



U N I V E R S I D A D
COMPLUTENSE
M A D R I D

Proyecto Simulación Física para Videojuegos

Physics Engine

Víctor Emiliano Fernández Rubio

Carlos Llames Arribas

Alejandro Ortega Álvarez

Mario Peña Jiménez

Ángel Romero Pareja

Carlos Sánchez Bouza

Universidad Complutense de Madrid

Madrid, 2017-2018

Sistema de Partículas

1-Introducción: ¿Qué problema se plantea y cómo se intenta resolver?

“Un sistema de partículas es un modelo de sistema físico formado por partículas o cuerpos cuyas dimensiones y estado interno son irrelevantes para el problema bajo estudio generalmente irrelevantes en los problemas físicos hasta que se empiezan a estudiar gases en recipientes, la estructura atómica y el espacio.”

-Wikipedia

En el año 1800 William Herschel, Alemania, colocó un termómetro de mercurio en el espectro obtenido por un prisma de cristal con el fin de medir el calor emitido por cada color. En dicho experimento descubrió que el calor era más fuerte al lado del rojo del espectro donde no había luz. Esta fue la primera experiencia que mostró que el calor podía transmitirse por una forma invisible de luz. Herschel denominó a esta radiación "rayos calóricos", denominación bastante popular a lo largo del siglo XIX que, finalmente, fue dando paso al más moderno de radiación infrarroja.

Desde ese momento comienza la génesis de lo que en un futuro daría el concepto de partícula y su estudio, no siendo hasta el siglo XX cuando dicho concepto se hizo tangible con el descubrimiento del electrón. Actualmente el concepto de “sistema de partículas” nos ayuda a estudiar diversos campos de la ciencia desde la estructura atómica hasta el sistema solar, que podríamos denominar como un sistema de macropartículas.

En la década de los 90 los desarrolladores de videojuegos comenzaron a mostrar interés en los sistemas de partículas dada su utilidad en la posible aplicación a efectos visuales dentro de los videojuegos tridimensionales, como pudo ser Doom (1993), como salpicaduras de sangre, fuego, lluvia, explosiones o “magias”, entre otros. Dado que el desarrollo de videojuegos viene limitado por el desarrollo tecnológico la introducción de los sistemas de partículas planteó un problema binomial en el que el balance entre la calidad de los efectos y el coste de procesamiento, el cual a día de hoy sigue vigente impulsando el desarrollo e investigación de la simulación física en entornos gráficos.

2-Retrospectiva: Origen, evolución y tendencias

Los sistemas de partículas vienen usándose en disciplinas como la Animación, los Gráficos 3D y la Realidad Virtual. Permiten simular modelos deformables como por ejemplo el agua, fenómenos atmosféricos como un tornado o las nubes e incluso comportamientos sociales distribuidos entre “mini seres” con algo de IA de un

Videojuego.

El origen de los sistemas de partículas tanto en videojuegos como, en general, en el ámbito de la simulación y animación 3D y 2D es básicamente por necesidad. A la hora de animar cualquier película de dibujos, efectos especiales en películas, series, simulaciones físicas y en videojuegos, en muchas ocasiones vamos a necesitar partículas. Ejemplos de estas partículas pueden ser desde la espuma de una cascada, hasta el fuego generado por el motor de un cohete, y en un videojuego hay infinitud de sistemas de partículas para crear una gran cantidad de efectos. Como bien vemos en los ejemplos anteriores no solo queremos crear efectos especiales, sino que queremos representar fenómenos de la naturaleza de la forma más realista posible.

A la hora de enumerar las características que tiene un modelo de partículas y como han ido evolucionando, vamos a tratar a las partículas sin volumen en el espacio. Aunque no tengan volumen, si que vamos a considerar que las partículas de nuestro sistema si que van a tener otras características esenciales.

Si nosotros dejásemos una partícula suspendida en el aire ésta acabaría cayendo al suelo, con este pequeño ejemplo vamos a determinar una gran cantidad de estas características.

El hecho de que la partícula caiga al suelo da a entender que nuestras partículas tienen un peso, es decir que están influidas por fuerzas, sin embargo dejaremos las fuerzas para más adelante. Lo que nos da también a entender es que necesitamos que nuestras partículas dispongan de masa (gramos, kilogramos, picogramos, etc).

Las partículas se mueven en un espacio, como nuestras partícula que cae, ya sea en la vida real o en un motor de videojuegos, debido a esto necesitaran una posición XYZ, si es un entorno 3D o XY si nuestro sistema está simplificado a un entorno 2D. Al estar en un espacio nuestras partículas se pueden desplazar en el libremente, o como habíamos propuesto anteriormente si nosotros dejásemos una partícula en el aire está acabaría cayendo. Estas dos ideas anteriores, la de que tenga masa y que nuestra partícula está en un espacio implican, por necesidad, también la idea del movimiento, la cual se basa en la idea de velocidad y de aceleración sobre el espacio correspondiente ya sea un entorno gráfico 3D o la vida real, o un entorno 2D.

Sin embargo esta velocidad y esta aceleración a la que le estamos refiriendo no son más que resultados de un apartado que habíamos obviado anteriormente que es el de las fuerzas.

En nuestro pequeño caso, la de la gravedad, a pesar de que todos sabemos que esta fuerza depende de la constante de gravedad, g que para nuestro planeta Tierra tiene valor 9.81 m/s^2 . Es curioso cómo en algunos Videojuegos, a veces se varía la constante haciéndola valer 10 para conseguir exagerar el efecto un poco y, aunque no siendo entonces físicamente correcto, aparezca visualmente más creíble.

Anteriormente hablábamos que buscábamos los efectos más realistas posibles y para ello, sobretudo a la hora de películas y videojuegos a veces hay que hacer pequeñas modificaciones para sumergir más de lo normal al espectador.

Este hecho no nos valdría para gran cantidad de simulaciones físicas más precisas.

Un ejemplo es cualquier simulación física de relevancia real, cualquier variación de una constante o de alguna variable podría suponer una gran catástrofe, sin embargo a la hora de los videojuegos puede ser más que interesante modificar estos valores para conseguir una calidad más bonita o fantástica, en lugar de buscar la realidad. Generalizando un poco nuestro ejemplo de que la partícula cae, podríamos suponer que la partículas se mueve debido a que está sometida a una fuerza ejercida hacia abajo por lo tanto cualquier fuerza ejercida en cualquiera de las direcciones de nuestro plano supondrá un movimiento hacia la dirección sobre la que se ejerce. $Fuerza = Masa \text{ por } Aceleración$.

Tan solamente con esto podríamos tener un pequeño sistema de partículas donde nuestras partículas se desplazarán en un espacio, es decir tendrán masa, posición, velocidad, aceleración y además están sometidas a fuerzas. A parte de estas características, las partículas tienen además una vida, es decir nacen y mueren. A pesar que lo más evidente a lo largo de un tiempo (frame, en nuestro caso) es que las partículas se mueven también hay que decir que aparecen y desaparecen. Para explicar mejor el concepto de vida y muerte podemos imaginar un fuego artificial. Este sube y explota, seguidamente después de la explosión genera miles de lo que podríamos llamar en el ámbito de un motor de videojuegos o de simulación, una gran cantidad de partículas, todas esas estelas que desprenden, estas van cayendo hasta que desaparecen, es decir mueren.

Por último una de las cualidades que más han ido evolucionando estos años es el apartado gráfico de las partículas. Ante podíamos ver explosiones en videojuegos donde prácticamente podíamos distinguir los píxeles cuadrados de la propia explosión. Esto ha ido evolucionando de una forma impresionante hasta lograr unos aspectos hiperrealistas que superan a la imágenes que tenemos de la vida real, con colores más vivos de los que serían reales.

Si quisiéramos subdividir un poco la características del aspecto de una partícula podríamos sacar tres subdivisiones:

- El color, cualquier color que necesitemos ya sea el verde de la hierba, el blanco de una nube o los tonos azules, morado y verdes de una aurora boreal.
- El tamaño, usualmente pequeño para que no ocurra lo que hace tiempo y distinguir cuadrados dentro del fuego. Si queremos una chispa más grande lo que se hará será colocar un mayor número de partículas pero no aumentar el tamaño.
- Forma, generalmente redonda pero se puede adaptar a las necesidades del producto.

3-Teoría: Modelado físico, formulación y matemáticas asociadas

Dinámica de una Partícula

- La diferencia entre una partícula y un cuerpo rígido es que la partícula sólo puede tener movimiento de traslación, no de rotación.
- Las partículas se pueden emplear para generar fenómenos naturales, explosiones, agua, fuego, etc.

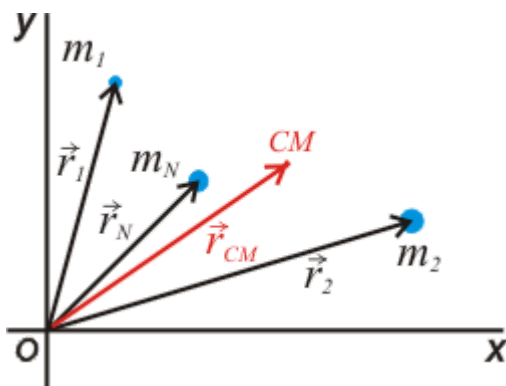
- Cada partícula ha de tener:
 - Dinámica (magnitudes): posición, velocidad, aceleración.
 - Aspecto: tamaño, forma, color.
 - Creación (aleatoriedad normalmente)
 - Actualización (cada frame)
 - Destrucción de la misma (por agentes externos o por tiempo de vida)
- Una partícula aparentemente no posee volumen, por lo que las magnitudes asociadas son la masa, la posición y la velocidad. (También la fuerza en el caso de ser aplicada).

Sistema de partículas

- En mecánica consideramos un sistema de partículas como un conjunto de N partículas que se mueven por separado, si bien interactúan entre sí y están sometidos a fuerzas externas.
- El número de partículas que forman un sistema puede ser muy variado e ir desde 2 (por ejemplo, al estudiar un átomo de hidrógeno), hasta cantidades gigantescas (por ejemplo, en 1 L de agua hay del orden de 10^{24} partículas).
- Cuando el número de partículas es reducido se puede abordar el problema dinámico analizando cada una por separado. Cuando es elevado, es preciso recurrir a promedios y descripciones colectivas (como la mecánica estadística, la elasticidad o la mecánica de fluidos).

Centro de masas

- El centro de masas de un sistema de partículas es un punto que, a muchos efectos, se mueve como si fuera una partícula de masa igual a la masa total del sistema sometida a la resultante de las fuerzas que actúan sobre el mismo.
- Se utiliza para describir el movimiento de traslación de un sistema de partículas.



Formulación

<u>Concepto</u>	<u>Fórmula</u>
Momento lineal o cantidad de movimiento	$P = m \cdot v$
Movimiento cinético o angular con respecto a un punto	$L = r \cdot mv = m \cdot (r \times v)$ (producto vectorial del vector posición (r) por el vector del momento lineal (mv), donde m es la masa)
Dinámica (2º ley Newton), Cálculo de fuerzas	$F = m \cdot a \Rightarrow \ddot{x} = \frac{f(x, \dot{x}, t)}{m}$
Fuerza de arrastre (rozamiento)	$f_{drag} = -k_{drag}v$
Fuerza de muelle (Kepler), para 2 partículas	$f_1 = - \left[k_s(\Delta x - r) + k_d \left(\frac{\Delta v \cdot \Delta x}{ \Delta x } \right) \right] \frac{\Delta x}{ \Delta x }$
Ciclo de resolución (actualización en cada frame)	$v(t + dt) = v(t) + \frac{f}{m} \cdot dt$ $x(t + dt) = x(t) + v(t) \cdot dt$
Ecuación Diferencial Ordinaria (EDO)	$\dot{x} = f(x, t)$
Método de Euler (integración numérica para EDO)	$x(t_0 + h) = x_0 + h\dot{x}(t_0)$ (h = paso de discretización del tiempo)
Método del punto medio (ecuación de Taylor)	$x(t_0 + h) = x(t_0) + h(f(x_0 + \frac{h}{2}f(x_0)))$
Energía cinética (para el sistema, la suma de las E_c individuales)	$K = K_1 + K_2 + \dots = \frac{1}{2}m_1v_1^2 + \frac{1}{2}m_2v_2^2 + \dots$
Velocidades después de un choque entre dos partículas (u_1 y u_2 son las velocidades antes del choque)	$v_1 = \frac{2m_2u_2 + (m_1 - m_2)u_1}{m_1 + m_2}$ $v_2 = \frac{2m_1u_1 + (m_2 - m_1)u_2}{m_1 + m_2}$ (se multiplica por k en el caso de ser choque elástico)

Sólido Rígido

Definición

Un cuerpo sólido real es deformable. Es decir, puede cambiar de forma o tamaño, por ejemplo debido a cambios de temperatura o tensiones aplicadas. En el estudio de la dinámica de un sólido el aspecto más importante es que esté compuesto por una cierta densidad de masa, siendo un aspecto secundario las pequeñas variaciones que pueda sufrir. Las variaciones de forma, por ejemplo, es un aspecto que trata la teoría de la elasticidad. Un sólido rígido por tanto es un sólido ideal no deformable. La característica o definición de un sólido no deformable sería la de un medio material en el que la distancia entre dos puntos cualesquiera permanece invariable. El primer aspecto de interés en el estudio del sólido rígido es el conocimiento del número de grados de libertad que presenta. El número de grados de libertad de una partícula es 3, el de dos partículas 6, el de N partículas $3N$. Un sólido, compuesto por N partículas (imaginemos por ejemplo que estas partículas son átomos o aglomerados de átomos) tendría $3N$ grados de libertad.

Todo sólido real está formado por un gran número de partículas materiales que ocupan una extensión finita en el espacio y poseen, en conjunto, una forma definida. Habrá puntos del espacio que estarán ocupados por alguna partícula material y habrá puntos en los que no habrá partícula alguna, bien porque se hallen en el exterior del sólido real, bien porque se encuentren en los intersticios entre las partículas materiales. Sin embargo, desde el punto de vista del análisis de las velocidades y aceleraciones de un sólido, no necesitamos considerar este tamaño finito, ni la forma de los sólidos. Podemos suponer un sólido ideal extendido a todo el espacio, cuya distribución de velocidades es la correspondiente al sólido real que estemos estudiando. Al aplicar este sólido ideal a un caso concreto, basta tener en cuenta que para aquellos puntos exteriores (o interiores) al sólido en los que no hay partículas de este, no tiene sentido asignarles una velocidad.

Un sólido ideal queda entonces identificado por un cierto sistema de referencia, y cada punto del espacio, sean cuales sean sus coordenadas, puede tratarse como parte del sólido.

Mecánica

La mecánica es la parte de la física que estudian el estado de reposo o movimiento de los cuerpos bajo la acción de las fuerzas que estudia el estado de reposo o movimiento de los cuerpos bajo la acción de las fuerzas. Aunque el estudio de la mecánica se remonta a los tiempos de Aristóteles (384-322 a.C.) y Arquímedes (287-212 a.C.) se tuvo que esperar hasta Newton (1647-1727) para encontrar la formulación satisfactoria de sus principios fundamentales. Estos principios fueron expresados posteriormente en una forma modificada por Alembert, Lagrange y Hamilton. Sin embargo la validez de los mismos permaneció inalterada hasta que Einstein formuló su teoría de la relatividad (1905). Aunque actualmente se han reconocido las limitaciones de estos principios, la mecánica newtoniana aun constituye la base de las ciencias ingenieriles de hoy en día. Los principios de la Mecánica como ciencia incorporan el rigor de la matemática, de la cual dependen en sumo grado. Así pues, la matemática representa un importante papel en la consecución del objetivo de la Mecánica Técnica, objetivo que no es sino la aplicación de aquellos principios como la aplicación de estos. El número de principios fundamentales de la Mecánica es

relativamente pequeño, pero su campo de aplicación rebasa todo límite y los métodos empleados por la Mecánica se extiende a un gran número de ramas de la Ingeniería. La mecánica se divide lógicamente en dos partes: La estática, que trata del equilibrio de los cuerpos bajo la acción de fuerzas, y la dinámica que trata del movimiento del cuerpo.

