Voice from Communities

- Many questions are asked in Wechat groupchat, Github discussion. We replied over 200 questions during the past week, please keep asking questions!
- We are working on a document which contains questions been asked before, please keep asking us questions no matter how fundamental the questions are, this will help everyone in the community

Q&A about Piccolo Engine

- Q1: Why does Piccolo Engine use Vulkan as graphics API? It is quite difficult, any recommendations on references?
    - Next-gen graphics API
    - Cross-platform compatibility, supporting Windows, Linux, macOS, Android, and iOS
    - Recommended references:
        - https://vulkan-tutorial.com/
        - https://www.vulkan.org/learn
- Q2: Is there an open-source plan for MetaParser?
    - In the up-coming update!
- Q3: How to configure development environment for newbies?
    - Instructions in README.md
    - Detailed instructions in Document of Programming Assignment 1
      https://cdn.boomingtech.com/games104_static/upload/GAMES104_PA01.pdf
    - Feel free to ask questions!

Piccolo Engine Features

- **Editor**
  - load / save level
  - add/delete/move/rotate/scale objects
  - Play In Editor (PIE)
- **Renderer**
  - forward shading
  - shadow
  - RHI
- **Animation**
  - simple skeleton animation
  - blending
- **Collision**
  - integrated Jolt physics
- **Character/Camera**
  - first / third-person camera
- **Motor**
  - eight-direction moving + sprinting
- **Single-threaded object-based ticking**
- **Resource manager**
- **Windows, Linux, and macOS compatible**
- **More features are on the way…**

Lecture 13

Tool Chains

Modern Game Engine - Theory and Practice

Outline of Tool Chains

01.

Foundation of Tool Chains

- What is Game Engine Tool Chains
- Complicated Tool GUI
- How to Load Asset - Deserialization
- How to Make a Robust Tools
- How to Make Tool Chain
- What You See is What You Get
- One More Thing - Plugin

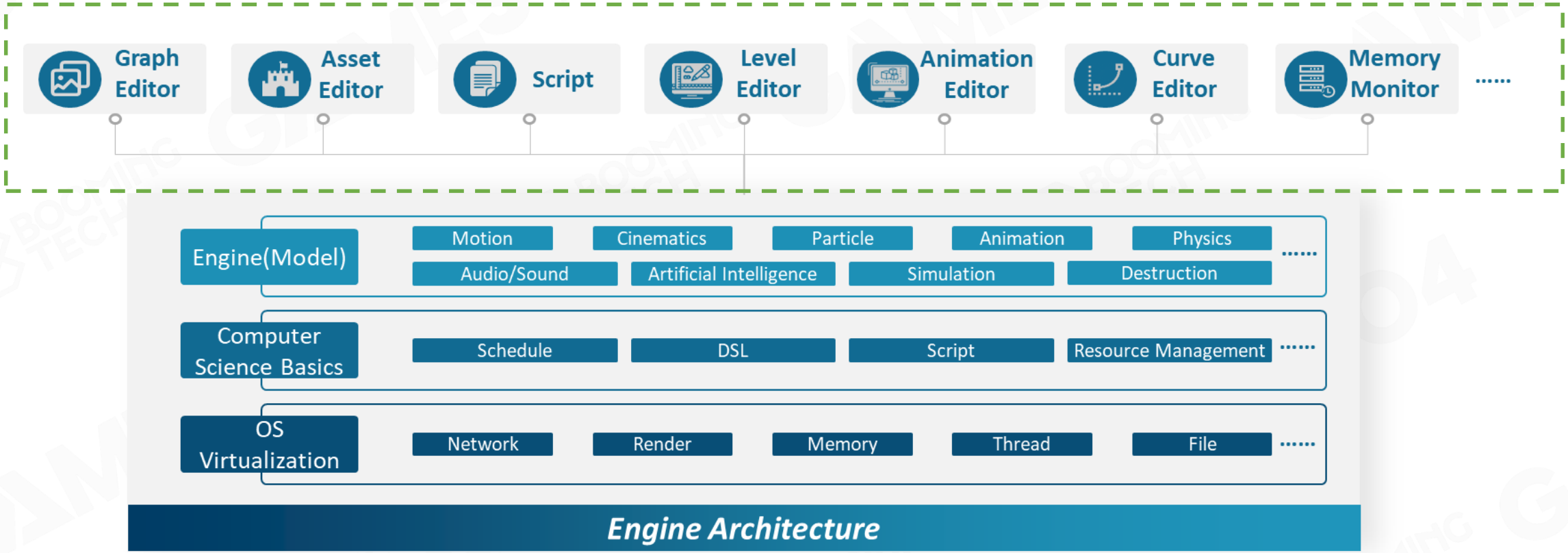
02.

Applications & Advanced Topic

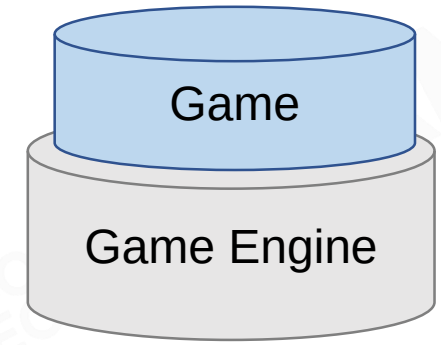
- Common Game Production Workflow
- Common Editors
- Reflection
- Collaborative Editing

What is Game Engine Tool Chain

Layer Between Users and Engine Runtime

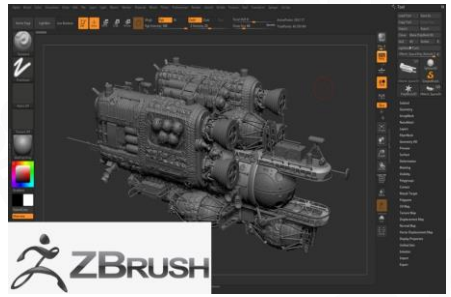


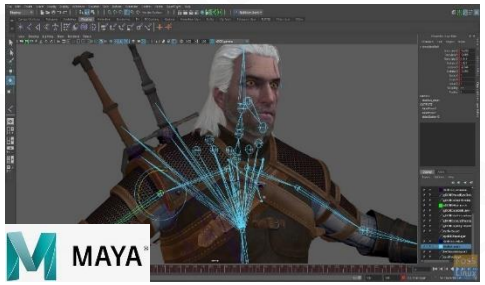
Bridge Between DCC Tools and Game Engine



Asset Conditioning Pipeline

DCC





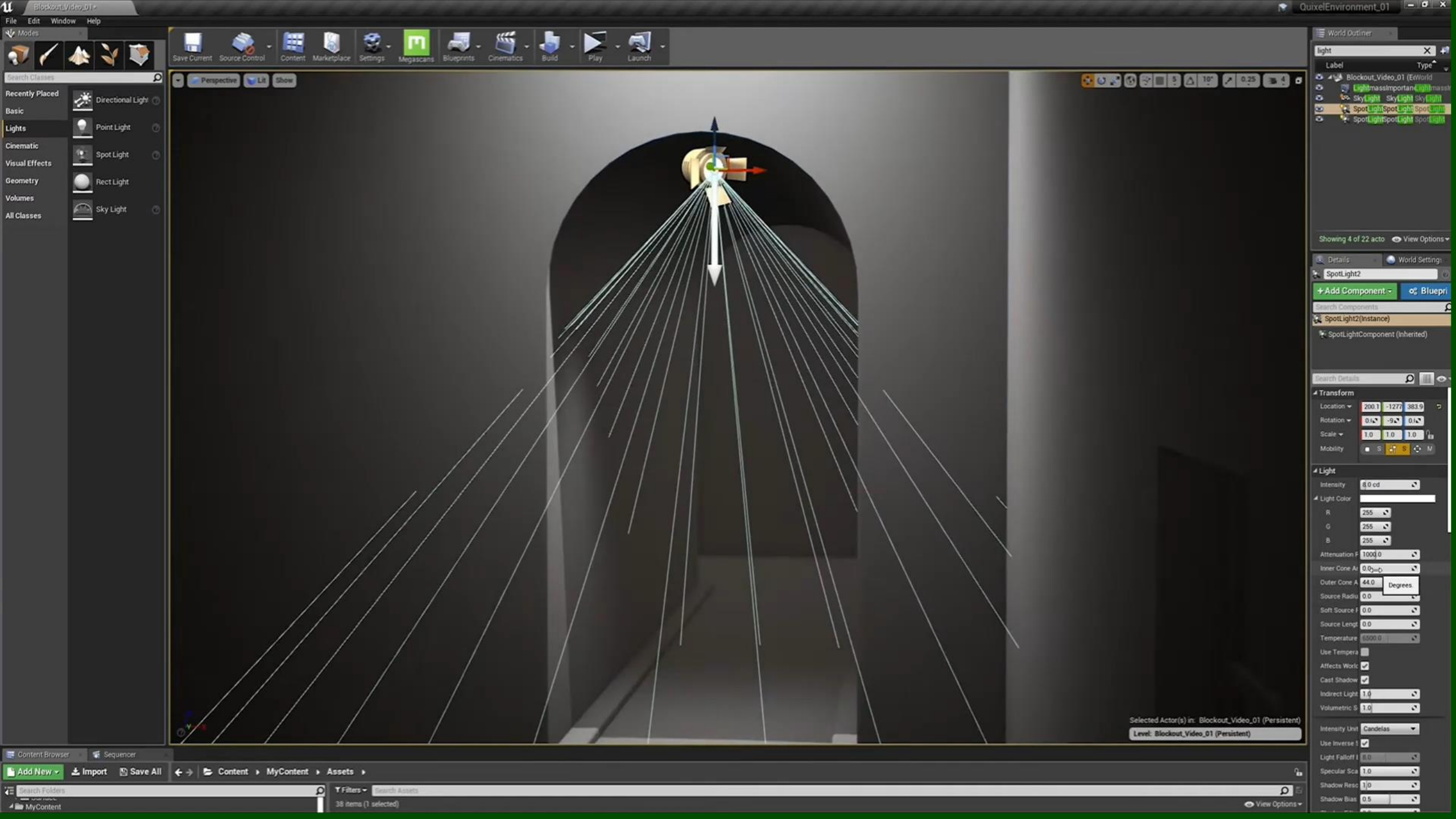
Let Huge Different Mindset Users Work Together

For Designers

- Iterate the gameplay quickly
- Implement game logic prototype quickly even without programming
- Edit massive data easily

For Artists

- The quality of the result
- Convenient workflow
- What you see is what you get (WYSIWYG)

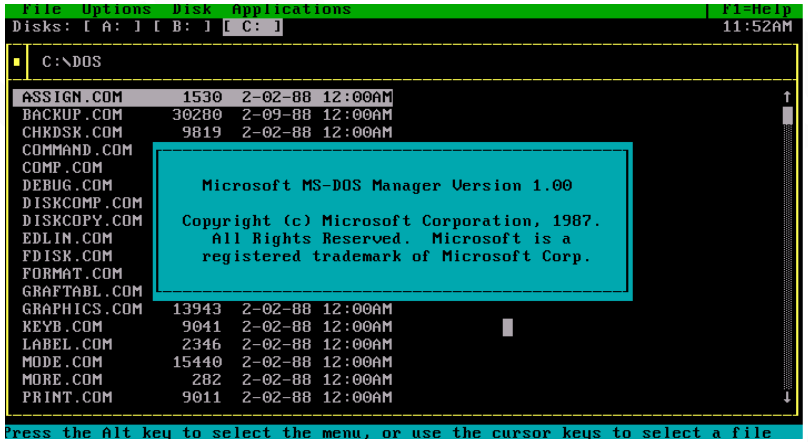

Blockout_Video_01
QuixelEnvironment_01
File
Edit
Window
Help
Modes
Save Current
Source Control
Content
Marketplace
Settings
Megascans
Blueprints
Cinematics
Build
Play
Launch
Search Classes
Recently Placed
Basic
Lights
Cinematic
Visual Effects
Geometry
Volumes
All Classes
Directional Light
Point Light
Spot Light
Rect Light
Sky Light
Perspective
Lit
Show
World Outliner
light
Label
Type
Blockout_Video_01 (Editor)
LightmassImportanceVolume
SkyLight
SpotLight
Showing 4 of 22 actors
View Options
Details
World Settings
SpotLight2
Add Component
Search Components
SpotLight2(Instance)
SpotLightComponent (Inherited)
Search Details
Transform
Location
Rotation
Scale
Mobility
Light
Intensity
Light Color
Attenuation Radius
Inner Cone Angle
Outer Cone Angle
Degrees
Source Radius
Soft Source Radius
Source Length
Temperature
Use Temperature
Affects World
Cast Shadows
Indirect Lighting Intensity
Volumetric Scattering Intensity
Intensity Units
Candelas
Use Inverse Square Falloff
Light Falloff Exponent
Specular Scale
Shadow Resolution Scale
Shadow Bias
Selected Actor(s) in: Blockout_Video_01 (Persistent)
Level: Blockout_Video_01 (Persistent)
Content Browser
Sequencer
Add New
Import
Save All
Content
MyContent
Assets
Search Folders
Filters
Search Assets
38 items (1 selected)
View Options

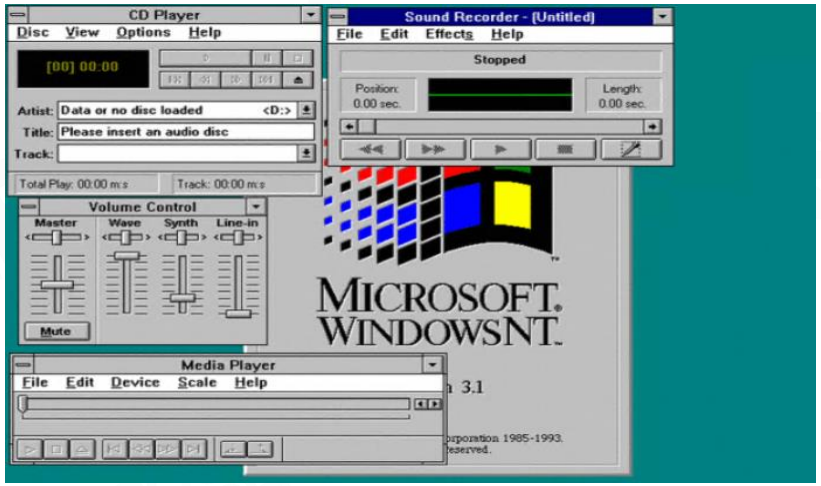
Complicated Tool GUI

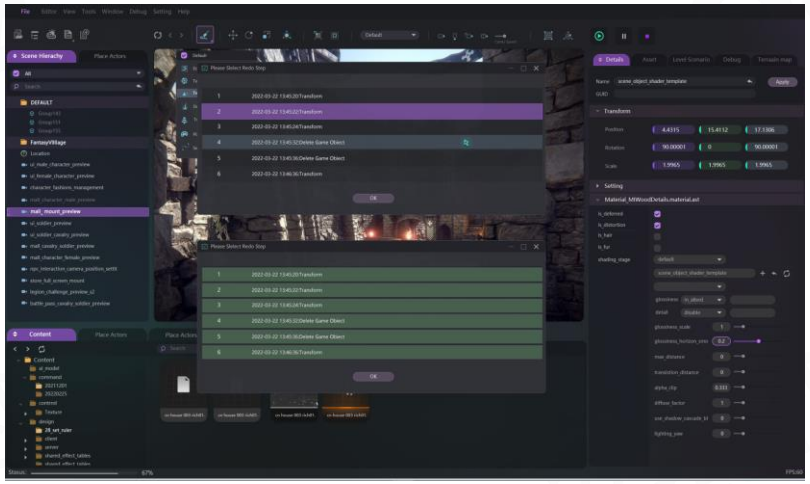
Graphics User Interface (GUI)

GUI is getting more and more complex

- Fast iteration
- Separation of design and implementation
- Reusability
- ...



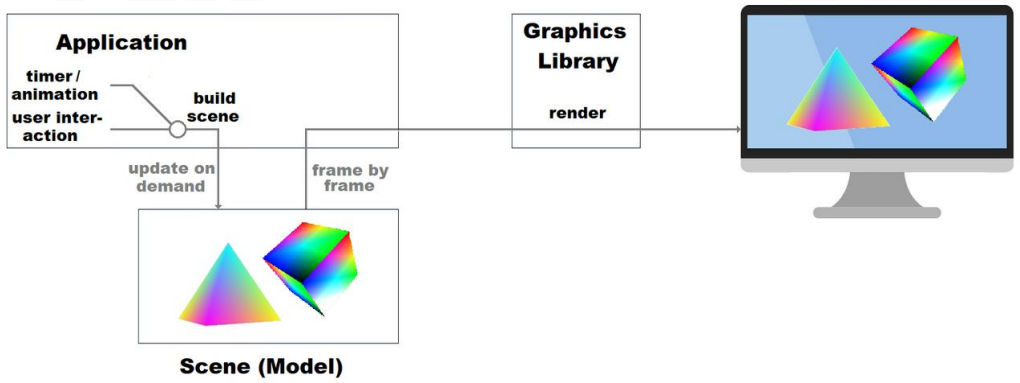




Immediate Mode (1/2)

Immediate Mode

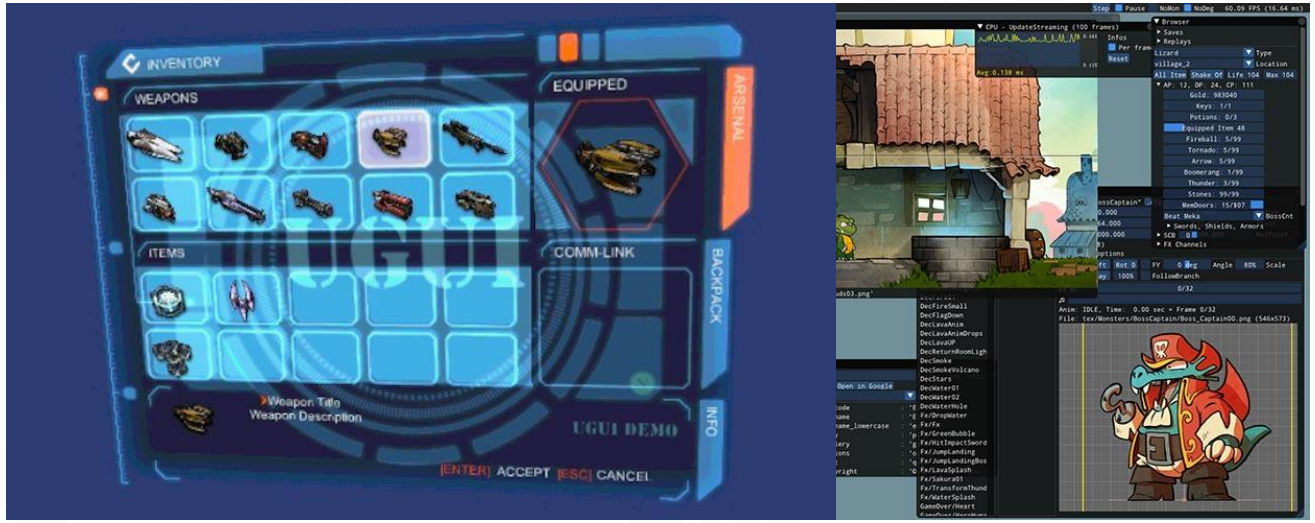
- The client calls cause rendering of graphics objects to the display.
- the data to describe rendering primitives is inserted frame by frame directly from the client into a command list.



