Físicas de Videojuegos

Colisiones

Introducción

¿De Qué vamos a hablar?

- Colisiones
- Definición de colisión y aplicación en videojuegos
- Colisiones 2D vs 3D
- Tipos de colisiones
  - Punto vs Rectángulo
  - Punto vs Círculo
  - Rectángulo vs Rectángulo
  - Círculo vs Círculo
  - Círculo vs Rectángulo
  - Cubo vs Cubo
- Ejemplo en C con Raylib de detección de colisiones

Colisiones en Videojuegos

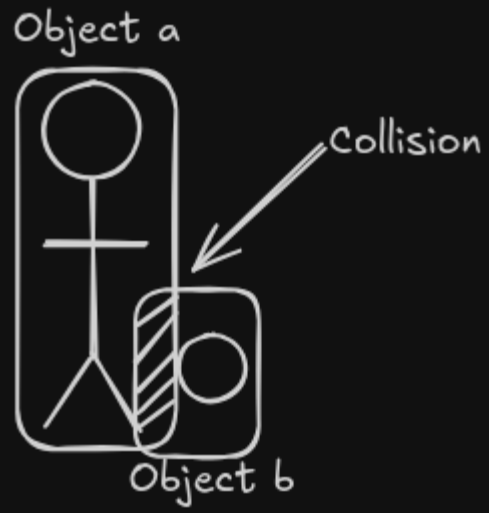
En un videojuego, una colisión se refiere al evento en el que dos o más objetos del juego entran en contacto o se superponen entre sí. Las colisiones son fundamentales para la interacción entre los objetos del juego y pueden afectar la jugabilidad, la física y la lógica del juego.

Es importante distinguir entre la detección de colisiones y la respuesta a las colisiones. La detección de colisiones implica identificar cuándo y dónde ocurre una colisión, mientras que la respuesta a las colisiones implica determinar qué sucede después de que se detecta una colisión, como rebotar, detenerse o aplicar daño.

Definición de Colisión y Aplicación en Videojuegos

Una colisión en videojuegos se define como el evento que ocurre cuando dos o más objetos del juego entran en contacto o se superponen entre sí. Este evento es crucial para la interacción entre los objetos del juego y puede influir en la jugabilidad, la física y la lógica del juego.

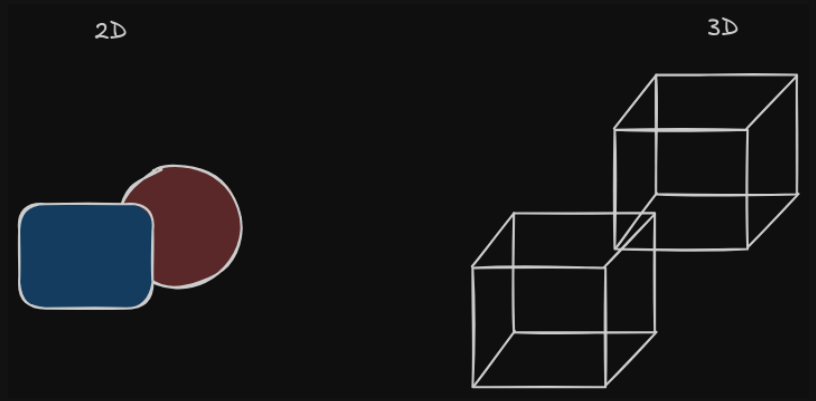
Normalmente cada objeto del juego tiene una "caja de colisión" o "hitbox" que define su área de interacción. Cuando las cajas de colisión de dos objetos se superponen, se considera que ha ocurrido una colisión.



Colisiones 2D vs 3D

Las colisiones en 2D y 3D difieren principalmente en la dimensionalidad de los objetos y las técnicas utilizadas para detectar y manejar las colisiones.

Mientras que las colisiones 2D se manejan en un plano bidimensional utilizando formas como rectángulos, círculos y polígonos, las colisiones 3D se manejan en un espacio tridimensional utilizando formas como cajas, esferas y mallas complejas.



Tipos de Colisiones

Existen varios tipos de colisiones comunes en videojuegos, cada una con sus propias técnicas de detección:

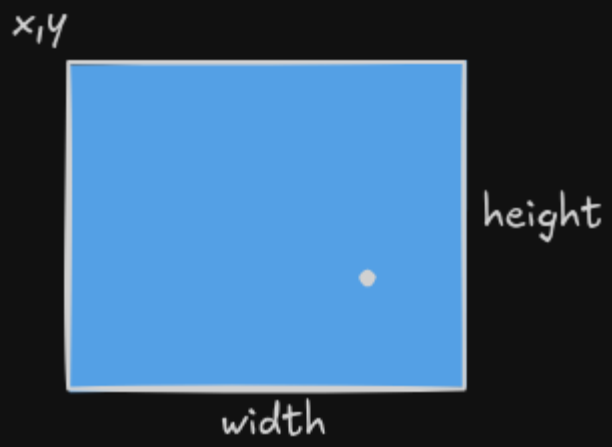
- Punto vs Rectángulo
- Punto vs Círculo
- Rectángulo vs Rectángulo
- Círculo vs Círculo
- Círculo vs Rectángulo
- Cubo vs Cubo

Punto vs Rectángulo

Se detecta si un punto (x, y) está dentro de los límites de un rectángulo definido por su posición (*rect.X*, *rect.Y*), ancho (*rect.Width*) y alto (*rect.Height*).

Puede calcularse con la siguiente fórmula:

$$(rect.x \leq point.x \leq rect.x + rect.width) AND (rect.y \leq point.y \leq rect.y + rect.height)$$



Punto vs Círculo

Se detecta si un punto (x, y) está dentro de un círculo definido por su centro (*circle.x*, *circle.y*) y su radio (*circle.radius*).

