# VRUE Tutorial 4

Project Task
14.11.2017

Iana Podkosova

# Project Task

- VR game for 2 players
  - Can also be a serious game or a simulator/educational application
- Compulsory features (you have to implement them)
  - Leap player creates different objects that are picked up and used by both players
  - Throwing objects (correct and correctly synchronised physics)
  - LeapPlayer can move (but both players always see each other)
  - Interactive game space extention: 2 methods
- Feedback on your game idea in the comments section next to the submission (TUWEL)

# Game Space Extention

- Multiple floors + elevators
  - Modify y-position of the camera
- Increased walking speed
- Flying with specific gesture input (not on button press!)
- Redirected walking: add rotation gains
- Redirected walking: modify geometry on the fly
- Use overlapping spaces
- Suggest your own method!

# Use of Compulsory Features

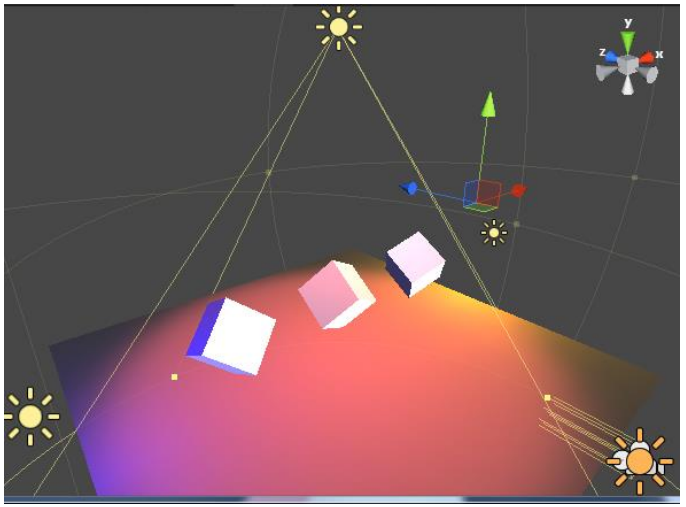
- They should **add** something to the game
  - *Example 1*: Leap-player and Vive-player are 1 meter apart from each other and both can move. Leap-player throws an object to the Vive-player
    - No need in throwing here! Simply passing would do
  - *Example 2*: Leap-player and Vive-player are on the opposite sides of a playground and are not allowed to come too close to each other (volleyball simulator). Leap-player throws an object to the Vive-player
    - The use of throwing makes sense here

# Common Corrections for Project Idea

- Movement of the Leap player alone is not interactive game space extention
- Vive player moving around by controller input
  - It can already move by walking! Need something better
- Increased walking speed
  - Should not  happen on button press
  - Gesture + walking – walking with increased speed
- Elevators
  - Player should do something to make them move

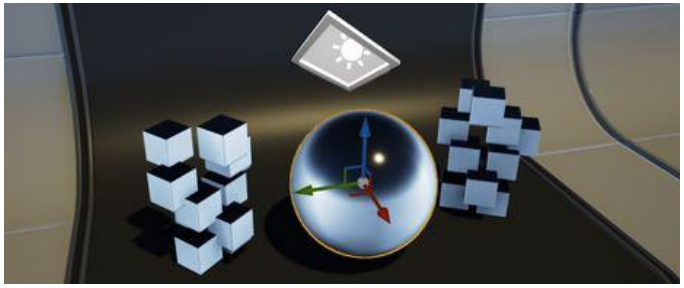
# General Things

# Light - Unity



- Light sources are different
  - Point
  - Directional
  - Spot
- Occlusions produce shadows
  - Culling Mask might be used to cancel the unwanted shadows
  - It works with levels
- When using multiple light sources it is recommended to bake the light for static objects
  - Use light probes for realistic light distribution
  - This will allow you to minimize the load on GPU
  - For more information see https://unity3d.com/learn/tutorials/topics/graphics/lights

# Light - Unreal



https://docs.unrealengine.com/latest/INT/Engine/Rendering/LightingAndShadows/LightTypes/index.html

- Light sources are different
  - Point Light
  - Directional Light
  - Spot Light
  - Sky Light
    - captures the distant parts of your level and applies that to the scene as a light
- Mobility Settings
  - Static
  - Stationary
  - Movable
- For more information see
  - https://docs.unrealengine.com/latest/INT/Engine/Rendering/LightingAndShadows/LightTypes/index.html
  - https://docs.unrealengine.com/latest/INT/Engine/Rendering/LightingAndShadows/Shadows/