Immediate Mode (2/2)

Characteristic
- Lightweight
- Procedural programming
- Widgets don’t maintain any data or state

```
Imgui::DrawButton("hello", 12, 24, &callback_func);
```



Pros
- Straightforward
- Simple
- Quick prototype

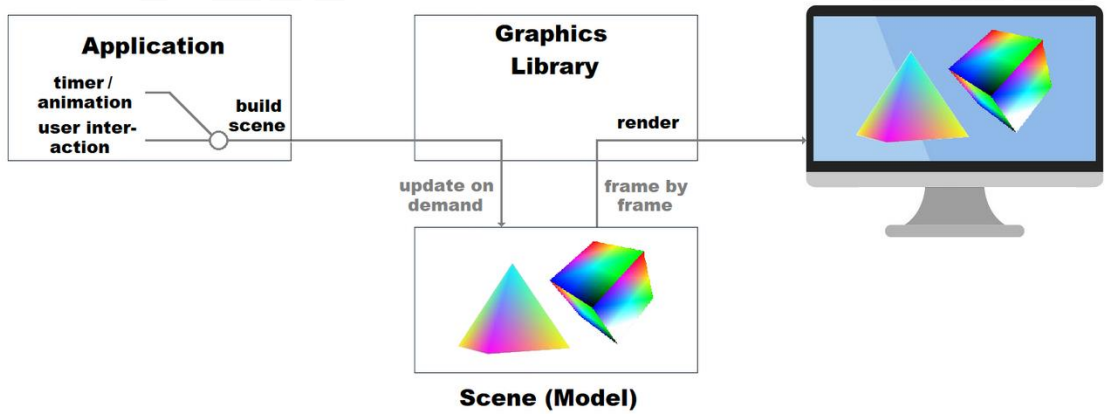
Cons
- Poor scalability
- Poor performance
- Poor maintainability

Examples
- Unity UGUI
- Omniverse GUI
- Piccolo GUI

Retained Mode (1/2)

Retained Mode

- The graphics library, instead of the client, retains the scene to be rendered.
- The client calls into the graphics library do not directly cause actual rendering, but make use of extensive indirection to resources, managed by the graphics library.



Retained Mode (2/2)

Characteristic

- Object-oriented
- Widgets contain their own state and data
  - Draw widgets as needed
  - Complicated effects (animation et.al.)

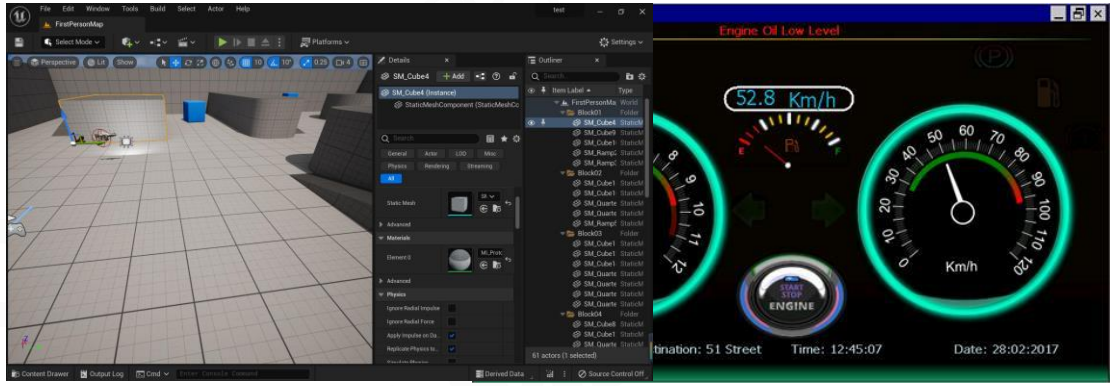
Pros

- High scalability
- High performance
- High maintainability

Cons

- Complex for developers
  - Message queue / callbacks
  - Synchronization between GUI and application

```
HorizontalLayout layout = new HorizontalLayout();
Button button = new Button();
button.SetText("Hello!");
button.SetWidth(12);
button.SetHeight(24);
button.SetCallback(&callback_func);
layout.Add(button);
```



Examples

- Unreal UMG
- WPF GUI
- QT GUI

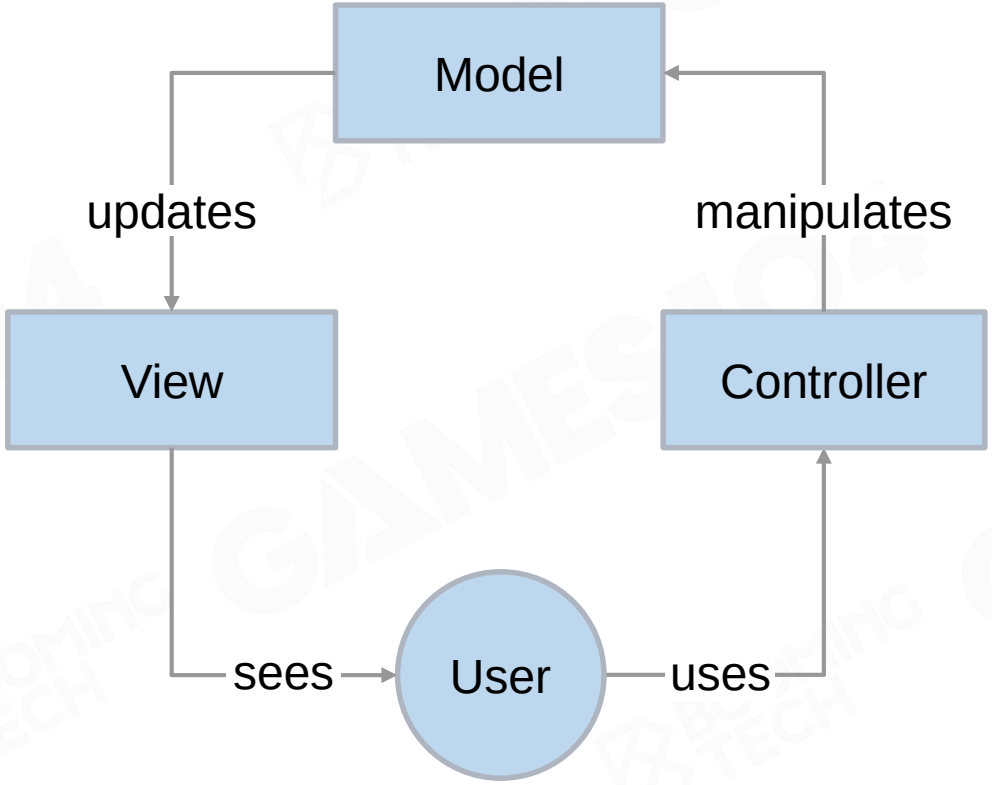
Design Pattern – MVC

Invented by **Trygve Reenskaug** in 1978, to bridge the gap between the human user's mental model and the digital model that exists in the computer.

Model: The central component of the pattern, responsible for managing the data of the application.

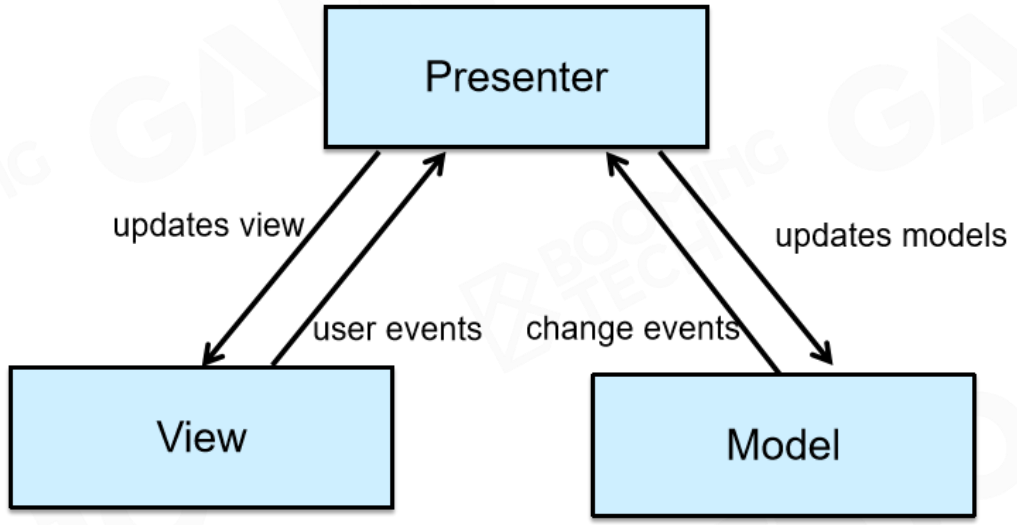
View: Any representation of information such as a chart, diagram or table.

Controller: Accepts input and converts it to commands for the model or view.



Design Pattern – MVP

The evolution of the MVC design pattern,wherein the controller is replaced by the presenter.



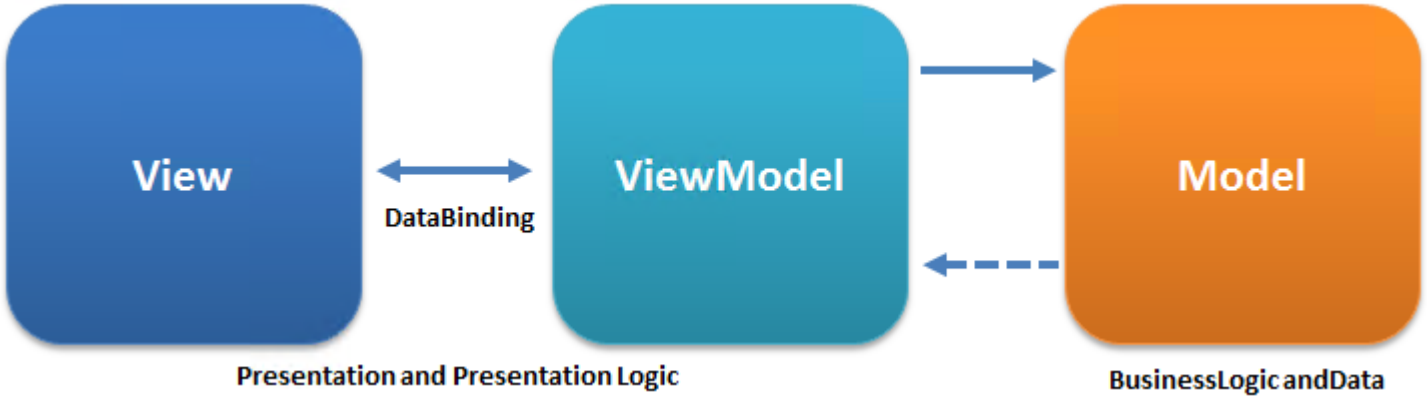
Model: An interface defining the data to be displayed or otherwise acted upon in the user interface.

View: A passive interface that displays data (the model) and routes user commands (events) to the presenter to act upon that data.

Presenter: Acts upon the model and the view. It retrieves data from repositories (the model), and formats it for display in the view.

Design Pattern – MVVM (1/3)

A variation of Model/View/Controller (MVC)



In MVVM,View is the responsibility of a designer rather than a classic developer.

The designer is generally a more graphical,artistic focused person, and does less classic coding than a traditional developer.

Design Pattern – MVVM (2/3)

View

```
<WrapPanel Name="wrapnel"
          DataContext="{Binding CurrentStudent}">
        <Label Content="name:"/>
        <TextBox Width="100" Text="{Binding m_name}"/>
        <Label Content="score:"/>
        <TextBox Width="100" Text="{Binding m_score}"/>
        <CheckBox   IsChecked="{Binding m_pass}"
              Content="pass"></CheckBox>
</WrapPanel>
```

Binding

ViewModel

```
class StudentViewModel:INotifyPropertyChanged
{
    private Student _CurrentStudent;
    public Student CurrentStudent { get {
            return _CurrentStudent;
        } set {
            _CurrentStudent = value;
            NotifyPropertyChange("CurrentStudent");
        }
    }
    public void GetCurrentStudentById(int id)
    {
        CurrentStudent = studentModel.Students[id];
    }
    StudentModel studentModel ...
}
```

Model

```
class Student
{
    public int m_id { get; set; }
    public string m_name { get; set; }
    public double m_score { get; set; }
    public bool m_pass { get; set; }
}
class StudentModel
{
    public List<Student> Students { get; set; }
    public StudentModel()
    {
        Students = QueryStudent();
    }
    public List<Student> QueryStudent()...
}
```

View: using a WYSIWYG tool such as Dreamweaver,VS Blend and save as html/xaml , view state that MVC encodes in its View classes is not easy to represent.

Binding: bind View Data to the Model ,no more code in View classes.

ViewModel - Model of View: The Model is very likely to have a data types that cannot be mapped directly to controls,ViewModel contains data-transformers that convert Model types into View types.

Design Pattern – MVVM (3/3)

Pros

- Independent development
- Easy to maintain and test
- Easy to reuse components

Cons

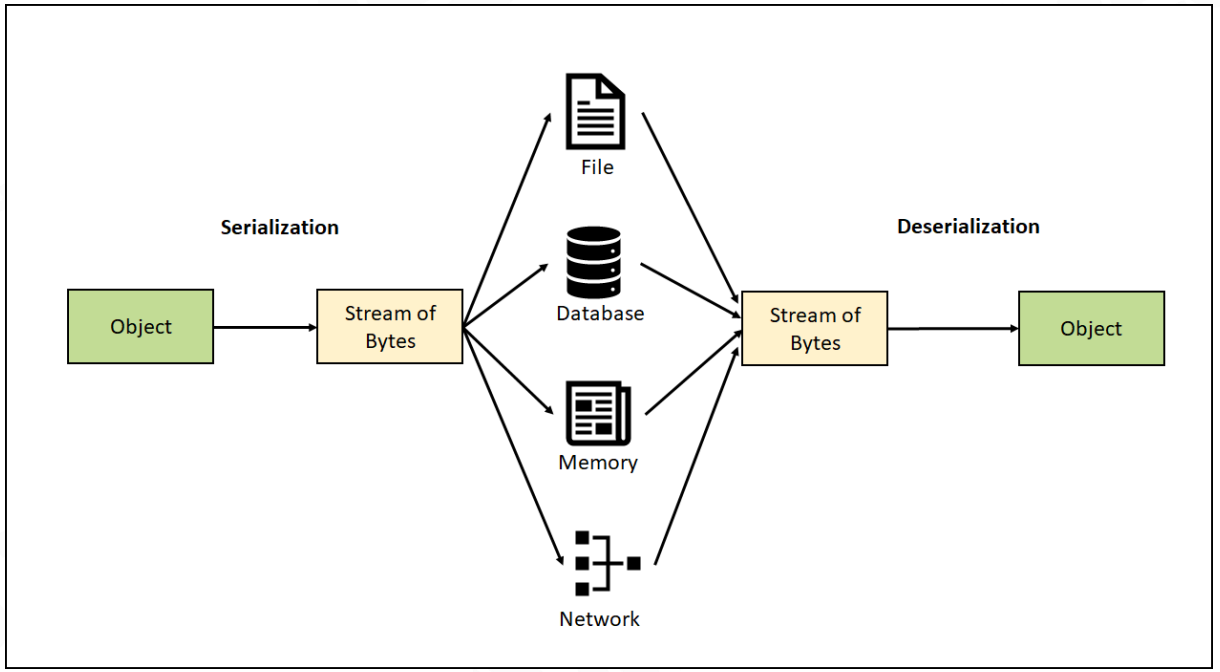
- For simple UI, MVVM can be overkill
- Data-binding is declarative and harder to debug


Unreal Editor - StackOBot
Unreal Editor 5.0.0
Loading...
Copyright © Epic Games, Inc.

Load & Save

Serialization and Deserialization

Serialization is the process of translating a data structure or object state into a format that can be stored (for example, in a file or memory data buffer) or transmitted (for example, over a computer network) and reconstructed later.



Deserialization is the opposite operation, extracting a data structure from a series of bytes.