Puede calcularse con la siguiente fórmula:

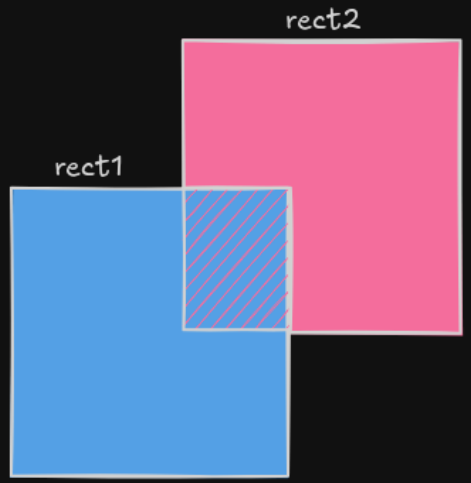
$$(point.x - circle.x)^2 + (point.y - circle.y)^2 < circle.radius^2$$

Rectángulo vs Rectángulo

Se detecta si dos rectángulos se superponen. Cada rectángulo está definido por su posición (*rect1.X*, *rect1.Y*) y sus dimensiones (*rect1.Width*, *rect1.Height*).

Puede calcularse con la siguiente fórmula:

$$(rect1.x < rect2.x + rect2.width) AND (rect1.x + rect1.width > rect2.x)$$
$$AND (rect1.y < rect2.y + rect2.height) AND (rect1.y + rect1.height > rect2.y)$$

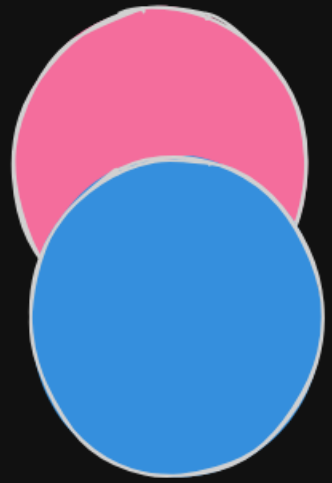


Círculo vs Círculo

Se detecta si dos círculos se superponen. Cada círculo está definido por su centro (*circle1.x*, *circle1.y*) y su radio (*circle1.radius*).

Puede calcularse con la siguiente fórmula:

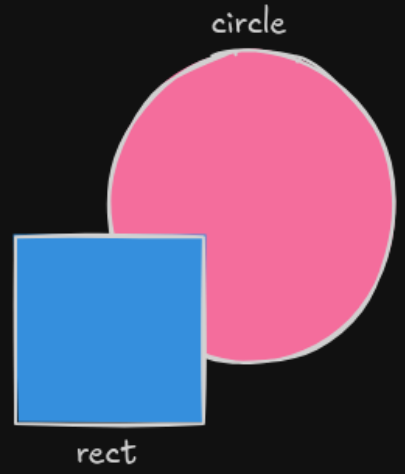
$$(circle1.x - circle2.x)^2 + (circle1.y - circle2.y)^2 < (circle1.radius + circle2.radius)^2$$



Círculo vs Rectángulo

Se detecta si un círculo y un rectángulo se superponen. El círculo está definido por su centro (*circle.x*, *circle.y*) y su radio (*circle.radius*), mientras que el rectángulo está definido por su posición (*rect.x*, *rect.y*) y sus dimensiones (*rect.width*, *rect.height*). Puede calcularse con la siguiente fórmula:

$$closestX = clamp(circle.x, rect.x, rect.x + rect.width)$$
$$closestY = clamp(circle.y, rect.y, rect.y + rect.height)$$
$$(circle.x - closestX)^2 + (circle.y - closestY)^2 < circle.radius^2$$

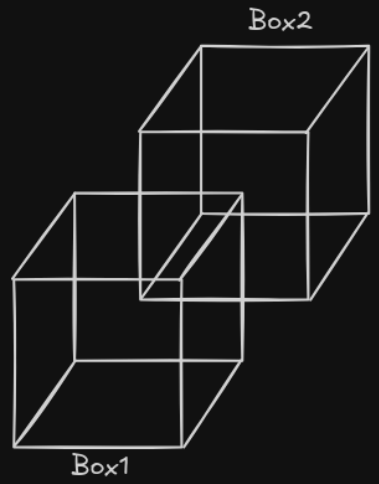


Cubo vs Cubo

Se detecta si dos cubos se superponen. Cada cubo está definido por su posición (*box1.x*, *box1.y*, *box1.z*) y sus dimensiones (*box1.width*, *box1.height*, *box1.depth*).

Puede calcularse con la siguiente fórmula:

$$(box1.x < box2.x + box2.width) AND (box1.x + box1.width > box2.x)$$
$$AND (box1.y < box2.y + box2.height) AND (box1.y + box1.height > box2.y)$$
$$AND (box1.z < box2.z + box2.depth) AND (box1.z + box1.depth > box2.z)$$



Ejemplo en C con Raylib de Detección de Colisiones

A continuación dejamos un enlace para ver un ejemplo práctico de cómo implementar la detección de colisiones en C utilizando la biblioteca Raylib:

Ejemplo de Colisiones con Raylib

Conclusiones

- Las colisiones son fundamentales para la interacción en videojuegos.
- Existen diferentes tipos de colisiones, cada una con sus propias técnicas de detección.
- La detección de colisiones es crucial para la jugabilidad y la física del juego.
- Implementar colisiones correctamente mejora la experiencia del jugador.

Referencias

- Desarrollo Homebrew para 16 bits - V. Suárez
- PlutieDev - Collisions in 2D Games
- "Real-Time Collision Detection" de Christer Ericson
- Documentación de Raylib: https://www.raylib.com/
- Artículos y tutoriales sobre físicas en videojuegos en Gamasutra y GameDev.net