Logbook (2020)

Entry	Date	Rating
Tutorial: Getting Started in Unreal Engine (Part 1) M	02.01	Exemplary
Tutorial: Getting Started in Unreal Engine (Part 2, 3) M	03.01	Exemplary
Tutorial: Getting Started in Unreal Engine (Part 4) M	05.01	Incomplete
Accomplish the Expose M	07.01	Exemplary
1st Meeting M	10.01	
Tutorial: Getting Started in Unreal Engine (Review) M	03.02	Exemplary
Tutorial: Exploring Blueprints (Part 2, 3) M	04.02	Exemplary
Tutorial: Exploring Blueprints (Part 4) M	05.02	Exemplary
Rewrite the Expose (Specify details) M	06.02	Exemplary
Tutorial: Unreal Academy 2019 (4 Parts) M	07.02	Exemplary
2nd Meeting M	10.02	
GitHub and GitKraken M	11.02	Exemplary
C++ Basic Reviewing M	12.02-14.02	Exemplary
Getting Started with Pixel Streaming M	17.02-20.02	Exemplary
3rd Meeting (Mainly about Assistant Work) M	02.03	
Finish Git Tutorial Video M	03.03-04.03	Exemplary
Add LFS part to Git Tutorial Video M	05.03-06.03	Exemplary
C++ Further Study	09.03-13.03	Exemplary
UE4 Further Study	30.03-03.04	Exemplary
Car Configurator: General Frame	06.04-17.04	Exemplary
Prepare for the IVW course	20.04-24.04	Exemplary
Car Configurator: Refine	27.04-08.05	Exemplary
Bachelor Thesis: Draft	11.05-22.05	Exemplary
Bachelor Thesis: Refine Draft	25.05-29.05	Exemplary
Final Examination and Final Submission Preparation	01.06-08.06	Exemplary
Preparation for Colloquium	15.06-19.06	Exemplary

02.01.2020

Tutorial: Getting Started in Unreal Engine (Part 1)

Reflections: Understand well.

Part 1: Your First Hour with Unreal Engine

1. Welcome to the launcher

Epic Games Launcher

Home: see hottest games available from the Epic Games store

Store: list games and purchase games

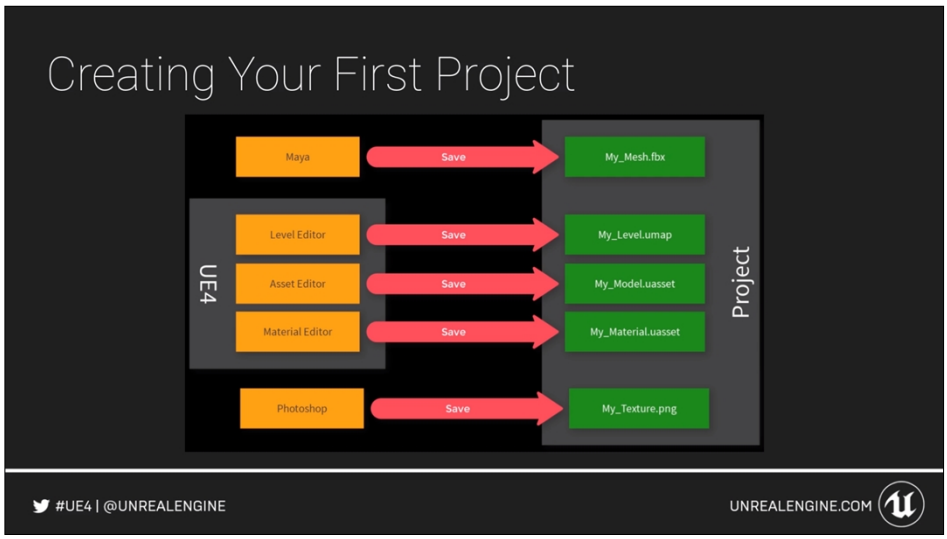
Library: manage games you have purchased

Friends: networking tools, build the list of friends and co-workers

Unreal Engine: developer area of the launcher

- Unreal Engine: lists everything new and exciting, focus on development
- Learn: suggested learning resources, hotlinks to documentation
- Marketplace: free and paid content
- Library: engine version, project list

2. Creating your first project



Useful hotkeys:

- W, E, R: move, rotate, scale
- G: toggle game mode
- Ctrl+1: store a bookmark; 1: retrieve bookmark
- F11: toggle immersive mode
- F: focus selected object

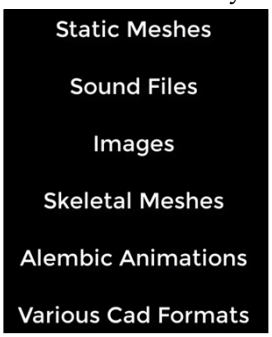
3. Working with content

Four methods:

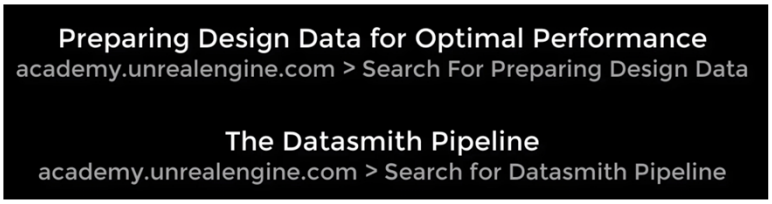
- Starter content

Marketplace
Migrate from other projects
Import content directly

Import content directly includes:



To learn more:



4. Building your first level

New folder: Levels

In the folder: Right-click → Create Basic Asset → Level

Example:

