



MOTOR DE COLISIONES
SHIFT videogame

GAMELAB
UNIVERSIDAD DE OVIEDO

Alberto Carlos del Blanco Maraña
Madrid 28 de junio de 2007



Tabla de Contenidos

1. Introducción	4
2. Parámetros de la clase <i>collision</i>	6
2.1. Introducción	6
2.2. Cálculo del área de colisión	7
2.3. Detección de la colisión	8
2.3.1. Caso sin colisión entre objetos A y B	10
2.3.2. Caso con colisión entre objetos A y B	11
2.4. Recolocación de los objetos	12
2.5. Intercambio de parámetros físicos	15
2.5.1. Choque “ <i>Móvil-Móvil</i> ”	16
2.5.2. Choque “ <i>Móvil-Fijo</i> ” (fijo con física)	19
2.5.3. Choque “ <i>Móvil-Fijo</i> ” (fijo sin física)	20



Lista de Figuras

Imagen 1 : Límites del motor de colisiones.....	4
Imagen 2 : Cálculo del área de colisión en un objeto.....	6
Imagen 3 : Cálculo del área de colisión en un objeto.....	7
Imagen 4 : Evaluando colisión (situando puntos)	8
Imagen 5 : Evaluando colisión (evaluando puntos).....	9
Imagen 6 : Detección A-B sin colisión.....	10
Imagen 7 : Detección B-A sin colisión.....	10
Imagen 8 : Detección A-B con colisión.....	11
Imagen 9 : Detección B-A con colisión.....	11
Imagen 10 : Colisiones sin recolocación	12
Imagen 11 : Colisiones con recolocación	12
Imagen 12 : Ejemplo de recolocación	14
Imagen 13 : Esquema tras la recolocación	15
Imagen 14 : Simplificación al choque de dos vectores.....	16
Imagen 15 : Parámetros físicos considerados.....	17
Imagen 16 : Parámetros físicos considerados.....	19
Imagen 17 : Parámetros físicos considerados.....	20



1. Introducción

El presente documento detalla el sistema de colisiones diseñado para el videojuego SHIFT, como parte del proyecto fin de carrera realizado por Alberto Carlos del Blanco para la E.P.S.I.G. de la Universidad de Oviedo.

Una de las tareas más importantes del motor gráfico es el sistema de detección de colisiones, ya que supondrá el mayor consumo computacional del programa y de él dependerá no sólo el rendimiento, sino la jugabilidad y el realismo final de cara al usuario

Dentro del esquema general del motor gráfico, el módulo físico queda limitado por el ámbito rayado:

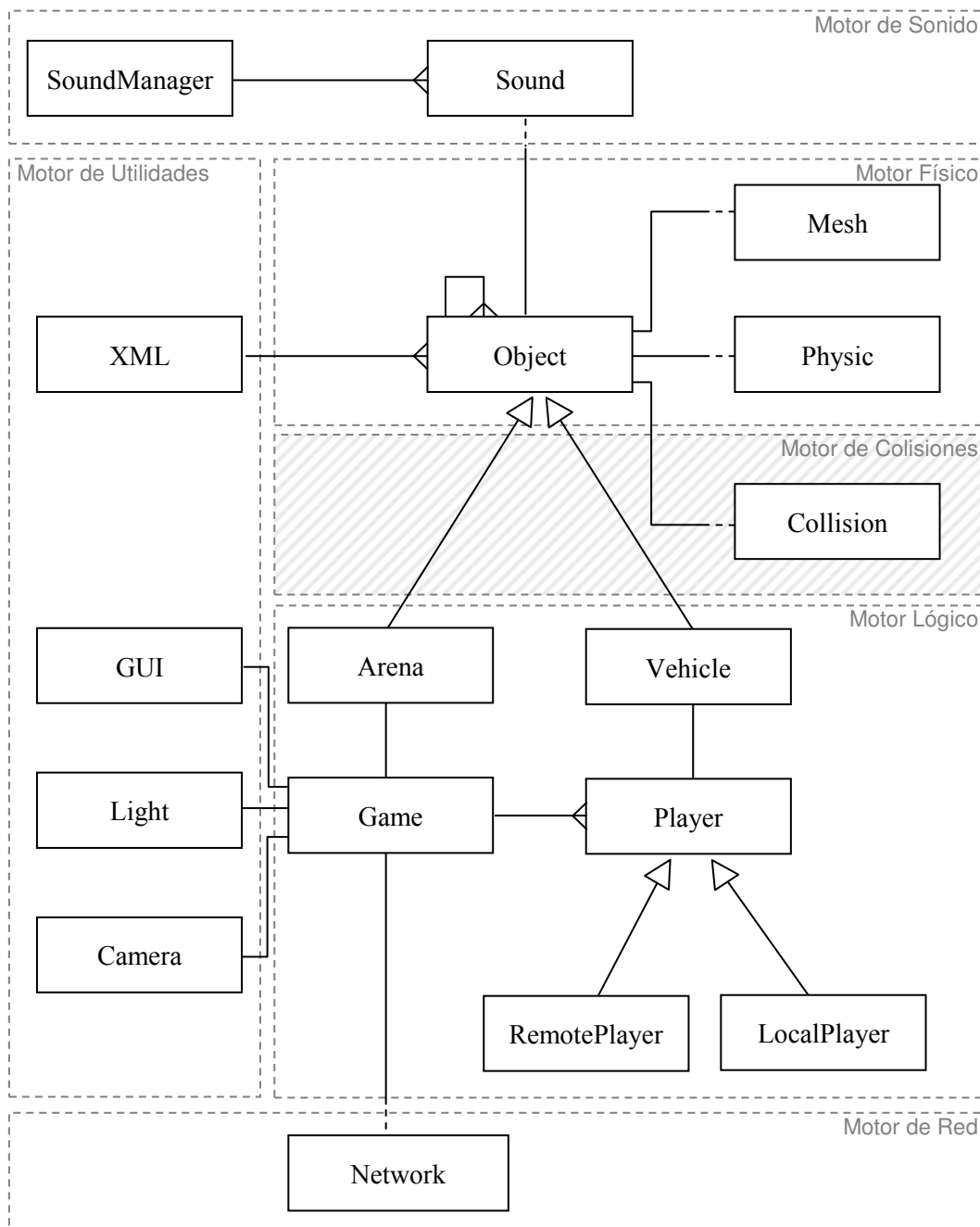


Imagen 1 : Límites del motor de colisiones



Cómo refleja en el diagrama, el módulo de colisiones se abstrae con una única clase:

- **Collision**

Clase que permite implementar toda la lógica de colisiones entre objetos, como son la detección y el intercambio de parámetros físicos.

Cada instancia de esta clase irá asociada a un objeto, y gestionará en cada momento sus colisiones a través de la instancia física que posea. Sin embargo no se gestionarán colisiones para todos los objetos, ya que muchos de los que se encuentren en escena no entrarán dentro de la lógica aquí descrita.

