Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación


# Laboratorio 2

Vídeo Juegos, Programación, Diseño

*Cámaras Virtuales*

## 1. Pre-Laboratorio

- Investigar:
  1. Vídeo juegos en primera persona.
  2. Vídeo juegos en tercera persona.
  3. Side-scrollers.
  4. Perspectiva.
  5. Proyección Ortogonal.
  6. Generos de video juegos.
- Investigar segun su framework, biblioteca o motor de juego escogido.
  1. Implementacion de camaras en el juego.
  2. Manejo de escenas.
  3. Scripting o programación de la logica de juego.

## 2. Definición

En los vídeo juegos se utilizan cámaras virtuales para mostrar un mundo 3D o 2D en algún dispositivo visual. Un sistema virtual de cámaras se encarga de controlar una o mas cámaras en un escenario de juego. Usualmente la cámara en muchos frameworks y bibliotecas para desarrollo de vídeo juegos no es mas que otro objeto en escena [10].

# 3. Conceptos

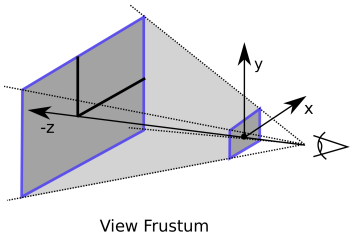


Figura 1: *Viewing Frustum* de una cámara virtual donde se muestra el far y near clipping plane [6].

Algunos conceptos necesarios para entender el uso de cámaras en la gran mayoría de los frameworks, bibliotecas y motores para el desarrollo de video juegos [9].

## 3.1. Tipos de Proyección

Existen dos tipos proyección en la mayoría de los frameworks y bibliotecas para el desarrollo de vídeo juegos estos son proyección perspectiva y proyección ortogonal.

### 3.1.1. Perspectiva

Es la forma natural en que el ojo humano percibe una escena, en la proyección perspectiva los objetos distantes se ven mas pequeños que los objetos cercanos dando profundidad a distintos objetos en una escena. Este tipo de proyección es utilizada usualmente en juegos 3D.

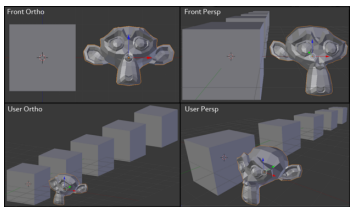


Figura 2: Proyección ortogonal y perspectiva.

### 3.1.2. Ortogonal

En una proyección perspectiva un objeto lejano es mas pequeño que un objeto cercano, en proyección ortogonal se ignora este efecto, eliminando la profundidad de escena. Este tipo de perspectiva es utilizada en muchos juegos 2D.

## 3.2. Field Of View (FoV) o Campo de Vista

Es el ancho del angulo de vista de la cámara, indica la extension de lo puede ver la cámara en cualquier momento, usualmente se mide en ángulos, este angulo puede ser el field of view vertical o el field of view horizontal dependiendo del framework, biblioteca o motor de juego usado (ver 3) [1].

## 3.3. Clipping Planes o Planos de Clipping

En computación se manejan términos discretos por lo tanto una cámara no puede ver hacia el infinito, para esto están el far y near clipping planes los cuales indican donde termina el renderizado de escena y donde empieza según la posición de la cámara respectivamente (ver 1 y 3).

### 3.3.1. Near Clipping Plane o Plano de Clipping Cercano

Es donde empieza a dibujarse los objetos de escena en display, los objetos antes de este punto son ignorados por el motor gráfico.

### 3.3.2. Far Clipping Plane o Plano de Clipping Lejano

Es donde termina de dibujarse los objetos de escena en display, los objetos mas lejanos a este punto son ignorados por el motor gráfico.

