本节课程内容

1. UE学习途径和方法的介绍
2. UE编辑器使用和编程技巧
3. UE引擎工具了解

本节课程作业

1. 源码编译UE5，新建一个C++工程，进行简单场景编辑和工程设置
2. 编译并构建安装包，确保能够用来将游戏工程安装到手机正常运行（若无Andorid手机可构建桌面版本）

课程目标

1. 熟悉UE引擎的编辑器操作，了解UE引擎的游戏模式框架
2. 能够独立获取和编译UE源码
3. 能够在UE引擎中实现蓝图编程和C++编程
4. 能够使用UE引擎构建Andorid平台游戏安装包

游戏团队典型结构

<table>
<tr><td colspan="5">制作人</td><td></td></tr>
<tr><td colspan="2">程序</td><td colspan="2">美术</td><td>策划</td><td rowspan="8">PM</td></tr>
<tr><td>前台</td><td>后台</td><td colspan="2">原画</td><td>数值</td></tr>
<tr><td colspan="2">游戏性</td><td colspan="2">建模</td><td>关卡</td></tr>
<tr><td colspan="2">周边系统</td><td colspan="2">地编</td><td>系统</td></tr>
<tr><td colspan="2">工具</td><td colspan="2">动画</td><td>剧情</td></tr>
<tr><td colspan="2">性能分析</td><td colspan="2">特效</td><td>战斗</td></tr>
<tr><td colspan="2">引擎</td><td colspan="3"></td></tr>
<tr><td>音频</td><td>安全</td><td>运营</td><td>发行</td><td>账号</td></tr>
</table>

> 游戏工作室(game studio)通常由5个基本专业领域的人员构成，包括工程师(engineer)、艺术家(artist)、游戏
> 设计师(game designer)、制作人(producer)及其他管理/支持人员(市场策划、法律、信息科技/技术支持、行政等)。
>
> ——游戏引擎架构

游戏引擎简介

Unreal Engine的优势

- 渲染品质：电影级别PBR渲染，先进的着色模型
- 美术制作：成熟的美术资产制作管线
- C++与蓝图：性能与可视化编程并重
- 开发周期：基于射击类的GamePlay框架
- 跨平台：移动、主机、PC、VR...
- 开源：有利于技术提升和定制化改造

还有Unity，CryEngineV，Open3DEngine，Source2，FrostbiteEngine等等.....

游戏引擎架构

- 游戏引擎：专门为游戏而设计的工具及科技集合
- Game enaine are data-driven architectures that are reusable and therefore do not contain game content（游戏引擎是可重用的数据驱动架构，因此不包含游戏内容）
- 通用性与偏向性 游戏行为全部或部分由美术或者策划提供的数据控制
- 可扩展性
- 完善工具链

<table>
<tr><td colspan="2">游戏子系统</td><td rowspan="2">世界编辑器</td></tr>
<tr><td>动画</td><td>游戏基础系统</td></tr>
<tr><td>渲染引擎</td><td>物理引擎</td><td rowspan="2">工具</td></tr>
<tr><td colspan="2">资源管理</td></tr>
<tr><td colspan="3">核心系统</td></tr>
<tr><td colspan="3">平台独立层</td></tr>
<tr><td colspan="3">第三方SDK</td></tr>
<tr><td colspan="3">操作系统</td></tr>
<tr><td colspan="3">驱动</td></tr>
<tr><td colspan="3">硬件</td></tr>
</table>

图 1.15　运行时引擎架构

游戏引擎: 渲染

- Deferred Renderer 延迟渲染
  - 编辑器、PC、Console默认渲染管线
  - Feature levels “SM4”,“SM5” （Shading Model）
- Forward Rendered 前向渲染
  - 用于桌面VR游戏，支持MSAA
  - Mobile Renderer
- Mobile Rendering

- Forward Render, Deferred Render

由于手机性能限制，所以移动渲染分Tile。

在传统的渲染管线中，图像的渲染通常是按照逐像素的方式进行，即每次处理整个屏幕上的像素。而在Tile渲染中，整个屏幕被划分为多个较小的矩形区域（瓦片），每个瓦片会被独立渲染。