Text Files

- Save data as text files
- Example: TXT, Json, YAML, XML …
- Can read by common text editors

Engine applications:

- Unity Editor(optional): subset of YAML
- Piccolo: Json
- Cryengine: XML / Json (optional)

TXT

```
# OBJ file format with ext .obj
# vertex count = 2503
# face count = 4968
v -3.4101800e-003 1.3031957e-001 2.1754370e-002
v -8.1719160e-002 1.5250145e-001 2.9656090e-002
v -3.0543480e-002 1.2477885e-001 1.0983400e-003
v -2.4901590e-002 1.1211138e-001 3.7560240e-002
v -1.8405680e-002 1.7843055e-001 -2.4219580e-002
v 1.9067940e-002 1.2144925e-001 3.1968440e-002
v 6.0412000e-003 1.2494359e-001 3.2652890e-002
v -1.3469030e-002 1.6299355e-001 -1.2000020e-002
v -3.4393240e-002 1.7236688e-001 -9.8213000e-004
v -8.4314160e-002 1.0957263e-001 3.7097300e-003
v -4.2233540e-002 1.7211574e-001 -4.1799800e-003
v -6.3308390e-002 1.5660615e-001 -1.3838790e-002
v -7.6903950e-002 1.6708033e-001 -2.6931360e-002
v -7.2253920e-002 1.1539550e-001 5.1670300e-002
v 1.2981330e-002 1.1366375e-001 3.8302950e-002
v -3.7857280e-002 1.7010102e-001 1.4236000e-003
v 4.8689400e-003 3.7962370e-002 4.5867630e-002
v -5.7180550e-002 4.0918830e-002 4.6301340e-002
v -4.5209070e-002 3.8839100e-002 4.4503770e-002
v -3.3761490e-002 1.2617876e-001 1.7132300e-003
v -5.0242270e-002 1.5773747e-001 9.3944500e-003
v -2.1216950e-002 1.5887938e-001 -4.6923700e-003
v -5.6472950e-002 1.5778406e-001 8.1786500e-003
v -5.2802060e-002 4.1319860e-002 4.6169800e-002
v -4.9960340e-002 4.3101950e-002 4.4462650e-002
v -2.9748750e-002 3.6539860e-002 5.2493310e-002
v -3.5438900e-003 4.2659770e-002 4.7541530e-002
```

Json

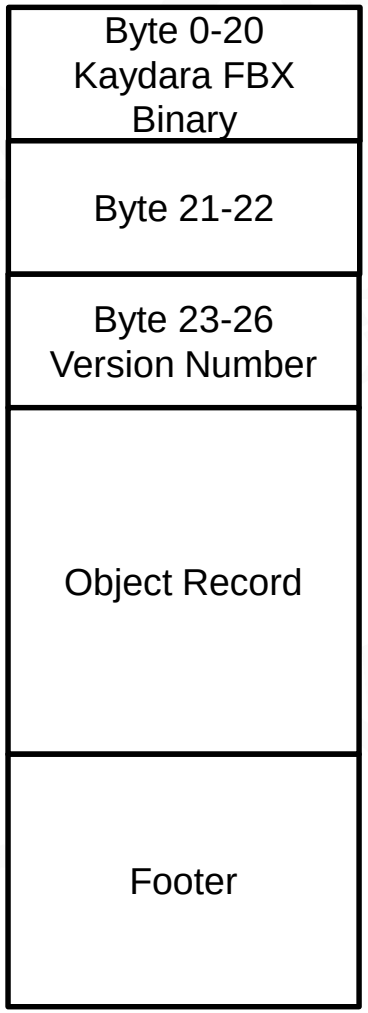
```
"shapes":
{
  "typeName": "RigidBodyShape",
  "local_transform": {
    "typeName": "Transform",
    "position": {
      "x": 0,
      "y": 0,
      "z": -0.1
    },
    "rotate": {},
    "scale": {
      "x": 1,
      "y": 1,
      "z": 1
    }
  },
  "geometry": {
    "$context": {
      "half_extents": {
        "x": 10,
        "y": 10,
        "z": 0.1
      }
    },
    "$typeName": "Box"
  }
}
```

Binary Files

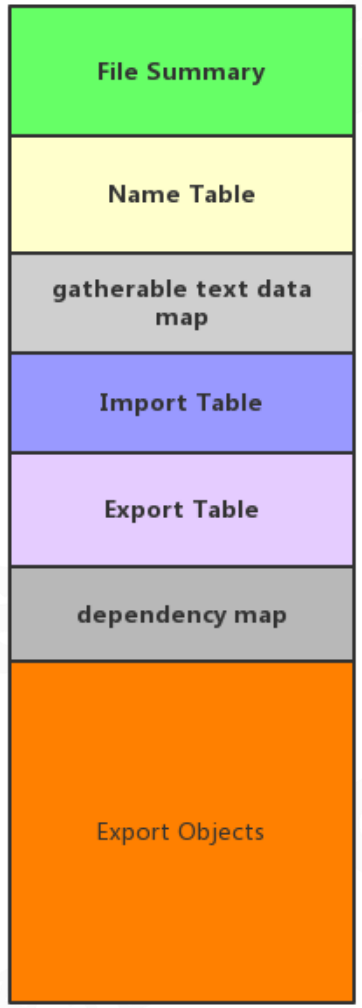
- Save data as bytes stream
- Need additional tools for read/write
- Example: UAsset, FBX Binary ...

Engine applications:

- Unity Runtime, Unity Editor (optional)
- CryEngine (optional)
- Unreal: UAsset



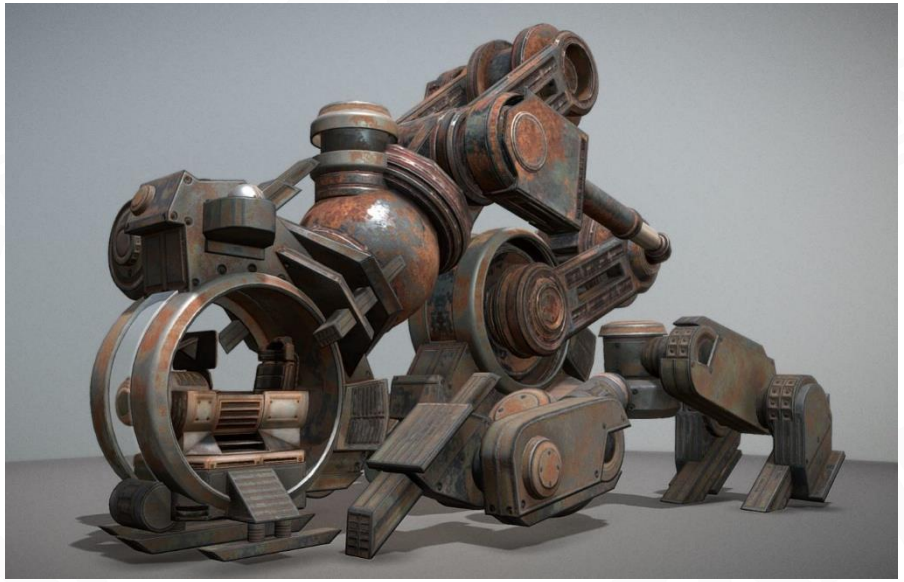
FBX Binary



UAsset

Storage Comparison -Text vs. Binary

Text vs. Binary



Name	Type	Size
Neck_Mech_Walker (FBX binary).fbx	3D Object	10,126 KB
Neck_Mech_Walker (FBX text).fbx	3D Object	56,384 KB

FBX Binary

Neck_Mech_Walker (FBX binary).fbx

```
4b61 7964 6172 6120 4642 5820 4269 6e61
7279 2020 001a 00e8 1c00 005f 0700 0000
0000 0000 0000 0012 4642 5848 6561 6465
7245 7874 656e 7369 6f6e 5c00 0000 0100
0000 0500 0000 1046 4258 4865 6164 6572
5665 7273 696f 6e49 eb03 0000 7800 0000
0100 0000 0500 0000 0a46 4258 5665 7273
696f 6e49 e81c 0000 9800 0000 0100 0000
0500 0000 0e45 6e63 7279 7074 696f 6e54
7970 6549 0000 0000 8101 0000 0000 0000
0000 0000 1143 7265 6174 696f 6e54 696d
6553 7461 6d70 cf00 0000 0100 0000 0500
0000 0756 6572 7369 6f6e 49e8 0300 00e5
0000 0001 0000 0005 0000 0004 5965 6172
49e2 0700 00fc 0000 0001 0000 0005 0000
0005 4d6f 6e74 6849 0b00 0000 1101 0000
0100 0000 0500 0000 0344 6179 4906 0000
0027 0100 0001 0000 0005 0000 0004 486f
7572 4913 0000 003f 0100 0001 0000 0005
0000 0006 4d69 6e75 7465 4910 0000 0057
0100 0001 0000 0005 0000 0006 5365 636f
6e64 492f 0000 0074 0100 0001 0000 0005
0000 000b 4d69 6c6c 6973 6563 6f6e 6449
0403 0000 0000 0000 0000 0000 0000 0000
00c9 0100 0001 0000 0034 0000 0007 4372
6561 746f 7253 2f00 0000 426c 656e 6465
```

FBX Text

```
; FBX 6.1.0 project file
; Created by Blender FBX Exporter
; for support mail: ideasman42@gmail.com
; ------------------------------------------------------------

FBXHeaderExtension:  {
    FBXHeaderVersion: 1003
    FBXVersion: 6100
    CreationTimeStamp:  {
        Version: 1000
        Year: 2018
        Month: 11
        Day: 06
        Hour: 19
        Minute: 16
        Second: 50
        Millisecond: 0
    }
    Creator: "FBX SDK/FBX Plugins build 20070228"
    OtherFlags:  {
        FlagPLE: 0
    }
}
CreationTime: "2018-11-06 19:16:50:000"
Creator: "Blender version 2.79 (sub 0)"

```

Assert Data Repeatance

Meshes which in the red boxes are redundant data.

How do game developers solve the problem?

Asset Reference

Assets Reference is a way to separate redundant data into asset files and complete association by establishing reference relationships.

```
<Layout >
    <element type="String" >
        <name>courtyard_1</name>
        <value>/yard/courtyard.object.ast</value>
        <position>0.000000 0.000000 0.000000</position>
        <rotation>0.000000 0.000000 0.000000</rotation>
        <scale>1.000000 1.000000 1.000000</scale>
    </element>
    <element type="String" >
        <name>courtyard_2</name>
        <value>/yard/courtyard.object.ast</value>
        <position>0.500000 0.820000 0.410000</position>
        <rotation>0.000000 0.000000 0.000000</rotation>
        <scale>2.000000 2.000000 2.000000</scale>
    </element>
    <element type="String" >
        <name>courtyard_3</name>
        <value>/yard/courtyard.object.ast</value>
        <position>0.500000 0.820000 0.410000</position>
        <rotation>0.000000 0.000000 0.000000</rotation>
        <scale>2.000000 2.000000 2.000000</scale>
    </element>
</Layout>
```

Layout.ast

```
<ObjectRSA>
    <element type="Mesh" >
        <index_buffers>
            <primitive_count>2</primitive_count>
            <indices>0 1 2 2 3 0</indices>
        </index_buffers>
        <vertex_buffers>
            <vertex_count>4</vertex_count>
            <positions>
                <channel_count>3</channel_count>
                <decl_type>Float3</decl_type>
                <buffer>9.600000 0.000000 11.000000 -9.600000 0.000000 11.000000
                -9.600000 0.000001 0.000205 9.600000 0.000001 0.000205</buffer>
            </positions>
        <vertex_buffers>
        <material>
            <url>objects/ourtyard.material.ast</url>
        </material>
    </element>
</ObjectRSA>
```

courtyard.object.ast

Object Instance in Scene

Data instance is a way to create a parent data that you can use as a base to make a wide variety of different children and can also be used directly.

```
<ObjectRSA>
    <material>
        <url>objects/environment/stones.material.ast</url>
    </material>
</ObjectRSA>
```

object.ast

Instance

```
<MaterialRSA >
    <element type="Float" >
        <name>Glossiness_scale</name>
        <value>0.040000 </value>
        <min_value>0.000000 </min_value>
        <max_value>1.000000 </max_value>
    </element>
    <element type="Float">
        <name>Alpha_clip</name>
        <value>0.333</value>
        <min_value>0.000000</min_value>
        <max_value>1.000000</max_value>
    </element>
    <element type="Vector4" >
        <name>Color</name>
        <value>0.800000 0.000000 0.000000 0.000000</value>
        <min_value>0.000000 0.000000 0.000000 0.000000</min_value>
        <max_value>1.000000 1.000000 1.000000 1.000000</max_value>
    </element>
    <element type="String">
        <name>Texture_map</name>
        <value>/Stone.meta</value>
    </element>
</MaterialRSA>
```

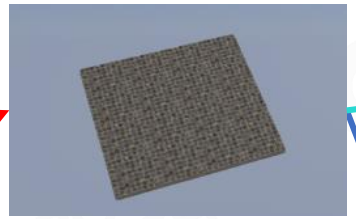
stones.material.ast

Object Instance Variance

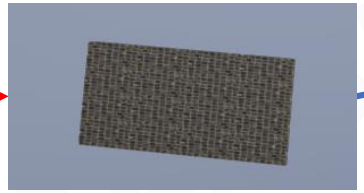
How to change the texture of Ground1

from stone to castle stone?

Layout.ast



Ground1.mesh.ast



Ground2.mesh.ast

castle stone texture

stone texture

Build Variance by Copying

Intuitive way: make a copy of instance data, modify the copy

- add lots of reduntant data

```
<MaterialRSA >
    <element type="Float" >
        <name>Glossiness_scale</name>
        <value>0.040000 </value>
        <min_value>0.000000 </min_value>
        <max_value>1.000000 </max_value>
    </element>
    <element type="Float">
        <name>Alpha_clip</name>
        <value>0.333</value>
        <min_value>0.000000</min_value>
        <max_value>1.000000</max_value>
    </element>
    <element type="Vector4" >
        <name>Color</name>
        <value>0.800000 0.000000 0.000000 0.000000</value>
        <min_value>0.000000 0.000000 0.000000 0.000000</min_value>
        <max_value>1.000000 1.000000 1.000000 1.000000</max_value>
    </element>
    <element type="String">
        <name>Texture_map</name>
        <value>/Stone.meta</value>
    </element>
</MaterialRSA>
```

Stones_Material.ast

copy

```
<MaterialRSA >
    <element type="Float" >
        <name>Glossiness_scale</name>
        <value>0.040000 </value>
        <min_value>0.000000 </min_value>
        <max_value>1.000000 </max_value>
    </element>
    <element type="Float">
        <name>Alpha_clip</name>
        <value>0.333</value>
        <min_value>0.000000</min_value>
        <max_value>1.000000</max_value>
    </element>
    <element type="Vector4" >
        <name>Color</name>
        <value>0.800000 0.000000 0.000000 0.000000</value>
        <min_value>0.000000 0.000000 0.000000 0.000000</min_value>
        <max_value>1.000000 1.000000 1.000000 1.000000</max_value>
    </element>
    <element type="String">
        <name>Texture_map</name>
        <value>/Castle_Stone.meta</value>
    </element>
</MaterialRSA>
```

Modify

Castle_Stone.material.ast

Build Variance by Data Inheritance

Data Inheritance: Inherit the data of the inherited object and allow overriding assignments to the data defined in its data structure.

```
<MaterialRSA >
    <element type="Float" >
        <name>Glossiness_scale</name>
        <value>0.040000 </value>
        <min_value>0.000000 </min_value>
        <max_value>1.000000 </max_value>
    </element>
    <element type="Float">
        <name>Alpha_clip</name>
        <value>0.333</value>
        <min_value>0.000000</min_value>
        <max_value>1.000000</max_value>
    </element>
    <element type="Vector4" >
        <name>Color</name>
        <value>0.800000 0.000000 0.000000 0.000000</value>
        <min_value>0.000000 0.000000 0.000000 0.000000</min_value>
        <max_value>1.000000 1.000000 1.000000 1.000000</max_value>
    </element>
    <element type="String">
        <name>Texture_map</name>
        <value>/Stone.meta</value>
    </element>
</MaterialRSA>
```

Stones_Material.ast

inheritance

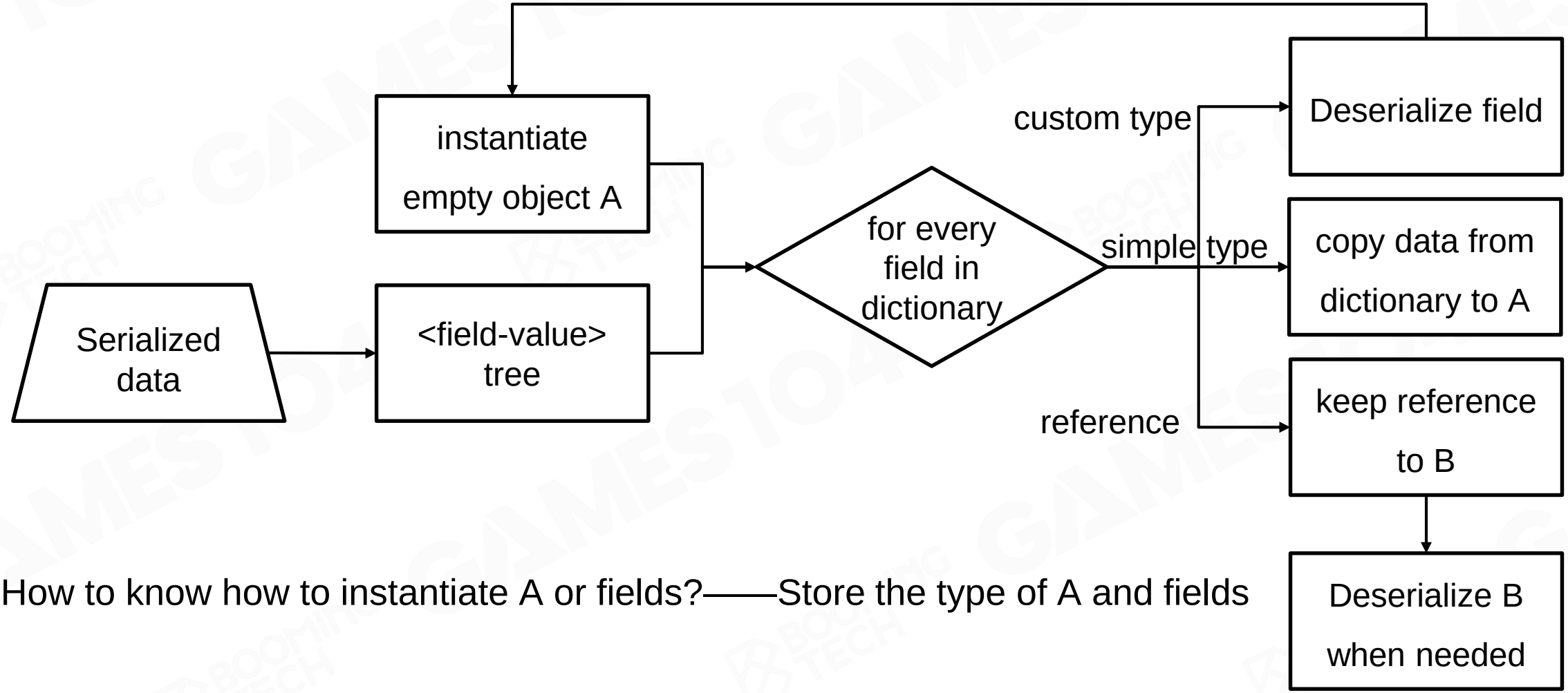
```
<MaterialRSA >
`
    <ref>Stone_Material.ast </ref>
```

override

```
    <element type="String">
        <name>Texture_map</name>
        <value>/Castle_Stone_Material.meta</value>
    </element>
</MaterialRSA>
```

