones y el trabajo realizado en cada sesión de clase. Debe aparecer la fecha de realización y la firma de los componentes del equipo que estaban presentes. Se trata simplemente de un breve resumen a modo de acta, y no un lugar donde realizar las tareas de clase. De hecho, permite al secretario y al coordinador tener un control del trabajo realizado cada día.

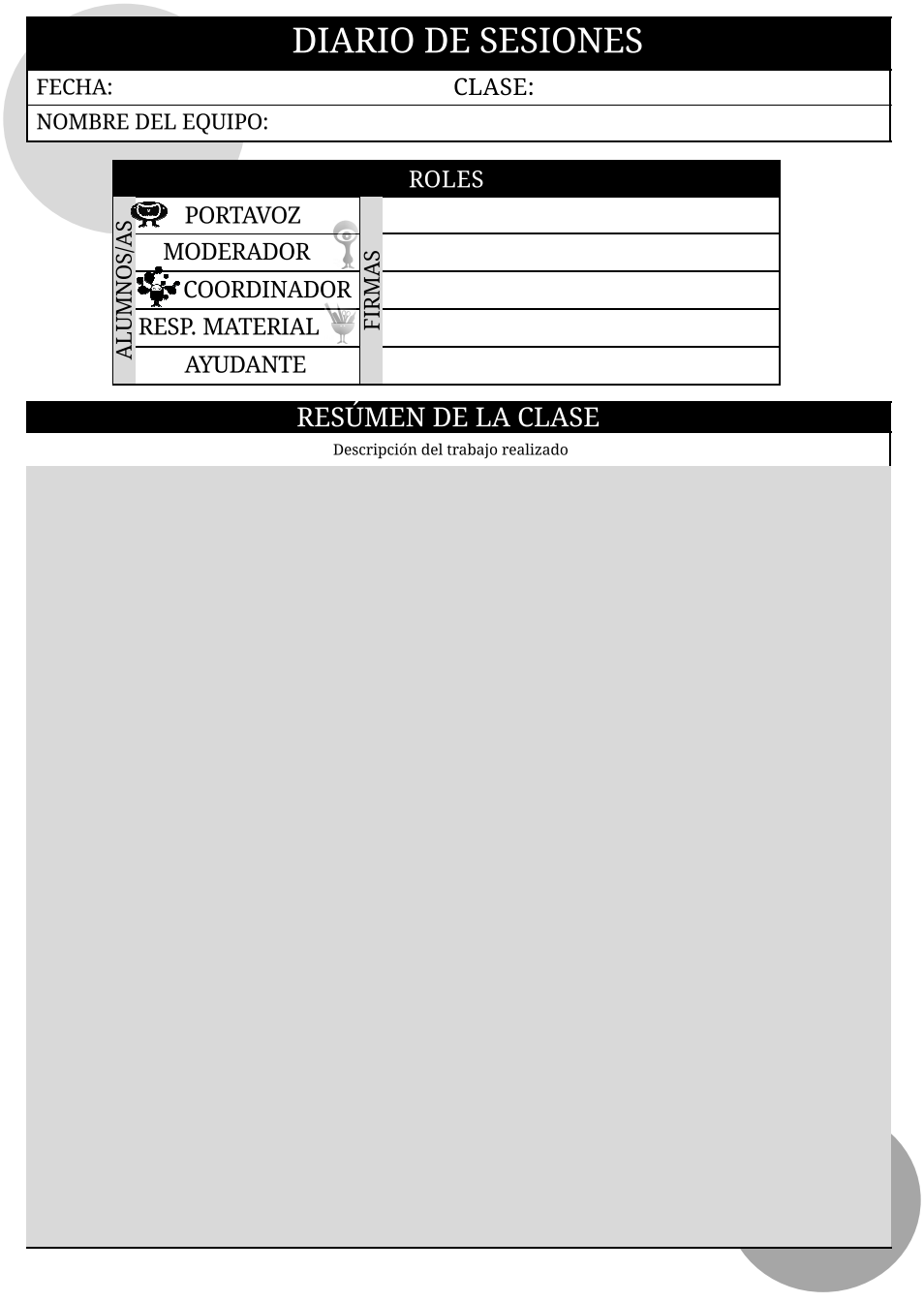
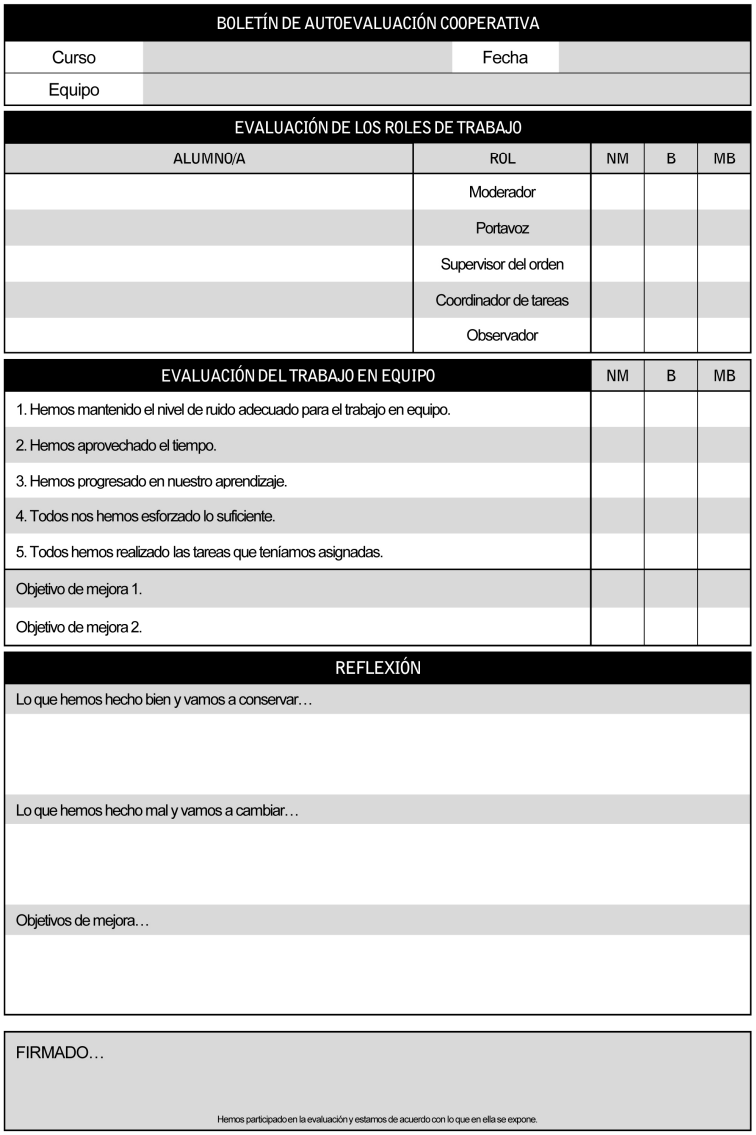


Figura 1.2.6: Diario de sesiones. Fuente: Elaboración propia a partir de la figura 1.2.5.

※ Revisiones periódicas y autoevaluaciones Es importante que los grupos se autoevalúen y valoren su trabajo de forma periódica, más allá de la tareas en equipo y consideraciones individuales que pueda realizar el profesor o profesora. Un ejemplo de ficha de autoevaluación es la siguiente:



BOLETÍN DE AUTOEVALUACIÓN COOPERATIVA

Curso		Fecha	
Equipo			

EVALUACIÓN DE LOS ROLES DE TRABAJO

ALUMNO/A	ROL	NM	B	MB
	Moderador			
	Portavoz			
	Supervisor del orden			
	Coordinador de tareas			
	Observador			

EVALUACIÓN DEL TRABAJO EN EQUIPO	NM	B	MB
1. Hemos mantenido el nivel de ruido adecuado para el trabajo en equipo.			
2. Hemos aprovechado el tiempo.			
3. Hemos progresado en nuestro aprendizaje.			
4. Todos nos hemos esforzado lo suficiente.			
5. Todos hemos realizado las tareas que teníamos asignadas.			
Objetivo de mejora 1.			
Objetivo de mejora 2.			

REFLEXIÓN

Lo que hemos hecho bien y vamos a conservar…

Lo que hemos hecho mal y vamos a cambiar…

Objetivos de mejora…

FIRMADO…

Hemos participado en la evaluación y estamos de acuerdo con lo que en ella se expone.

Figura 1.2.7: Ejemplo de autoevaluación grupal. Fuente: Lab! [9].

No obstante, la evaluación tendrá una parte individual que incluya el trabajo desempeñado por cada alumno o alumna en las tareas propuestas, y existe la posibilidad de realizar exámenes al final de cada tema. En cuanto a la evaluación grupal, debe considerar el cuaderno de equipo y el trabajo realizado de manera diaria. Asimismo, puede incluir la valoración de producciones finales como murales, trabajos escritos o presentaciones orales. Trataremos este tema con más detalle en la subsección 1.2.4 en la página 16.

1.2.3 Técnicas cooperativas

En general, el tipo de aprendizaje explicado en este capítulo es un método complejo que requiere de un tiempo de adaptación en nuestro alumnado. Son necesarios diversos automatismos, que deben desarrollarse con el tiempo, y que pueden potenciarse si se utiliza simultáneamente en diversas asignaturas. Por ello, no debemos prescindir de este tiempo de adaptación y tenemos que tener en cuenta que nuestros alumnos y alumnas están desarrollando capacidades sociales y actitudes de trabajo que pueden requerir un proceso que dificulte, solo al principio, la dinámica fluida de las clases.

Así, las técnicas de aprendizaje que se explican a continuación deben exponerse gradualmente. Es más, es recomendable emplear inicialmente solo una de las técnicas, y avanzar con otras nuevas cuando la técnica empleada esté ya completamente automatizada (quince días de asimilación, al menos) y la dinámica de la clase con este tipo de aprendizaje se desarrolle de manera fluida.

En un comienzo, adaptaremos las indicaciones propuestas por Lab! [9], limitando las dinámicas de cooperación (diez o quince minutos por clase que iremos ampliando de forma progresiva) y con una señal de ruido cero (por ejemplo, levantar las manos), que permitan hacer comentarios e indicaciones con un nivel de ruido mínimo.

1.2.3.1 Simples

※ 1-2-4 Adaptada por Pujolás Maset, Riera, Pedragosa y Soldevilla [16]. En ella debe contestarse a una pregunta realizada por el profesor o profesora mediante el siguiente esquema de trabajo:

1. Cada uno, de forma individual, piensa la respuesta a la pregunta formulada en el aula.
2. En parejas, comentan la respuesta que han pensado a la pregunta.
3. Finalmente, el equipo de trabajo en su conjunto decide qué respuesta es la más adecuada.

※ 4-2-1 Adaptada en Torrego y Negro [21]. Se proponen tres cuestiones o problemas similares a los equipos de base, que deben resolver de la siguiente forma:

1. El primer problema se resuelve por el conjunto del equipo.
2. El equipo comienza a trabajar el siguiente problema en parejas. Cuando finalizan el problema pueden consultar los resultados con la otra pareja.
3. El último problema se resuelve de manera individual, pudiendo solicitar ayuda a su pareja. Finalmente, las soluciones se comparan dentro del equipo.

El principal objetivo es aprovechar el andamiaje de conocimientos, retirando de forma progresiva la ayuda del equipo base.

※ Parada de 3 minutos Adaptada por Pujolás Maset y otros [16]. En el desarrollo normal de una explicación al gran grupo o clase, se realiza una parada de tres minutos, donde los equipos deben reflexionar sobre lo que se ha explicado hasta el momento. Además, deben plantear tres preguntas sobre el tema en cuestión.

Una vez cumplidos los tres minutos, cada equipo plantea una de las preguntas al resto de equipos. Si se proponen preguntas muy similares se sustituyen por otra de las que el equipo haya pensado.

Finalmente, terminado el turno de preguntas de todos los equipos, se prosigue con la explicación hasta una nueva parada de tres minutos.

※ Los cuatro Adaptada por Pujolás Maset y otros [16]. El profesor o profesora selecciona a cuatro alumnos o alumnas que conozcan en profundidad el tema que se está desarrollando, conformando un grupo de "sabios". Estos estudiantes, no obstante, deben preparar el tema para explicárselo a sus compañeros y compañeras. El día que se organiza la sesión, la dinámica es la siguiente:

1. Un miembro de cada equipo de base acude al grupo de sabios para que le explique el tema.
2. El alumno o alumna vuelve al equipo de base y explica al resto del equipo lo aprendido.

※ El número Elaborada por Ortiz de Pinedo Montoya y Alonso Guinea [12]. En esta estrategia el profesor o profesora propone una tarea determinada a la clase. Cada individuo dentro del equipo debe asegurarse de saber hacer la tarea, pero a su vez comprobando que el resto de compañeros y compañeras también saben realizarla. Una vez acabado el tiempo para hacer la tarea el docente elige un número al azar (existen numerosos generadores de aleatorios online que pueden sernos útiles), teniendo cada integrante dentro de cada equipo un número asignado. Finalmente, el alumno o alumna con el número elegido tiene que explicar ante la clase la tarea que han realizado o hacerla en la pizarra.

Pueden realizarse diversas variantes de esta estrategia simple. En primer lugar podemos elegir solo un número, por lo que seleccionaremos un único alumno o alumna de un equipo que explique la tarea, o asignar números del 1 al 4 y que seleccionemos una persona por cada equipo.

※ Lápices al centro El profesor o profesora da al equipo una hoja con tantas preguntas como miembros incluya. Cada estudiante se hará cargo de una de las preguntas. La metodología de trabajo debe realizarse como sigue:

1. Los lápices quedan en el centro (o en la mesa) para indicar que no puede escribirse. En este momento uno de los miembros del equipo lee en voz alta su pregunta, se asegura que todo el grupo expresa su opinión y comprueba que todos comprenden la respuesta acordada en el equipo.
2. Cada alumno o alumna, ahora sin hablar, responde a la pregunta tratada.
3. Vuelven a dejarse los lápices y se aborda otra pregunta.

Puede combinarse esta estrategia con la anterior, pudiendo elegir por sorteo un estudiante que saldrá y realizará una de las tareas propuestas frente al gran grupo.

※ Uno para todos En este caso los alumnos y alumnas realizan una serie de ejercicios dentro de sus grupos, elaborando una producción escrita individual de la tarea. Finalizado el tiempo de trabajo, el profesor o profesora recoge el trabajo realizado por uno de los miembros del equipo, lo corrige, y pone la calificación al equipo completo. El objetivo de esta estrategia es evaluar el trabajo que el individuo aporta al equipo.

※ El folio giratorio En este caso el profesor o profesora asigna una tarea a cada equipo (que puede consistir en una lista de palabras, palabras clave sobre un tema determinado, frase sugerente, etc.). Un miembro del equipo escribe la aportación sobre el tema en el folio y pasa al compañero o compañera a su izquierda el folio para que complete la información. No obstante, mientras un compañero o compañera escribe, el resto deben estar atentos, pues el producto final es responsabilidad del equipo en su conjunto. Puede considerarse la opción de que cada miembro escriba en un color diferente para observar su aportación individual y conocer las dinámicas de trabajo dentro de los equipos.

1.2.3.2 Complejas

※ Jigsaw ó puzzle Propuesta originalmente por Aronson y otros [1].

1. En este caso, la tarea se divide en tantas partes como miembros existen en cada equipo, entregando información de una parte distinta de la tarea a cada miembro del equipo. Cada persona tiene que entender la tarea de forma individual y explicársela posteriormente a sus compañeros y compañeras.

2. Posteriormente, se divide la clase en grupos de expertos. Es decir, reunimos a las personas que tienen tareas en común de los distintos equipos de trabajo, para analizar el conocimiento de la tarea entre ellos.

3. De nuevo, cada persona vuelve a su equipo y aclara los conceptos nuevos a sus compañeros y compañeras.

4. Para asegurar que la tarea ha sido comprendida correctamente, se selecciona un estudiante al azar y se pide que explique cualquiera de las partes de la tarea, que ya debe dominar.

※ Juegos de torneo Propuestos por Slavin [20] (TGT, por sus siglas en inglés). Trataremos una variante de la propuesta original. Una vez a la semana, el profesor o profesora realiza un cuestionario con una serie de preguntas y selecciona un número de equipo al azar por el que comenzar. De nuevo selecciona un número al azar para elegir al miembro del equipo que responderá a la pregunta del cuestionario. Cada respuesta positiva sumará puntos a la clasificación del equipo, lo que permite realizar un ránking al finalizar la ronda de preguntas y repartir premios positivos. De esta manera, cada alumno o alumna representa al equipo de trabajo en una competición con el resto de equipos, pero nunca de manera individual.

1.2.4 Evaluación

Para el alumnado, aquellos aspectos que no se evalúan pueden carecer de importancia. Por tanto, es fundamental incluir la evaluación de las dinámicas de los equipos en la calificación final, pues estaremos otorgando relevancia al aprendizaje cooperativo como algo que deben realizar correctamente.

Las propuestas realizadas hasta ahora generan una cantidad abundante de material susceptible de ser evaluado. Los planes de equipo, donde se observa si los objetivos que ellos mismos se proponen son alcanzados, y la realización de las actividades diarias, son una parte importante. Asimismo, el cumplimiento de las funciones asociadas a los roles dentro de los equipos y de los objetivos personales también deben ser evaluados, junto con aspectos actitudinales del trabajo diario.

Pese a que queda constancia escrita en el plan de equipo y el diario de equipo, donde deben estar todas las actividades realizadas en clase, la observación directa del trabajo y el interés mostrado por el alumnado juega un papel clave en la evaluación. De hecho, convierte parte de la evaluación en cualitativa.

Puede realizarse también, al final de cada trimestre, una autoevaluación por parte del gran grupo, donde ellos mismos valoren si el funcionamiento global y el cumplimiento de las normas establecidas ha sido el correcto a nivel de la clase. Allí, se exponen opiniones y se proponen mejoras y hándicaps del trabajo realizado. Dichas opiniones deben facilitar el cambio metodológico de los proyectos o actividades que abordan los distintos puntos del temario. A partir de esta evaluación, puede establecerse un Plan de Grupo más general y con objetivos concretos para toda la clase (Torrego y Negro [21]).

Además de la evaluación de los equipos y del grupo, no debe obviarse la evaluación individual. Estamos exigiendo un esfuerzo por trabajar codo con codo junto con los compañeros y compañeras que debe verse recompensado en las dinámicas intragrupo.

Estos aspectos deben suponer un porcentaje de la evaluación y responden a la calificación específica de las dinámicas cooperativas en el aula, pero queda aún otro porcentaje que pretende evaluar si se han conseguido los objetivos de la unidad didáctica o el trabajo por proyectos que hayamos diseñado para el año escolar. En este trabajo, no abordaremos la elaboración de unidades didácticas o proyectos orientados al aprendizaje cooperativo, pero puede encontrarse un capítulo completo en Torrego y Negro [21].

En caso de una unidad didáctica clásica, adaptada a una estructura de aprendizaje cooperativo, podemos proponer un porcentaje de examen en grupo y otro porcentaje de un examen individual donde queda recogido el esfuerzo y conocimientos de cada alumno y alumna por separado.

Por otro lado, si realizamos una unidad didáctica clásica, el examen grupal puede sustituirse por un producto final (trabajo escrito) o exposición, que en el caso de una estructura por proyectos sería más extensa y constituiría el producto final del tema o trimestre. La exposición final de un trabajo realizado por los equipos debe ser un buen indicador de si se han conseguido los objetivos inicialmente propuestos en el currículum. Tras cada exposición, los alumnos y alumnas deben proponer una nota a las exposiciones de manera totalmente anónima (ya sea de forma individual o por equipos), y otra al proyecto realizado y lo que en él han aprendido. Asimismo, el profesor o profesora debe aprovechar dichas exposiciones para realizar preguntas al equipo que expone y que pueda servir de test para comprobar el nivel de conocimiento sobre el trabajo que han realizado. Así, el estudiante es parte activa del proceso de evaluación al poner una calificación numérica a las exposiciones de los compañeros y compañeras.

De la misma manera, puede sustituirse en ciertos casos el examen individual por la entrega individual de actividades sobre pequeñas cuestiones del proyecto realizado y que recoja el temario explicado.