SM_Exo_Bridge: select ground

Add light: Direction Light

Alt key to duplicate ground

Modes: choose Player Start; add Atmosphere Fog; add Sphere Reflection Capture

5. Visual Control

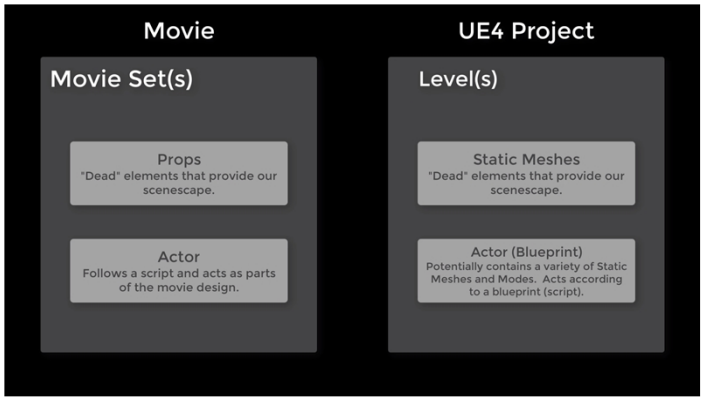
Sky Light

Post Process Volume: Details → check Infinite Extent

BP_Sky_Sphere

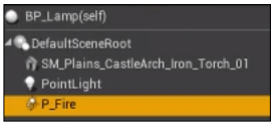
Put in one folder: !World Lighting

6. Actor



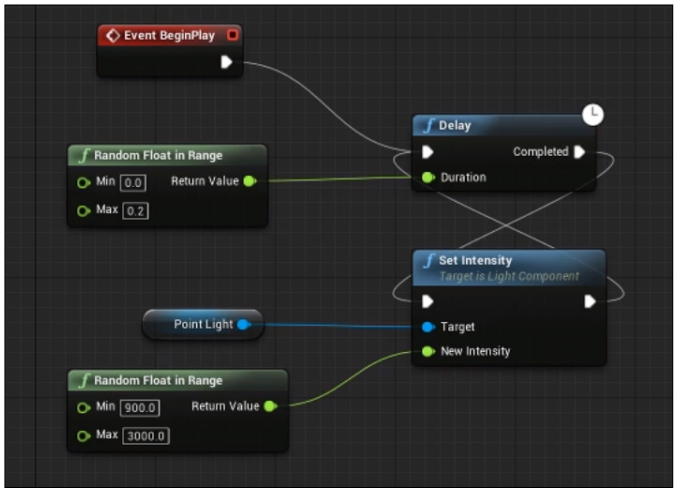
New folder: BP_Actors

Right-click: Blueprint Class → Actor → name BP_Lamp



After accomplish: compile and save

Event graph:



3.1.2020

Tutorial: Getting Started in Unreal Engine (Part 2 & 3)

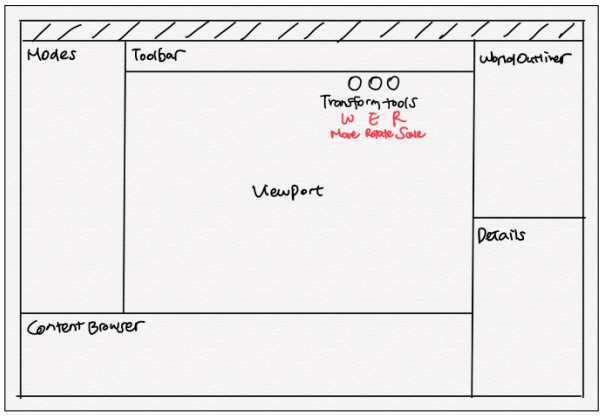
Reflections: Understand well.

Part 2: Introducing Unreal Engine

1. Editor Basics Introduction

Overview: Editor layout customization
Viewpoint: Options navigation
World Outliner: How it works
Detail Panels: What it does features
Modes Panel: Available modes; place modes
Content Browser: What is it for customization
Main Toolbar: Shortcuts to common actions

2. Editor Basics Overview



Window → Reset Layout

3. The Viewport

Transform Tools: WER
Change view: Use mouse or WASD QE

4. World Outliner

The World Outliner is a hierarchy of all the items in a level, and it provides tools for selecting, searching, deleting, parenting, grouping, and displaying/hiding items.

5. Detail Panel

The Details panel shows items selected, components, and their properties. On the Details panel, you can change property values, add components, and convert chosen items to Blueprints. You can also use multiple Details panels to match properties between objects.

6. Modes Panel

The Modes panel gives access to operation such as Place for adding common objects to your scene, Paint, Landscape, Foliage, and Mesh Editing. Being in a particular Mode affects the available options in other panels in the Modes tab.

7. Content Browser

The Content Browser displays and organizes all your content in your project, including geometry, blueprints, and textures. The Content Browser shows the content in your folders, and the view can be filtered to find specific items. You can open up to four Content Browsers.

8. Main Toolbar

The Main Toolbar provides quick access to your project and its settings and provides you with the ability to Build changes such as lighting and navigation. It also offers a quick way for Play-test your project.

9. Editor Basics Conclusion

In the previous section, you learned how to customize and reset the Editor, use snap and navigation in the Viewport, work with items in the World Outliner, adjust properties in Details panel, use Place mode to place objects, and use the Content Browser to organize your content.

10. Settings Overview Introduction

Editor Preferences
Project Setting
World Setting

11. Editor Preferences

Edit → Preferences: Region & Languages; Tutorial

12. Project Settings

The Project Settings allows you to set modes, maps, and rendering options. It enables effects or optimizations for a particular target platform.

13. World Settings

Window → World Settings
With World Settings, you can override settings such as vertical level, gravity, and light settings for the entire current level.

14. Settings Overview Conclusion

Editor preferences: appearance of the graph editors; options when playing in editor; editor options
Project settings: project information; maps and modes; input; rendering

World settings: override game mode; gravity; light mass

Part 3: Comprehending Projects and File Structure

1. Introduction to the Launcher
The Launcher
Community Tab
Learn Tab
Marketplace Tab
Library Tab
Modding Tab

2. Welcome to the Epic Games Launcher
Overview of the Epic Games Launcher

3. The Epic Games Launcher Conclusion
Review of the Epic Games Launcher features

4. Introduction to Projects & Templates
Managing our projects: finding; working
Getting started: creating; templates; starter content; feature packs
Project settings: new project; commonly changed; important

5. Managing Your Projects
Old project with lower edition: open a copy/convert in-place/skip conversion

6. Starting a New Project
Starter content: ~500MB

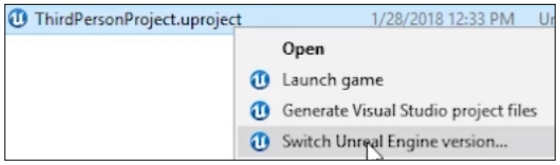
7. The Project Settings Overview
Project Settings → Target Hardware or Maps & Modes

8. Projects & Templates Conclusion
Manage & Create
Templates & Features
Starter Content
Projects Settings

9. Introduction to Important Files and Folders
.uproject File
Project Folder Structure
Cached Downloads
DDC Folders

10. Understanding the .uproject File

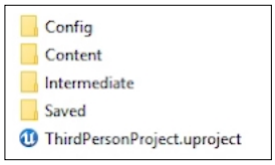
Includes basic settings of how this project interacts with the engine



11. Understanding Project Structure

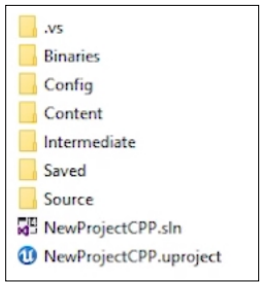
Blueprint Project:

Not necessary: intermediate



C++ Project:

Not necessary: .vs; binaries; intermediate; .sln

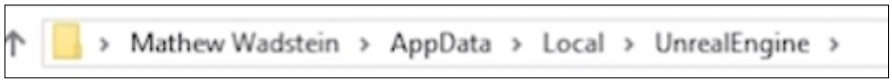


12. Cached Downloads

Vault Cache Location

13. DDC Folders

Derived Data Cache: stores versions of your assets in the specific format that is used by UE4 for the target platform



14. Important Files and Folders Conclusion

05.01.2020

Tutorial: Getting Started in Unreal Engine (Part 4)

Reflections: The part of textures and materials is too hard to understand. I'll watch these parts again later.