Como se ha dicho, el motor de colisiones supone el mayor consumo computacional del videojuego, pero además constituye el módulo más complejo de cara al diseño y la implementación, teniendo un carácter más matemático que informático.

Existen varios motores de colisiones en dos y tres dimensiones, tanto a nivel teórico, como a nivel práctico, en muchos casos gracias a la reciente apertura de código de clásicos en el software de entretenimiento. Sin embargo, como añadido al desarrollo del proyecto, se ha diseñado un motor de colisiones ad-hoc, atendiendo las necesidades propias de este proyecto.

En este sentido, el sistema de colisiones toma la simplicidad como principal y único criterio de diseño, ya que no se pretende realizar un algoritmo avanzado y porque en este caso se prefiere sacrificar la eficiencia en pro de un diseño sencillo y comprensible.

Siguiendo esta idea, se realizan varias simplificaciones en el sistema:

- Aunque el resto del motor gráfico está basado en tres dimensiones, las colisiones se reducen al plano del escenario, por lo que sólo se consideran **dos dimensiones**.
- Ahora bien, en dos dimensiones, la proyección de los objetos colisionables modelados tienen una geometría rectangular, o en todo caso su geometría puede descomponerse en varias rectangulares. Por este motivo, sólo se considera el tipo de colisión **rectángulo-rectángulo**.
- Por último, y como consecuencia de las dos simplificaciones anteriores, **no todos los objetos de la escena serán colisionables**, ya que no todos estarán situados al nivel bidimensional del plano de colisiones considerado.

Y teniendo ambas simplificaciones en cuenta, en los siguientes apartados se van a diseñar los algoritmos necesarios para incorporar:

- Cálculo del área de colisión
- Sistema de detección de la colisión
- Recolocación de los objetos (tras la colisión)
- Intercambio de parámetros físicos (tras la colisión)

2. Parámetros de la clase *collision*

2.1. Introducción

Desde un punto de vista técnico, el videojuego es en realidad un proceso iterativo continuo, en el que tras cada ejecución se actualizan los parámetros de la escena, considerando para ello el tiempo transcurrido desde la anterior iteración.

Actualizar la escena conlleva recalcular las nuevas ubicaciones de los objetos, según su física o comandos de usuario. Tras ello, algunos objetos pueden resultar superpuestos por lo que será necesario modificar sus propiedades para evitar incoherencias físicas.

Las colisiones se van a reducir a un plano ficticio, llamado **plano de colisiones**. Los objetos 3D que crucen ese plano tendrán asociado en él un área en forma de rectángulo 2D, llamado **área de colisión**. En otro caso los objetos que no interactúen con el plano de colisiones serán excluidos del proceso aquí descrito.

- **Objetos colisionables**, cruzan el plano imaginario de colisión, por lo que implementan lógica de colisiones, reduciendo su geometría a un área de colisión en forma de rectángulo.
- **Objetos no colisionables**, están a otro nivel diferente del plano de colisión, por lo que no son considerados por este motor.

Además, dentro de los objetos colisionables, hay que tener en cuenta que los objetos pueden ser:

- **Objetos fijos**, como por ejemplo los edificios, caracterizados por tener siempre física y ubicaciones fijas, por lo que de cara al proceso de colisión no pueden moverse ni ver modificada su física.
- **Objetos móviles**, como por ejemplo los vehículos, que en este caso sí pueden moverse y ver modificada su física.

Por lo que, desde el punto de vista de las colisiones, tendremos la siguiente clasificación de los objetos:

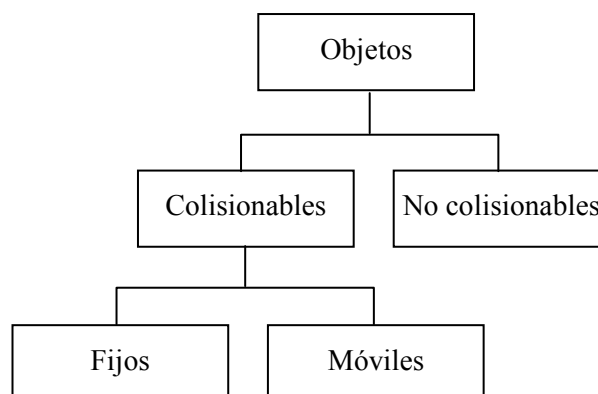


Imagen 2 : Cálculo del área de colisión en un objeto

En todo momento se tienen identificados los objetos colisionables, sus áreas rectangulares de colisión y su tipo de movilidad física, por lo que para el resto del diseño trataremos sólo estos datos.

2.2. Cálculo del área de colisión

El primer paso para plantearse el sistema de colisiones consiste en un algoritmo que permita determinar el área de colisión para cada objeto colisionable.

Para ello, dentro del sistema de referencia del objeto (es decir el objeto centrado en el eje de coordenadas) recorreremos los vértices que lo componen y tomamos el máximo y mínimo valor de las posiciones Z y X de cada punto 3D. Al final tenemos un rectángulo en el plano ZX determinado por esos cuatro valores calculados, tal y cómo se puede ver en la imagen:

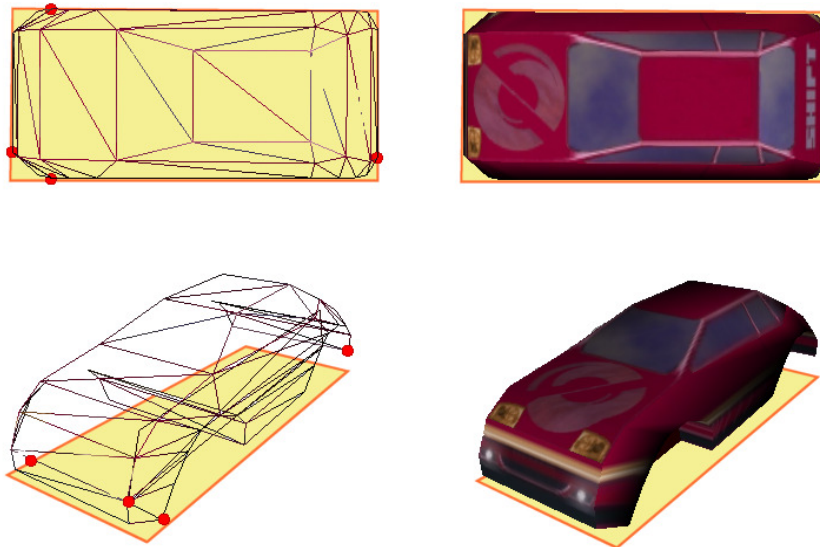


Imagen 3 : Cálculo del área de colisión en un objeto

Una vez determinados estos cuatro valores resulta entonces sencillo determinar las posiciones (Z, X) de cada uno de los cuatro vértices del área colisionable.

Este algoritmo permite encontrar el área de colisión de cada objeto marcado como colisionable. Este proceso sólo será necesario realizarlo una única vez para cada objeto, pudiendo realizarse durante la carga del mismo, al instanciar la clase collision.