La evaluación de los puntos del currículum, o de las competencias alcanzadas, es un tema controvertido. Con el sistema de evaluación propuesto en la tabla 1.2, podemos asegurar que un alumno o alumna que apruebe el trimestre, habrá conseguido alcanzar al menos, las competencias que se establezcan en la legislación correspondiente. La recomendación, apoyada en Torrego y Negro [21], es centrar la evaluación en las competencias básicas[1] y en los procesos implicados, considerando los conceptos de las áreas como ingredientes que se alcanzarán inevitablemente de manera gradual. No obstante, si se pretende una evaluación que dé especial importancia a los elementos del currículum, puede establecerse una nota mínima en el examen individual para aprobar la asignatura en el trimestre (3 o 3.5, por ejemplo).

Porcentaje	Actividad
40 %	Aprendizaje cooperativo: actitud y objetivos alcanzados
20 %	Examen en equipo, exposición oral o trabajo final
40 %	Entrega de actividades individuales o examen final

Cuadro 1.2: Propuesta de evaluación

[1] Las aplicaciones desarrolladas en el capítulo 2 en la página 20 se orientan a 4º de ESO, por lo que las competencias básicas se enmarcan en la LOE 2/2006 y son: competencia en comunicación lingüística, competencia matemática, competencia en el conocimiento y la interacción con el mundo físico, tratamiento de la información y competencia digital, competencia social y ciudadana, competencia cultural y artística, competencia para aprender a aprender y autonomía e iniciativa personal.

1.2.5 Implementación en el aula

La implantación del aprendizaje cooperativo no es una tarea sencilla, pues el alumnado, y también el profesorado, deben asumir y aprender las dinámicas de trabajo que conlleva un cambio de una enseñanza directa a una estructura completamente nueva. En esencia, ambos necesitan "*aprender a aprender*".

Previo al proceso de implementación en el aula es aconsejable tener previstas y elaboradas, para cada unidad del temario, una serie de actividades "*esponja*", que absorban el tiempo entre los grupos que van terminando y el último en hacerlo o que realicen hasta la finalización del tiempo establecido. Un equipo base inactivo durante el tiempo de trabajo en equipo supone una fuente de ruido y distracción asegurada y es probable que dificulte una buena dinámica en la clase e incomode al resto de equipos que deben finalizar.

Por ello, una propuesta de adaptación del Laboratorio de Innovación de la Cooperativa José Ramón Otero (Lab! [9]), citado en Torrego y Negro [21], es dividir la implantación en el aula entre fases principales que ayuden a que el alumnado asuma las estrategias de aprendizaje cooperativo y no suponga una experiencia traumática y desalentadora para el profesorado que intente implementarlo.

※ Fase 1 Formación-orientación En primer lugar, debemos tener en cuenta que no existe estructura grupal en el aula, por lo que debemos generar un ambiente favorable para el establecimiento de nuevas estructuras. Esta fase puede servir, a su vez, para recoger información del alumnado y empezar a configurar, sobre el papel, posibles equipos de trabajo heterogéneos de larga duración. De hecho, es el momento de realizar pruebas al agrupar alumnos y alumnas.

El primer agrupamiento que podemos realizar es una estructura de parejas que se apoyen en el desarrollo de las clases. Podríamos empezar a enseñarles alguna estructura simple, pero muy clara y bien explicada, que desarrollen en parejas y afianzarles el objetivo final: trabajar en equipos base. La duración de las actividades debe establecerse en unos 5-10 minutos durante la clase.

En el caso de que el grupo responda favorablemente, podemos emplear estructuras simples como lápices al centro, 1-2-4 o 4-2-1 para dos parejas. De la misma forma, al final de esta fase, debemos explicar la función de los roles y qué se pretende con su establecimiento. Esta fase no debe durar un tiempo excesivo (más de dos o tres semanas, pues se trata de establecer rutinas básicas de funcionamiento).

※ Fase 2 Establecimiento de normas y conflictos En esta fase es el momento de implementar los equipos de 4 personas, heterogéneos y con duración mínima de un trimestre. Además, ampliaremos la duración de las actividades cooperativas a 15-20 minutos por clase, divididos en dos períodos para asegurar la eficacia, y de forma similar a lo establecido en la figura 1.2.1 en la página 4. Se establecerán también los roles dentro de cada equipo (aunque el establecimiento de funciones puede ser progresivo), debidamente explicados y asumidos y se implementarán técnicas simples como las expuestas anteriormente, pero de una en una y asegurándonos que funcionan correctamente.

No debemos olvidar que esta fase es crucial para el desarrollo del curso completo, y no debemos tratar las estructuras y características del aprendizaje cooperativo como algo innato en nuestro alumnado, sino como un contenido más a enseñar en nuestras clases: es algo que deben aprender.

Una vez pongamos en funcionamiento los equipos base es normal que surjan conflictos y tensiones dentro de ellos, que deben resolverse mediando con el alumnado y proponiendo objetivos en los planes de equipos que se realicen.

※ Fase 3 Rendimiento eficaz Una vez llegado a este punto los equipos son eficientes y productivos. De hecho, si por algún problema determinado hemos de modificar las agrupaciones podemos incorporar algún grupo de cinco alumnos. La duración de los equipos puede ser trimestral, o para el resto del curso, y la duración de las actividades cooperativas puede modelarse en función de la clase y los contenidos que queramos explicar, pero siempre deben dejarse pausas para aclaraciones y explicaciones como se indica en la figura 1.2.1 en la página 4.

Además, podemos incorporar las actividades complejas propuestas y con nivel de dificultad creciente, aunque deben estar bien explicadas y asimiladas. Pese a que la dinámica de los equipos debe estar establecida, no debemos olvidar recordar y reforzar los conceptos establecidos en la fase 2.

※ Fase 4 Finalización del curso Una o dos sesiones de evaluación y valoración final e intercambio de opiniones con el alumnado, pueden ser satisfactorias y ayudar a corregir errores de implementación para el siguiente curso académico. Es importante finalizar con sentimientos positivos y una valoración global del trabajo realizado.

Como reflejan Johnson, Johnson, Holubec y Vitale [8], citados en Torrego y Negro [21]: *"Implementar el aprendizaje cooperativo en el aula exige esfuerzo y disciplina. No es fácil. Pero vale la pena".*

Capítulo 2

Aplicaciones con VPython

***

2.1 Introducción

"Enseñar a los niños el uso del software libre en las escuelas formará individuos con sentido de libertad"

Richard Stallman

El desarrollo de aplicaciones orientadas a las clases de Física y Química que se desarrolla en este capítulo pretende facilitar un acercamiento por parte del alumnado a fenómenos inaccesibles desde el laboratorio. No obstante, suponen solo un punto de partida o herramientas útiles para que nuestros alumnos y alumnas indaguen en su desarrollo del conocimiento. De hecho, más allá del marco propuesto de aprendizaje cooperativo, es muy recomendable que se empleen también enmarcadas en un aprendizaje por proyectos (Maaβ y otros [10]) o como posibles soluciones a preguntas abiertas en nuestras clases que generen dudas en nuestro alumnado.

Las aplicaciones aquí desarrolladas y explicadas, se orientan a un nivel de 4º de ESO, pero pueden extenderse, adaptarse y emplearse fácilmente para otros niveles educativos donde también se explican conceptos similares (1º y 2º de ESO o 1º de Bachillerato). Además, es muy aconsejable el uso de aplicaciones similares para ilustrar conceptos en Física de 2º Bachillerato.

Existen numerosas aplicaciones y desarrollos previos en multitud de lenguajes de programación (Fortran, C, Applets en Java, etc.), pero pretendemos desarrollar códigos abiertos y bajo software libre, que puedan ejecutarse en un sistema multiplataforma. Python garantiza sencillez en los códigos y suficientes librerías y herramientas para desarrollar nuestro objetivo. Además, herramientas como VPython permiten visualizar y crear escenas dinámicas especialmente didácticas para nuestros alumnos y alumnas.

Podemos encontrar multitud de propuestas ya realizadas por diversos autores (García Corzo [6], Rojas, Morales, Rangel y Torres [19], Rojas, Martínez y Morales [18], Robb [17], Urbano [22]), y una base de datos con aplicaciones en la página web oficial del proyecto VPython [24]. Para la creación de las aplicaciones que desarrollaremos a continuación, utilizaremos y modificaremos algunos códigos libres existentes con licencia Creative Commons[1], y generaremos nuevas aplicaciones explicando detalladamente su funcionamiento.

2.1.1 VPython

VPython [24] incluye el lenguaje de programación Phyton, más un módulo de gráficos en 3D denominado "*visual*", y su versión inicial fue desarrollada en el año 2000. Facilita la creación de pantallas y animaciones navegables en 3D, empleando una sintaxis sencilla que puede seguirse con una formación básica en programación. De la misma forma, es interesante su conexión con Python como lenguaje de programación. Esta estructura visual es especialmente didáctica porque la realización de experimentos es sencilla e intuitiva, tanto a niveles básicos, medios o incluso universitarios.

[1] Las licencias Creative Commons, nacen para compartir y reutilizar las obras de creación bajo ciertas condiciones. Con las licencias Creative Commons, el autor autoriza el uso de su obra, pero la obra continua estando protegida. Frente al "*copyright*" que quiere decir "todos los derechos reservados", las Creative Commons proponen "algunos derechos reservados".

En esta sección no nos centraremos en detalles sobre el desarrollo del código, pero pueden encontrarse unas breves nociones de programación en el lenguaje empleado en el Anexo A en la página 49.

Para trabajar con los programas aquí desarrollados, desde un sistema Debian GNU/Linux (como el Proyecto Debian, Ubuntu o Guadalinex, esta última promovida por la Junta de Andalucía), debemos instalar los paquetes "*python*" y "*python-visual*". Pueden ejecutarse de forma sencilla desde la línea de comandos empleando la ejecución *"python nombre.py"*. No obstante, VPython proporciona en la web un sistema gráfico de ejecución multiplataforma para cualquier sistema operativo.

2.2 Aplicaciones

"La física es el sistema operativo del universo"
Steven R Garman

Para cada una de las aplicaciones desarrolladas indicamos el modelo seguido, un esquema de su desarrollo y los resultados de las simulaciones. El código de cada una de ellas se encuentra en el Anexo B en la página 67. Además, todo el material se encuentra disponible para su descarga y visualización en la página web indicada al final de este trabajo [R1].

En cuanto al uso de la ventana de cada aplicación, basta saber que el botón derecho del ratón permite girar la escena mostrada, y arrastrando el ratón con ambos botones pulsados (izquierdo y derecho) podremos hacer zoom sobre la imagen. Los datos necesarios para el funcionamiento de las aplicaciones se solicitan por pantalla, por lo que pueden seleccionarse diversos parámetros sin necesidad de acceder o modificar el código del programa.

2.2.1 Cinemática: Tiro parabólico

La primera aplicación desarrollada es un problema estudiado ampliamente en diversos cursos educativos (2º ESO, 4º ESO y 1º Bachillerato), aunque nosotros lo enfocaremos principalmente a su aplicación en 4º ESO. Supone un punto de partida que puede derivar hacia experimentos de caída libre, el estudio de la gravedad o multitud de problemas presentes en la vida cotidiana.

La cinemática es la parte de la mecánica que trata el movimiento en sus condiciones de espacio y tiempo, sin tener en cuenta las causas que lo producen, limitándose al estudio de la trayectoria en función del tiempo.

A partir de conceptos sencillos como la posición, velocidad y aceleración se derivan los diversos movimientos, cuyo estudio está completamente integrado en el currículum (Movimiento rectilíneo uniforme y uniformemente acelerado).

Figura 2.2.1: Trayectoria parabólica: fuente de agua. Fuente: (Wikipedia [26])

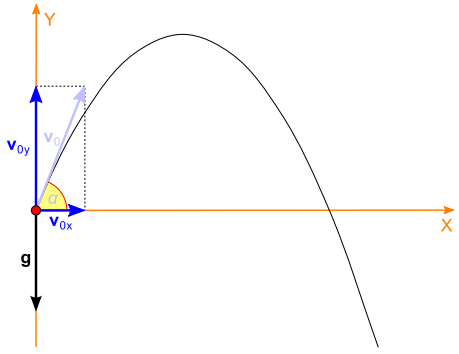


Figura 2.2.2: Análisis vectorial. Fuente: (Wikipedia [26])

2.2.1.1 Modelo

Partiremos de las ecuaciones que definen la velocidad y la aceleración de un cuerpo:

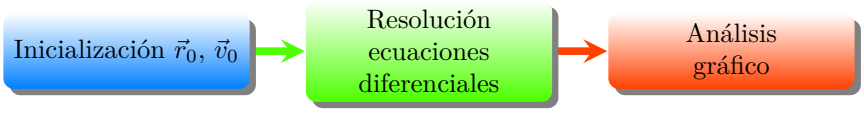
$$\begin{cases} \vec{v} = \frac{d\vec{r}}{dt} \\ \vec{a} = \frac{d\vec{v}}{dt} \end{cases} \tag{2.2.1}$$

lo que nos conduce a nuevas ecuaciones para modificar la posición y la velocidad del cuerpo,

$$\begin{cases} \vec{r} = \vec{r}_0 + \vec{v}dt \\ \vec{v} = \vec{v}_0 + \vec{a}dt \end{cases} \tag{2.2.2}$$

donde solo debemos conocer la posición y velocidades iniciales, y el valor vectorial de la aceleración. Además, $dt \to 0$ para asegurar la precisión numérica en nuestro código y evitar errores numéricos.

※ Esquema



2.2.1.2 Simulación

El código construido se basa en el sistema de coordenadas de la figura 2.2.3.

El programa necesita los siguientes parámetros:

- Por defecto el punto de lanzamiento es el $(0, 0, 0)$.
- La aceleración queda fijada en función de la gravedad $\vec{g} = (0, -g, 0)$, donde $g = 9{,}81\frac{m}{s^2}$.
- Solicita por pantalla las componentes de la velocidad v_x y v_y.

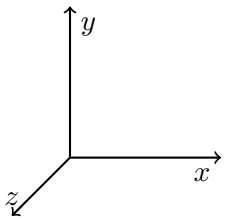


Figura 2.2.3: Eje de coordenadas

Y muestra el siguiente conjunto de datos:

- Trayectoria y vectores de la velocidad en cada punto.
- Aceleración, componentes de la velocidad y de la posición en función del tiempo.
- Altura máxima y tiempo transcurrido.

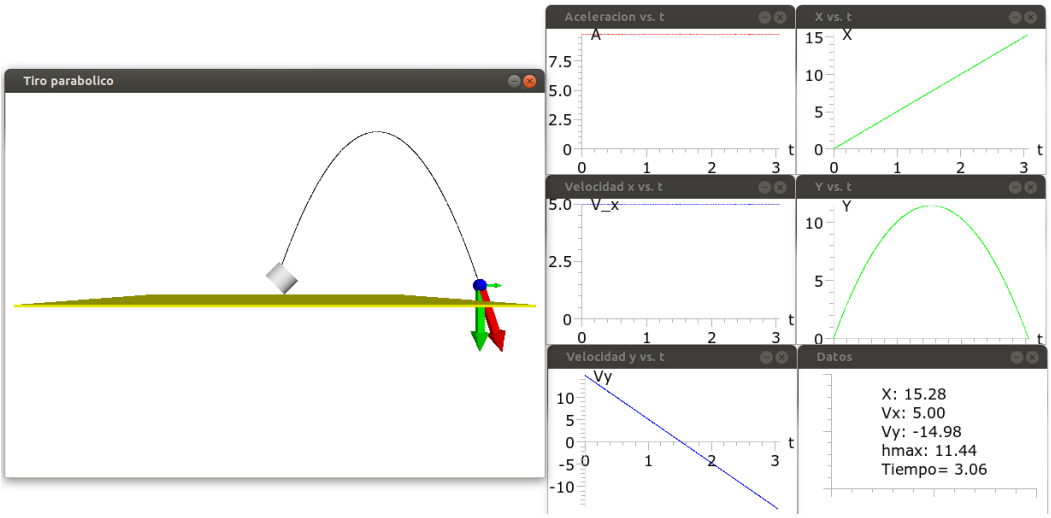


Figura 2.2.4: Ejemplo de funcionamiento con valores de la velocidad $v_x = 5$, $v_y = 15$.

El conjunto de datos mostrado al ejecutar la aplicación es completo, y sirve para resolver la mayoría de cuestiones que pueden plantearse en clase. Por tanto, permite hacer experimentos mediante el empleo de simulaciones que serían complicados en laboratorio (o bien, necesitarían un mayor tiempo de preparación y desarrollo, sin asegurar que sean exitosos para el aprendizaje) y que ayudan al alumnado a comprender la teoría de forma práctica e interactiva. De hecho, ejemplos de experimentos que pueden tratarse bajo simulación se proponen posteriormente en el capítulo 3.

2.2.2 Energía: Planos inclinados

Esta aplicación aborda el problema de la caída en un plano inclinado. Permite el estudio de distintas partes del temario, desde el movimiento en el plano, al tratamiento del problema por conservación de energía. De hecho, complementa la aplicación anterior, al añadir el uso de fuerzas y el estudio de los principios de Newton al movimiento de los cuerpos.

Por otro lado, más allá de la mera visualización, pretende servir para realizar simulaciones que permitan resolver problemas en tiempo real y tener en cuenta consideraciones físicas implícitas en la aplicación y que veremos posteriormente en el capítulo 3.

2.2.2.1 Modelo

En primer lugar, para resolver este problema hemos de tener en cuenta la resolución del plano inclinado.

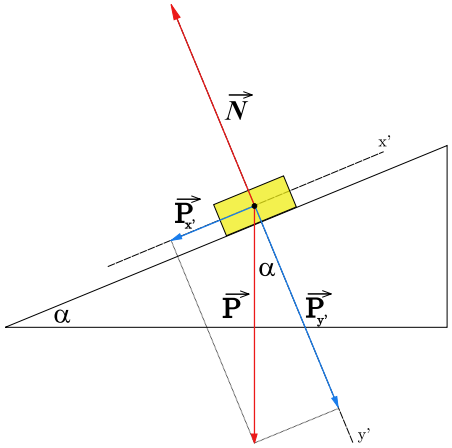


Figura 2.2.5: Sistema de coordenadas rotado

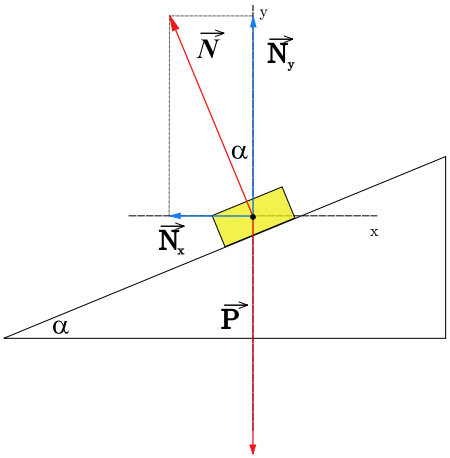


Figura 2.2.6: Sistema de coordenadas no rotado

A partir de la figura 2.2.5 se obtienen las ecuaciones para la fuerza, por componentes,

$$\begin{aligned} F_{y'} &= N - P_{y'} = N - mg\cos\alpha \\ F_{x'} &= P_{x'} = -mg\,\mathrm{sen}\,\alpha \end{aligned} \tag{2.2.3}$$

No obstante, para la simulación, necesitaremos cambiar el sistema de coordenadas, pues nuestro sistema de referencia se encuentra situado en la esquina inferior derecha del plano inclinado, y no está rotado, complicando la resolución del problema. Para ello, basta aplicar la siguiente matriz de rotación en dos dimensiones al vector $\vec{F}$,

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\,\mathrm{sen}\,\alpha \\ \mathrm{sen}\,\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix} \tag{2.2.4}$$

Y aplicando la ecuación 2.2.4 sobre las componentes de la fuerza (2.2.3), teniendo en cuenta que podemos aplicar la condición $F_{y'} = 0$, y que la rotación es de ángulo $-\alpha$,

$$\begin{aligned} F_y &= -mg\,\mathrm{sen}^2\,\alpha = ma_y \\ F_x &= -\tfrac{mg}{2}\,\mathrm{sen}\,\alpha\cos\alpha = ma_x \end{aligned} \tag{2.2.5}$$

Conocida la fuerza, y teniendo en cuenta que nos encontramos en un movimiento con aceleración constante, basta aplicar las siguientes ecuaciones,

$$\begin{cases} x = x_0 - v_{0x}t - \frac{1}{2}a_x t^2 \\ y = y_0 - v_{0y}t - \frac{1}{2}a_y t^2 \end{cases} \tag{2.2.6}$$

siendo $v_{0x} = v_{0x'} \cos\alpha$ y $v_{0y} = v_{0x'} \operatorname{sen}\alpha$, pues normalmente la velocidad inicial se da en el sistema de referencia mostrado en la figura 2.2.5.