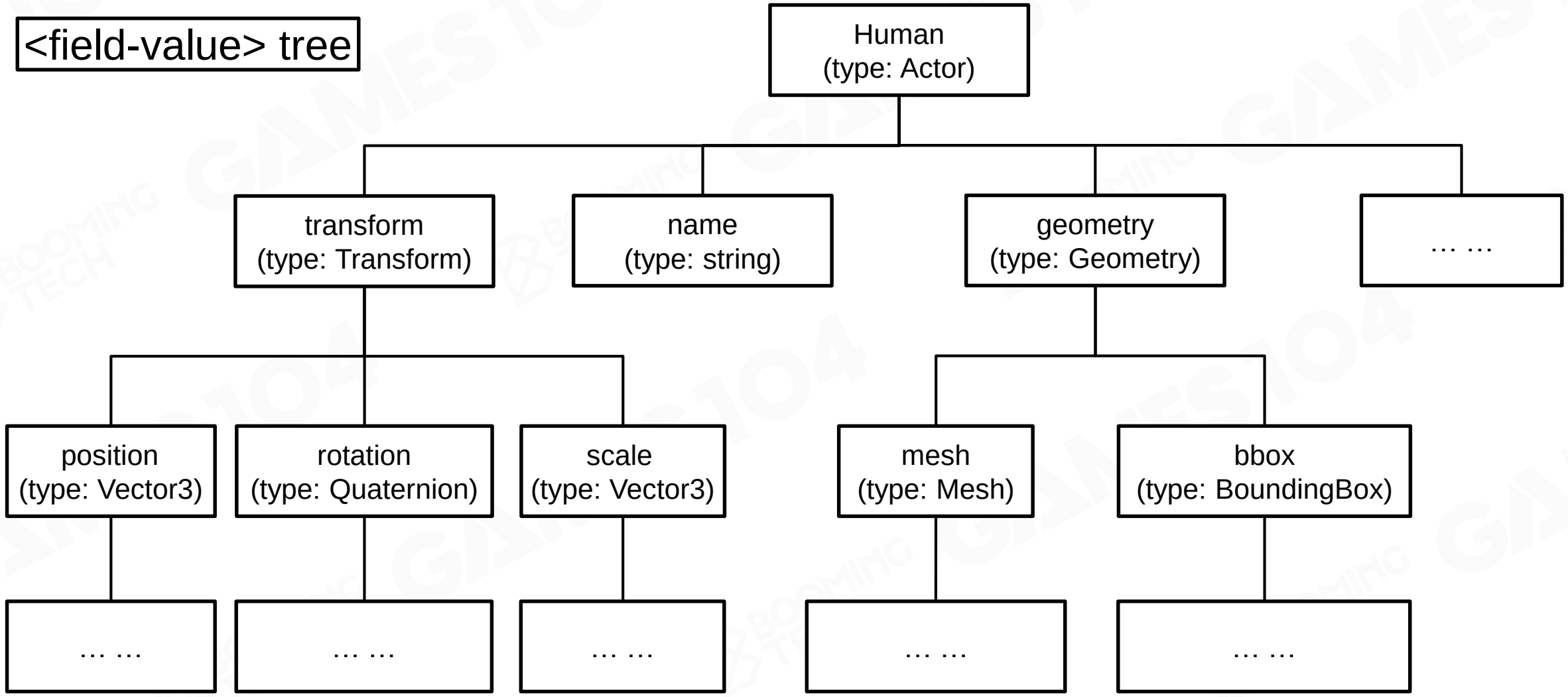
How to Load Asset - Deserialization

Parse Asset File



How to know how to instantiate A or fields?——Store the type of A and fields

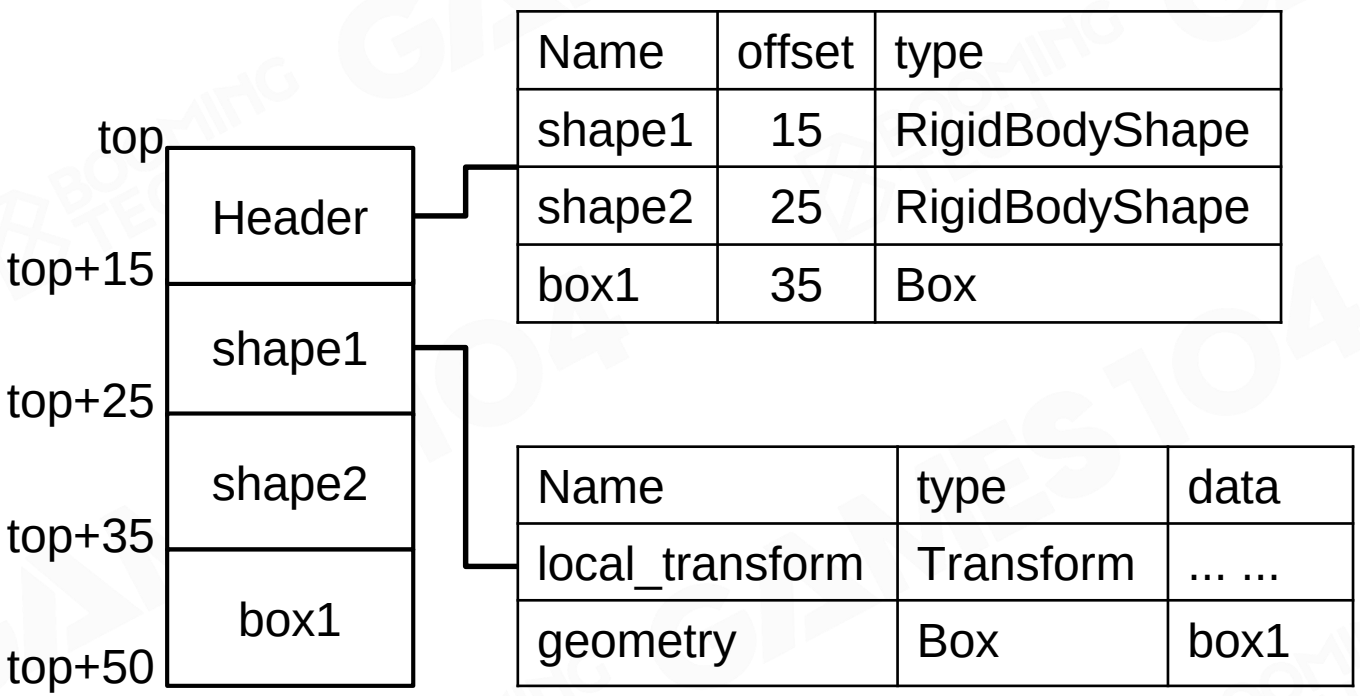
Build Key-Type-Value Pair Tree



Binary vs. Text

Where to store the objects and fields type?

- Text: store in asset
- Binary: store in a table



Name	offset	type
shape1	15	RigidBodyShape
shape2	25	RigidBodyShape
box1	35	Box

Name	type	data
local_transform	Transform	... ...
geometry	Box	box1

Binary

```
"shapes":
{
  "typeName": "RigidBodyShape",
  "local_transform": {
    "typeName": "Transform",
    "position": {
      "x": 0,
      "y": 0,
      "z": -0.1
    },
    "rotate": {},
    "scale": {
      "x": 1,
      "y": 1,
      "z": 1
    }
  },
  "geometry": {
    "$context": {
      "half_extents": {
        "x": 10,
        "y": 10,
        "z": 0.1
      }
    },
    "$typeName": "Box"
  }
}
```

Text

Endianness

Binary – Endianness

Big Endian

e.g. 0x1234567

Little Endian

Big Endian:
- begin with most significant byte
- end with least significant byte

Little Endian:
- begin with least significant byte
- end with most significant byte

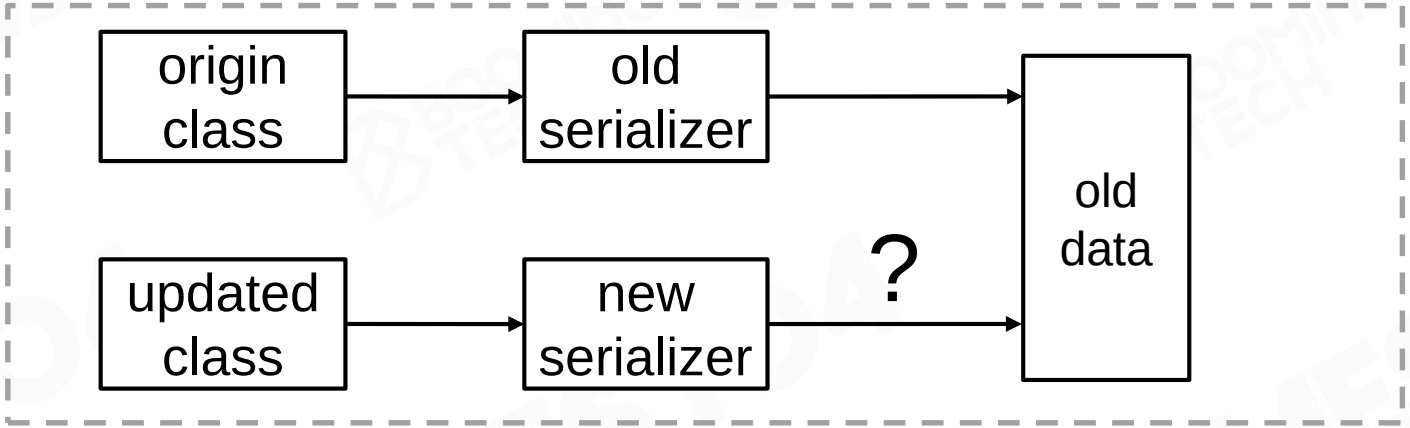
Endianness vary among different processors

Processor	Endianness
PowerPC (PPC)	Big Endian
Sun Sparc	Big Endian
IBM S/390	Big Endian
Intel x86 (32 bit)	Little Endian
Intel x86_64 (64 bit)	Little Endian
ARM	Bi (Big/Little) Endian

Unreal

```
/**
Returns true if data larger than 1 byte should
be swapped to deal with endian mismatches.
*/
FORCEINLINE bool IsByteSwapping()
{
#if PLATFORM_LITTLE_ENDIAN
    bool SwapBytes = ArForceByteSwapping;
#else
    bool SwapBytes = this->IsPersistent();
#endif
    return SwapBytes;
}
```

Asset Version Compatibility



Add or Remove Field

origin class

```
class GameObject
{
    private:
        GUID guid;
        string name;
        Transform transform;
}
```

updated class 1

```
class GameObject
{
    private:
        GUID guid;
        string name;
}
```

updated class 2

```
class GameObject{
    private:
        GUID guid;
        string name;
        Transform transform;
        BoundingBox bbox;
}
```

old data

```
{
  "guid": "092xtwg2u4ik1359",
  "name": "Alice",
  "transform": {
    "position": {
      "x": 0,
      "y": 0,
      "z": -0.1
    },
    "rotate": {},
    "scale": {
      "x": 1,
      "y": 1,
      "z": 1
    }
  }
}
```

Solve Compatibility by Version Hardcode

Unreal: add version to asset

- Load asset: check if field exists then load data
- Save asset: write all data to asset file

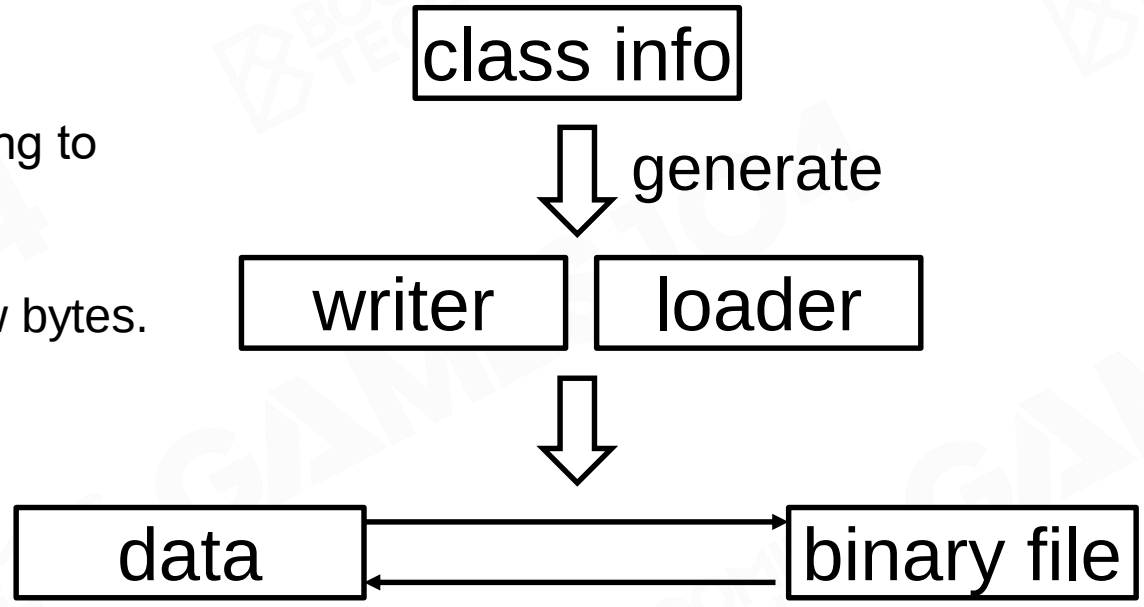
```
class GameObject:

    int x = default;
    float y = default;
    bool z = default; // new field

    function Deserialize(data):
        x = data.GetValue<int>("x");
        y = data.GetValue<float>("y");
        if(GetCurrentVersion() >= data.version){
            z = data.GetValue<bool>("z");
        }

    function Serialize(data):
        data.SetValue<int>("x", x);
        data.SetValue<float>("y", y);
        data.SetValue<bool>("z", z);
        data.UpdateVersion(GetCurrentVersion());
```

Solve Compatibility by Field UID

Google protocol buffers:
unique number for field

```
message PrefabObjectBinary{
    string guid = 1;
    string file_name = 2;
    repeated string game_object_guid_list = 3;
}
```

- Every field has a unique number, never change the number.
- Serialization:
  1. For every field, generate a “key” (fixed size) according to its field number and type.
  2. Store field data with key, key is stored in the first few bytes.
- Deserialization:
  1. Field not in schema but in data:
     key would not be recognized, skip the field.
  2. Field in schema but not in data: set default value.



