Introduction to 2D game design

Dani Tost

Story versus game

Stories are linear (not meaning single voice or chronological) in the sense that each time you read (watch) them, the same occurs (except for interactive narrations)

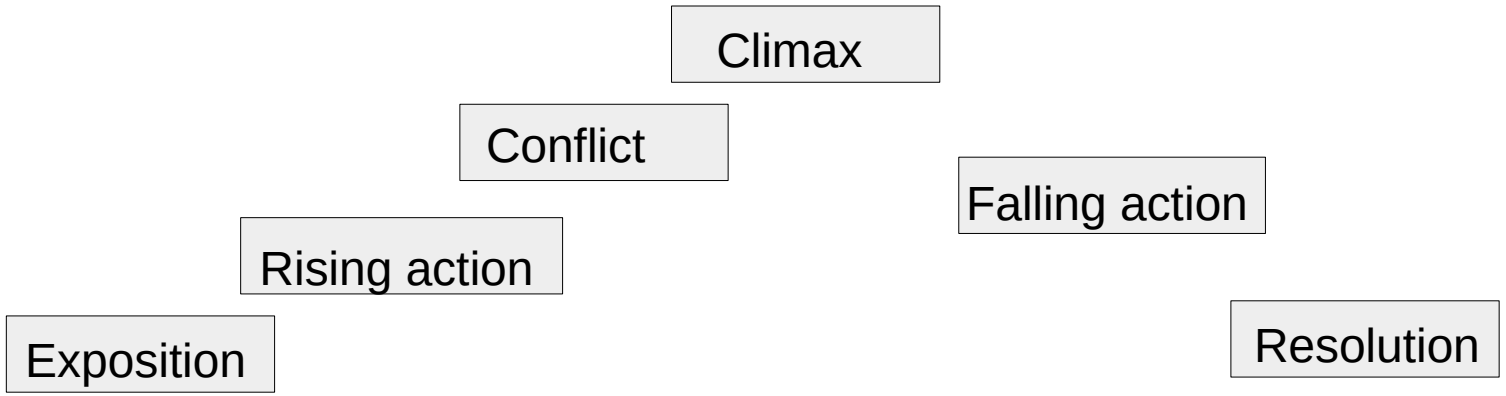
Games are non-linear because, being based on players interactions, they behave differently, the “story” changes, or at least they provide the illusion of change.

Narrative structure

In general, game narrative follow the 3-acts classic model:

Approach | Middle | End

These steps hat can be further subdivided into:



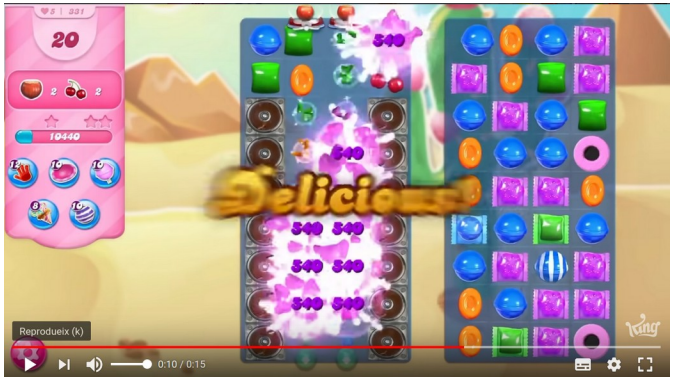
Narrative structure

The narrative structure of a game is:

- embedded narrative: pre-generated narrative content that exists prior to a player's interaction with the game. It is presented in the introduction, the cut-scenes and within the game itself

- emergent narrative: that is created from the player's interaction with the gameworld

Example:
The embedded narration occurs in the Candy Kingdom with various characters such a as Tiffi



https://en.wikipedia.org/wiki/Candy_Crush_Saga

Graphical elements

A game occurs in a series of different **Environments** (places where the action take place)
Inside an environment, different narrative "**acts**" ("**scenes**", "**episodes**") can be played with different narration, goals and rules.