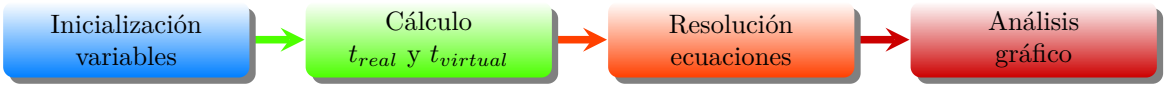
Aun así, debemos tener en cuenta la diferencia entre tiempo real (t_r) y tiempo simulado (t). Para calcular la transformación $t \to t_r$, que puede ser una función compleja $f(\alpha, L)$, basta realizar una vez la simulación y conocer el tiempo que tarda en caer la pelota en la simulación, pues el tiempo final real es conocido.

Para el cálculo de las magnitudes que se muestran en pantalla al ejecutar la aplicación, se emplean las ecuaciones usuales, siendo L la altura del plano inclinado.

$$\begin{cases} e = vt_r + \frac{1}{2}g \operatorname{sen}\alpha t_r^2 \\ v = v_0 + g \operatorname{sen}\alpha t_r \end{cases} \tag{2.2.7}$$

$$\begin{cases} E_c = \frac{1}{2}mv^2 \\ E_p = mg(L - e \operatorname{sen}\alpha) \end{cases} \tag{2.2.8}$$

※ Esquema



2.2.2.2 Simulación

La aplicación solicita por pantalla tres parámetros:

1. Altura del plano inclinado (en metros).
2. Ángulo del plano inclinado (en grados).
3. Velocidad inicial de la pelota ($\frac{m}{s}$).

y al ejecutarse muestra tanto la caída de la pelota como las siguientes variables en tiempo real:

1. Tiempo transcurrido.
2. Espacio recorrido por la pelota.
3. Velocidad de la pelota.
4. Energía cinética (en negro) y energía potencial (en naranja).

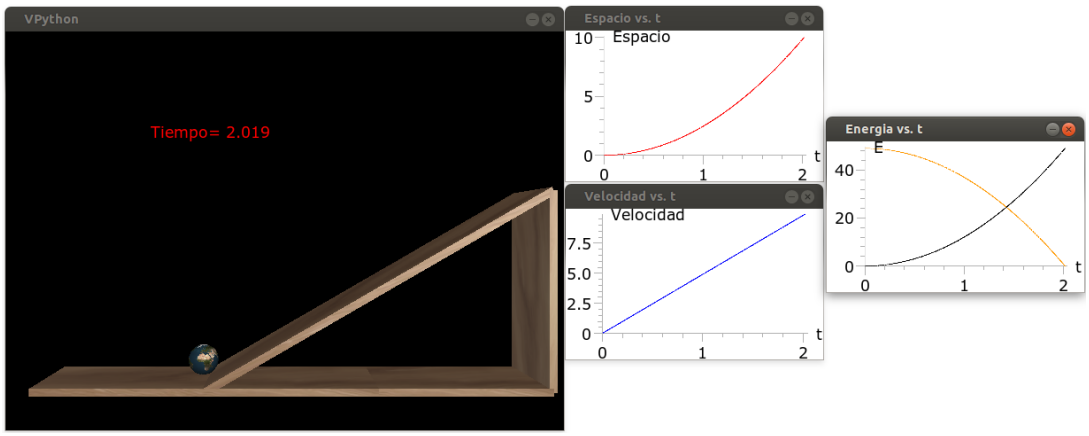


Figura 2.2.7: Captura de pantalla de la aplicación

2.2.3 Partículas de un gas

La aplicación es una modificación y traducción del código desarrollado por Urbano [22]. Trata la teoría cinética de los gases ideales y permite observar el comportamiento de las partículas al subir o bajar la temperatura del mismo.

2.2.3.1 Modelo

En este caso la simulación del gas se compone de las siguientes partes principales:

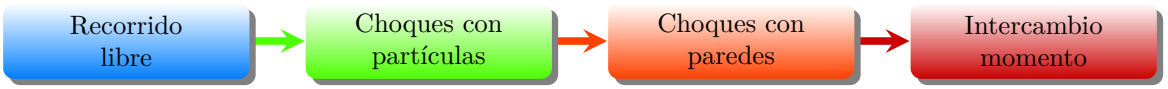
1. Recorrido libre de las partículas, con movimiento uniforme, pues se desprecia la fuerza gravitatoria.
2. Comprobación de choques entre partículas, si la distancia entre ellas es menor que dos veces el radio de cada una (modelo de esferas duras).
3. Comprobar choques con las paredes.
4. Intercambio de momento en caso de que se produzca algún choque.

El principal hándicap de este tipo de algoritmos es que para cada paso temporal, debemos comprobar la posición de todas y cada una de las partículas respecto a las otras para efectuar los choques. No obstante, funciona razonablemente bien para un número pequeño de partículas.

Por último, para el cálculo de la presión se ha empleado la fórmula del gas ideal,

$$PV = Nk_BT \tag{2.2.9}$$

donde P es la presión, V el volumen, N el número de partículas, k_B la constante de Boltzmann y T la temperatura.



2.2.3.2 Simulación

Las partículas del gas se mueven más rápido o más despacio dependiendo de la temperatura.

Además, el color de las partículas cambia en función de sus energías cinéticas individuales de forma análoga a la ley de Wien. Así, una mayor energía cinética corresponde a una temperatura más alta y conlleva longitudes de onda de la luz más cortas (azul). Es fácil observarlo modificando la posición de la barra de temperaturas.

Por otro lado, al alcanzar el 0 absoluto, el movimiento de las partículas se detiene (prácticamente) y las partículas desaparecen (se convierte en negro), al no emitir luz.

La aplicación solicita por pantalla el tamaño de la caja y el número de átomos que queremos incluir (hasta 100 para una visualización rápida), y a continuación muestra los parámetros:

- Colisiones con las paredes, entre átomos y totales.
- Temperatura, número de átomos, volumen y presión.

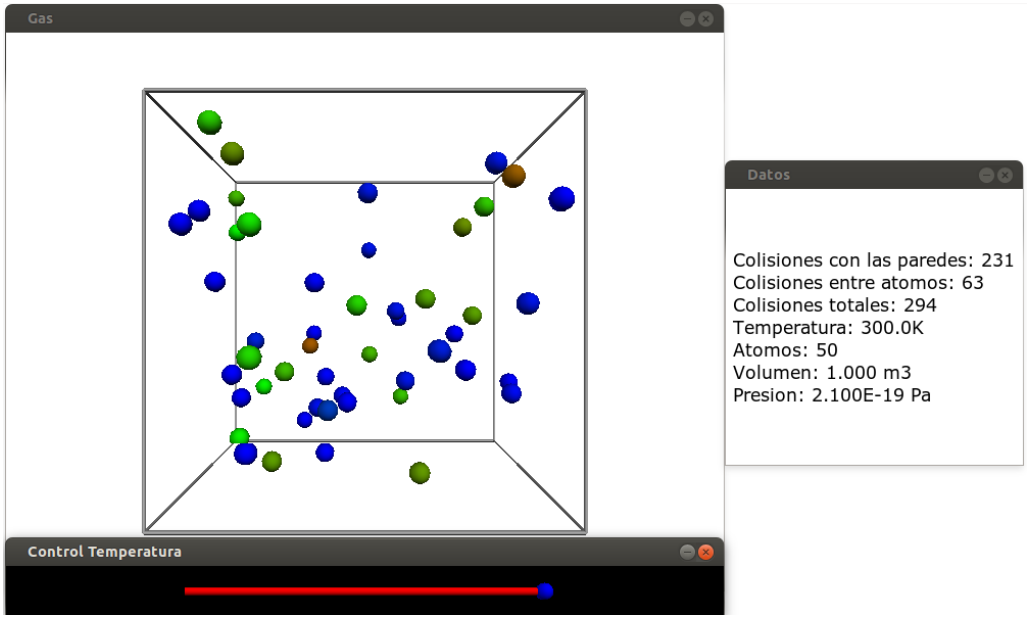


Figura 2.2.8: Captura de pantalla para una caja mediana con 50 átomos

2.2.4 Movimiento planetario

El movimiento planetario ha atraído al hombre desde la Antigüedad, ya en la antigua Grecia se establecen modelos heliocéntricos (Aristarco de Samos), pero se asumen los modelos geocéntricos de Ptolomeo hasta el siglo XVI. Observar y entender qué es un epiciclo (órbita circular de un planeta que gira en otra órbita circular en torno a la Tierra) puede ser una tarea compleja, y más aún entender su comportamiento y trabajar con este concepto. De hecho, es muy útil que nuestro alumnado comprenda la evolución histórica del pensamiento humano, y cómo surgen los conceptos progresivamente, entendiendo el porqué de ciertas explicaciones ahora consideradas erróneas (o más complejas).

El código aquí propuesto es una ampliación y adaptación de la aplicación creada por Robb [17], que permite observar el movimiento planetario eligiendo un sistema de referencia determinado (el Sol, o alguno de los planetas interiores), y para observar las diferencias existentes entre ellos.

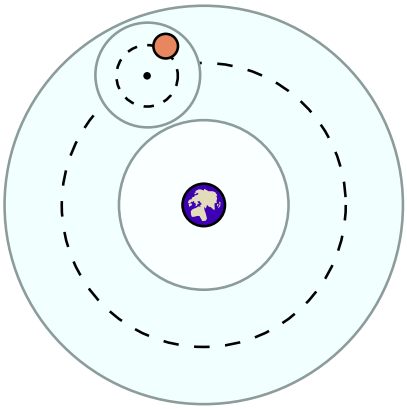


Figura 2.2.9: Astronomía ptolemaica, mostrando un planeta en un epiciclo (línea punteada pequeña) y la deferente (línea punteada grande). Fuente: Wikipedia [25]

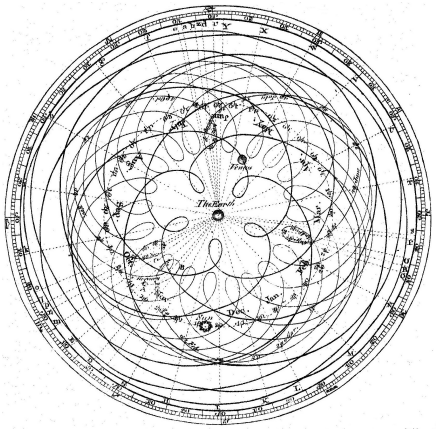


Figura 2.2.10: Modelo geocéntrico: órbitas. Fuente: Wikipedia [25]

2.2.4.1 Modelo

En este caso no resolveremos las ecuaciones diferenciales de los distintos planetas para calcular su órbita en torno al sol, sino que partiendo de los datos conocidos (masa, densidad, radio orbital, etc.), estableceremos un modelo que nos permita observar qué ocurre al cambiar los sistemas de referencia en el Sistema Solar.

Un punto importante que debe tenerse en cuenta al diseñar este tipo de aplicaciones es emplear una escala correcta. No obstante, las distancias entre planetas y sus tamaños son tan diferentes en orden de magnitud que sería imposible visualizar correctamente la aplicación. Por ello, se emplean distancias astronómicas para mostrar la distancia al Sol y se toma el radio de la tierra como la base para calcular el radio del resto de planetas, logrando un equilibrio entre el tamaño de las órbitas y el tamaño de los planetas. Pese a todo, el radio del Sol sí que debe establecerse en unidades astronómicas si pretendemos observar las órbitas de los planetas más cercanos.

Como hemos comentado, la complejidad de establecer ecuaciones para los epiciclos sería casi imposible, por ello, el código se fundamenta en tres pasos principales:

1. Calcular la posición del planeta de referencia (fija).
2. Calcular la posición del Sol (circular al planeta de referencia).
3. Calcular la posición de terceros cuerpos (circular al sol).

Por tanto, la única diferencia objetiva es situar o no al Sol como sistema de referencia, pues al situar como punto central cualquier planeta serán necesarios los epiciclos para explicar el movimiento del resto de objetos. No obstante, debemos tener en cuenta que simplemente estamos mostrando el movimiento de los planetas y no teniendo en cuenta las ecuaciones de Newton para resolver el problema. Además, se utiliza un movimiento circular y no elíptico y todos los planetas se sitúan en el mismo plano.

※ Esquema

2.2.4.2 Simulación

En primer lugar, el programa despliega un menú con tres opciones que nos permiten elegir el número de planetas que se mostrarán por pantalla:

1. Planetas internos (Mercurio, Venus, Tierra y Marte).
2. Planetas clásicos (Mercurio, Venus, Tierra, Marte, Júpiter y Saturno).
3. Nueve planetas.

Por defecto, recomendamos emplear los Planetas clásicos, pues visualmente es la opción mejor optimizada.

En segundo lugar, nos pide establecer qué planeta queremos como sistema de referencia (podemos elegir cualquiera dentro de la opción anteriormente seleccionada). De nuevo, si elegimos los planetas clásicos, la visualización de la órbita está optimizada para mostrar los epiciclos con la Tierra como sistema de referencia.

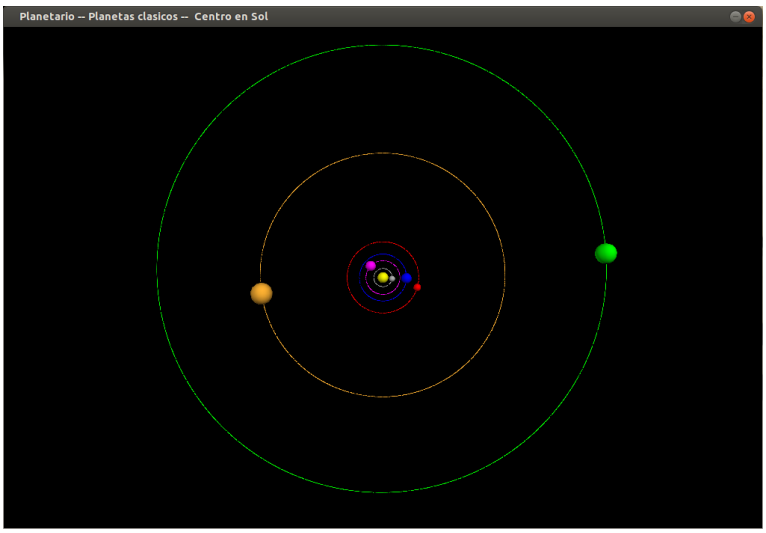


Figura 2.2.11: Sistema de referencia en el Sol

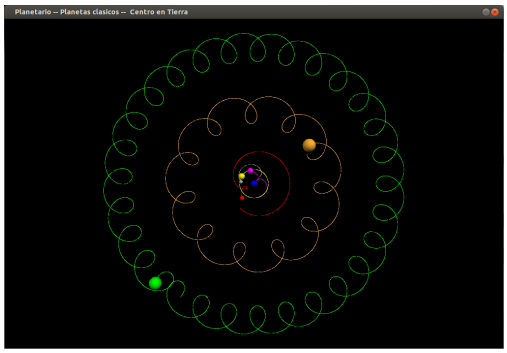


Figura 2.2.12: Sistema de referencia en la Tierra

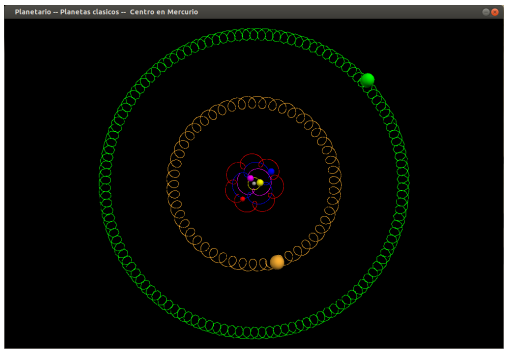


Figura 2.2.13: Sistema de referencia en Mercurio

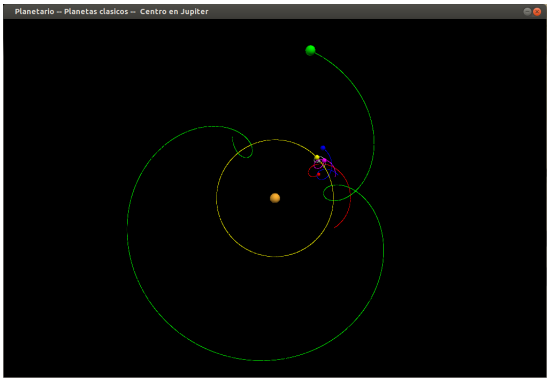


Figura 2.2.14: Sistema de referencia en Júpiter

Capítulo 3

Propuestas de trabajo

***

3.1 Actividades propuestas

"Es más fácil escribir diez volúmenes de principios filosóficos que poner en práctica uno solo de sus principios."

Leon Tolstoi

En este capítulo pretendemos aportar diversas actividades que pueden realizarse en el marco del aprendizaje cooperativo descrito en el Capítulo 1 empleando las herramientas desarrolladas en VPython en el Capítulo 2 y referentes a los conceptos y contenidos del currículum que allí deben explicarse. Para cada aplicación se han desarrollado dos propuestas de trabajo a modo de ejemplo. Cada una de ellas consta de cinco actividades que deben resolverse en grupos de aprendizaje, y que se prestan a ser actividades con ciertos componentes de respuesta abierta o que permiten el trabajo con técnicas simples y complejas.

Como ya se comentó en el capítulo anterior, las actividades están orientadas a un nivel de 4º ESO, en edades comprendidas entre los 15 y 16 años. Junto al trabajo por equipos, se busca en muchos casos que los ejercicios y problemas sean preguntas y cuestiones que generen conflictos cognitivos tanto a nivel individual como a nivel de equipo, y acompañar los problemas clásicos con cuestiones y problemas abiertos que puedan tener varias interpretaciones o puedan ser susceptibles de ampliarse para utilizarlos como actividades "esponja".

3.1.1 Cinemática: Tiro parabólico

3.1.1.1 Ficha 1

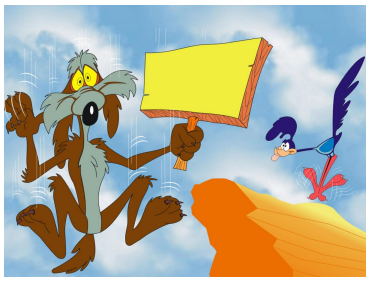
※ Indicaciones didácticas Para los ejercicios se pueden emplear, por ejemplo, estrategias simples como *"el número"*, *"1-2-4"* o *"lápices al centro"*. En cuanto a los problemas, para ambos puede utilizarse como recurso la aplicación 2.2.1 en la página 23.

Ejercicio. ¿Por qué tipo de movimientos está compuesto un tiro a canasta?

Ejercicio. ¿Podemos asegurar que no actúan fuerzas sobre un cuerpo que no tiene aceleración? ¿Puede un cuerpo seguir una trayectoria curva sin que actúen fuerzas sobre él?

Ejercicio. Lanzando a 45^o los siguientes objetos con la misma velocidad: una canica, una piedra, una bala de cañón. ¿Cuál llega más lejos? ¿Y si los dejamos caer desde un edificio de 60m?

Problema. En una de sus persecuciones tras el Correcaminos, el pobre Coyote (de 30 kg), cae por un precipicio de 50 m de altura. ¿Con qué velocidad aterriza el pobre animal?



Problema. Lanzamos una pelota con un ángulo de 30^o y una velocidad de $10\frac{m}{s}$. Calculad, despreciando el rozamiento con el aire:

1. La altura máxima que alcanza.
2. La velocidad que tiene cuando está a la mitad de la altura máxima.
3. La velocidad con la que llega al suelo.

3.1.1.2 Ficha 2

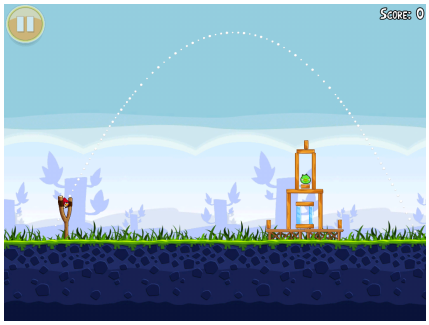
※ Indicaciones didácticas En este caso, de nuevo son útiles estrategias simples como "*1-2-4*" o "*lápices al centro*" para los ejercicios. En cuanto a los problemas, aplicación 2.2.1 en la página 23 es especialmente útil para el último. Para todos ellos, y de manera adicional, puede facilitarse el empleo de búsquedas en internet.

