

Tutorial on Game Development Using UE4

Siu-hang Or

Computer Science & Engineering Department

Outline

- Game development
- Create a room
- Interactive elements
- Adding enemies

Unreal Engine 4

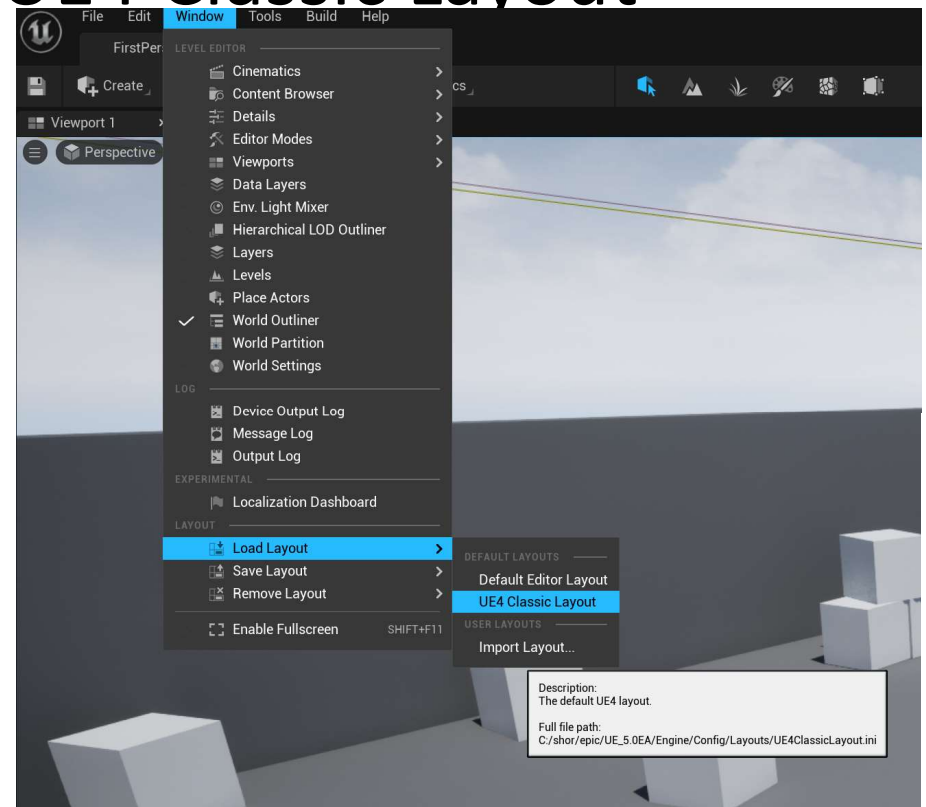
- facilitate game development
- <http://www.unrealengine.com>
- Provides sophisticated game development support
- First download the “Epic game Launcher” and install it
- If you are working behind firewall, Add “-http=wininet” to its “Target” field under property

Unreal Tutorial

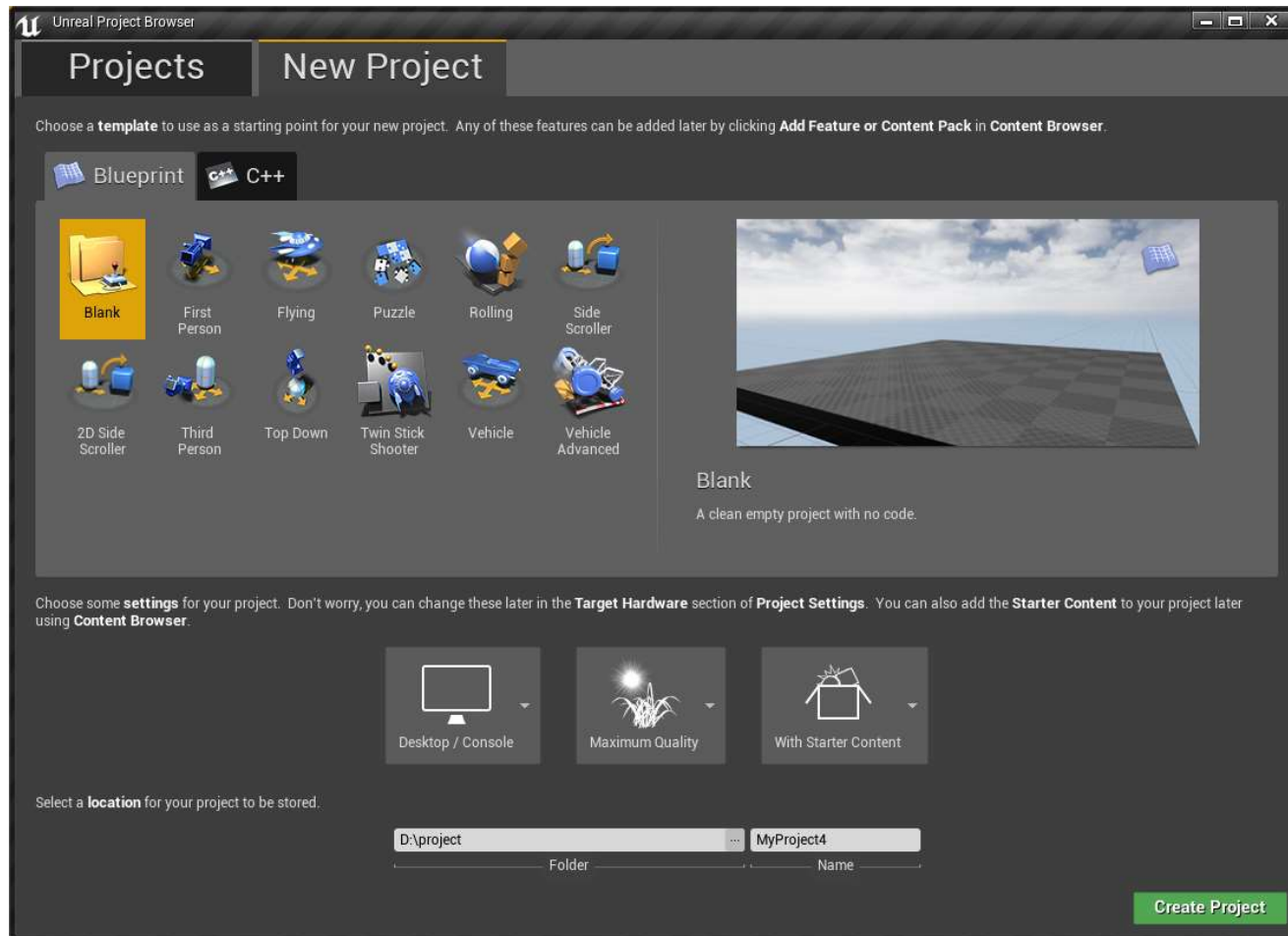
- You may choose to install 4.2X or 5 Early Access
- Choose 4.2X if your graphics card is not GTX1080 or above
- For our tutorial purpose, you are recommended to use 4.2X

Unreal Tutorial

- You can still follow our tutorial in Unreal 5 by toggling
“Window/Load Layout/UE4 Classic Layout”
- Then you can just following same instructions in tutorial



Click on New Project(4.22)

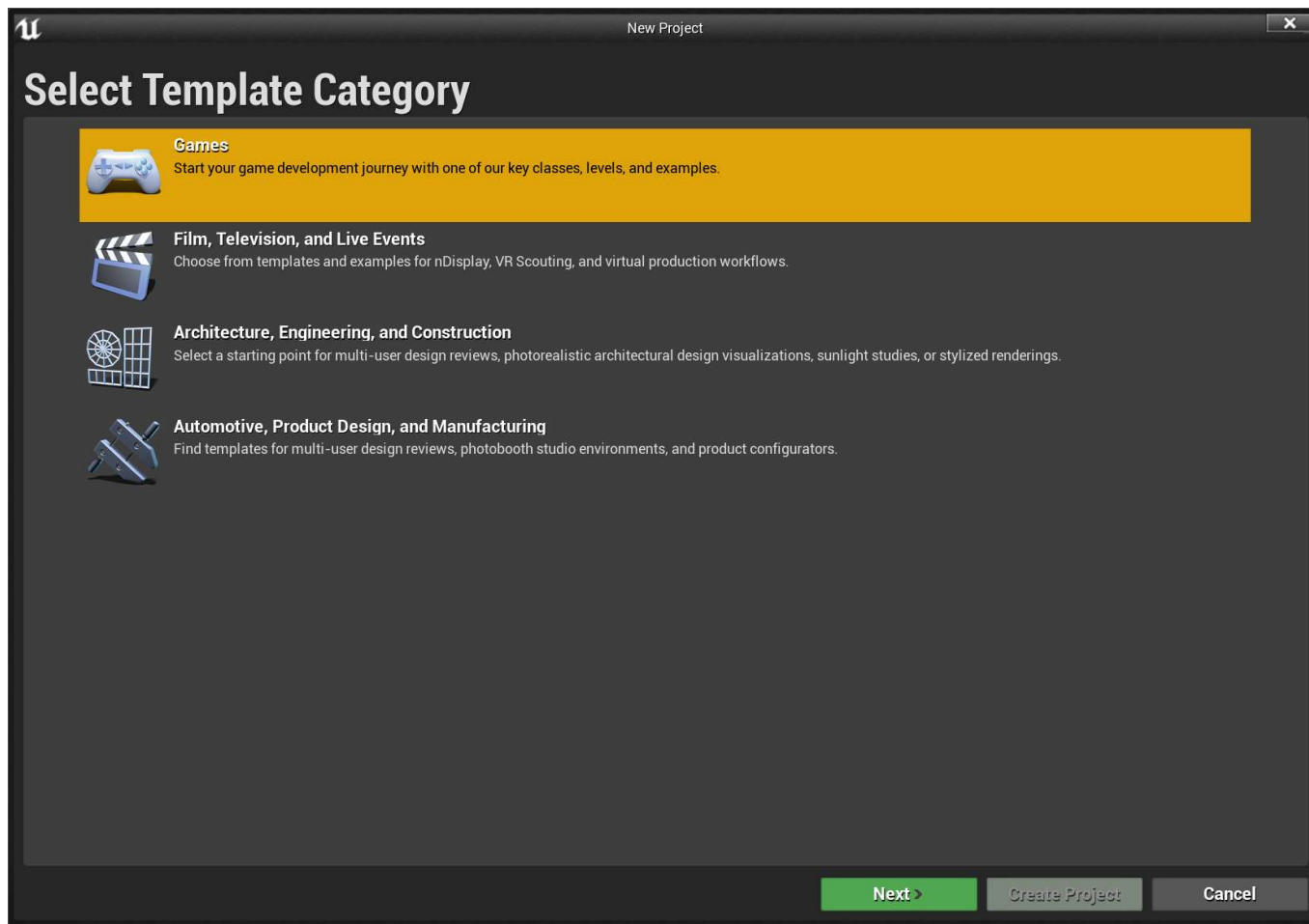


Various Game types(4.22)