Se ha definido la Mecánica diciendo que es la rama de la Física que trata de la respuesta de los cuerpos a la acción de las fuerzas. Por conveniencia, se divide su estudio en tres partes, cuales son: Mecánica de los cuerpos rígidos, Mecánica de los cuerpos deformables y Mecánica de los fluidos. A su vez, la Mecánica de los cuerpos rígidos puede subdividirse en Estática (equilibrio del cuerpo rígido) y Dinámica (movimiento del cuerpo rígido). La Estática fue la primera parte de la Mecánica que se desarrolló porque los principios de la Estática se necesitaban para la construcción de edificios. Los constructores de las pirámides de Egipto comprendieron y utilizaron dispositivos tales como la palanca, la polea y el plano inclinado. La Estática abarca el estudio del equilibrio tanto del conjunto como de sus partes constituyentes, incluyendo las porciones elementales de material. Uno de los principales objetivos de la estática es la obtención de esfuerzos cortantes, fuerza normal, de torsión y momento flector a lo largo de una pieza, que puede ser desde una viga de un puente o los pilares de un rascacielos. Su importancia reside en que una vez trazados los diagramas y obtenidas sus ecuaciones, se puede decidir el material con el que se construirá, las dimensiones que deberá tener, límites para un uso seguro, etc., mediante un análisis de materiales. Por tanto, resulta de aplicación en ingeniería estructural, ingeniería mecánica, construcción, siempre que se quiera construir una estructura fija. Para el análisis de una estructura en movimiento es necesario considerar la aceleración de las partes y las fuerzas resultantes.

El estudio de la Estática suele ser el primero dentro del área de la ingeniería mecánica, debido a que los procedimientos que se realizan suelen usarse a lo largo de los demás cursos de ingeniería mecánica. La estática se utiliza en el análisis de las estructuras, por ejemplo, en arquitectura e ingeniería estructural. La resistencia de los materiales es un campo relacionado de la mecánica que depende en gran medida de la aplicación del equilibrio estático. Un concepto clave es el centro de gravedad de un cuerpo en reposo, que constituye un punto imaginario en el que reside toda la masa de un cuerpo. La posición del punto relativo a los fundamentos sobre los cuales se encuentra un cuerpo determina su estabilidad a los pequeños movimientos. Si el centro de gravedad se sitúa fuera de las bases y, a continuación, el cuerpo es inestable porque hay un par que actúa: cualquier pequeña perturbación hará caer al cuerpo. Si el centro de gravedad cae dentro de las bases, el cuerpo es estable, ya que no actúa sobre el par neto del cuerpo. Si el centro de gravedad coincide con los fundamentos, entonces el cuerpo se dice que es metaestable. Para poder saber la fuerza que está soportando cada parte de la estructura se utilizan dos medios de cálculo:

- La comprobación por nudos.
- La comprobación por secciones.

Para lograr obtener cualquiera de estas dos comprobaciones se debe tomar en cuenta la sumatoria de fuerzas externas en la estructura (fuerzas en x y en y), para luego comenzar con la comprobación por nudos o por sección.

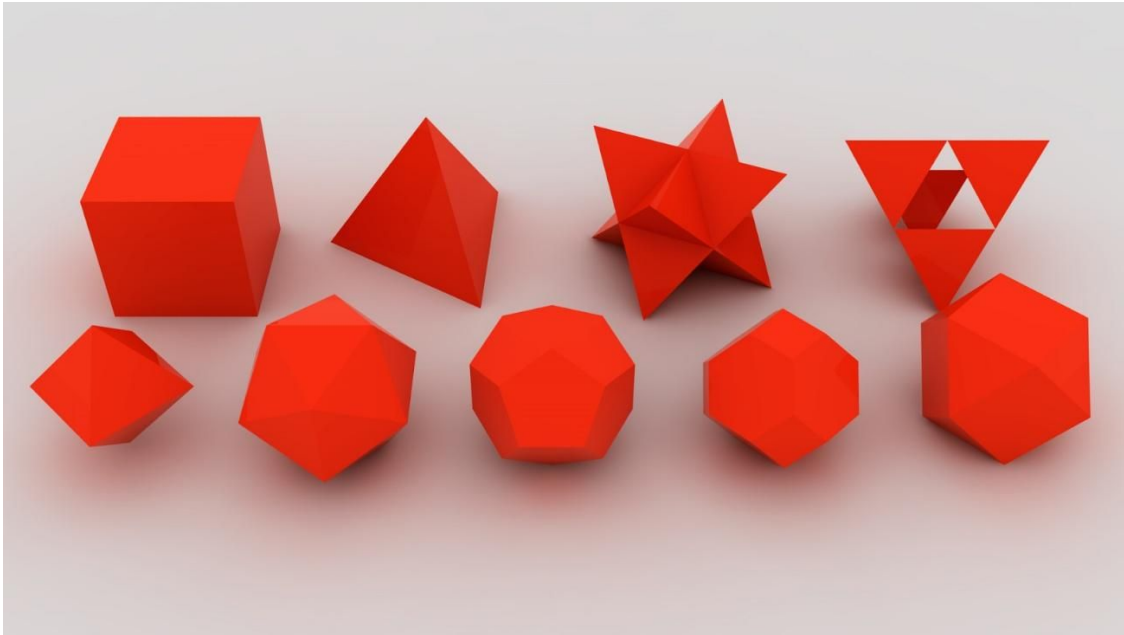
Dinámica

La Dinámica se desarrolló mucho después porque las magnitudes que en ella intervienen (velocidad y aceleración) dependen de la medida precisa del tiempo. Los experimentos de Galileo Galilei (1564-1642) de caída de cuerpos, péndulos y cilindros rodando por un plano inclinado dieron inicio al desarrollo de la Dinámica. En física existen dos tipos importantes de sistemas físicos los sistemas finitos de partículas y los campos. La evolución en el tiempo de los primeros pueden ser descritos por un conjunto finito de ecuaciones diferenciales ordinarias, razón por la cual se dice que tienen un número finito de grados de libertad. En cambio la evolución en el tiempo de los campos requiere un conjunto de ecuaciones complejas. En derivadas parciales, y en cierto sentido informal se comportan como un sistema de partículas con un número infinito de grados de libertad. La mayoría de sistemas mecánicos son del primer tipo, aunque también existen sistemas de tipo mecánico que son descritos de modo más sencillo como campos, como sucede con los fluidos o los sólidos deformables. También sucede que algunos sistemas mecánicos formados idealmente por un número infinito de puntos materiales, como los sólidos rígidos pueden ser descritos mediante un número finito de grados de libertad.

La dinámica de un sólido rígido es aquella que estudia el movimiento y equilibrio de sólidos materiales ignorando sus deformaciones. Se trata, por tanto, de un modelo matemático útil para estudiar una parte de la mecánica de sólidos, ya que todos los sólidos reales son deformables. Se entiende por sólido rígido un conjunto de puntos del espacio que se mueven de tal manera que no se alteran las distancias entre ellos, sea cual sea la fuerza actuante (matemáticamente, el movimiento de un sólido rígido viene dado por un grupo un paramétrico de isometrías).

2-Retrospectiva: Origen, evolución y tendencias

Los sistemas sólidos rígidos son una parte fundamental en el mundo de los videojuegos, es gran parte del core, podríamos decir de un juegos. Tanto en animaciones, simulaciones, y cualquier entorno grafico los sólidos y rígidos son una de las partes principales que vemos en la escena.



Origen:

Su origen se basa a la hora de representar todos los objetos que vemos en una escena.

Es uno que no modifica su tamaño o forma, cualquiera sea la fuerza o torque a la que esté sometido. Esta es su característica distintiva. Comenzaremos por estudiar la cinemática del sólido rígido. Para poder definir el sistema sólido rígido vamos a evaluar cada una de sus **características** además de su **cinemática** y su **dinámica**.

Características, evoluciones:

Comenzando con las características de un sólido, podemos hacer referencia a la masa, volumen y densidad todas ellas relacionadas entre sí muy estrechamente. Los sólidos van a poseer una **masa** (gramos, kilogramos, picogramos, etc.), la cual jugará un papel muy importante tanto en la dinámica y en la cinemática. También tienen un **volumen** y por tanto ocupan un espacio en la escena, las dimensiones y formas de este volumen pueden ser infinitas.

Claro está que si tenemos una masa y un volumen tendremos una **densidad**, concepto que relaciona ambas características anteriores.

Como hemos mencionado antes ocupan un volumen en la escena, es decir, tienen una **posición** XYZ, si es un entorno 3D o XY si nuestro sistema está simplificado a un entorno 2D y una **orientación** dirección a la que “mira” el cuerpo.

Podríamos enumerar una gran cantidad de características que poseen los sólidos como por ejemplo, la elasticidad, la dureza, etc. Sin embargo a la hora de ver su cinemática o su dinámica son menos relevantes.

Para comenzar describiendo la cinemática de un sólido rígido, debemos comenzar mencionando su condición de rigidez: "Las velocidades de los puntos alineados pertenecientes al sólido rígido dan la misma proyección sobre la recta que los une." Esta condición supone que dos puntos de un sólido siempre van a mantener la misma distancia, por tanto la definición de antes de un sólido, cuerpo quien no modifica su forma o tamaño se cumple. Los movimientos más característico de un sólido es la **traslación** y la **rotación** y claramente una combinación de ambos.

El movimiento de traslación es el más sencillo. Posee una trayectoria, esta puede ser rectilínea, si es una recta o curvilínea, si el movimiento describe una curva. Una dirección y un sentido. La dirección es la recta que muestra el movimiento del cuerpo, y el sentido que indica hacia donde se produce el movimiento.

Posee además una velocidad y aceleración. La **velocidad** es un vector con una dirección un sentido y un módulo, dependiendo del sistema puede ser (X, Y, Z) o (X, Y) .

Estas mismas condiciones se aplican al vector **aceleración**. Estos modificarán la posición del sólido a lo largo de un tiempo.

El movimiento de rotación Se dice que un sólido rígido está animado de un movimiento de rotación alrededor de un eje fijo cuando todos sus puntos describen trayectorias circulares centradas sobre dicho eje y contenidas en planos normales a éste.

El eje de rotación puede atravesar el cuerpo o ser exterior al mismo; en el primer caso, los puntos del sólido que están sobre el eje permanece en reposo en tanto que los demás puntos describen circunferencias en torno al eje; en el segundo caso, todos los puntos del sólido están en movimiento circular alrededor del eje exterior al sólido. En cualquier caso, la velocidad de un punto del sólido será tangente a la circunferencia.

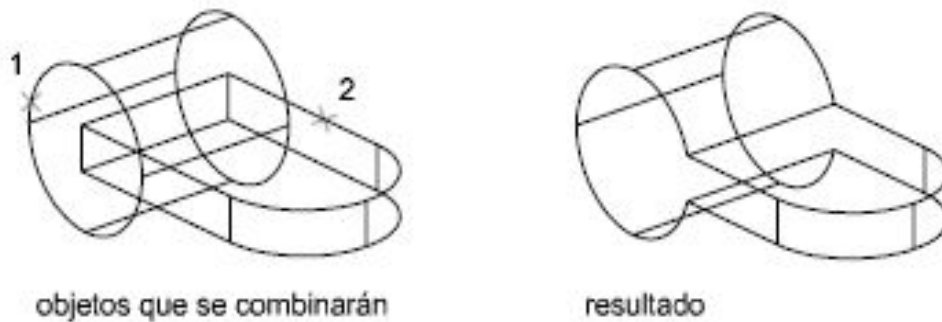
La combinación de las dos anteriores la podríamos denominar como el movimiento general que poseen, al fin y al cabo la mayoría de movimientos implican a ambos a la vez aunque uno de ellos sea muchísimo menor en comparación con el otro.

De lo dicho anteriormente el conjunto de "posiciones" generales de un sólido rígido en cada instante el posicionamiento general queda especificado por un vector de traslación que da el movimiento de uno de sus puntos respecto a la posición inicial, y una matriz de rotación que define la orientación del sólido rígido en cada momento, especificando como ha rotado respecto a su orientación original.

Como hemos mencionado antes la velocidad es un vector con un **módulo** una **dirección** y un **sentido**, en el caso de un movimiento rectilíneo esto se ve bien. Sin embargo en un movimiento rotacional vamos a explicar un poco más este vector. En el caso del módulo es similar, en cambio en la dirección viene definida como la recta tangente a la curva descrita por el cuerpo y el sentido viene dado por la regla de la mano derecha o del sacacorchos.

Un sólido no tiene por qué estar compuesto de un solo objeto, podemos tener sólidos compuestos por varios sólidos a su vez, en cierto modo esto es lo más común realmente.

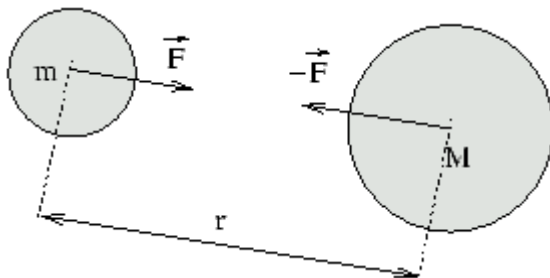
En muchas ocasiones nos vendrá mejor calcular la masa del conjunto y también su **centro de masas**, otra característica de los sólidos que están compuestos por sólidos distintos.



Para comenzar con la dinámica de los sólidos rígidos vamos a empezar por lo básico definiendo las **fuerzas**. La fuerza es una magnitud vectorial que mide la razón de cambio de momento lineal entre dos cuerpos, según la definición clásica, fuerza es todo agente capaz de modificar la cantidad de movimiento o la forma de los materiales. Depende de la masa y la aceleración descritas anteriormente.

Una fuerza muy presente en todos los sólidos rígidos es la de la **gravedad**, fuerza que también afectaba a nuestras partículas.

Sin embargo las partículas rara vez tomaban valor en sus masas muy grandes, al contrario, los sólidos pueden llegar a ser enormes y además ser el conjunto de muchos de estos. En este caso nos movemos más a sistemas espaciales donde están presentes las fuerzas gravitacionales, **fuerza de gravitación universal**.



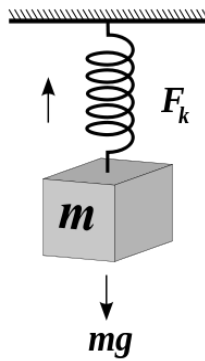
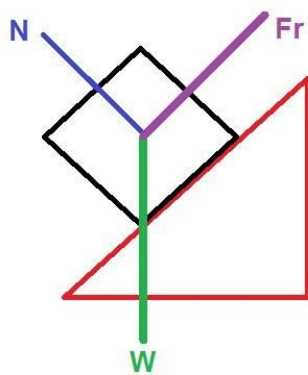
En nuestro sistema de sólido rígido tendremos una gran cantidad de fuerzas, las ejercidas sobre el cuerpo y las que resultan de ejercer estas al cuerpo, como pueden ser las de **rozamiento**.

Estas fuerzas dependen de otras variables como el **coeficiente de rozamiento** o si el plano en el que nos desplazamos está inclinado un número alfa de grados.

El coeficiente de rozamiento depende de los materiales implicados en el sistema. No tiene el mismo coeficiente de rozamiento la madera con el acero que por ejemplo el teflón con el hielo.