# Environment - Unity

- You can add a lot to your game by importing the Environment Unity package
  - Assets->Import->Environment

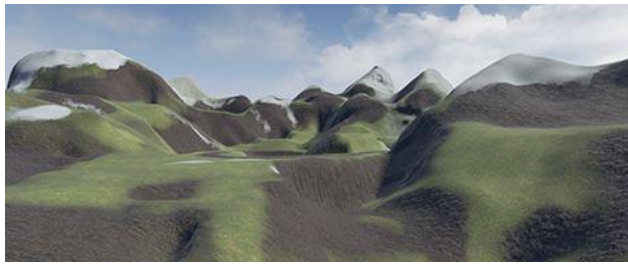
Terrain

- Large plain with changeable textures and height
- Including grass and trees
  - models can be found in the Environment package
- You can set the wind and other features as well

Water

- Circular mesh included to the Environment package
- You can configure the color and waves, etc.
- There are different modifications as Basic and Pro (more taxing on resources)

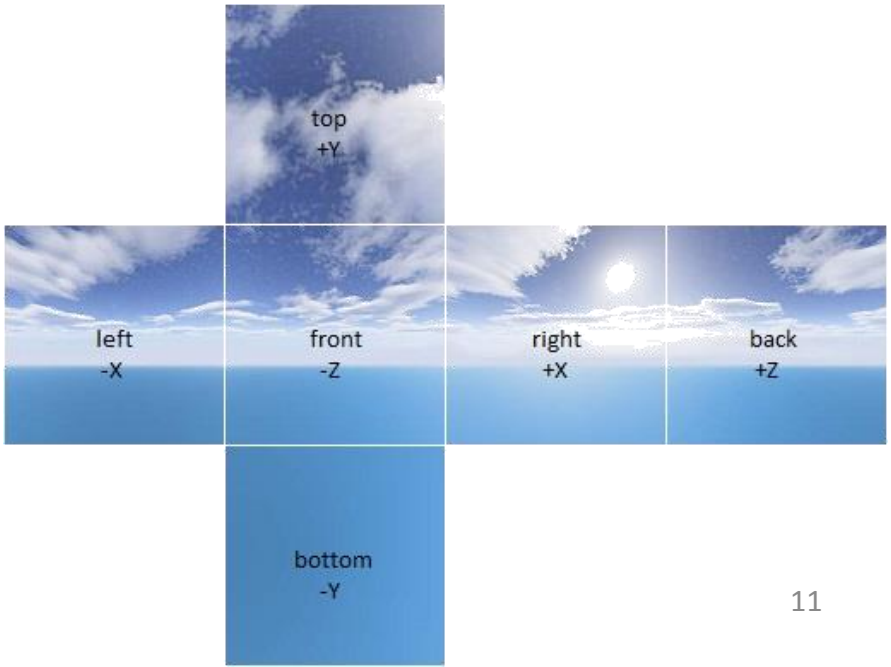
# Environment - Unreal



- Unreal provides a lot of resources in their Starter Content
  - You can select "With Starter Content" when creating a new project
- Epic Games Launcher → Learn: Here you can find many useful resources, for example
  - Content Examples
  - Particle Effects
  - Water Planes
  - Landscape Mountain
- More on landscape:
  - https://docs.unrealengine.com/ latest/INT/Engine/Landscape/