- Immediate Mode Rendering

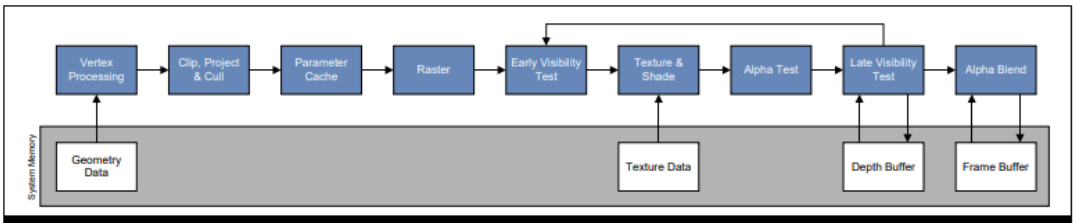


Figure 3. IMR pipeline

- Tile Based Rendering

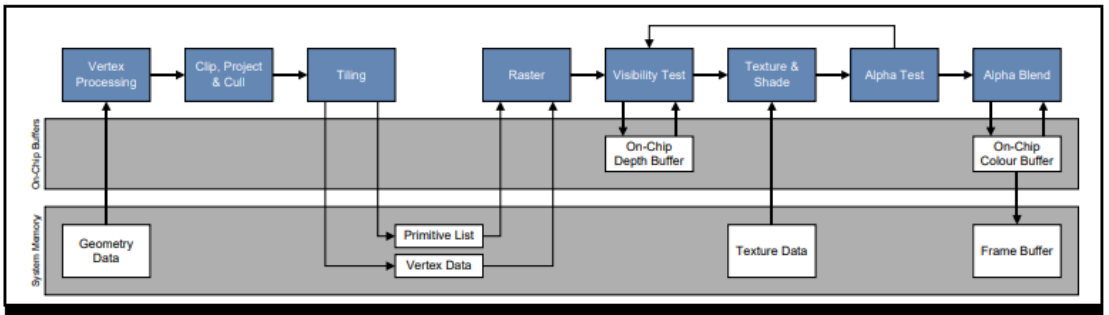


Figure 4. TBR pipeline

- Tile Based Deferred Rendering

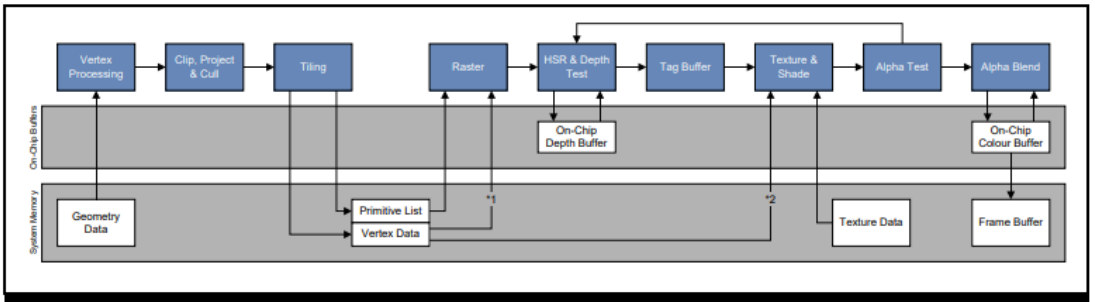
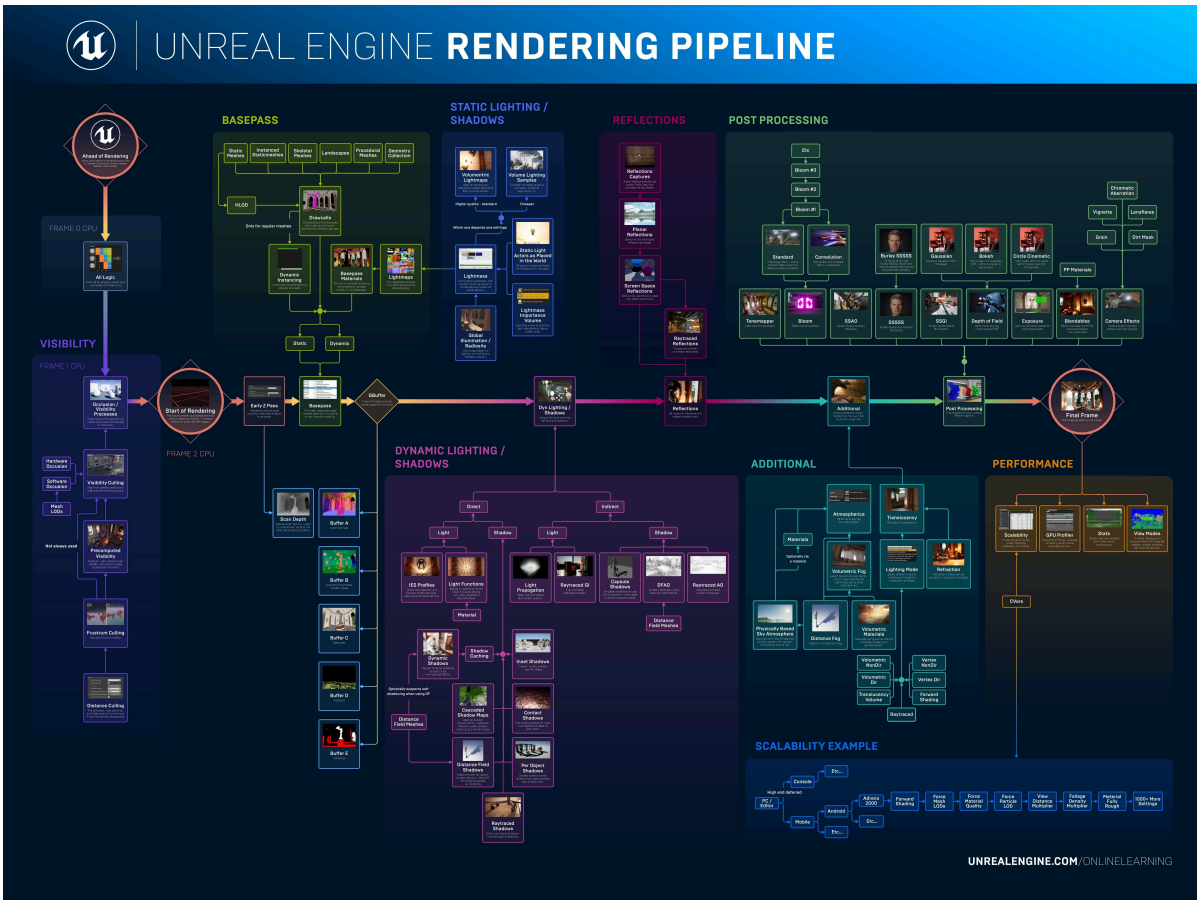


Figure 5. TBDR pipeline



手机端渲染还需要注意片上内存（手机显存）

> 片上内存带宽大访问速度快，主存大一些但是访问慢一些。
>
> - 当渲染时，GPU只需要在每个瓦片内执行光照、着色和纹理采样等计算，这样每个瓦片处理的像素数量较少，可以在更小的内存范围内进行计算。
> - 每个瓦片的处理是独立的，GPU可以有效地利用显存中存储的纹理、着色器和帧缓冲等资源，而不必频繁访问内存数据。这种分块的处理方式可以更高效地利用显存，避免大量的内存带宽消耗。由于瓦片大小相对较小，显存访问模式也更有规律，有助于提升缓存命中率。（缓存的局部性原理）
> - Tile渲染减少了GPU对主存的依赖，因为它减少了渲染过程中对主存数据的频繁访问。由于主存的速度通常低于显存，减少主存的访问可以进一步提高性能。

游戏引擎：物理

物理引擎包含的内容：

- 碰撞检查
- 动态约束
- 刚体物理
- 车辆物理
- 布娃娃系统

游戏物理引擎：

- Havok
  - Physics、Destruction、Cloth、AI、Behavior、Animation、FX
  - 被Intel、微软收购，CPU友好

    - https://www.havok.com/
- PhysX
    - 集成于UE4和Unity
    - 被NVIDIA收购，GPU友好
    - https://developer.nvidia.com/physx-sdk
    - https://github.com/NVIDIAGameWorks/PhysX
- Bullet
    - 最早开源，用于GTA5、荒野大镖客
    - https://pybullet.org/wordpress/
    - https://github.com/bulletphysics/bullet3
- Chaos in UE
    - 实时电影效级的大范围破坏

> 感觉Chaos的破碎类似Houdini，Houdini先使用Voronoi Fracture撒点破碎，破碎结果按大到小，每个碎块也有物理代理体，结果完成后进行物理模拟。UE也是先破碎，然后游戏里对破碎效果进行物理模拟，每个碎块也有物理代理体。

Unreal Engine 介绍

- UE引擎是什么样子的?
- 学习UE该怎么入手

背景介绍

UE历史随便看看

学习资料：Unreal Engine - YouTube，虚幻引擎官方-哔哩哔哩，虚幻引擎 - 知乎

Launcher

Launcher介绍了一下EpicGames启动器，有免费资产

推荐配置

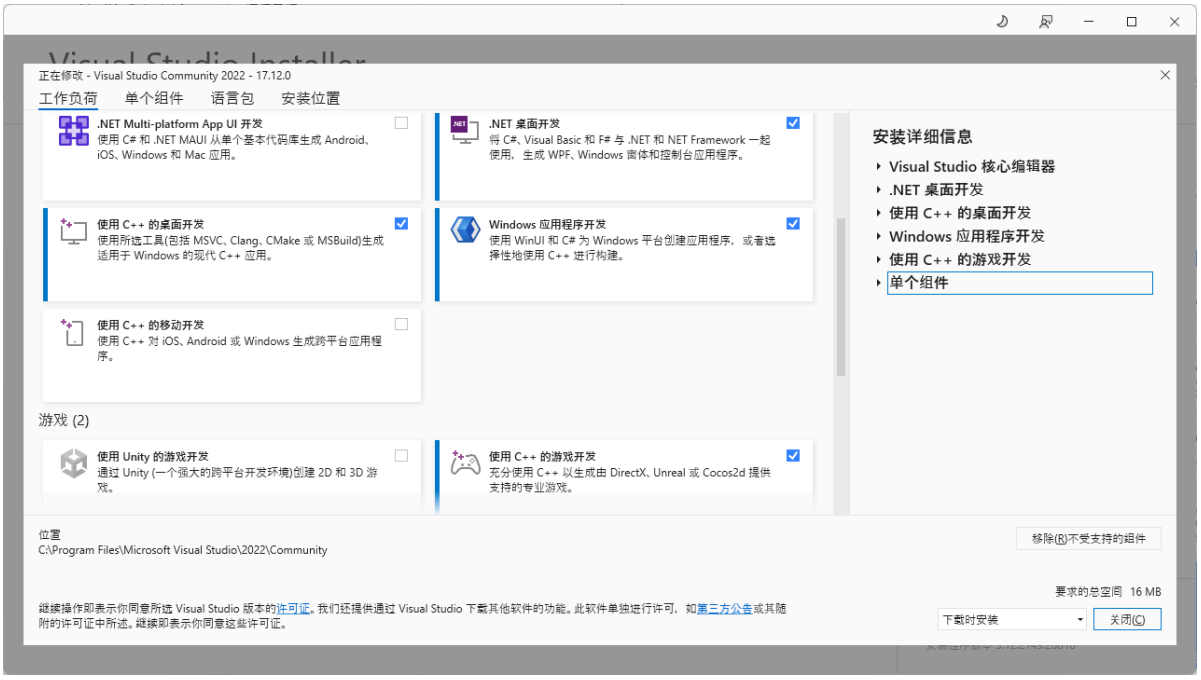
Hardware and Software Specifications for Unreal Engine | Unreal Engine 5.5 Documentation | Epic Developer Community | Epic Developer Community