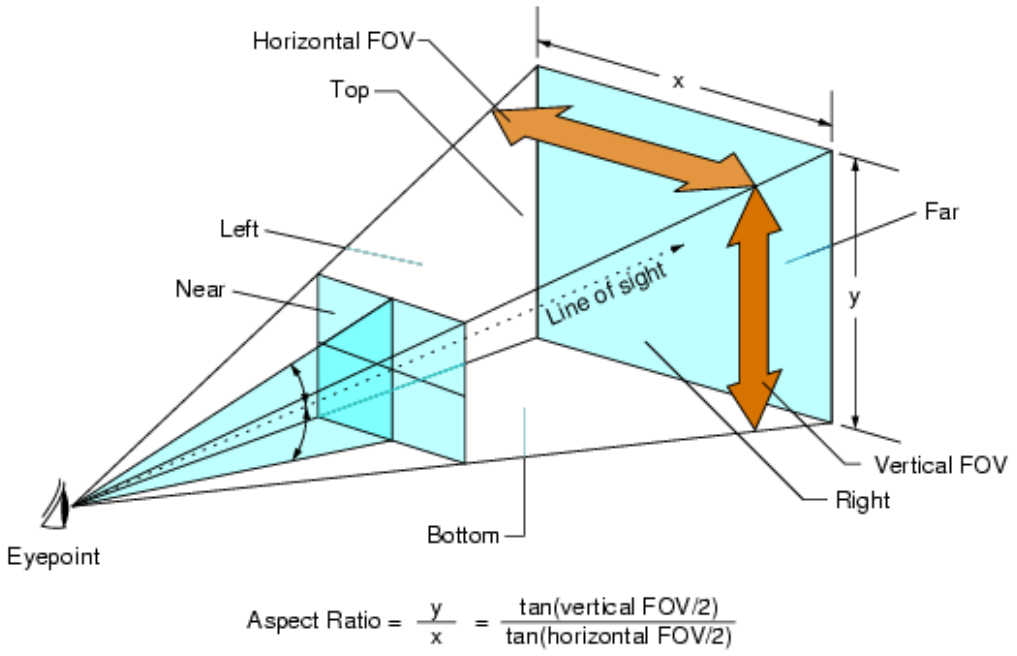


Figura 3: Parametros de una Camara Virtual

# 4. Tipos de Sistemas de Cámara en Vídeo Juegos

Existen principalmente tres tipos de sistemas de cámara en vídeo juegos. En un sistema *fixed* la cámara no cambia sus parámetros originales, un sistema *tracking* sigue algún objeto en juego, y un sistema *interactive* la cámara es parcialmente autónoma y cambia sus parámetros según distintas situaciones. Para implementar distintos sistemas de cámara los desarrolladores de vídeo juegos usan técnicas como programación con restricciones o inteligencia artificial.

## 4.1. Estáticas (*Fixed*)


Figura 4: Ejemplo de cámaras estáticas en *Resident Evil 2* [5] [2].

En este tipo de cámara las propiedades de la cámara como su posición, orientación y campo visual (*field of view*) son colocadas durante el desarrollo del juego y estas no cambian durante el *gameplay*. Algunos ejemplos de juegos con este tipo de camara son los primeros Resident Evil, y Alone In The Dark, usualmente es utilizada para crear tension [3][5].

## 4.2. Seguidoras (*Tracking*)

Este tipo de cámara sigue a algún objeto en el juego usualmente el personaje principal u otro objeto de considerable importancia. Este sistema presenta varios problemas sobretodo en ambientes tridimensionales y tercera persona donde la cámara podría quedar detrás de una objeto que ocluye totalmente la vista o no deja ver algún objeto de interés al jugador [10]. Su uso es muy común en los primeros juegos 3D en tercera persona como *Crash Bandicoot* [7] o *Tomb Raider* [4, p. 39], los juegos primera persona también utilizan una cámara seguidora, a diferencia de en tercera persona donde la cámara esta detrás del personaje en un juego primera persona la cámara esta como visión del personaje principal, en los juegos 2D esta cámara esta presente en todos los juegos tipo side-scroller.

### 4.3. Interactivas (*Interactive*)

Figura 5: En *Super Mario 64* la cámara rota de forma inteligente para mostrar el camino [8].

Las cámaras interactivas son cámaras que cambian su posición, campo visual u orientación según distintas situaciones, lugares o intereses del juego y sus desarrolladores, usualmente estas cámaras poseen alguna forma de inteligencia artificial. En su mayoría las cámaras interactivas son cámaras tracking mejoradas, estas siguen al personaje como lo hace una cámara tracking pero su posición y orientación puede cambiar si estas se ven obstruidas por algún objeto o cambian sus parámetros para mostrar objetos de interés claramente evitando de esta forma las desventajas principales de una cámara tracking. Algunos ejemplos de juegos con este tipo de camara son *Super Mario 64*, *Super Mario Sunshine*, *The Legend of Zelda: The Wind Waker.*

## 5. Actividad

- Durante esta actividad se creara la base del proyecto y se implementara inicialmente la cámara según el diseño de su juego. Los objetivos de la actividad son:

  1. Crear una pequeña escena con *placeholders* [1].
  2. Implementar la cámara principal de juego según el diseño del juego.
  3. Debe crear una estructura de datos (ejemplo una clase) que defina a los actores principales en su mundo de juego, incluyendo el jugador principal. Esto depende directamente del diseño del juego.
  4. Debe asociar esta estructura a un *placeholder* que representa cada actor, incluyendo el jugador principal.
  5. Agrege en la lógica de juego (esto se realiza usualmente a través de scripts u otro medio que provea las herramientas que eligió) comportamientos a uno o

[1] Un *placeholder* no es mas que un procurador de un objeto, de manera que el *placeholder* es remplazado en un futuro por el objeto real.

varios actores en escena, este comportamiento debe ser observable por la camara (ejemplo moverse constantemente)

## Referencias


[1] Feng Zhu School of Design: FoV in Video Games. http://artsygamer.com/fov-in-games/, 2014.

[2] Capcom. Resident Evil 2. http://www.giantbomb.com/resident-evil-2/3030-9418/, 1998.

[3] Casamassina, M. Resident Evil Review. http://uk.ign.com/articles/2002/04/26/resident-evil-3?page=3, 2002.

[4] Gates, B. *Game Design (2nd Edition)*. Thomson Course Technology, 2004.

[5] GiantBomb Community. Fixed Camera: Concept. http://www.giantbomb.com/fixed-camera/3015-1715/, 2014.

[6] Microsoft. What Is a View Frustum? https://msdn.microsoft.com/en-us/library/ff634570.aspx, 2014.

[7] Naughty Dog Software. Crash Bandicoot. http://www.ign.com/games/crash-bandicoot/ps-603, 1996.

[8] Nintendo. Super Mario 64. http://www.ign.com/games/super-mario-64/n64-606, 1996.

[9] Unity Technologies. Unity Manual: Camera. http://docs.unity3d.com/Manual/class-Camera.html, 2014.

[10] Rollings Andrew, E. A. *Fundamentals of Game Design*. Prentice Hall, 2006.