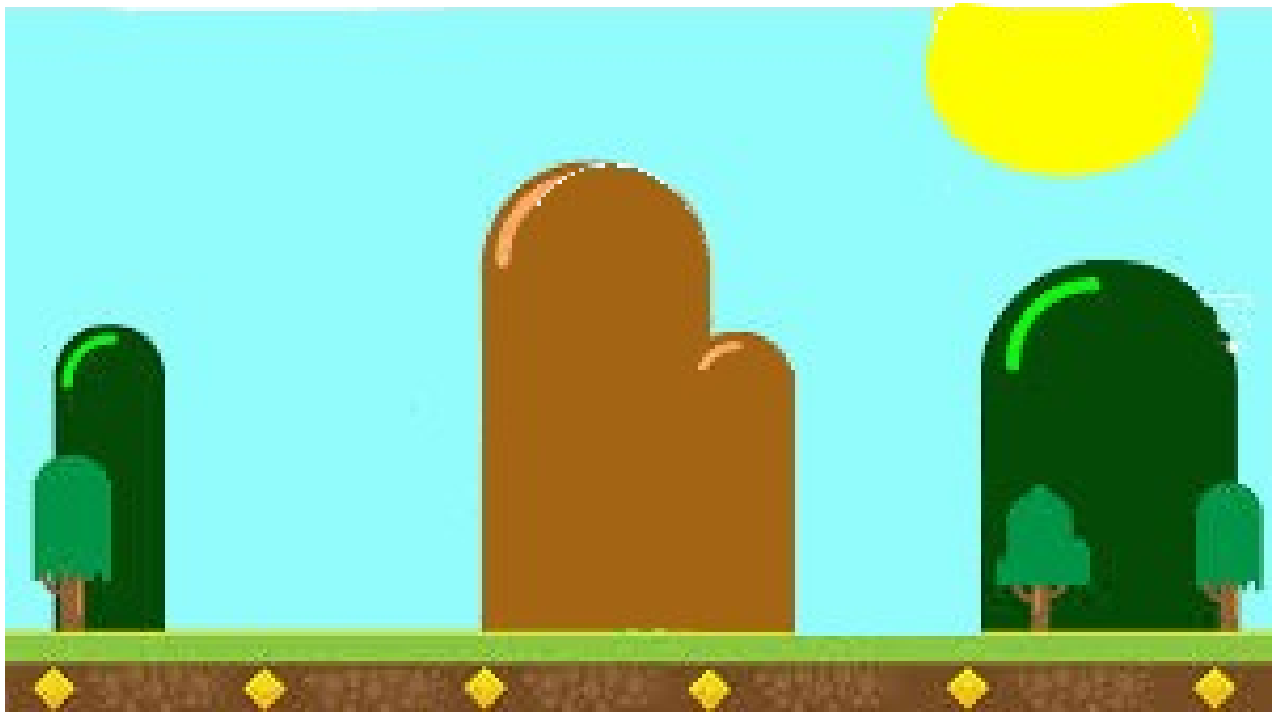
Game engines often use the word scene for environment.

The graphical objects in a scene:

- Scenario or background
- Characters
- Objects
- Virtual camera model
- Lighting model

