Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación


# Laboratorio 5

Videojuegos, Programación, Físicas

*Detección de Colisiones*

## 1. Pre-Laboratorio

- Investigar los siguientes conceptos:
  1. Arquitectura Sistemas-Componente-Entidad (Entity Component Systems o ECS), ver como esta puede ser utilizada para el desarrollo de videojuegos en general.
  2. Octrees y quadtrees, arboles BSP y sus usos.
- Explique al menos dos casos para los que se podría utilizar detección de colisiones en un videojuegos.
- Investigue para su herramienta de trabajo escogida como incorporar colisiones, si este ya posee detección de colisiones o debe ser incorporado a través de bibliotecas.

## 2. Definición

Consiste en el problema computacional de detectar la intersection entre dos o mas objetos. Ademas de detectar si dos objetos interceptan un sistema de colisión de objetos puede reportar tiempo y punto de impacto [3].

En una simulación física de colisiones se busca imitar la colisión entre uno o mas objetos de la forma mas precisa posible siguiendo las características físicas de los objetos en cuestión y las distintas propiedades de los materiales que los componen utilizando cuerpos rígidos (rigid-body physics) o cuerpos blandos (*soft-body physics*) [4, p. 340].

Mientras en un videojuego se busca simular colisiones de una forma aceptable, en tiempo real y robusta.

# 3. Intersecciones Objeto / Objeto

Real Time Rendering: Intersections Object / Object `http://www.realtimerendering.com/intersections.html`. Este sitio presenta referencias a soluciones de intersecciones objeto / objeto, la mayoría de estas soluciones se encuentran en la serie *Graphics Gems*. Muchos frameworks y bibliotecas para el desarrollo de videojuegos (o software general) ya tienen estas soluciones implementadas e incluso optimizadas [1].

# 4. Tecnicas y Optimizaciones Comunes

## 4.1. Partición Espacial (Spatial Partitioning)

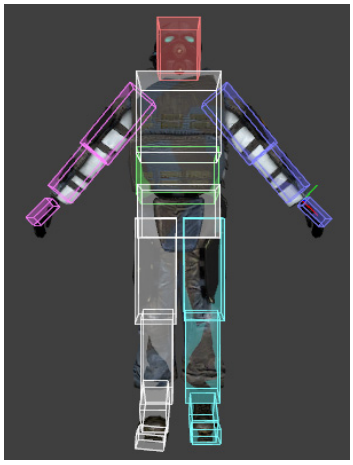

Figura 1: Hitboxes de un *Combine* de *Half Life 2* [6] [7].

Existen una variedad de algoritmos utilizados para dividir un espacio virtual entre los mas comunes están los octrees, quadtrees y arboles BSP (binary space partitioning). Esta partición espacial se realiza para evitar determinar intersecciones entre objetos que no pertenecen a una misma área determinada por el algoritmo de partición espacial, de esta forma se disminuye el numero de comparaciones aumentando el desempeño o *performance* [5].

## 4.2. Bounding Boxes o Bounding Volumes

Usualmente rectángulos en 2D o cubos en 3D pero también es posible usar otras figuras como esferas, círculos, elipses, paralelogramos, envolventes convexas entre otras. Son usadas como un inicial test de colisión de tal forma que solo se determina la intersección con objetos dentro del bounding volume [2].

### 4.3. Hit Boxes

Al igual que un Bounding Box estos suelen ser cubos o rectángulos pero también pueden ser otras figuras, el uso de elipses o elipsoides es común. Para objetos animados es usual que los hit boxes estén enlazados a las partes que se mueven de tal forma que los hit boxes también se muevan y roten en correlación con el personaje u objeto al que pertenecen. Los hit boxes son utilizados para detectar colisiones unidireccionales como un objeto en intersección con un bala u otro proyectil. Son poco precisos para colisiones con *feedback* como chocar contra una pared ya que la posición y orientación de los hit boxes cambia constantemente, este tipo de colisiones son mejor manejadas por bounding boxes alineadas a un eje [6].

## 5. Actividad

En esta actividad debe incorporar distintas acciones y eventos que hagan uso de detección de colisiones.

- Debe incorporar detección de colisiones para las siguientes condiciones.
    1. Jugador principal contra actores con movimiento.
    2. Jugador principal contra objetos inamovibles.
    3. Jugador principal contra objetos en movimiento.
    4. Actor con movimiento contra objetos inamovibles.
    5. Actor con movimiento contra objetos en movimiento.
    6. Actor con movimiento contra actor con movimiento.
- Debe crear un sistema de historial de eventos donde cada colisión e información de la misma es guardada. Considere como datos necesarios, tiempo de colisión e información de ambos objetos en colisión. Debe agregar mas datos del evento de colisión que vea de utilidad para su mecánica y diseño de juego.

## Referencias


[1] Brown, E. Object/Object Intersection. http://www.realtimerendering.com/intersections.html, 2011.

[2] Caldwell, D. R. Unlocking the Mysteries of the Bounding Box. http://www.stonybrook.edu/libmap/coordinates/seriesa/no2/a2.htm, 2005.

[3] Ericson, C. *Real-time Collision Detection*. Elsevier, 2005.

[4] Morgan McGuire, O. C. J. *Creating Games*. A K Peters, 2008.

[5] Nystrom, B. Spatial Partition - Game Programming Patterns / Optimization Patterns. http://www.stonybrook.edu/libmap/coordinates/seriesa/no2/a2.htm, 2014.

[6] Valve. Developers Community Wiki: Hitbox. https://developer.valvesoftware.com/wiki/Hitbox.

[7] Valve Software. Half Life 2. http://store.steampowered.com/app/220/?snr=1_5_9__300, 2004.