How are video games created?

• Once upon a time...
How are video games created?

- Programming languages such as C/C++, Java, Objective C
- Software packages such as Adobe Flash
- Game engines such as Unreal engine, Box2D engine, Unity3D
- Other tools for asset creation such as
  - 2D sprites or 3D models
  - Animation
  - Sound
  - UI (user interfaces)
- Other skills may include hardware design (new interaction mechanism) and creative design skills (e.g. storytelling, interaction design)

Structure

- What is a game engine
- Introducing Unity3D
- 3D concepts review
- Game objects and game components
EL639 Video Game Design

- The lectures:
  - Week 1  Virtual Worlds
  - Week 2  Scripting with C#
  - Week 3  Game Physics
  - Week 4  Rules and Mechanics
  - Week 5  Player Interactions
  - Week 6  Project week (no lecture/workshop)
  - Week 7  Artificial Intelligence
  - Week 8  User Interfaces
  - Week 9  Levels and Optimisation
  - Week 10  Advanced topics
  - Week 11  Peer Testing (Pitch) 10-1pm, Monday
  - Week 12  Project Week (no lecture/workshop)

EL639 Video Game Design

- The workshops:
  - Week 1  3D Islands (3D)
  - Week 2  Roll a Ball (3D)
  - Week 3  Alien tripod (3D)
  - Week 4  Fruit Ninja (2D)
  - Week 5  Nightmare horror (Isometric)
  - Week 6  Project week (no lecture/workshop)
  - Week 7  Nightmare horror II (Isometric)
  - Week 8  Spaceship attack (2D)
  - Week 9  Spaceship attack II (2D)
  - Week 10  Project work
  - Week 11  Peer Testing (Pitch) 10-1pm, Monday
  - Week 12  Project Week (no lecture/workshop)
Game Engines

- What are game engines?
  - software frameworks or systems for game creation
  - Usually provide rendering/graphic engine, physics engine with collision detection, sound, artificial intelligence, animation, event/input manager, networking, etc.
- Why do we use game engines?
  - Reusability of codes or software components
  - Possible cross-platform portability

**Game Engines**

- **Physics engine** - handles how game objects interact with each other and the environments by simulating real-world physics
- **Input Manager** - looks after interactions between the player and the game; manages the drawing of graphical user interfaces (GUI) and the handling of mouse click, etc.
- **Sound Manager** - initialises and controls how 2D/3D sound is delivered within the game environment
- **Game objects** - represent all the assets placed in a game environment: the terrain, sky, trees, weapons, rocks, nonplayer characters, rain, explosions, etc. Game objects contain components such as AI, graphics, sound and physics.
  - Artificial Intelligence(AI) - determines how a game object will behave
  - Graphics - dictates how the game object is drawn
Quiz

• Which component of the game engine decides if a rock should roll down the hill in the game world?
  a) Input Manager
  b) Sound Manager
  c) Physics Engine
  d) Renderer (graphics)

The main loop of a game

• What actually happens with the game engine when the player presses the “shoot” button?
The main loop of a game

- **Initialisation** - computer memory is allocated, saved information is retrieved, and graphics and peripheral devices are checked.
- **Main loop** - runs continuously over and over again until the player decides to quit the game.
  - the game executes a cycle of functions that processes user inputs; checks through all game objects and updates their state, including their position; updates the environment with respect to game object positions, user interaction, and the physics system; and finally renders the new scene to the screen.
  - each main loop renders one frame of graphics on the screen. Time between frames being rendered on the screen is called FPS (frames per second)
- Clean up

Quiz

- As the number of game objects increase, the amount of work in the main loop will also increase. This will **increase or decrease** FPS?

- Please select all reasonable ways to increase FPS.
  a) delete game objects no longer in use
  b) don’t render the object surfaces which are not visible to the users (a process known as occlusion culling)
  c) randomly select the game objects and exclude the selected objects from processing/update in the main loop
  d) only process the randomly selected components (such as sound, physics) of a game object in the main loop
Unity3D

- A 3D and 2D game engine for game development
- Cross platform development (PC, Mac, Linux, web)
- Game console support (PS2, XBox360, Wii U)
- Mobile capability (Android, iOS, Windows Phone 8)
- Simple for non-programmers (although scripting needed)
- Robust for experienced programmers

Cartesian coordinates

- To represent a location in a 3D space, we need X, Y, and Z-axis
  - X for horizontal, Y for vertical and Z for depth
  - This coordinate system is very important in scaling, rotating and positioning any 3D object
Local space v world space

• In diagram (i), both objects are shown in world space
• In diagram (ii), the smaller box is made a “child object” of the bigger box. The smaller box is now said to be (3,4) relative to the bigger box, because the origin (zero point) is now the world position of the bigger box (3,3)
• Question: why would we make an object a “child object”?