Part 4: Building Better Pipelines

Material: https://epicgames.ent.box.com/s/1bykqw0c4do5k4nlprlmjwwhr64x7qwn

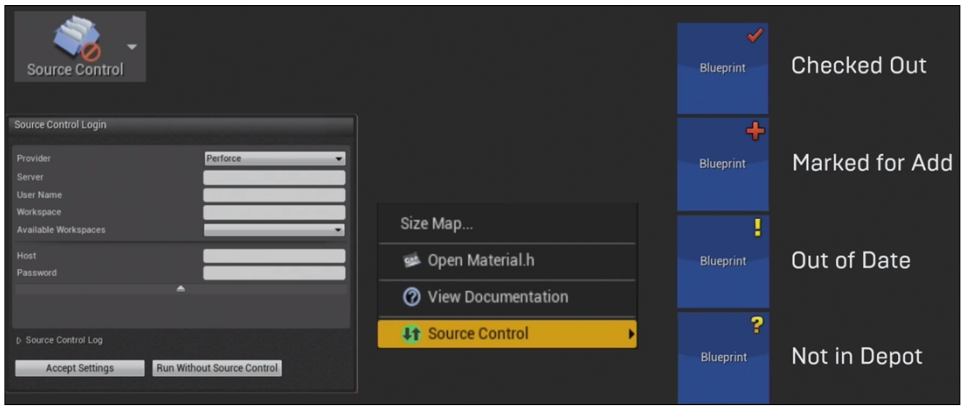
1. Source Control

Overview:

Working with others
Building better 3D meshes & textures
Creating materials & material instances
Performance & optimization

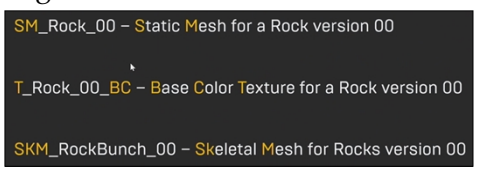
Source control:

Perforce, Subversion: sync data; add/remove data; revert data

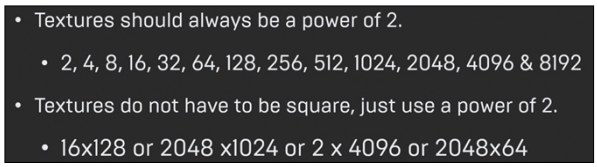


2. Textures

Naming conventions



Texture creation



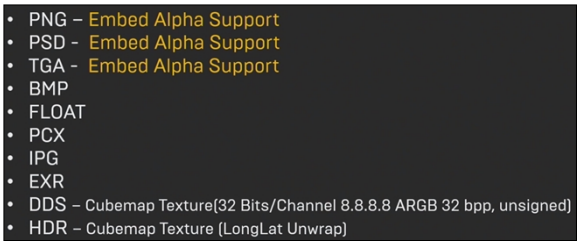
Alpha information

Embed alpha or separate alpha

RGB mask packing

A way to cut down on the amount of textures and memory used

Saving textures



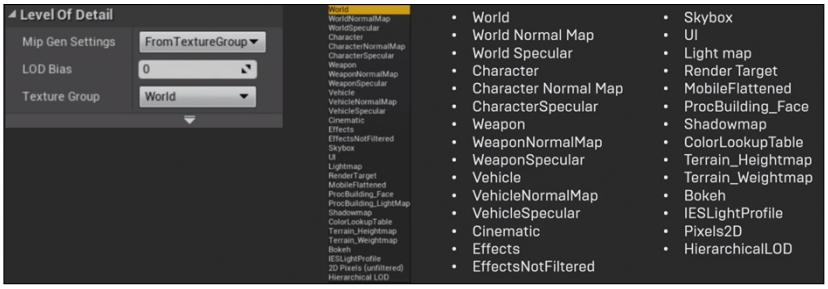
MIP mapping

Level of details for your textures

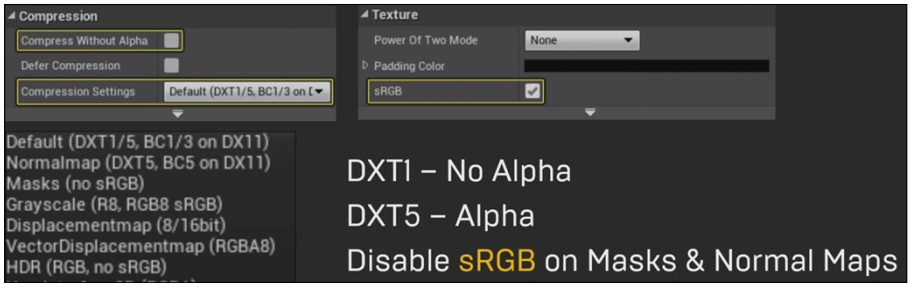
Important textures

2 ways: drag & drop; import button

Texture groups



Texture compression



07.01.2020

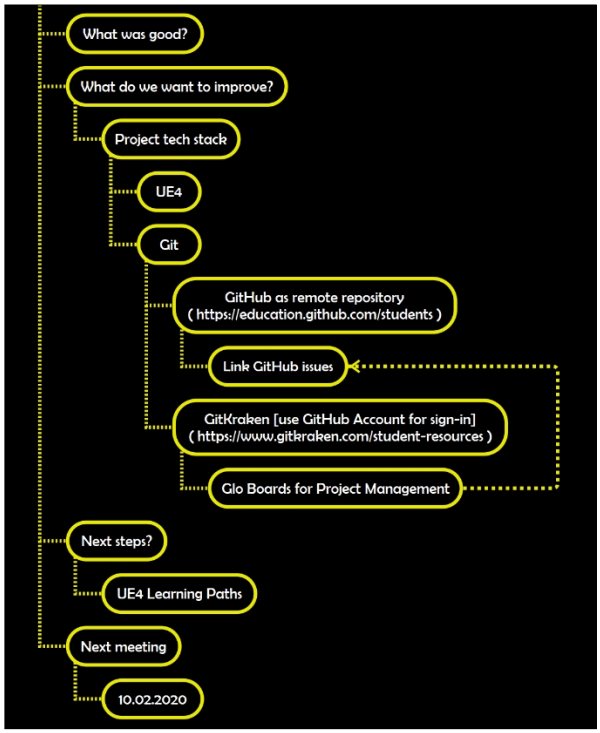
Accomplish the Expose

Reflections: Just outline. The detailed project to be done and the schedule need to be reconsidered later.

10.01.2020

1st Meeting

Reflections: Talk about the following things. I'll prepare these things after examination.



03.02.2020

Tutorial: Getting Started in Unreal Engine (Review)

Reflections: Review all parts of the tutorial. Also watch the last part in detail.

Part 4: Building Better Pipelines (continued)

First review the former knowlwdge.