De cara al proceso posterior, estos cuatro puntos los podremos traducir al SRU (sistema de referencia global o universal de la escena) o a cualquier SRO (sistema de referencia de un objeto concreto) mediante las matrices de transformación de los objetos (aplicando en un solo paso, rotaciones, traslaciones y escalados). De esta forma podemos tener traducido en la escena o comparado con otros objetos cualquier área de colisión existente.



2.3. Detección de la colisión

Una vez en el proceso iterativo del videojuego, tras obtener cada nueva escena es necesario determinar posibles colisiones entre objetos. Tras las consideraciones anteriores el problema se reduce a evaluar superposiciones de rectángulos situados en un plano bidimensional.

La detección se efectúa por cada combinación de pares de rectángulos colisionables. Es decir, se toma cada área y se comprueba superposición con el resto en escena. Como puede verse el proceso es realmente costoso, ya que por ejemplo:

- Con **2** objetos colisionables (A,B) hacemos **2** comprobaciones por iteración
 - Comprobamos **A** respecto **B**
 - Comprobamos **B** respecto **A**
- Con **3** objetos colisionables (A,B,C) hacemos **6** comprobaciones por iteración
 - Comprobamos **A** respecto **B**
 - Comprobamos **A** respecto **C**
 - Comprobamos **B** respecto **A**
 - Comprobamos **B** respecto **C**
 - Comprobamos **C** respecto **A**
 - Comprobamos **C** respecto **B**
- Con **n** objetos colisionables hacemos **n*(n-1)** comprobaciones por iteración

Es decir la detección de colisiones es de orden $O(n^2)$ respecto el número de objetos colisionables, por lo que hay que prestar especial atención de cara a la optimización en este punto.

Vamos a describir lo que supondría una comprobación genérica, por ejemplo al evaluar la posible colisión del **Objeto1** respecto el **Objeto2** (posteriormente sería necesario evaluar la posible colisión del Objeto2 respecto el Objeto1).

Primero tenemos que llevar los ocho puntos de colisión (dos rectángulos de colisión) a un mismo sistema de referencia, que en este caso será el del Objeto2, por lo que:

- Para el **Objeto2**, mantenemos en su SRO los cuatro vértices del área de colisión calculado durante la carga. Debido al algoritmo de cálculo anterior esta área de colisión se reduce a cuatro valores: X_{max} , X_{min} , Z_{max} y Z_{min} .
- Para el **Objeto1** hay que llevar sus cuatro vértices del área de colisión del **Objeto1** al SRO del Objeto2. Para ello hay que aplicar a cada vértice la matriz de transformación del Objeto1 y luego la matriz transformación inversa del Objeto2, resultando los valores: $(X_1, Z_1) \dots (X_4, Z_4)$

En este momento ya tendríamos los ocho puntos en el mismo sistema de referencia

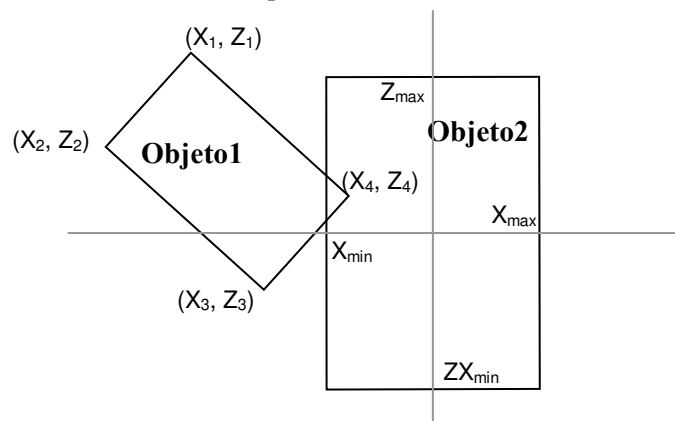


Imagen 4 : Evaluando colisión (situando puntos)



En un segundo paso, comprobamos si alguno de los cuatro vértices transformados del Objeto1 tienen sus componentes X y Z en los rangos $[X_{\max}, X_{\min}]$ y $[Z_{\max}, Z_{\min}]$ respectivamente. En ese caso (y sólo en ese caso) habrá colisión del Objeto1 respecto el Objeto2.

Como vemos en la imagen, en el ejemplo propuesto es el punto 4 el que cumple $(X_{\max} \geq X_4 \geq X_{\min})$ y además $(Z_{\max} \geq Z_4 \geq Z_{\min})$ por lo que hemos detectado la colisión.

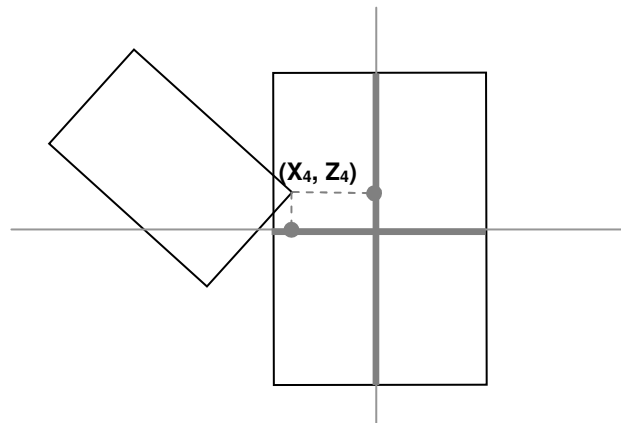


Imagen 5 : Evaluando colisión (evaluando puntos)

Conociendo esto, podemos reducir el estudio a la comprobación de colisión entre dos objetos (**A** y **B**), primero comprobamos **A** respecto **B**, y posteriormente **B** respecto **A**.

Este método es sencillo, comprensible y bastante eficiente, ya que por cada comprobación:

- Se halla la inversa de la matriz de **Objeto2**
- Se halla la matriz resultante de multiplicar la matriz de Objeto1 por la matriz de Objeto2, para aplicar luego en un solo paso la matriz a cada vértice de **Objeto1**
- Se aplica la matriz resultante sobre los cuatro vértices del área de colisión de **Objeto1**
- Se hacen $4 \times 4 = 16$ comparaciones de números reales

Hay que considerar que las multiplicaciones de matrices se harán a través de DirectX, por lo que esta carga de proceso la soportará en la medida de lo posible la tarjeta gráfica, descargando así de responsabilidad a la CPU.

En los siguientes apartados vamos a ver varios ejemplos sobre las diferentes posibilidades que existen.

2.3.1. Caso sin colisión entre objetos A y B

Supongamos dos objetos colisionables, A y B, cuyas áreas de colisión no se superponen, por lo la comprobación debería ser en este caso negativa.