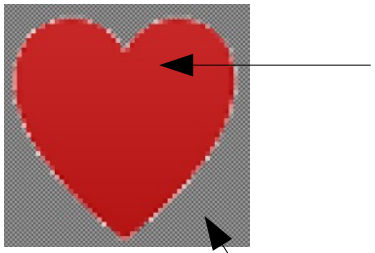
Environment

In 2D games, the environment is composed by one or more 2D background images



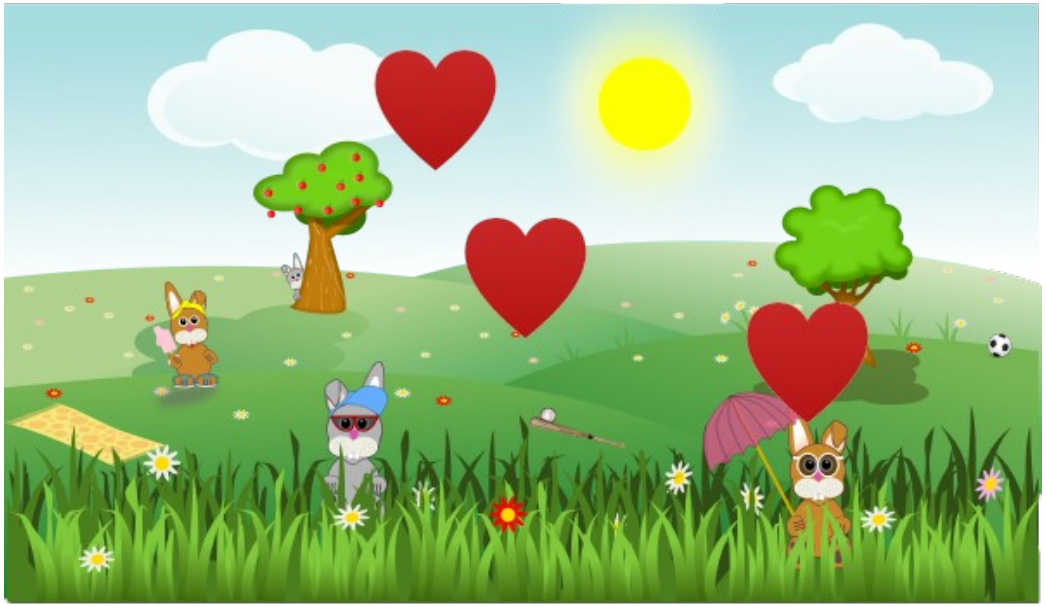
2D geometric models

Geometric models can be **vectorial** (made of lines, polygons, curves, meshes) or **raster** (2D images or subimages called **sprites)**



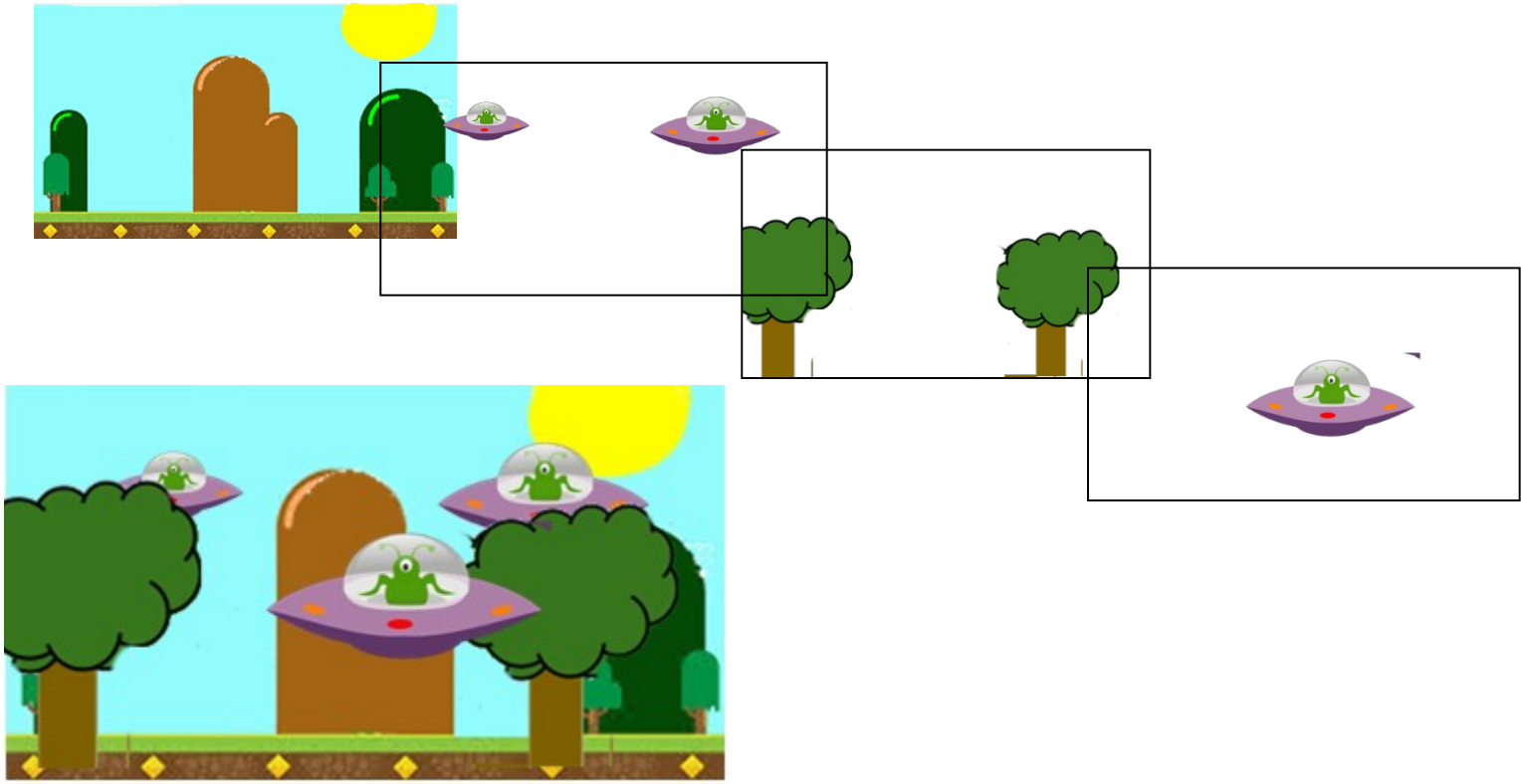
2D shape drawn on a rectangular subimage