How to Make a Robust Tools

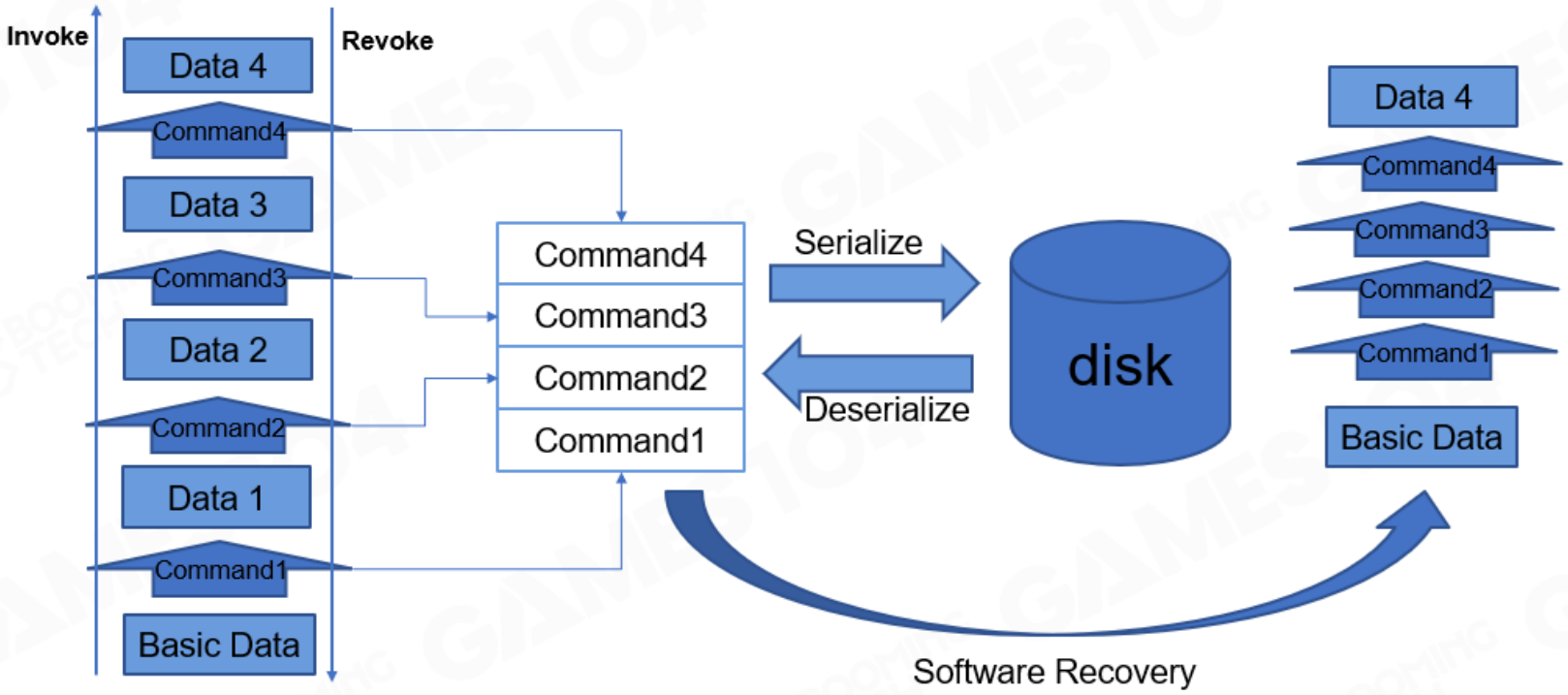

MainWorld
Foliage Mode
Platforms
Settings
Foliage
Select
All
Deselect
Invalid
Lasso
Paint
Reapply
Single
Fill
Erase
Remove
Move
Place Single Instances
Brush Options
Brush Size
20.0
Paint Density
0.5
Erase Density
0.0
Data Layer
<None>
Single Instance Mode:
All Selected
Place in Current Level
Filters
Landscape
Static Meshes
BSP
Foliage
Translucent
Foliage
Search Foliage
4.36K
4.11K
1.41K
420
27
SM_Bush_FoliageType
Mesh
SM_Bush
Perspective
Lit
Show
10
10°
0.25
4
Outliner
Search...
Item Label
Type
RockFlats_131
StaticMesh
RockFlats_47
StaticMesh
DecalActor21
DecalActor
414 actors (414 loaded)
Details
Select an object to view details.
Add
Import
Save All
All
Content
StackOBot
Settings
Collections

Undo & Redo


File Edit Window Tools Build Select Actor Help
StackOBot
MainWorld
Select Mode
Platforms
Settings
Perspective Lit Show
Video memory has been exhausted (154.198 MB over budget). Expect extremely poor performance.
Outliner
Search...
Item Label
Sequence
Type
SM_Ramp
ABP_Bot
NewLevelSequence
416 actors (416 loaded)
Details
Select an object to view details.
Content Browser
Sequencer
30 fps
NewLevelSequence*
Track
Search Tracks
0009
ABP_Bot
Animation
Spawned
Transform
16 items
Crash Recovery
Undo: Rename Track

Command

- Abstract all user operations to atomic commands which can invoke, revoke and serialize, deserialize.



Command – Definition

- ICommand<TData> provide a basic abstraction of the command.

- Every system (which want to support undo/redo/crash recory ...) needs to implement the system related commands inherantanced from ICommand<TData>.

```
public interface ICommand<TData>
{
    long UID { get; set; }

    TData Data { get; set; }

    void Invoke();

    void Revoke();

    byte[] Serialize();

    void Deserialize(byte[] data);
}
```

Command UID

Commands need strictly follow the sequence when recovery from disk.

- Monotonic increase over time
- Unique identification

```
public interface ICommand<TData>
{
    long UID { get; set; }

    TData Data { get; set; }

    void Invoke();

    void Revoke();

    byte[] Serialize();

    void Deserialize(byte[] data);
}
```

Command Serialize and Deserialize

- Provide functions to serialize command instance to data and deserialize data to command instance.
- TData type needs to provide serialize and deserialize interface.

```
public interface ICommand<TData>
{
    long UID { get; set; }

    TData Data { get; set; }

    void Invoke();

    void Revoke();

    byte[] Serialize();

    void Deserialize(byte[] data)
}
```

Three Key Commands

- Add
  - Data: Usually data is a copy of the runtime instance
  - Invoke: Create a runtime instance with data
  - Revoke: Delete the runtime instance
- Delete
  - Data: Usually data is a copy of the runtime instance
  - Invoke: Delete the runtime instance
  - Revoke: Create a runtime instance with data
- Update
  - Data: Usually data is the old and new values of the modified properties of the runtime instance and their property names
  - Invoke: Set the runtime instance property to the new value
  - Revoke: Set the runtime instance property to the old value

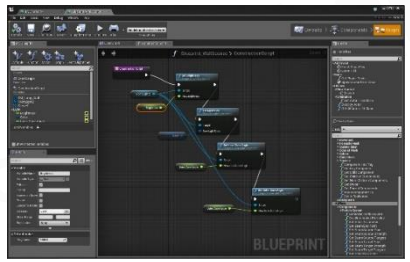
```
public interface ICommand<TData>
{
    long UID { get; set; }
    TData Data { get; set; }
    void Invoke();
    void Revoke();
    byte[] Serialize();
    void Deserialize(byte[] data);
}
```

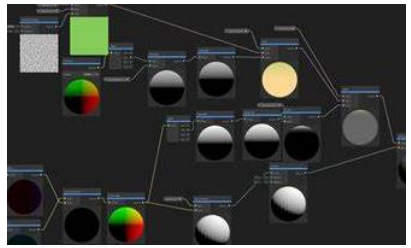
How to Make Tool Chain

Various Tools for Different Users

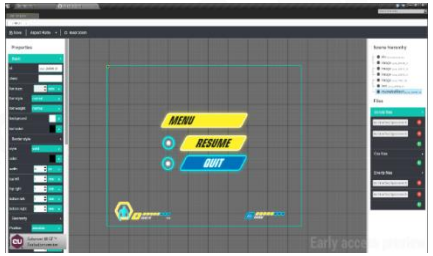
- Different viewes for different tools
- Each tool has it's owner data structure
- Same data may have different view for different user


Level Editor


Logical Editor


Shader Editor


Animation Editor



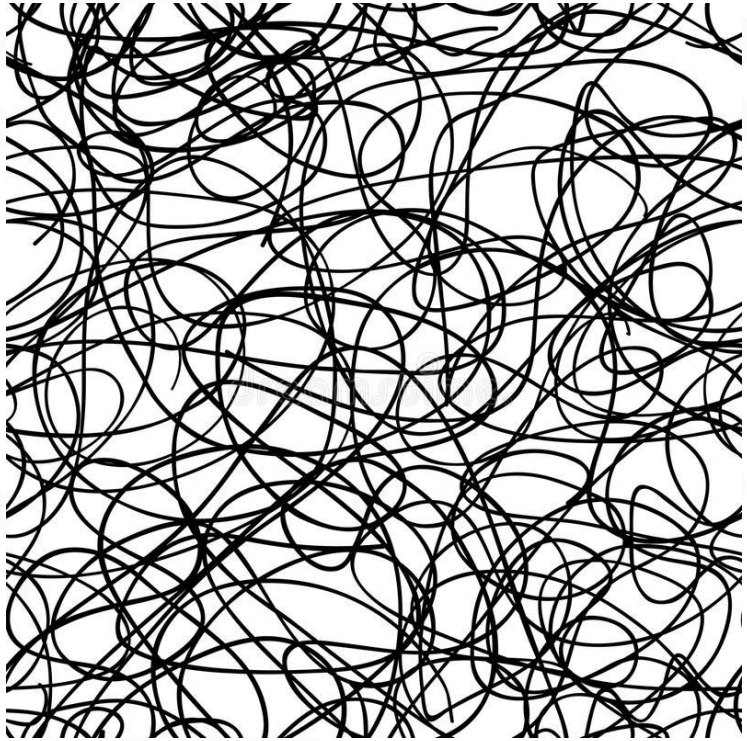
UI Editor


Static Mesh Editor

Develop all Tools Seperately?

Simplest Way

- No Scalability
- No maintainbility



Find Common Building Blocks

Any complex structure is made up of simple structures, we just need a standard language to describe it.

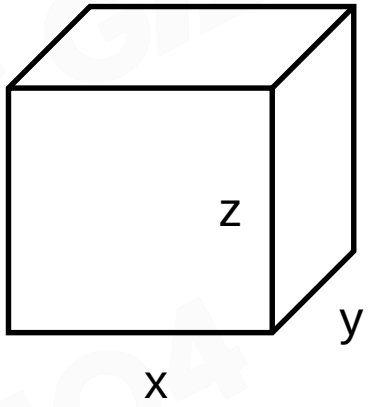

```cpp
class RuntimePointLight
{
public:
    TransForm  m_trans                          = TransForm::IDENTITY;
    Moability  m_moability                      = Moability::Movable;
    float      m_source_radius                  = 0.0f;
    float      m_soft_source_radius             = 0.0f;
    float      m_source_length                  = 0.0f;
    bool       m_use_temperature                = false;
    float      m_temperature                    = 0.0f;
    bool       m_affects_world                  = false;
    bool       m_cast_shadows                   = false;
    float      m_indirect_lighting_intensity    = 0.0f;
    float      m_volumetric_scattering_intensity = 0.0f;
    float      m_intensity                      = 0.0f;
    ColorValue m_light_color                    = ColorValue::White;
};

class TransForm
{
public:
    Vector3 m_location = Vector3::ZERO;
    Vector3 m_rotation = Vector3::ZERO;
    Vector3 m_scale    = Vector3::ZERO;

public:
    static const TransForm IDENTITY;
};
```

Schema - A Description Structure

A data schema is the formal description of the structures your system is working with.

Standardizing the world description language

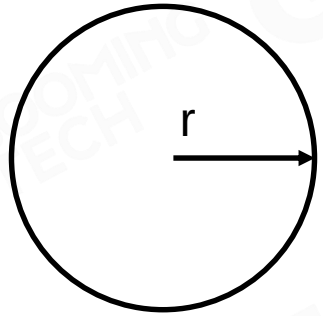
- Unified the data processor
- Normalized data between different tools
- Ability to automatically generate standardized UI



Float x;

Float y;

Float z;

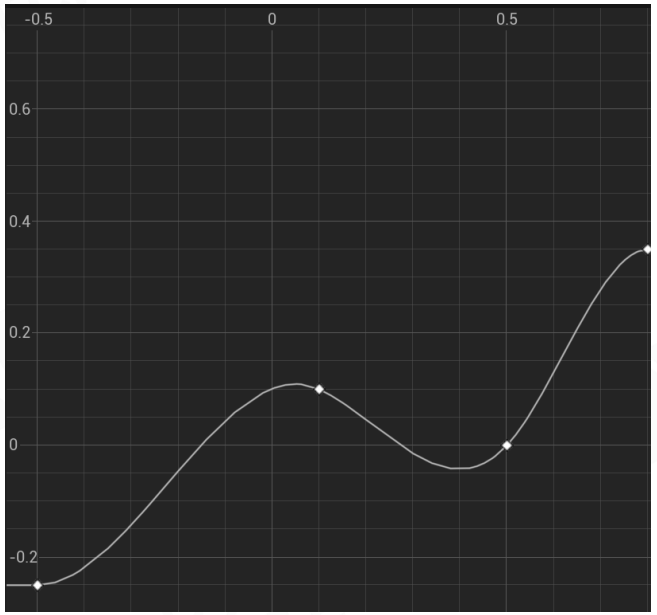


Float r;

Schema – Basic Elements

Abstraction of the basic building block of the world

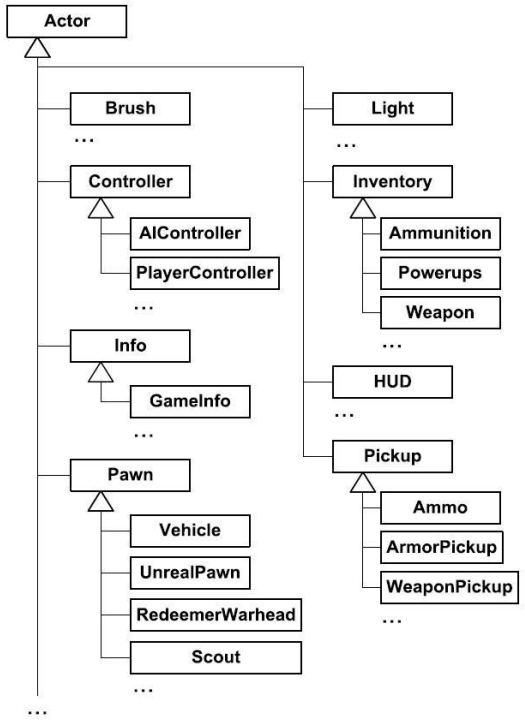
- Atomic Types: Int, Float, Double ...
- Class Type: Use atomic types to present complex data structure
- Containers: Array, Map ...




```
<enumDef name="CurvedParameterInterpolationType">
    <item name="linear"/>
    <item name="catmull"/>
</enumDef>

<classDef name="CurveControlPointBase">
    <element name="type" type="CurvedParameterInterpolationType" default="catmull"/>
    <element name="time" type="Float" default="0.0"/>
    <element name="guid" type="GUID" editor_only="true"/>
</classDef>

<classDef name="CurveControlPointFloat" base="CurveControlPointBase">
    <element name="value" type="Float"/>
</classDef>

<classDef name="CurveFloat">
    <array   name="control_points" type="CurveControlPointFloat" count="unbounded"/>
</classDef>
```

Schema – Inheritance

Abstraction of the inheritance relationship of the world




```
<classDef name="Texture3DDefinition"  base="TextureDefinitionBase">
    <element name="texture_size"            type="Int3" default="1 1 1"/>
    <array   name="texture_array"           type="rawRef:Texture" count="unbounded" />
</classDef>

<classDef name="RegularParamBase"  editor_virtual_type="true">
    <element name="name"       type="StringID"/>
    <element name="is_extern"  type="Bool"/>
    <element name="detail"     type="StringID"/>
</classDef>

<classDef name="RegularParamInt" base="RegularParamBase">
    <element name="value"                   type="Int"  default="0"/>
    <element name="is_range_check"          type="Bool" default="false"/>
    <element name="min_value"               type="Int"  default="0"/>
    <element name="max_value"               type="Int"  default="1"/>
    <element name="is_visible_parameter"    type="Bool" default="true"/>
</classDef>

<classDef name="RegularParamInt4" base="RegularParamBase">
    <element name="value"                   type="Int4" default="0 0 0 0"/>
    <element name="is_range_check"          type="Bool" default="false"/>
    <element name="min_value"               type="Int4" default="0 0 0 0"/>
    <element name="max_value"               type="Int4" default="1 1 1 1"/>
    <element name="is_visible_parameter"    type="Bool" default="true"/>
</classDef>
```

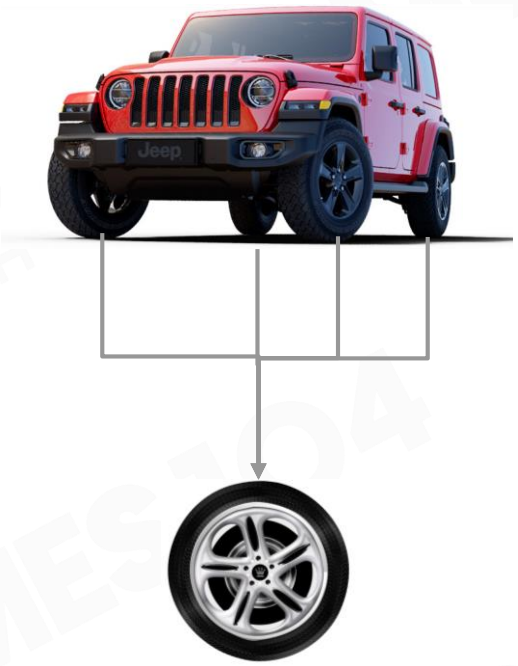
Inheritance of data structures

In the code,it is easy for a programmer.

```
class A{ /*...*/};
class B:A{/*...*/};
class C:A{/*...*/};
```

Schema – Data Reference

Abstract of the reference relationship of the world



```
<classDef name="Texture3DDefinition"  base="TextureDefinitionBase">
    <element name="texture_size"            type="Int3" default="1 1 1"/>
    <array   name="texture_array"           type="rawRef:Texture" count="unbounded" />
</classDef>

<classDef name="RegularParamBase"  editor_virtual_type="true">
    <element name="name"       type="StringID"/>
    <element name="is_extern"  type="Bool"/>
    <element name="detail"     type="StringID"/>
</classDef>

<classDef name="RegularParamInt" base="RegularParamBase">
    <element name="value"                  type="Int"  default="0"/>
    <element name="is_range_check"         type="Bool" default="false"/>
    <element name="min_value"              type="Int"  default="0"/>
    <element name="max_value"              type="Int"  default="1"/>
    <element name="is_visible_parameter"   type="Bool" default="true"/>
</classDef>

<classDef name="RegularParamInt4" base="RegularParamBase">
    <element name="value"                  type="Int4" default="0 0 0 0"/>
    <element name="is_range_check"         type="Bool" default="false"/>
    <element name="min_value"              type="Int4" default="0 0 0 0"/>
    <element name="max_value"              type="Int4" default="1 1 1 1"/>
    <element name="is_visible_parameter"   type="Bool" default="true"/>
</classDef>
```

Data reference

In the code, we need to read the data through the file path and instantiate it into the corresponding file class.