Primero hacemos la comprobación de A respecto B, vemos que ningún vértice de A está dentro de los rangos X y Z que define el área de B:

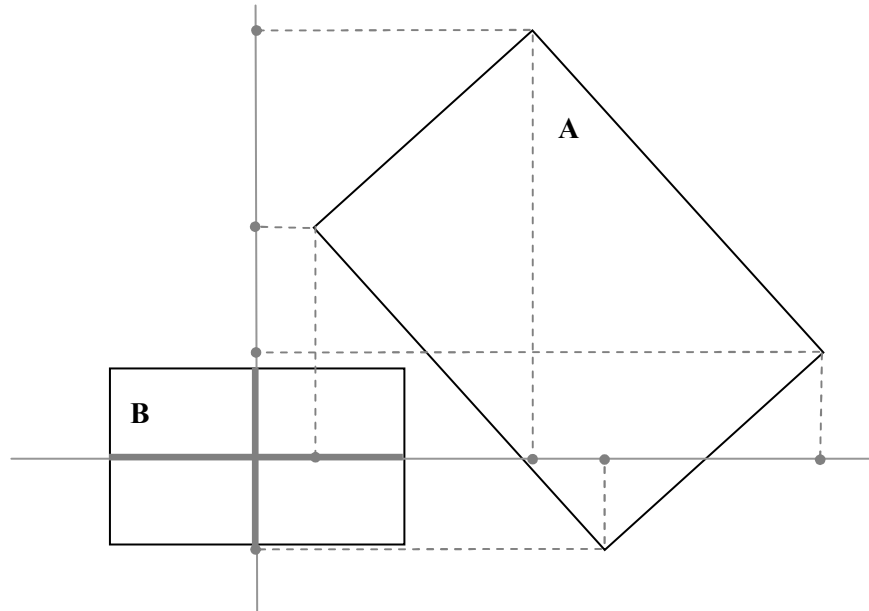


Imagen 6 : Detección A-B sin colisión

Después hacemos la comprobación de B respecto A, vemos que tampoco hay vértices de B dentro de los rangos X y Z que define el área de A:

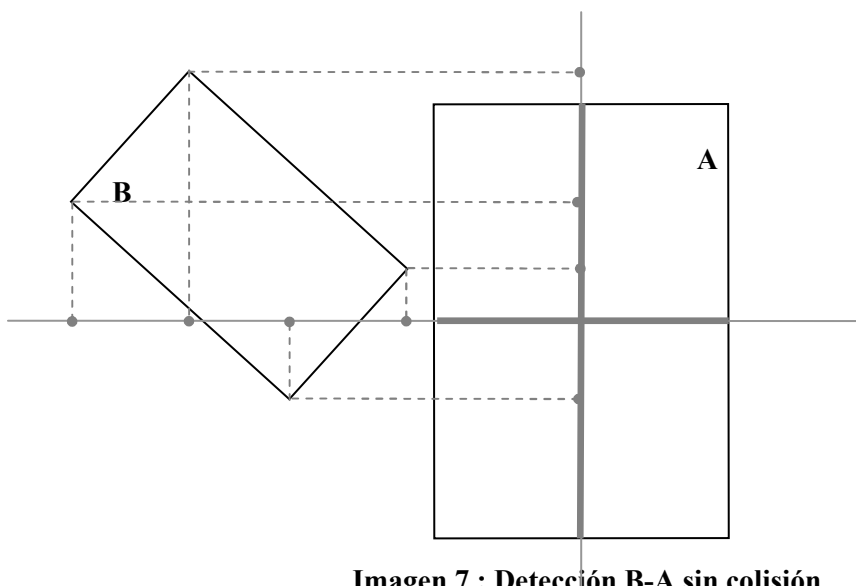


Imagen 7 : Detección B-A sin colisión



2.3.2. Caso con colisión entre objetos A y B

Ahora supongamos los mismos objetos colisionables, A y B, pero en este caso hay un vértice de B dentro del área colisionable de A, por lo que la comprobación deberá ser positiva.

Primero hacemos la comprobación de A respecto B, vemos que ningún vértice de A está dentro de los rangos X y Z que define el área de B:

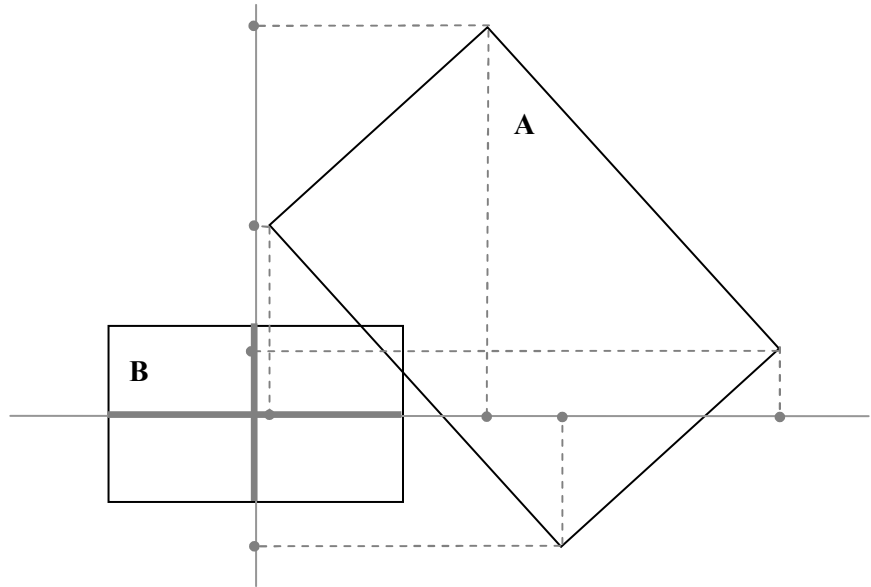


Imagen 8 : Detección A-B con colisión

Después hacemos la comprobación de B respecto A, vemos que el vértice de B señalado queda dentro de los rangos que establece el área de A, por lo que en este caso tenemos detectada la colisión:

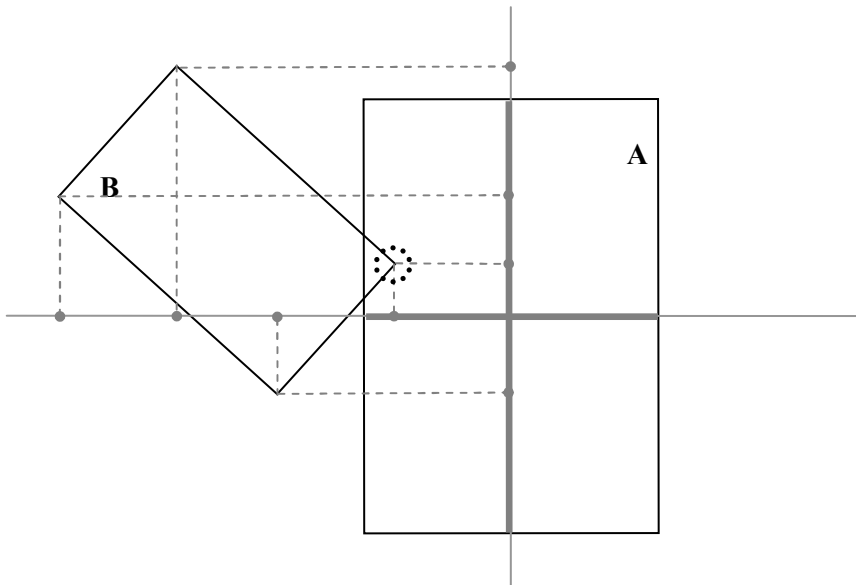


Imagen 9 : Detección B-A con colisión

2.4. Recolocación de los objetos

Este paso sólo se realiza si hemos detectado una colisión entre dos objetos.