Ejercicio. ¿Con qué ángulo de lanzamiento se consigue un mayor alcance?

Ejercicio. Imaginad que lanzamos una bala de cañón desde cierta altura h. ¿Sería posible que la bala nunca tocara el suelo?

Problema. Un avión que vuela a 200 $\frac{m}{s}$ y a 900 metros de altura, deja caer un paquete. Calculad el punto donde caerá dicho objeto y a qué velocidad lo hará.

Problema. Elige un nivel del juego "*Angry Birds*" e intentad resolver (proponiendo medidas reales) el problema para alcanzar a algún "*Bad Piggie*".



Problema. Queremos encestar una pelota de papel en la papelera de la clase. ¿Qué variables debemos tener en cuenta antes de lanzarla? ¿Podrías resolver el problema para encestarla?

3.1.2 Energía: Planos inclinados

3.1.2.1 Ficha 1

※ Indicaciones didácticas Para el primer ejercicio y los últimos dos problemas es imprescindible el uso de la aplicación 2.2.2 en la página 26. Los otros dos ejercicios pueden realizarse mediante búsquedas en internet y bajo técnicas de aprendizaje simples como *"la parada de 3 minutos"* o "*1-2-4*", entre otras.

Ejercicio. ¿Puede ser negativa la energía cinética de un cuerpo?

Ejercicio. Observad la caída de la pelota en el plano. ¿Podéis explicar su movimiento? ¿Qué fuerzas actúan? Si actúan fuerzas, haced un esquema.

Ejercicio. Si dejamos caer una pelota de goma, tras pasar cierto tiempo deja de botar. ¿Falla el principio de conservación de la energía? ¿Por qué?

Problema. Un bloque de 5kg se lanza desde abajo en un plano inclinado con una velocidad de $5\frac{m}{s}$. El plano tiene una inclinación de 30^o. Resolved las siguientes cuestiones:

1. ¿Qué distancia recorre el bloque? ¿Cuánto tiempo tarda?
2. ¿Cuál es la altura máxima alcanzada?
3. ¿Con qué velocidad llega al suelo?
4. Indicad la energía cinética y potencial al comenzar el movimiento y en el punto más alto.

Problema. Como habréis observado en el uso de la aplicación, la pelota no rueda. ¿Sabríais explicar por qué? ¿Qué deberíamos hacer para que tarde más en caer?

3.1.2.2 Ficha 2

※ Indicaciones didácticas Tanto para los dos primeros ejercicios como para el último problema puede ser útil el empleo de búsquedas en internet por parte de los equipos, empleando técnicas como "*el número*", "*uno para todos*" o "*el folio giratorio*". En cambio, para el resto de ejercicios y problemas sí que se recomienda el uso conjunto de la aplicación 2.2.2 en la página 26 con búsquedas en internet y técnicas simples de aprendizaje cooperativo. De hecho, el último problema planteado se presta a la interpretación y resolución abierta por parte del alumnado.

Ejercicio. Hemos estudiado los planos inclinados. ¿Podríais poner ejemplos de planos inclinados que se usen en la vida cotidiana?

Ejercicio. ¿Cuándo se realiza más trabajo, al subir un cuerpo por una rampa hasta una altura de 2m o al elevarlo verticalmente?

Problema. Una chica se lanza con un trineo en una superficie helada y hasta la meta hay un desnivel de 10m. ¿Con qué velocidad llegará?

Problema. Un bloque de 5kg se deja caer desde el reposo en un plano inclinado. Se desliza $2m$ en un tiempo de 1,5s.

1. ¿Qué ángulo tiene el plano?
2. ¿Qué velocidad tiene en ese punto?
3. ¿Qué aceleración tiene en ese punto?
4. Si parte desde una altura de 5m ¿Qué altura tiene en ese momento?

Problema. Supongamos que se pretende subir un barril de agua al maletero de un vehículo ¿Cómo lo harías? Explicadlo detalladamente.

3.1.3 Partículas de un gas

3.1.3.1 Ficha 1

※ Indicaciones didácticas Para el segundo ejercicio es recomendable el uso de la aplicación 2.2.3 en la página 28. Además, para la resolución de los primeros dos ejercicios es muy útil que los equipos realicen búsquedas en internet. En cuanto a los problemas, pueden ser susceptibles de utilizar la técnica uno para todos y de planificar una sesión conjunta con la asignatura de tecnología o métodos de la ciencia para comprobar experimentalmente el tercer ejercicio, o emplear "*la parada de tres minutos*" para desarrollar el ejercicio entre la explicación de otros conceptos en una sesión de clase.

Ejercicio. Si de un gas conocemos el volumen, ¿es suficiente para conocer la cantidad de gas que tenemos?

Ejercicio. Como habréis observado en la aplicación, los átomos cambian de color al variar la temperatura. ¿Sabríais decir por qué? ¿Por qué pasan de negro a azul?

Ejercicio. ¿Cómo podemos introducir un globo medio inflado en un matraz? Si puede hacerse, explica que procesos físicos ocurren.


Problema. Una habitación tiene las siguientes medidas: 10m de largo, 5m de ancho y 3m de alto. Si la temperatura pasa de $10^o\,C$ a $25^o\,C$ al encender la calefacción, ¿Qué volumen de aire (a $25^o\,C$) entrará o saldrá de la habitación por las puertas y las ventanas?

Problema. Un globo de Helio para fiesta, que se supone es una esfera perfecta, tiene un radio de 18cm. A temperatura ambiente ($20^o\,C$), su presión interna es de 1.05 atm. Determinad el número de moles de Helio en el globo, el número de partículas y la masa de Helio necesaria para inflar el globo a estos valores.

3.1.3.2 Ficha 2

※ Indicaciones didácticas Para el primer ejercicio es imprescindible el uso de la aplicación 2.2.3 en la página 28. Podemos emplear "*la parada de tres minutos*" o "*el número*", y complementar la explicación del tercer ejercicio con algún experimento rápido si nos encontramos en el laboratorio o con alguna práctica en métodos de la ciencia o tecnología. Para los problemas es interesante emplear técnicas como "*uno para todos*", "*el número*" o "*4-2-1*".

Ejercicio. Aunque en la aplicación las partículas se mueven bastante rápido, la presión que se muestra es muy pequeña. ¿Por qué?

Ejercicio. En los pueblos de alta montaña lleva más tiempo cocinar las legumbres en agua hirviendo que en los tiempos de costa. ¿Por qué?

Ejercicio. Calentad, con la ayuda del profesor, una lata de refresco con 15ml de agua hasta que hierva. Una vez hecho esto sumergidla en agua fría. ¿Qué está ocurriendo?

Problema. Cierta cantidad de Helio, ocupa 3L a una presión de 1 atm y a $40^o\,C$. Calculad:

1. El volumen a $0^o\,C$.
2. La temperatura si el volumen fuera de $1L$.
3. Las partículas de gas que hay en la muestra.

Problema. Las bombonas de butano suelen tener 20L y 14atm de presión cuando están llenas a temperatura ambiente ($20^o\,C$). La bombona puede explotar si la presión supera las 30atm. ¿Podría explotar la bombona si se produce un incendio y conseguimos una temperatura de $400^o\,C$? ¿Qué temperatura necesitaríamos?

3.1.4 Movimiento planetario[1]

3.1.4.1 Ficha 1

※ Indicaciones didácticas Es recomendable complementar las explicaciones y el tercer ejercicio y el último problema con la aplicación 2.2.4 en la página 30, pues permite visualizar de forma muy clara el concepto de epiciclo. Además, se explicará al alumnado por qué no se respetan las escalas y puede mostrarse algún esquema con la escala correcta. De nuevo podemos emplear cualquiera de las técnicas cooperativas simples ya explicadas. Es muy interesante "*la parada de tres minutos*" para el primer ejercicio, dando posteriormente la explicación correcta a los equipos o desarrollar grupos homogéneos para la técnica de "*los cuatro*" en una sesión de clase.

Ejercicio. Si la Tierra atrae a la Luna, ¿por qué no "cae" ésta sobre la Tierra? ¿Y la Tierra sobre el Sol?

Ejercicio. La órbita de la Tierra alrededor del Sol es una elipse en la cual el Sol se encuentra en uno de los focos. ¿En qué época del año crees que está la Tierra más cerca del Sol? ¿Dependen las estaciones de la distancia de la Tierra al Sol? ¿Por qué?

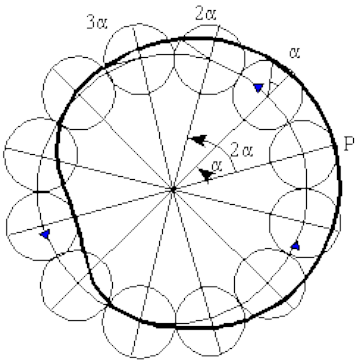
Ejercicio. Discutid por qué hubo que pasar del modelo de Aristóteles al de Ptolomeo. ¿En qué consiste el modelo de Copérnico y por qué fue tan bien aceptado?

Problema. El período de rotación de la Tierra alrededor del Sol es un año y el radio de la órbita es $1,5 \cdot 10^{11} m$. Si Júpiter tiene un período de aproximadamente 12 años, y si el radio de la órbita de Neptuno es de $4,5 \cdot 10^{12} m$, calculad:

1. El radio de la órbita de Júpiter.
2. El período del movimiento orbital de Neptuno.

[1] Las imágenes y diversos ejercicios de astronomía provienen de Villegas Martín [23].

Problema. A la combinación de movimientos consistente en que cada vez que el planeta da una vuelta sobre el epiciclo mientras que el centro del epiciclo da una vuelta sobre el deferente, ambos en el mismo sentido, la representaremos con la notación (1e, 1d). Como se puede ver en el dibujo, para obtener las posiciones sucesivas del planeta y trazar aproximadamente su órbita, hay que medir los ángulos como está indicado:



en la posición P' el centro del epiciclo ha avanzado un ángulo, arrastrando con él el propio epiciclo y al planeta situado en él. A su vez, el epiciclo ha girado un ángulo alrededor de su centro, de forma que la posición del planeta es la indicado. En las posiciones sucesivas se miden los ángulos de la misma manera.

Si con (Xe, Yd) notamos el epiciclo en el cual el planeta da X vueltas sobre el epiciclo, mientras que el centro del epiciclo da Y vueltas sobre el deferente (ambos en el mismo sentido), representad los epiciclos (3e,1d), (5e,1d) y (5e,2d).

3.1.4.2 Ficha 2

※ Indicaciones didácticas En este caso es recomendable emplear búsquedas en internet para los tres primeros ejercicios y técnicas cooperativas como "*4-2-1*" para los dos últimos problemas que presentan una dificultad algo mayor.

Ejercicio. ¿Cuánto pesaría cada integrante de vuestro equipo en los distintos planetas del Sistema Solar?

Ejercicio. ¿La Luna se ve durante el día? ¿Por qué no se ven las estrellas de día?

Ejercicio. Completa los dibujos (colocando la Luna) para que se produzca un eclipse de Sol y otro de Luna.

Discutid las siguientes cuestiones:

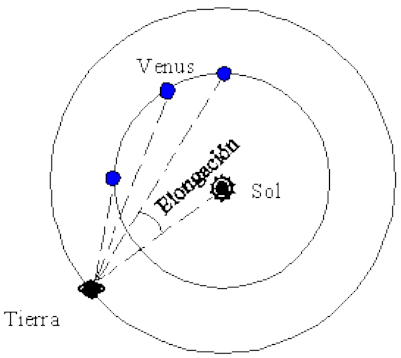
1. En el dibujo que representes el eclipse de Luna dibuja dos rectas tangentes a la vez a las circunferencias que representan la Tierra y la Luna. Sombrea la parte que queda entre el punto de corte y la tierra. A este cono se le llama cono de sombra. ¿A qué crees que se debe este nombre?

2. Si estuviesen la Tierra, el Sol y la Luna en el mismo plano, ¿cuántos eclipses se producirían al año?

3. Ya los babilonios determinaron que los eclipses se producían cada 223 meses lunares, por tanto, ¿es posible que los tres astros anteriores se encuentren en el mismo plano?, razona la respuesta.

Problema. Sabiendo que la distancia máxima de la Tierra a la Luna es de 407.600 Km (apogeo) y que la distancia mínima es de 356.700 Km (perigeo), dibuja a la escala que creas conveniente la Tierra, la Luna y la órbita que la Luna describe alrededor de la Tierra.

Problema. Sabiendo que el valor máximo de la elongación de Venus es de unos 46^o, calcula la distancia del Sol a Venus. (Nota: tomad distancia Sol-Tierra = 150.000.000 Km; este problema lo propuso Copérnico en su obra *de Revolutionibus*).



3.1.5 Ejemplo de sesión de clase

Pese a las indicaciones didácticas y los ejercicios y problemas explicados hasta ahora en los modelos de actividades que se han expuesto, en este apartado expondremos el desarrollo de una sesión de clase en la que suponemos que los equipos de trabajo están ya formados y pueden aplicar técnicas simples que habremos enseñado a nuestro alumnado previamente.

En este caso concreto, desarrollaremos los conceptos de energías, fuerzas y movimientos en un plano inclinado. La destreza principal que debe adquirir el alumnado en esta sesión es reconocer e identificar las fuerzas y movimientos que se producen en un plano inclinado y las energías involucradas en el proceso.

Capacidades	Conocimientos	Actitudes
Reconocer las fuerzas reales en un plano inclinado y el estado de movimiento que puede iniciarse. Tipos de energía en el sistema.	Las fuerzas como aquello que es capaz de modificar el estado de reposo o movimiento de un cuerpo.	Demuestran un buen trabajo en equipo y cooperación con una actitud positiva en el desarrollo de la clase

Cuadro 3.1: Capacidades, conocimientos y actitudes que deben desarrollar

La sesión está planificada para una duración de 60 minutos, un nivel de 4º de ESO y un alumnado dividido en equipos de trabajo de cuatro personas que es capaz de emplear técnicas de aprendizaje cooperativo ya asimiladas. A continuación, emplearemos la ficha 1 (subsubsección 3.1.2.1 en la página 36) sobre planos inclinados y la aplicación 2.2.2. Así, las acciones didácticas y la duración prevista es la siguiente:

※ Acciones de inicio (tiempo aproximado: 15 minutos)

El alumnado se distribuye en equipos de trabajo cooperativo (supondremos de 4 integrantes), y reciben una pequeña explicación donde se recuerdan los fundamentos de los planos inclinados y las fuerzas que actúan en dicho problema, que se asumen descritos en una sesión anterior. Se les proporciona además la ficha 1 y se muestra alguna simulación con la aplicación del plano inclinado en el ordenador del profesor o profesora (con $6\,m$ de altura, 30^o de inclinación y partiendo del reposo, por ejemplo).

※ Desarrollo de técnicas cooperativas (tiempo aproximado: 15 minutos)

Tras observar la caída de la pelota en la aplicación responden a las tres primeras cuestiones empleando las siguientes técnicas simples para cada una de ellas:

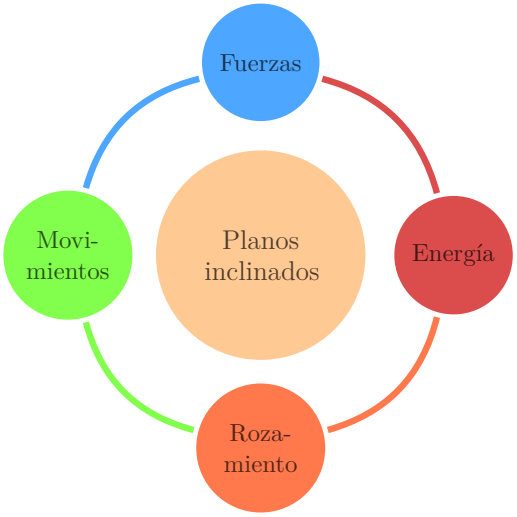
1. "*Parada de tres minutos*"
2. *"1-2-4"*
3. "*Parada de tres minutos*"

El secretario recoge y anota la respuesta a cada pregunta, archivándola en el cuaderno de equipo.

※ Aclaraciones y explicaciones (tiempo aproximado: 10 minutos)

Se comentan brevemente las cuestiones anteriores, asegurándonos que se comprende la respuesta correcta. Si el alumnado dispone de ordenador, se les permite que accedan a él para poder utilizar la aplicación desarrollada en la siguiente parte de la clase. En otro caso, se muestra la caída de la pelota de nuevo haciendo hincapié en que ésta no gira al ejecutar el programa. Preparamos a nuestro alumnado para la parte de finalización de la clase y hacemos una "*parada de tres minutos*" para que contesten a la última cuestión antes de abordar el problema.

※ Acciones de finalización (tiempo aproximado: 15 minutos) Por último, aplicando "*el número*" realizarán el problema propuesto en la ficha de trabajo, recogiendo uno de ellos al azar en cada equipo para corregirlo y evaluarlo posteriormente. Si disponen de ordenadores pueden resolverlo con ayuda de la aplicación, pero de cualquier manera, deben entregar el problema resuelto. En la siguiente sesión de clase se devolverá para que comprueben los errores y pregunten las dudas que puedan surgir.



※ Recursos

- Pizarra o pizarra electrónica.
- Acceso a ordenador para cada equipo o un proyector de clase.
- Ficha 2 en papel o mostrada a la clase.

※ Evaluación Los indicadores que seguiremos son los siguientes:

- Pueden reconocer y explicar la fuerzas que actúan y los movimientos que provocan.
- Emplean correctamente las unidades establecidas en el S.I. y las magnitudes físicas a las que corresponden.
- Conocen los conceptos de energía cinética y potencial y saben aplicarlos correctamente a situaciones reales.

Y para ello dispondremos de diversos instrumentos cualitativos y cuantitativos:

- Observación directa del trabajo en equipo y las cuestiones propuestas que deben responder. Las respuestas quedan recogidas en los cuadernos de equipo.
- Problema entregado a través de "*el número*", que evaluaremos mediante una calificación numérica y distribuiremos posteriormente para que comprueben sus propios errores.

3.2 Transversalidad

"La naturaleza no tiene ninguna culpa de los planes de estudios vigentes en escuelas y universidades"
Jorge Wagensberg

La temática tratada y la metodología propuesta fomentan un desarrollo interdisciplinar al conectar con el resto de asignaturas del currículum, permitiendo realizar un trabajo coordinado durante el desarrollo del curso académico. Además, el trabajo debe compaginarse de manera que se mantengan los equipos de trabajo en el máximo de asignaturas posibles, y tratando los temas de manera interdisciplinar y transversal.

※ Informática Es inmediata la conexión con esta asignatura. En primer lugar, es importante aprender a realizar búsquedas de información productivas y filtrar los contenidos que podamos encontrar en internet. La enseñanza e introducción de un lenguaje de programación como Python puede abordarse de manera sencilla y puede ser sumamente útil para nuestro alumnado. Es interesante también el desarrollo de productos finales como murales o presentaciones empleando ordenadores (a través de *LibreOffice Impress, LyX* o *Inkscape*), redacción y diseño de textos para elaborar el cuaderno de equipo (mediante *LyX o LibreOffice Writer*), así como el trabajo con plataformas donde se puede compartir información y potenciar la actitud de trabajo en equipo como Dropbox o Moodle, e incluso diseñar y realizar experiencias y vídeos que puedan subirse a webs como Youtube o blogs de aula creados para ello.

De manera adicional puede tratarse la generación de gráficas y tratamiento de datos en software como *Qtiplot* o *LibreOffice Calc.*

※ Matemáticas Por otro lado, es interesante y altamente recomendable compaginar el desarrollo del temario de Física con la asignatura de Matemáticas, pues es fácil poner en común la mayoría del temario y abordar contenidos muy similares. De hecho, en la parte de cinemática y dinámica es común la explicación de los vectores o es también parte común la astronomía y los modelos planetarios. En cuanto a las funciones, creación e interpretación de gráficas son fácilmente adaptables ambos temarios.