También entre en juego otra fuerza muy presente la **normal** que es perpendicular al plano en el que se desplaza el objeto. En muchas ocasiones suele ser igual al peso (como ya sabemos $m \cdot g$). Sin embargo como hemos mencionado antes si el plano está inclinado a la hora de ver las fuerzas del rozamiento esa inclinación, alfa puede jugar un importante papel.

Otra fuerza muy presente en este capítulo de sólidos y rígidos es la que ocurre en los comportamientos de los muelles. Es una fuerza elástica definida por la **ley de Hook**.



Momento angular

Como hemos definido antes la rotación es un movimiento de un cuerpo con respecto a un eje de giro interior o externo al mismo. Normalmente, en la rotación de los cuerpos actúan diversos tipos de fuerzas de arrastre, centrales, de rozamiento.

En el estudio de la rotación se maneja como magnitud fundamental el **momento angular**, y con respecto a un origen de referencia, el momento angular total se define como la suma de los momentos angulares de cada partícula para dicho punto, en el que intervienen el vector de posición respecto al eje de giro su masa y su velocidad.

La variación del momento angular con respecto al tiempo se conoce por momento total de las fuerzas del sistema. Como la velocidad angular de giro de un sólido rígido es idéntica para todas sus partículas constituyentes, el momento angular del sólido vendrá dado por la velocidad angular y el vector de posición de la partícula i con respecto al eje de giro y la masa.

Momento de inercia

El momento de inercia o momento angular longitudinal no depende de las fuerzas que intervienen en un sistema físico, sino tan sólo de la geometría del cuerpo y de la posición del eje de giro.

Evolución:

La evolución respecto a este campo en el mundo de los videojuegos ha sido enorme. No tanto respecto a los cálculos que también han mejorado gracias a las nuevas tecnologías y algoritmos más rápidos, a la hora de calcular. Sino quizás sobre todo por la forma.

Antes los videojuegos pongamos de coches, el propio coche en si era un rectángulo. Sin embargo hoy en día un coche de un juego de este estilo es un sólido complejo. Cada pieza es tratada como un sólido por tanto cálculos que necesiten como por ejemplo el centro de masas de todos los sólidos que lo componen pueden ser complejos, algo que anteriormente no se podía ni imaginar.

No solo en ejemplo de coches, sino pongamos un arma de un juego de guerra. Antes tú disparabas y la bala seguía una dirección recta. Posteriormente se le añadió una caída a la bala conforme se alejaba, simulando las fuerzas de gravedad. Actualmente sobre esa bala infieren gran cantidad de cálculos desde la fuerza de gravedad hasta el

propio viento que tenga la escena, o incluso la posible variación de trayectoria por atravesar una pared.



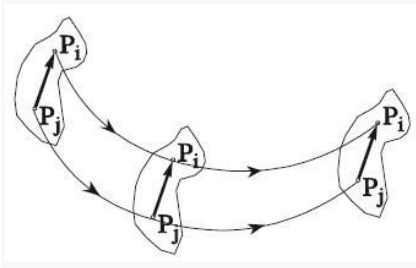
3-Teoría: Modelado físico, formulación y matemáticas asociadas

Se entiende por **sólido rígido** un conjunto de puntos del espacio que se mueven de tal manera que no se alteran las distancias entre ellos, sea cual sea la fuerza actuante.

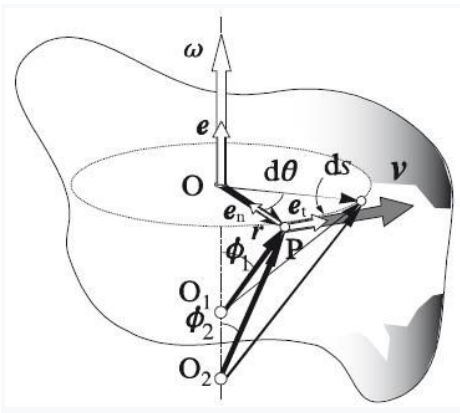
Cinemática del sólido rígido

La **cinemática del sólido rígido** es una aplicación de la cinemática al movimiento de un objeto tridimensional rígido en el espacio. El movimiento más general del sólido rígido puede considerarse como la superposición de dos tipos de movimiento básicos: de traslación y de rotación.

En el movimiento de traslación todos los puntos del sólido tienen la misma velocidad.



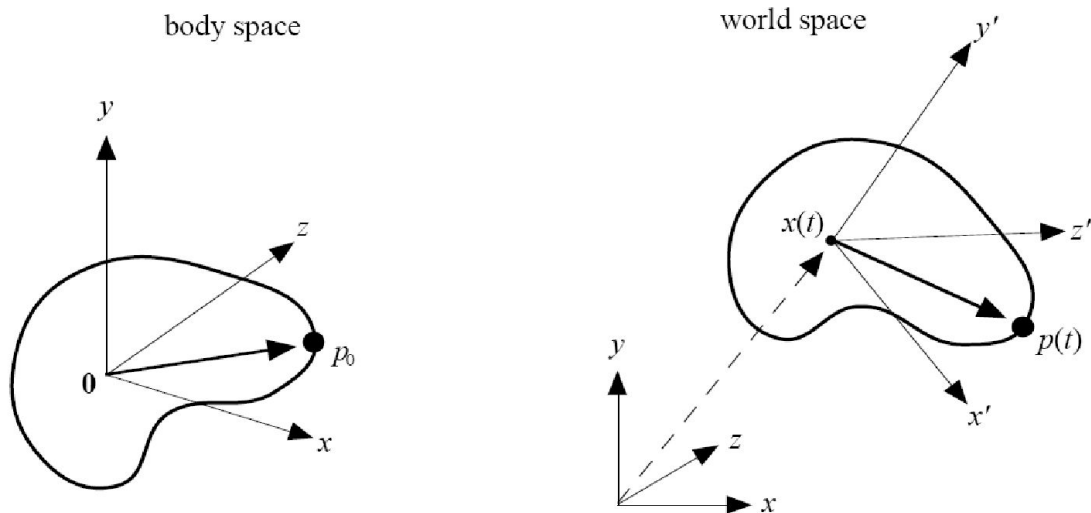
Movimiento de rotación. El vector velocidad angular es único (invariante), pero cada punto del sólido tiene una velocidad diferente de la de los otros.



Matriz de rotación: Las columnas de $\mathbf{R}(t)$ son los vectores del sistema ejes cuerpos representados en el sistema mundo.

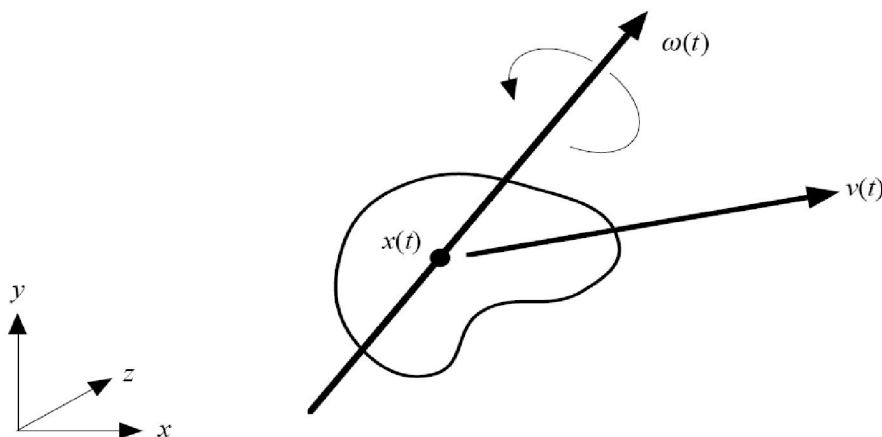
$$R(t) = \begin{pmatrix} r_{xx} & r_{yx} & r_{zx} \\ r_{xy} & r_{yy} & r_{zy} \\ r_{xz} & r_{yz} & r_{zz} \end{pmatrix}$$

$$p(t) = R(t)p_0 + x(t)$$



- **Velocidades lineal y angular**

Velocidad lineal v y velocidad angular ω :



La velocidad de un punto del cuerpo debido a la rotación es:

$$\dot{\mathbf{r}}_b(t) = \omega(t) \times \mathbf{r}_b$$

Para sacar $\dot{\mathbf{R}}$:

$$\dot{\mathbf{R}}(t) = \omega(t) * \mathbf{R}(t) = [\omega(t) \times \mathbf{R}_1(t), \omega(t) \times \mathbf{R}_2(t), \omega(t) \times \mathbf{R}_3(t)]$$

- **Masa total del cuerpo**

$$M = \sum_{i=1}^N m_i$$

- **Centro de masa (o de gravedad):** punto en el que no se genera rotación

$$\frac{\sum m_i r_i(t)}{M}$$

- **Tensor de inercia**

El tensor de inercia cte. para ejes cuerpos:

$$\begin{aligned} \mathbf{I}_b = & \begin{aligned} & I_{xx} = \int (y^2 + z^2) dm & I_{xy} = \int (x \cdot y) dm & I_{xz} = \int (x \cdot z) dm \\ & I_{yy} = \int (z^2 + x^2) dm & I_{yz} = \int (y \cdot z) dm \\ & I_{yx} = I_{xy} & I_{zy} = I_{yz} \\ & I_{zx} = I_{xz} \end{aligned} \end{aligned}$$

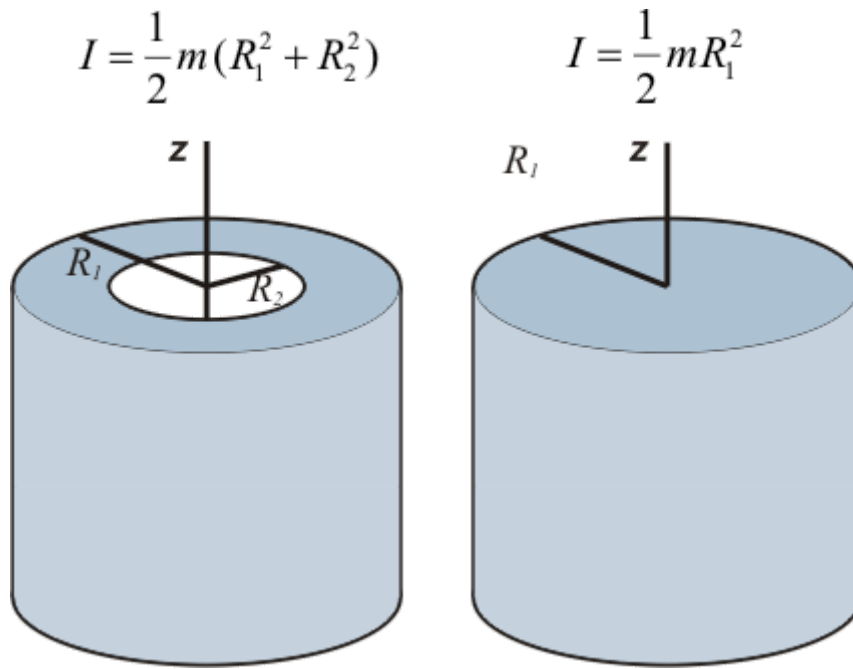
A partir del Tensor de Inercia en ejes cuerpos, podemos obtener el tensor de inercia en ejes mundo:

$$\mathbf{I}(t) = \mathbf{R}(t) \mathbf{I}_b \mathbf{R}(t)^T$$

$$\mathbf{I}^{-1}(t) = \mathbf{R}(t) \mathbf{I}_b^{-1} \mathbf{R}(t)^T$$

- **Momentos de inercia**

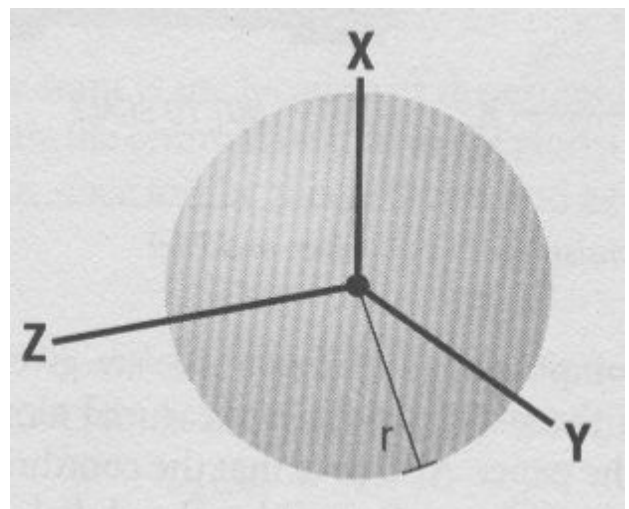
El **momento de inercia** (símbolo I) es una medida de la inercia rotacional de un cuerpo. Cuando un cuerpo gira en torno a uno de los ejes principales de inercia, la inercia rotacional puede ser representada como una magnitud vectorial llamada momento de inercia.



$$I_{xx} = I_{yy} = \frac{mr^2}{2} + \frac{ml^2}{12} \quad I_{zz} = mr^2$$

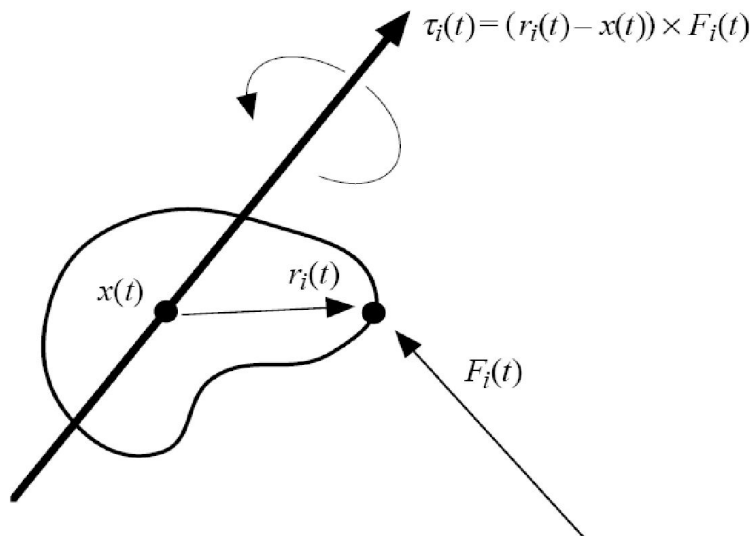
$$I_{xx} = I_{yy} = \frac{mr^2}{4} + \frac{ml^2}{12} \quad I_{zz} = \frac{mr^2}{2}$$

$$I_{xx} = I_{yy} = I_{zz} = \frac{2mr^2}{5}$$



- **Fuerzas**

Si la fuerza no se aplica en el *centro de masa* ésta crea un Par o Torque.



Momento lineal **$\boldsymbol{p} = M \cdot \boldsymbol{v}$**

La fuerza resultante es el cambio de momento lineal del cuerpo:

$$\sum \boldsymbol{F} = \frac{d\boldsymbol{p}}{dt} = \dot{\boldsymbol{p}} = M \cdot \dot{\boldsymbol{v}}$$

Momento angular **$\boldsymbol{L} = I \cdot \boldsymbol{\omega}$**

El Torque resultante es igual al cambio de momento angular:

$$\sum \boldsymbol{\tau} = \frac{d\boldsymbol{L}}{dt} = \dot{\boldsymbol{L}} = I \cdot \dot{\boldsymbol{\omega}}$$

Colisiones

1-Introducción

Formas de colisionar con un objeto

Las colisiones son necesarias en todo el mundo de la simulación y de los videojuegos. Son necesarias para poder “simular” lo que va a ocurrir. Son utilizadas en gran cantidad de proyecto como puede ser ver si un nuevo diseño de chasis ayuda a la hora de amortiguar un impacto. Evidentemente, y siendo el campo que más nos atañe a nosotros, el mundo de los videojuegos, las colisiones estan presentes y juegan un papel muy importante.