Tras el paso anterior tenemos localizado un vértice dónde sabemos que hay superposición entre las áreas de colisión de los objetos. Es posible que haya más vértices con superposición, pero tomaremos este primer vértice como el único a tratar de cara a la colisión.

Tras encontrar una colisión entre objetos es necesario recolocarlos, de forma que ambos no queden “pegados” en las siguientes iteraciones del proceso. Esto es debido a que tras la colisión y el ajuste de los parámetros físicos, la siguiente iteración puede no resultar suficiente para “despegar” ambos objetos, por lo que volverá a haber colisión y volverán a invertir sus parámetros físicos, entrando en un bucle hasta que de algún modo eviten la colisión por otro de los extremos de los rectángulos. La siguiente imagen ilustra varias iteraciones donde ocurre este efecto “pegado” al no recolocar los objetos:

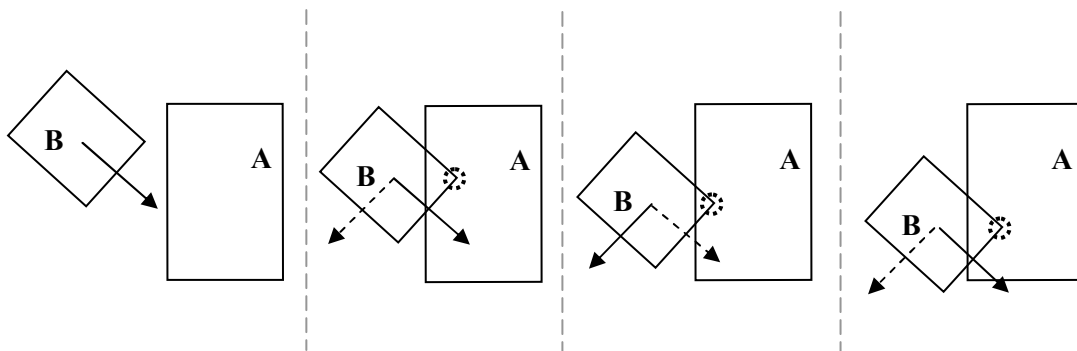


Imagen 10 : Colisiones sin recolocación

Por el contrario, si se recoloca el objeto B correctamente tras la primera colisión, este problema no se producirá:

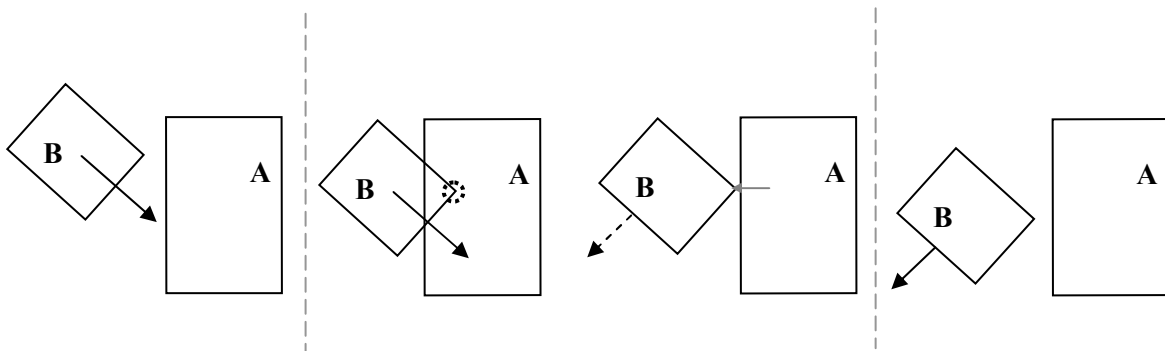


Imagen 11 : Colisiones con recolocación

Por lo tanto se ve necesaria la recolocación de los objetos, separándolos cierta distancia (ϵ) para evitar futuras colisiones inexistentes. Lo que si resulta importante es que el objeto a recolocar debe ser movable, por lo que habrá que tener en cuenta este detalle.



Para la recolocación partiremos del vértice dónde encontramos la colisión (X_p, Z_p) y hallaremos la menor distancia en X o Z para que ese vértice quede fuera de uno de los intervalos $[X_{\max}, X_{\min}]$ ó $[Z_{\max}, Z_{\min}]$.

Por lo tanto tomamos el mínimo entre los cuatro valores siguientes:

- $X_{\max} - X_p$
- $X_p - X_{\min}$
- $Z_{\max} - Z_p$
- $Z_p - Z_{\min}$

Una vez hallado, determinamos qué objeto vamos a recolocar, en función del tipo de cada objeto (movible o fijo):

- Si el objeto de referencia (el del SRO en el que estamos) es movible, entonces recolocaremos este objeto (ya que es menos costoso calcular su traslación).
- En otro caso recolocaremos el objeto que tiene el vértice de colisión encontrado.

Ahora hay que hallar la traslación que le vamos a hacer al objeto seleccionado, para ello:

- Si hay que mover el objeto de referencia:
 - Si el mínimo es $X_{\max} - X_p$ lo movemos en $X - (X_{\max} - X_p + \epsilon)$
 - Si el mínimo es $X_p - X_{\min}$ lo movemos en $X + (X_p - X_{\min} + \epsilon)$
 - Si el mínimo es $Z_{\max} - Z_p$ lo movemos en $Z - (Z_{\max} - Z_p + \epsilon)$
 - Si el mínimo es $Z_p - Z_{\min}$ lo movemos en $Z + (Z_p - Z_{\min} + \epsilon)$
- Si hay que mover el objeto con el vértice de colisión:
 - Si el mínimo es $X_{\max} - X_p$ lo movemos en $X + (X_{\max} - X_p + \epsilon)$
 - Si el mínimo es $X_p - X_{\min}$ lo movemos en $X - (X_p - X_{\min} + \epsilon)$
 - Si el mínimo es $Z_{\max} - Z_p$ lo movemos en $Z + (Z_{\max} - Z_p + \epsilon)$
 - Si el mínimo es $Z_p - Z_{\min}$ lo movemos en $Z - (Z_p - Z_{\min} + \epsilon)$

Dónde ϵ lo tomamos como un valor pequeño para asegurar que los objetos no vuelven a colisionar en la siguiente iteración.