Schema – 2 Definition Ways

Standalone schema definition file

```
syntax = "proto2";

package tutorial;

message Person {
  optional string name = 1;
  optional int32 id = 2;
  optional string email = 3;

  enum PhoneType {
    MOBILE = 0;
    HOME = 1;
    WORK = 2;
  }

  message PhoneNumber {
    optional string number = 1;
    optional PhoneType type = 2 [default = HOME];
  }

  repeated PhoneNumber phones = 4;
}

message AddressBook {
  repeated Person people = 1;
}
```

Defined in code

```
UCLASS()
class UMG_API USpacer : public UWidget
{
    GENERATED_UCLASS_BODY()

public:

    /** The size of the spacer */
    UPROPERTY(EditAnywhere, BlueprintReadOnly, Category=Appearance)
    FVector2D Size;

public:

    /** Sets the size of the spacer */
    UFUNCTION(BlueprintCallable, Category="Widget")
    void SetSize(FVector2D InSize);

    // UWidget interface
    virtual void SynchronizeProperties() override;
    // End of UWidget interface

    // UVisual interface
    virtual void ReleaseSlateResources(bool bReleaseChildren) override;
    // End of UVisual interface

#if WITH_EDITOR
    virtual const FText GetPaletteCategory() override;
#endif

protected:
    // UWidget interface
    virtual TSharedRef<SWidget> RebuildWidget() override;
    // End of UWidget interface

protected:
    TSharedPtr<SSpacer> MySpacer;
};
```

Schema – 2 Definition Ways

Standalone schema definition file

Pros

- Comprehension easily
- Low coupling

Cons

- Ease to mismatch between engine version and schema version
- Difficult to define function in the structure
- Need to implement complete syntax

Defined in code

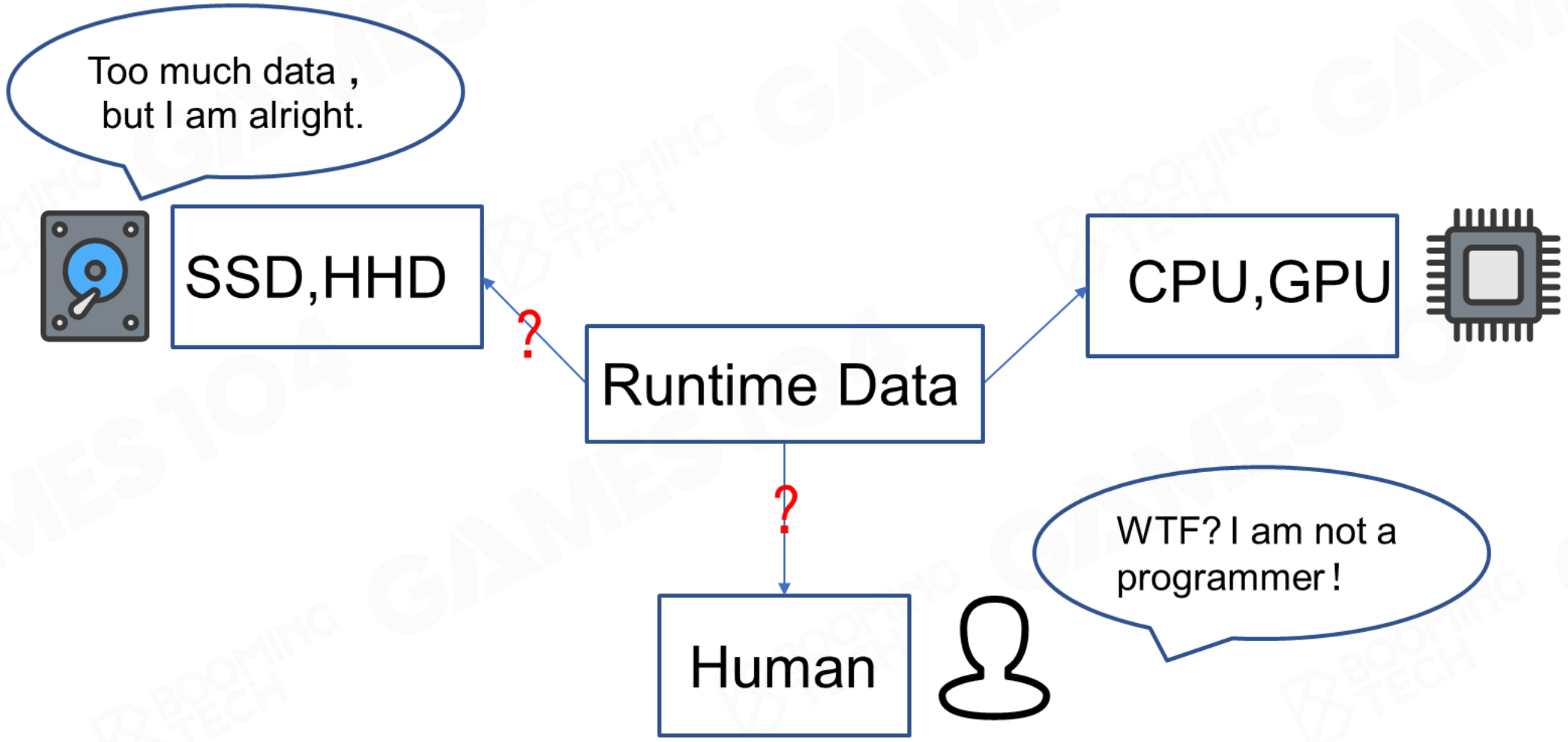
Pros

- Ease to accomplish Function reflection
- Natural support for inheritance relationships

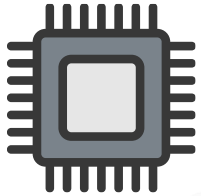
Cons

- Difficult to understand
- High coupling

Three Views For Engine Data



Runtime View

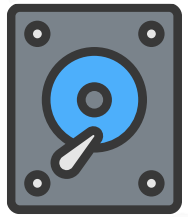


Focus:

- Read at a faster speed
- Calculate at a faster speed

```
class RuntimeSpotLight
{
public:
    // Spot Light Translation Matrix
    Matrix4x4 light_trans {Matrix4x4::IDENTITY};
    // Spot Light Cone
    float inner_cone_radian = 0.0f;
    float outer_cone_radian = 0.0f;
    // Spot Light intensity and units
    float      intensity = 0.0f;
    LightUnits unit      = CANDELA;
    // Spot Light Color
    Vector4 light_color {Vector4::ZERO};
    // other light data like shadow...
};
```

Storage View



Focus:

- Write at a faster speed
- Occupies less hard disk space

```
//Trans
"Position:X": 1.0,
"Position:Y": 1.0,
"Position:Z": 1.0,
"Rotation:X": 0.0,
"Rotation:Y": 0.0,
"Rotation:Z": 0.0,
"Rotation:W": 1.0,
"Scale:X": 1.0,
"Scale:Y": 1.0,
"Scale:Z": 1.0,
//cone_degree
"inner_cone_degree": 30,
"outer_cone_degree": 60,
//sds
"intensity ": 0.0,
"unit": 1
//other data...
```

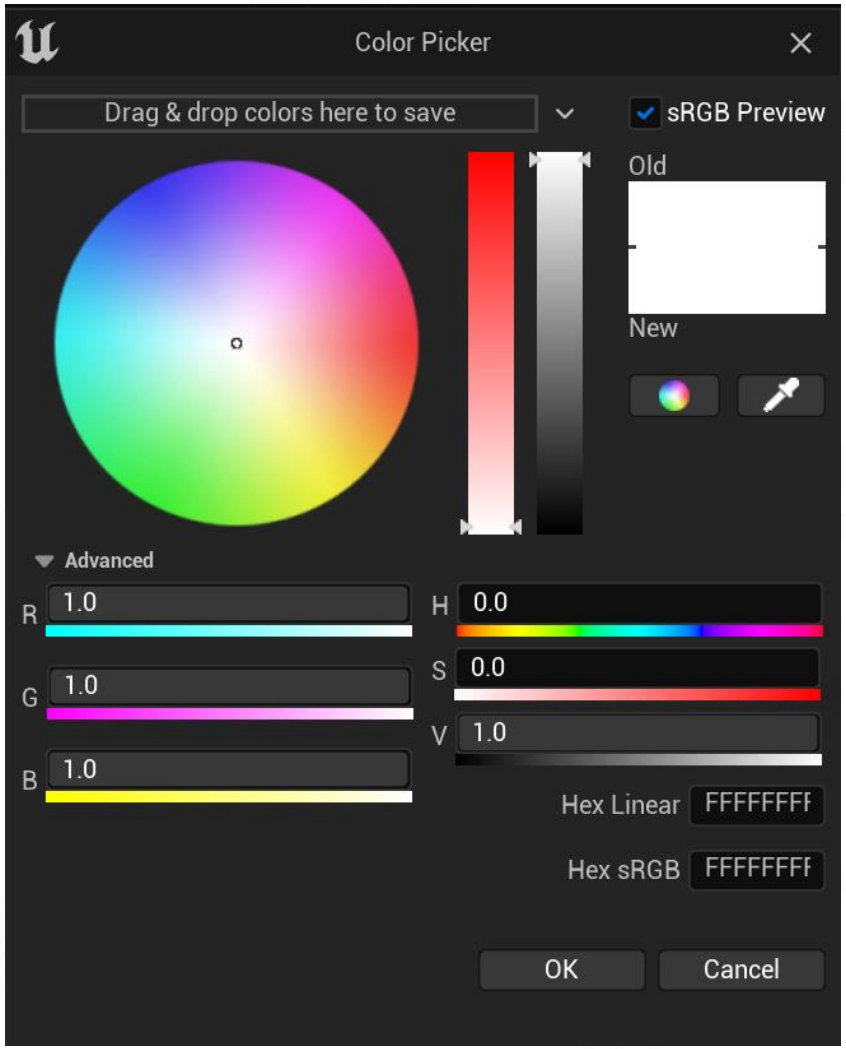
Tools View

Focus:

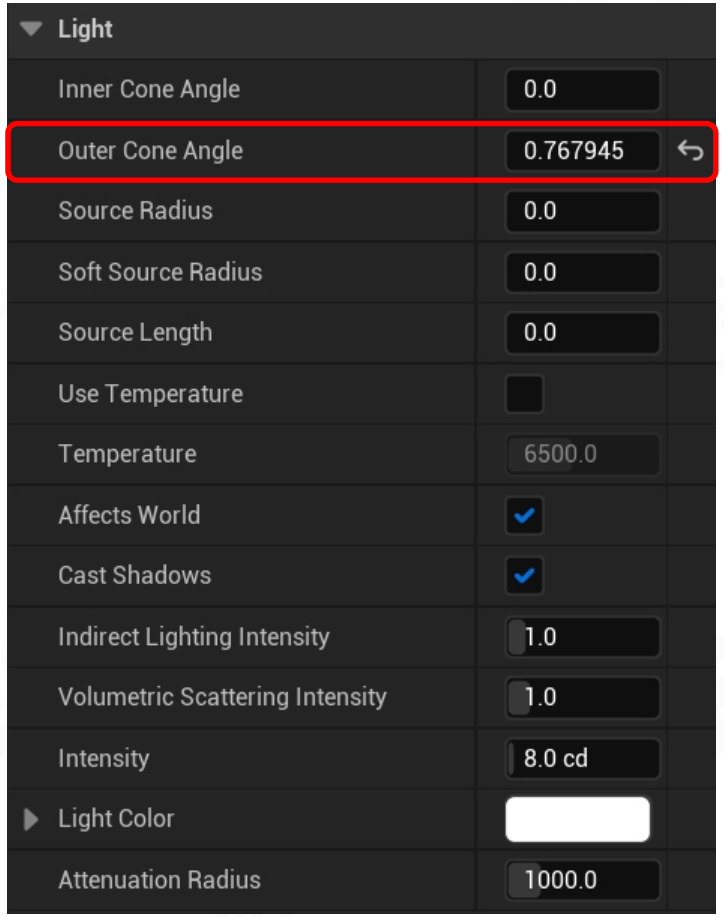
- More understandable form
- The need for multiple editing modes

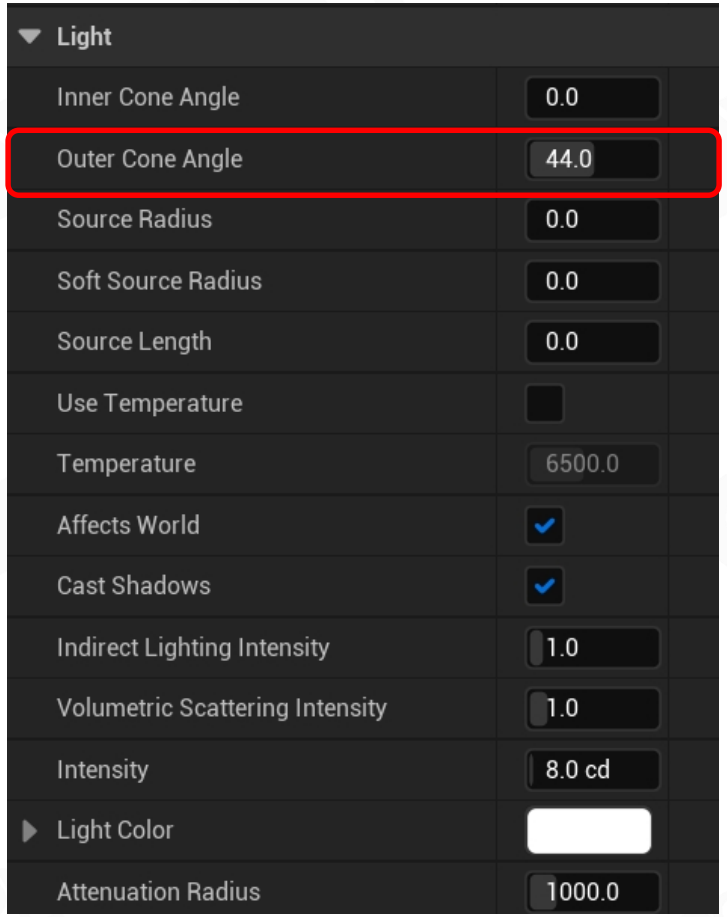
Other Point：

Tool data does not generally exists. Usually, special processing is done when the UI interface is generated.



Tools View – Understandable

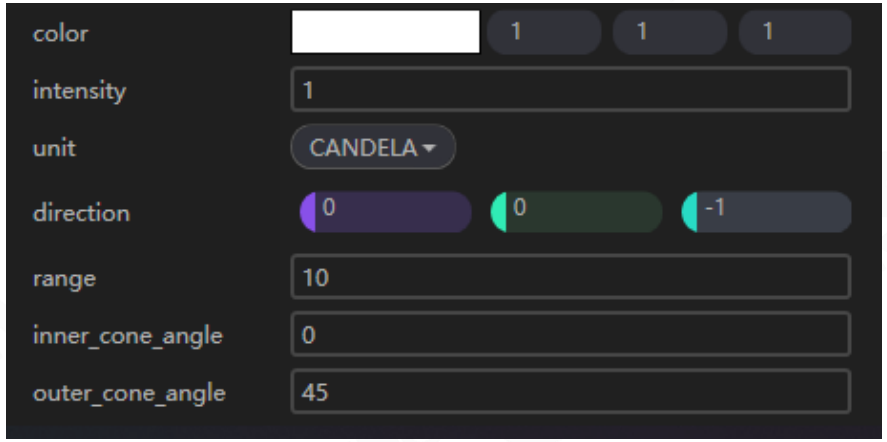




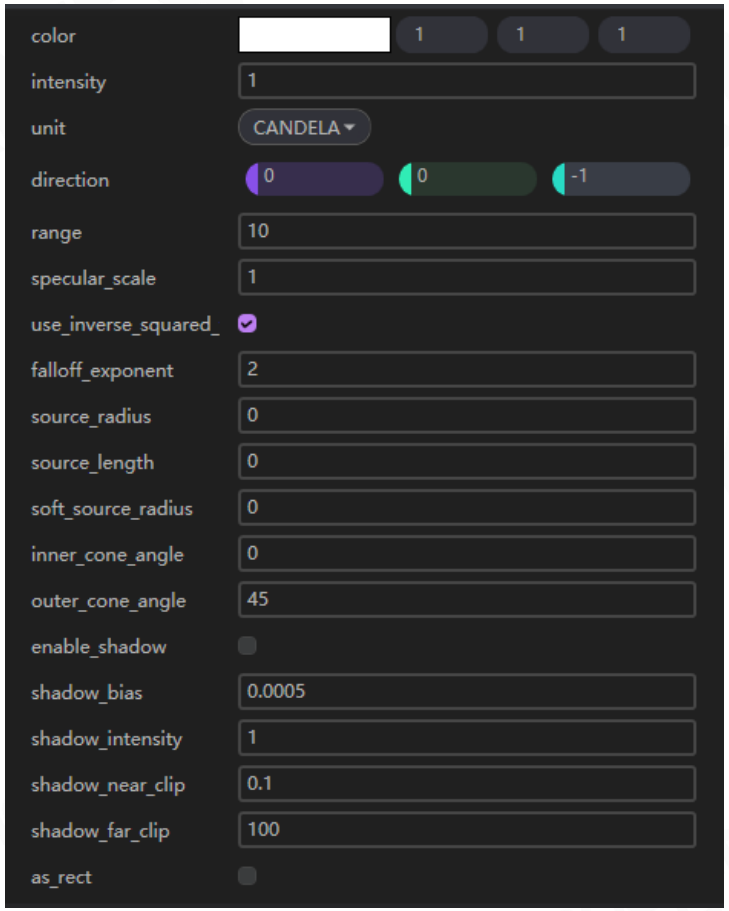
Euler angle is your friend！ Radian is not.