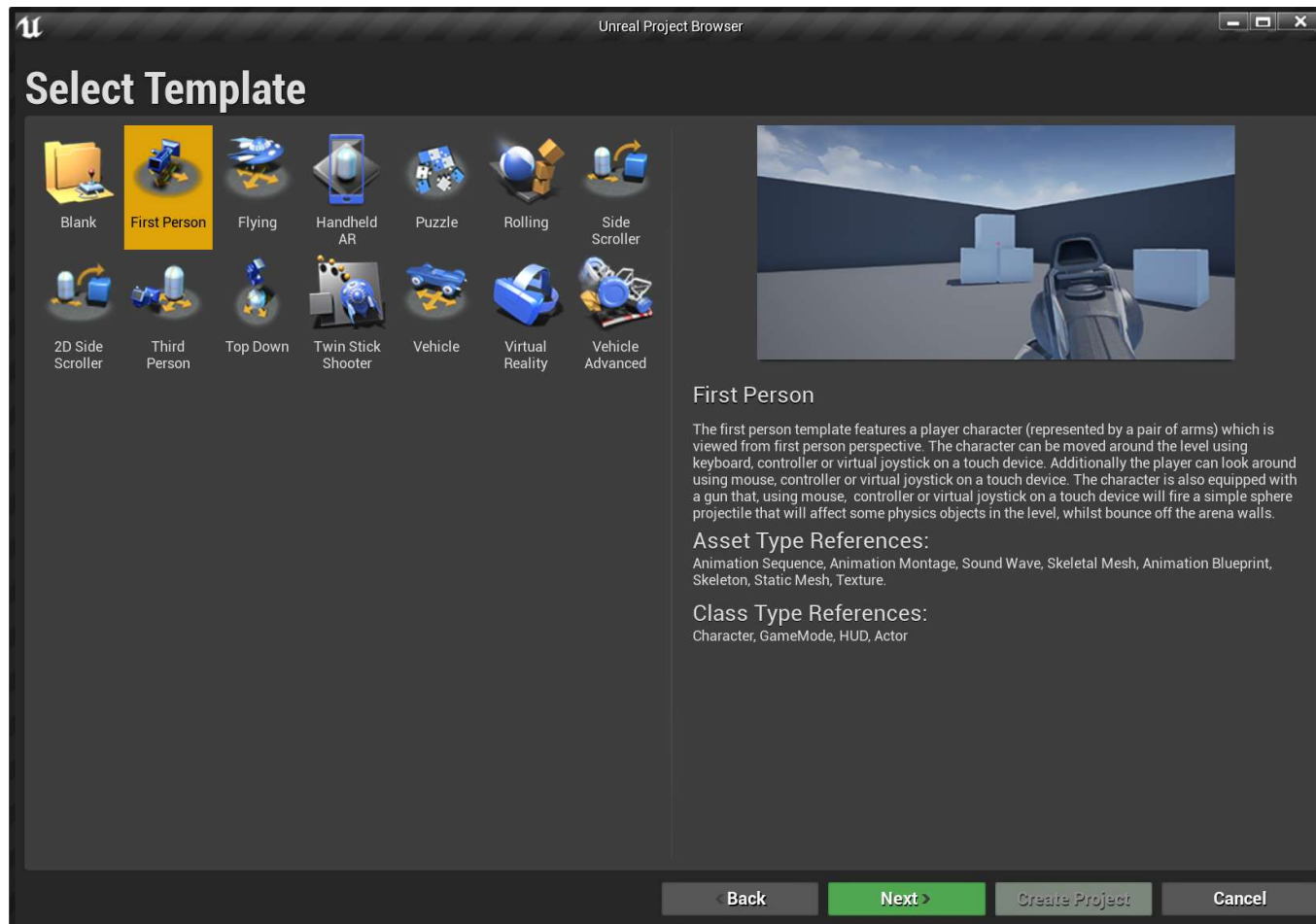
- Choose First Person with all other default setting



Click on New Project & select “Games”(4.24 or later)



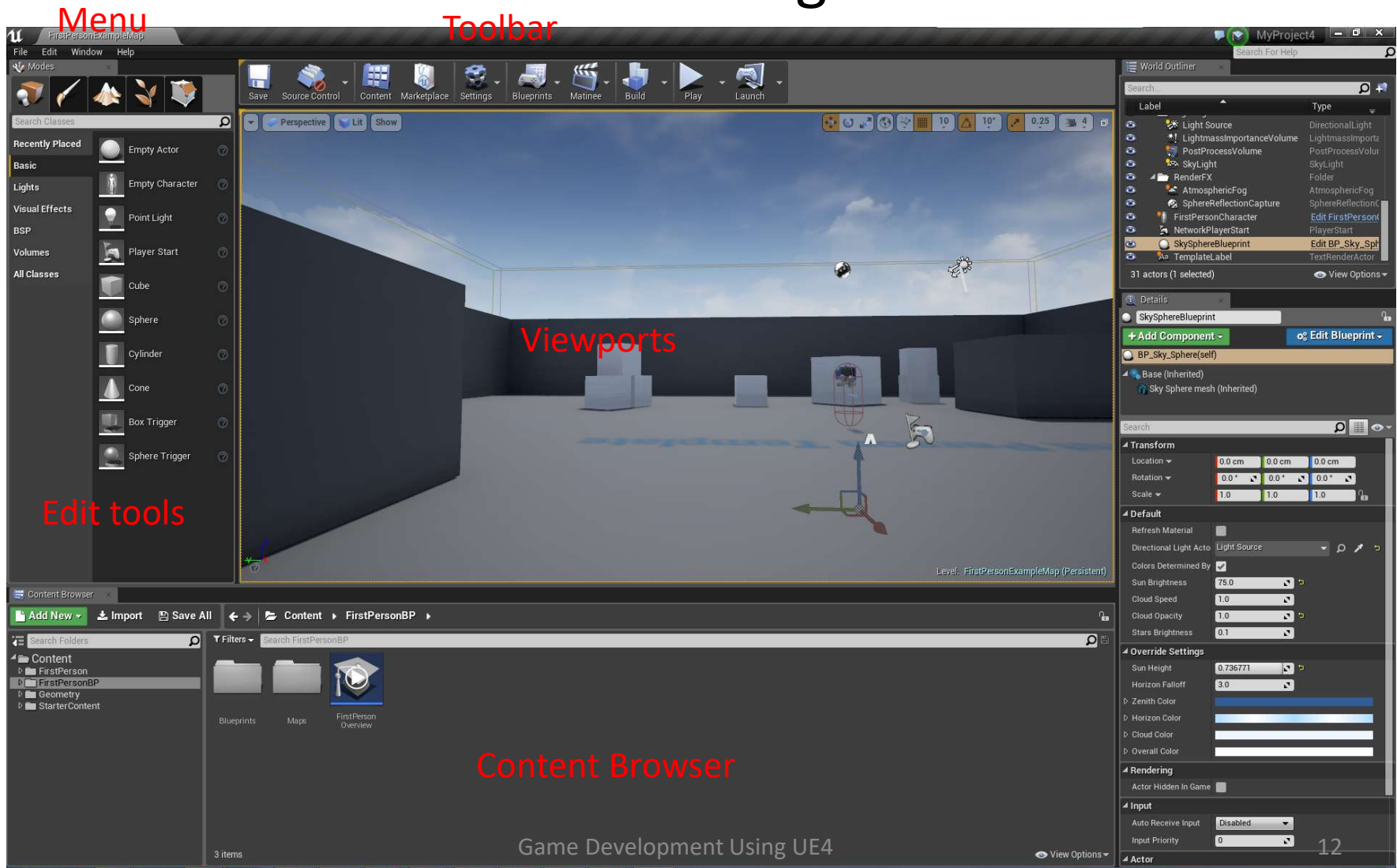
Choose “First Person”(4.24)



“Next”, Then “Create Project”

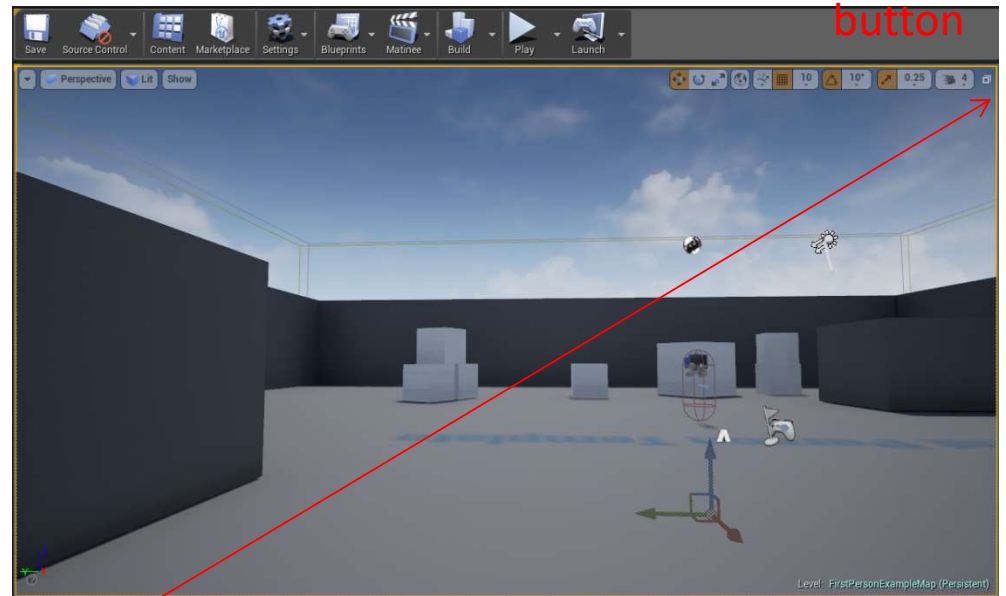
Basics

- We use the editor to build game levels



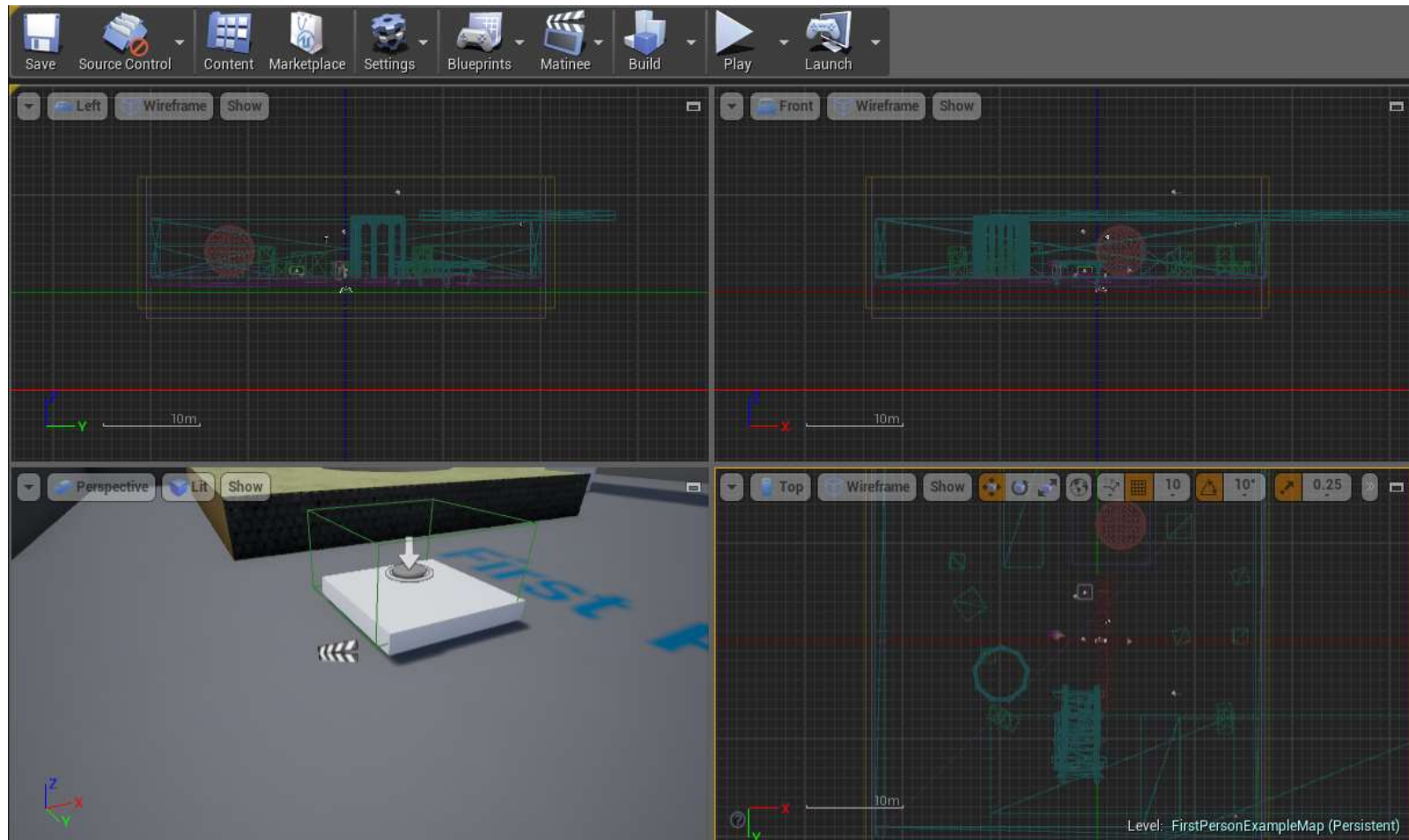
Basics

- Default is single viewport
- You can switch to 2x2 views for more precise positioning
- Click on the rearrange button