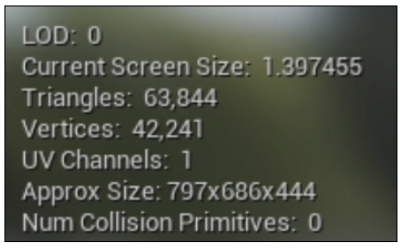
3. Static Meshes

System Units

UE4 use Centimeters for unit of measurement

1 Unreal Unit = 1 Centimeter

Triangle Counts



Material ID's

Each material ID you have will make the model more expensive to render

Pivot Points

Usually 0,0,0

What Are Lightmaps

Used to store complex light and shadow information inside of a texture

Lightmap UV's

0 to 1 UV Space

Collision Meshes

Used in physical simulation

Convex Decomposition

What Are LOD's

Level of Detail: lower triangle versions of the same maps

Help lower the rendering cost of distance objects

Creating LOD's

By hand or programmatically

Limiting Overdraw

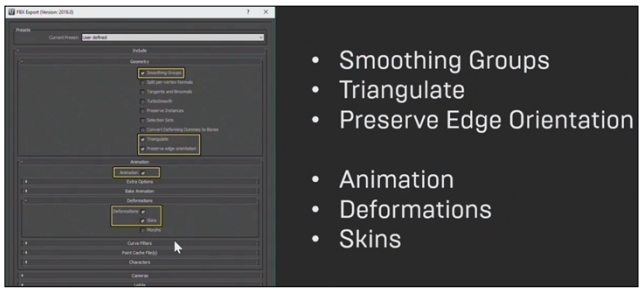
Refers to your GPU having to draw a bunch of transparent or see-through areas of the texture which do not actually display any useful information

4. Exporting and Importing

Exporting Static Meshes

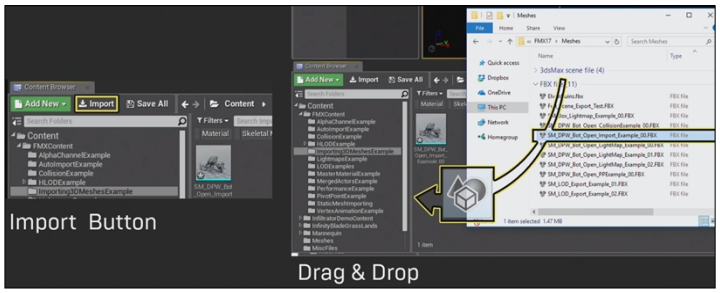
Exporting Skeletal Meshes

Export using the FBX format



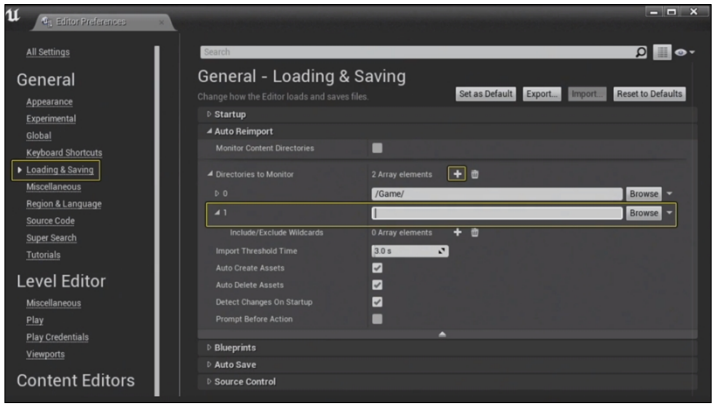
3D Mesh Import

3D Mesh Import Options



Re-Importing Assets

Auto Re-Import



Full Scene Import

Exporting Assets from UE4

5. Materials Part 1

PBR: Physical Based Rendering

Material Domain

Material/Material Instance

Master Material

6. Materials Part 2: Master Materials and Material Functions

Setting up a Master Material for Environment Objects

Setting up a Master Material for Glass

Creating & using Material functions

7. Materials Part 3: Using Material Instances

Part 1:

Parent/Master Materials and their children
Creating & Using Material Instances
Overriding Parent Material Functionality

Part 2:

What is Vertex Animation
Why you would want to use Vertex Animation
How to setup and use Vertex Animation in Materials

8. Texture Streaming

Texture Streaming helps to organize different levels of texture detail as the texture gets further or closer to the camera.

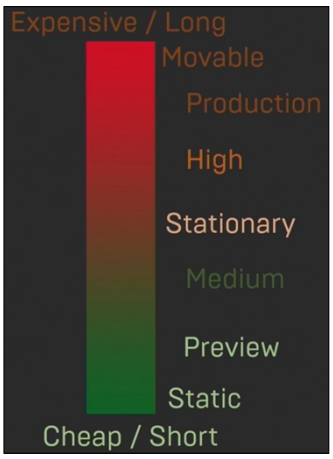
9. LODs & Merging Static Meshes

Level of Detail or LOD
Merge Actor Tool
Hierarchical Level of Detail or HLOD

10. Lighting, Shadowing & Post Process

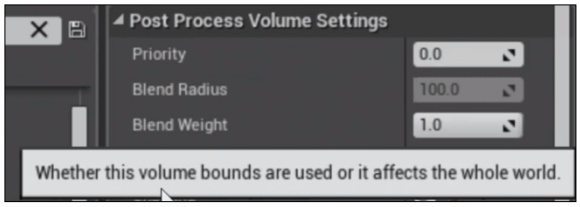
Lighting

Movability: Static; Stationary; Movable



Shadow

Post Process



11. Volumes

Volumes

Lightmass Volumes

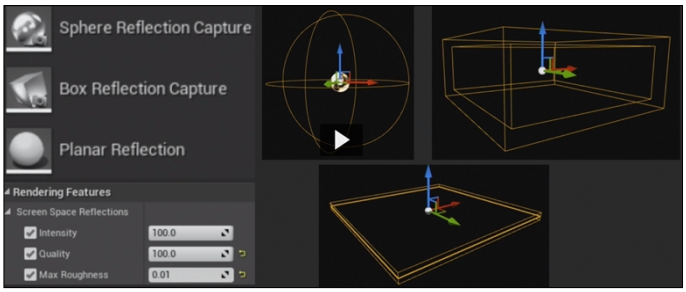
Cull Distance Volumes

12. Reflections

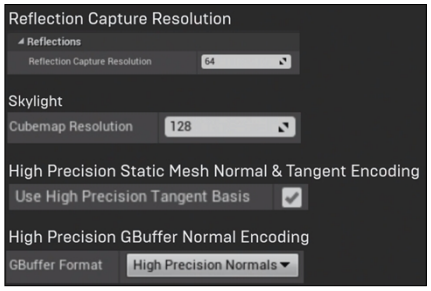
Reflections require extra levels of calculation, and can greatly impact performance.