> 由于接触的工程可能会比较大，如果内存较小，会导致资产频繁和硬盘交换，导致编辑器卡顿
>
> 工程资产也需要从硬盘加载到内存，所以硬盘读写过慢也会导致卡顿
>
> UBT编译虚幻引擎源码会占用C盘空间，建议C盘100G以上空闲空间，并且UE引擎源码编译所在盘预留400G以上空间
>
> （开源不从源码编译简直是失去了上开源社区的乐趣）
>
> 一句话：能上多好上多好，按打游戏的配置上最好的

虚幻引擎5.5版本说明 | 虚幻引擎 5.5 文档 | Epic Developer Community | Epic Developer Community

简而言之：VS用最新版，安卓打包还需要安装AndroidStudio，也是安装最新版，AndroidStudio打包安卓包需要额外安装**Android SDK Command-line Tools (latest)**

VS安装完成后选择这四个选项继续安装



编辑器使用

虚幻引擎编辑器路径：`\Engine\Binaries\Win64\UnrealEditor.exe`

> 这个路径挺有用的，所有编译的可执行文件都在这里，例如如果写服务器，并且服务器的引擎模块，编译的服务器也会在这里

- 不仅是一个世界编辑器
- 管理整个游戏资产数据
- 提供统一、实时、所见即所得的资产数据库视图

创建目录时不要有中文

代码项目结构：
├─ Config
├─ Source
    ├─ 项目模块名
        ├─ 模块.Build.cs 当前模块需要引入的其他模块
        ├─ C++代码...
    └─ 项目xx.Target.cs 这里的xx一般可以没有，或为Game、Targe、Client、Server、Editor等等，包括关于项目的编译、链接和构建设置的配置信息。当前构建目标是什么（客户端还是服务器，或者是游戏），构建过程中需要依赖什么模块。
└─ 项目名.uproject

项目模块名目前还只是项目名称，如果项目包括多个模块，这里就可以写有其他名字。

游戏世界编辑器的典型功能

- 世界关卡的创建和分层
- 快速迭代
- 安装与对齐辅助工具
- 可视化游戏世界
- 导航

- Volume
- 选取
- 光源
- 属性设置

先介绍了界面，这么多看了也记不到，全是常用功能，多用自然熟练

快捷键

比较容易忘又常用的：`End：中心贴地，Alt+End：轴心贴地，Shift+End：碰撞盒贴地`

UE把世界编辑器放在游戏子系统上，所以可以PIE（Play in Editor）

项目结构

- Content目录下文件结构与Windows资源管理器下文件结构对应
- UE资源为`*.uassset`文件
- 非UE资源（非`.uasset`文件）建议单独目录存放（不要放到Content目录下）

资源迁移

- 使用资源迁移工具保证内部资源之间的相互引用关系不丢失
- 资源必须在Content目录下

> 跨工程迁移资产，第一种方法按Content目录下原样复制，第二种方法就是资源迁移工具，之间从当前工程将资产复制到其他工程中

DDC

工程目录下有`DerivedDataCache`文件夹

- Separate source and derived data 分离源数据和派生数据
  - Compressed textures 压缩纹理
  - Shader code 着色器代码
  - Optimized meshes 优化网格
- Different representation per platform 针对不同平台不同表示
  - When engine needs derived data for platform 当引擎需要针对特定平台的派生数据
  - Key= asset + target platform, Request key from DDc Server
- Many advantages 许多优势
  - Doesn't bloat `.uasset` size 不会导致`.uasset`文件膨胀
  - Change processing without touching `.uasset` 可以更改处理过程而不修改`.uasset`文件

命名规范

thejinchao/ue5-style-guide: 让Unreal Engine工程更加规范

Allar/ue5-style-guide: An attempt to make Unreal Engine 4 projects more consistent

以上文档之前均为UE4版本，之后更新为UE5版本，另外推荐Epic的风格指南：Epic C++ Coding Standard for Unreal Engine | Unreal Engine 5.5 Documentation | Epic Developer Community | Epic Developer Community

任意代码风格选一个，有良好并且统一的代码规范

项目结构

项目结构自上而下是：Project>Levels>Sub-Levels>Actors>Component

Project 项目

项目(Project) 是一个自成体系的单元，保存所有组成单独游戏的所有内容和代码，并与硬盘上的一组目录相一致。

项目文件: `.uproject`

World 世界

世界场景(World) 中包含载入的关卡列表。它可处理关卡流送和动态Actor的生成(创建)。

> 一个World可以有多个Level，一般整个游戏运行时只有一个World，一个`world`包含了游戏中所有的资源、场景、对象和逻辑，关卡切换通过切换Level完成。虚幻引擎也支持多个World。

Level 关卡

level (关卡)是用户定义的游戏区域，

我们主要通过放置、变换及编辑Actor的属性来创建、查看及修改关卡，

在虚幻编辑器中，每个关卡都被保存为单独的.umap文件，也被称为"地图"。

> `Level` 是一个比 `World` 更小的单元，代表游戏中的一个具体区域或场景。每个 `Level` 包含了该区域内的所有游戏元素，如地形、建筑、NPC、物品、光源等。可以将 `Level` 看作是组成 `World` 的模块。
>
> 在一个关卡式游戏中，第一关可能是一个 `Level`，第二关是另一个 `Level`。在大多数情况下，每个 `Level` 会有独立的设计和表现，但它们都构成了一个大的游戏世界。

Sub-Levels

What is an Sub-Levels?

A small area of the whole Map 整个地图的一小部分

Decouple the work between artist and designer 设计师和艺术家的工作脱钩

> `Sub-Level` 是 `Level` 内部的子级场景，通常用于进一步划分和管理一个大 `Level`。
>
> 允许开发者将一个大关卡拆分成多个更小的部分，分别进行加载和处理。通过这种方式，游戏可以在运行时动态加载和卸载不同的 `Sub-Level`，优化性能并减少内存占用。
>
> 这里举例是和平精英（大战场大逃杀游戏），一般不可能一次性加载所有地图区块，将地图区块分为不同的Sub-Level，同时也可以用Sub-Level用来组织和管理复杂的场景。

Actors

可放入关卡中的对象都是 `Actor`

`Actor`是支持三维转换(如平移、旋转和缩放)的泛型类，通常包括一个或者多个`Actor Components`

可通过游戏进程代码(C++或蓝图)创建及销毁`Actor`

在C++中，`AActor`是所有`Actor`的基本类

Things to know about Actors

- Don't have Location, Rotation (stored in root component)
- Created with `SpawnActor()` method
- Must be destroyed explicitly with `Destroy()` method