Transparent background



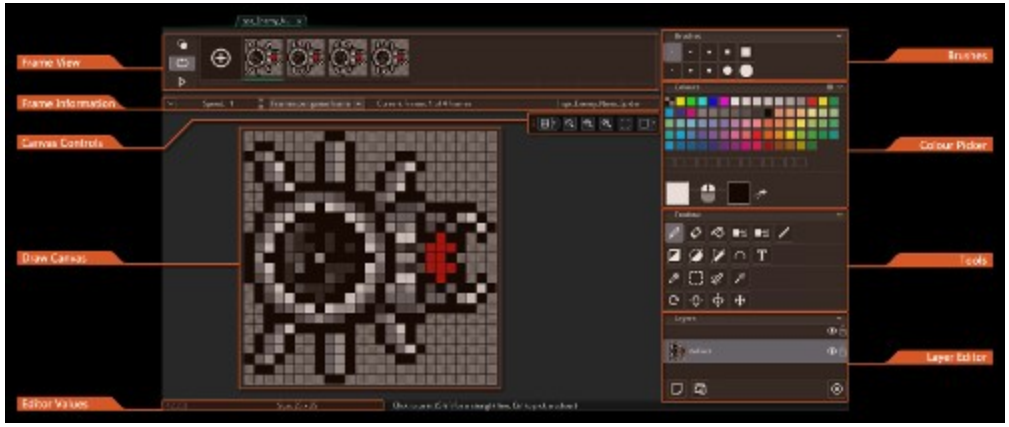
2D models: layers

To simulate depth, 2D sprites are generally grouped into layers. Layers are rendered orderly beginning by the furthest and ending by the narrower to the viewer.