※ Lengua Castellana y Literatura La elaboración de productos finales y presentaciones, textos escritos y redacción de documentos, puede trabajarse de forma conjunta con esta asignatura. También es importante la preparación de exposiciones orales ante los demás equipos. Asimismo pueden tratarse textos de astronomía para su análisis y discusión.

※ Idioma extranjero (Inglés, francés...) Puede mejorarse el aprendizaje de la lengua extranjera realizando actividades, problemas y redactando en otro idioma. Si bien, parece recomendable que se traten textos y temarios propios de Física, Química o Matemáticas en la asignatura de lengua extranjera de forma complementaria, de forma anexa a los clásicos (comida, el tiempo, viajes, etc.). Además, en las búsquedas por internet podemos encontrar resultados en otros idiomas que pueden ser muy interesantes para nuestro alumnado.

※ Tecnología Esta asignatura facilita la realización de montajes experimentales de forma conjunta con Física y Química, pues presenta parte del temario común. Ejemplos de actividades que pueden realizarse son planetarios, experimentos de tiro parabólico o planos inclinados.

※ Métodos de la ciencia Pese a ser una optativa, pueden tratarse todos los temas de Física y Química de manera experimental. De hecho, puede abordarse un proyecto anual donde se elabore un producto final bien estructurado que se presente ante otras clases o compañeros de distinto nivel en el propio instituto.

※ Ciencias sociales (geografía e historia) y cultura clásica En la parte de astronomía, es especialmente interesante el estudio de textos y concepciones de la Antigua Grecia sobre el tema, y su conexión con la religión, mitos y pensamiento. Aunque geografía e historia si es una asignatura común, cultura clásica es una optativa que no coincide con el itinerario de física y química. No obstante, pueden hacerse grupos de expertos en física y química y cultura clásica, realizando sesiones de trabajo conjuntas entre grupos.

※ Educación Plástica y Visual La elaboración del logo y los elementos identitarios de cada grupo pueden realizarse en alguna sesión de esta asignatura.

※ Educación Física Pueden relacionarse con la asignatura de Física la mayoría de movimientos y disciplinas que abarca el atletismo (carreras, saltos, lanzamientos, pruebas combinadas y marcha) y otros deportes. Por ejemplo, usando conceptos de la misma para su aplicación práctica en la mejora de resultados en pruebas físicas.

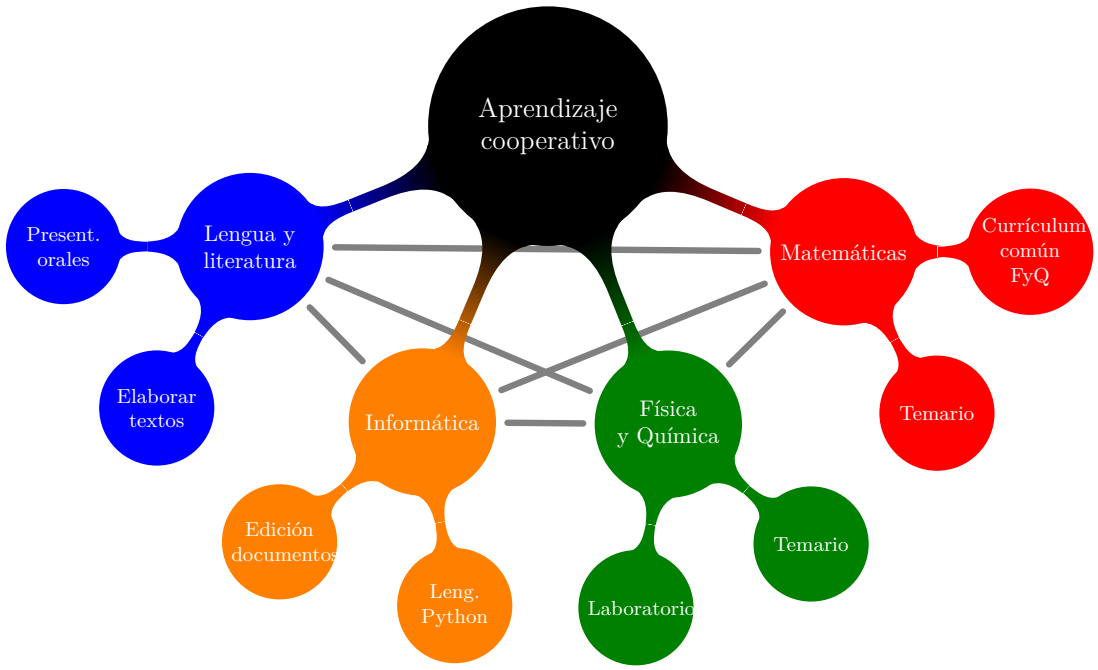


Figura 3.2.1: Gráfica donde se muestran algunas interacciones con otras áreas del currículum.

Concluyendo, deben plantearse dinámicas y estructuras de trabajo en equipo que trasciendan una asignatura aislada y conformen, de manera transversal, un bloque de contenidos interrelacionados que nuestro alumnado no perciba como materias disconexas y aisladas. Es sustancial mantener los equipos de aprendizaje cooperativo conformados y mantener las estructuras organizativas de la clase, así como una comunicación fluida entre el profesorado y con el alumnado.

3.3 Conclusiones

"Necesitamos otra educación para otra sociedad
y otra sociedad para otra educación."
Karl Marx

El aprendizaje cooperativo conforma la base sobre la que se elabora este trabajo. Numerosos autores abordan su desarrollo (Lab! [9], Pujolás Maset [13], Gavilán Bouzas [7], Slavin [20], Torrego y Negro [21]) y explican mucho más detalladamente cómo implementar y profundizar las estrategias aquí planteadas en el aula. Nuestro objetivo es que, como experiencia de enseñanza social y de participación activa de nuestros alumnos y alumnas, ayude a superar las estructuras de enseñanza tradicional que fomenten una educación basada en el individualismo o la competitividad.

Así, las aplicaciones desarrolladas en el capítulo 2 podrían haberse planteado en un marco de enseñanza tradicional con una breve inclusión de herramientas digitales. En este caso, no sería sino una simple inclusión de elementos digitales y herramientas TIC en el aula pero manteniendo las estructuras individualistas y competitivas que muchas veces allí se desarrollan. Para evitar esto, se toma como base una dinámica de clase que persiga la cooperación, solidaridad y apoyo mutuo entre el alumnado que trabaja en equipos de base, facilitando un aula inclusiva y que permite una constante atención a la diversidad. De la misma manera, se fomenta el aprendizaje entre iguales y el profesor o profesora puede actuar, una vez asentados los automatismos de funcionamiento en clase, como un mero guía que puede negociar con los alumnos y alumnas las temáticas tratadas y cómo abordarlas. De hecho, tanto el establecimiento de normas básicas en clase como el criterio de evaluación se propone negociado en el capítulo 1, aunque debe plantearse en base a unos mínimos y es fácil que se dirija a elementos de común acuerdo entre el profesorado y el alumnado.

Por otro lado, se han desarrollado herramientas en Python que se espera que sean especialmente útiles para su aplicación didáctica y como alternativa a la realización experimentos que muchas veces no pueden llevarse a cabo (a su vez es recomendable el uso de otras aplicaciones como las que se exponen en García Corzo [6] y VPython [24]). Lo cierto es que además, permiten potenciar la inclusión en el aula de tecnologías TIC con las que nuestros alumnos y alumnas tendrán que crecer y aprender a utilizar en la sociedad actual. Es importante que desarrollen la capacidad de manejar útilmente dichas herramientas.

Las actividades de trabajo propuestas persiguen que se emplee la propuesta didáctica del capítulo 1 de manera práctica, sin ser solo una exposición teórica del aprendizaje cooperativo. No obstante, estas actividades se exponen a modo de ejemplo, y no con el objeto de limitarnos simplemente a ellas. Además, las indicaciones didácticas pretenden mostrar modelos y sugerencias de aplicación en clase. La idea de realizar un documento autocontenido nos ha llevado a proponer un ejemplo de sesión de clase que acompañe también a estas actividades. Cabe destacar que este trabajo de fin de Máster se engloba en la modalidad de innovación educativa y materiales didácticos, por lo que todo lo aquí mostrado se orienta principalmente al profesorado para que pueda emplearlo y adaptarlo al desarrollo de sus clases.

Las propuestas de trabajo interdisciplinar entre asignaturas responden a la necesidad de que las estrategias de aprendizaje cooperativo se realicen de manera transversal. Es mucho más difícil su aplicación en una asignatura aislada, aunque sea viable, que su asimilación si se emplean en varias asignaturas que desarrollen una propuesta de trabajo y un temario con relaciones permanentes, que nuestros alumnos y alumnas puedan percibir con facilidad.

Finalmente, el desarrollo de este trabajo me ha permitido comprender mejor qué es y como se implementa el aprendizaje cooperativo, entendiendo que permite fomentar estructuras de escuela democrática dentro del sistema educativo. Su objetivo debe ser también superar estructuras de enseñanza tradicional que generen élites que completan su formación académica sin ninguna dificultad, mientras gran parte del alumnado queda excluido de una formación superior. Otro punto importante es desarrollar personas instruidas en un ambiente democrático y de cooperación social. También introduce una actitud solidaria en nuestras aulas y permite enseñar a los alumnos y alumnas estructuras asamblearias que aumentan la calidad democrática de la sociedad. No debemos olvidar que la educación es necesaria para crear una sociedad más libre e igualitaria y que permita realizar avances sociales a la nuevas generaciones.

Anexos

Apéndice A

Introducción a Python

***

A.1 Python y Vpython

Python es un lenguaje sencillo, orientado a objetos e interactivo y del que existe abundante documentación, tanto en su página web oficial (https://www.python.org) como en la siguiente wiki en español (https://wiki.python.org/moin/SpanishLanguage). Para aprender Python con más profundidad y como referencia principal de estas notas, recomendamos el libro de Newman [11].

Su sintaxis es especialmente legible y cuenta con numerosas librerías, módulos y clases que facilitan su uso a través de funciones ya implementadas por la comunidad.

Un programa de Python está constituido por una lista de instrucciones, mezcla de inglés y matemáticas, que constituye el código. Para trabajar con estas instrucciones en un ordenador es necesario un entorno de desarrollo. Hay muchos entornos de programación disponibles para Python, y el código puede editarse en cualquier editor de textos básico, pero su ejecución necesita de un entorno como IDLE, disponible para Linux, Windows y Mac. Una vez arrancado el IDLE, veremos la siguiente información en "*shell*" o línea de comandos.

```
Python 2.7.6 (default, Jun 22 2015, 17:58:13)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Detrás del símbolo >>> puede escribirse cualquier sintaxis propia de Python. No obstante, lo usual es escribir el código en un editor de textos (IDLE proporciona uno integrado) y guardar el código bajo la extensión **.py*. De la misma forma desde IDLE puede ejecutarse el código haciendo clic en el menú *Run* o pulsando la tecla *F5*. En cualquier línea de comandos de un sistema Linux, basta con ejecutar el comando *"python fichero.py"*.

A.1.1 Sintaxis básica

Los programas construidos se basan fundamentalmente en variables, que suelen ser números, vectores o matrices. Pueden tener la longitud que deseemos e incluir caracteres alfanuméricos. Además, Python distingue entre mayúsculas y minúsculas. Los tipos de variables existentes son los siguientes:

- Enteras: Números enteros de cualquier signo.
- Reales: Números reales como 1.0, 3.1415 o $-6{,}63 \cdot 10^{-34}$.
- Complejas: Números complejos como $1 + 5j$.
- Caracteres: Cadenas de caracteres como *hola* o 1234.

Al contrario que en otros lenguajes de programación, las variables no se declaran, sino que se crean la primera vez que se usan.

El siguiente paso es aprender a leer y escribir variables por pantalla, tal y como vemos en las siguientes figuras,

```
1 x=1
2 print(x)
3 x=2
4 print(x)
```

⇒

```
1
2
```

Figura A.1.1: Sintaxis y salida por pantalla

```
1 x=input(Introduzca el valor de x: )
```

Figura A.1.2: Lectura de la variable por pantalla

※ Aritmética Las operaciones básicas en Python se escriben de la siguiente manera[1]:

- Suma: x+y
- Resta: x-y
- Multiplicación: x*y
- División: x/y
- Potenciación (x^y): x**y

Además, dichas operaciones pueden combinarse como deseemos siempre y cuando estén correctamente escritas empleando paréntesis. De la misma forma, pueden asignarse variables de forma automática, es decir, el código $x = x+1$ (carente de sentido en matemáticas) incrementará la variable x en una unidad.

[1] Las versiones previas a Python 3.0 pueden presentar problemas con la división de enteros. Por ejemplo, la sintaxis $a = 3/4$ redondea al entero más cercano generando el valor $a = 0$, por lo que nos puede interesar trabajar con una variable real renombrando nuestros número como $b = 3./4.$ y obteniendo $b = 0{,}75$.

```
1 h=float(input("¿Altura?"))
2 t=float (input("¿Tiempo?"))
3 g=9.81
4 s=g*t**2/2
5 print(h-s)
```

Figura A.1.3: Ejemplo: Caída de un objeto desde una torre

※ Funciones, paquetes y módulos No obstante, un programa suele utilizar operaciones más complejas que las que hemos explicado hasta ahora. Para ello Python incorpora paquetes ya predefinidos con multitud de utilidades. Uno de ellos es el paquete *visual* sobre el que se fundamenta gran parte de este trabajo, pero en este apartado explicaremos las operaciones incluidas en el paquete *math,* que incluye las funciones matemáticas estándar (log, cos, exp, *etc.*) y números como π. Es conveniente llamar a la función al comienzo del programa, y podemos llamar únicamente a la operación que deseemos realizar o al paquete completo.

```
1 from math import log
2 #Y una vez inicializada
     podemos usarlo
3 x=log(2.5)
```

Figura A.1.4: Ejemplo: Llamada a una operación

```
1 from math import *
```

Figura A.1.5: Llamada al paquete completo

※ Comentarios Es especialmente recomendable el uso de comentarios en nuestros códigos para indicar que elementos o estructuras vienen a continuación. Para ello, Python no tiene en cuenta las líneas de código que comienzan con el símbolo #. Nada de lo que venga inmediatamente después afectará a la ejecución del programa.

Además, no tienen por qué comenzar al principio de la línea. No obstante, al escribir los comentarios debemos tener cuidado con escribir acentos en ellos, pues pueden dar problemas en algunos sistemas operativos.

```
1 #Pedimos la altura
2 h=float(input("¿Altura?"))
3 t=float (input("¿Tiempo?"))
4 g=9.81 #Valor de la
     gravedad
5 #Calculamos la posición
6 s=g*t**2/2
7 #Imprimimos la posición por
      pantalla
8 print(h-s)
```

Figura A.1.6: Ejemplo: Comentarios

※ Decisiones Los programas mostrados hasta ahora son meras estructuras secuenciales, pero podemos conseguir que se ejecuten sólo ciertas partes del código según ciertas condiciones que podremos imponer. Por ejemplo, podemos querer que un usuario teclee un número mayor que 0 por pantalla, empleando para ello el comando *if,*

```
1 x=int(input("Teclee un número
     entero mayor que cero: "))
2 if x<0:
3  print("Tecleó un número menor que
       cero.")
4  print("Se lo voy a corregir.")
5  x=1
6  print("El número es",x)
```

Figura A.1.7: Ejemplo: Condición *'if'*

El tipo de condiciones que podemos escribir son las siguientes:

- x==1 ($x = 1$)
- x<1 ($x < 1$)
- x>1 ($x > 1$)
- x<=1 ($x \leq 1$)
- x>=1 ($x \geq 1$)
- x!=1 ($x \neq 1$)

Pueden combinarse además varios criterios en una misma línea empleando las condiciones lógicas *and* y *or,*

```
1 x=int(input("Teclee un número entero
     mayor que cero y menor que cinco: "))
2 if x<0 or x>5:
3  print("Tecleó un número incorrecto.")
4  print("Se lo voy a corregir.")
5  x=1
6  print("El número es",x)
```

Figura A.1.8: Ejemplo: Condición *'if' combinada con una sentencia lógica or*

```
1 x=int(input("Teclee un número entero
     mayor que cero y menor que cinco: "))
2 if x>0 and x<5:
3  print("El número es correcto.")
4  print("El número es",x)
```

Figura A.1.9: Ejemplo: Condición *'if' combinada con una sentencia lógica and*

Existe una variante del comando *if*, que garantiza que repetimos las operaciones mientras no se cumpla la condición deseada. Es el comando *while*.

```
x=int(input("Teclee un número
    entero mayor que cero: "))
while x<0:
 print("El número es negativo.")
 x=int(input("Teclee otro número
     mayor que cero: "))
```

Figura A.1.10: Ejemplo de aplicación de *while*

※ Vectores y matrices En Física y Matemáticas es normal que una variable represente varios números a la vez, como ocurre con los vectores y las matrices. Dos de las herramientas que nos proporciona Python para trabajar con estas estructuras son las listas y matrices (*arrays*).

Listas Una lista es una serie de elementos, que normalmente son todos del mismo tipo, pero pueden ser mezcla de enteros, reales, complejos o caracteres. Las listas en Python se escriben como sigue: $r = [2, 3, 4, 3]$, obteniendo dichos valores si escribimos el comando $print(r)$. Cada elemento individual se denota por $r[0], r[1], ...$ siendo importante recordar que la primera posición siempre es la 0 y no la 1.

```
from math import sqrt
r=[1.0, 1.5, -2.2]
modulo=sqrt(r[0]**2 + r[1]**2 + r[2]**2)
print(modulo)
```

Figura A.1.11: Ejemplo: Cálculo del modulo de un vector

Una característica de Python que lo diferencia de otros lenguajes de programación es que podemos operar con listas enteras en bloque. Por ejemplo, el siguiente código suma los valores de una lista:

```
r=[1.0, 1.5, -2.2]
total = sum(r)
print(total)
```

Figura A.1.12: Suma de valores de una lista

De la misma forma son especialmente útiles las funciones máx, mín, *y len*, que proporcionan el máximo, el mínimo y el número de elementos de la lista. También es práctico aplicar de golpe una función a todos los elementos de una lista. En el caso particular de la función logaritmo neperiano, esto se consigue así:

```
1 from math import log
2 r=[1.0, 1.5, 2.2]
3 logr = list(map(log,r))
4 print(logr)
```

Figura A.1.13: Aplicación de funciones en una lista

donde *map(log,r)* va calculando los logaritmos de los elementos de r secuencialmente, y la instrucción *list* los convierte en la nueva lista *logr*.

Con frecuencia es necesario añadir elementos al final de una lista. Suponiendo que ya existe una lista r y que se necesita ampliarla con el valor 6,1, se escribiría *r.append(6.1)*. También puede asignarse el valor mediante una expresión matemática. Para eliminar el elemento final que acabamos de añadir escribiríamos *r.pop()*. *r.pop(n)* elimina el elemento que ocupa la posición n (recuérdese que el primero es el 0). Hay que tener en cuenta que esto implica desplazar un lugar hacia atrás todos los elementos más allá del borrado y que esta operación puede ser muy lenta.

Matrices Una matriz también es un conjunto de valores ordenados, pero presenta las siguientes diferencias con las listas:

1. El número de elementos es fijo. Una vez creada, no pueden añadirse ni eliminarse.
2. Todos los elementos son del mismo tipo.

Presenta a su vez las siguientes ventajas:

1. Pueden ser multidimensionales (las listas son unidimensionales)
2. Puede hacerse aritmética con ellas (sumar, multiplicar por números, etc.)
3. Trabajar con ellas es rápido.

El paquete *numpy* contiene muchas funciones para crear y manipular matrices. Por ejemplo, el siguiente código crea una matriz de ceros:

```
1 from numpy import zeros
2 a = zeros(4,float)
3 print(a)
```

Figura A.1.14: Creación de una matriz con
4 ceros

```
1 from numpy import zeros
2 a = zeros([3,4],float)
3 print(a)
```

Figura A.1.15: Creación de una matriz con ceros, en 3 filas y 4 columnas

La función *ones* crea una matriz de unos. Pero para no perder tiempo creando una matriz con números que van a cambiar inmediatamente lo mejor es definirla directamente vacía mediante *empty*, por ejemplo

```
1 from numpy import empty
2 a=empty(4,float)
```

Figura A.1.16: Definición de una matriz vacía

Otra forma de crear una matriz es a partir de una lista previa mediante la función array del paquete *numpy*:

```
1 r=[1.0, 1.5, -2.2]
2 a=array(r,float)
```

Figura A.1.17: Creación de una matriz

De forma parecida se crean las matrices bidimensionales con elementos dados:

```
1 a=array([[1,2,3],[4,5,6]],int)
2 print(a)
```

Figura A.1.18: Creación de una matriz

donde los elementos se han introducido como una lista de listas.
Para referirse a elementos particulares de la matriz se usa la notación $a[i, j]$. Así,

```
1 from numpy import zeros
2 a=zeros([2,2],int)
3 a[0,1]=1
4 a[1,0]=-1
5 print(a)
```

$\Rightarrow$

```
[[0 1]
[-1 0]]
```

Figura A.1.19: Elementos de una matriz

Se sigue el convenio de que el primer índice la matriz denota la fila y el segundo la columna.
También se puede trabajar con ellas como un todo. Por ejemplo, este programa multiplica por 2 cada elemento de la matriz a.

