

# Videogame engine

Videogames Technology  
Asignatura transversal

Departamento de Automática

## Objectives

- Understand the need of videogame engines
- Introduce the videogame main loop
- Extend basic vocabulary about videogames

## Bibliography

1. Desarrollo de Videojuegos, Arquitectura del Motor de Vieojuegos. UCLM.

# Table of Contents

## 1. Game concept

- Game concept
- Game architecture overview
- Main loop
- Frame rate

## 2. Videogame engine

- History
- Definition
- Commercial engines

## 3. Videogames genres

- Third person videogames
- Fighting videogames
- Racing videogames
- Strategy
- Massively Multiplayer Online Game (MMOG)

# Game concept

## Game components

A videogame involves one or more **entities** to get a **goal** in a limited **environment**

- **Environment:** 2D or 3D world - simple or complex
- **Entities:** Hero, soccer teams, zombies, armies
- There are **rules** for the iteration entities-environment (**physics**)
- Some key terms in videogames
  - **Game state:** Data about entities and environment
  - **Game mechanics:** Rules to interact with the game
  - **NPC:** Non-playable character

# Game concept

## Game architecture overview

### Several subsystems

- Collision detection and handling
- Physics engine
- Graphics engine
- AI engine



(Source)



(Source)



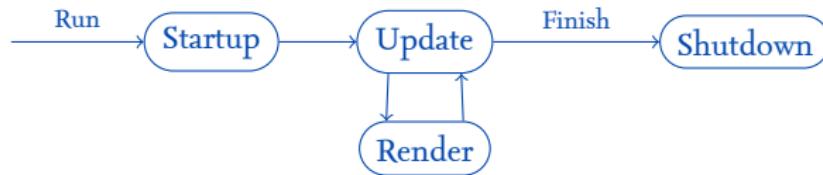
(Source)

# Game concept

## Main loop (I)

Any game is based on a **main loop** (or game loop)

1. Game state update
2. Game rendering
3. Wait
4. Go to 1



Not all subsystems must be run in each iteration!

# Game concept

## Main loop (II): Main game loop with Slick2D

```
class SlickLoop extends BasicGame {  
    @Override  
    public void init(GameContainer container) throws SlickException {  
        // Init game  
    }  
  
    @Override  
    public void update(GameContainer container, int delta) throws  
        SlickException {  
        // Update world  
    }  
  
    @Override  
    public void render(GameContainer container, Graphics g) throws  
        SlickException {  
        // Render world  
    }  
  
    public static void main(String arg[]) {  
        // Launch game  
        try {  
            new SlickLoop();  
        } catch (SlickException e) { }  
    }  
}
```

# Game concept

## Main loop (III): Main game loop with Arcade

```
import arcade

SCREEN_WIDTH = 800
SCREEN_HEIGHT = 600

class MyGame(arcade.Window):
    def __init__(self):
        super().__init__(SCREEN_WIDTH, SCREEN_HEIGHT, "Arcade")

    def setup(self):
        pass

    def on_draw(self):
        arcade.start_render()

    def on_update(self, delta_time):
        pass

def main():
    window = MyGame()
    window.setup()
    arcade.run()

main()
```

(Source code)

# Game concept

## Frame rate (I)

- The main loop must be executed with a certain frequency
  - High enough to generate an immersive (realistic) experience
  - Low enough not to waste computational resources
- The main loop frequency determines the **Frame rate (fps)**
  - The higher fps, the higher realism (and computation)
  - 30 fps is considered as sufficient
  - Nowdays, moving to 60 fps on new generation consoles
- If the hardware is not able to maintain the given fps, the game suffers slowdown

(Video 1) (Video 2) (Video 3)

# Game concept

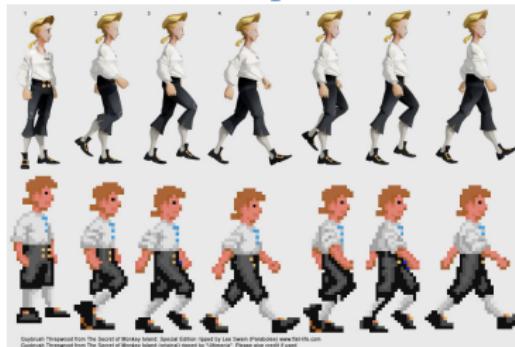
## Frame rate (II)

- A trade-off is needed: fps versus game realism
- Complex computational models reduce fps
  - Graphical components (mostly)
  - Physics
  - AI
- Highly realistic models of the real world is not practical
- Videogames use simplified mathematical models
  - Simplified physics
  - Simplified NPC interaction
- Tricks to reduce computational needs
  - Different strategies on 2D/3D games
  - Videogame programming ⇒ ``Tricks programming''