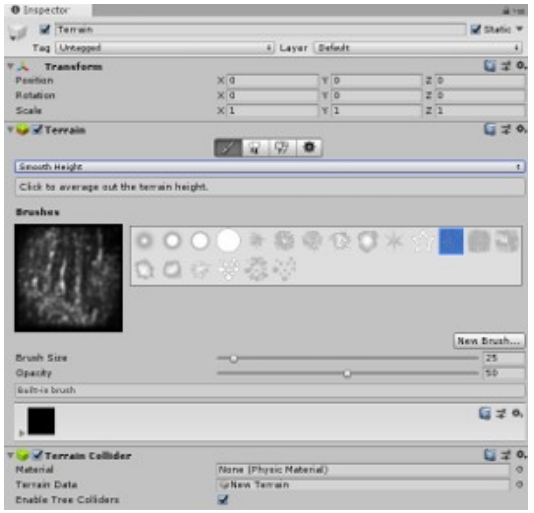
Environment

Most game development software suites provide editing tools of creating 2D layers and 3D landscapes



The gameMaker Image Editor.

Image from : https://docs2.yoyogames.com/source/_build/2_interface/1_editors/images.html



Creating terrains with Unity3D

Image from https://docs.unity3d.com/Manual/terrain-UsingTerrains.html

Content Creation Software

2D

Vectorial

Adobe Illustrator
Sketch
CorelDraw
Affinity
Inkspace (open source)
Canvas
SVGator (animations)

Raster

Adobe Photoshop
Krita (open source)
Gimp (open source)
Pixel Studio

3D

3DS Max
Zbrush
Maya
Blender (open source)
Cinema 4D
CAD Systems:
- Rhino3D
- Catia
- SolidWorks
- AutoCad

.... and many others

Game assets

- Designing 2D (or 3D) graphical models is very costly
- You need to create the geometric model, its materials, textures and animations
- Using existing graphic assets can be a solution, at least for the proof of concept
- A large variety on on-line 2D and 3D assets stores exist
- E. g., see:

https://graphicriver.net/game-assets
https://itch.io/game-assets

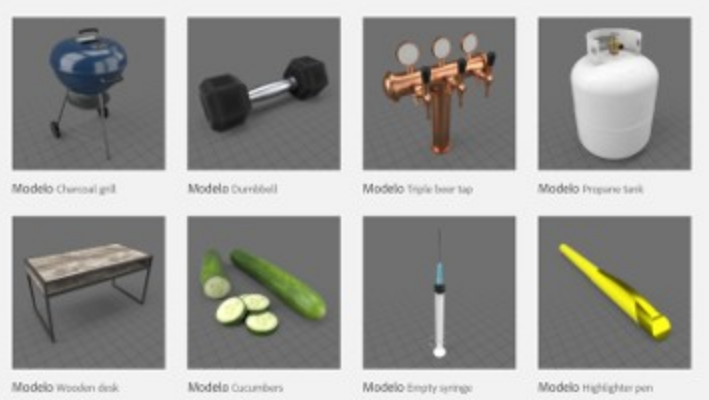


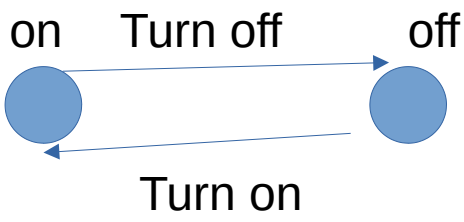
Image from https://stock.adobe.com

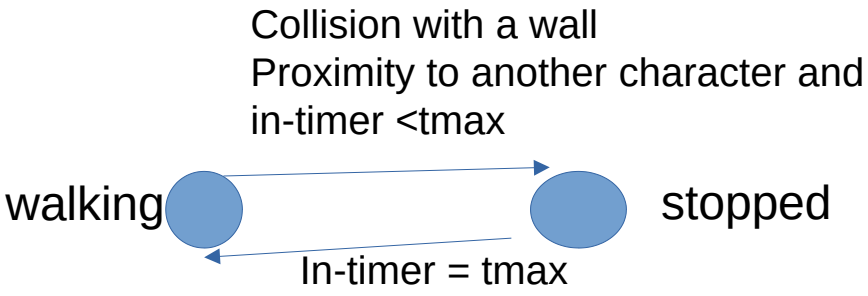
Finite state automaton

Games are divided into different “scenes” or “acts” or “episodes” located in different environments. This structure can be represented as a Finite State Machine (FSM), also called Finite State Automaton.