Siguiendo con el ejemplo del apartado anterior, tenemos dos objetos colisionados: A objeto fijo y B objeto móvil. Al comprobar el objeto B respecto el objeto A encontramos la colisión en el vértice (X_p, Z_p) del objeto B del modo que se muestra en la imagen:

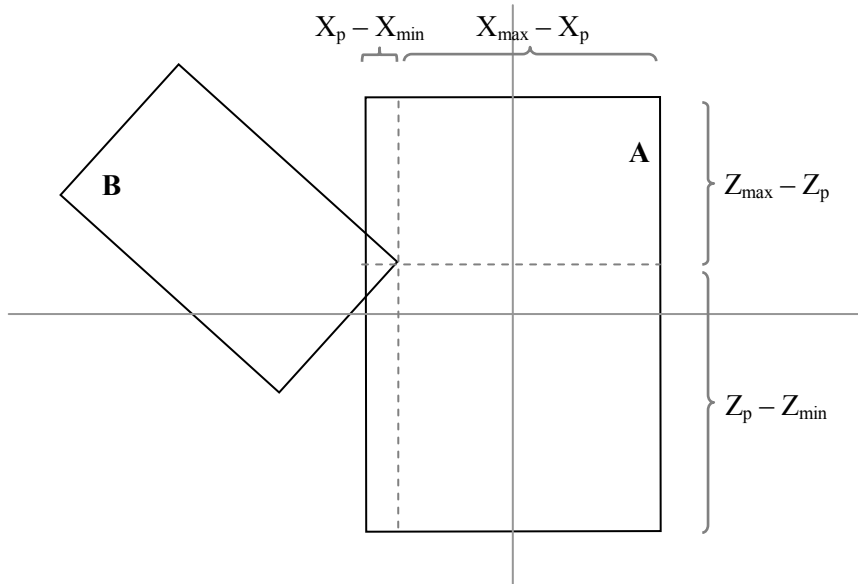


Imagen 12 : Ejemplo de recolocación

Si resulta que A es un objeto fijo, tenemos que mover el objeto B $[-(X_p - X_{min} + \epsilon)]$ en el eje X, de forma que quede desplazado ligeramente hacia la izquierda según la imagen. Este movimiento se puede aplicar fácilmente a la matriz de transformación del objeto B, multiplicando la traslación por la matriz inversa de transformación del objeto A.

En el caso contrario, si el objeto A hubiese sido móvil, habría que moverlo $[(X_p - X_{min} + \epsilon)]$ en el eje X, algo que se puede aplicar directamente sobre la matriz de transformación de A.

En caso que tras la recolocación vuelva a haber colisión en otro punto, se tratará en la siguiente iteración del proceso global.

2.5. Intercambio de parámetros físicos

Este es último paso del motor de colisiones, se realiza tras la recolocación y sólo en el caso de haber localizado una colisión de objetos.

Supone además la mayor complejidad, tanto de cara al diseño como de cara a la implementación y ejecución del algoritmo. Sin embargo, hay que considerar que aquí no es tan necesaria la eficiencia, ya en muy pocas iteraciones se van a encontrar colisiones de objetos, por lo que es un proceso que (al igual que el de recolocación) se va a dar pocas veces.

Tras el proceso de recolocación tenemos el esquema mostrado, dónde resulta sencillo encontrar el punto exacto de colisión (P_{colision}) obviando el ϵ adicional aplicado en el proceso de recolocación.

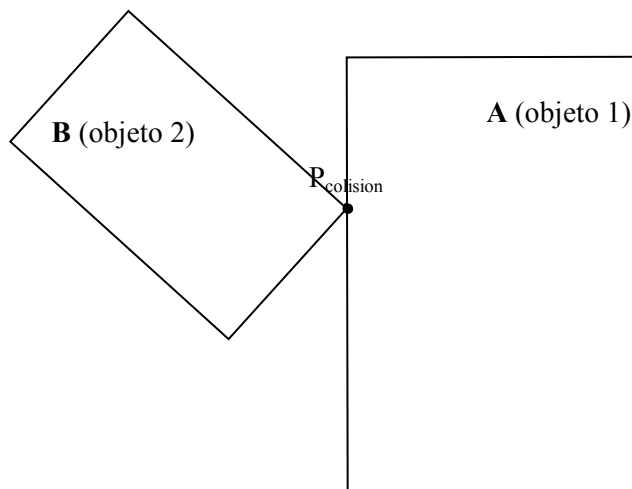


Imagen 13 : Esquema tras la recolocación

A partir de ahora, de cara al intercambio de parámetros físicos, diferenciamos dos tipos de choque:

- **Choque “Móvil-Móvil”** Caso básico y más complicado, cuando ambos objetos colisionados son móviles y tienen una instancia física asociada.
- **Choque “Móvil-Fijo” (con física)** Caso concreto del anterior, cuando uno de los objetos es fijo, pero sin embargo tiene una instancia física asociada (por ejemplo un aspa que gira o un muro que se desplaza).
- **Choque “Móvil-Fijo” (sin física)** Caso concreto del primero, cuando uno de los objetos es fijo y además no tiene una instancia física asociada.

En los dos siguientes apartados veremos el procedimiento de actualización de parámetros físicos para cada uno de los tipos de choque de los objetos.



2.5.1. Choque “Móvil-Móvil”

En este caso ambos objetos son móviles, por lo que el intercambio de parámetros se hará en ambos. Considerando el centro de cada objeto, situado en el punto (0,0) de cada SRO, podemos reducir el problema a la colisión de dos vectores (\vec{r}_1 y \vec{r}_2) del siguiente modo:

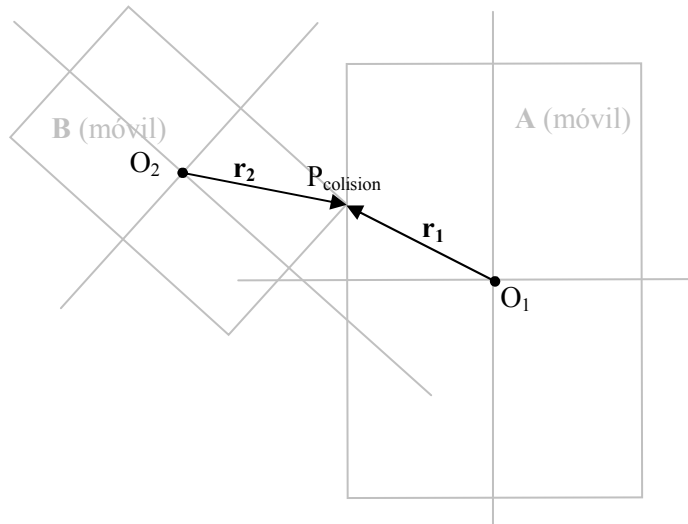


Imagen 14 : Simplificación al choque de dos vectores

Lógicamente, el cálculo de los dos vectores (\vec{r}_1 y \vec{r}_2) sería:

$$\vec{r}_1 = P_{colision} - O_1$$

$$\vec{r}_2 = P_{colision} - O_2$$

Y podemos normalizarlos en otros dos (\vec{d}_1 y \vec{d}_2) del siguiente modo:

$$\vec{d}_1 = \frac{\vec{r}_1}{\|\vec{r}_1\|}$$

$$\vec{d}_2 = \frac{\vec{r}_2}{\|\vec{r}_2\|}$$

Además de poder hallar los ortogonales a los normalizados (\vec{d}_1^\perp y \vec{d}_2^\perp) del siguiente modo:

$$\vec{d}_1 = (x_1, z_1) \Rightarrow \vec{d}_1^\perp = (-z_1, x_1)$$

$$\vec{d}_2 = (x_2, z_2) \Rightarrow \vec{d}_2^\perp = (-z_2, x_2)$$

Ahora vamos a identificar todos los parámetros de choque y físicos que intervienen en la colisión:

- \vec{v}_1 Vector de velocidad lineal (en unidades/segundo) del objeto 1 antes del choque
- \vec{v}_2 Vector de velocidad lineal (en unidades/segundo) del objeto 2 antes del choque
- ω_1 Velocidad angular (en radianes/segundo) del objeto 1 antes del choque
- ω_2 Velocidad angular (en radianes/segundo) del objeto 2 antes del choque
- m_1 Masa del objeto 1
- m_2 Masa del objeto 2



En un último cálculo, desglosamos los vectores velocidad \vec{v}_1 y \vec{v}_2 en dos componentes cada uno. Uno como proyección sobre \vec{d}_1 o \vec{d}_2 y otro vector proyección sobre el ortogonal \vec{d}_1^\perp o \vec{d}_2^\perp

$$\begin{aligned}\vec{v}_1' &= (\vec{v}_1 \cdot \vec{d}_1) \cdot \vec{d}_1 & \vec{v}_1'' &= (v_1 \cdot \vec{d}_1^\perp) \cdot d_1^\perp \\ \vec{v}_2' &= (\vec{v}_2 \cdot \vec{d}_2) \cdot \vec{d}_2 & \vec{v}_2'' &= (v_2 \cdot \vec{d}_2^\perp) \cdot d_2^\perp\end{aligned}$$

Todos estos parámetros físicos quedan representados en el siguiente diagrama:

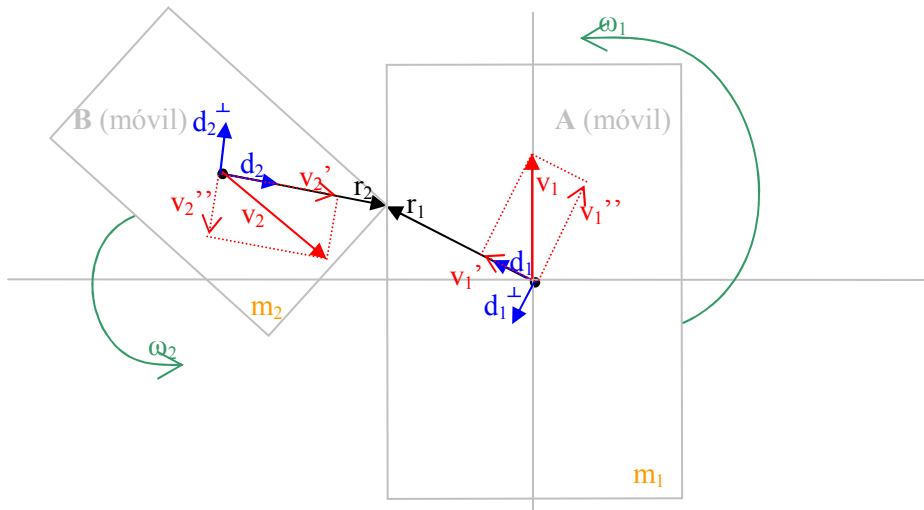


Imagen 15 : Parámetros físicos considerados

Ahora vamos a ver cómo se realiza el intercambio de cantidades de movimiento entre los diferentes parámetros físicos que intervienen.

Vamos a ver cómo desglosa su cantidad de movimiento lineal y angular el objeto A (1):

- La cantidad de movimiento lineal de A (\mathbf{v}_1) se descompone así:
 - Parte se mantendrá en A como energía lineal ($\mathbf{v}_1_v_1$)
 - Parte se cederá a B como energía lineal ($\mathbf{v}_1_v_2$)
 - Parte se cederá a B como energía rotativa ($\mathbf{v}_1_w_2$)
- La cantidad de movimiento angular de A (\mathbf{w}_1) se descompone así:
 - Parte se cederá a B como energía lineal ($\mathbf{w}_1_v_2$)
 - Parte se cederá a B como energía rotativa ($\mathbf{w}_1_w_2$)

Con el objeto B (2) ocurre justamente el caso simétrico:

- La energía lineal de B (\mathbf{v}_2) se descompone así:
 - Parte se mantendrá en A como energía lineal ($\mathbf{v}_2_v_2$)
 - Parte se cederá a B como energía lineal ($\mathbf{v}_2_v_1$)
 - Parte se cederá a B como energía rotativa ($\mathbf{v}_2_w_1$)
- La energía rotativa de B (\mathbf{w}_2) se descompone así:
 - Parte se cederá a B como energía lineal ($\mathbf{w}_2_v_1$)
 - Parte se cederá a B como energía rotativa ($\mathbf{w}_2_w_1$)



Sin embargo, no toda la cantidad de movimiento inicial se distribuirá en las componentes listadas, ya que existe cierta pérdida de energía en el choque. Esta pérdida se implementará mediante dos coeficientes, uno de pérdida de cantidad de movimiento lineal (μ_L) y otro de pérdida de cantidad de movimiento angular (μ_A).

Teniendo en cuenta lo anterior, la cantidad de movimiento lineal de A (1) se distribuye así:

$$\begin{aligned}\vec{v}_1 - v_1 &= \vec{v}_1'' \\ \vec{v}_1 - v_2 &= \mu_L \cdot \frac{m_1}{m_2} \cdot (\vec{v}_1' \cdot \vec{d}_2) \cdot \vec{d}_2 \\ v_1 - \omega_2 &= \mu_A \cdot \frac{m_1}{m_2 \cdot \|\vec{r}_2\|} \cdot (\vec{v}_1' \cdot \vec{d}_2^\perp)\end{aligned}$$

La cantidad de movimiento angular de A (1) se distribuye así:

$$\begin{aligned}\vec{\omega}_1 - v_2 &= \mu_L \cdot \frac{((m_1 \cdot \omega_1 \cdot \|\vec{r}_1\| \cdot \vec{d}_1^\perp) \cdot \vec{d}_2)}{m_2} \cdot \vec{d}_2 \\ \omega_1 - \omega_2 &= \mu_A \cdot \frac{((m_1 \cdot \omega_1 \cdot \|\vec{r}_1\| \cdot \vec{d}_1^\perp) \cdot \vec{d}_2^\perp)}{m_2 \cdot \|\vec{r}_1\|} \cdot (\vec{d}_2^\perp \cdot (-\vec{d}_1^\perp))\end{aligned}$$

La cantidad de movimiento lineal de B (2) se distribuye así:

$$\begin{aligned}\vec{v}_2 - v_2 &= \vec{v}_2'' \\ \vec{v}_2 - v_1 &= \mu_L \cdot \frac{m_2}{m_1} \cdot (\vec{v}_2' \cdot \vec{d}_1) \cdot \vec{d}_1 \\ v_2 - \omega_1 &= \mu_A \cdot \frac{m_2}{m_1 \cdot \|\vec{r}_1\|} \cdot (\vec{v}_2' \cdot \vec{d}_1^\perp)\end{aligned}$$