Rearrange
button

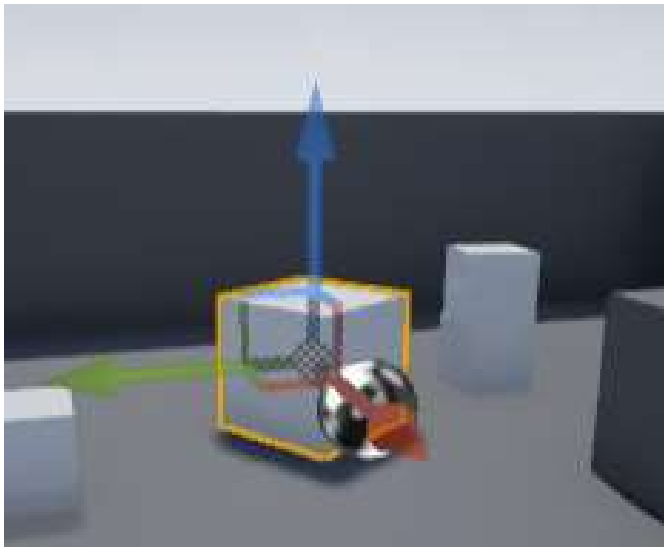
Basics



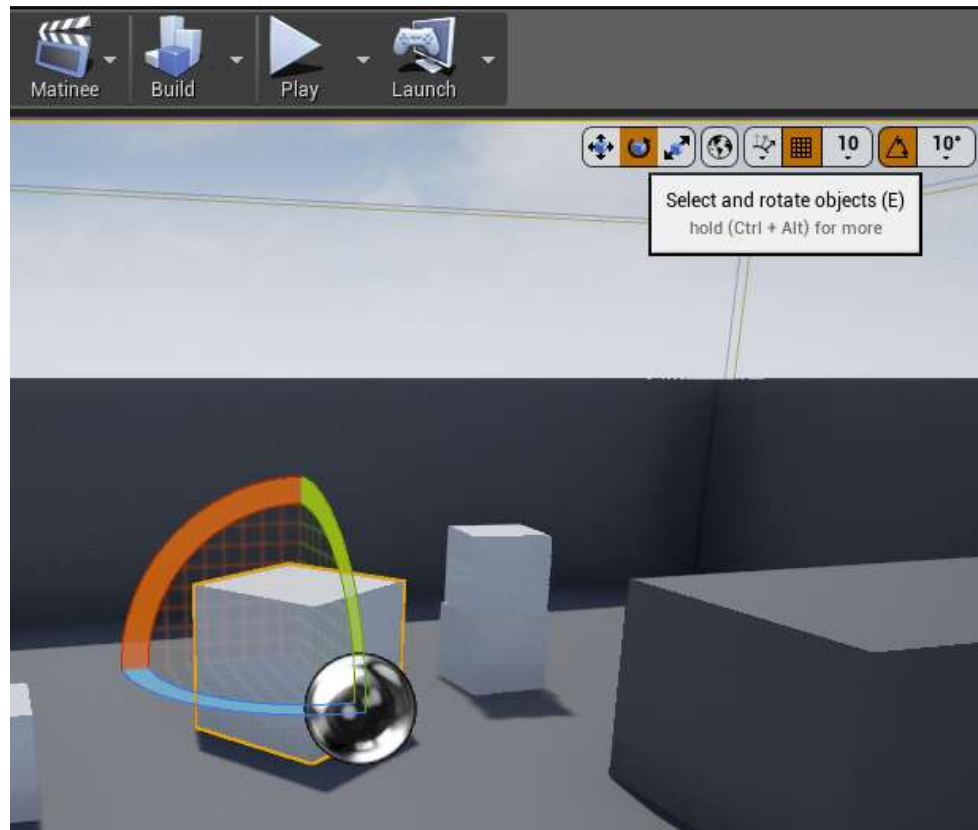
Using top and side views help more precision positioning

Editing

- We can move or rotate any entities in the editor



Move

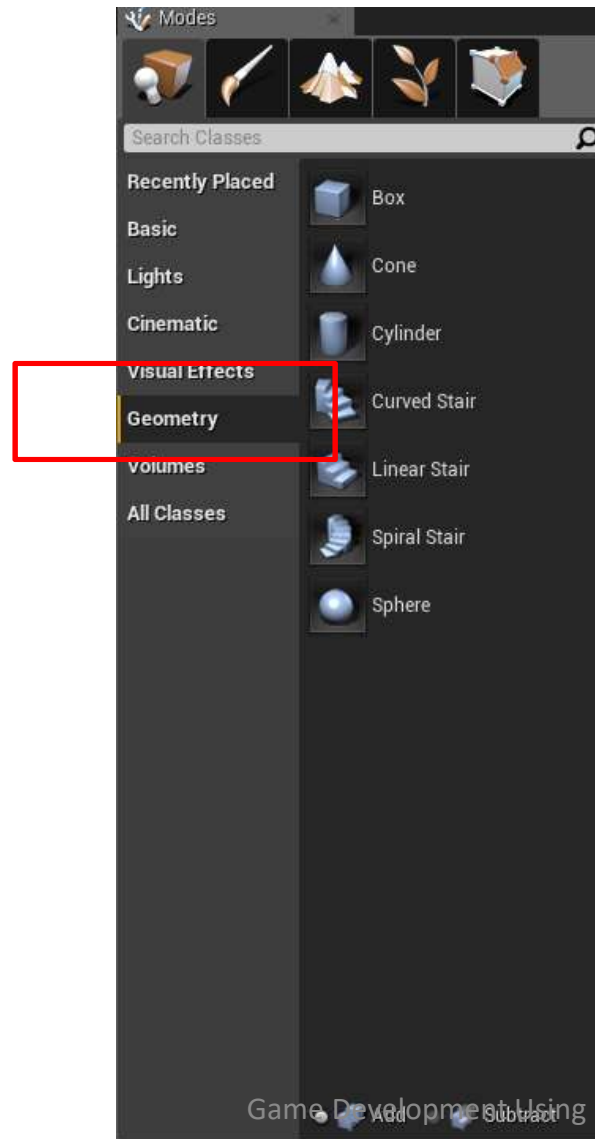


Building level

- Game world is being crafted out through
 1. BSP (Goemetry)
 - used for rapid prototyping levels in the early stages
 2. Static Meshes (Basic)
 - Set of polygons drawn through hardware
 - Provide much more fine details