```
1 from numpy import array
2 a=array([1,2,3,4],int)
3 b=2*a
4 print(b)
```

Figura A.1.20: Manipulación de matrices

No obstante, para realizar sumas y restas de matrices éstas deben tener el mismo tamaño.

Como con la suma, si se multiplican dos matrices el resultado es otra matriz en la que cada elemento es el producto del elemento correspondiente de la primera matriz por el de la segunda. Si se pretende obtener el producto escalar, la sintaxis es

```
1 from numpy import array,dot
2 a=array([1,2,3,4],int)
3 b=array([2,4,6,8],int)
4 print(dot(a,b))
```

Figura A.1.21: Producto escalar

Con matrices bidimensionales todo funciona igual. Por ejemplo, el cálculo matricial

$$\begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}\begin{pmatrix} 4 & -2 \\ -3 & 1 \end{pmatrix}+2\begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix}=\begin{pmatrix} -3 & 5 \\ 0 & 2 \end{pmatrix}$$

en Python sería,

```
1 from numpy import array,dot
2 a=array([[1,3],[2,4]],int)
3 b=array([[4,-2],[-3,1]],int)
4 c=array([[1,2],[2,1]],int)
5 print(dot(a,b)+2*c)
```

Figura A.1.22: Cálculos con matrices

※ Bucles Para que una parte de nuestro programa se ejecute de forma repetida se emplea el comando *for,* similar al comando *while* que vimos anteriormente pero sin condición ninguna. Dentro del uso del bucle *for,* se emplea la función *range*, que crea una lista de la longitud indicada. Por ejemplo, *range*(3) crea la lista $[0, 1, 2]$.

```
1 for x in range(0, 4):
2   print "Número %d" %(x)
```

$\Rightarrow$

```
Número
0
Número
1
Número
2
Número
3
```

Figura A.1.23: Ejemplo de aplicación de *for* y salida por pantalla

```
for i in "PYTHON":
    print("Dame una ", i)

print("¡PYTHON!")
```

⇒

('Dame una ', 'P')
('Dame una ', 'Y')
('Dame una ', 'T')
('Dame una ', 'H')
('Dame una ', 'O')
('Dame una ', 'N')
¡PYTHON!

Figura A.1.24: Ejemplo de aplicación de *for* y salida por pantalla

También pueden usarse, junto con una condición *if*, las instrucciones *break* (termina el bucle y continúa por la siguiente línea inmediatamente siguiente) y *continue* (abandona el bucle y continúa por la línea inmediatamente siguiente).

※ Gráficos Hasta el momento sólo hemos mostrado letras y números, pero puede convenir representar los resultados como gráficos o animaciones. Para ello puede emplearse el subpaquete *pylab* del paquete *mathplotlib*. Los gráficos más simples se crean con la función *plot*, cuyo argumento son vectores de valores.

```
from pylab import plot,show

x=[0.5, 1.0, 2.0, 4.0, 7.0, 10.0]
y=[1.0, 2.4, 1.7, 0.3, 0.6, 1.8]
plot(x,y)
show()
```

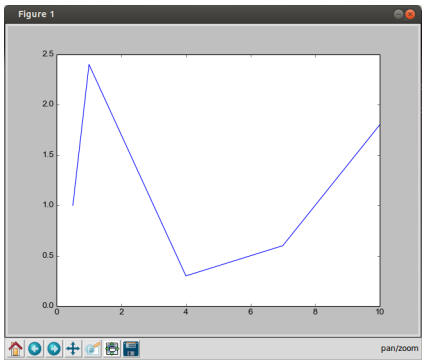


Figura A.1.25: Ejemplo de aplicación representación gráfica

¿Por qué no combinar las funciones *plot* y *show* en una sola que cree el gráfico y lo muestre en la pantalla?: porque cuando se desea pintar varias curvas en un mismo gráfico se usan tantas funciones *plot* como curvas haya y después se crea un gráfico de todas ellas con la función *show*.

Para hacer un gráfico de la función seno entre x=0 y x=10 primero se crea una matriz con los valores de las x y después se calcula el seno de éstos para obtener las coordenadas y de los puntos. Por ejemplo, así:

```
from pylab import plot,show
from numpy import linspace,sin
x = linspace(0,10,100)
y = sin(x)
plot(x,y)
show()
```

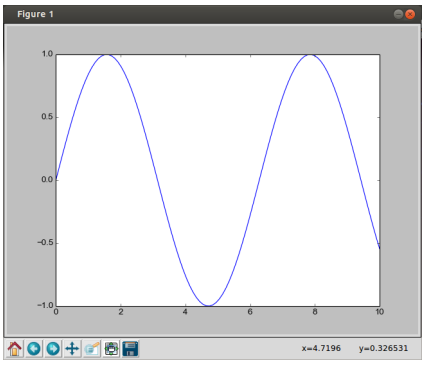


Figura A.1.26: Ejemplo de aplicación representación gráfica

Obsérvese que no se ha usado la función *sin* del *math*, sino la de *numpy*, que se aplica a matrices enteras. Naturalmente, también podría haberse usado la función *sin* de *math* y luego haber calculado cada seno individualmente con un bucle *for* o con *map(sin,x)*. Siempre hay varios caminos para conseguir un mismo resultado. El gráfico consta de cien segmentos rectilíneos que a ojo desnudo no se aprecian y que aparentan ser una curva, aunque en realidad no lo es.

A.1.2 Programación visual: VPython

Para la elaboración y tratamiento de animaciones e imágenes 3D emplearemos el módulo *visual* que permite la creación de pantallas y animaciones 3D navegables, empleando una sintaxis sencilla como la de Python. Consta de una serie de subrutinas que permiten construir y visualizar en tiempo real algunos objetos geométricos sencillos.

Al emplear VPython nos movemos en un espacio 3D donde $(0, 0, 0)$ es el centro de nuestro eje de coordenadas. El sentido positivo del eje x va hacia la derecha, el sentido positivo del eje y hacia arriba, y el sentido positivo del eje z es saliente con respecto a la pantalla. Las unidades que elijamos en el eje de coordenadas x,y,z son arbitrarias, pues la pantalla de visualización escalará automáticamente.

A diferencia que en Python, la salida de cualquier declaración (*print*) aparece en la pantalla visual, por lo que podremos seguir imprimiendo valores de las variables, listas, mensajes, ..., por lo que deberemos asegurarnos que su posición permite su correcta visualización.

En primer lugar deberemos cargar el módulo visual en nuestro código para poder utilizar todas las funciones presentes en el módulo, ejecutando la orden,

```
1 from visual import *
```

Figura A.1.27: Carga del módulo *visual*

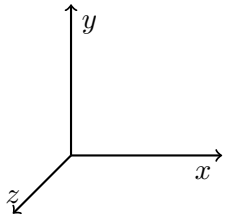


Figura A.1.28: Eje de coordenadas

※ Vectores No todos los objetos en VPython tienen por qué ser visibles. Por ejemplo, podemos crear vectores y hacer operaciones con ellos. Asumiendo por ejemplo un vector a, podemos referirnos a sus componentes en el código como $a.x$, $a.y$ o $a.z$. Además, podemos emplear todas las operaciones con vectores vistas previamente en un lenguaje Python convencional.

※ Visualizando la escena Pulsando el botón derecho del ratón sobre la ventana visual podremos girar la escena como deseemos y además, pulsando simultáneamente los botones derecho e izquierdo del ratón, podremos acercar y alejar la escena que estamos visualizando. Un primer programa de ejemplo es el siguiente:

```
1 from visual import *
2 scene.background=(1,1,1)
3 redcyl=cylinder(pos=vector
    (1,1,3), axis=(0,1,0),size
    =(8,4,6),radius=1,color=
    color.blue)
4 ball=sphere(pos=vector(1,4,3),
    radius=2,color=color.red)
```

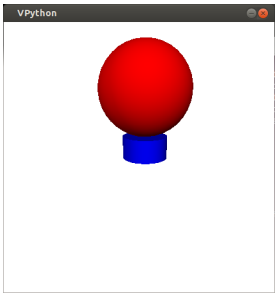


Figura A.1.29: Ejemplo de representación gráfica

※ Objetos, nombres y atributos Los objetos gráficos que creemos, como cajas, esferas o pirámides, siguen existiendo mientras dure el programa. Por ello, debemos asignar un nombre a cada objeto (como *redbox* o *sphere_moon*) si pretendemos moverlos o referirnos a ellos en distintas líneas de nuestro programa. Además, todos los objetos tienen distintos atributos: *ball.pos* indica la posición de la esfera, *ball.color* su color o *ball.radius* su radio. Cualquier modificación en los atributos de un objeto se vera modificada automáticamente en pantalla.

Además del conjunto de atributos, podremos crear nuevos atributos. Por ejemplo, si ya tenemos una esfera llamada luna; además de su radio y la ubicación, es posible darle atributos tales como la masa (*moon.mass*) o el momento lineal (*moon.momentum*).

A.1.2.1 Objetos principales

※ Pantalla de animación

scene.background Define el color de fondo de la pantalla de animación. Por ejemplo, *scene.background=color.white.*

scene.foreground Color por defecto de los objetos. No obstante, lo usual es definir el color de cada objeto.

scene.fullscreen Por defecto no visualizaremos nuestros programas en pantalla completa. Si queremos activarla debemos añadir *scene.fullscreen=1.*

scene.autoscale Por defecto, la pantalla escala con el tiempo para visualizar todos los objetos correctamente. Para desactivarlo, debemos añadir *scene.autoscale=0.*

scene.center Punto donde mira la cámara. Por defecto es *scene.center=(0,0,0).*

scene.range Sección que abarca la cámara. Su valor por defecto es *scene.range=(10,10,10).*

※ Colores VPython tiene los colores predefinidos: *color.red, color.green, color.blue, color.cyan, color.black, color.white, color.yellow, color.orange, color.magenta.* No obstante podemos indicar cualquier otro color que deseemos a partir de su código de colores RGB.

※ Objetos Cualquier programa de VPython se basa en objetos que se visualizan en pantalla. Existen ya objetos predefinidos en el módulo cargado que solo necesitan unas pocas instrucciones: *cylinder, arrow, cone, pyramid, sphere, ring, box, ellipsoid.* Y existen también otros objetos que nos permitirán hacer construcciones más complejas: *curve, convex, label, frame.*

cylinder Creará un cilindro en pantalla. Un ejemplo de aplicación es el siguiente:

cyl=cylinder(pos=(0,2,1), axis=(5,0,0),up=(01,0), radius=1,length=1,color=(1,0,0))

donde la opción *pos* sitúa el cilindro en nuestro espacio tridimensional, *axis* indica la orientación del cilindro, *up* indica que lado del cilindro se sitúa "arriba", *radius* indica el radio del cilindro, *length* su longitud y la opción *color* está indicada en código RGB, pero puede emplearse cualquier color predefinido en VPython. La primera parte del código (*cyl*) es solo una etiqueta para poder modificar posteriormente cualquiera de las opciones que hemos definido.

cone Crea un cono en pantalla.

cono=cone(pos=(5,2,0), axis=(12,0,0), radius=1)

sphere Crea una esfera en pantalla.

esfera=sphere(pos=(1,2,1), radius=0.5)

box Crea un ortoedro en pantalla, siendo *length* su longitud, *heigth* su altura y *width* su anchura.

box = box(pos=(x0,y0,z0), axis=(a,b,c), length=L, height=H, width=W, up=(q,r,s))

ellipsoid Crea un elipsoide en pantalla.

ellip = ellipsoid(pos=(x0,y0,z0), length=L, height=H, width=W)

arrow Crea una flecha, muy útil para representar vectores.

flecha= arrow(pos=(0,2,1), axis=(5,0,0), shaftwidth=1)

Pueden incluirse las opciones *shaftwidth, headwidth, headlength*, prefijadas por defecto y que cambian el aspecto de la flecha.

pyramid Crea una pirámide.

piram=pyramid(pos=(5,2,0), size=(12,6,4))

curve Une con segmentos rectos un conjunto de puntos, y da sensación de curva si la separación de puntos es pequeña. Por ejemplo, podemos construir el contorno de un cuadrado,

square = curve(pos=[(0,0),(0,1),(1,1),(1,0),(0,0)])

Puede añadirse la opción *radius* que fija el tamaño de la línea. Otro ejemplo de curva paramétrica es el siguiente:

t = arange(0, 10, 0.1)

*curve(x = sin(t), y = 1.0/(1+t), z = t**0.5, red = cos(t),*
green = 0, blue = 0.5(1-cos(t)))*

frame Puede ser necesario mover objetos compuestos. Para ello se utiliza la opción *frame*, que permite agrupar bloques de objetos. Por ejemplo,

f = frame()

cylinder(frame=f, pos=(0,0,0), radius=0.1, length=1, color=color.cyan)

sphere(frame=f, pos=(1,0,0), radius=0.2, color=color.red)

f.axis = (0,1,0) # Cambia la orientación de ambos

f.pos = (-1,0,0) # Cambia la posición de ambos

A.1.2.2 Animaciones

El objetivo principal de emplear VPython, junto con la posibilidad de representación de figuras 3D, es la actualización de la posición de los objetos para crear animaciones 3D. Para ello crearemos un pequeño programa con el que simular el movimiento de una bola dentro de una caja (VPython [24]).

※ Una bola en una caja En primer lugar debemos situar nuestros elementos en un espacio tridimensional. Para ello, generamos una esfera y una pared rectangular.

```
1 from visual import *
2 scene.background=(1,1,1)
3
4 bola = sphere(pos=(-5,0,0),
      radius=0.5, color=color.red)
5 pared = box(pos=(6,0,0), size
      =(0.2,12,12), color=color.
      blue)
```

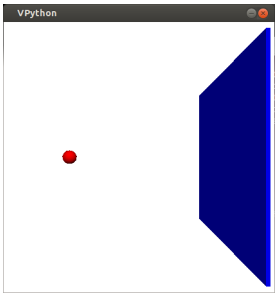


Figura A.1.30: Ejemplo de representación gráfica

※ Cambiando la posición de la bola Haremos que nuestra pelota se mueva por la pantalla y rebote contra la pared. Este proceso no es sino una sucesión de "fotografías" a la posición de la pelota en distintos tiempos que indican su movimiento por la pantalla. Para caracterizar su movimiento únicamente necesitaremos la velocidad a la que se mueve la bola y el tiempo transcurrido.

Para especificar la velocidad de la bola generaremos el atributo *bola.velocidad,* que debe ser un vector de tres dimensiones, pues nuestra bola se mueve en el espacio. Basta añadir la siguiente línea de código a nuestro programa:

```
1 bola.velocidad=vector(25,0,0)
```

Figura A.1.31: Vector velocidad

El siguiente paso es definir un paso temporal entre "fotografías". Llamaremos a este pequeño intervalo temporal *deltat* para representar $\triangle t$. En nuestro programa representa un tiempo virtual que servirá para la representación en pantalla. Añadiremos también las siguientes líneas de código:

```
1 deltat = 0.005
2 t = 0
```

Figura A.1.32: Paso temporal

Si ahora ejecutamos el programa, no observaremos nada, porque no hemos dado instrucciones de como actualizar la posición de la pelota conocida la velocidad. Asumiendo que es un movimiento lineal, la ecuación a utilizar es la siguiente:

$$\vec{r}_f = \vec{r}_i + \triangle t\vec{v}$$

No obstante, debemos traducir esta ecuación a lenguaje VPython:

```
1 bola.pos=bola.pos+bola.velocidad*deltat
```

Figura A.1.33: Actualización de la posición

Como la velocidad es un vector, también lo es el vector de posición, *bola.pos*, y simplemente estamos asignando una nueva posición añadiendo la distancia recorrida en el paso de tiempo definido. Nuestro programa principal tendrá ahora el siguiente aspecto:

```
1  from visual import *
2  scene.background=(1,1,1)
3
4  bola = sphere(pos=(-5,0,0), radius=0.5, color=color.red)
5  pared = box(pos=(6,0,0), size=(0.2,12,12), color=color.blue)
6
7  #Definimos la velocidad
8  bola.velocidad=vector(25,0,0)
9  deltat = 0.005
10 t = 0
11
12 #Actualizamos la posicion
13 bola.pos=bola.pos+bola.velocidad*deltat
```

Figura A.1.34: Movimiento de una pelota

※ Animación de la bola Si ejecutamos el programa desarrollado hasta ahora no observaremos diferencia ninguna, esto es porque la bola solo ha avanzado un paso temporal, y la diferencia de posición apenas es observable. Para ver una animación en nuestro código debemos añadir un ciclo *while,* que permita avanzar una serie de pasos temporales. Añadiremos también la opción *rate()*, que asegura el número máximo de veces que nuestro ordenador realizará el bucle *while* por segundo, para evitar una visualización excesivamente rápida.