La cantidad de movimiento angular de B (2) se distribuye así:

$$\begin{aligned}\vec{\omega}_2 - v_1 &= \mu_L \cdot \frac{((m_2 \cdot \omega_2 \cdot \|\vec{r}_2\| \cdot \vec{d}_2^\perp) \cdot \vec{d}_1)}{m_1} \cdot \vec{d}_1 \\ \omega_2 - \omega_1 &= \mu_A \cdot \frac{((m_2 \cdot \omega_2 \cdot \|\vec{r}_2\| \cdot \vec{d}_2^\perp) \cdot \vec{d}_1^\perp)}{m_1 \cdot \|\vec{r}_2\|} \cdot (\vec{d}_1^\perp \cdot (-\vec{d}_2^\perp))\end{aligned}$$

Actualizamos los nuevos parámetros para el objeto A (1) y B (2):

$$\begin{aligned}\vec{v}_1 &= \vec{v}_1 - v_1 + \vec{v}_2 - v_1 + \vec{\omega}_2 - v_1 \\ \omega_1 &= v_2 - \omega_1 + \omega_2 - \omega_1 \\ \vec{v}_2 &= \vec{v}_2 - v_2 + \vec{v}_1 - v_2 + \vec{\omega}_1 - v_2 \\ \omega_2 &= v_1 - \omega_2 + \omega_1 - \omega_2\end{aligned}$$



2.5.2. Choque “Móvil-Fijo” (fijo con física)

Este caso se da cuando uno de los objetos es fijo, pero sin embargo tiene una instancia física asociada. Puede ser por ejemplo un aspa que rote con una velocidad angular fija, un muro que se desplace con una velocidad lineal fija, etc.

En el ejemplo que tenemos, imaginemos que el objeto A es fijo con velocidad lineal y angular fijas. En este caso consideraríamos los siguientes parámetros:

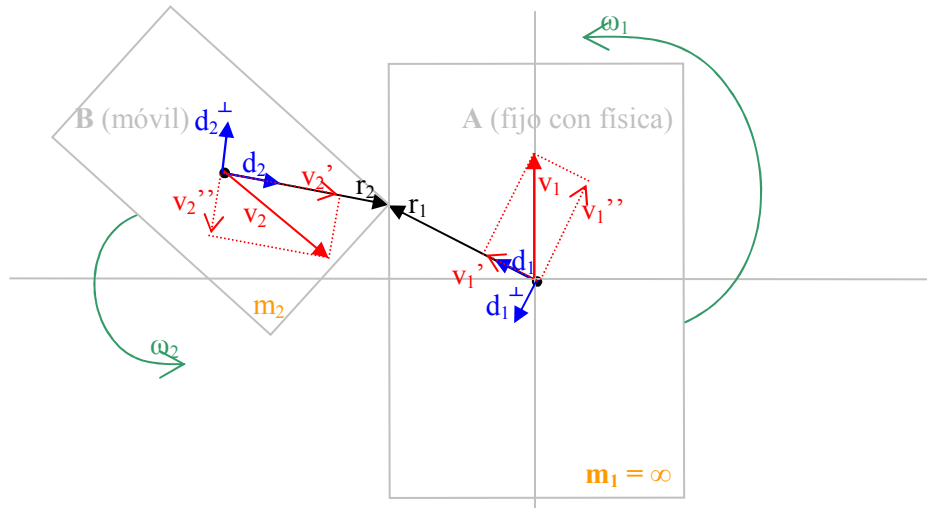


Imagen 16 : Parámetros físicos considerados

Realmente, este es un caso concreto del choque “Móvil-Móvil” anterior, donde el objeto A tiene una masa infinita (m_1) con velocidad (\vec{v}_1) y rotación (ω_1) específicas.

El objeto A por tanto entrará en el intercambio de energías (tiene cantidad de movimiento lineal y/o angular) pero no en la actualización de parámetros (es un objeto fijo).

Partiendo de las fórmulas anteriores y sustituyendo los valores correctos del objeto A, actualizaremos los parámetros del objeto B de modo similar al que hacíamos antes:

$$\vec{v}_1 = \vec{v}_1 - \vec{v}_1 + \vec{v}_2 - \vec{v}_1 + \omega_2 - \omega_1$$

$$\omega_1 = \omega_2 - \omega_1 + \omega_2 - \omega_1$$

2.5.3. Choque “Móvil-Fijo” (fijo sin física)

Este caso se da cuando uno de los objetos es fijo y además no tiene una instancia física asociada.

En el ejemplo anterior, imaginemos que el objeto A es fijo, por lo que el estudio se reduce únicamente al objeto B. En este caso sólo consideraríamos los siguientes parámetros:

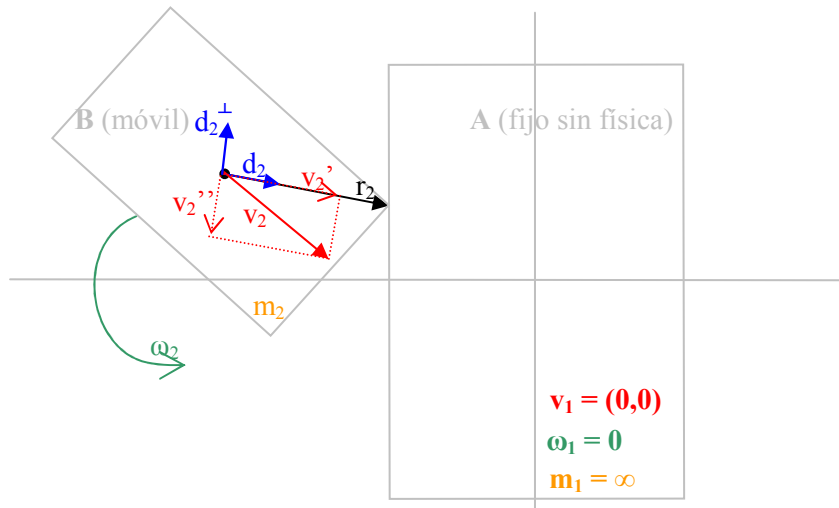


Imagen 17 : Parámetros físicos considerados

Realmente, este es un caso concreto del choque “Movil-Móvil” inicial, dónde el objeto A tiene una masa infinita (m_1), pero no tiene velocidad (\vec{v}_1) ni rotación (ω_1).

El objeto A por tanto no entrará en el intercambio de energías (no tiene cantidad de movimiento lineal ni angular) ni en la actualización de parámetros (es un objeto fijo).

Partiendo de las fórmulas anteriores y sustituyendo los valores correctos del objeto A, actualizaremos los parámetros del objeto B del siguiente modo:

$$\vec{v}_2 = \vec{v}_2''$$

$$\omega_2 = -\omega_2$$

Alberto Carlos del Blanco Maraña
Madrid 28 de junio de 2007