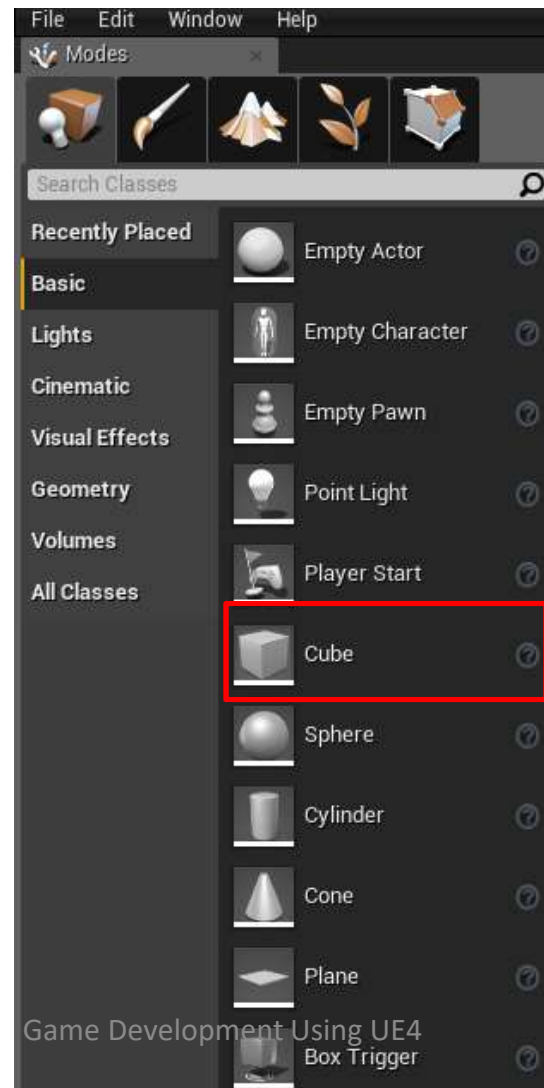
Building level

- BSP



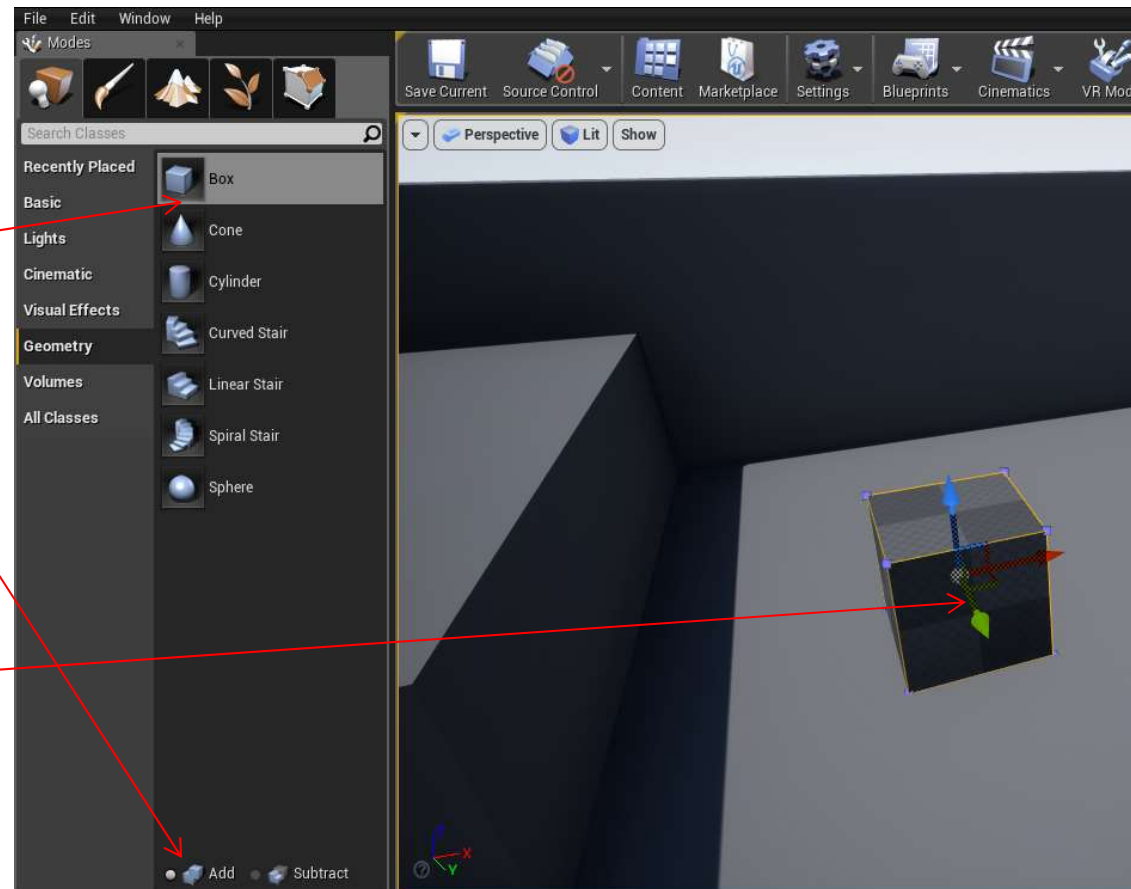
Building level

- Static Meshes



Let's create

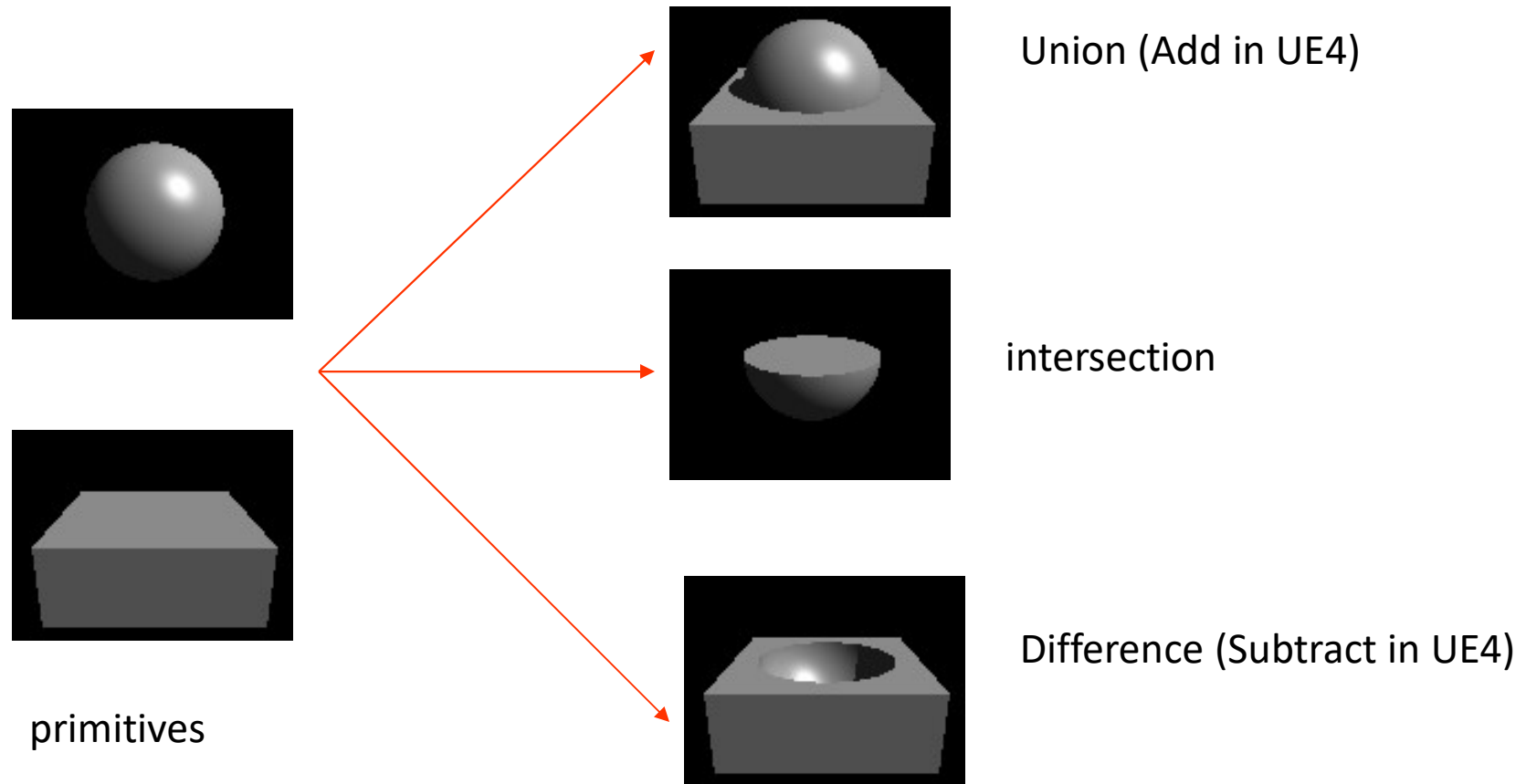
- A BSP brush
 - Click on cube brush
 - Note the “Add” button is default in this case
 - Drag the icon into a place designated in level



Constructive Solid Geometry (CSG)

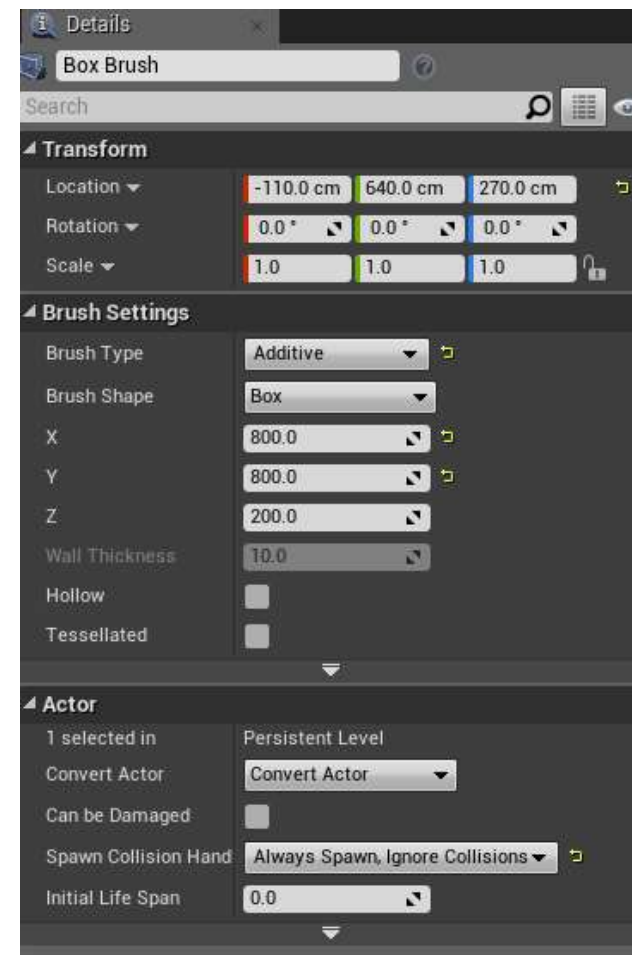
- Consists of *Boolean* set operations on closed primitives in 3D space.
- The three CSG operations are *union*, *intersection* and *difference*.
- Produce polygon models after the modeling phase
- Used in level design in game as it is more intuitive, thus easy for artists to build complex level from basic primitives such as cone, rectangular boxes etc.

Constructive Solid Geometry (CSG)



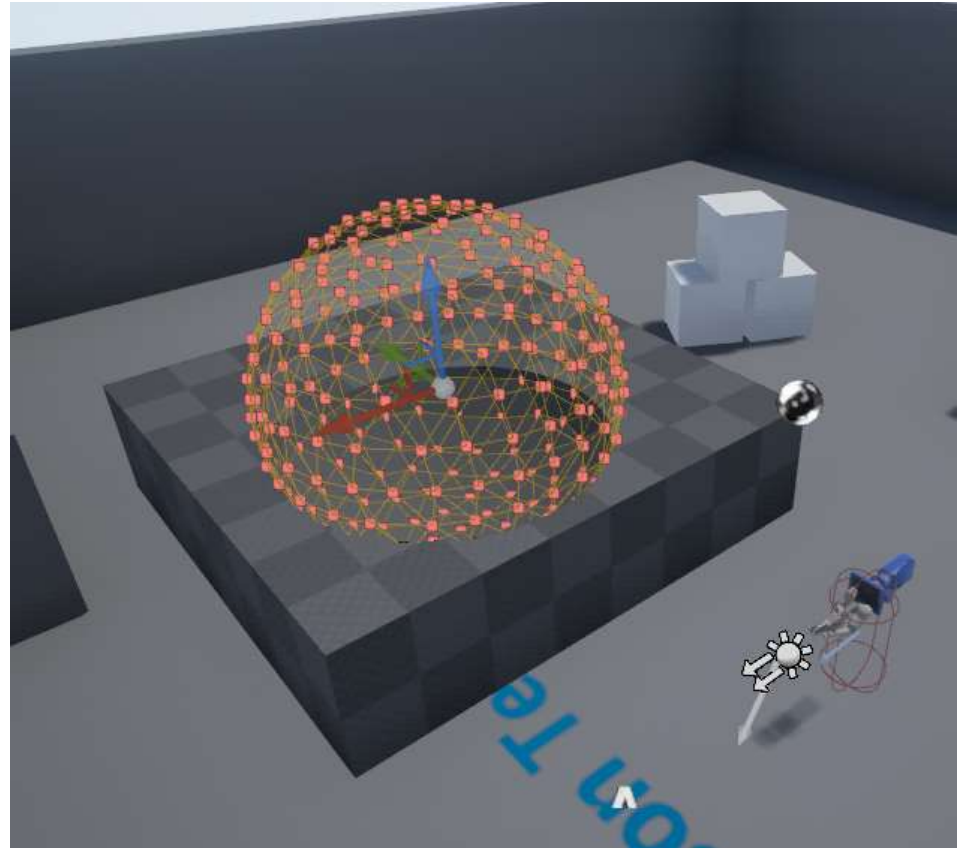
Changing Brush

- You can modify the brush after creation by changing the parameters of the brush (on lower right of editor screen)
- X, Y corresponds to the plane, Z is height
- Build a bigger block here either adjusting scale field or using scale handles



Subtraction

- You may also try the cut (subtracting) brush
- This is using a spherical subtractive brush to craft out a pit
- Experiment with different shapes to see the result



Test it!

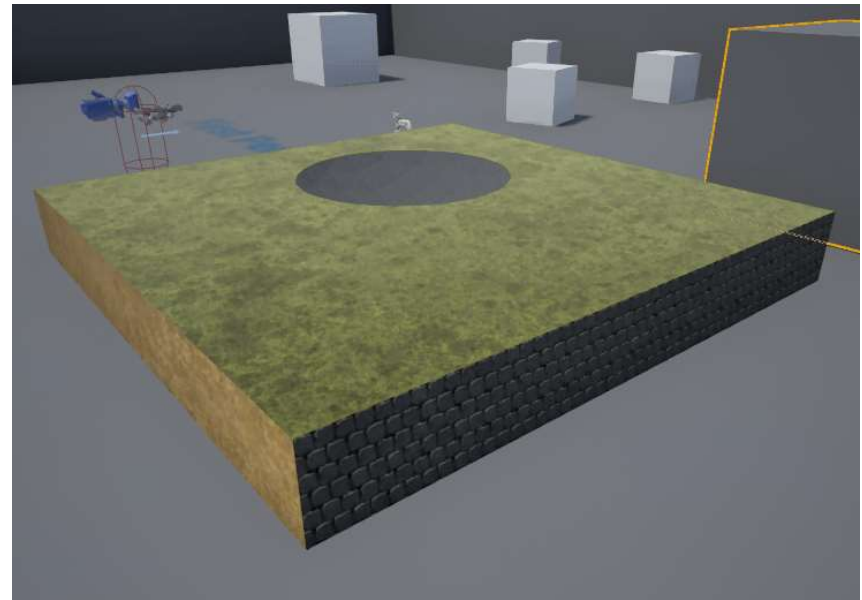
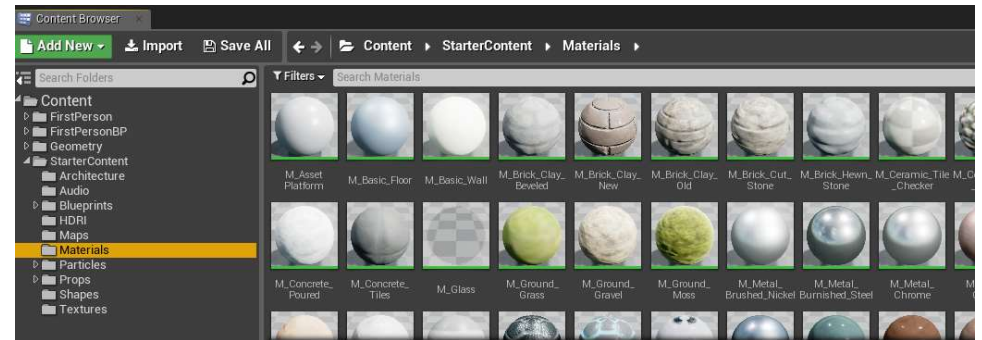
- Click “Build” button (default is “All”)
- Test your first level by click on “Play” button



or right click in level and choose “Play from here”

Texturing

- Decorating your BSP brushes!
- Select any brush and click the content browser
- Check “Materials” to filter only materials
- Pick a material and drag it from content browser to a brush you created
- Rebuild the level to see the result



Static Mesh

- They are more recommended for modern day game development
- Take advantage of hardware accelerate features of display card
- Fine crafted meshes with textures for decorating level

Marketplace

- You can purchase and download some prefab from Marketplace under EpicLauncher
- Click on Environment category and choose “Infinity Blade: Grassland” then choose “Add to project” and choose your project

The screenshot displays the 'Content Detail' page for 'Infinity Blade: Grass Lands' on the Epic Games Marketplace. The interface includes a top navigation bar with links for '< Back', 'Content Detail', 'Home', 'Categories', 'New Content', and 'Submit Content', along with a search bar. The main content area features a large image of the game environment, a title 'Infinity Blade: Grass Lands' with a green checkmark, and a release date of 'September 9, 2015'. Below this, the 'Average Rating' is shown as '(26)' with five stars. A descriptive paragraph follows: 'Infinity Blade: Grass Lands is the earthy citadel adorned with stone set pieces and beautiful props. It's one of three environments that ship with the Infinity Blade Collection, a massive suite of content designed for high-quality mobile experiences.' To the right, a section titled 'Supported Platforms' shows a Windows icon, and 'Supported Engine Versions' are listed as '4.9 - 4.10'. A 'Share' section includes Facebook and Twitter icons. A prominent yellow 'Add To Project' button is located below the main image. At the bottom, a navigation bar shows 'Description' and 'Game Development Using UE4'.