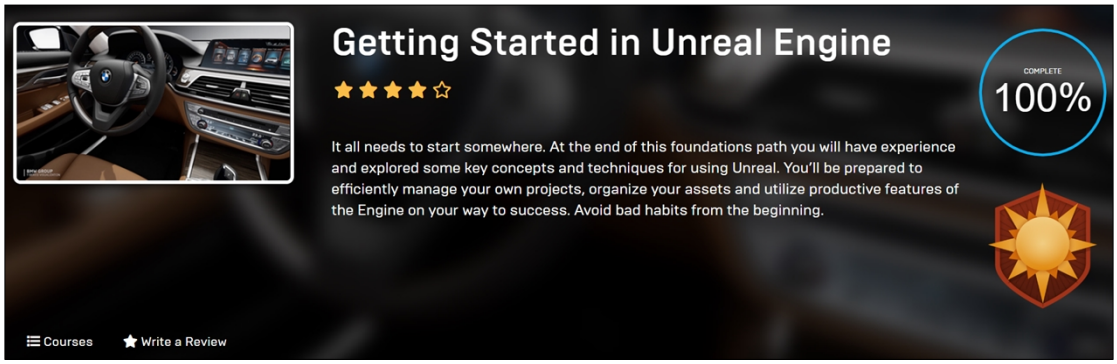
Reflections Actor Types

- Screen Space Reflections
- Sphere Reflections
- Box Reflections
- Planar Reflections



Getting Higher Quality Reflections





04.02.2020

Tutorial: Exploring Blueprints (Part 2, 3)

Reflections: Understand but need to exercise more. The part highlighted in yellow may be useful for my future project.

Part 2: Blueprints - Essential Concepts

Material: https://epicgames.box.com/s/zz9ydnf9bfgdhk771w6z1ss5ldit2dzp

1. Introduction to Blueprint Essentials

Introduce outline

2. What is Blueprint?

Blueprint is visual scripting, object-oriented visual scripting language
Think of Blueprints as a container for content: hold components, script, data
Types: Level/Class Blueprint

3. Creating Blueprints and Editor UI

UI: Viewport; Construction Script; Event Graph

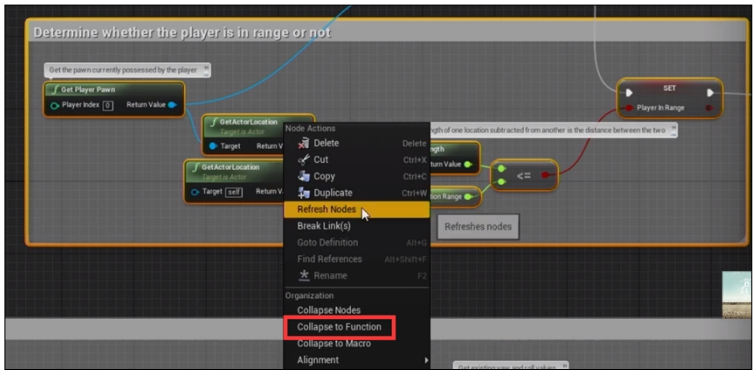
4. Blueprint Components

Default Scene Root
- Static Mesh
- Skeletal Mesh
- Multiple Components

5. Blueprint Graphs

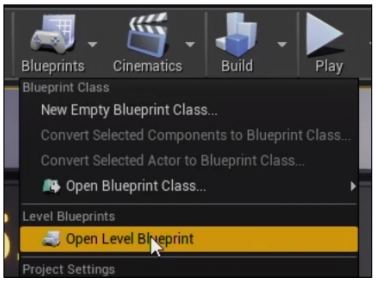
Simple event graph:

Comment: select and push "C"
Collapse to function/Macro/Nodes:



6. Exploring Different Types of Blueprints

Level Blueprint

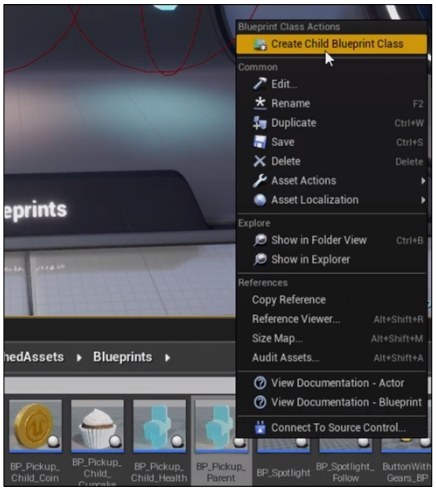


Actor Blueprint

Animation Blueprint

UMG UI

Child Blueprint



7. Caveats to Using Blueprint

- Cost associated with using Blueprint versus native C++ code
  - Avoid doing complex math or heavy operations every frame
  - Blueprint relies on a Virtual Machine (VM)
- C++ is going to be faster and more powerful
- Blueprint is Event Based
- Uses References and possible Circular Dependency

Potential Problems:

References
Casting
Circular Dependency

8. Wrapping up Blueprint Essential Concepts

Recap the topics that were covered

9. Additional References

Blueprint Essentials

- Blueprint best practices
- Editor Cheat Sheet
- Blueprint How-tos
- Anatomy of a Blueprint
- Blueprint Workflow
- Introduction to Blueprints
- Blueprint Editor Reference

Anatomy of a BP

- Variables
- Blueprint Classes
- Actors
- Using Blueprint Nodes
- Functions
- Adding Components to Blueprints
- Blueprint Macros
- Collapsing Graphs
- Construction Script

Going Beyond

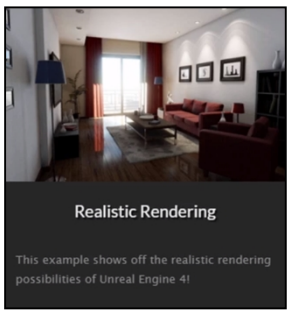
- Blueprint Communication
- Blueprint Macro Library
- Blueprint Projects Quick Start
- Example Blueprints

Reference Guides

- Technical Guide for programmers

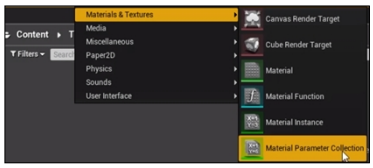
Part 3: Interactive Material Swaps Using Blueprints

Free Demo:



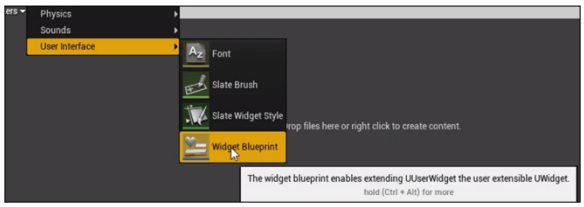
1. Introduction and Material Collections

Add Material Collection:



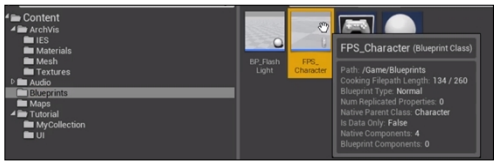
2. Creating a Widget to Control our Material Collection

UI → Widget

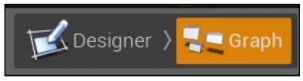


3. Showing the Widget in the User Interface

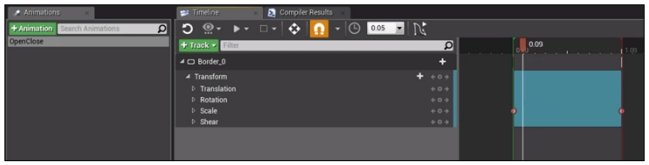
Add to here:



4. Connecting the Widget Buttons to the Material Controls



5. Animating the UI Widget



6. Interactive Material Swaps Using Blueprints Conclusion

Material Collection is shared with every material that uses it.
Material Parameter would only affect that specific material.

