Virtual Reality Engine
Unity3D Theory

EPFL Immersive Interaction Group
Thibault Porssut
Dr. Ronan BOULIC
Outline

Theoretical Part:

• Available material for the VR Project
• One interface to rule them all (VR Game Engine)
• Unity: Current VR engine in IIG
• Interface (Unity Editor)
• Unity Elements
• Virtual Reality in Unity
Available material for the VR Project

• 3 Oculus Rifts
• 2 HTC Vives
• 1 Katwalk VR + 1 HTC Vive + 3 Vive Trackers
• 1 Hololens
• 1 GearVR + Samsung Galaxy S7
• Some Oculus Go
• Some Cardboards
• 6 Vive Trackers
• 2 Kinects V2
• 5 Leap Motions
Available material for the VR Project

• VR Headset

  Oculus Rift

  HTC Vive

• AR Headset

  Hololens

• Cardboard Headset

  Cardboard

  GearVR + Samsung Galaxy S7

  Oculus Go
Available material for the VR Project

• Tracking System
  - Vive Tracker
  - Kinect V2
  - Leap Motions

• Treadmill
  - Katwalk VR
Programming Assignment

• **1 programming assignment**: a real-time physically-based application with tracking done with Unity; this year the project is to be done in *groups of 3 (40%)*, during **weeks 7-12**, demo and project grading on **week 13**.

• Select among projects given or propose a project

• Material only available in the lab: KatwalkVR, HTC Vive, Oculus, Hololens

• The other material can be borrowed.

• Each device can be used by two groups.
One interface to rule them all (VR Game Engine)

• A system designed for development of VR scenes
• Provides a software framework that the users use to create interactive scenes / implement game scenarios
• Typical VR engines include:
  • 2D/3D graphics
  • Physics engine (collision detection)
  • Sound
  • Scripting
  • Animation
  • Networking
  • etc
One interface to rule them all (VR Game Engine)

- Open Source VR engine
  - OpenGL
  - Ogre3D
  - Panda3D
  - OpenSceneGraph
  - Godot Engine
  - Etc.
One interface to rule them all (VR Game Engine)

• VR engine with license
  • Unity3D
  • Unreal Engine
  • 3Dvia Studio (Virtools 5.1 last version)
  • Cryengine
  • Worldviz
  • Amazon Lumberyard
  • etc
One interface to rule them all (VR Game Engine)

• Comparison

<table>
<thead>
<tr>
<th></th>
<th>Open-Source VR engine</th>
<th>VR engine with license</th>
</tr>
</thead>
<tbody>
<tr>
<td>GUI (visual development)</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Latest features availability</td>
<td>Fast</td>
<td>Slow (Closed)</td>
</tr>
<tr>
<td>(e.g. Geometry Shader)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VR adaptation (e.g. HMD)</td>
<td>Libraries</td>
<td>Plug-ins (API)</td>
</tr>
<tr>
<td>Development</td>
<td>Time consuming</td>
<td>Time saving</td>
</tr>
<tr>
<td>Resources</td>
<td>Rich</td>
<td>Limited</td>
</tr>
<tr>
<td>Multi-platform development</td>
<td>Not limited</td>
<td>Limited</td>
</tr>
<tr>
<td>External device integration</td>
<td>Not easy (via APIs)</td>
<td>Easy (via Plug-ins)</td>
</tr>
</tbody>
</table>
Unity: Current VR engine in IIG

- **Unity3D**
  - Interface (GUI)
  - Powerful scripting
  - Multiple platforms supported
  - Features (animation, network sound, physics engine, etc...)
  - Support (forum)
  - Resources (asset store, demos, tutorial samples, etc...)
  - Not expensive and free for education/personal use
Unity: Current VR engine in IIIG

- **Unity3D - GUI**

Unity: Current VR engine in IIG

- Intuitive tool
  - Low cost development system
  - 2D and 3D interactive content
  - Build-in physics engine
  - Multiplatform publishing
  - Quality control
  - Ready-made assets
  - Knowledge-sharing community
Interface (Unity Editor)

Play/Pause/Step (Test your game without compiling)

Grab/Translate/Rotate/Rescale your GameObject in your scene

Several Tabs: Scene (Manipulate your objects)/Game (rendered on your camera)/Asset Store (download Unity packages)
Unity Elements: Overview

- Assets
- Scenes
- Game Objects
- Components
- Scripts
- Prefabs
- Interface
Unity Elements: Overview

- 3D coordinate system
- Local space/World space
- Camera and view port
- Polygons, edges, vertices, and meshes
- Materials, textures, and shaders
- Rigid Body physics
- Collision detection
Unity Elements: GameObject

**Name of the Object:** Way to find it.

**Tag:** group membership (Useful for raycasting or finding a group of objects)

**Layer:** Access to the GameObject properties.

**Inspector:** Access the GameObject properties.

**Static:** Non-moving object (Useful for rendering optimization)

**Transform:** 3D coordinates (Position, Rotation and scale) left-handed coordinate system.

Icon to highlight your object in the scene.
Unity Elements: Components

Mesh: Vertex of your 3D object (3D shape)