< Back Content Detail Home Categories New Content Submit Content Search Content...

Infinity Blade: Grass Lands

Epic Games - September 9, 2015

Average Rating: (26) ★★★★★

Infinity Blade: Grass Lands is the earthy citadel adorned with stone set pieces and beautiful props. It's one of three environments that ship with the Infinity Blade Collection, a massive suite of content designed for high-quality mobile experiences.

Supported Platforms

Supported Engine Versions
4.9 - 4.10

Share

Add To Project

Description Game Development Using UE4

Static Mesh

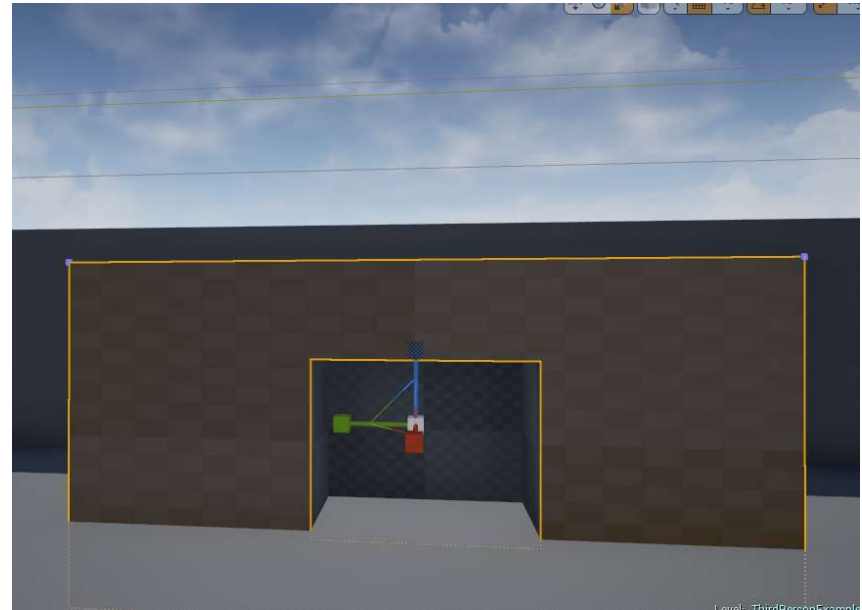
Click into InfinityBlade
GrassLands folder

1. Choose
“Environments/plains/”,
there are various static
mesh folder under
different description. (or
use filter with “static
mesh” checked
2. Select a static mesh
3. Right click at point you
wish to add mesh in edit
viewport and choose
“Place Actor:... (can also
just drag)



Editing BSP

- BSP has the great property that allow you to do fine tuning of material in editor
- First build an Add BSP brush, and a subtract BSP brush as right
- Select the outer brush, note that the front face is selected



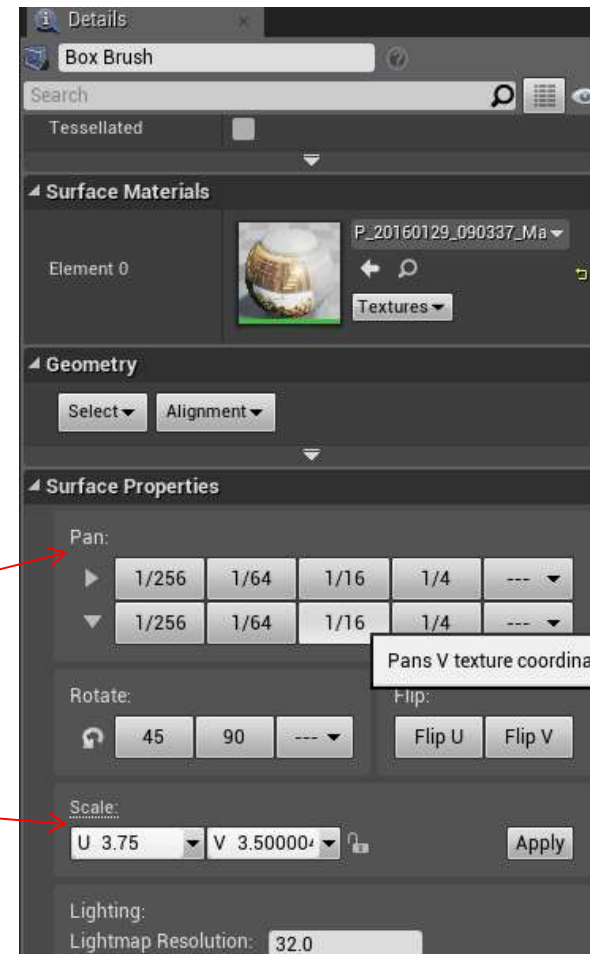
Editing BSP

- In content browser, click “import” and select your image to be mapped on the model
- A material should now be created using the name of the image
- With the front face still selected, drag and drop the material on the face



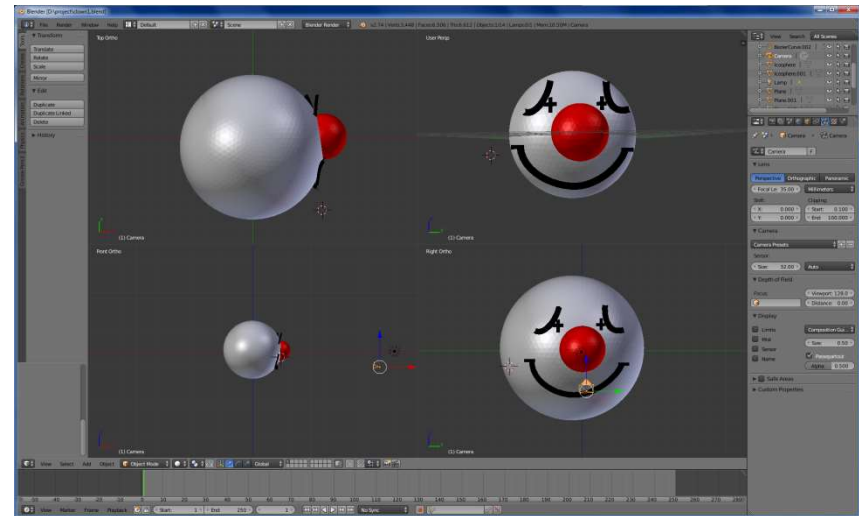
Editing BSP

- The details pane should now have the settings similar to on right
- You can adjust the surface properties to fit your texture to the surface
- Usually pan, Scale will be used



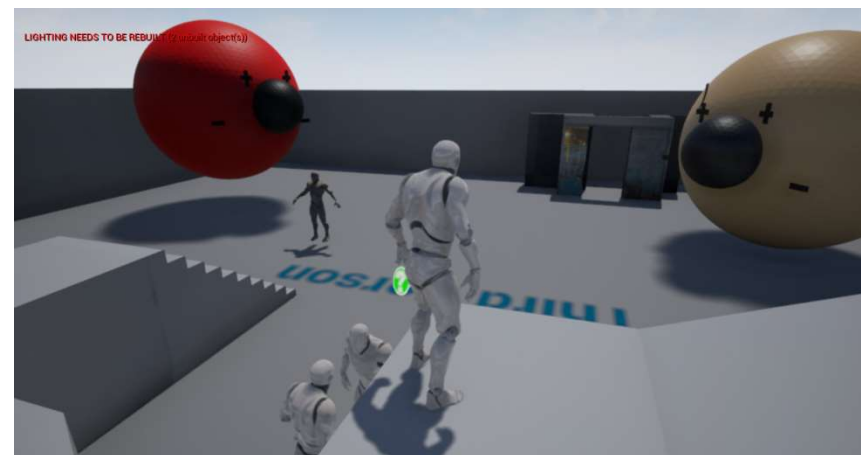
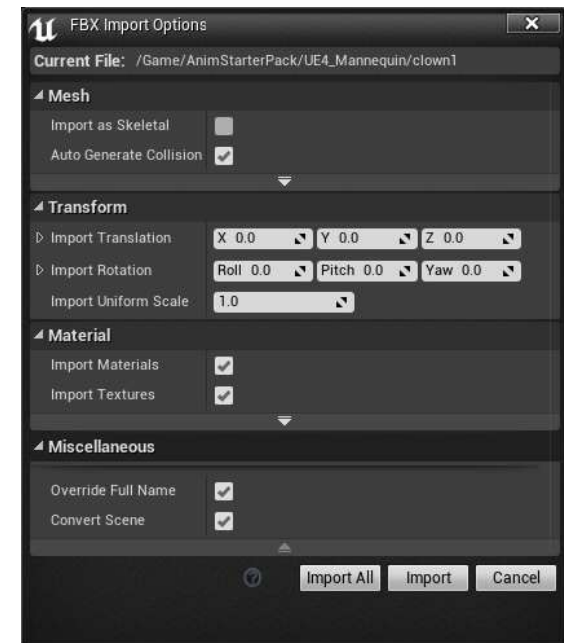
Static Mesh

- On the other hand, you can also use static mesh
- Modeling package is expected to be used to build your model first
- We use Blender here
- First build your own model
- Exported to FBX file



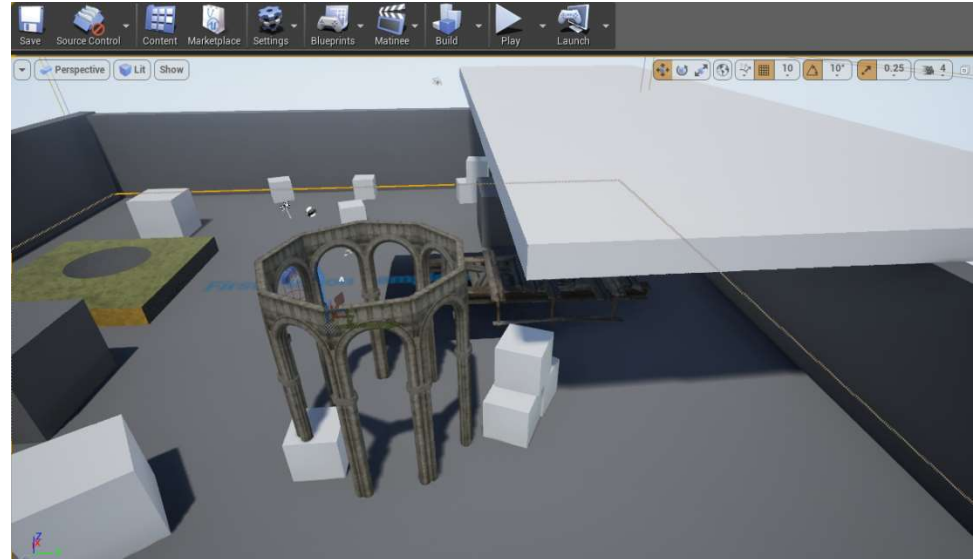
Static Mesh

- In content browser, choose “import” and browse to the fbx file
- In Import menu, uncheck the skeletal mesh option as we only incorporating static mesh
- Drag and drop the imported mesh to any place in level



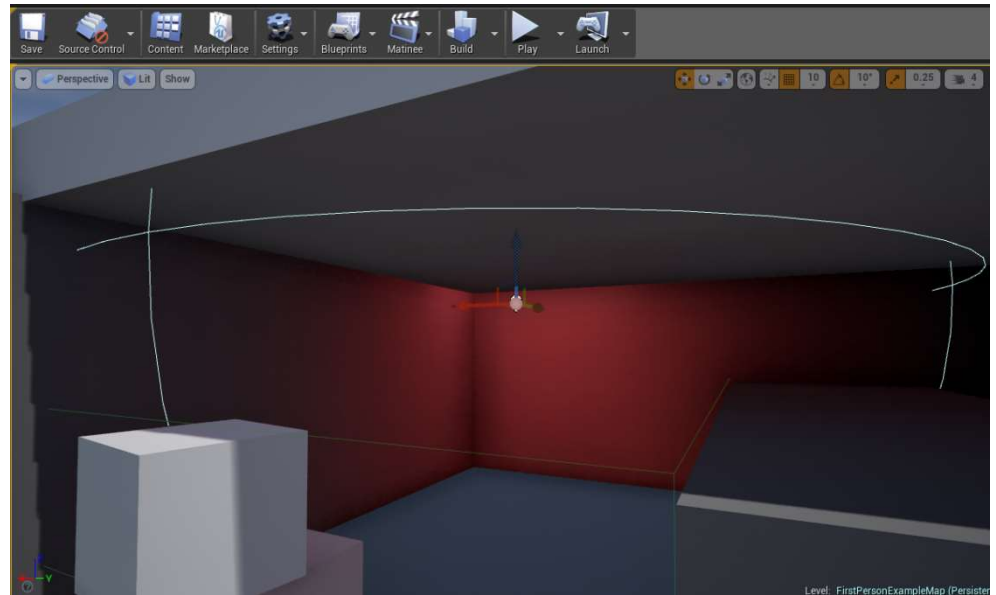
Lights

- Lighting is important in atmosphere building
- Build a ceiling to shield the sunlight from sky first