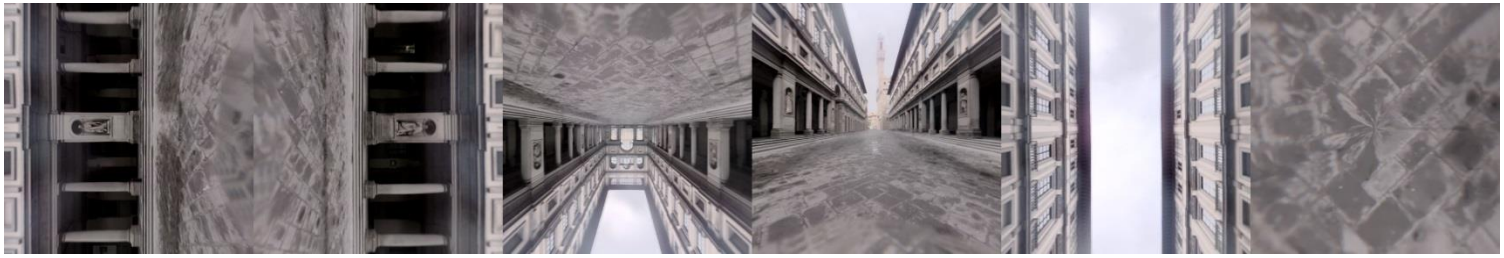
# Skybox- Unity


- A set of textures that are placed in infinity
- Might include halos and flares (light source visual effects)
- For a variety of free skyboxes check the Asset store



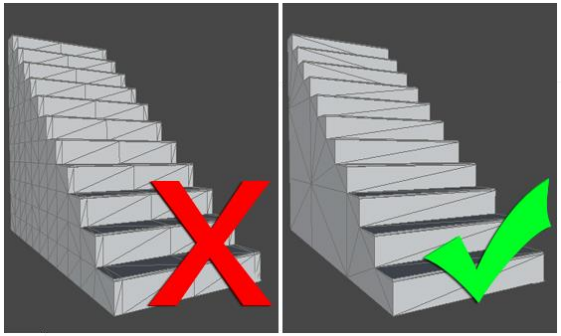
# Skybox - Unreal


- Sky Sphere Blueprint
  - Is used in most templates
  - Can be edited via Blueprints
  - Nice YouTube Tutorial:
    - https://www.youtube.com/watch?v=6Wp8xYbvhYg
- More information:
  - https://wiki.unrealengine.com/Skybox_from_DDS_cubemap
  - https://docs.unrealengine.com/latest/INT/Engine/Content/Types/Textures/Cubemaps/index.html

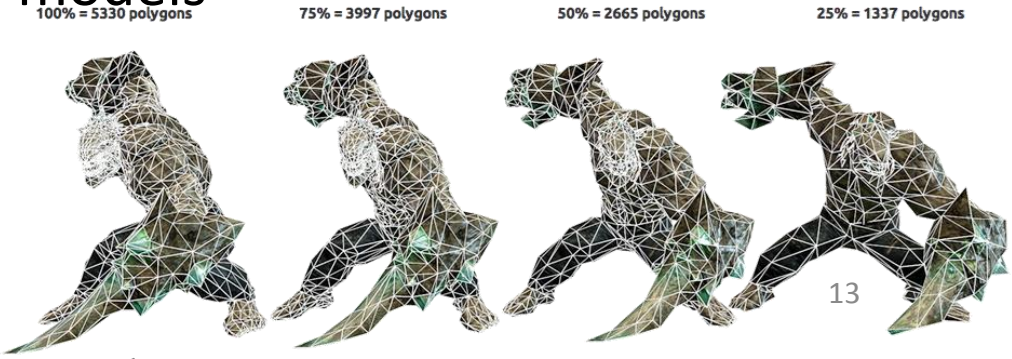


# To Consider: Polygon Count

- Objects with polygon count over several thousand are taxing for CPU and GPU
  - Remember why high-poly collider was a bad idea for physics?
  - Same here with light and shadows
- Try using the low-poly models when possible
  - If not very possible – use a 3D modelling tool to simplify your model
  - If you are loosing to many frames or your application is getting slow - Check your models



http://blog.mixamo.com/



# To Consider: Textures

- A lot of large textures is not a good idea
  - Your project get heavier
  - The game is loading longer
  - Run-time operations like light estimation take more time
- Good practice:
  - Use small textures
  - Use tiling of the texture to cover bigger area
  - To avoid visible edges between the tiles, modify your texture as described here.

# To Consider: Number of Imported Assets

- During the development the number of the imported assets is growing fast – greatly increasing the size of the project
- Make sure to remove the assets that you don’t use

# Grading

- Project is worth 75 points
- You get points for:
  - Compulsory features
  - Networking implementation quality
    - **The project must be a networked 2-player game! Single-player projects will not count at all**
  - Thought-through gameplay
  - Visuals
  - Project structure and organization (naming, hierarchy)

  - Extra points: Use of available controller features and GUI