05.02.2020

Tutorial: Exploring Blueprints (Part 4)

Reflections: This project is so cool! I need exercise it step by step later.

Part 4: Blueprint for Enterprise

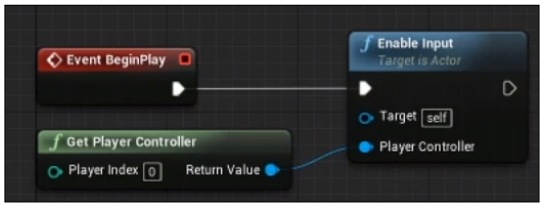
Material: https://app.box.com/s/ahi0tzzlxyuvzca2fgwujsu4qoz8yqfh

1. **Introduction to Blueprint for Enterprise**
2. **What is a Blueprint?**

A Blueprint is an object that inherits functionality its Parent Class, such as an Actor or Pawn, and extends that functionality with assets and node-based programming in a graph.

3. **Blueprint Event Graphs**

Event Graphs hold all the Blueprint code that handles events in your game, such as Begin Play and when a controller button or keyboard key is pressed.
Enable Input:



4. **Level Blueprint**

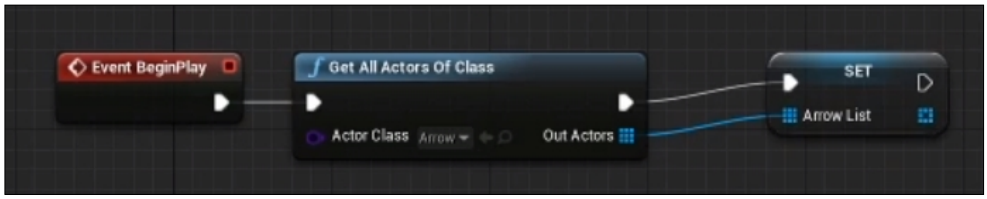
A Level Blueprint, unlike a Blueprint Object, is an Event Graph that is part of your level and allows you to handle events from a master location.

5. **Functions**

A function is a type of container, one that holds reusable Blueprint nodes. A Function can be used by numerous Blueprints and can simplify your code.

6. **Variables and Arrays**

A variable is a container for data such as character strings, numbers and arrays.



7. **DataTables**

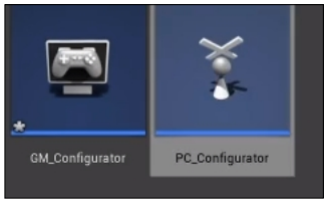
A DataTable (read only) is like a spreadsheet of values that can be fed into inputs in a

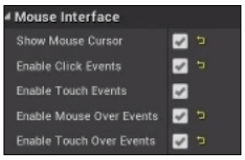
Blueprint.

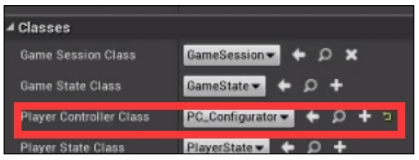
8. HUD (Heads-Up Display)
A HUD (Heads-Up Display) provides text and other feedback to the user.

9. Bringing It All Together
To create a product configurator

10. Game Mode
Target: set up a still camera and enable the mouse
Game Mode and Player Controller Blueprint:







11. Creating the Speaker Blueprint
12. Receiving Clicks on Our Speakers Using the Event Graph
Events are sent to and handled by the Event Graph.
Add and group on click events.

13. Material Swapping Part A: Building a Controller Blueprint
Controller Blueprint: control other blueprints

14. Material Swapping Part B: Initializing Our Blueprints
To communicate between the speakers and speaker logic, we will need to pass references to our objects.

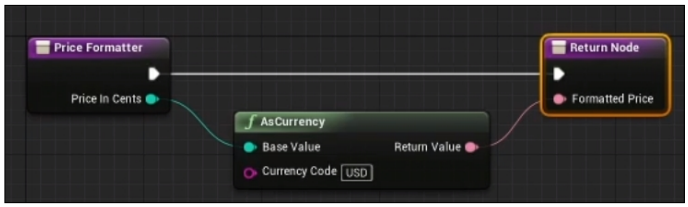
15. Material Swapping Part C: Responding to Control
16. Enabling and Disabling the Cover Screen on Startup
Enable and disable the Screen Cover object using a public Boolean variable

17. Swapping Materials on the Speaker
18. Swapping Materials on the Accents
19. Adjusting the Color of the Cover Screen and Final Troubleshooting
20. Cleanup and Review
Completed

21. Creating a HUD
22. Importing DataTables
Price should be Integer

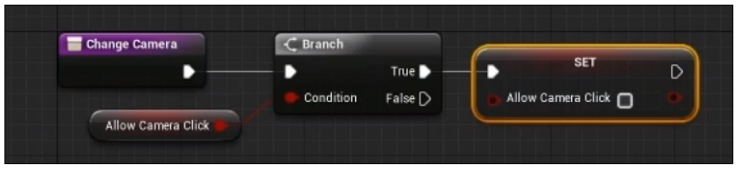
23. Preparing Our HUD for Dynamic Content

24. Setting up Communication to Our HUD
25. Pulling Data from the DataTables
26. Pushing the Data to the HUD
Pure function:

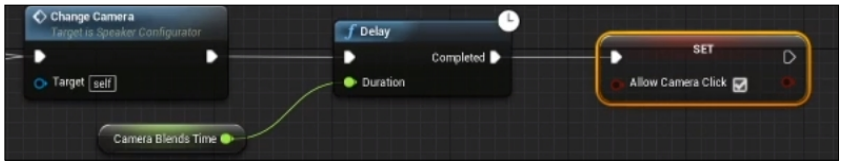


27. Concatenate Sentences from Our Long Descriptions
28. Creating a Quit Button
Add button and use image

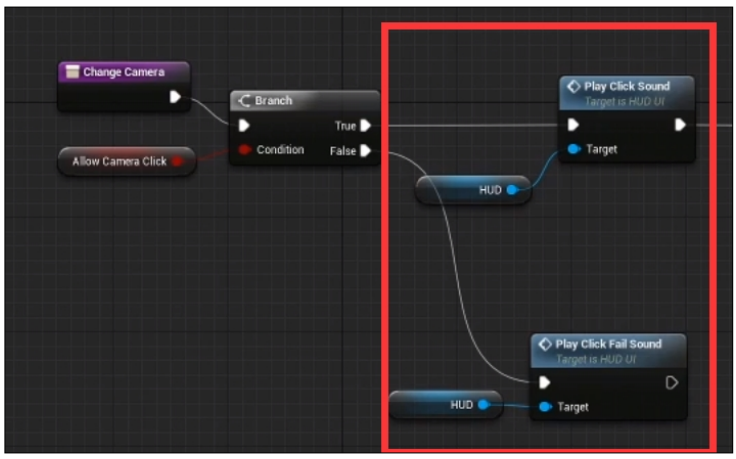
29. Setting up Multiple Cameras for Various Views of Our Speakers
30. Preparing the HUD for User Interaction
31. Developing Logic to Manage the Camera Interaction
Build a gate:



32. Switching the Camera
After change the camera, reset AllowCameraClick (avoid spamming buttons):

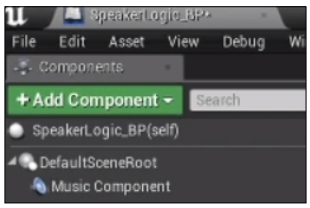


33. Displaying a Camera Name
34. User Feedback with Button Sounds

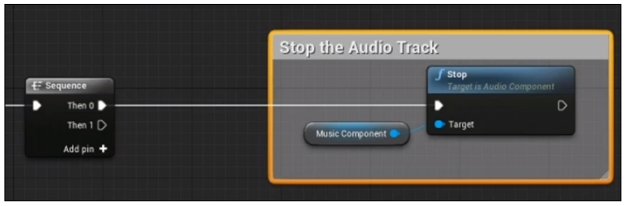


35. Adding Music Tracks and a Simple Music Player

Add Music Component:

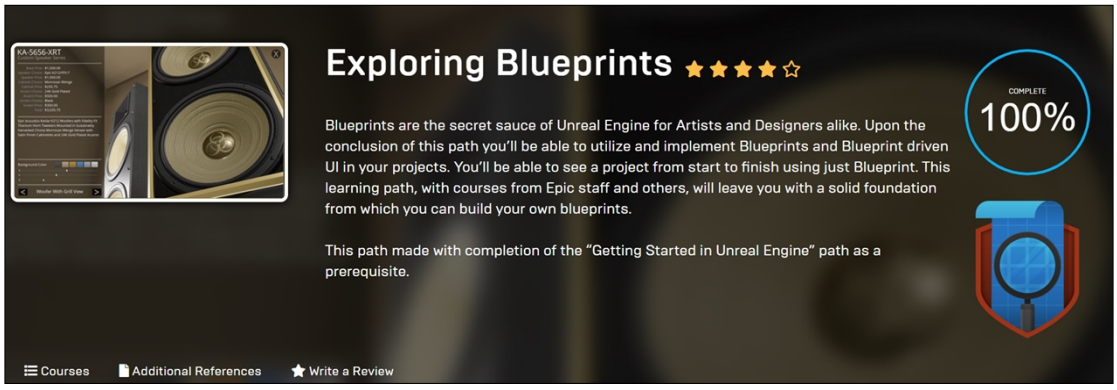


Should first stop:



36. Conclusion to Blueprint for Enterprise

Cool!!!!!



06.02.2020

Rewrite the Expose (Specify details)

Reflections: I rewrite the expose and specify the project what I'd like to make. Also reschedule my timetable.

07.02.2020

Tutorial: Unreal Academy 2019 (4 Parts)

Reflections: Just understand these knowledges. For the C++, I need to review the programming skills later.

Blueprint In-depth

Correct use of BP is important for: future proofing; collaboration; performance

Mastering the Editor with Blueprints

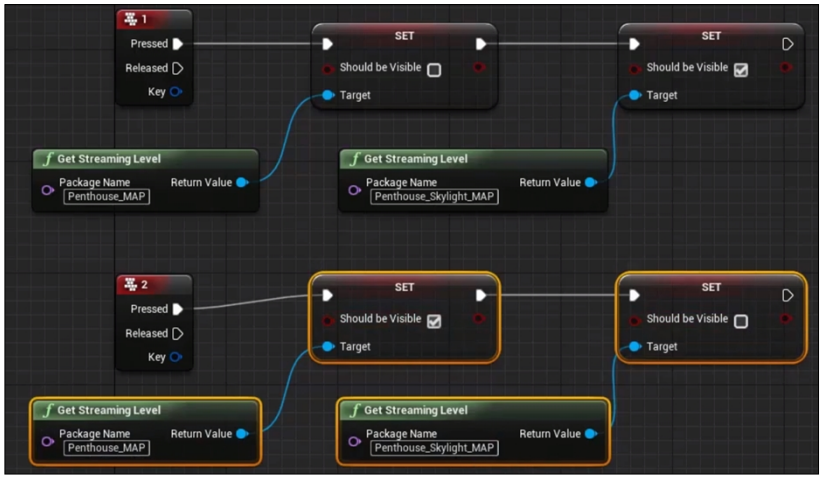
1. Introduction to Mastering the Editor with Blueprints
2. Creating a Template Maker
3. Creating a Viewport Bookmarker

Using Blueprints and C++ Together

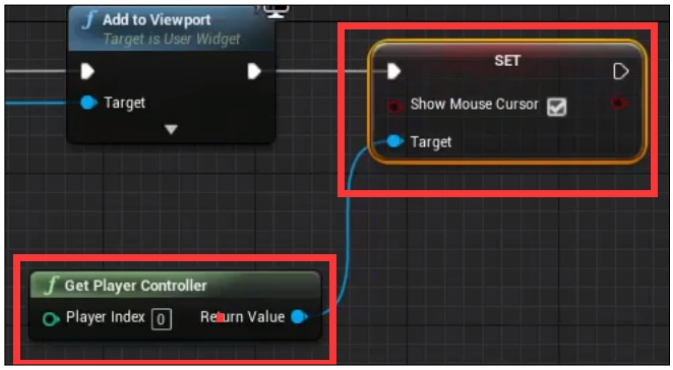
1. The Five W's of Action RPG
2. Creating a Potion Item
3. Mixing Blueprints and C++ to Create a Pick Up
4. Dropping Loot
5. Going from Blueprint to C++ and Live++

User Interface with Blueprint and UMG

Goal: develop a system to switch between design alternatives using a mouse-driven user interface
UMG: Unreal Motion Graphics
Level Streaming: method for loading and unloading levels