Lights

- Right click at the point you wish to add light and choose “Place Actor\Add Light(point)”
- Use the transform widget to move it to a desired position
- Modify the intensity at its properties tab for desired lightness
- Rebuild level to check

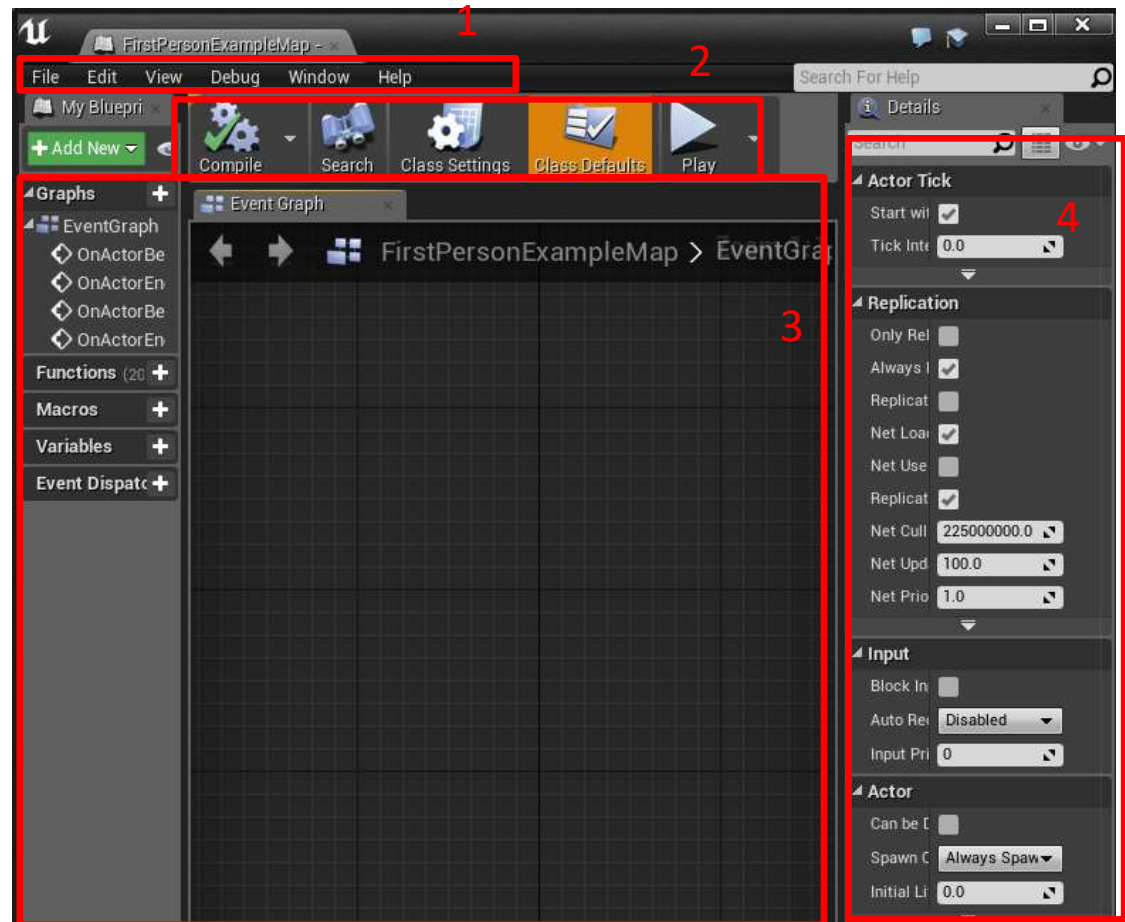


Scripting Gameplay

- To facilitate interaction of player with in game elements is vital to gameplay
- In a game engine, this is called “Scripting”
- Scripting can be done in UE4 through Blueprint, a visual scripting system
- Activate in UE4 by pressing the  button in level editor menu

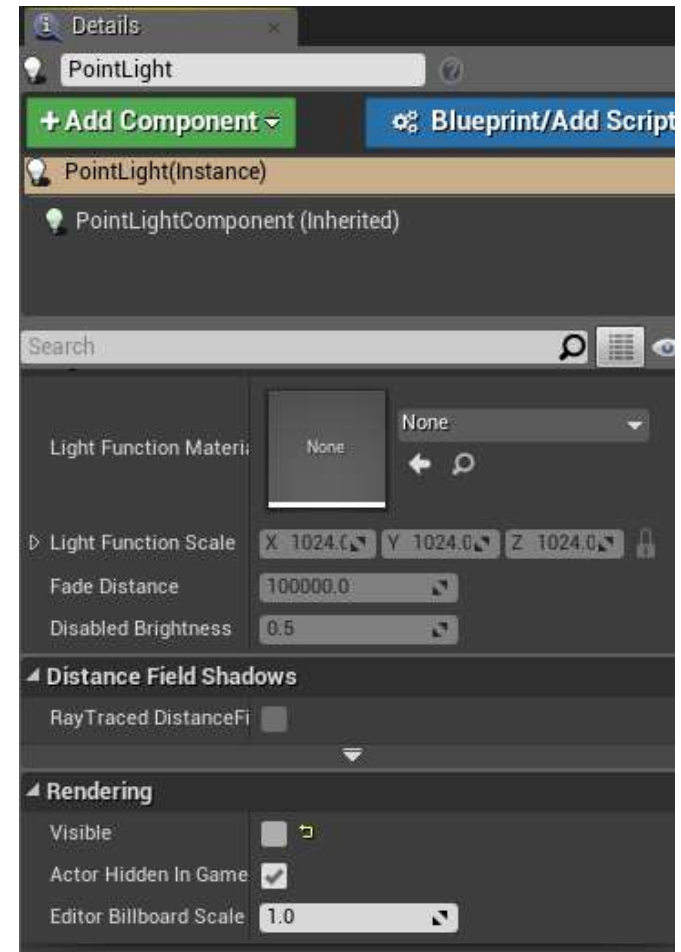
Blueprint interface

1. Menu Bar
2. Tool Bar
3. Graph Pane
4. Details Pane
5. Sequences Pane



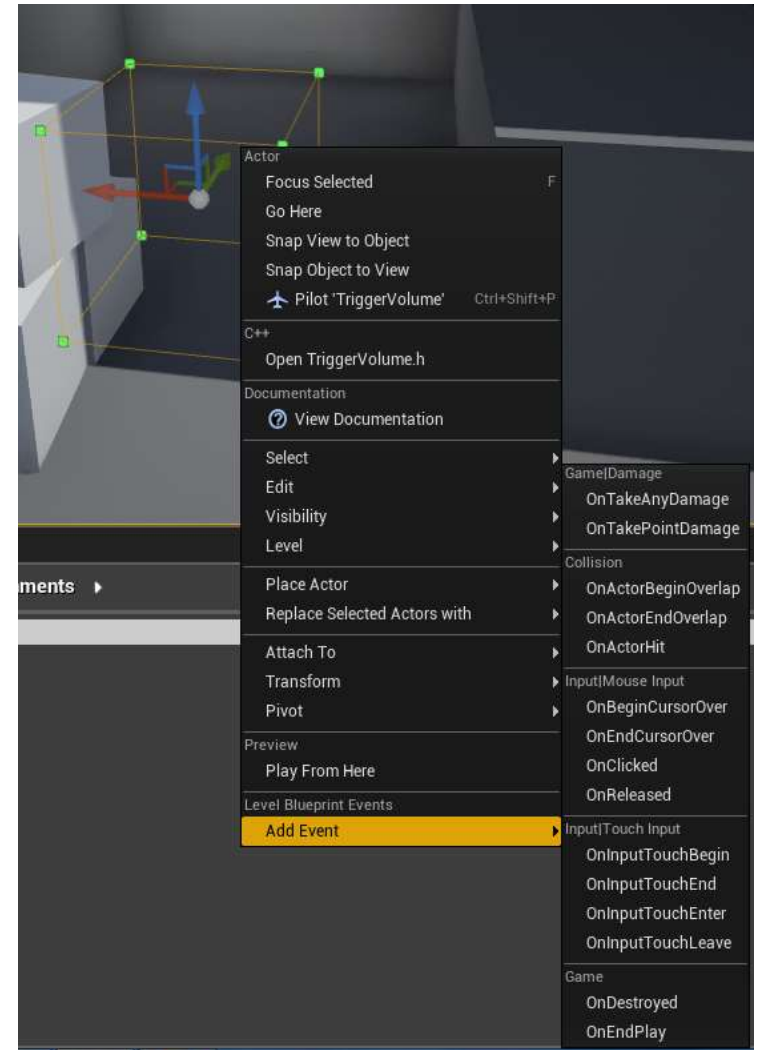
Scripting

- Let us script a level design that when the player touch an object, a nearby light will turn on
 1. Select the light we just created by clicking on it and uncheck “Visible” under Rendering to turn light off
 2. Under the Modes/Volume tab, choose Volumes and scroll down to choose “Trigger Volume”



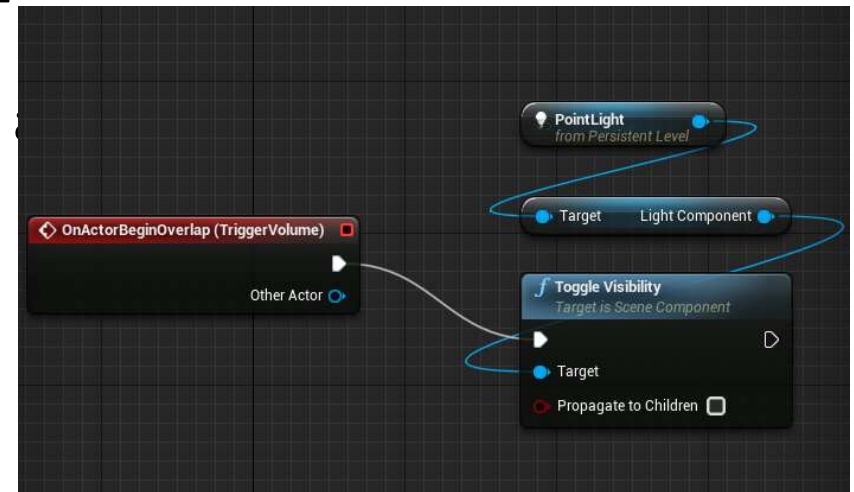
Scripting

- Let us script a level design that when the player touches an object, a nearby light will turn on
2. Under the Mode tab, choose Volumes and scroll down to choose “Trigger Volume”
 3. Just drag the icon to the viewport, and move to a point you wish to switch the light on
 4. Right click on the TriggerVolume and select “Add Event/OnActorBeginOverlap”
 5. Blueprint will be opened