A FSM represents a game as machine made of one or more states. Only one state is active at each moment and the machine transition from one state to the other. They are generally represented as graphs in which nodes are states and edges are the transition conditions.

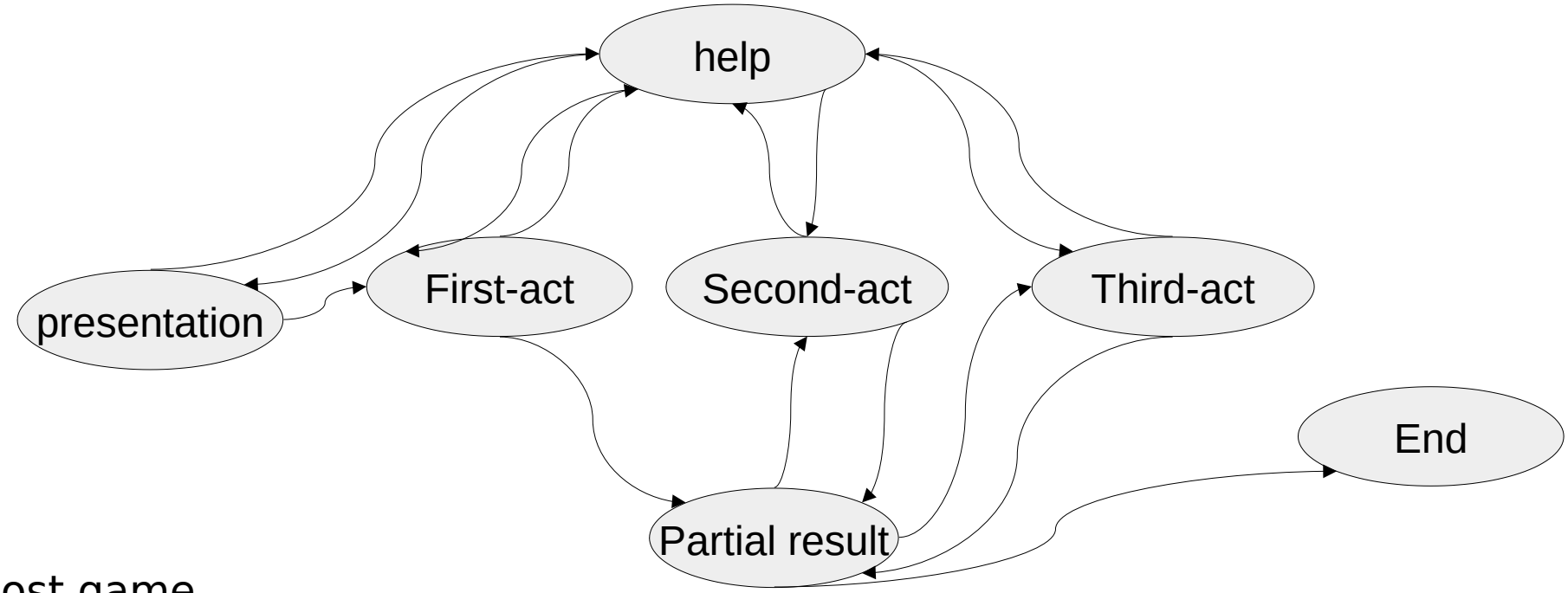
FSM are also used to represent the AI of a character in a game, as in the following examples.