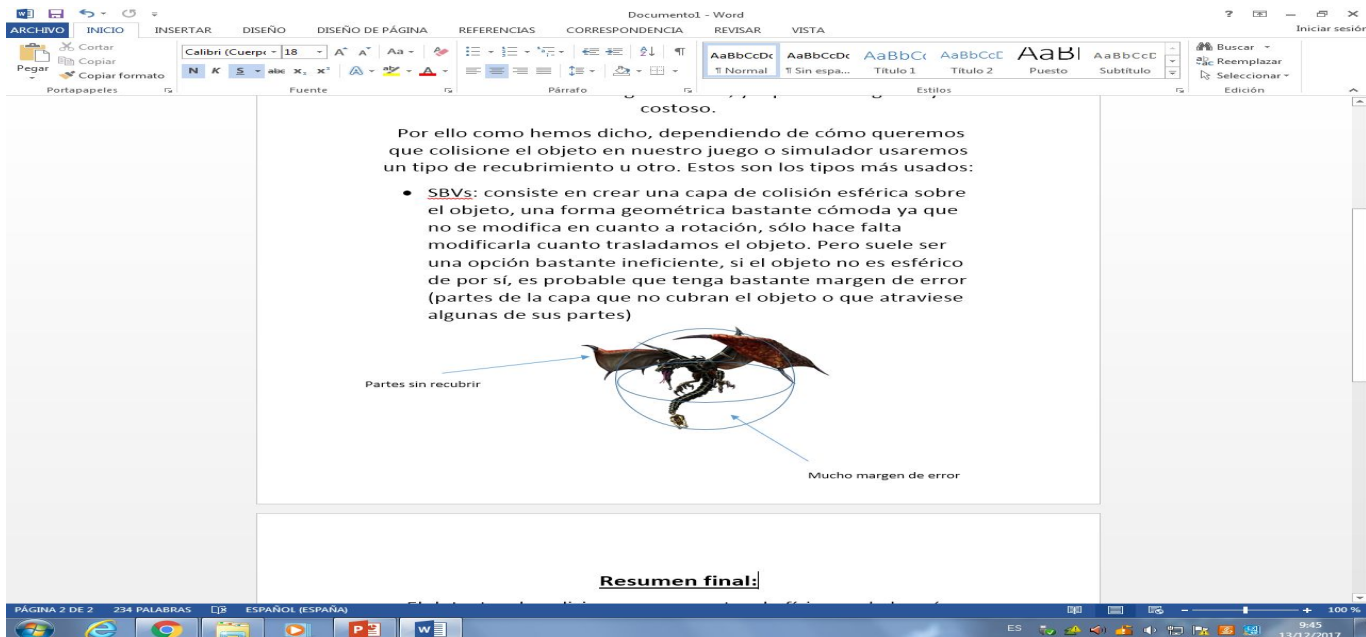
Podemos definir que es una colisión como el encuentro violento de dos o más cuerpos, de los cuales al menos uno está en movimiento.

En una colisión intervienen dos objetos ambos en movimiento que ejercen fuerzas mutuamente. Cuando los objetos están muy cerca entre sí o entran en contacto, interaccionan fuertemente durante un breve intervalo de tiempo. Las fuerzas de este tipo reciben el nombre de fuerzas impulsivas y se caracterizan por su acción intensa y breve. Un caso de este tipo de interacción, por ejemplo, es la colisión de dos carros que lleven montados parachoques magnéticos. Estos interactúan incluso sin llegar a tocarse, es lo que se considera colisión sin contacto.

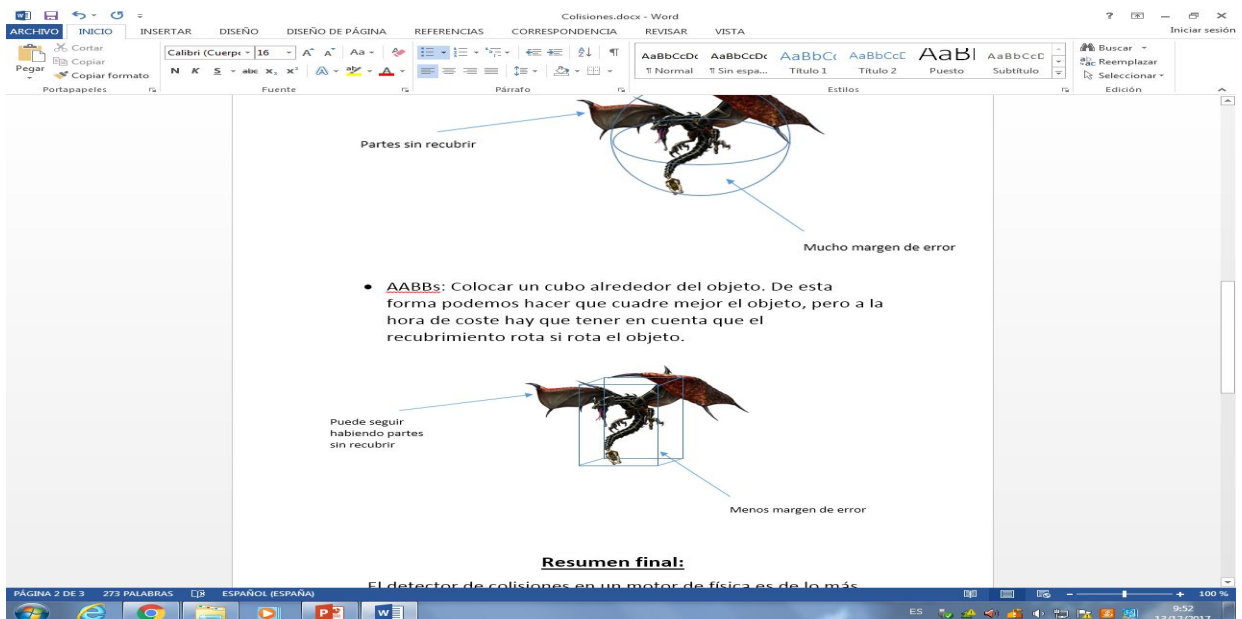
Dependiendo del tipo de objeto, su importancia en la escena y su tamaño, nos interesará crear una “capa” de colisión sobre él con una forma geométrica u otra. Para objetos complejos con muchas formas que lo compongan como por ejemplo un cuerpo de persona o animal o un vehículo, prácticamente nunca interesa crear una capa de colisión que coincida con su forma geométrica, ya que sería algo muy costoso.

Por ello como hemos dicho, dependiendo de cómo queremos que colisione el objeto en nuestro juego o simulador usaremos un tipo de recubrimiento u otro. Estos son los tipos más usados:

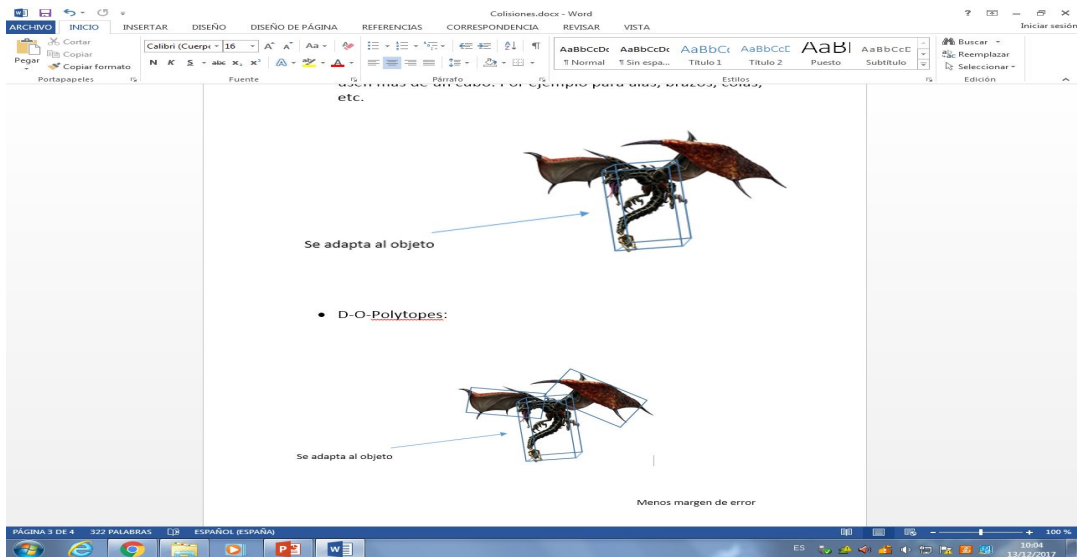
- SBVs: consiste en crear una capa de colisión esférica sobre el objeto, una forma geométrica bastante cómoda ya que no se modifica en cuanto a rotación, sólo hace falta modificarla cuanto trasladamos el objeto. Pero suele ser una opción bastante ineficiente, si el objeto no es esférico de por sí, es probable que tenga bastante margen de error (partes de la capa que no cubran el objeto o que atravesase algunas de sus partes). Suele usarse para objetos pequeños que no necesiten gran capacidad de colisión.



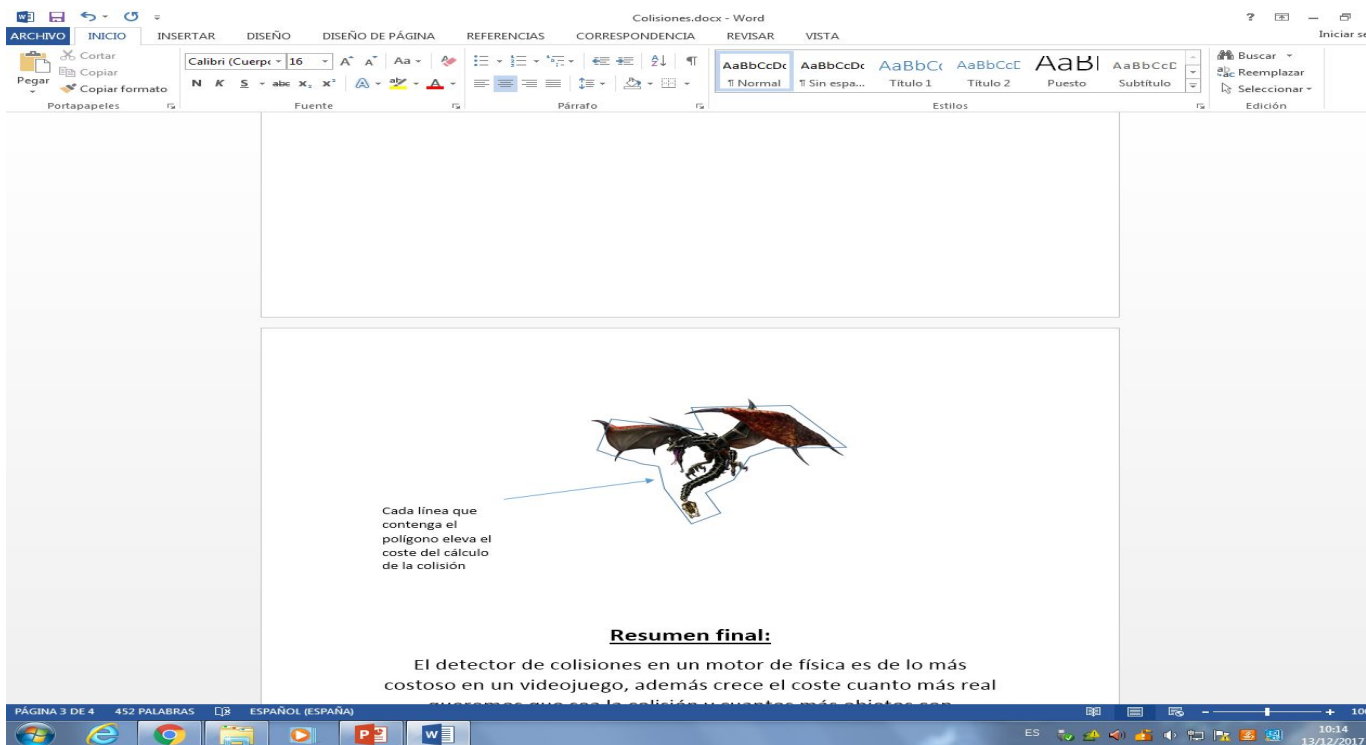
- **AABBs:** Colocar un cubo alrededor del objeto. De esta forma podemos hacer que cuadre mejor el objeto, pero a la hora de coste hay que tener en cuenta que el recubrimiento rota si rota el objeto. Muy útil para juegos hechos con tiles.



- **OBBs:** consiste en colocar un cubo orientado a la forma geométrica del objeto, que se adapte lo más posible. Tiene un coste de complejidad mayor. Lo normal es que si el objeto tiene partes que resaltan considerablemente, se use más de un cubo. Por ejemplo para alas, brazos, colas, etc.



- D-O-Polytopes: Recubrir totalmente el objeto con su forma geométrica, exceptuando resaltos insignificantes como pueden ser pinchos, púas, lengua, dedos, etc. Es elevadamente costoso, sólo debería hacerse para un objeto o entidad que requiera enorme similitud con la física real, como puede ser la colisión con un terreno uniforme, un cuerpo de persona en un juego de francotiradores con una física muy realista, etc.



2-Retrospectiva: Origen, evolución y tendencias

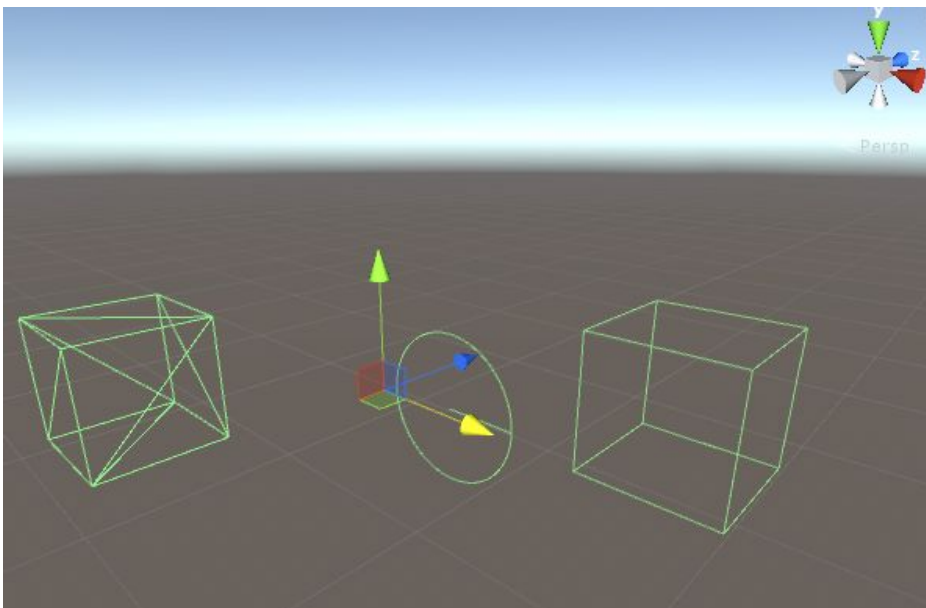
Las colisiones en el mundo de los videojuegos y de la simulación cobran gran importancia. En cualquier otro tipo de gremio se podría “falsear”, es decir en un video tu puedes imitar una colisión de dos objetos para mostrar un tema, ya que realmente no va a ser ninguna otra repercusión. Sin embargo, en el mundo de la simulación y de los videojuegos estas colisiones van a afectar de forma considerable al mundo que les rodea. Un ejemplo, de entre muchos, es por ejemplo la simulación de un coche chocando como afectaría al pasajero. En el caso de los videojuegos, desde la pisada de un personaje, hasta el rebote de una granada en una pared para modificar su trayectoria.