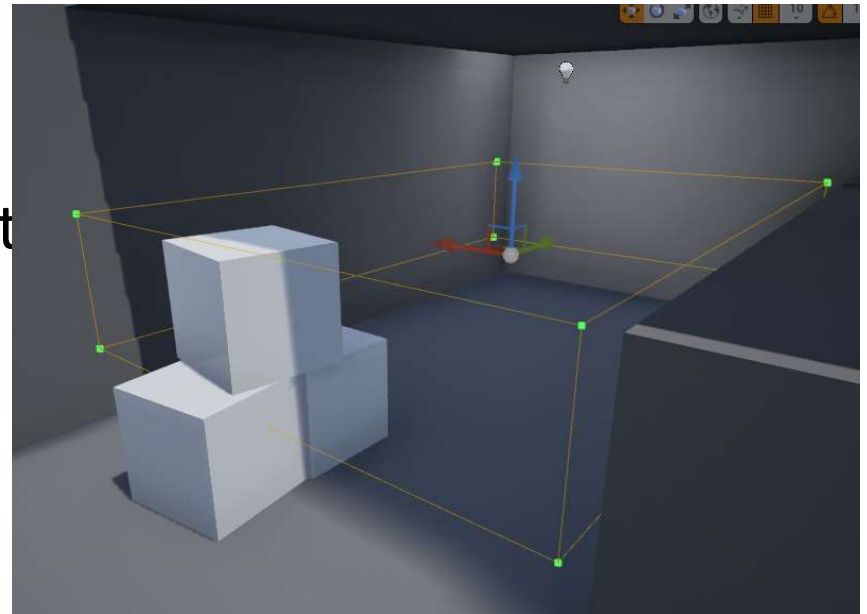
Scripting

- Go to editor and select the pointlight1 again
- Right click in the Blueprint pane and select “Add Reference to PointLight1”
- Click and drag from the blue pin on the **PointLight1** reference node into empty area to let menu pop up
- Search for visible and choose “ToggleVisibility”
- Connect the node of OnActorBeginOverlap to “Toggle Visibility”, the result should like that on right



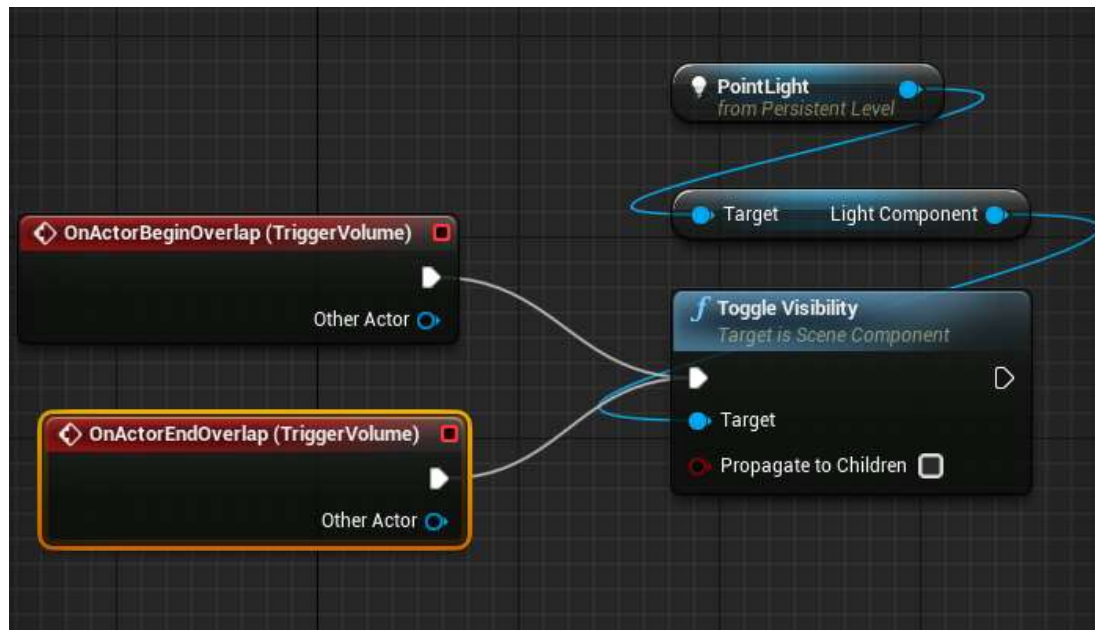
Scripting

- Adjust the trigger volume so that it cover the entire volume that you wish to control the light on whenever player move into
- Now we want to turn off the light when we leave the area
- Right click on TriggerVolume in level editor
- Select AddEvent under Level Blueprint Events, choose OnActorEndOverlap



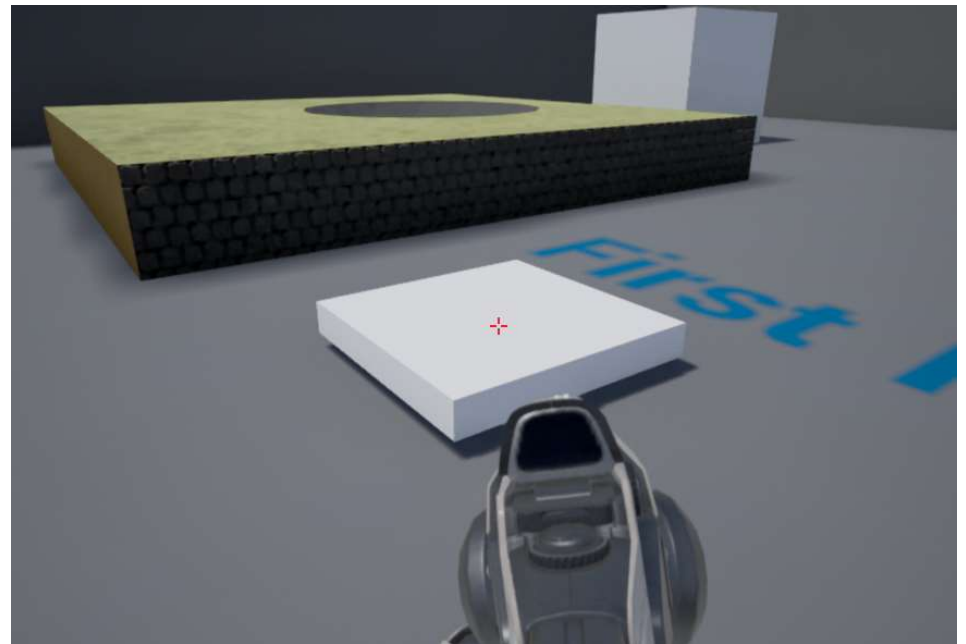
Scripting

- Click output pin of OnActorEndOverlap and connect to ToggleVisibility as previous
- Final Blueprint should be similar to below



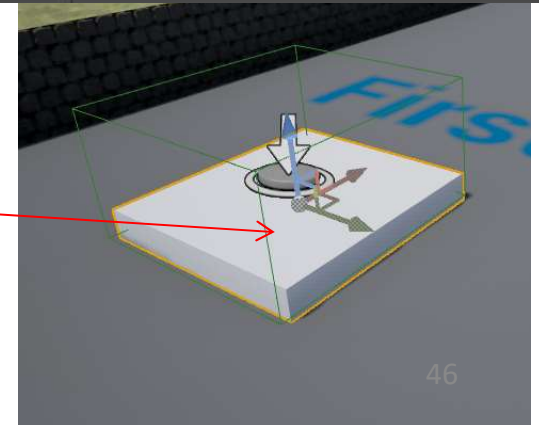
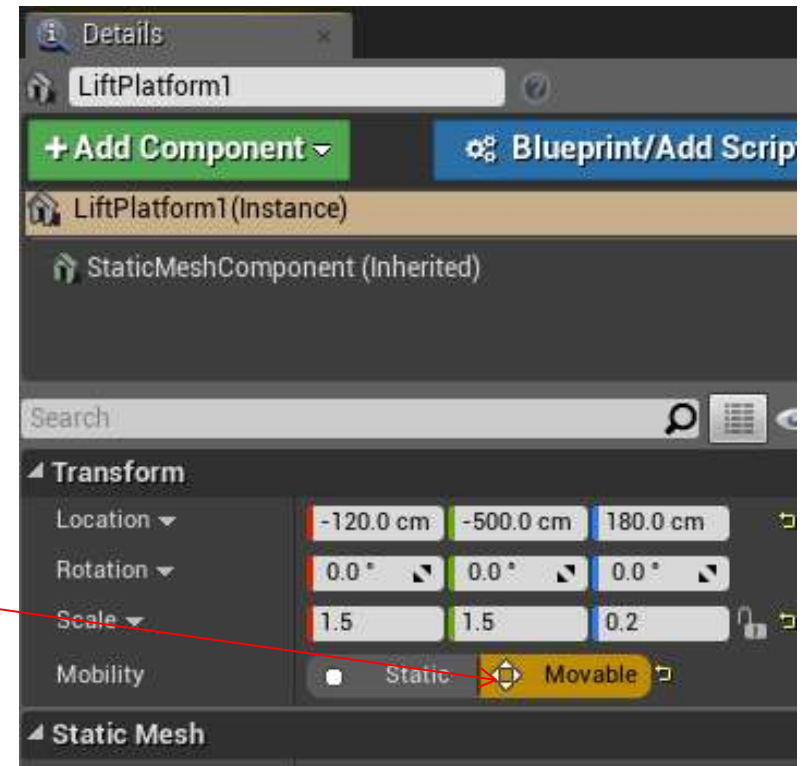
Moving Platforms

- In many games, lifts or moving platforms are vital to the fun of playing
- We will create a lift-like platform in the following tutorial
- Sequencer, which is the animation system will be used



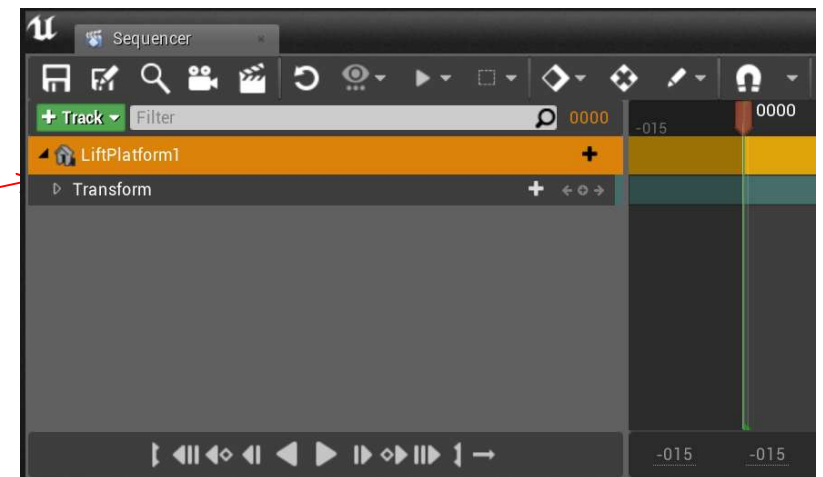
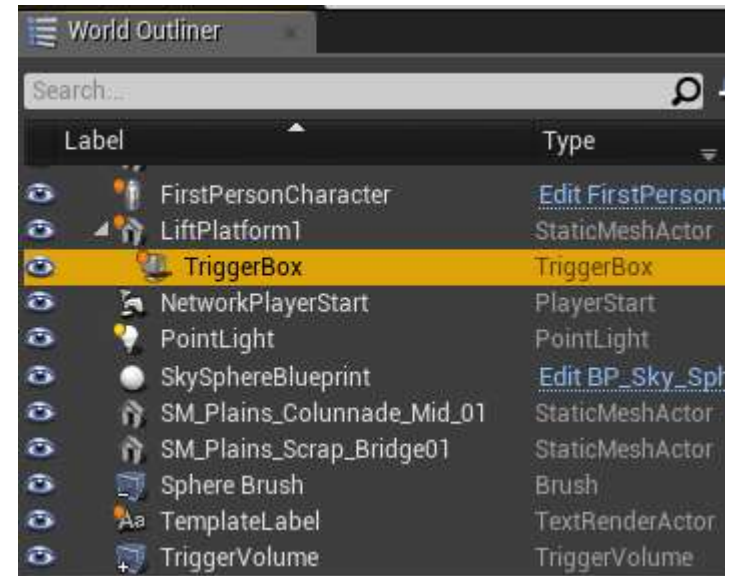
Moving Platforms

- First create a static mesh and make it into a platform like
- Rename it to “LiftPlatform1” under Details tab
- Also change the Mobility to “Movable”
- Right click in the level and choose “PlaceActor/ Trigger/ Box Trigger” to add the trigger
- Adjust the trigger so that it just cover the platform we just created (as right)