Component 组件

组件(Component) 是可添加到Actor的一项功能。

组件不可独立存在，但在将其添加到Actor后，该Actor便可以访问并可以使用该组件所提供的功能。

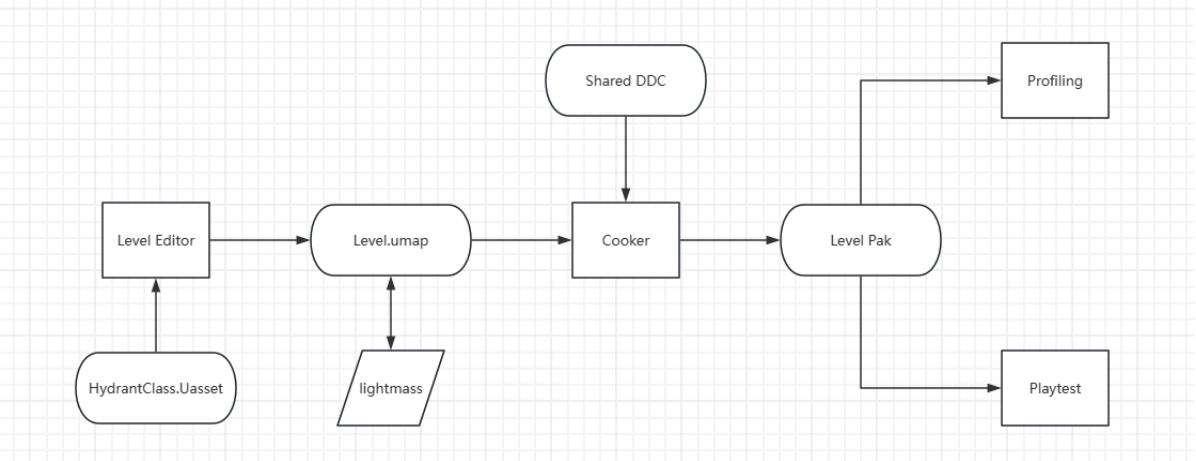
What is an ActorComponent?

- Reusable functionality that can be added to an Actor
- Contain the most interesting functionality & events

Example Components:

- Scene Component - Adds transforms and attachments
- Primitive Component - Adds collision and rendering
    - Hit - Called when bumping into a wall
    - Begin/EndOverlap - Walk into or out of a trigger
    - "Begin/EndCursorOver
    - Clicked/Released

- InputTouchBegin/End
- Begin/EndTouchOver



配置Android Studio与安卓打包

源码构建

[下载虚幻引擎源代码 | 虚幻引擎 5.5 文档 | Epic Developer Community | Epic Developer Community]

[GitHub上的虚幻引擎 - Unreal Engine]

首先在EpicGames上绑定Github账号，然后加入EpicGames组织，之后才能进入Unreal Engine的GitHub界面Git

Git获取到Unreal Engine源码后

编辑器路径`Engine\BinarieslWin64\UE4Editor.exe(UnrealEditor.exe)`

- 运行Setup.bat，下载依赖文件

> 下载依赖文件服务器在国外，所以下载依赖文件可能需要特殊上网方式加速

- 运行GenerateProjectFiles.bat生成工程文件
- UE4.sln -VS2019(UE5.sln -VS2022)

生成项目文件

- 运行`Engine\Binaries\Win64\UnrealVersionSelector-Win64-Shipping.exe`注册引擎
- 工程文件右键菜单中生成项目*.sln文件

编译通过Unreal Build Tool (UBT)完成

- 调用Unreal Header Tools(UHT)预处理头文件，解析头文件中引擎相关类元数据，产生胶水代码
- 调用特定平台普通C++编译器,正式开始编译

工程目录结构

最外层目录

- /Engine - All code, content & configuration for the Engine
- /Samples - StarterContent
- /Templates - Templates for creating new projects

/Engine 和 /MyProject 内部目录

- /Binaries - Executables & DLLs for the Engine
- /Build - Files needed for building the Engine
- /Config - Configuration files
- /Content - Shared Engine content
- /DerivedDataCache - Cached content data files
- /Intermediate - Temporary build products
- /Plugins - Shared and project specific plug-ins
- /Saved - Autosaves, local configs, screenshots, etc.
- /Source - Source code for all the things!

> 引擎源码目录和项目目录是一致的，可以发现编译时生成了`Binaries`，`Intermediate`，`Build`，`DerivedDataCache`这几个文件夹，所以这几个文件夹是可以删除的，编译时会自动生成（删除了这些文件夹会重新编译）
>
> 另外`Saved`文件夹为编辑器生成，如果没有手动更改过内容也可以删除
>
> GitHub Unreal Engine的.gitignore已经自动配置以上文件不需要pull

编译规则

- 目标文件
  - 支持客户端、服务器、编辑器等
  - 通过C#源文件声明，扩展名为.target.cs，存储在项目 Source 目录下

虚幻引擎构建工具目标参考 | 虚幻引擎 5.5 文档 | Epic Developer Community | Epic Developer Community

- 模块
  - 通过C#源文件声明，扩展名为.build.cs，存储在项目 Source 目录下
  - 属于一个模块的C++源代码与.build.cs文件并列存储

虚幻引擎中的模块属性 | 虚幻引擎 5.5 文档 | Epic Developer Community | Epic Developer Community

Module

Module 类型

- Developer - Code for additional tools (output log, source control, profiling, cooking, derived data cache, ...)
- Editor - Code for implementing the Unreal Editor. Many systems have separate Runtime and Editor modules.
- Runtime - Code needed while the game is running
- ThirdParty - External code from other companies
- Plugins - Extensions for Editor, Games, or both
- Programs - Build and header tools, lightmass light baking, shader compiler, ...

Module 依赖规则

- Runtime modules must not have dependencies to Editor or Developer modules

- Plug-in modules must not have dependencies to other plug-ins

> GameFeature 除外，他一定依赖项目模块

Important Modules for Beginners

- Engine - Game classes & engine framework
- UMG - Unreal Motion Graphics implementation
- Slate - Widget library & high-level Ul features
- SlateCore - Fundamental Ul functionality
- Core - Fundamental core types & functions

Runtime Modules

- Basic functionality for the engine and major subsystems: `*Core`, `Engine*`
- Rendering: `*Render*`, `*RHI*`（Render Hardware Interface 渲染硬件接口）
- Slate Ul: `*Slate*`
- Audio, Video: `*Audio*`, `*Media*`
- Al: `Al*`, `Nav*`