Origen:

Las colisiones ocurren durante todo o casi todo el tiempo en un videojuego. Para poder hablar algo mejor de las colisiones vamos a explicar sus características dentro del mundo del videojuego.

Características, evoluciones:

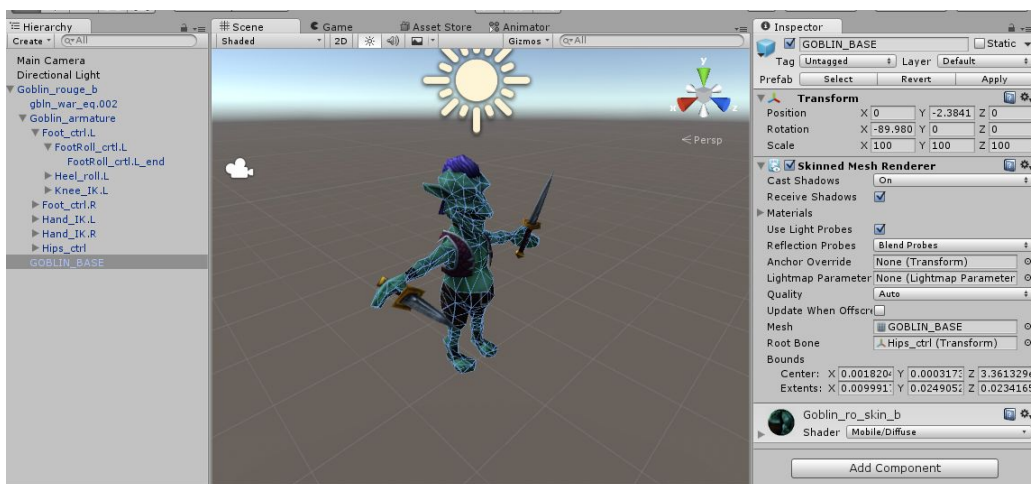
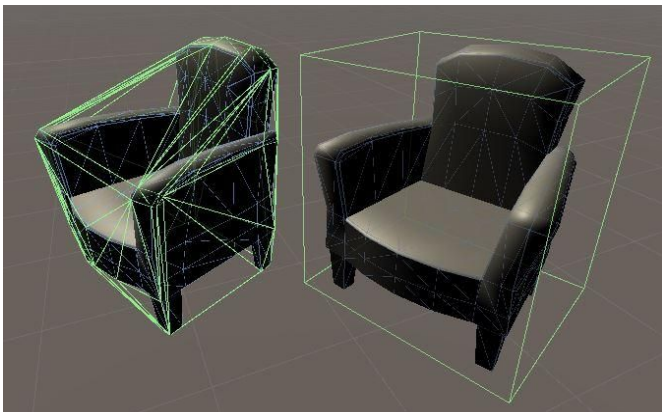
Para empezar vamos a dividir el mundo 2D del mundo 3D. A su vez, vamos a dividir las características en dos grandes apartados. Las cajas de colisión, BoxColliders, la propia física de colisiones.



Las Box Colliders son necesarias para representar el objeto en el mundo virtual, es decir, tú puedes introducir un cubo en un videojuego sin embargo, ese cubo no va a tener “cuerpo”, es decir, va a poder ser atravesado por cualquier otro objeto, cuando en la realidad esto no pasa. Para ello hay que adjuntarle un BoxCollider, es decir darle un elemento con el que poder manejar todo esto.

Empezando con el mundo 2D y sus Box Colliders, usualmente tenemos las formas más clásicas, cuadrados, círculos, polígonos y capsulas. Dependiendo de la imagen en 2D que tengas deberás utilizar uno u otro. Este BoxCollider debe estar bien ajustado al sprite en este caso porque si no puede producirse solapamiento de las imágenes y en ese caso parecer que se establece una colisión cuando no la hay realmente. Al no tener espacio como tal, sino que nos encontramos en un plano, este mundo de los Box Colliders es más reducido.

En cuanto al mundo 3D esto se complica, siguen permaneciendo los referentes como pueden ser las formas cúbicas, y esféricas o cápsulas, utilizadas usualmente para los personajes. Muchos motores nos incluyen modificaciones de estas para por ejemplo los terrenos del juego que pueden ser desde suelos lisos hasta complicados relieves de montaña.



En ambos casos, tanto en el mundo 2D como en el mundo 3D, se pueden utilizar varios para un mismo elemento. Esto tiene mucho que ver con la evolución de este campo, tema que explicaremos más adelante. Al poder tener varios, nos podemos facilitar mucho las colisiones. Un ejemplo de esto, pongamos que tenemos un personaje con su collider, cada vez que se choque recibe daño. Sin embargo ahora queremos añadir a nuestro juego enemigos que además queremos matar. Para poder hacer esto debemos detectar las colisiones de ambos objetos. Pero tal y como lo tenemos ahora tendríamos que hacer grandes modificaciones para que cuando atacásemos no se nos hiciese daño al chocarnos con tal objeto. Al poder tener varios podemos activar o desactivarlos a nuestro antojo algo que es infinitamente más fácil, y tener uno para el daño y otro para atacar.

Un apartado algo más alejado de las colisiones es el mundo de los triggers, cuando tu estableces un BoxCollider puedes indicarle si es trigger o no. Esto implica el poder o no poder atravesar dicho objeto. Se puede pensar que esto es inútil, para qué crear un objeto que hace que no lo atraviesen para generar colisiones y luego darle dicha propiedad. Bien esto puede ser muchísimo más útil de lo que pensamos, nos puede servir por ejemplo para crear un objeto invisible en una puerta de un juego con dicho componente, de tal forma que cuando atravesemos la puerta ocurra algo. Realmente lo que haces es colisionas con la caja pero no afecta físicamente, podríamos decir que nos sirve como informadores de la escena.

Por último el mundo de las físicas de las colisiones es el que hemos visto anteriormente en el sólido rígido, son fuerzas que actúan sobre distintos cuerpos y generan colisiones. Más que física en este caso entran en juego los materiales de los que estén hechos los objetos. No es lo mismo el rebote de una pelota de goma que el de una de plomo, y a su vez con el tipo de objeto con el que reboten. Otro punto a considerar es la dirección con la que impactan los objetos no es lo mismo impacta perpendicularmente que en cualquier otra combinación de vectores.

Podemos diferenciar entre tres tipos de colisiones.

Los **choques elásticos** se producen cuando dos objetos chocan y rebotan entre sí sin ningún cambio en sus formas. Los choques de las bolas de billar o los choques entre partículas subatómicas son un buen ejemplo de colisiones elásticas. En los choques elásticos se conservan tanto la cantidad de movimiento como la energía cinética.

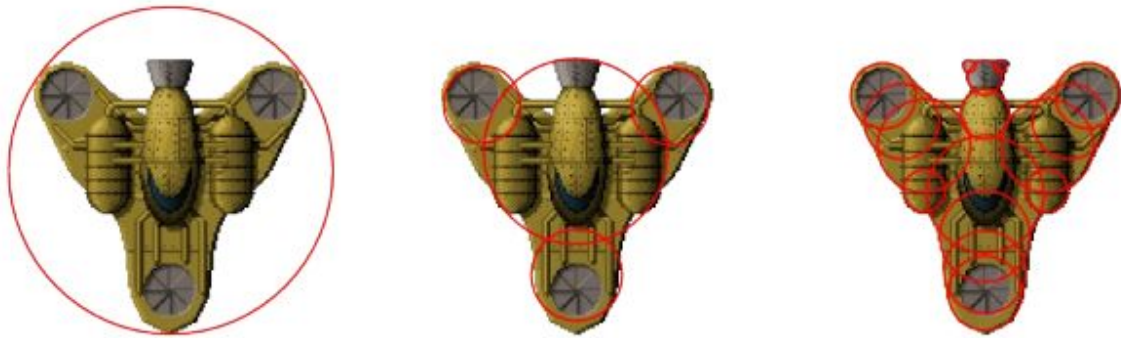
En los **choques inelásticos**, uno o los dos objetos que chocan se deforman durante la colisión. En estos choques la cantidad de movimiento se conserva, pero la energía cinética no se conserva ya que parte de ella se transforma en otro tipo de energía en el proceso de deformación de los cuerpos.

En los **choques totalmente inelásticos**, los cuerpos que chocan se mueven tras la colisión con la misma velocidad de manera que parecen estar pegados y se comportan como un único cuerpo. En este tipo de choques se conserva la cantidad de movimiento pero toda la energía puesta en juego en el choque se transforma en calor o deformación y no se recupera para el movimiento.

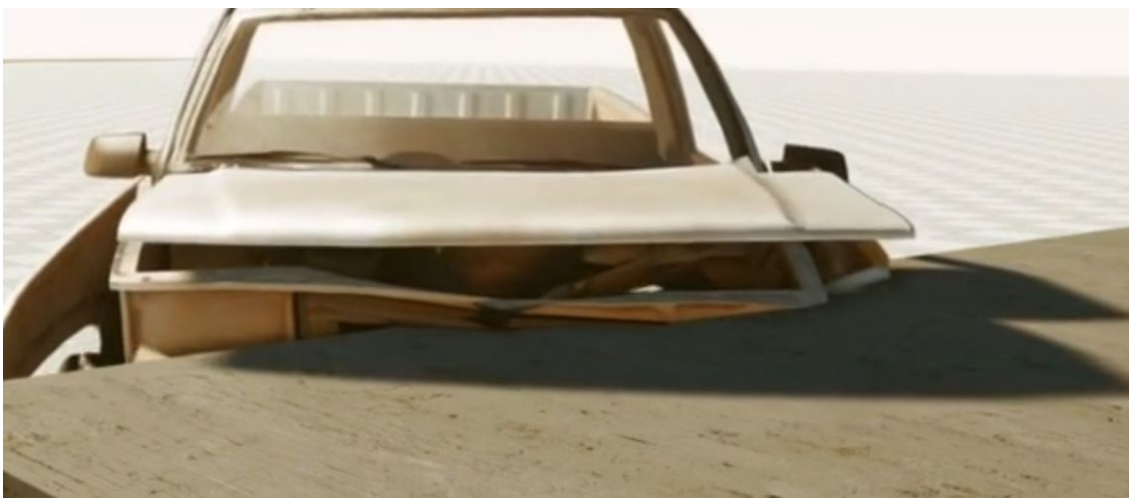
Evolución:

En cuanto a la evolución hay que mencionar que se han hecho grandes avances en el mundo de las colisiones, no solo a la hora de crear un BoxCollider sino de su modificación y de la modificación de la mallas debido a una colisión.

Como hemos dicho anteriormente tú puedes asignar varios colliders a un mismo objeto y que se comporten de la misma manera de esta forma consigues una mayor precisión a la hora de calcular la colisiones del objeto.



Otro avance ha sido la modificación de mallas, antes tú chocabas en un juego de coches y el coche seguía en perfectas condiciones. Esto evolucionó a cambiar la imagen, es decir tú te chocabas y lo que se hacía era cambiar la textura del objeto 3D para darle una apariencia distinta. A su vez esto fue evolucionando a cambiar las mallas de la textura de tal forma que ya no fuera una simple imagen sino que se deformara el modelo 3D. Sin embargo aún se puede avanzar para hacer aún más realista este campo, pues siempre se podrá deformar mejor una malla cuantos más vértices tenga, sin embargo tenemos un límite de memoria para esto.



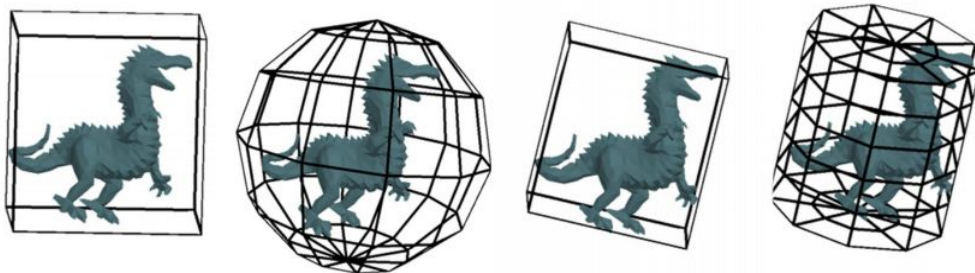
3-Teoría: modelado físico, formulación y matemáticas asociadas

Dada la complejidad actual de los modelos utilizados en los videojuegos se plantea el problema de cómo acercar el realismo de las colisiones mediante algoritmos a figuras geométricas complejas, dependiendo los mismos de la complejidad e importancia de las mismas dentro del sistema propuesto, tomando como precepto que el mismo se desarrolla en un entorno tridimensional. Esto nos plantea varios supuestos a los que atendernos a la hora de establecer la capa de colisiones.

El coste de la detección de colisiones es muy grande, en el orden de complejidad $O(N^2)$. Existen diferentes fases para calcular las colisiones de objetos:

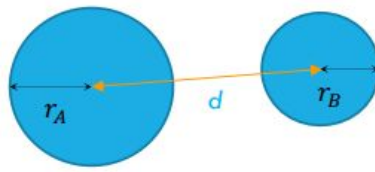
- Broad Phase (nivel de objetos)
- Narrow Phase (nivel de caras)
- Single Phase (ambos niveles)

Si el objeto es puramente geométrico, primitivo, la capa de colisión viene dada por un mismo volumen, lo que dota de gran simplicidad y bajo coste a la detección de dichas colisiones. Este mismo método de usar como capa colisionable una figura geométrica simple se aplica también a figuras geométricas complejas cuando las mismas carecen de gran importancia en el entorno planteado dado que la exactitud de su capa colisionable no supone una prioridad, por ejemplo en un juego de fútbol la grada es un cubo en cuanto a colisión y no un conjunto de personas con colisiones propias. A este método de colisión lo denominamos "Volumen envolvente". Dicho volumen envolvente puede categorizarse en:



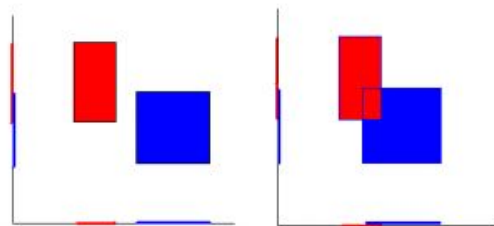
- SBVs o Bounding Sphere:

Esta solución crea la menor esfera posible que envuelva al objeto y mide la distancia de los centros de distintas esferas, por lo que las esferas colisionan si esta distancia es menor que la suma de sus radios. Sus ventajas son la invariancia respecto a la rotación y su actualización es una simple traslación.



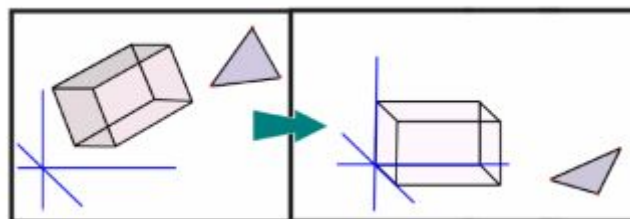
- AABB o Axis-Aligned Bounding Box:

Se crea un cubo con aristas siempre alineadas con los ejes de coordenadas y produce intervalos de proyecciones en cada eje para cada cuerpo y, por último, comprueba si estas proyecciones se solapan. Esta solución es bastante eficiente respecto al coste de computación, sin embargo, no es invariante respecto a la rotación por lo que hay que actualizar las AABB.