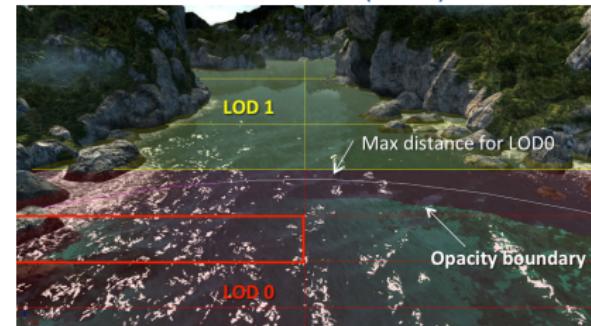
# Game concept

## Frame rate: Tricks (II)

2D - Sprites

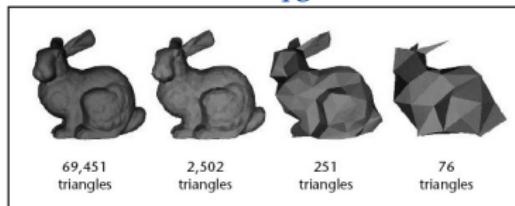


Level of Detail (LOD)



(Source)

3D - Polygons



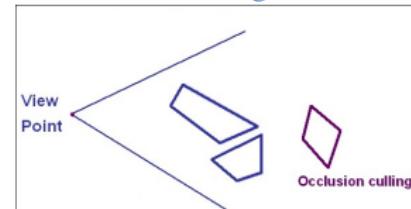
# Game concept

## Frame rate: Tricks (II)

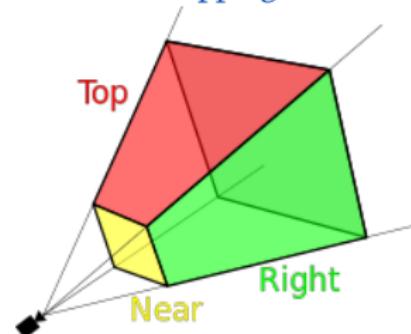


(Source)

### Culling



### Clipping



# Videogame engine

## History (I)

The first videogame engine emerged in the mid-nineties with **Doom**

- Doom is a classic first-person shooter (Video)
- Created by id Software under the supervision of John Carmack
- (History of Id Software)



# Videogame engine

## History (II)

- Doom was designed using a reuse-oriented architecture
- The architecture was based on several modules
  - Rendering
  - Collision detection
  - Audio
  - Artistic components (i.e., levels)
  - Rule system that controls the game



Doom 1



Doom 2



Doom 3

# Videogame engine

## Definition (I)

### Videogame engine

The videogame engine is a tool that ease game development by means of task automation and hidding some low-level issues

- The main goal of any videogame engine is code re-use
- Videogame of the **same type** with almost no engine coding
  - Development can focus on art design and game rules



Freedoom



Heretic

# Videogame engine

## Definition (II)

Decoupling the engine and the videogame is not possible

- It limits the complete reuse of the engine

The main dependence is the game genre

- Different genres involves different needs

Some low level modules are easier to generalize

- User events, audio, text rendering, ...

Game engine tuning: Capacity of videogame engine to adapt and satisfy specific needs

# Videogame engine

## Commercial engines (I)



BioShock, Gears of War  
(List of games)



Far Cry, Crysis  
(List of games)



Angry Birds (List of games)



Battlefield, Need for Speed  
(List of games)



Half Life (List of games)

# Videogame engine

## Commercial engines (II)

**libGDX**



# Videogames genres

## Third person videogames

### Features

- Puzzles
- Platforms
- Advanced motion

### Challenges

- Camera motion
- Camera collision detection
- Player cinematics



Mass effect 2



Ghostbusters 2

# Videogames genres

## Fighting videogames

### Features

- 3D environment and players
- 2D motion (lateral scroll)
- Environments with limited size
- Complex players animation

### Challenges

- Great player graphical quality
- Collision detection with players and weapons
- Complex input (variety of attacks)



Street fighter



Mortal Combat 9

# Videogames genres

## Racing videogames

### Features

- Two subgenres: Simulation and arcade
- Related genre: Flight simulators

### Challenges

- Custom hardware
- Physics
- High graphical quality
- Speed
- Create speed perception



Need for speed



# Videogames genres

## Strategy

### Features

- Two subgenres: Real-time and turn-based strategy
- Isometric view
- 2D or 3D environments
- Low resolution players

### Challenges

- Large number of units
- Complex user interfaces
- AI
- Technology trees



Age of Empires 2



Warcraft 3

# Videogames genres

## Massively Multiplayer Online Game (MMOG)

### Features

- Large (or huge) number of users
- Low resolution players

### Challenges

- Intense networking usage
- Business model



World of Warcraft



Galactica Online

# Videogames genres

## Others

- Sports
- RPG
- Puzzles
- Platforms