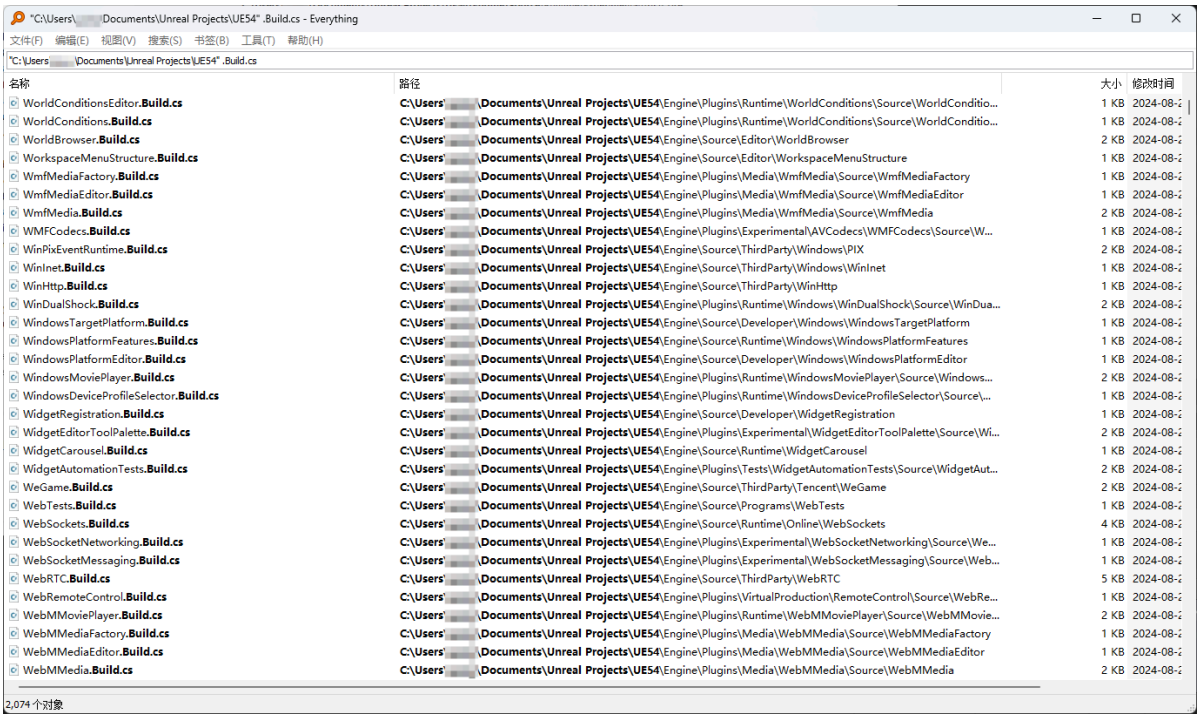
Editor Modules

- Blueprint: `Kismet*`（早期蓝图就叫Kismet）, `*Blueprint*`
- Animation: `Anim*`
- Various independent editor panels: `*Editor`

Developer Modules

- Asset cooking: `*Target*`, `*Format*`
- Mesh tools: `*Mesh*`
- Profiling: `Profile*`
- Shader compiler support: `*Shader*`
- Logging: `*Log*`

这些模块都在虚幻引擎源码下



[Unreal Engine for Unity Developers | Unreal Engine 5.5 Documentation | Epic Developer Community | Epic Developer Community](#)

引擎Loop

虚幻引擎也是从Main函数开始的，并且也有Loop

程序入口main函数 `int32 WINAPI WinMain`

虚幻引擎运行大概分为四个部分

`\Engine\Source\Runtime\Launch\Private\Launch.cpp`文件

```C++
int32 GuardedMain( const TCHAR* CmdLine )
{
...
    // 创建一个对象，在超过作用域时调用析构函数
    struct EngineLoopCleanupGuard
    {
       ~EngineLoopCleanupGuard()
       {
          if (!GUELibraryOverrideSettings.bIsEmbedded)
          {
             EngineExit();
          }
       }
    } CleanupGuard;
...
    // 引擎预初始化
    int32 ErrorLevel = EnginePreInit( CmdLine );
...
       if (GIsEditor)
       {
            // 编辑器初始化
          ErrorLevel = EditorInit(GEngineLoop);
       }
```

```
        else
        {
            // 引擎初始化
           ErrorLevel = EngineInit();
        }
...
            // 引擎帧更新
           EngineTick();
...
         // 编辑器退出
        EditorExit();
...
}
```

Tick会调用

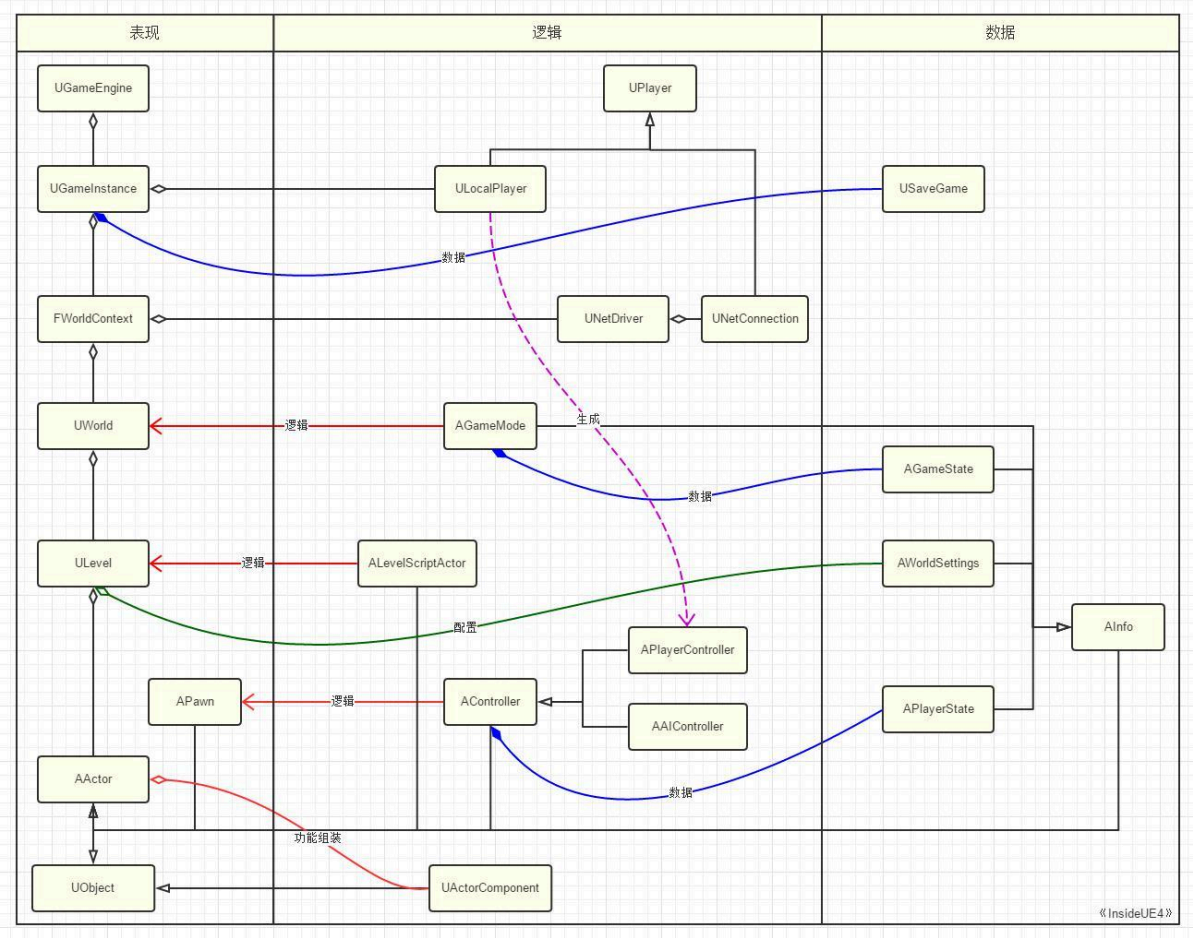
```
LAUNCH_API void EngineTick( void )
{
    GEngineLoop.Tick();
}
```

进入`\Engine\Source\Runtime\Launch\Private\LaunchEngineLoop.cpp`文件执行Tick逻辑

```
void FEngineLoop::Tick()
{
    ...
}
```