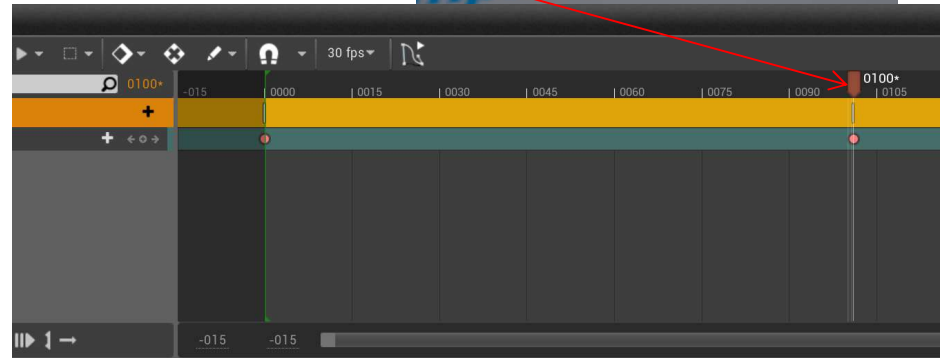
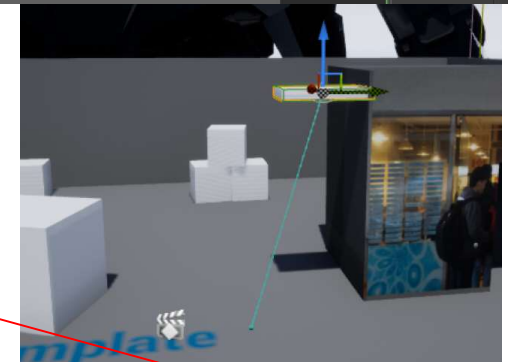
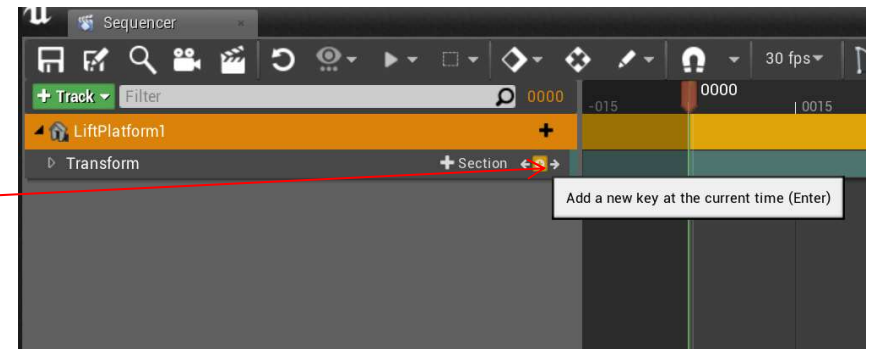
Moving Platforms

- In the world Outliner, drag and drop the triggerBox entry under LiftPlatform1, this will make them moving as a whole
- Select “Cinematics/Add Level Sequence” and enter “LiftSequence”
- Select “LiftPlatform1” and drag & drop into left pane of level sequence



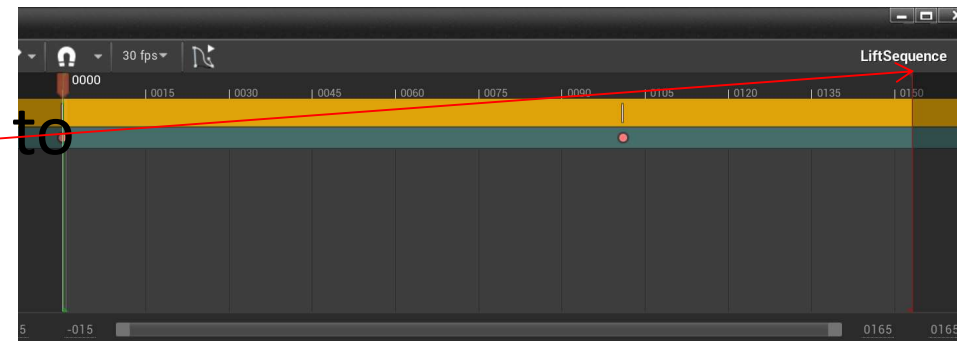
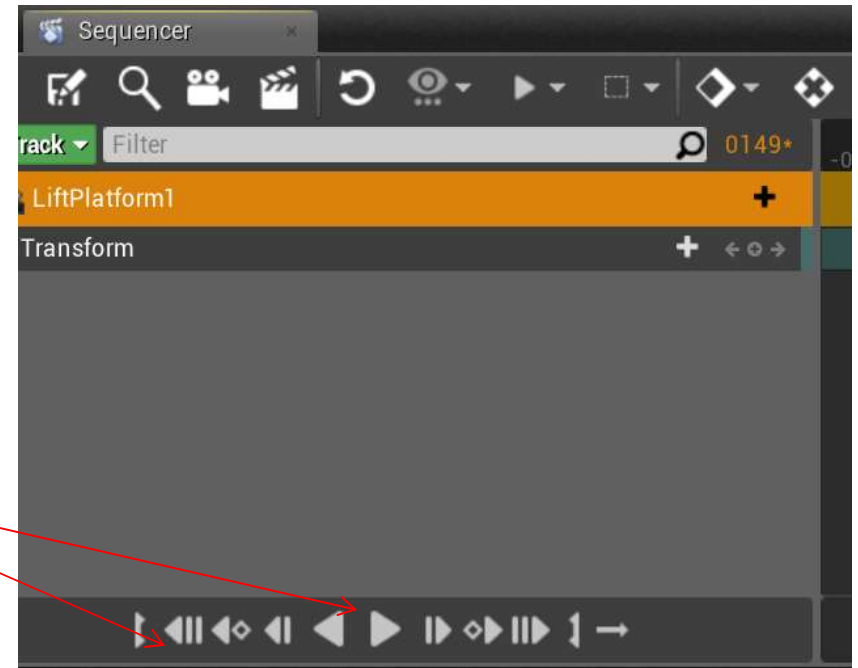
Moving Platforms

- Move the mouse to over the centre button under '+', click at 'o' button to add the first key frame
- Drag the play pointer to frame 100, now select the LiftPlatform1 in editor and move it to a higher position
- Add a new key frame by clicking at the same button



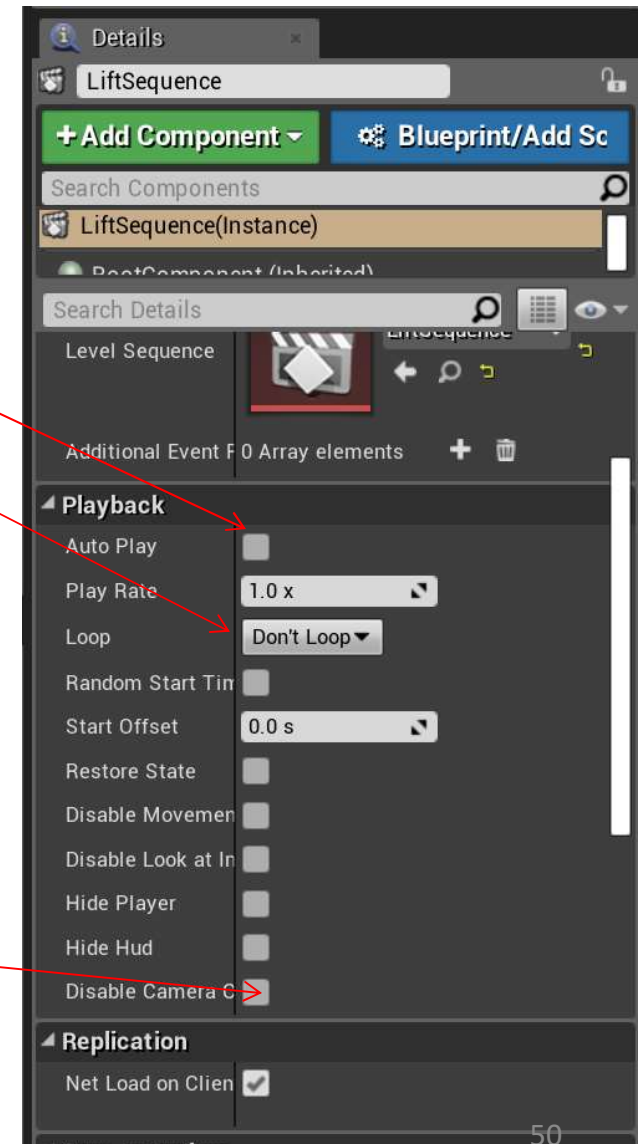
Moving Platforms

- You can preview the sequence using the control in sequencer
- Click on “To front” button and click “play” to see the effect
- Note that you may see the playing goes over frame 100 as we have longer play area
- Drag the play range handle to frame 100



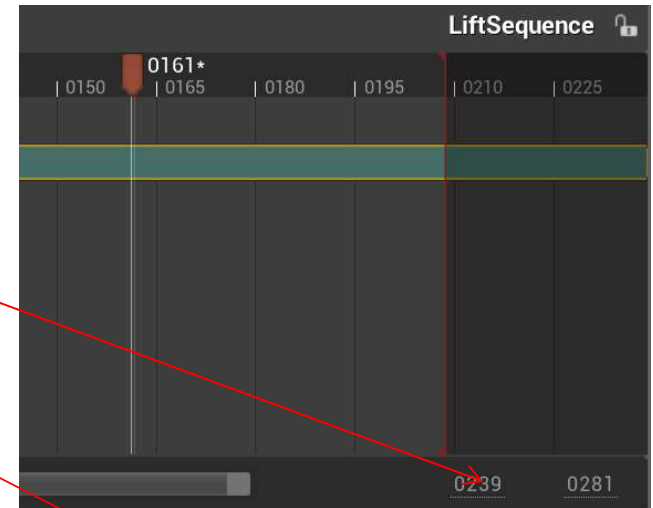
Moving Platforms

- Click on the level sequence in editor and check the “Auto Play” and choose “Loop indefinitely”
- Play in editor to see the effect
- The sudden flashback of platform to the start position is no good
- We want it to return from top to bottom smoothly



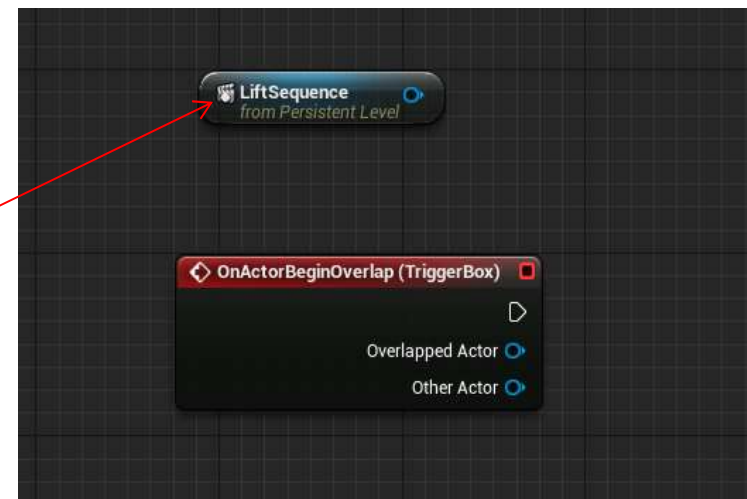
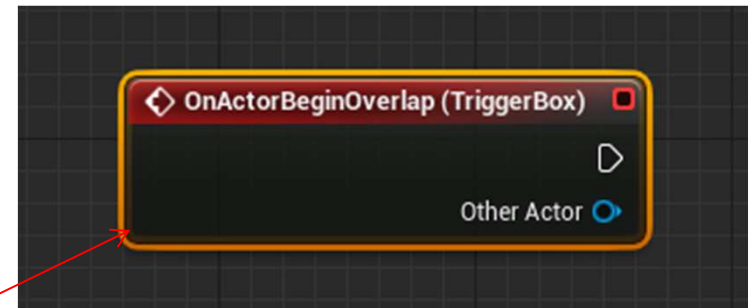
Moving Platforms

- First drag the view range to right so that it covers frame 200
- Drag the play handle to frame 200 and add a key frame there
- Click on frame 1 and right click to choose “copy”
- Click on frame 200 and right click to “paste/transform”
- Play again to see the effect
- Now clear the “autoPlayback” and choose “don’t loop” in levelsequencer



Moving Platforms

- Close the level sequencer
- Select Triggerbox in level editor
- Open Level Blueprint in level editor tab
- Right click in Blueprint pane and type “overlap” (look into “Add event for triggerbox”)
- Select “OnActorBeingOverlap”
- Select “LiftSequence” in level editor
- Back to Blueprint and right click and select “Create Reference to LiftSequence”



Moving Platforms

- From LiftSequence blue output hole, drag and drop off and type “player”, select the “Get Sequence Player”
- From SequencerPlayer output drag and drop and type play to choose “Play(SequencePlayer)”
- Connect output pin of OnActorBeginOverlap to input pin of Play
- Lift created!