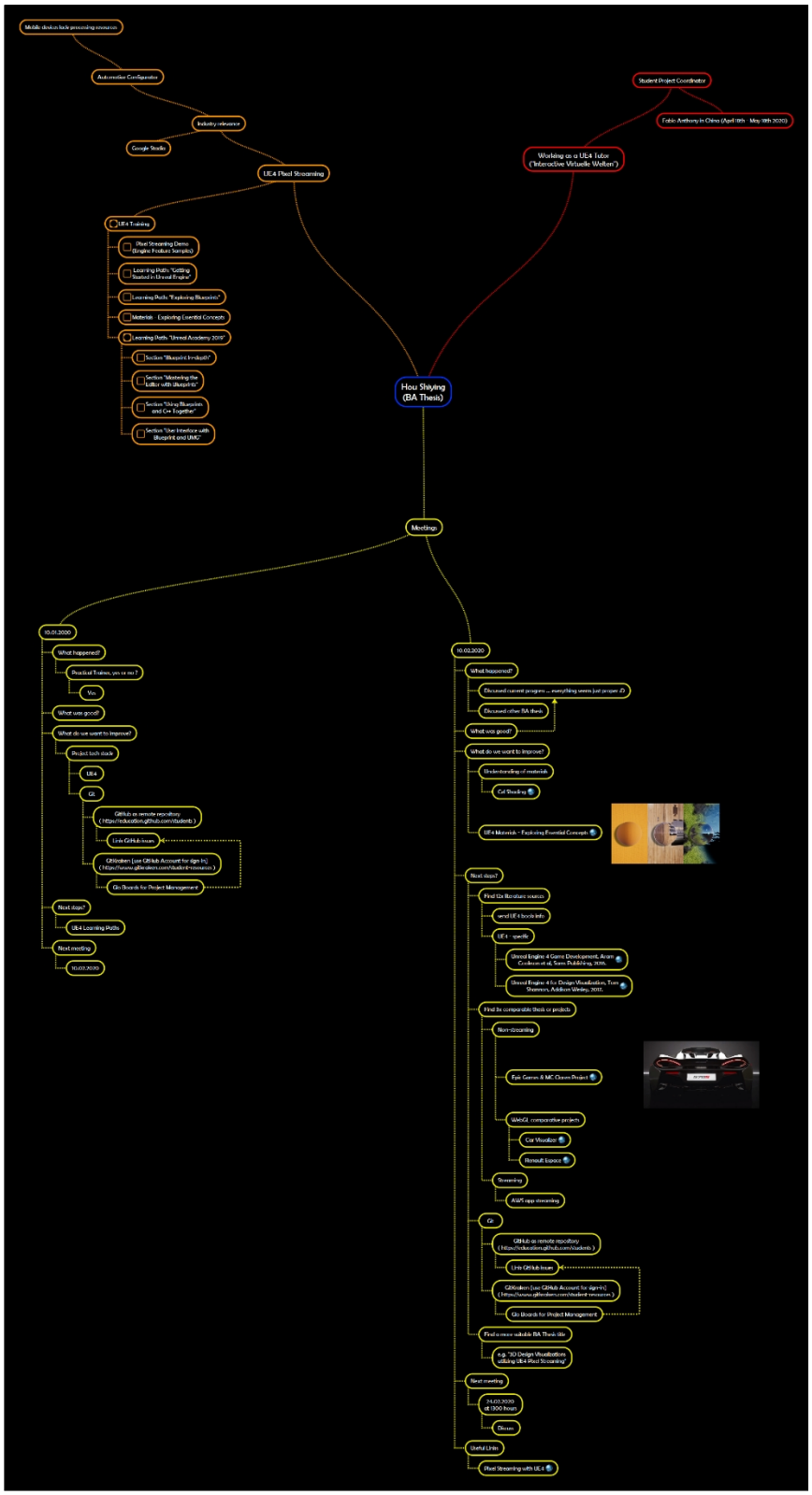
Add cursor:


Add to Viewport
Target is User Widget
Target
SET
Show Mouse Cursor
Target
Get Player Controller
Player Index 0
Return Value

10.02.2020

2nd Meeting

Reflections: Talk about the following things. Many things to do, fighting!



11.02.2020

GitHub and GitKraken

Reflections: SSH Error.

GitHub

Terms: Repository, Star, Fork, Pull Request, Watch, Issue

12.02.2020 - 14.02.2020

C++ Basic Reviewing

Reflections: Reviewed the basic concepts so that I can understand the code.

C++ Primer Chapter 1-7

12.02.2020 - 14.02.2020

Getting Started with Pixel Streaming

Reflections: Finish the following two official tutorial.

https://docs.unrealengine.com/en-US/Platforms/PixelStreaming/index.html
https://docs.unrealengine.com/en-US/Resources/Showcases/PixelStreamingShowcase/index.html

Pixel Streaming

Run your Unreal Engine application on a server in the cloud, and **stream its rendered frames and audio** to browsers and mobile devices over WebRTC.
With Pixel Streaming, you run a packaged Unreal Engine application on a desktop PC or a server in the cloud, along with a small stack of web services that are included with the Unreal Engine. People connect using any modern Web browser on their platform of choice, whether desktop or mobile, and stream the rendered frames and audio from the Unreal Engine application. There's no need for users to install or download anything. It's just like streaming a video from YouTube or Netflix — except that users can also **interact with the application** using keyboard, mouse, touch input, and even **custom HTML5 UI** that you create in your player Web page.

WebRTC

Web Real-Time Communication: allow direct connection between browsers
Signaling
How: API; identify; type of data; NAT traversal; security
Challenges: WebRTC works via UDP; no standard signaling protocol; not fully compatible with all browsers
Why: removes the need for extra apps; embedded in web technologies; secure

02.03.2020

3rd Meeting (Mainly about Assistant Task)

Reflections: One more task! To be busy afterwards, fighting!

- GitKraken/ Gitkraken Glo Animated/ Video Tutorial (**Deadline: 04.03.2020**)
- Preparation for Pixel Streaming Lectures (incl. related course assignment)
- C++ Program Design course
    - Visual Studio vs. Visual Studio Code + Build Tools
    - C++ & SDL2 vs. C++ only vs. C++ & UE4
    - project-based vs. language-based
    - possible projects:
        - Digital Card Game
        - Tic-Tac-Toe
        - ShiTou-JianDao-Bu
        - Chat room app
        - Twin-Stick Shooter
            - UE4 Tanks vs. Zombies Tutorial (https://forums.unrealengine.com/unreal-engine/events/72059-training-stream-tanks-vs-zombies-c-live-from-epic-hq?100199=)
- Miscellaneous:
    - https://support.gitkraken.com/git-workflows-and-extensions/intro-and-requirements/

03.03.2020 - 04.03.2020

Finish Git Tutorial Video

Reflections: This is the first time I recorded a teaching video.

Video Recorder: ApowerREC (https://www.apowersoft.com/record-all-screen)
Video Editor: ApowerEditor (https://www.apowersoft.com/video-editor)
GitHub Repo: https://github.com/hsybb0918/GitTutorialVideo

05.03.2020 - 06.03.2020

Add LFS part to Git Tutorial Video

Reflections: LFS is necessary when using big files.

Git LFS (https://git-lfs.github.com)
(1)https://support.gitkraken.com/git-workflows-and-extensions/intro-and-requirements
(2)https://www.youtube.com/watch?v=Hv_v3tPuNj4

Git Large File Storage
My Video: https://github.com/hsybb0918/GitTutorialVideo