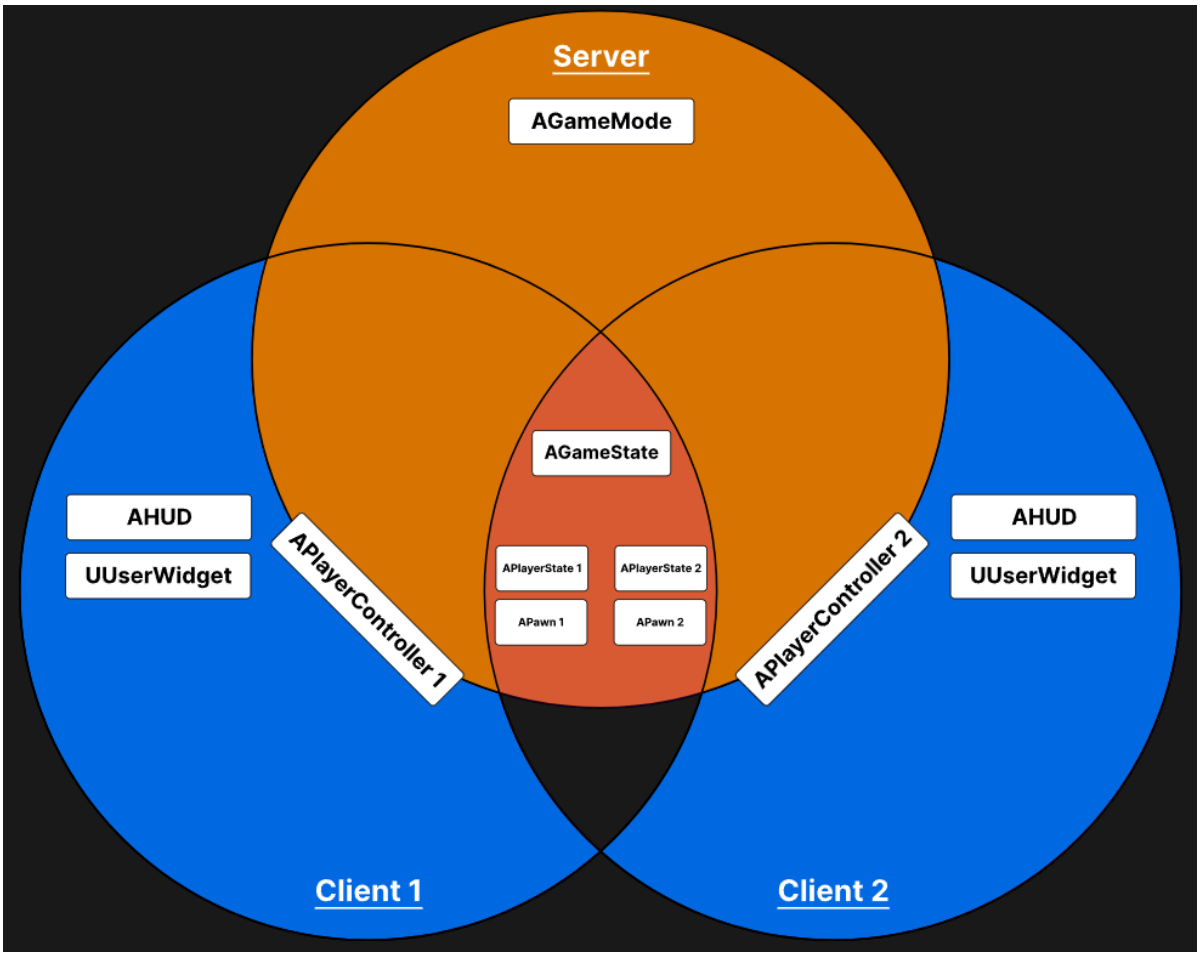
UE编程技巧

游戏框架



这里简单介绍了一部分类的功能

> 个人建议以下内容，不用记忆，写一些虚幻引擎Gameplay工程或者阅读虚幻引擎Gameplay的源码，均能熟练掌握以下内容。
>
> 虽然但是也都需要熟练掌握，网络同步时以下内容在服务器，客户端上同步方式是不一样的。



《InsideUE4》GamePlay架构（一）Actor和Component - 知乎

GameMode游戏模式

游戏模式(GameMode)类负责设置正在执行的游戏的规则。

规则可包括玩家如何加入游戏、是否可暂停游戏、关卡过渡，以及任何特定的游戏行为(例如获胜条件)

Things to know about GameModes

- Only exists on the server and single player instances!
- GameState is used to replicate game state to clients
- Default game mode can be set in Project Settings
- Per-map overrides in World Settings

GameState 游戏状态

游戏状态(GameState) 包含要复制到游戏中的每个客户端的信息，它表示整个游戏的"游戏状态"。

它通常包含有关游戏分数、比赛是否已开始和基于世界场景玩家人数要生成的AI数量的信息等等

PlayerState 玩家状态

玩家状态(PlayerState) 是游戏玩家的状态，非玩家AI将不会拥有玩家状态，

在玩家状态中的数据包括玩家姓名或得分、当前等级或生命值等，

Pawn 和 Character

Pawn 是Actor的一个子类，充当游戏中的生命体。

角色(Character) 是Pawn Actor的子类，用作玩家角色。

角色子类包括碰撞设置、双足运动的输入绑定，以及由玩家控制的运动附加代码。

What is a Pawn?

- An agent in the world
- Optionally possessed by a Controller

Things to know about Pawns

- Good place to implement health
- No movement or input code by default

What is a Character?

- Special Pawn that can walk
- Comes with useful Components

Things to know about Character

- Handles collision
- Client-side movement prediction (UCharacterMovementComponent)

> Character特指两足能走的动物……

Player Controller

玩家控制器(PlayerController) 类用于在游戏中获取玩家输入并将其转换为交互

玩家控制器通常拥有一个Pawn或角色作为游戏中玩家的代表

What is a Controller?

- A brain that can possess a Pawn
- PlayerController:
    - Represents a human player
- AIController:
    - Computes AI behavior for Pawns

Things to know about Controllers

- Possess one Pawn at a time
- Can persist after possessed Pawn dies

PlayerController

- Interface for players to agents
- Handles touches, clicks, keyboar
- Showing/hiding mouse cursor
- Good place for non-agent code

- Menus, voice chat, etc.
- Many other useful options

Blueprint 与 Lua

Blueprint是什么

[蓝图可视化脚本 | 虚幻引擎 5.4 文档 | Epic Developer Community | Epic Developer Community]

The Blueprints **Visual Scripting** system in Unreal Engine is a complete **gameplay** scripting system based on the concept of using **a node-based interface** to create gameplay elements from within Unreal Editor. As with many common scripting languages, it is used to define **object-oriented (OO)** classes or objects in the engine. As you use UE4, you'l often find that objects defined using Blueprint are colloguially referred to as just

特殊的asset, 在编译后会生成对应字节码

引擎运行时会读取字节码，并交由蓝图虚拟机动态解释执行

[深入蓝图编译和运行（一）猫都能看懂，以实际例子解析蓝图函数编译运行原理 - 知乎]

Blueprint分类

- Level Blueprint
  - 自动创建，每个Level-个
  - 生命周期是整个关卡存在的时间
  - 监听Level级别的事件
- Blueprint Class
  - 自己可以随意添加，需要继承已有的类
  - 可以添加不同的组件来丰富功能
  - 通常被放置在关卡中，执行自身的逻辑功能
- Data-Only Blueprint
- Blueprint Interface
- Blueprint Macros

Blueprint的使用

- 从监听的事件开始
- 其他Actor接口调用
  - 从关卡中拖进来Actor，调用其接口
    - 右键选择系统提供的默认Actor
- 调用蓝图库中的函数
- Event
  - BeginOverlap
  - EndOverlap
  - Hit
  - BeginPlay