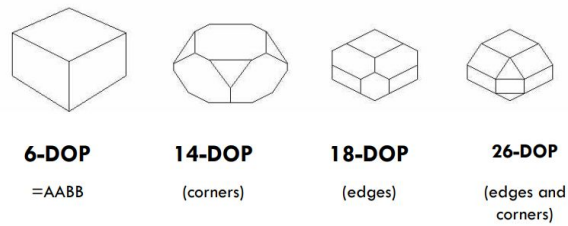
- OBB u Oriented Bounding Box:

Es una “caja” cuyos ejes están alineados al objeto tal que esta cubre al objeto de una manera óptima. De esta manera las colisiones son más precisas que los AABB y las esferas, pero su coste es mayor.

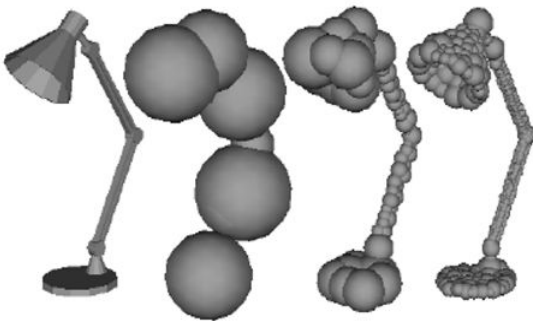


- D-O-Polytopes o Discreted Oriented Polytopes:

Es una extensión de los AABB/OBBs pero con un mayor número de lados. Es fácil de calcular y cubre de una manera más óptima.



El método SBVs es usado comúnmente mediante jerarquías de objetos, en este caso esferas, que nos permiten bajar el nivel de detalle. Esto se realiza gracias al algoritmo de Hubbard time-critical sphere tree algorithm que crea un árbol jerarquizado en cuyos nodos están las esferas de colisión.



El problema se plantea cuando debemos gestionar las colisiones de un gran número de entidades de geometría compleja que no podemos simplificar dada la exactitud necesaria en la comprobación, por ejemplo un combate de varios espadachines. En Narrow Phase se calculan las colisiones al nivel de caras de un objeto, para ello se emplea el algoritmo de Moor y Wilhelms en el que se realizan tres tests:

1. Comprobar si algún vértice del poliedro Q está contenido en el poliedro P y viceversa.
2. Comprobar si alguna de las aristas de Q penetran en las caras de P y viceversa.
3. (Infrecuente) Dos polígonos idénticos se muevan perfectamente alineados. Se comprueba aplicando 1 a los centroides de las caras.

Muchos sistemas solo contemplan el primer test, pues detectará la mayoría de las colisiones

La Single Phase administra las colisiones de los objetos utilizando estructuras espaciales, comúnmente empleada QSP trees.

Estado del arte, motores y juegos.

En la actualidad hay muchos motores de física, cuyos fines son variados, aunque en su mayoría a los videojuegos o a la simulación.

Vamos a analizar tres motores pretendiendo ejemplificar el estado actual de este mercado. Estos motores son Box2D, PhysX y, por último, Havok.

Physx:

PhysX es un motor propietario de capa de software intermedia o "middleware" y un kit de desarrollo diseñados para llevar a cabo cálculos físicos muy complejos. Anteriormente era conocido como la SDK de NovodeX, fue diseñado originalmente por AGEIA y tras la adquisición de AGEIA, es actualmente desarrollado por Nvidia e integrado en su chip gráficos más recientes.



Los motores físicos de middleware permiten a los desarrolladores de videojuegos abstraerse durante el desarrollo, ya que PhysX proporciona funciones especializadas en simulaciones físicas complejas, lo cual da como resultado una alta productividad en la escritura de código.

El 20 de julio de 2005, Sony firmó un acuerdo con AGEIA para usar la SDK de NovodeX en la consola PlayStation 3. Esto provocó que muchos desarrolladores empezaran a crear juegos muy complejos, antes impensables.. AGEIA afirmó que el PhysX era capaz de realizar estos procesos de cálculos físicos cien veces mejor que cualquier CPU creada anteriormente.

La PPU, una Unidad de Procesamiento de Física es un procesador especialmente diseñado para llevar a cabo cálculos físicos en un entorno 3D de videojuego. Procesos como partículas, humo o colisiones son ahora calculados y desarrollados por la PPU, en lugar de ser animaciones prediseñadas como se hacía hace unos años.

Se puede hablar ya de más de cien títulos que puedan aprovechar las cualidades de su procesador de física; algunos de ellos son los siguientes.

- B.A.S.E. Jumping
- Bet on Soldier: Blackout Saigon
- Bet on Soldier: Blood of Sahara
- Bet on Soldier: Blood Sport
- Beowulf: The Game
- Bladestorm: The Hundred Years' War
- Borderlands 2
- Borderlands: The Pre-Sequel!
- Brothers in Arms: Hell's Highway
- Captain Blood
- CellFactor: Combat Training
- CellFactor: Revolution
- City of Villains
- Crazy Machines 2
- Cryostasis: Sleep of Reason
- Dark Physics
- Desert Diner
- Dragonshard
- Dusk 12
- Empire Above All
- Empire Earth III
- Entropía Universe
- Fallen Earth
- Fallout 4 (en el parche 1.3)
- Frontlines: Fuel of War
- Fury
- Gears of War
- Gears of War 2
- Kuma\War
- Life Is Strange
- Mafia 2
- Mass Effect
- Mass Effect 2
- Medal of Honor: Airborne
- Metro 2033
- Metro: Last Light
- Mirror's Edge
- Mobile Suit Gundam: Crossfire
- Monster Madness: Battle for Suburbia
- Monster Truck Maniax
- Myst Online: Uru Live
- Nights: Journey of Dreams
- Nurien
- Open Fire
- Paragraph 78
- Pirates of the Burning Sea
- Prince of Persia
- PT Boats: Knights of the Sea
- Rail Simulator
- Resident Evil 5
- Red Steel
- Rise of Nations: Rise of Legends
- Robert Ludlum's The Bourne Conspiracy
- Roboblitz
- Sacred 2
- Shadowgrounds: Survivor
- Sherlock Holmes: The Awakened
- Showdown: Scorpion
- Silverfall
- Sovereign Symphony
- Sonic and the Secret Rings
- Tom Clancy's Ghost Recon Advanced Warfighter
- Tom Clancy's Ghost Recon Advanced Warfighter 2
- Tom Clancy's Rainbow Six: Vegas
- Tom Clancy's Splinter Cell: Double Agent
- Unreal Tournament 3
- Unreal Tournament 3: Extreme Physics Mod
- Warfare
- Warframe

Box2D:

Box2D es un simulador de física de código abierto, escrito en C++, para simular cuerpos rígidos en dos dimensiones.

Fue lanzado como Box2D Lite, un motor de muestra para una presentación de Erin Catto en 2006. En 2008 lanzaron su versión 2.0 introduciendo detección continua de colisiones y remodelando su API.

Se puede usar bajo la licencia *zlib*. Esto quiere decir que la licencia sólo tiene los siguientes puntos para tener en cuenta: El software es usado simplemente como tal. Los autores no son responsables de ningún daño provocado por el uso del mismo. Se puede usar en cualquier sistema que tenga un compilador de C++.

Ha sido usado en Nintendo DS, Wii, Android e iOS, entre otros.



Las características de este motor de simulación de cuerpos rígidos son las siguientes:

Colisión

- Detección continua de colisiones
- Callbacks de contacto (begin, end, pre-solve y post-solve)
- Polígonos convexos y círculos
- Formas múltiples en un cuerpo
- Un tiro puede tocar muchas formas
- Manejo eficiente de pares
- Consultas de fase amplia AABB rápidas
- Grupos de colisiones y categorías

Física

- Física continua con tiempo para resolver impactos

- Gráfica de contacto persistente de cuerpos
- Solución aislada y manejo de reposo
- Contacto, fricción y restitución
- Pila estable con solución en tiempo lineal
- Tipos como revolute, prismatic, distancia, pulley, gear, mouse joint
- Límites de las uniones, motores y fricción
- Corrección del momentum de la posición desacoplada
- Precisión razonable a las fuerzas, reacciones e impulsos

Sistema

- Bloques pequeños y asignación en la pila (stack)
- Ajuste de parámetros centralizados
- C++ portable, pues no usa contenedores STL

Cama de pruebas

- OpenGL con Freeglut
- Interfaz gráfica con GLUI
- Cambio fácil entre pruebas usando GUI
- Marco de pruebas para añadir nuevas pruebas
- Selección de mouse
- Archivos para el build del sistema

Documentación

- Manual del usuario
- Documento Doxygen con comentarios en el código
- Foro de usuarios activo

Ha sido usado en diferentes juegos y es popular por su uso en dispositivos móviles. Algunos de los videojuegos más populares que lo emplean son:

- 6 Dimensions
- Angry Birds
- Crayon Physics Deluxe
- Fantastic Contraption
- Incredibots
- Rolando
- Tiny Wings
- Transformice

HAVOK:



Havok Game Dynamics SDK es un motor físico (simulación dinámica) creado por la compañía irlandesa havok, es utilizado en videojuegos y recrea las interacciones entre objetos y personajes del juego. Detecta colisiones, gravedad, masa y velocidad en tiempo real llegando a recrear ambientes mucho más realistas y naturales.

Havok en sus últimas versiones se ejecuta entero por hardware mediante el uso de la GPU, liberando así de dichos cálculos a la CPU. Este se apoya en las bibliotecas Direct3D y OpenGL compatibles con Shader Model 3.0. En 2015 Havok fue adquirida por Microsoft.

Havok Physics ofrece un gran rendimiento en el tiempo de detección de colisiones y simulación real de soluciones físicas. Ofrece rapidez y robustez. Es totalmente personalizable. Dispone de un debugger visual que muestra información en tiempo real para la facilidad del programador.

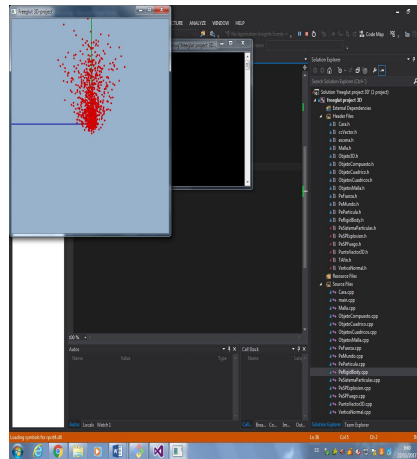
Havok ha sido implementado en más de 600 juegos y franquicias, incluyendo a Halo, Destiny y Tom Clancy entre otras.

Algunos ejemplos de videojuegos que utilizan Havok son:

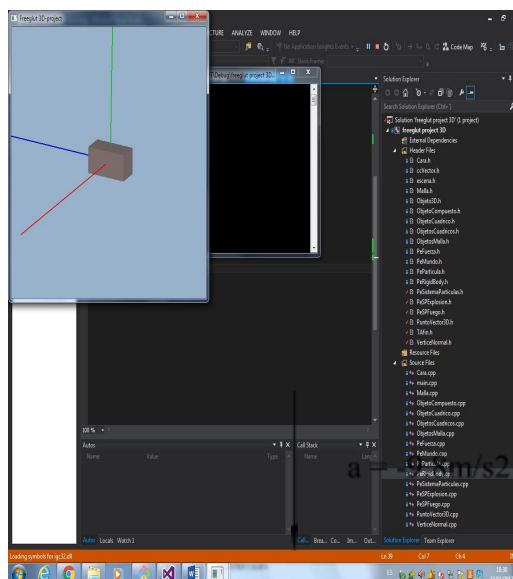
- Alan Wake
- Alan Wake's American Nightmare
- Assassin's Creed
- Assassin's Creed II
- Assassin's Creed: Brotherhood
- Assassin's Creed: Revelations
- Assassin's Creed III
- Battlefield 3
- BioShock
- Bioshock 2
- Bioshock Infinite
- Call of Duty 4: Modern Warfare
- Call of Duty: Black Ops II
- Call of Duty: Black Ops Declassified
- Counter-Strike: Source
- Dark Souls
- Grand Theft Auto IV
- Half-Life: Source
- Half-Life Deathmat ch: Source
- Half-Life 2
- Half-Life 2: Deathmat ch
- Half-Life 2: Episode One
- Half-Life 2: Episode Two
- Halo: Combat Evolved Anniversary
- Halo 2
- Halo 3
- Halo 3: ODST
- Halo 4
- Halo Reach
- Medal of Honor: European Assault
- Medal of Honor: Heroes
- Medal of Honor: Heroes 2
- Medal of Honor: Pacific Assault
- Medal of Honor: Vanguard
- Medal of Honor:
- Tom Clancy's Ghost Recon 2
- Tom Clancy's Ghost Recon Advanced Warfighter
- Tom Clancy's Ghost Recon Advanced Warfighter 2
- Tom Clancy's Splinter Cell: Chaos Theory
- Tom Clancy's Splinter Cell: Double Agent
- Tom Clancy's Rainbow Six: Uncharted: Drake's Fortune
- Uncharted 2: Among Thieves
- Uncharted 3: Drake's Deception
- The Last of Us
- Grand Theft Auto V
- Far Cry 4

Nuestro proyecto

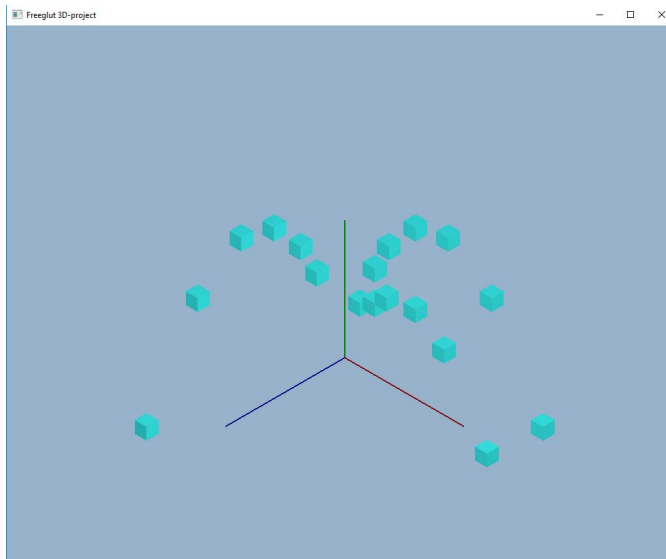
Acto seguido lo que hicimos fue hacer que la explosión se destruyera y volviera a aparecer indefinidamente e introdujimos también otro tipo de sistema de partículas: un fuego que regenera las partículas cada vez que se destruyen (a una cierta altura) de forma que parece que es constante y no se ralentiza con el tiempo.



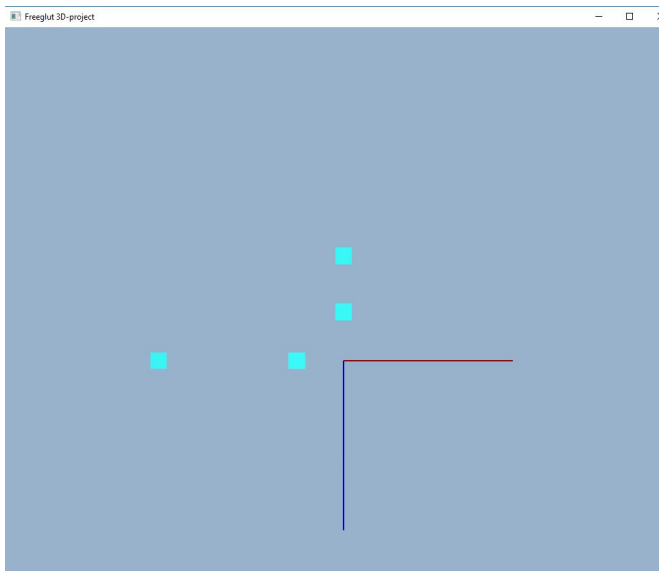
Después comenzamos ya con el siguiente objetivo que es introducir físicas a sólidos rígidos. Creamos una clase para los RigidBody que dibujara un ladrillo y le añadimos los atributos para la aceleración, velocidad, masa, fuerza externa, etc.