```
1  from visual import *
2  scene.background=(1,1,1)
3
4  bola = sphere(pos=(-5,0,0), radius=0.5, color=color.red)
5  pared = box(pos=(6,0,0), size=(0.2,12,12), color=color.blue)
6
7  #Definimos la velocidad
8  bola.velocidad=vector(25,0,0)
9  deltat = 0.005
10 t = 0
11
12 #Actualizamos la posicion añadiendo un ciclo
13 while t<3:
14     rate(100)
15     bola.pos=bola.pos+bola.velocidad*deltat
16     t=t+deltat
```

Figura A.1.35: Animación de la pelota

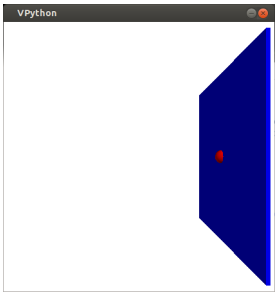


Figura A.1.36: Animación: La bola atraviesa la pared.

Si ejecutamos el programa podemos observar que la bola se mueve hacia la derecha, atravesando el muro y deteniéndose solo al alcanzar el tiempo impuesto, $t = 3$. VPython no lleva implementada física alguna: debemos imponer restricciones lógicas que eviten este tipo de efectos.

Por tanto, el siguiente paso es establecer estas condiciones lógicas que impidan que nuestra pelota atraviesa la pared. Para ello, basta agregar de nuevo las siguientes líneas de código, para hacer que nuestra pelota rebote en el eje x si alcanza la pared:

```
if bola.pos.x > pared.pos.x:
    bola.velocidad.x = -bola.velocidad.x
```

Figura A.1.37: Actualización de la posición

Con lo cual, nuestro nuevo código es:

```
from visual import *
scene.background=(1,1,1)

bola = sphere(pos=(-5,0,0), radius=0.5, color=color.red)
pared = box(pos=(6,0,0), size=(0.2,12,12), color=color.blue)

#Definimos la velocidad
bola.velocidad=vector(25,0,0)
deltat = 0.005
t = 0

#Actualizamos la posicion añadiendo un ciclo
while t<3:
    rate(100)
    if bola.pos.x > pared.pos.x:
        bola.velocidad.x = -bola.velocidad.x
    bola.pos=bola.pos+bola.velocidad*deltat
    t=t+deltat
```

Figura A.1.38: Animación de la pelota

No obstante, cuando la bola rebota sale disparada hacia la izquierda. Añadiremos una nueva pared en el lado izquierdo para mantener nuestra pelota entre ambas,

```
1  from visual import *
2  scene.background=(1,1,1)
3
4  bola = sphere(pos=(0,0,0), radius=0.5, color=color.red)
5  paredI = box(pos=(6,0,0), size=(0.2,12,12), color=color.blue)
6  paredD = box(pos=(-6,0,0), size=(0.2,12,12), color=color.blue)
7
8  #Definimos la velocidad
9  bola.velocidad=vector(25,0,0)
10 deltat = 0.005
11 t = 0
12
13 #Actualizamos la posicion anadiendo un ciclo
14 while t < 3:
15   rate(100)
16   if bola.pos.x > paredI.pos.x or bola.pos.x < paredD.pos.x :
17     bola.velocidad.x=-bola.velocidad.x
18   bola.pos=bola.pos+bola.velocidad*deltat
19   t=t+deltat
```

Figura A.1.39: Animación de la pelota con dos paredes

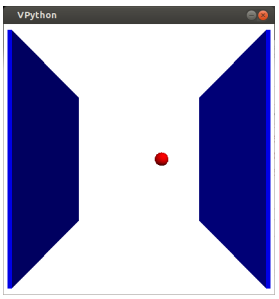


Figura A.1.40: Animación de la pelota con dos paredes

Apéndice B

Códigos

***

B.1 Cinemática: Tiro parabólico

```
1  from visual import *
2  from visual.graph import *
3  import math
4
5  #Inicializamos variables y pedimos componentes de la velocidad por
       pantalla
6  pos=vector(0,0,0)
7  vel=vector(10,10,0)
8  vel[0]=input("Componente x de la velocidad: ")
9  vel[1]=input("Componente y de la velocidad: ")
10 #vel[2]=input("Componente z de la velocidad: ")
11 grav=9.81
12 m=1
13 acel=vector(0,-grav,0)
14 t=0
15 hmax=0.
16 posold=0.
17
18 #Pantalla principal
19 scene.width=640
20 scene.height=480
21 #scene.fullscreen = True
22 scene.title='Tiro parabolico'
23 scene.autoscale=1
24 scene.background=(1,1,1)
25 userzoom = True
26 fps=50
27 dt=1./(5.*fps)
28
29 #Definimos las graficas para mostrar aceleracion, velocidad y
       posicion
30
31 graph1=gdisplay(x=700,y=0,width=300,height=200,
32 title='Aceleracion vs. t',xtitle='t',ytitle='A',
33 foreground=color.black,background=color.white)
34
35 graph2=gdisplay(x=700,y=250,width=300,height=200,
36 title='Velocidad x vs. t',xtitle='t',ytitle='V_x',
37 foreground=color.black,background=color.white)
38
39 graph3=gdisplay(x=700,y=500,width=300,height=200,
40 title='Velocidad y vs. t',xtitle='t',ytitle='Vy',
41 foreground=color.black,background=color.white)
42
```

```
43 graph4=gdisplay(x=1000,y=0,width=300,height=200,
44 title='X vs. t',xtitle='t',ytitle='X',
45 foreground=color.black,background=color.white)
46
47 graph5=gdisplay(x=1000,y=250,width=300,height=200,
48 title='Y vs. t',xtitle='t',ytitle='Y',
49 foreground=color.black,background=color.white)
50
51 datos=gdisplay(title='Datos',x=1000,y=500,width=300,height=200,
52 foreground=color.white,background=color.white,xmin=0,xmax=10, ymin
      =0,ymax=10)
53 out=label(opacity=0, color=color.black,display=datos.display, pos
      =(5,5), text="X: ")
54
55 aceleracion = gcurve(gdisplay=graph1,color=color.red)
56 velocix = gcurve(gdisplay=graph2,color=color.blue)
57 velociy = gcurve(gdisplay=graph3,color=color.blue)
58 xlugar = gcurve(gdisplay=graph4,color=color.green)
59 ylugar = gcurve(gdisplay=graph5,color=color.green)
60
61 #Definimos el entorno grafico de trabajo
62 proyectil=sphere(pos=pos,color=color.blue,radius=0.5)
63 suelo=box(pos=(0,-1,0),size=(25,0.1,25),color=color.yellow)
64 cannon=cylinder(pos=pos,axis=(1,1,0))
65 trayectoria=curve(color=color.black)
66 velocidadtotal=arrow(color=color.red,pos=proyectil.pos,axis=
67 vel/3.)
68 velocidadx=arrow(color=color.green,pos=proyectil.pos,axis=(vel
69 .x/3.,0,0))
70 velocidady=arrow(color=color.green,pos=proyectil.pos,axis=(0,
71 vel.y/3.,0))
72
73 #Bucle para resolver las ecuaciones diferenciales y pintar en las
      graficas
74 while pos.y>=0:
75   vel=vel+acel*dt
76   pos=pos+vel*dt
77   trayectoria.append(pos)
78   proyectil.pos=pos
79   velocidadtotal.pos=pos
80   velocidadx.pos=pos
81   velocidady.pos=pos
82   velocidadtotal.axis=vel/3.
83   velocidadx.axis=vector(vel.x/3.,0,0)
84   velocidady.axis=vector(0,vel.y/3.,0)
85   aceleracion.plot(pos=(t,acel.mag))
86   velocix.plot(pos=(t,vel.x))
87   velociy.plot(pos=(t,vel.y))
88   xlugar.plot(pos=(t,pos.x))
89   ylugar.plot(pos=(t,pos.y))
90   t=t+dt
```

```
91      if posold<pos.y:
92        hmax=pos.y
93        posold=pos.y
94      out.text="X: %.2f" %pos.x + "\nVx: %.2f" %vel.x + "\nVy: %.2f" %
            vel.y+ "\nhmax: %.2f" %hmax + '\nTiempo= %.2f' %t
95      rate(fps)
```

B.2 Energía: Planos inclinados

```
1  from visual import *
2  from visual.graph import *
3  import time
4
5  scene.width=640
6  scene.height=480
7
8  #Pedimos por pantalla los parametros necesarios, aunque fijamos el
       tamano del plano y los atributos fisicos
9  L=0
10 theta=0
11 veloc=-1
12 while(L<=0):
13  L=input('Indica la altura del plano: ')
14 while(theta<=0):
15   theta=input('Indica el angulo del plano (en grados): ')
16 theta=radians(theta) #Convertimos a radianes el angulo
17 while(veloc<0):
18   veloc=input('Indica la velocidad inicial de la bola: ')
19 fps=50
20 dt = 1./(50.*fps)  #Paso temporal
21 t=0.0
22 g=9.81
23 radio = .2
24
25 scene.center=(L/3, L/2, 0)  #Inicializo la camara
26 leyenda=label(opacity=0, box=0, color=color.red, pos=(0,L/1.5,0),
27 text='Tiempo=')
28 texto='Tiempo= 0.000'
29 leyenda.text=texto
30
31 #Definimos las graficas para mostrar ec, ep, velocidad y espacio
       recorrido
32 graph1=gdisplay(x=700,y=0,width=300,height=200,
33 title='Espacio vs. t',xtitle='t',ytitle='Espacio',
34 foreground=color.black,background=color.white)
35
36 graph2=gdisplay(x=700,y=250,width=300,height=200,
37 title='Velocidad vs. t',xtitle='t',ytitle='Velocidad',
38 foreground=color.black,background=color.white)
39
40 graph3=gdisplay(x=1000,y=0,width=300,height=200,
41 title='Energia vs. t',xtitle='t',ytitle='E',
42 foreground=color.black,background=color.white)
```

```
43
44 espaciograph= gcurve(gdisplay=graph1,color=color.red)
45 velocidadgraph= gcurve(gdisplay=graph2,color=color.blue)
46 Epgraph= gcurve(gdisplay=graph3,color=color.orange)
47 Ecgraph= gcurve(gdisplay=graph3,color=color.black)
48
49
50 #Plano inclinado
51 planinc = box(pos=((L*0.5*cos(theta)), (L*0.5*sin(theta)),0), axis=(
       L*cos(theta), L*sin(theta), 0), height=0.1,material=materials.
       wood)
52 vertic = box(pos=(L*cos(theta), (L*0.5*sin(theta)),0), axis=(0, L*
       sin(theta), 0), height=0.1,material=materials.wood)
53 horiz = box(pos=(L*0.5*cos(theta), 0,0), axis=(L*cos(theta), 0, 0),
       width=1, height=0.1,material=materials.wood)
54 suelo = box(pos=(0, 0,0), axis=(L*cos(theta), 0, 0), width=1, height
       =0.1,material=materials.wood)
55 #Bola
56 bola=sphere(pos=(planinc.length*cos(theta)-radio*sin(theta),planinc.
       length*sin(theta)+radio*cos(theta)+radio-planinc.height,0),radius
       =radio, material=materials.earth)
57 vel=vector(0,0,0)
58 vel.x=-veloc*cos(theta) #Hemos rotado las componentes de la
       velocidad
59 vel.y=-veloc*sin(theta)
60
61 #Guardamos la posicion inicial y Ajustamos tiempos para que sea
       realista
62 bola.x0=bola.x
63 bola.y0=bola.y
64 treal=(sqrt(2*g*L+veloc*veloc)-veloc)/(g*sin(theta))
65 while (True):
66   bola.visible=0
67   if bola.pos.y < radio+planinc.height:
68     break
69   #Actualizacion de la posicion
70   bola.x=bola.x+vel.x*t-0.25*g*sin(2*theta)*t*t #Posicion x
71   bola.y=bola.y+vel.y*t-0.5*g*sin(theta)*sin(theta)*t*t #Posicion
       y
72   t=t+dt
73   tvirt=t
74 #Recolocamos la bola y la dejamos caer
75 bola.x=bola.x0
76 bola.y=bola.y0
77 bola.visible=1
78 t=0
79 time.sleep(0.5)
80 while (True):
81   if bola.pos.y < radio+planinc.height:
82     break
83   rate(fps) #FPS
```

```
84      #Actualizacion de la posicion
85      bola.x=bola.x+vel.x*t-0.25*g*sin(2*theta)*t*t  #Posicion x
86      bola.y=bola.y+vel.y*t-0.5*g*sin(theta)*sin(theta)*t*t   #Posicion
            y
87      t=t+dt
88      ti=treal*t/tvirt
89      #Actualizacion de los parametros reales
90      e=veloc*ti+0.5*g*sin(theta)*ti*ti
91      velocidad=veloc+g*sin(theta)*ti
92      ep=g*L-g*e*sin(theta)
93      ec=0.5*velocidad*velocidad
94      #Pintamos la Ec y Ep en cada instante y mostramos el tiempo
95      texto='Tiempo= %.3f' %ti
96      leyenda.text=texto
97      espaciograph.plot(pos=(ti,e))
98      velocidadgraph.plot(pos=(ti,velocidad))
99      Epgraph.plot(pos=(ti,ep))
100     Ecgraph.plot(pos=(ti,ec))
```

B.3 Partículas de un gas

```
from visual import *
from time import clock
from visual.graph import *
from random import random

# Gas ideal de esferas duras
# Emplea arrays en Numeric Python para optimizar el algoritmo
# Modificado a partir del codigo (gas.py) en VPython
# de Lensyl Urbano - January, 2005 -
# anadiendo una serie de parametros mostrados por pantalla

l_slomo = 0
slomo = 4.0
colisionesp=0
colisionesw=0
colisionest=0

class hslider:
    "Slider horizontal"
    def __init__(self, pos, L, mini, maxi, init=None):
        self.bar = cylinder(pos=pos, axis = (L,0.0,0.0), color =
            (1,0,0), radius = L/80.0)
        self.but = sphere(pos=pos, color = (0,0,1), radius = L/40)
        self.min = mini
        self.max = maxi
        self.init = mini
        if init:
            self.but.pos.x = self.but.pos.x + ((init-mini) / (maxi-
                mini))*L

class T_change_event:
    "Slider horitzontal"
    def __init__(self, tstart, tend, finT):
        self.start = tstart      #event starttime
        self.end = tend          #event end time
        self.finT = finT    #angle of rotation in degrees
        self.type = "T change"
        self.init = 1

def atom_color(ke):
    #ke_max = 1e-23 * (1.0-((T_max - T)/T_max))
    rmax = 2.5e-24
    gmax = 7.5e-24
    bmax = 1e-23
```

```
43      if ke < rmax:
44          r = ke / rmax
45          b = 0.0
46          g = 0.0
47      elif ke < gmax:
48          r = max(0.0, (gmax-ke)/gmax)
49          g = 1.0 - r
50          b = 0.0
51      else:
52          r = 0.0
53          g = max(0.0, (bmax-ke)/bmax)
54          b = 1.0 - g
55      #print ke, ke/rmax, r, b
56      col = vector(r, g, b)
57      return col
58
59  win=500
60
61  Natoms = 0  # Numero atomos
62  num=0
63  while num==0:
64    caja=raw_input('Elija el volumen de la caja (grande, mediana,
        diminuta):\n')
65    if caja=='grande': L=2.; num=1
66    if caja=='mediana': L=1.; num=1
67    if caja=='diminuta': L=.5; num=1
68  while True and (Natoms<=0 or Natoms>=100):
69    try:
70       Natoms=int(raw_input('Introduzca el numero de atomos (<100):\n'
           ))
71    except ValueError:
72       continue
73  # Valores tipicos
74  #L = 1. # Lado del contenedor para las particulas
75  gray = (0.7,0.7,0.7) # Color de los ejes del contenedor
76  Raxes = 0.005 # Radio de las lines para los ejes
77  Matom = 4E-3/6E23 # Masa helio
78  Ratom = 0.03 # Radio exagerado del atomo de helio
79  k = 1.4E-23 # constante Boltzmann
80  T = 300. # Temperatura
81  dt = 1E-5
82
83  Ladj = L
84
85  ##scene = display(title="Gas", width=win, height=win, x=0, y=0,
86  ##                range=L, center=(L/2.,L/2.,L/2.))
87  scene = display(title="Gas", range=L*1.1, center=(L/2.,L/2.,L/2.))
88  origx = 0
89  origy = 0
90  w = 704+4+4
91  h = 576 #+24+4
```

```
92  low_win = 80
93  scene.width=w
94  scene.height=h
95  scene.x = origx
96  scene.y = origy
97  #scene.ambient = 0.5
98  #scene.lights[0] =  2* vector(0.17, 0.35, 0.70)
99  #scene.background = vector(1,1,1)
100
101 xaxis = curve(pos=[(0,0,0), (L,0,0)], color=gray, radius=Raxes)
102 yaxis = curve(pos=[(0,0,0), (0,L,0)], color=gray, radius=Raxes)
103 zaxis = curve(pos=[(0,0,0), (0,0,L)], color=gray, radius=Raxes)
104 xaxis2 = curve(pos=[(L,L,L), (0,L,L), (0,0,L), (L,0,L)], color=gray,
        radius=Raxes)
105 yaxis2 = curve(pos=[(L,L,L), (L,0,L), (L,0,0), (L,L,0)], color=gray,
        radius=Raxes)
106 zaxis2 = curve(pos=[(L,L,L), (L,L,0), (0,L,0), (0,L,L)], color=gray,
        radius=Raxes)
107
108 scene.background = color.white
109
110 Atoms = []
111 colors = [color.red, color.green, color.blue,
112           color.yellow, color.cyan, color.magenta]
113 poslist = []
114 plist = []
115 mlist = []
116 rlist = []
117
118 for i in range(Natoms):
119     Lmin = 1.1*Ratom
120     Lmax = L-Lmin
121     x = Lmin+(Lmax-Lmin)*random()
122     y = Lmin+(Lmax-Lmin)*random()
123     z = Lmin+(Lmax-Lmin)*random()
124     r = Ratom
125     Atoms = Atoms+[sphere(pos=(x,y,z), radius=r, color=colors[i %
            6])]
126     mass = Matom*r**3/Ratom**3
127     pavg = sqrt(2.*mass*1.5*k*T) # Energia cinetica media p**2/(2
            mass) = (3/2)kT
128     theta = pi*random()
129     phi = 2*pi*random()
130     px = pavg*sin(theta)*cos(phi)
131     py = pavg*sin(theta)*sin(phi)
132     pz = pavg*cos(theta)
133     poslist.append((x,y,z))
134     plist.append((px,py,pz))
135     mlist.append(mass)
136     rlist.append(r)
137
```

```
138 pos = array(poslist)
139 p = array(plist)
140 m = array(mlist)
141 m.shape = (Natoms,1) # Numeric Python: (1 by Natoms) vs. (Natoms by
        1)
142 radius = array(rlist)
143 t = 0.0
144 Nsteps = 0
145 pos = pos+(p/m)*(dt/2.) # initial half-step
146 time = clock()
147
148 #ldu
149 ptot = pavg * Natoms
150 T_mod = 1.0
151 T_mod_old = T_mod
152
153 #Ventana control
154 cwin = display(title="Control Temperatura", width=w, height=low_win,
155                x=0, y=h,
156                center = (L/2.0,0,0), range = L)
157
158 #Sliders control
159 Tbar = cylinder(pos=(0,0,0), color = (1,0,0), axis=(L,0,0), radius =
        L/80.0)
160 Tball = sphere(pos=(L, 0, 0), color = (0,0,1), radius = L/40)
161 T_init = T
162 dt_init = dt
163
164 #Display para datos
165
166 datos = display(title="Datos", width=300, height=300,
167                x=w, y=0, background=(1,1,1))
168 datos1=label(display=datos,opacity=0, color=color.black, text='
        Colisiones :')
169
170 ##Vbar = cylinder(pos=(0,L/6.0,0), color = (1,0,0), axis=(L,0,0),
        radius = L/80.0)
171 ##Vball = sphere(pos=(L, L/6.0,0), color = (0,0,1), radius = L/40)
172
173
174 events = []
175 ##events.append(T_change_event(5.0, 10.0, T / 2.0))
176 ##events.append(T_change_event(15.0, 20.0, 300.0))
177 ##
178 framerate = T_init
179 runtime = 0
180 pick = None
181
182 while 1:
183     datos1.text="Colisiones con las paredes: %i" %colisionesw + "\
          nColisiones entre atomos: %i" %colisionesp + "\nColisiones
```

```
            totales: %i" %colisionest + "\nTemperatura: %.1fK" %(300*
            T_mod) + "\nAtomos: %i" %Natoms + "\nVolumen: %.3f m3" %(L*L*
            L) + "\nPresion: %.3E J/K" %((Natoms*k*300*T_mod)/(L*L*L))
184         rate(60)
185         runtime += 1.0/framerate
186         if l_slomo == 1:
187             rate(slomo)
188
189         #Cambio temperatura
190
191
192         if cwin.mouse.events:
193             m1 = cwin.mouse.getevent() # obtain drag or drop event
194             if m1.drag and (m1.pick == Tball or m1.pick == vol.but):
195                 drag_pos = m1.pickpos
196                 pick = m1.pick
197                 cwin.cursor.visible = 0 # Hacer cursor invisible
198             elif m1.drop:
199                 pick = None # end dragging
200                 cwin.cursor.visible = 1 # cursor visible
201
202         if pick:
203             new_pos = cwin.mouse.project(normal=(0,0,1),point=(0,0,0))
204             if new_pos != drag_pos and (new_pos.x > 0 and new_pos.x < L)
                    :
205                 pick.pos.x += new_pos.x - drag_pos.x
206                 drag_pos = new_pos
207             elif (new_pos.x < 0.0):
208                 pick.pos.x = 0.0
209             elif (new_pos.x > L):
210                 pick.pos.x = L
211
212
213         #Actualizacion del tamano de la caja
214         xaxis.pos[1]=(Ladj,0,0)
215         yaxis.pos[1]=(0,Ladj,0)
216         zaxis.pos[1]=(0,0,Ladj)
217         xaxis2.pos=[(Ladj,Ladj,Ladj), (0,Ladj,Ladj), (0,0,Ladj), (Ladj
                ,0,Ladj)]
218         yaxis2.pos=[(Ladj,Ladj,Ladj), (Ladj,0,Ladj), (Ladj,0,0), (L,Ladj
                ,0)]
219         zaxis2.pos=[(Ladj,Ladj,Ladj), (Ladj,Ladj,0), (0,Ladj,0), (0,Ladj
                ,Ladj)]
220
221         #Actualizacion de temperatura y ec
222         T_mod = Tball.pos.x / L      #Nueva temperatura
223         T_mod = max(T_mod, 0.01)
224         T_modf = T_mod / T_mod_old
225         #print T_mod, pavg
226         if T_mod <> T_mod_old:
227             p = p * T_modf
```