- EndPlay
- Destoryed
- Tick
- Custom
- Function
- Variables

Blueprint与C++之间的成员变量

UE5标识符详解 | 史上最全 - 知乎

```
// UPROPERTY是虚幻引擎标记和描述成员变量的宏，是反射系统的一部分
// 使用UPROPERTY标记的变量将受编辑器支持，可以序列化，在反射系统中可见，支持垃圾回收，支持网络同步
// 使用标识符EditAnywhere，使得以下变量在虚幻引擎编辑器中可以编辑，但是只能在编辑器编辑值，不能在蓝图中使用，详情看以上知乎文章
UPROPERTY(Category=MyActor, EditAnywhere)
uint32 bUseOffset :1;
UPROPERTY(Category = MyActor, EditAnywhere, meta =(EditCondition ="bUseOfset"'')
FVector Offset:
```

Blueprint与C++之间的相互函数调用

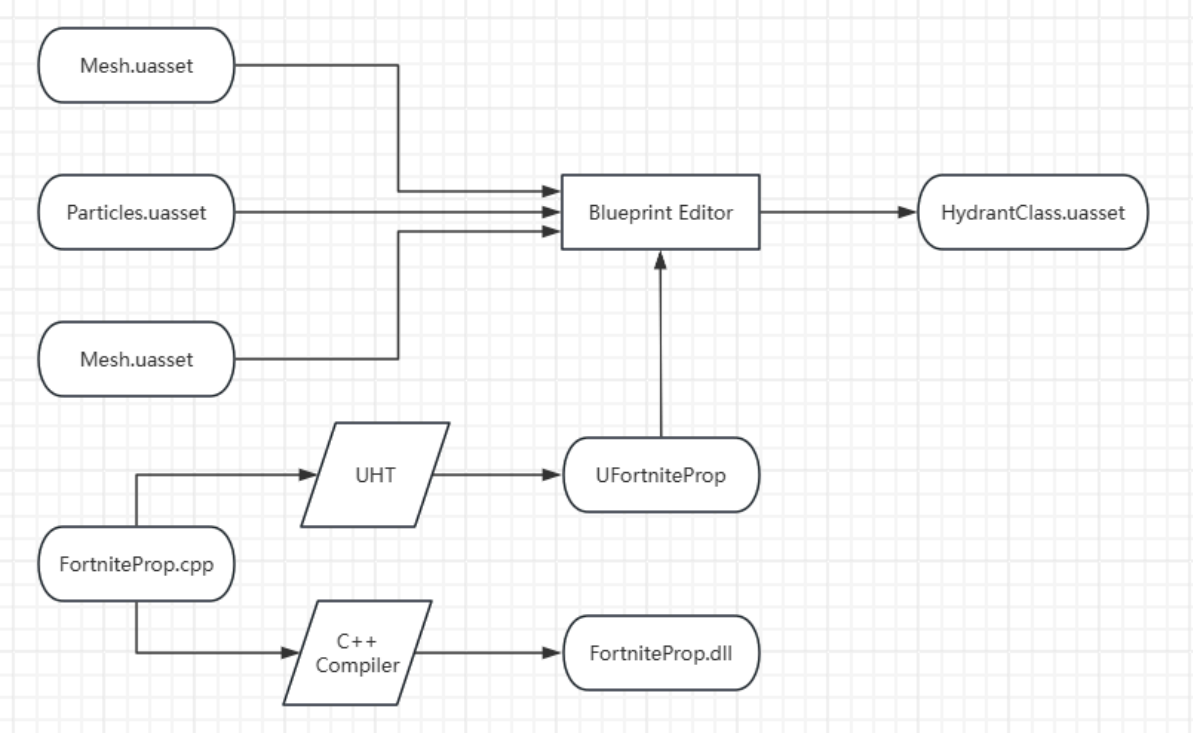
```
UFUNCTION(BlueprintCallable)
UFUNCTION(BlueprintlmplementableEvent)
UFUNCTION(BlueprintNativeEvent)
```

- BlueprintCallable: C++中实现函数体，蓝图调用。
- BlueprintImplementableEvent: 蓝图中实现函数体，C++中声明和调用
- BlueprintNativeEvent: C++中实现默认函数体，蓝图中可以选择重载。
    - C++函数体定义有语法规定，[FunctionName]_Implementation instead of [FunctionName]

这里举例是一个消防栓的，C++写Actor，然后C++的FortniteProp继承，最后蓝图的Hydrant蓝图类继承FortniteProp类，在编辑器中使用。

使用编辑器来创建蓝图类，将不同功能的组件结合起来

暴露可视化变量，函数和事件



> 给一个Lyra的例子，Lyra的C++继承链ALyraCharacter->AModularCharacter->ACharacter
>
> 在C++中编写好大部分逻辑和框架，包括绑定输入等行为
>
> 在蓝图中构建玩法和设置模型等操作
>
> 最后在编辑器中放置蓝图的Character
>
> 在Lyra中C++实现了游戏架构，蓝图编写游戏逻辑。
>
> 这些操作其实均可在蓝图或C++中完成，但蓝图提供了即时反馈和快速迭代，**C++** 和 **蓝图** 的结合，允许程序员和设计师在同一个项目中并行工作。程序员可以专注于游戏的底层架构、性能优化和复杂算法的实现，而设计师则可以通过蓝图实现游戏机制、调整角色行为、设计关卡和交互。

Blueprint的使用

- 缺点
  - 容易形成蜘蛛网
  - 二进制格式，Diff和Merge不方便
- 使用原则:
  - 用于数值配置
  - 用于简单的效果展示
  - 用于特别简单的逻辑(代码不超过屏幕范围)

Lua简介

Lua简介自己查

Lua Plugin for UE

- slua-unreal：https://github.com/Tencent/sluaunreal
- UnLua：https://github.com/Tencent/UnLua

> sLua听说▲粥也在用

> 黑神话程序频slua：[如何评价腾讯开源 Unreal Engine 的 Lua 解决方案 sluaunreal？ - 知乎]

- unreal引擎的插件
- 通过unreal反射能力，导出蓝图接口和静态c++接口给lua语言
- 支持lua到c++双向，lua到蓝图双向调用

c++ In UE

在UE里创建C++类（Rider右键也能创建Unreal类）

> Unreal C++类推荐使用虚幻引擎创建或Rider创建，手写Unreal类除了继承自UObject类，还需要在头文件包含文件的最后一个包含 #include "文件名.generated.h" 这个generated.h是UHT收集并且生成的，包括了编译和反射系统需要的额外代码。

编码规范

Epic C++ Coding Standard for Unreal Engine | Unreal Engine 5.5 Documentation | Epic Developer Community | Epic Developer Community

T - 模板类的前缀,i.e. `TArray`, `TMap`, `TQueue`
U - 继承自 UObject 的类前缀, i.e. `UTexture`
A - 继承自 AActor 的类前缀,i.e. `AGameMode`
F - structs及其他多数类均以F为前缀,i.e. `FName`, `FVector`
I - 抽象接口类前缀,i.e. `ITransaction`
E - 枚举类型的前缀。,i.e. `ESelectionMode`
b - 布尔变量必须以b为前缀,i.e. `bEnabled`

UnrealHeaderTool需要正确的前缀才能正常编译

基本类型

不使用C++原生的整型(char, short,int, long, etc.)
自定义ints & strings in GenericPlatform.h (int32, uint32, uint64, TCHAR, ANSICHAR etc.)