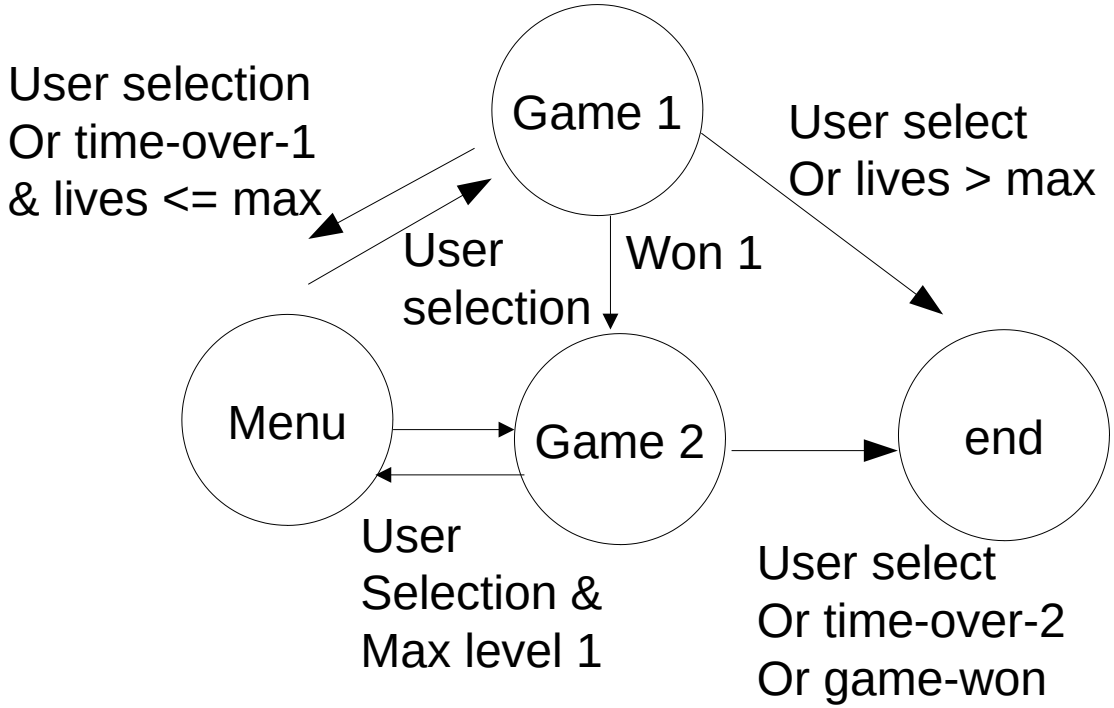
Finite state automaton

Example of a game structure FSA:



Most game development software suites provide a Finite State Automaton

Finite state automaton



Can you describe the behavior of this game?

Game loop

The structure of a game is, as in any other interactive application, based on a loop (game-loop). At each instant of the loop objects/actors location and aspect are recomputed, user interactions are evaluated, and all interactions between game elements computed.

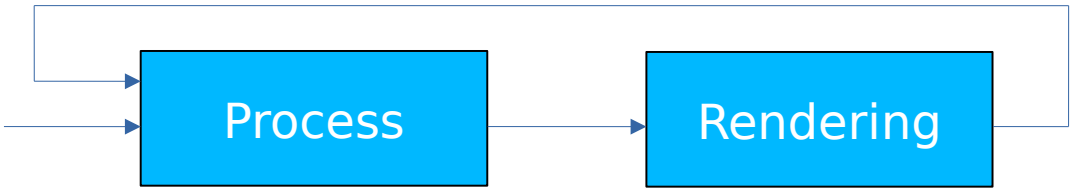
```
Inicializations
while not end
    Update
    Process interaction
    Draw
```

Most game development software suites provide the game loop so it is transparent for programmers