Aplicamos a una fuerza a un cubo que lo desplaza y luego actúa sobre él la fuerza de la gravedad.

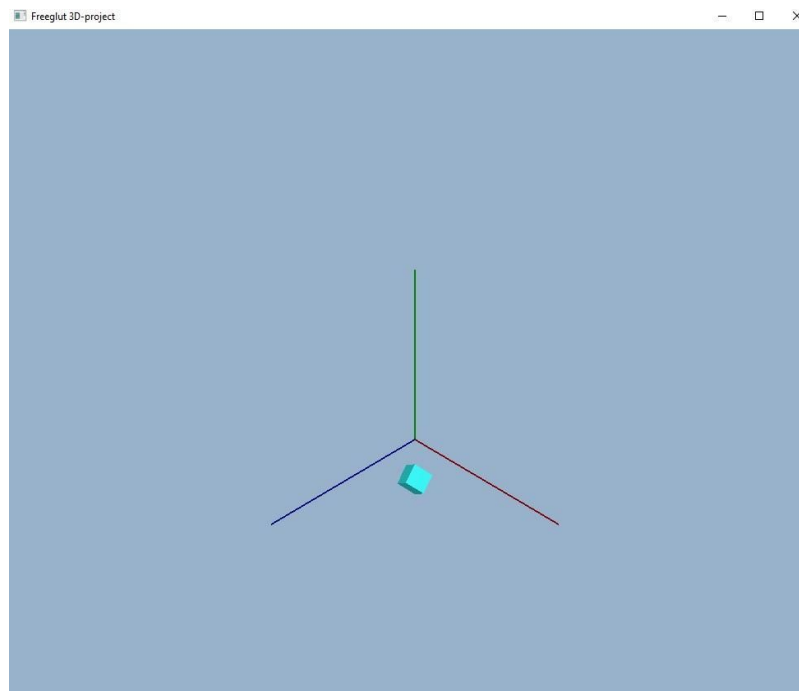


Movemos un cubo con impulsos sobre un suelo y este se frena debido a las fuerzas de rozamiento.

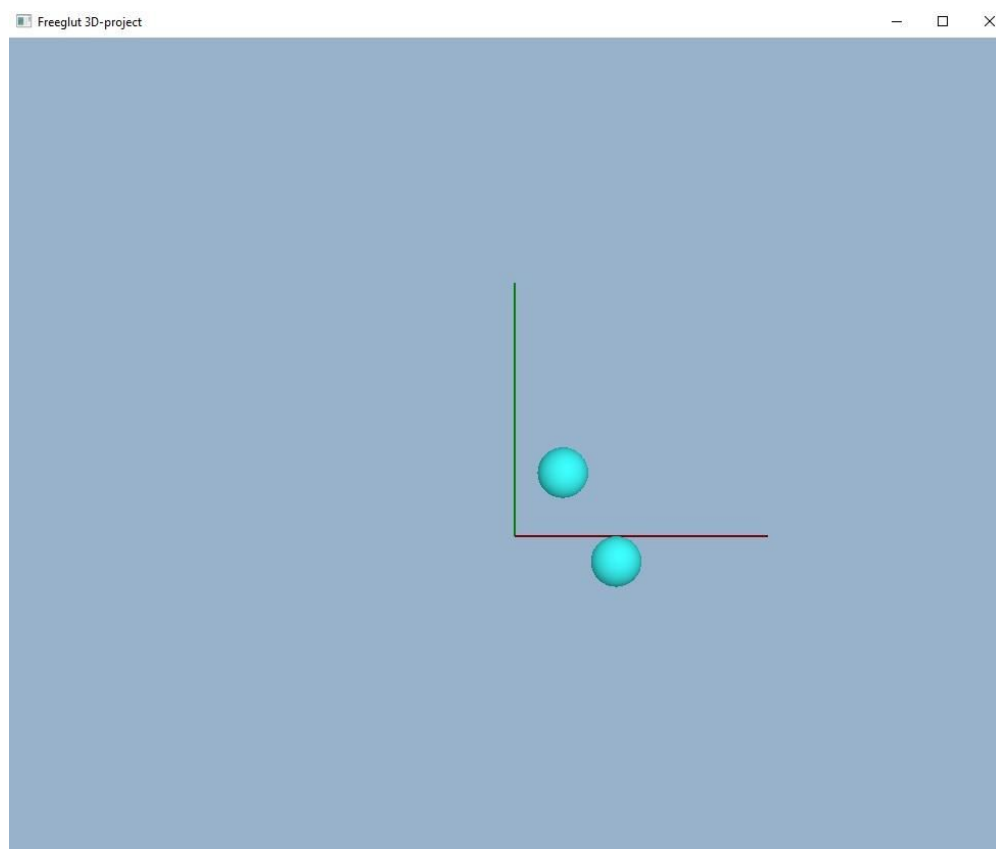


Hasta ahora teníamos cubos a los que les aplicábamos una fuerza y se movían en dicha dirección. Además estos estaban influidos por las fuerzas de rozamiento, que dependían del coeficiente de rozamiento.

Primero perfeccionamos la rotación, apartado que se nos había quedado algo pendiente del anterior sprint.



Respecto a las colisiones, hemos implementado los detectores de colisión de Cubos y Esferas. Y el choque entre objetos.



Demolición de Edificios

1.-Nuestra idea inicial del proyecto:

La idea que planteamos para llevar a cabo la programación de un motor de físicas fue intentar recrear la demolición y caída de edificios. Nuestra misión al comienzo de este proyecto era lograr simular el impacto de un proyectil que choca contra una serie de bloques apilados de forma que estos hiciesen de edificio, para que a la hora de impactar, la simulación de la fuerza aplicada al conjunto de bloques fuese realista y estos actuaran como trozos del edificio roto. Al producirse esa demolición, el impacto tanto del proyectil contra el edificio como el impacto de los restos del mismo contra el suelo debe de producir un sistema de partículas que simule el polvo y el humo que se formaría como resultado de esas situaciones.

Como referencia, buscamos diferentes videos y documentación que nos ayudaran a visualizar de forma clara cómo debía de ser el resultado de esa simulación. Encontramos videos echos con el motor de física que incorpora blender que se parecían bastante a los resultados que deseábamos tener al final de la implementación del motor.

2.-Para que íbamos a usar cada parte del proyecto:

Partículas

Como hemos adelantado anteriormente, el derrumbamiento de un edificio lleva consigo una cantidad enorme de consecuencias: trozos del edificio, polvo, humo, etc. Así pues, las partículas nos sirven para recrear todos estos ítems mediante la creación de sistemas independientes que recreen cada uno de esos elementos. Por ejemplo, para recrear el polvo que se produce al golpear el proyectil contra el edificio, se necesita un sistema de partículas que simule el color y el comportamiento del polvo, y que surja del punto del impacto.

Sólido Rígido

Para crear el edificio, creímos conveniente formarlo a partir del agrupamiento de múltiples bloques más pequeños. Para ello, necesitábamos implementar el comportamiento de un sólido rígido para que el choque del proyectil con el edificio proporcionase las fuerzas correspondientes a cada uno de los bloques con los que impactaba, para que el derrumbamiento del edificio se simulase de las formas más reales posible.

Colisiones

Para poder crear el edificio, necesitábamos que los bloques que lo forman se detectasen entre sí mediante la detección de colisiones. Además, para que el proyectil pudiera proporcionar las fuerzas correspondientes a los bloques con los que impacta, es necesaria la resolución de la colisión y el cálculo de las velocidades resultantes, así como de las fuerzas y los torques.

3.-Lo que hemos conseguido, lo que nos falta:

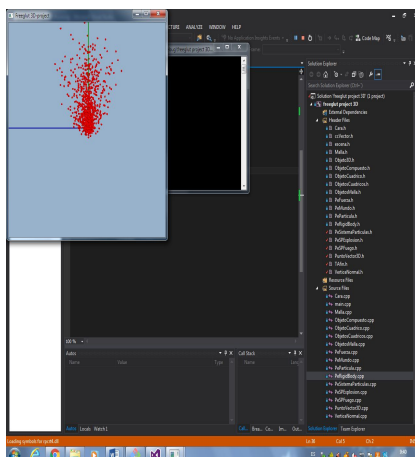
Para redactar lo que hemos conseguido en el proyecto vamos a hacer un resumen de todo lo que hemos ido avanzando.

Iremos en orden a la hora de enumerar lo que hemos conseguido en el proyecto, empezando por el sistema de partículas y finalizando con nuestra idea, la demolición de edificios.

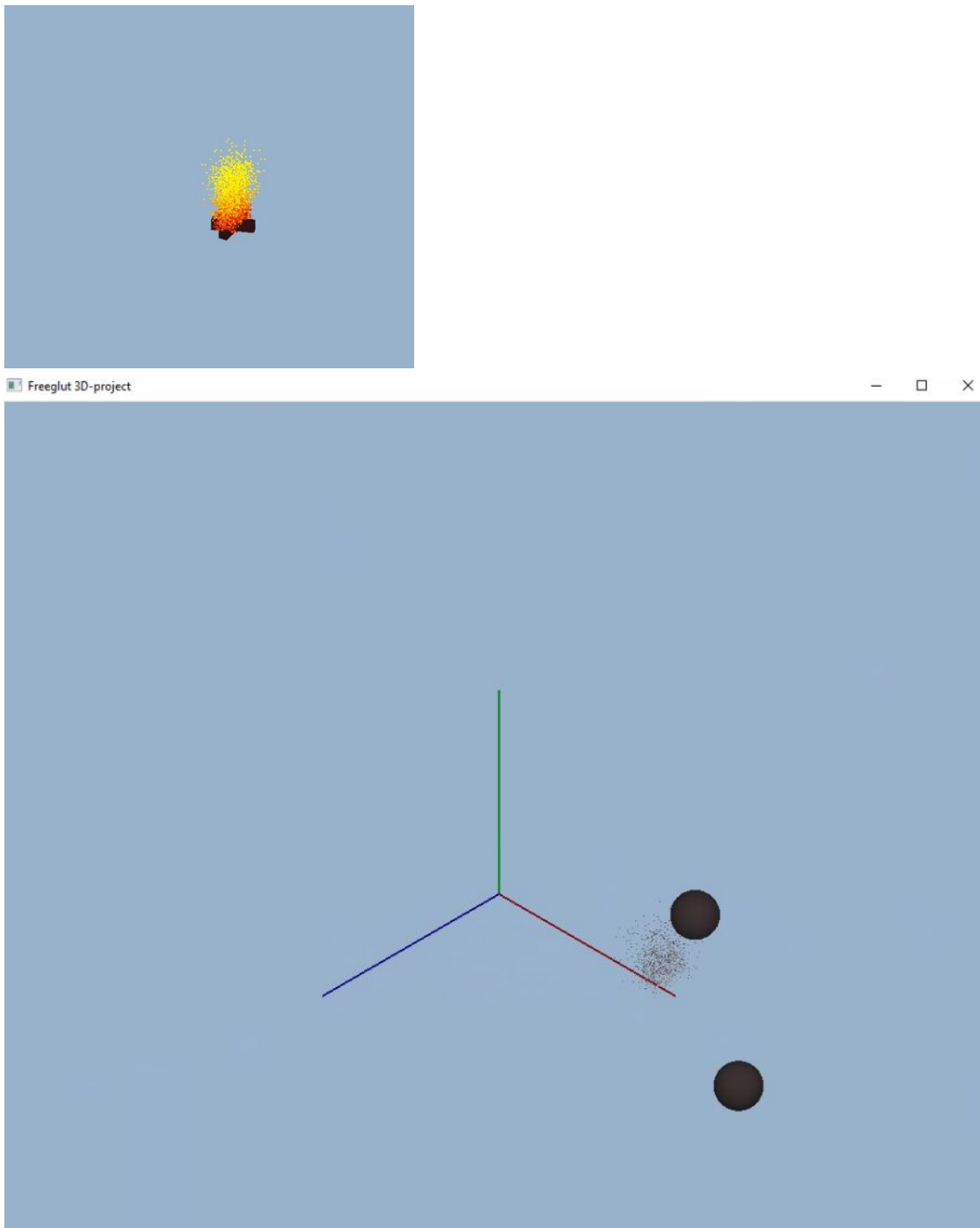
Empezamos trasteando con partículas y pequeños ejemplos como una explosión.



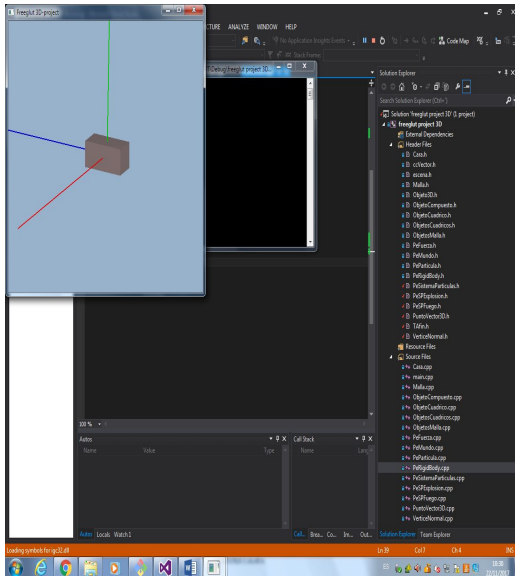
Seguimos haciendo varios experimentos con partículas.



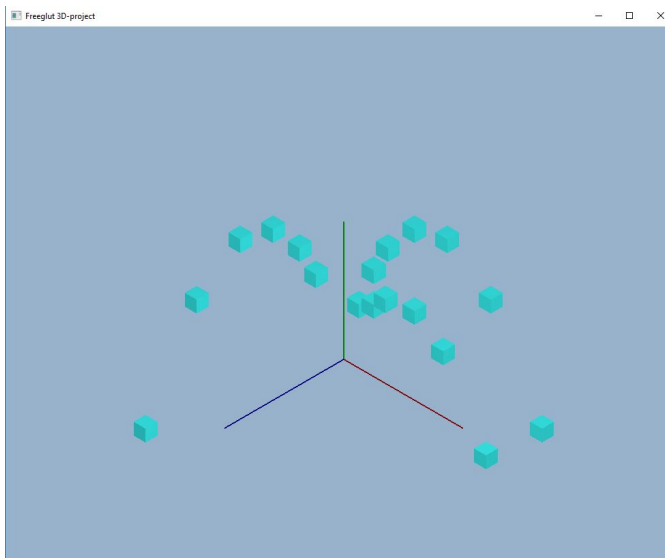
Tras esto, conseguimos tras intentos perfeccionar las partículas para poder hacer gran variedad de ejemplos. Hemos realizado un fuego y un efecto de partículas al colisionar dos esferas.



En cuanto a Sólidos Rígidos hemos conseguido que les afecte las fuerzas de gravedad, algo básico y fundamental para cualquier proyecto. Podemos crear cubos de distinto tamaños, con diferente masa, lo cual nos abre un gran mundo de pruebas con las distintas combinaciones.