Tool View – Various Editor Modes

Different edit mode for groups with different needs



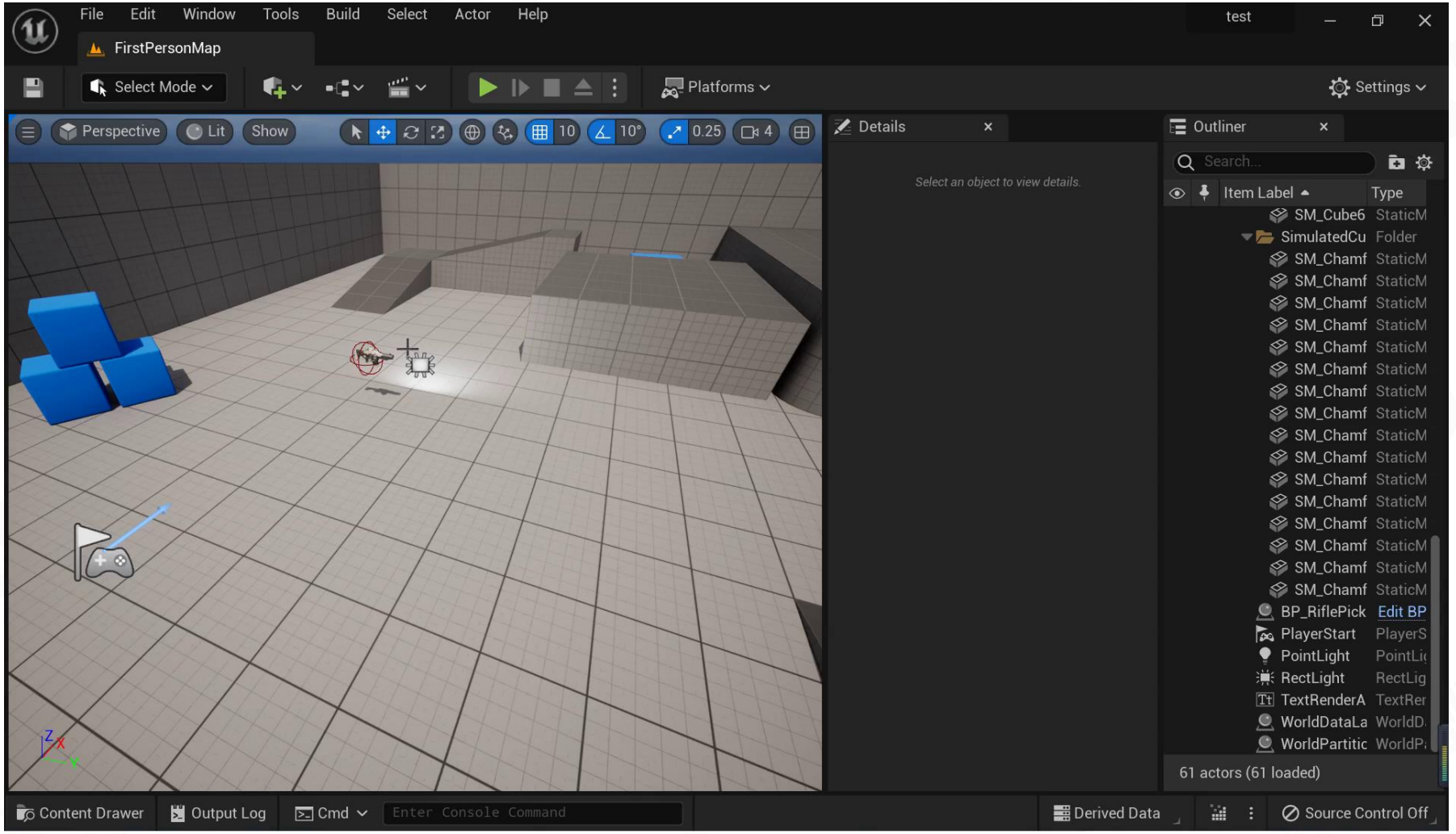
Beginer Mode



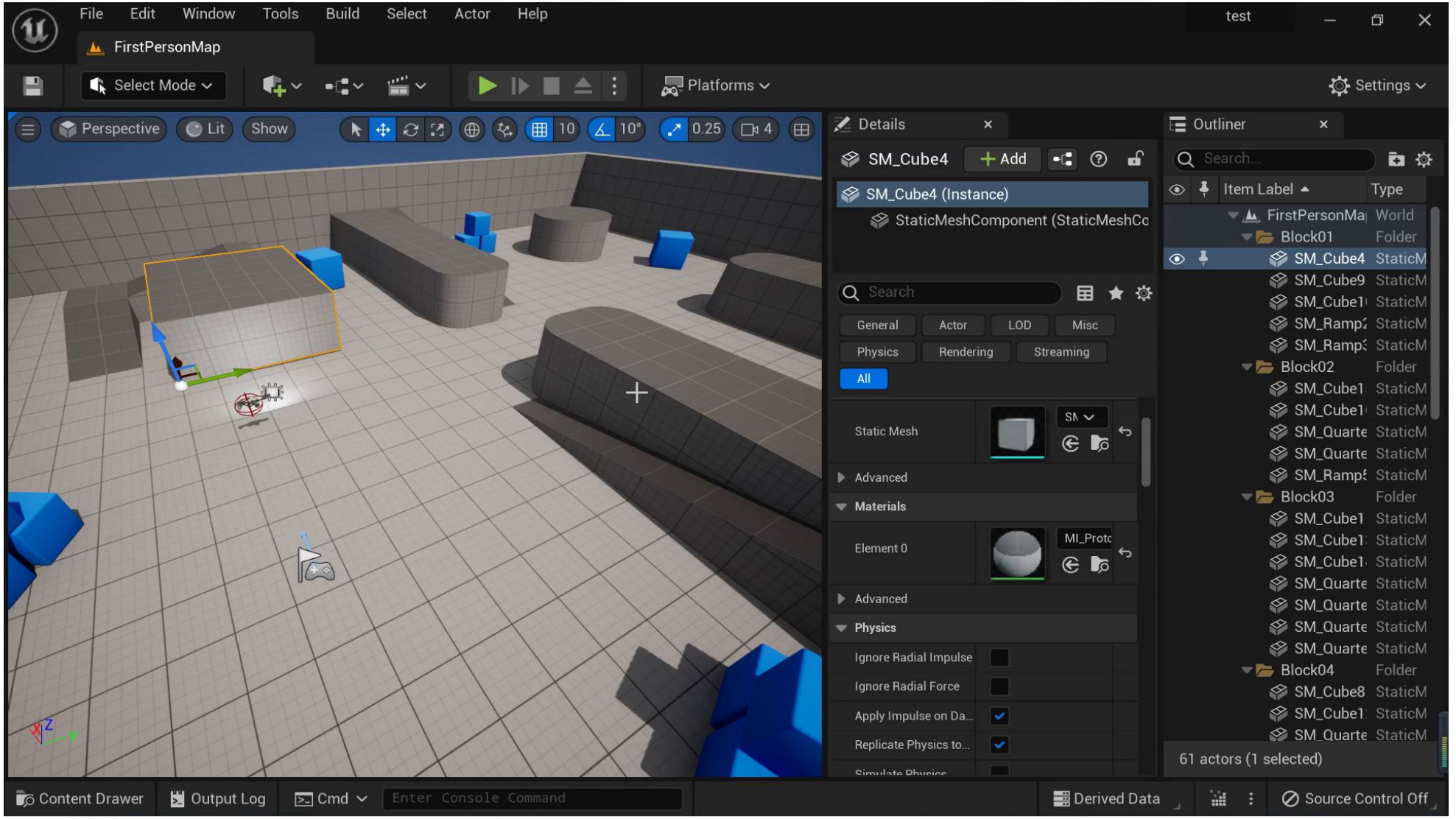
Expert Mode

What You See is What You Get (WYSIWYG)

User Friendly for Artists



User Friendly for Designer



Stand-alone Tools

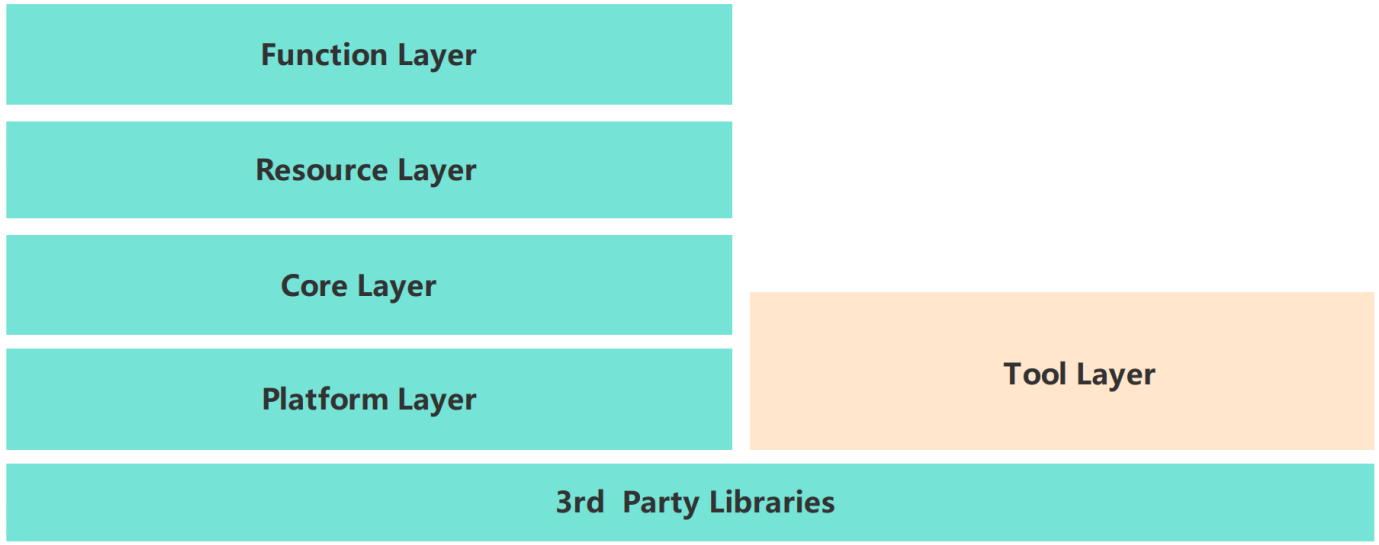
Stand-alone Tools is a kind of tool that can run independently of the engine.

Pros

- Suitable for use as a DCC tool plug-in
- Easy to start developing tools

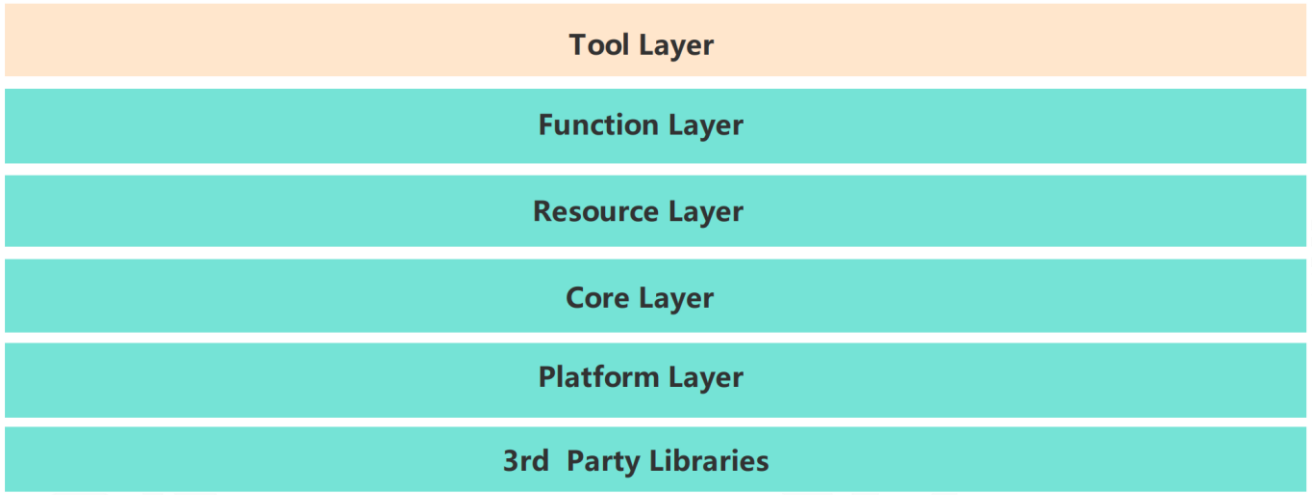
Cons

- Difficult to achieve WYSIWYG



In Game Tools

In Game Tools is a kind of tool based on engine runtime system work.



Pros

- Access to all engine data directly
- Easy to preview the game in the editor
- Easy to make live in-game editing

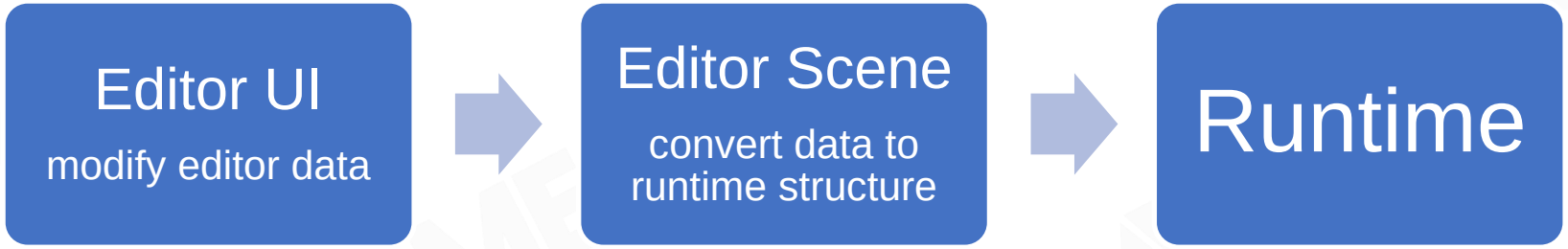
Cons

- Complex engine architecture
- Requires a complete engine UI system to make the editor UI
- When the engine is crashing, the tools become unusable as well

In Game Tools – Editor Mode

Editor Mode: Support to modify and preview scene data

- Real time preview of scene data modification
- Logic systems do not tick, so there are more hardware resources to display more scene details
- ...



Play in Editor (PIE)

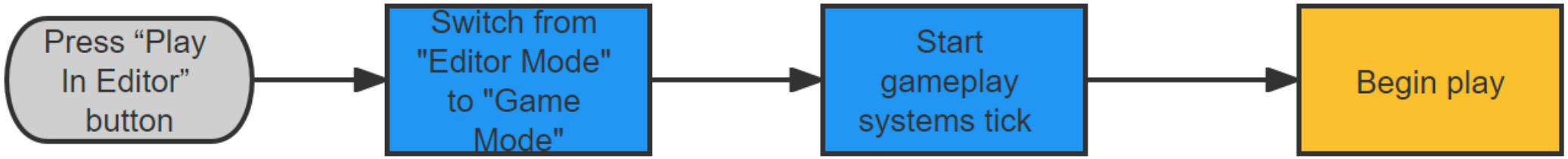
PIE: Directly play game in editor, no need to close editor and start game mode

- Save loading time
- The continuity of creation is maintained
- Quickly test modifications
- ...

Two implemation ways

- **Play in editor world:** Start gameplay systems tick in editor world and play in it
- **Play in PIE world:** Duplicate editor world to create a PIE world and play in it

PIE Mode - Play in Editor World



Pros

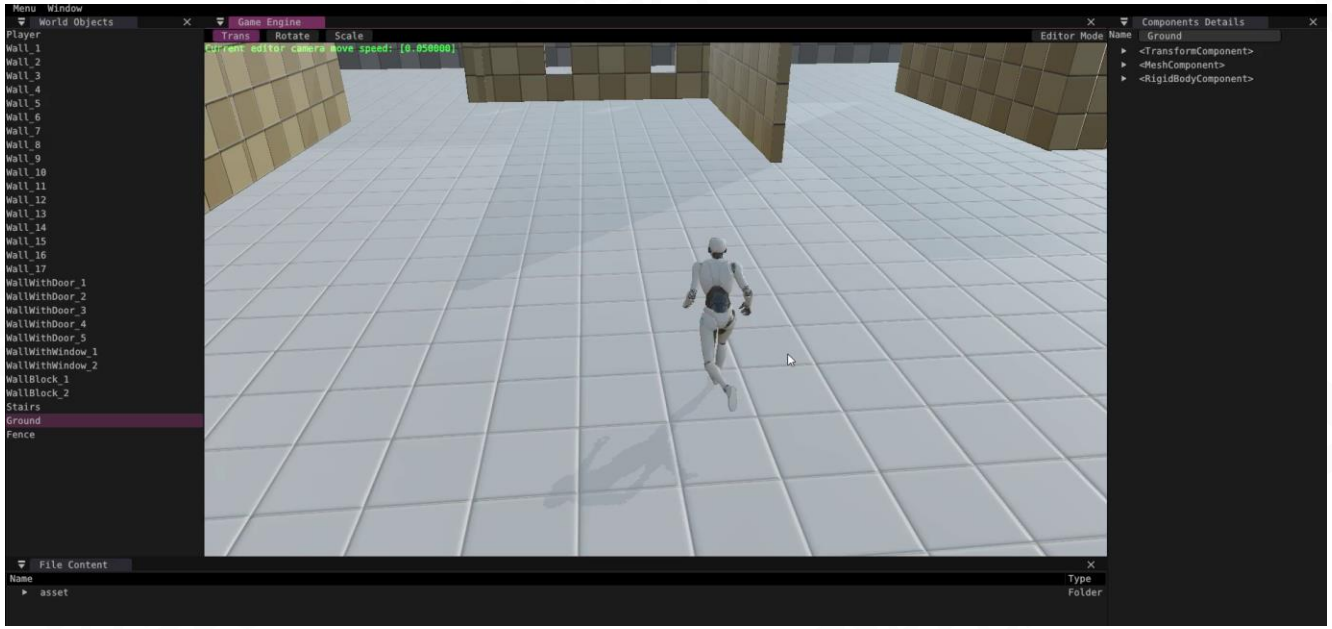
- Easy architecture tools layer
- Quick state change