```
228         T_mod_old = T_mod
229 
230         # Actualizamos posiciones
231         pos = pos+(p/m)*dt
232 
233         try:
234             r = pos-pos[:,newaxis] # Pares de vectores atomo-atomo
235             #print r
236         except:
237             r = pos-pos[:,newaxis] # Pares de vectores atomo-atomo
238         rmag = sqrt(add.reduce(r*r,-1)) # Distancias escalares atomo-
                atomo
239         try:
240             hit = less_equal(rmag,radius+radius[:,newaxis])-identity(
                    Natoms)
241         except:
242             hit = less_equal(rmag,radius+radius[:,NewAxis])-identity(
                    Natoms)
243         hitlist = sort(nonzero(hit.flat)).tolist() # i,j encoded as i*
                Natoms+j
244         #print hit
245 
246         # Colisiones entre atomos:
247         #print "hitlist", hitlist, hitlist[0]
248         for ij in hitlist[0]:
249             #print ij, Natoms
250             i, j = divmod(ij,Natoms) # decode atom pair
251             #print i,j
252             colisionesp+=1
253             colisionest+=1
254             #print colisionesp
255             hitlist[0].remove(j*Natoms+i) # Eliminamos pares i,j
                    simetricos de la lista
256             ptot = p[i]+p[j]
257             mi = m[i,0]
258             mj = m[j,0]
259             vi = p[i]/mi
260             vj = p[j]/mj
261             ri = Atoms[i].radius
262             rj = Atoms[j].radius
263             a = mag(vj-vi)**2
264             if a == 0: continue # Mismas velocidades
265             b = 2*dot(pos[i]-pos[j],vj-vi)
266             c = mag(pos[i]-pos[j])**2-(ri+rj)**2
267             d = b**2-4.*a*c
268             if d < 0: continue # something wrong; ignore this rare case
269             deltat = (-b+sqrt(d))/(2.*a) # t-deltat is when they made
                    contact
270             pos[i] = pos[i]-(p[i]/mi)*deltat # back up to contact
                    configuration
271             pos[j] = pos[j]-(p[j]/mj)*deltat
```

```
272             mtot = mi+mj
273             pcmi = p[i]-ptot*mi/mtot # transform momenta to cm frame
274             pcmj = p[j]-ptot*mj/mtot
275             rrel = norm(pos[j]-pos[i])
276             pcmi = pcmi-2*dot(pcmi,rrel)*rrel # bounce in cm frame
277             pcmj = pcmj-2*dot(pcmj,rrel)*rrel
278             p[i] = pcmi+ptot*mi/mtot # transform momenta back to lab
                    frame
279             p[j] = pcmj+ptot*mj/mtot
280             pos[i] = pos[i]+(p[i]/mi)*deltat # move forward deltat in
                    time
281             pos[j] = pos[j]+(p[j]/mj)*deltat
282
283         # Choques con las paredes
284         outside = less_equal(pos,Ratom) # walls closest to origin
285         if outside.any()==True:
286           colisionesw+=1
287           colisionest+=1
288         p1 = p*outside
289         p = p-p1+abs(p1) # force p component inward
290         outside = greater_equal(pos,L-Ratom) # walls farther from origin
291         if outside.any()==True:
292           colisionesw+=1
293           colisionest+=1
294         p1 = p*outside
295         p = p-p1-abs(p1) # force p component inward
296
297         #print max(p), min(p)
298         # Actualizacion de la posicion de los objetos
299         for i in range(Natoms):
300             Atoms[i].pos = pos[i]
301             #print p[i]
302             Atoms[i].color = atom_color(mag(p[i]))
303
304         Nsteps = Nsteps+1
305         t = t+dt
306
307         for e in events:
308             if runtime >= e.start and runtime <= e.end:
309                 if e.type == "T change":
310                     if e.init == 1:
311                         e.init = 0
312                         Tchange = (T_mod - e.finT) / (framerate * L)
313                         Tsch = (Tball.pos.x - (e.finT/T)) / (framerate *
                                (e.end - e.start))
314                         #print "Tball.pos.x, e.finT, Txch = ", Tball.pos
                            .x, e.finT, Tsch
315                     Tball.pos.x = Tball.pos.x - Tsch
316                     #T_mod = T_mod + Tchange
317                     #print "Tchange, T_mod", Tchange, Tball.pos.x
```

B.4 Movimiento planetario

```
1  '''
2  Planetario que muestra el movimiento de los planetas
3  Modified from original author: Robb (Jun 16, 2014)
4  '''
5
6  from visual import *
7
8  userzoom = True
9  width = 1024
10 height = 700
11
12 modo = 'Planetas internos'
13 #modo = 'Planetas clasicos'
14 #modo = 'Nueve Planetas'
15 num=0
16 #Inicializo el sistema de referencia para poder leer luego por
       pantalla
17 sistema_referencia='inicio'
18
19 while (num!=1)and(num!=2)and(num!=3):
20   num=raw_input('Elija una de las tres opciones para la cantidad de
         planetas: \n(1) Planetas internos (Mercurio, Venus, Tierra y
         Marte) \n(2) Planetas clasicos (Mercurio, Venus, Tierra, Marte,
          Jupiter y Saturno) \n(3) Nueve planetas\n')
21   num=int(num)
22   if num==1:
23     modo = 'Planetas internos'
24     while (sistema_referencia!='Sol')and(sistema_referencia!='
           Mercurio')and(sistema_referencia!='Venus')and(
           sistema_referencia!='Tierra')and(sistema_referencia!='Marte')
            :
25       sistema_referencia=raw_input("Elige el Sol o el planeta que
             situaremos en el centro (Sol, Mercurio, Venus, Tierra,
             Marte):")
26   if num==2:
27     modo = 'Planetas clasicos'
28     while (sistema_referencia!='Sol')and(sistema_referencia!='
           Mercurio')and(sistema_referencia!='Venus')and(
           sistema_referencia!='Tierra')and(sistema_referencia!='Marte')
           and(sistema_referencia!='Jupiter')and(sistema_referencia!='
           Saturno') :
29       sistema_referencia=raw_input("Elige el Sol o el planeta que
             situaremos en el centro:")
30   if num==3:
```

```
31      modo = 'Nueve Planetas'
32      while (sistema_referencia!='Sol')and(sistema_referencia!='
            Mercurio')and(sistema_referencia!='Venus')and(
            sistema_referencia!='Tierra')and(sistema_referencia!='Marte')
            and(sistema_referencia!='Jupiter')and(sistema_referencia!='
            Saturno')and(sistema_referencia!='Urano')and(
            sistema_referencia!='Neptuno')and(sistema_referencia!='Pluton
            ') :
33          sistema_referencia=raw_input("Elige el Sol o el planeta que
              situaremos en el centro:")
34    if num>3 or num<1:
35      print('Esa opcion no existe')
36
37  #Propieades de los planetas relativos a la Tierra
38  planet_propiedades = dict([('Mercurio', dict([('radio_orbital',
      0.39),('diametro', 0.38),('masa', 0.05 ),('densidad', 5.5),('
      n_Lunas', 0),('periodo_orbital', 0.24),('inclinacion', 7 ),('
      tilt', 0 ),('rotational_period', '59 days' ),('color',color.
      gray(0.7) )])), ('Venus', dict([('
      radio_orbital', 0.72),('diametro', 0.95),('masa', 0.82 ),('
      densidad', 5.3),('n_Lunas', 0),('periodo_orbital', 0.62),('
      inclinacion', 3.4 ),('tilt', 177),('rotational_period', '-243
      days' ),('color',color.magenta)])), ('
      Tierra', dict([('radio_orbital', 1 ),('diametro', 1 ),('
      masa', 1 ),('densidad', 5.5),('n_Lunas', 1),('
      periodo_orbital', 1 ),('inclinacion', 0 ),('tilt', 23 ),('
      rotational_period', '23.9 hours'),('color',color.blue )])),
                                ('Marte', dict([('radio_orbital',
      1.52),('diametro', 0.53),('masa', 0.11 ),('densidad', 3.9),('
      n_Lunas', 2),('periodo_orbital', 1.88),('inclinacion', 1.9 ),('
      tilt', 25 ),('rotational_period', '24.5 hours'),('color',color.
      red )])), ('Jupiter', dict([('
      radio_orbital', 5.2 ),('diametro', 11.21),('masa', 317.9 ),('
      densidad', 1.3),('n_Lunas', 67),('periodo_orbital', 11.9),('
      inclinacion', 1.3 ),('tilt', 3 ),('rotational_period', '10
      hours'),('color',(1,0.7,0.2))])), ('
      Saturno', dict([('radio_orbital', 9.54),('diametro', 9.45),('
      masa', 95.2 ),('densidad', 0.7),('n_Lunas', 62),('
      periodo_orbital', 29.5),('inclinacion', 2.5 ),('tilt', 27 ),('
      rotational_period', '11 hours'),('color',color.green )])),
                                ('Urano', dict([('radio_orbital',
      19.2),('diametro', 4.01),('masa', 14.5 ),('densidad', 1.3),('
      n_Lunas', 27),('periodo_orbital', 84 ),('inclinacion', 0.8 ),('
      tilt', 98 ),('rotational_period', '-17 hours'),('color',color.
      cyan )])), ('Neptuno', dict([('
      radio_orbital', 30.1),('diametro', 3.88),('masa', 17.1 ),('
      densidad', 1.6),('n_Lunas', 14),('periodo_orbital', 165 ),('
      inclinacion', 1.8 ),('tilt', 28 ),('rotational_period', '16
      hours'),('color',color.blue )])), ('
      Pluton', dict([('radio_orbital', 39.4),('diametro', 0.18),('
      masa', 0.002),('densidad', 2 ),('n_Lunas', 5),('
```

```
    periodo_orbital', 248 ),('inclinacion', 17.1),('tilt', 122),('
    rotational_period', '-6.4 days' ),('color',color.white )]))])
39
40
41  #Propiedades del sol
42  Sol_propiedades = dict([('name','Sol'),('radio_orbital', 0.),('
    diametro',0.01),('periodo_orbital', 0.)])
43
44  #Propiedades de la Luna
45  Luna_propiedades = dict([('name','Luna'),('radio_orbital', 0.00257)
    ,('diametro', 0.273),('masa', 0.012),('densidad', 3.34),('n_Lunas
    ', 0),('periodo_orbital',0.07479 ),('inclinacion',5.1),('tilt',
    6.6 ),('rotational_period','27.3 days' ),('color',color.gray(0.7)
    )])
46
47  factores_escala_orbitas = dict([('unit',5),('diametro',5),('masa',5)
    ,('densidad',5),('log-diametro',5)])
48
49  if modo == 'Planetas internos':
50      DT = 0.001
51      mostrar_planetas = ['Mercurio','Venus','Tierra','Marte']
52      SIZE_modo = 'log-diametro'
53  elif modo == 'Planetas clasicos':
54      DT = 0.005
55      mostrar_planetas = ['Mercurio','Venus','Tierra','Marte','Jupiter
        ','Saturno']
56      SIZE_modo = 'log-diametro'
57  elif modo == 'Nueve Planetas':
58      DT = 0.05
59      mostrar_planetas = planet_propiedades.keys()
60      SIZE_modo = 'log-diametro'
61
62  class ObjetoCeleste(object):
63      '''Un objeto celeste'''
64
65      def __init__(self, name, radio_orbital, periodo_orbital, color=
        color.yellow, diametro=1, masa=1, densidad=1, pos=vector
        (0,0,0), ang=0, **kwargs):
66          '''Constructor for ObjetoCeleste'''
67          self.name = name
68
69          #Parametros orbitales
70          self.radio_orbital = radio_orbital
71          self.periodo_orbital = periodo_orbital
72
73          #ObjetoCeleste tamano
74          self.size_options = dict([('unit',1),('diametro',diametro),(
            'masa',masa),('densidad',densidad),('log-diametro',log
            (10*diametro))])
75          self.color = color
76
```

```
77          #Posicion absoluta
78          self.pos = pos
79          self.ang = ang
80
81          #Cuerpos relativos
82          self.focus_body = False
83          self.orbitas = dict([])
84
85          self.set_sphere()
86
87      def set_sphere(self):
88          '''Creamos una esfera'''
89          #Objeto visual
90          self.sphere = sphere(color=self.color, radius=self.
               size_options[SIZE_modo]/2.)
91          self.trail = curve(pos=[self.sphere.pos,], color=self.color)
92
93
94
95      def del_sphere(self):
96          '''Hacemos la esfera invisible'''
97          self.sphere.visible = False
98
99
100     def update(self,t):
101         '''Actualizamos la posicion de los cuerpos'''
102         if self.focus_body:
103             self.ang = 2*pi*(t %self.periodo_orbital) / self.
                   periodo_orbital
104             offset = vector(self.radio_orbital*
                   factores_escala_orbitas[SIZE_modo]*cos(self.ang),
105                             self.radio_orbital*
                                factores_escala_orbitas[SIZE_modo]*
                                sin(self.ang),
106                             0)
107             self.pos = self.focus_body.pos + offset
108
109         self.sphere.pos = self.pos
110         self.trail.append(pos=self.sphere.pos, retain=1+int(self.
               periodo_orbital / DT))
111
112         for orbita in self.orbitas.values():
113             orbita.update(t)
114
115     def add_orbita(self, orbita):
116         '''Anadimos la orbita al planeta'''
117         if not self.orbitas.has_key(orbita.name) or orbita not in
               self.orbitas.values():
118             self.orbitas[orbita.name] = orbita
119             orbita.focus_body = self
120
```

```
121     def pop_orbita(self, orbita_name):
122         '''Quitamos la orbita'''
123         if self.orbitas.has_key(orbita_name):
124             orbita = self.orbitas.pop(orbita_name)
125             orbita.focus_body = None
126             return orbita
127         else:
128             return None
129 
130     def swap_reference_frame(self,orbita_name):
131         '''Switches the reference frame with the orbita'''
132         if self.orbitas.has_key(orbita_name):
133             orbita = self.pop_orbita(orbita_name)
134 
135             self.radio_orbital, orbita.radio_orbital = orbita.
                radio_orbital, self.radio_orbital
136             self.periodo_orbital, orbita.periodo_orbital = orbita.
                periodo_orbital, self.periodo_orbital
137 
138             self.del_sphere()
139             self.set_sphere()
140 
141             orbita.add_orbita(self)
142             return orbita
143         else:
144             return None
145 
146     def __str__(self):
147         '''String of the Celestial Body'''
148         orbitas_str = ',\n\t'.join([str(orbita) for orbita in self.
            orbitas.values()])
149         if len(orbitas_str) > 0:
150             orbitas_str = ', orbitas: \n\t' + orbitas_str
151         return self.name + \
152                 ': [pos: (' + ','.join([str(p) for p in self.pos]) +
                     ')' + \
153                 ', ang: ' + str(self.ang) + \
154                 orbitas_str + ']'
155 
156 def setup():
157     '''Construimos el sistema solar'''
158     global Sol, ref_body
159     display(title='Planetario -- ' + modo + ' -- ' + ' Centro en ' +
            sistema_referencia, width=width,height=height)
160     #Sol = ObjetoCeleste('Sol', 0, 0, diametro=2)
161     Sol = ObjetoCeleste(**Sol_propiedades)
162     for planet in mostrar_planetas:
163         Sol.add_orbita(ObjetoCeleste(planet,**planet_propiedades[
            planet]))
164 
165     Tierra = Sol.orbitas['Tierra']
```

```
166     #Tierra.add_orbita(ObjetoCeleste(**Luna_propiedades))
167
168     if sistema_referencia != 'Sol':
169         ref_body = Sol.swap_reference_frame(sistema_referencia)
170     else:
171         ref_body = Sol
172
173     ref_body.update(0)
174
175     print Sol
176     print ref_body
177     print Tierra
178
179 def run():
180     '''Run the solar system'''
181     t = 0.0
182     while True:
183         rate(100)
184         t += DT
185         ref_body.update(t)
186         #print Sol
187
188 if __name__ == '__main__':
189     setup()
190     run()
```

Bibliografía

***

[1] Aronson, Elliot y otros (1978). *The jigsaw classroom.* Sage.

[2] Chomsky, Noam (2001). *La (des) educación.* Crítica.

[3] Dewey, John (1995). *Democracia y educación: una introducción a la filosofía de la educación.* Ediciones Morata.

[4] Ferrer Guardia, Francisco (1976). *La escuela moderna.* Tusquets.

[5] Freire, Paulo (2005). *Pedagogía del oprimido.* Siglo XXI.

[6] García Corzo, Pablo (2008). «Proyecto Physthones». *Simulaciones Físicas en Visual Python (0.1). Un libro libre de Alqua.*

[7] Gavilán Bouzas, Paloma (2004). *Algebra en secundaria: trabajo cooperativo en matemáticas.* volumen 54. Narcea Ediciones.

[8] Johnson, David; Johnson, Roger; Holubec, Edythe y Vitale, Gloria (1999). *El aprendizaje cooperativo en el aula.* Paidós Barcelona.

[9] Lab!, Laboratorio de innovación educativa (2012). «Aprendizaje cooperativo: Propuesta para la implantación de una estructura de cooperación en el aula». [Consultada Agosto 2015].
URL: http://www.jrotero.org/index.php/jro/lab/.

[10] Maaβ, Katja y otros (2009). *Explora el mundo! Actividades interdisciplinares para el aprendizaje de las matemáticas y las ciencias.* Compass Project. DG Educación y Cultura.

[11] Newman, Mark (2013). *Computational Physics.* CreateSpace.

[12] Ortiz de Pinedo Montoya, Yolanda y Alonso Guinea, María Jesús (2005). «Del cuaderno de equipo al método de proyectos». *Cuadernos de pedagogía*, **(345)**, pp. 62–65.

[13] Pujolás Maset, Pere (2008). *9 Ideas clave. El aprendizaje cooperativo.* volumen 8. Graó.

[14] Pujolás Maset, Pere (2009). «Aprendizaje cooperativo y educación inclusiva: una forma práctica de aprender juntos alumnos diferentes». *VI jornadas de cooperación educativa con Iberoamérica sobre educación especial e inclusión educativa. Antigua.*

[15] Pujolás Maset, Pere y Doñate, María del Carmen (2010). *Aprender juntos alumnos diferentes: los equipos de aprendizaje cooperativo en el aula.* Octaedro.

[16] Pujolás Maset, Pere; Riera, Gemma; Pedragosa, Olga y Soldevilla, Jesús (2005). «Aprender juntos alumnos diferentes (I): El "quéz el çómo"del aprendizaje cooperativo en el aula». *Facultad de Educación. Laboratorio de Psicopedagogía (UVIC).*

[17] Robb (2014). «30 Days of Python: Day 10: An Orrery». [Consultada Agosto 2015].
URL: https://robotwhale.wordpress.com/2014/06/17/orrery/.

[18] Rojas, J.F.; Martínez, R. y Morales, M.A. (2014). «Mecánica 3d: python y el algoritmo de Verlet». *Education*, **60(2014)**, pp. 51–65.

[19] Rojas, J.F.; Morales, M.A.; Rangel, A. y Torres, I. (2009). «Fısica computacional: una propuesta educativa». *Revista mexicana de física E*, **55(1)**, pp. 97–111.

[20] Slavin, Robert (1990). *Cooperative learning: Theory, research, and practice*. volumen 14. Allyn and Bacon Boston.

[21] Torrego, Juan Carlos y Negro, Andrés (2014). *Aprendizaje cooperativo en las aulas*. Alianza Editorial.

[22] Urbano, Lensyl (2005). «Gas-KE Model — GeoMod». [Consultada Agosto 2015].
URL: http://earthsciweb.org/GeoMod/index.php?title=Gas-KE_Model.

[23] Villegas Martín, Francisco (1999). «Astronomía». [Consultada Agosto 2015].
URL: http://www.picasa.org/descargas/matematicas/astronomia/.

[24] VPython (2015). «3D Programming for Ordinary Mortals». [Consultada Agosto 2015].
URL: http://vpython.org.

[25] Wikipedia (2015). «Deferent and epicycle — Wikipedia, The Free Encyclopedia». [Consultada Agosto 2015].
URL: https://en.wikipedia.org/wiki/Deferent_and_epicycle.

[26] Wikipedia (2015). «Projectile motion — Wikipedia, The Free Encyclopedia». [Consultada Agosto 2015].
URL: https://en.wikipedia.org/wiki/Projectile_motion.

Material descargable

***

[R1] Todo el material contenido en este documento (pdf, códigos y fichas de trabajo) puede descargarse en la siguiente referencia:
Recursos educativos: "http://ic1.ugr.es/members/pvillegas/educational-resources/"