Quiz

• Assume that we have a big cube on the world space (3, 5, 2), and a small cube on the world space (6, 2, 4). We now make the small cube a “child object” of the big cube. What is the local position of the small cube?
  a) (3, 3, 2)
  b) (3, -3, 2)
  c) (2, 2, 4)

• Now consider the case where both the big and small cubes are on the world space (3,5,2). What would the local position of the small cube after being made a child object of the big cube?
Polygons, edges, vertices and meshes

- All 3D shapes (or meshes) are made up of interconnected 2D shapes known as polygons.
- Polygons are made up of three connected edges.
- The locations at which these edges meet are called vertices.
- It is important to understand polygon counts – the total number of polygons of a 3D model.

Collider mesh

- Mesh data can be used to specify a shape for collision that is less detailed than a visible object. This can improve the performance of game engine.
Materials, textures and shaders

- **Materials** provide the mean to set the visual appearance of a 3D model.
- You can assign images as **textures** to a material.
- You can also select a **shader** of the material, which defines the rendering style. E.g. in a reflective shader, the material will render reflections of surrounding objects.

Rigidbody physics

- Physics engines use Rigidbody dynamics system to create realistic motion by allowing an object to have properties such as mass, gravity, velocity (speed) and friction (drag). This can be done by adding “rigidbody” component to a game object.
- Once the “rigidbody” component is added to an object, the object will behave according to real world physical law. E.g. a ball with a “rigidbody” will fall to the ground and bounce.
Quiz

- Physics calculations consume a large amount of computing resources. Therefore not all game objects should be made “rigidbody.” Please tick all the game objects which should have a “rigidbody” in a car racing game.
  a) The race cars
  b) Distant audience cheering
  c) Trees waving at the background
  d) Obstacles on the street, such as traffic cones

Collision detection

- By giving game objects a “collider component”, the game engine will analyse inter-object collisions for these objects.
- Primitive shapes colliders are simple geometric objects whilst Mesh shapes colliders are based on more complex meshes
Objects, components and variables

• In Unity3D, a game object is made of various components, and each component in turn has some variables.

• A game object contain at least one component called Transform, which simply tells Unity the position, rotation and scale of the object.

Objects, components and variables

• Let’s take an example of a 3D island (a game object). What are its components?
Assets and scenes

- Assets refer to all building blocks of your game, such as 3D models (meshes), sound files, images files, etc
- In Unity3D, scenes are individual levels
- Some games contain only 1 scene (such as a puzzle game that dynamically load content using scripts), or multiple scenes

Unity3D user interface

- Hierarchy panel: Where game objects are
- Scene window: game level design
- Project panel: Where assets are
- Inspector: settings for assets and game objects
Workshop

• We will try to build something like this:

Workshop

• In this workshop, you will create a virtual island, and use the Unity built in first person controller to walk around the island.

• **Workshop objectives:**
  • create a 3D island with the terrain tools.
  • understand the difference between game objects and game components
  • import a first person controller to the virtual world
  • use particle system to create fog and rain effects
• Advice: save often, test often!
Home exercises

You are highly encouraged to do all these exercises at home (even if you just spend 5-10 min thinking about each exercise if you have no time!)

Questions

• What is a game engine?
• What is rigidbody physics in a game engine?
• What is softbody physics?
• What is the difference between world and local space?
• What is the difference between render mesh and collision mesh?
Think

• Although it is common for a game studio to use off the shelf engines such as Unity3D, many major studios still develop their own engine. Why is it the case? What are the advantages and limitations of using off the shelf engines?

Try

• Please reorder the correct steps of a typical game main loop:
  • update environment
  • render graphics
  • process user input
  • update game object
Design

• In Unity3D, a game object is made of various components, and each component in turn has some variables (see slide 21)
• Let’s say you want to build a simple game that allows users to shoot moving balloons using a mouse.
• Please design the game objects, together with the components and variables for each object.