Cons

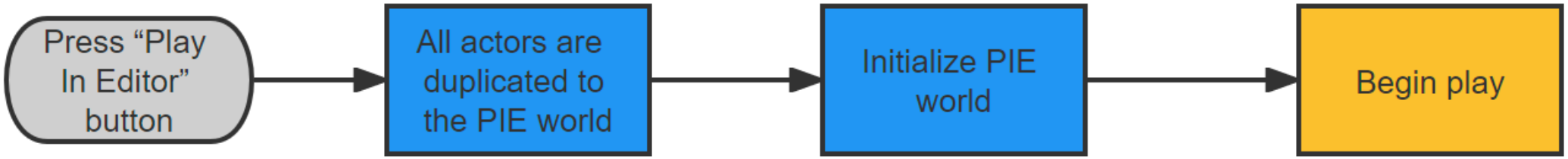
- Game mode may cause data changes

Example

- Piccolo



PIE Mode - Play in PIE World



Pros

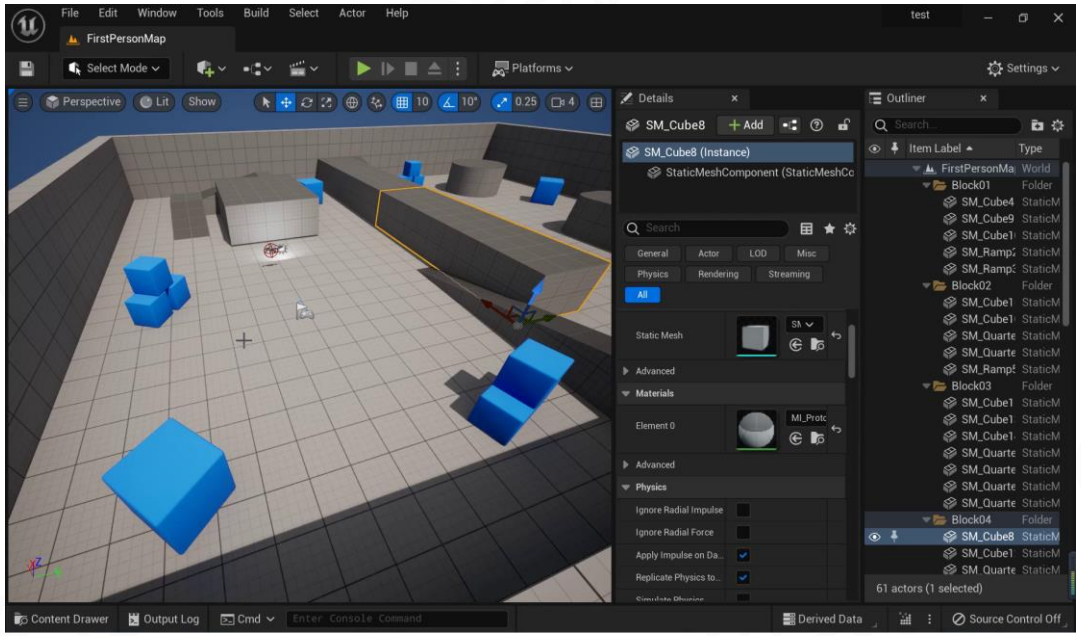
- Data separation
- Easy to instantitate multiple game instances

Cons

- Architecture complex

Example

- Unreal



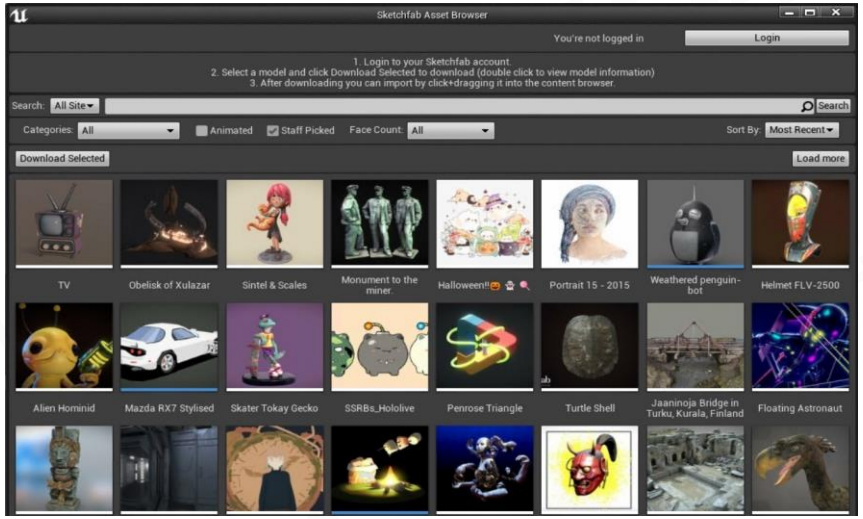
One More Thing - Plugin

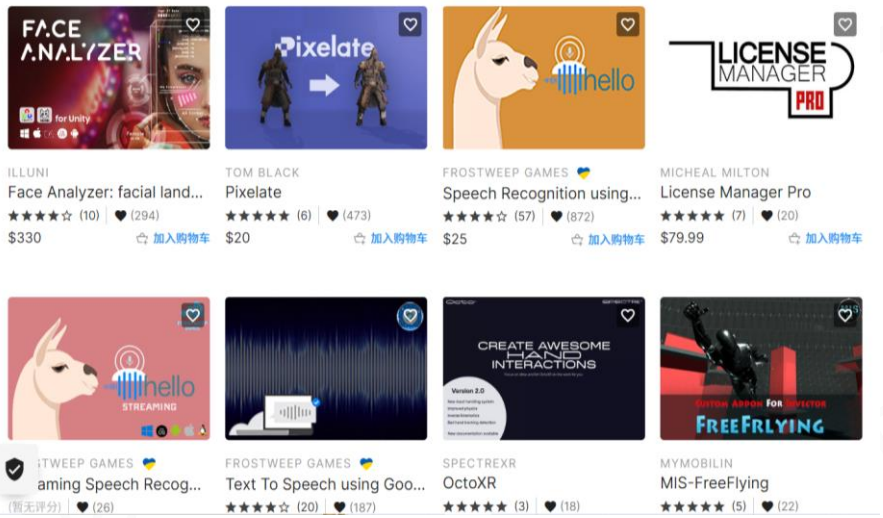
Extensibility

Different games need different customization of engine tools.

Engine tools use **plug-in** mechanism to satisfy the needs.

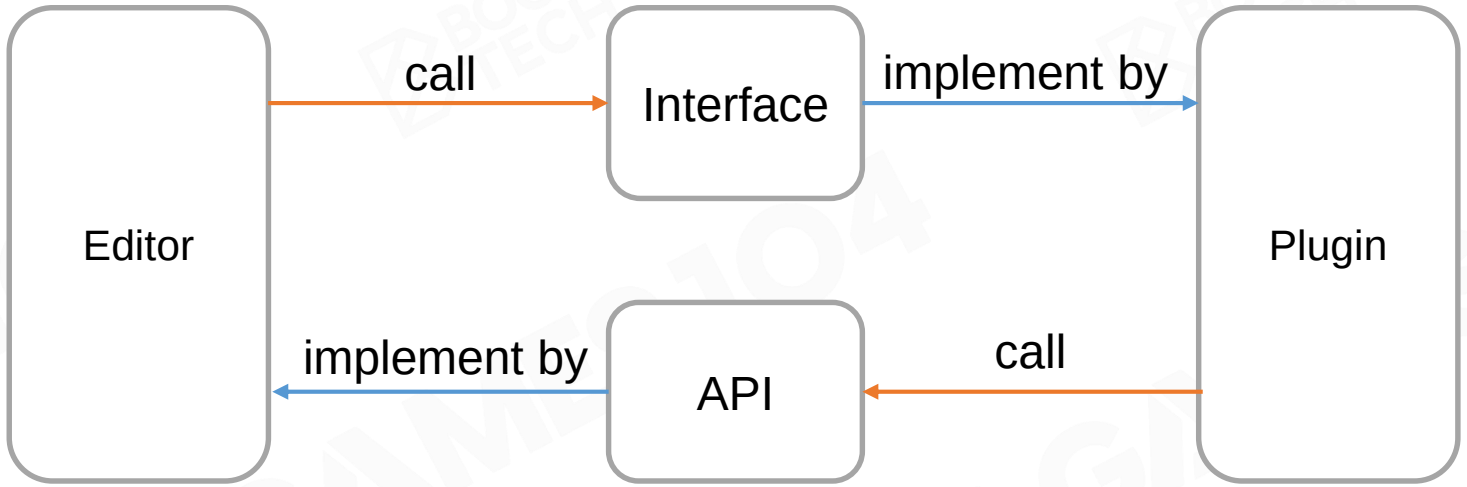
Plug-in - Showcases





Plug-in – Framework (1/2)

Plug-in : A software component that adds a specific feature to an existing computer program.



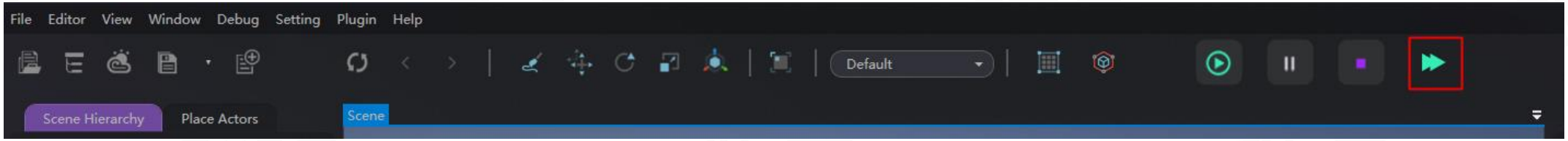
Plug-in – Framework (2/2)

PluginManager: Manage plugin loading and unloading.

Interface: A series of abstract classes provided to plug-ins, plug-ins can choose to instantiate different classes to realize the development of corresponding functions.

API: A series of functions exposed by the engine, plug-ins can use functions to execute the logic what we want.

Plug-in – Add a Toolbar Button



API & Interface

```
class IPluginInterface
{
public :
    virtual void LoadPlugin();
    virtual void UnLoadPlugin();
};

class SDK
{
public:
    static void AddToolbarButton();
    static void AddSettingMenu();
    static void AddToolBarContextMenu();
    //...other
};
```

plugin

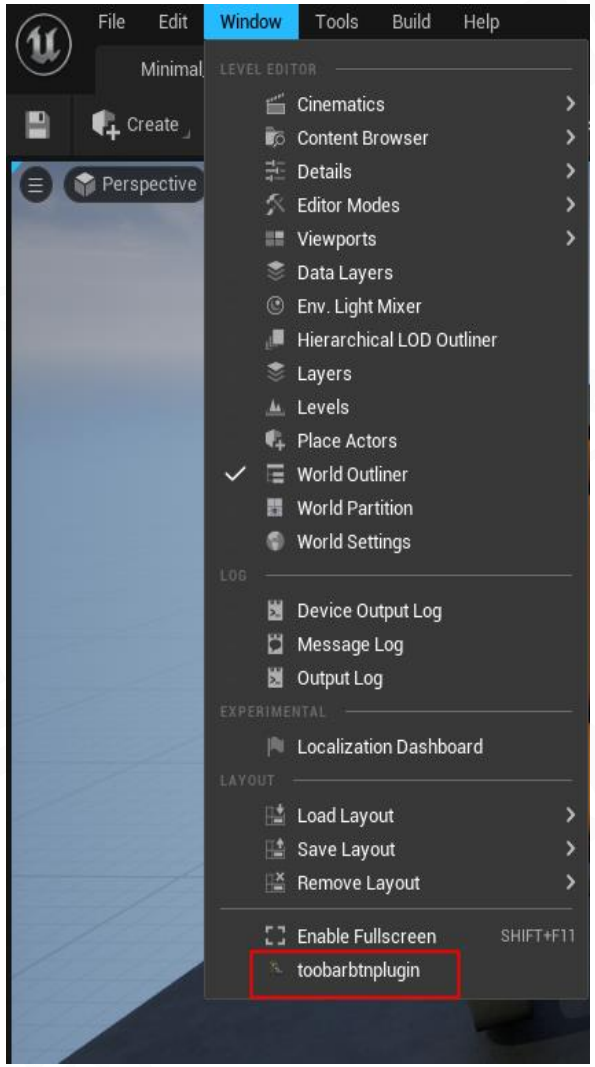
```
class MyPlugin :IPluginInterface
{
public:
    virtual void LoadPlugin() override {
        SDK::AddToolbarButton();
    }
    //...other implement
};
```

editor

pluginManagerModule

```
class PluginManager
{
    void LoadPlugin(string plugin_dll_path)
    {
        IPluginInterface * instance
            = GetPluginInstance(plugin_dll_path);
        instance->LoadPlugin();
    }
    IPluginInterface * GetPluginInstance(string path)
};
```

Plug-in – Add a Plug-in Menu in Unreal5



Implement interface

```
class FtoobarbtnpluginModule : public IModuleInterface
{
public:
    /** IModuleInterface implementation */
    virtual void StartupModule() override;
    virtual void ShutdownModule() override;
othercode
};
```

In StartupModule() call RegisterMenus()

```
void FtoobarbtnpluginModule::RegisterMenus()
{
    // Owner will be used for cleanup in call to UToolMenus::UnregisterOwner
    FToolMenuOwnerScoped OwnerScoped(this);
    {
        UToolMenu* Menu = UToolMenus::Get()->ExtendMenu("LevelEditor.MainMenu.Window");
        {
            FToolMenuSection& Section = Menu->FindOrAddSection("WindowLayout");
            Section.AddMenuEntryWithCommandList(FtoobarbtnpluginCommands::Get().PluginAction, PluginCommands);
        }
    }
    {
        UToolMenu* ToolbarMenu = UToolMenus::Get()->ExtendMenu("LevelEditor.LevelEditorToolBar");
        {
            FToolMenuSection& Section = ToolbarMenu->FindOrAddSection("Settings");
            {
                FToolMenuEntry& Entry = Section.AddEntry(FToolMenuEntry::InitToolBarButton(FtoobarbtnpluginCommands
                Entry.SetCommandList(PluginCommands);
            }
        }
    }
}
```

Plug-in – Summary

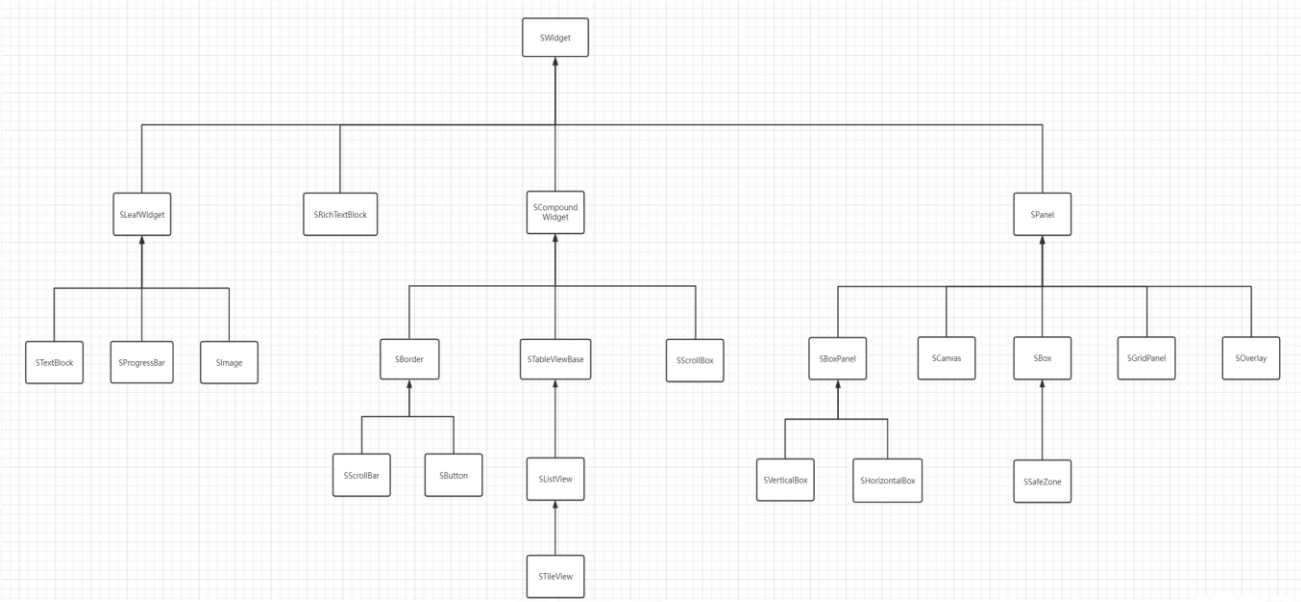
The meaning of plug-in framework

- Extend editor functionality
- Ease to hot update as decoupling
- Facilitate the construction of engine development ecology

Plug-in framework requirements

- Full API support
- Common interface support

API example: unreal slate



References

- Tools Tutorial Day: A Tale of Three Data Schemas, Ludovic Chabant, GDC 2018:https://media.contentapi.ea.com/content/dam/eacom/frostbite/files/gdc18-tools-tutorial-day-a-tale-of-three-data-schemas.pptx
- Creating a Tools Pipeline for 'Horizon: Zero Dawn', Dan Sumaili, Sander van der Steen, GDC 2017: https://www.guerrilla-games.com/media/News/Files/GDC2017_Sumaili_VanDerSteen_CreatingAToolsPipelineForHorizonZeroDawn.pdf
- Unreal Engine UProperties:https://docs.unrealengine.com/5.0/en-US/unreal-engine-uproperties/
- Command Pattern:https://www.tutorialspoint.com/design_pattern/command_pattern.htm
- Unreal Plugins:https://docs.unrealengine.com/4.27/en-US/ProductionPipelines/Plugins/

- Model–view–controller:https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller
- Trygve Reenskaug:https://en.wikipedia.org/wiki/Trygve_Reenskaug
- MVC:https://developer.mozilla.org/en-US/docs/Glossary/MVC
- MVC:https://folk.universitetetioslo.no/trygver/themes/mvc/mvc-index.html
- Benefits and Drawbacks of MVC Architecture:https://shreysharma.com/benefits-and-drawbacks-of-mvc-architecture/
- Model–view–presenter:https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93presenter
- Model–view–viewmodel:https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel

Lecture 13 Contributor

- Wood
- 霓虹甜心
- 新之助
- BOOK
- 阿甘
- 爵爷
- 令狐冲
- 大喷
- Qiuu
- Adam
- Arthas
- 喵小君
- 小鲤鱼
- 布鲁布鲁
- kaiwei
- 33
- 小明
- 蓑笠翁
- 晨晨
- 怡宝

Q&A

Enjoy ;)
Coding

Course Wechat

Please follow us for further information