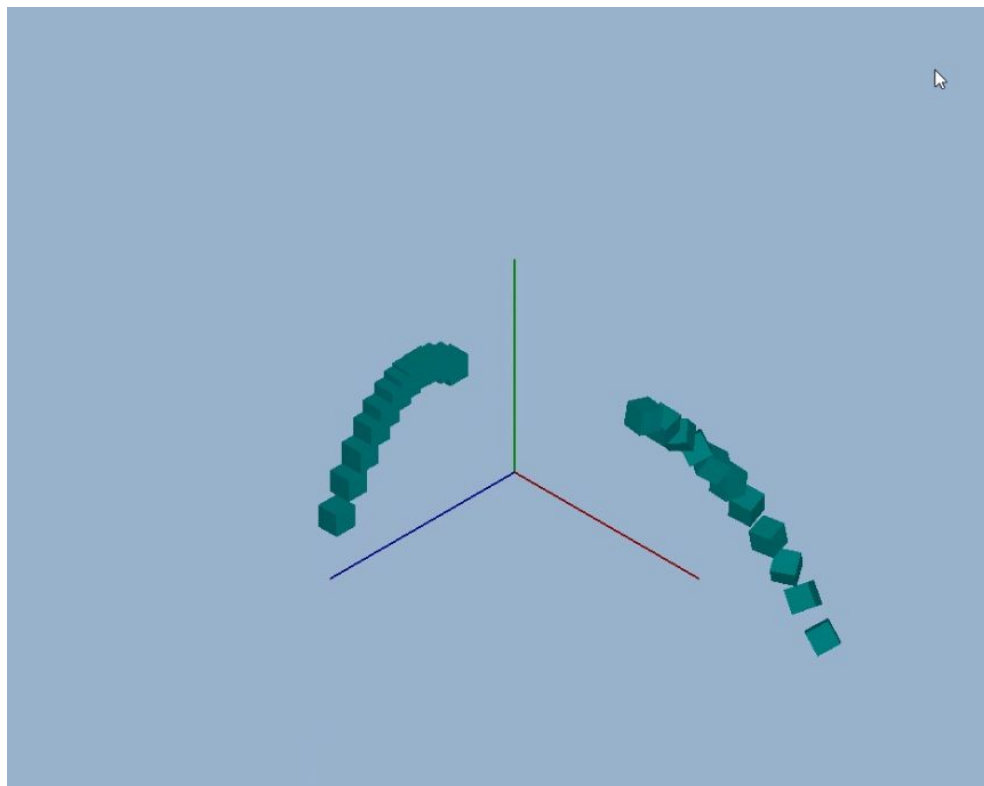
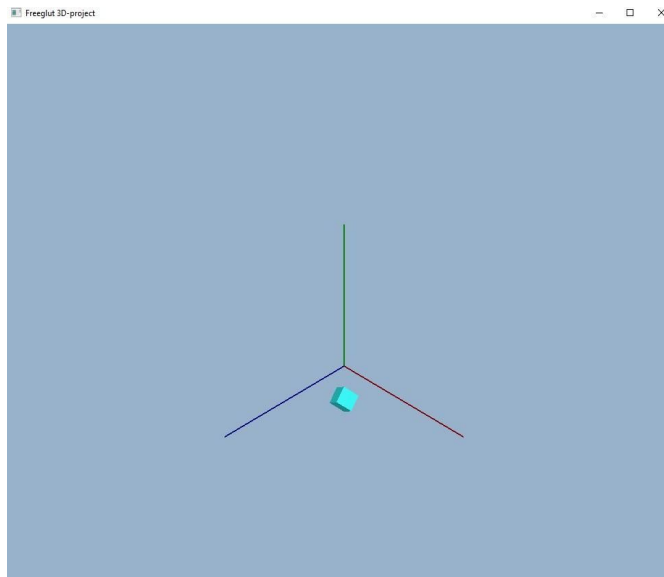
El siguiente paso fue aplicarles una fuerza, pero en este caso durante un instante de tiempo, es decir un impulso. Poder darles un impulso suponía poder lanzar, por ejemplo, un cubo hacia arriba y después que este se viese afectado por la gravedad de tal forma que este subía y bajaba.



También conseguimos darles un rozamiento con la que cubos de misma masa y mismo volumen pero con rozamientos distintos avanzaban más o menos en un suelo ficticio.

Una vez conseguido estos tres puntos, más o menos fáciles, comenzamos con las rotaciones de estos mismos. Este fue un apartado muy complejo en el que tuvimos grandes dificultades y que ocupó una gran cantidad de tiempo en nuestro proyecto.

Si embargo, conseguimos hacer la rotación de un cubo en cualquiera de sus tres ejes y en cualquier combinación de estos.



Respecto a las colisiones, la parte más compleja del proyecto, hemos implementado los detectores de colisión de Cubos y Esferas. También conseguimos un principio de la colisión entre esferas y cubos en un sistema 2D, es decir donde solo se tienen en cuenta dos de los ejes. Sin embargo no conseguimos avanzar más en colisiones.



En cuanto a la demolición de edificios, finalmente no hemos podido hacer nada, ya que básicamente el core de esta idea es la colisión entre los objetos, apartado que como hemos mencionado antes no hemos logrado completar a la perfección. Si que es cierto que se podría mostrar un choque entre dos cubos, uno simulando una bola de demolición y el otro un edificio, no obstante muchos de los resultados sería irrealistas.

En lo que a partículas se refiere no hemos tenido grandes dificultades, hemos conseguido todo lo que nos proponíamos, simplemente es cuestión de cambiar parámetros para conseguir distintos efectos.

En cuanto a sólido rígido, al igual que en los sistemas de partículas, en principio hemos conseguido todo lo previsto.

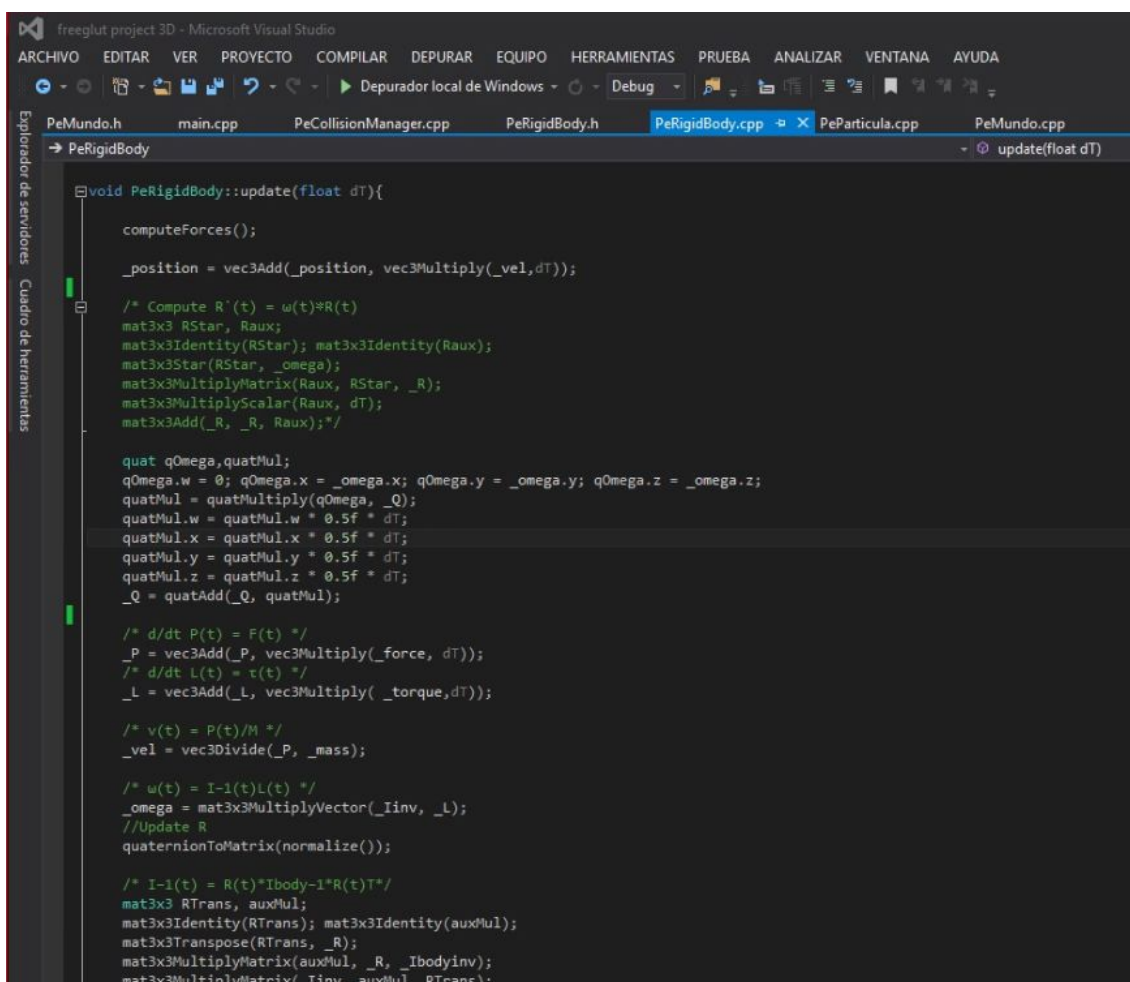
Pero en colisiones, como hemos dicho anteriormente, solo hemos conseguido una introducción a estas mismas. Los diferentes problemas encontrados a la hora de calcular las nuevas trayectorias, supusieron que no consiguiésemos el funcionamiento deseado.

4.-Dificultades encontradas:

Básicamente podríamos definir tres grandes problemas que encontramos a la hora de realizar este proyecto:

Fuerte contenido teórico. El mundo de las físicas es un campo muy extenso y muy complejo. Realizar un motor de físicas requiere grandes conocimientos físicos. Uno de los problemas es que por parte de todos los integrantes del grupo, solo tenemos conocimientos de física de 2º de Bachillerato. Esto supuso una pequeña dificultad para el equipo.

Rotaciones, a pesar de que al final conseguimos obtener las rotaciones, fue uno de los puntos más complejos para el equipo. En principio decidimos utilizar matrices de rotación en vez de cuaterniones para su realización. Pero el problema de las matrices era que acumulaban errores de cálculo, lo que suponía que deformase el objeto en sí. Ya que los cuaterniones eran un tipo de datos que no habíamos ni sabíamos manejar, simplemente sabíamos que existían. A partir de aquí, decidimos usar cuaterniones, aprender como usarlos, como funcionaban, y que significaba la variación de cada una de sus componentes. Además cambiar los cálculos de la rotación en código.



```
freelut project 3D - Microsoft Visual Studio
ARCHIVO  EDITAR  VER  PROYECTO  COMPILAR  DEPURAR  EQUIPO  HERRAMIENTAS  PRUEBA  ANALIZAR  VENTANA  AYUDA
Depurador local de Windows  Debug
PeMundo.h  main.cpp  PeCollisionManager.cpp  PeRigidBody.h  PeRigidBody.cpp  PeParticula.cpp  PeMundo.cpp
→ PeRigidBody
- update(float dT)

void PeRigidBody::update(float dT){
    computeForces();
    _position = vec3Add(_position, vec3Multiply(_vel,dT));

    /* Compute R'(t) = ω(t)*R(t)
    mat3x3 RStar, Raux;
    mat3x3Identity(RStar); mat3x3Identity(Raux);
    mat3x3Star(RStar, _omega);
    mat3x3MultiplyMatrix(Raux, RStar, _R);
    mat3x3MultiplyScalar(Raux, dT);
    mat3x3Add(_R, _R, Raux);*/

    quat qOmega, quatMul;
    qOmega.w = 0; qOmega.x = _omega.x; qOmega.y = _omega.y; qOmega.z = _omega.z;
    quatMul = quatMultiply(qOmega, _Q);
    quatMul.w = quatMul.w * 0.5f * dT;
    quatMul.x = quatMul.x * 0.5f * dT;
    quatMul.y = quatMul.y * 0.5f * dT;
    quatMul.z = quatMul.z * 0.5f * dT;
    _Q = quatAdd(_Q, quatMul);

    /* d/dt P(t) = F(t) */
    _P = vec3Add(_P, vec3Multiply(_force, dT));
    /* d/dt L(t) = τ(t) */
    _L = vec3Add(_L, vec3Multiply(_torque,dT));

    /* v(t) = P(t)/M */
    _vel = vec3Divide(_P, _mass);

    /* ω(t) = I-1(t)L(t) */
    _omega = mat3x3MultiplyVector(_Iinv, _L);
    //Update R
    quaternionToMatrix(normalize());

    /* I-1(t) = R(t)*Ibody-1*R(t)T*/
    mat3x3 RTrans, auxMul;
    mat3x3Identity(RTrans); mat3x3Identity(auxMul);
    mat3x3Transpose(RTrans, _R);
    mat3x3MultiplyMatrix(auxMul, _R, _Ibodyinv);
    mat3x3MultiplyMatrix(_Iinv, auxMul, RTrans);
```

Colisiones, apartado en el que apenas hemos avanzado, conseguimos hacer una pequeña introducción a colisiones, a pesar de tener calculada la fórmula de traspaso de fuerzas, es decir las fuerzas resultantes tras la colisión, no conseguimos obtener el plano de colisión. Este plano hubiese facilitado obtener la normal y a su vez obtener los grados de incidencia del choque con respecto al plano que formaba los objetos. De esta forma podríamos haber calculado la dirección resultante después del choque.

No definimos una clara arquitectura desde el principio, si que es cierto que seguimos una pequeña orientación, sin embargo no es suficiente. A la hora de generar nuevo código es cierto que ha habido problemas sobre donde realmente era mejor introducirlo. Hubiese sido más fácil haber planteado una fuerte organización del código, ya que hubiese sido a su vez mucho más fácil añadir código.

Conclusiones:

El sistema de partículas se emplea para objetos o instancias cuyas características no se pueden definir como si fueran las de un sólido rígido; por ejemplo para representar fenómenos como el fuego, el agua, el humo, etc.

El sistema de partículas se emplea constantemente en videojuegos que contengan algo de ambientación, incluso en algunos juegos en 2D.

Algunos ejemplos de motores de físicas que se emplean son Physx y Havok, y hay muchos ejemplos de videojuegos que emplean dichos motores para la representación de partículas y demás componentes y efectos físicos.

Toda entidad en un videojuego que tenga una forma definida necesita un RigidBody, un sólido rígido al que poder aplicarle una fuerza, aceleración, y que pueda detectar colisiones.

Un sólido rígido se diferencia de una partícula en que es capaz de tener movimiento de rotación.

En nuestro proyecto, usaremos sólidos rígidos para la creación de un edificio, al que poder aplicarle una fuerza externa mediante una explosión o una bola de demolición para derribarlo.

Tenemos un RigidBody al que le podemos aplicar fuerzas, el siguiente paso es introducir colisiones entre sólidos rígidos. (Emplearemos un sólido rígido para representar el suelo y que colisione al caer).

Toda entidad en un videojuego que tenga una forma definida necesita un RigidBody, un sólido rígido al que poder aplicarle una fuerza, aceleración, y que pueda detectar colisiones.

Dependiendo de su forma puede ser más útil utilizar un tipo u otro de BoxCollider, ya sea un cubo, una esfera, o cualquier otro tipo de poliedro

En nuestro proyecto utilizaremos la colisión a la hora de hacer colisionar una bola contra un edificio. También en el momento de construcción del edificio.

Referencias

- *hacer.forestaes.upm.es*
- *fisicalab.com*
- *laplace.us.es*
- *estudiarfisica.com*
- *havok.com*
- *nvidia.es*
- *wikipedia*
- *mosaic.uoc.edu*
- *Unity 5.3.1*
- *hacer.forestaes.upm.es*
- *monografias.com*
- *es.scribd.com*
- *sc.ehu.es*
- http://www.educaplus.org/momentolineal/tipos_choques.html
- <https://www.genbetadev.com/programacion-de-videojuegos/teoria-de-colisiones-2d-conceptos-basicos>
- <http://www.vandal.net/video/19401/rigs-of-rods-cryengine-3>
- <https://www.scss.tcd.ie/Michael.Manzke/CS7057/cs7057-1516-05-BroadphaseCD-mm.pdf>
- Documentos del Campus Virtual