Effects: Special renderer effect (particle effect)

Physics: Gravity (Rigidbody), Collision (Collider)

Navigation: Pathfinding Algorithm (Nav Mesh Agent)

Audio: Audio Listener (your avatar), Audio Source (3D sound) and different Filters

Rendering: Camera and Skybox

Layout: Canvas (Menu in your game)

Miscellaneous: Terrain and Wind Zone.
Unity Elements: Main Camera

**Tag:** Main Camera (the default camera used by Unity to render your game)

**Background:** The colour of the area when there is no Skybox

**Field of View:** The more important it is, the more you need to render.

**Clipping Plane:** The size of the area that needs to be rendered.
Unity Elements: Directional Light

By default, Unity creates one directional light but you can create several to make special effects:

- Attach to an object
- Light some confined space
- Etc....
Unity Elements: 3D object
Unity Elements: Plane vs Terrain

• Plane: 3D object (Primitive) with a simple Collider (less computation)
• Terrain: 3D object with a terrain component with a specialized tool.
Unity Elements: Scripting

• Inherit from Monobehaviour
• Visual Studio Community  C#
• Attach to a GameObject to change its behavior
• Instanciate Prefab on runtime

```csharp
public class InstantiateCube : MonoBehaviour {
    public GameObject CubePrefab;
    public Transform location;

    void Start () {
        // Use this for initialization
    }

    void Update () {
        // Update is called once per frame
        Instantiate (CubePrefab, location);
    }
}
```
Unity Elements: Prefab

“Prefab acts as a template from which you can create new object instances in the scene.”*1

*1 https://docs.unity3d.com/Manual/Prefabs.html
Unity Elements: Audio

3D Sound = “all channels downmixed to mono and attenuated according to distance and direction.”*1

The sound can be output through an audio listener or an audio mixer.

Original Sound Clip

Sound Effect to reverberate your sound (Cave like)

Change 2D sound to 3D Sound

*1 https://docs.unity3d.com/Manual/class-AudioSource.html
Unity Elements: Audio

Tab to manage your different mixers

Manage the different mixers under different subgroups

Add affect to a group of sound

Simple output without any effect
Unity Elements: Audio

**Volume**
How loud the sound is at a distance of one world unit (one meter) from the **Audio Listener**.*1

**Pitch**
Amount of change in pitch due to slowdown/speed up of the **Audio Clip**. Value 1 is normal playback speed.*1

*1 https://docs.unity3d.com/Manual/class-AudioSource.html
Unity Elements: Mecanim

The Animation Tab allows you to create simple animations directly in Unity.

- Add Key Frame to your animation
- Record your animation when moving your object
- Control your animation manually
- Move in your animation
- Created on the GameObject you have animated

Add Property
Unity Elements: Mecanimit

This tab opens automatically when an animator is created. One animator can control different animation clips thanks to the Animator tab.

Different variables to control different states.

Transition and State (animation clip)
Unity Elements: Networking

- Add this component to an Empty GameObject named NetworkManager.
- Test your application on one machine.
- Add the player prefab to be instantiated in the scene.
- Randomly spawn your player in the different location implemented.
- HUD to control the Network Manager on runtime.
Unity Elements: Networking

All these components need to be added on the player prefab

HUD to control the Network Manager on runtime

HUD to control the Network Manager on runtime

HUD to control the Network Manager on runtime
Virtual Reality in Unity

To have a good VR application you need:

- Good Interaction (Grabbing/UI/Controller)
- Good Physics
- Good AI (Social/Behavior)
- Locomotion (Motion Sickness)
Virtual Reality in Unity

• Edit -> Project Settings -> Player -> XR Settings -> Virtual Reality Supported

-> The camera tagged Main Camera will move according to your head

It is not enough to have a VR application. It is just the beginning of your journey.
Install the Last Unity Version for next course

• Link to download:
  https://store.unity.com/fr/?ga=2.187361902.794958388.1519564968-856357652.1519564968

Different use means different plans but one application:
• Personal License: Annual revenue or funds raised of $100k or less.
• Plus: Annual revenue or funds raised of $200k or less.
• Pro: No limits on revenue or funding.
=> Choose Personal License
Questions?
Useful Resources

• Unity3D:
  https://unity3d.com/fr/learn/tutorials
  https://unity3d.com/fr/learn/tutorials/topics/asset-store/merry-fragmas-multiplayer-fps-part-1

• VR:
  https://unity3d.com/fr/learn/tutorials/s/virtual-reality
Useful Resources

• Mecanim:
  https://unity3d.com/fr/learn/tutorials/topics/animation/animate-anything-mecanim

• Networking:
  https://unity3d.com/fr/learn/tutorials/topics/multiplayer-networking/introduction-simple-multiplayer-example?playlist=29690