Game loop

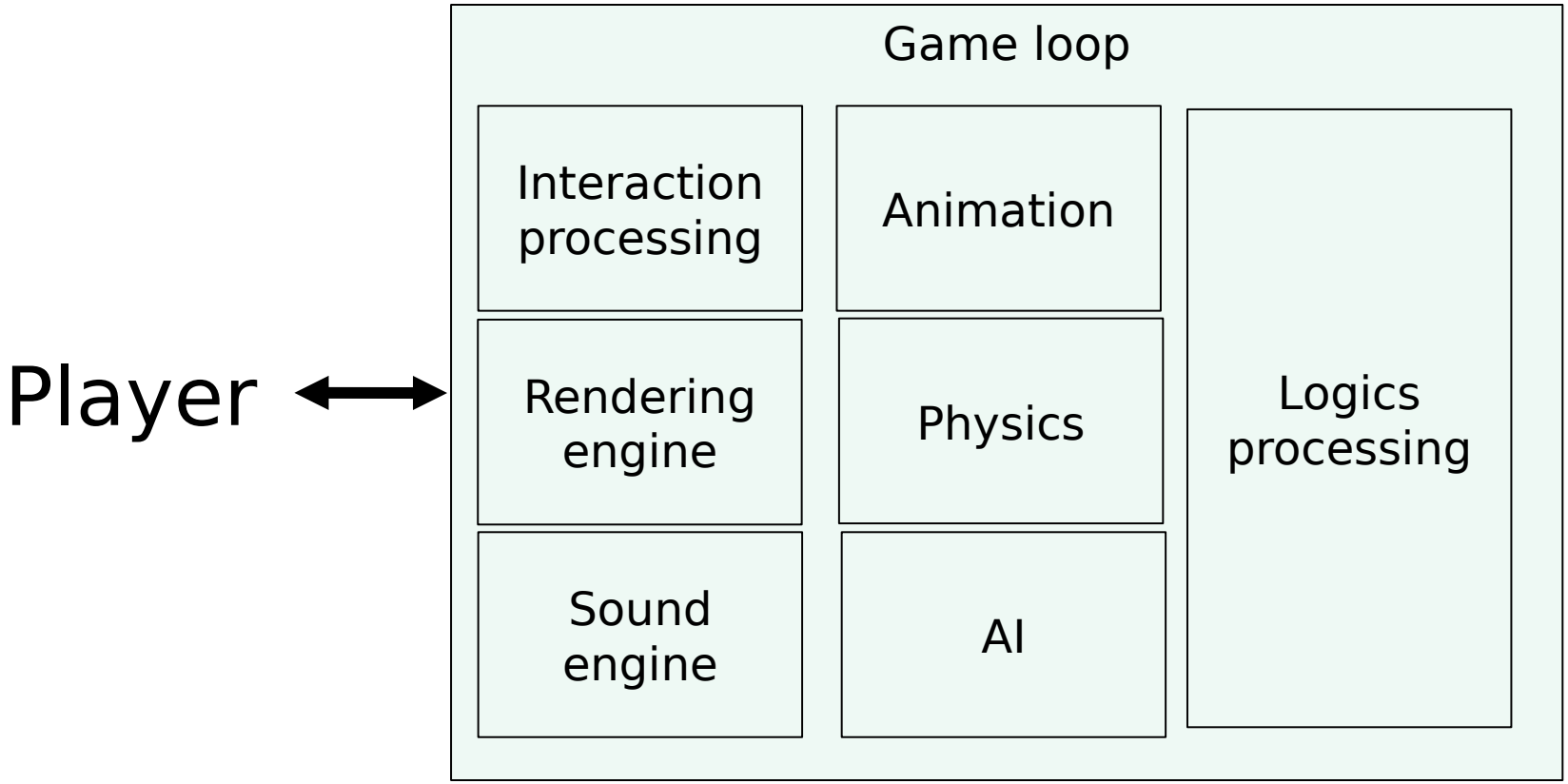
At each cycle of the loop:

- The input and schedulers processing
- The **Artificial Intelligence** to determine game-driven characters and objects actions (**non-player characters = NPC**)
- Collisions are detected
- Game update:
    - Some objects/effects disappear
    - Some objects/effects appear
    - Some objects change their attributes
        - New positions and orientations of the objects
        - Deformations
        - Changes in the textures and materials
- The scenario is rendered



Game loop

A simplied vision of what happens in a game loop



Next week…
GAME ENGINES