容器

- TArray, TSparseArray - Dynamic arrays
- TLinkedList, TDoubleLinkedList
- TMap - Key-value hash table
- TQueue - Lock free FIFO
- TSet - Unordered set (without duplicates)
- And many more in Core module

智能指针

- TSharedPtr, TSharedRef - for regular C++ objects
- TWeakPtr -for regular C++ objects
- TWeakObiPtr - for UObjects
- TAutoPtr, TScopedPtr
- TUniquePtr

其他常用结构体

- FBox, FColor, FGuid, FVariant, FVector, TBigInt, TRange

UObject

UObjects 增加了标准C++ 的很多功能

- Run-time reflection of class properties and functions
- Serialization from/to disk and over the network
- Garbage collection
- Meta data
- Also: Blueprint integration

Magic Macros

- UCLASS-类
- USTRUCT-结构体
- UFUNCTION-成员函数
- UPROPERTY-成员变量

```
USTRUCT()                       // magic!
struct FVector2D
{
    UPROPERTY()                 // magic!
    float x;

    UPROPERTYO                  // magic!
    float Y;

    UFUNCTION ()                // magic!
    float GetLength() const;
}
```

UObject中包含了一个重要的函数GetWorld()，可以方便获取需要重要信息

UObject还支持对变量进行序列化和反序列化

详情：

《InsideUE4》UObject（一）开篇 - 知乎

局部关闭优化

调试(Debug)	**该配置包含用于调试的符号。该配置在调试配置中同时构建引擎和游戏代码。**
调试游戏(DebugGame)	该配置按最优方式构建引擎，但游戏代码为可调试状态。此配置适用于调试游戏模块。
开发(Development)	该配置启用所有功能，但最费时间的引擎和游戏代码优化除外。从开发和性能角度看，它是最理想的配置。
交付(Shipping)	这是最佳性能配置，用于交付游戏。此配置剥离了控制台命令、统计数据和性

Development模式会去掉许多调试信息，可以通过单个cpp文件或者单个函数关闭优化

```
#ifdef __clang__
#pragma clang optimize off
#else
#praqma optimize("", off)
#endif

#ifdef __clang__
#pragma clang optimize on
#else
#praqma aoptimize("", on)
#endif
```

可以在*.build.cs中按模块关闭优化
OptimizeCode = CodeOptimization.Never;

Android平台真机调试

- 安装Nsight Tegra
  - Engine\Extras\AndroidWorks\Win64\CodeWorksforAndroid-1R7u1-windows.exe
  - 安装时关闭VS及UE4进程
  - 安装后重启系统
- VS 工程编译设置
  - DebugGame
  - Android
  - Nsight Tegra Debugger
- F5启动调试
  - 等待编译完成
  - 等待自动部署APP到手机
- AGDE
  - Debugging Unreal Engine Projects for Android in Visual Studio with the AGDE Plugin | Unreal Engine 5.2 Documentation | Epic Developer Community | Epic Developer Community
  - 适用于 Visual Studio 的 Android Game Development Extension | Android game development | Android Developers

引擎工具了解及学习资料

日志

```
UE_LOG(LogGameSettings,Log,TEXT("[Render] set shadow : [$d]"), bshadow);
```

Log:

- Channel
- Verbosity Level

```
// 日志通道定义方法
DECLARE_LOG_CATEGORY_EXTERN(LogGameSettings, Log, AII); // .h文件声明
DEFINE_LOG_CATEGORY(LogGameSettings) // .cpp文件定义
```

可视化日志

[使用虚幻引擎4可视化记录器]

VisualLogger工具是UE4提供的可视化记录工具。

通过在逻辑代码中通过日志形式收集信息，然后用工具进行绘制显示来进行回放，同时收集到的信息可以序列化为磁盘文件。

内置控制台

Stat FPS (显示帧数)

t.MaxFPS 1000(最高帧率限制到1000)

Stat UNIT(对游戏线程，渲染线程，GPU耗时进行统计，分析瓶颈

Stat GAME (对游戏各个模块的tick耗时进行统计)

Stat SceneRendering (渲染基本信息统计，可以看DrawCall数)

Stat Engine (渲染信息统计，可以看三角形数量)

Stat InitViews(可以看到视口剔除的三角形数量)

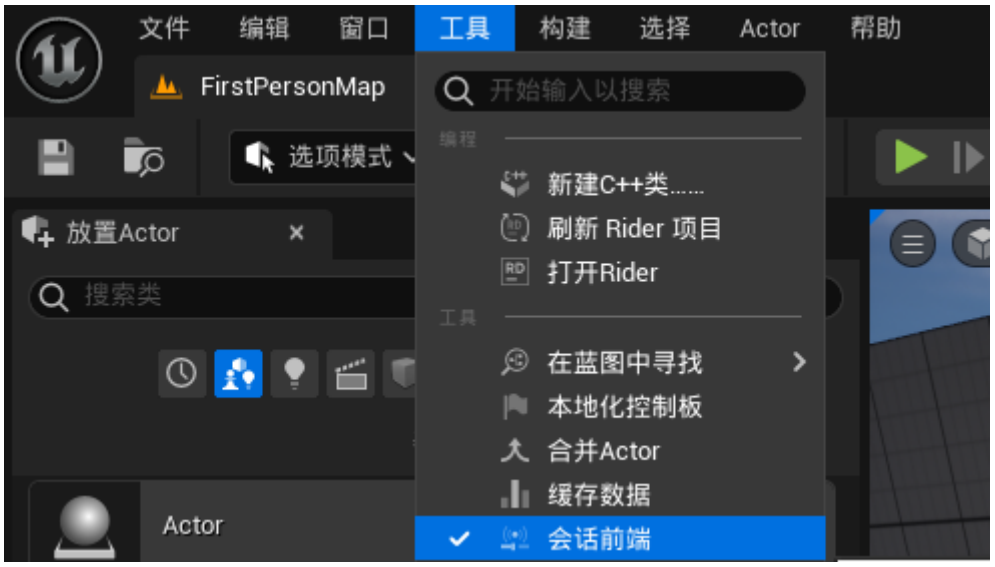
Stat RHI(可以看到所有的DrawCall)

游戏运行时，~键用来打开控制台输入窗口(Mobile 使用四指同时滑屏操作)

Stat StartFile, Stat StopFile

Windows平台:`工程文件/Saved/Profiling/UnrealStats` 文件夹下自动保存性能分析文件

Window->Developer Tools->Session FrontEnd->Profiler->Load打开分析文件



> 虚幻5换地方了，在工具->会话前端，其实UE5这个功能被隐藏了，不能使用，转向Unreal Insights
>
> [UE 5 Unreal Frontend Profiler removed? - Platform & Builds / Debugging, Optimization, & Profiling - Epic Developer Community Forums]

r.ScreenPercentage 50(低分辨率渲染后缩放)
grass.Enable
ShowFlag.*
    ShowFlag.Landscape
    ShowFlag.Staticmeshes
    ShowFlag.DynamicShadows
    ShowFlag.Particles
    ShowFlag.PostProcessing
MemReport -full
ViewActor #name#(视角转为指定pawn视角，并且将其设为默认角色)
ShowDebug Animation(显示当前角色的动画状态调用)
ShowDebug Al ()

调试相机

输入命令freezerendering可以冻结当前帧的渲染
输入命令toggledebugcamera可以自由移动相机
show bounds可以显示没有被culling掉的物体的包围盒

-game

以游戏模式启动和调试

- 从工程文件中右键选择Launch game可以启动游戏
- 启动游戏实际上是在启动编辑器时加个-game参数
- 在Debug启动项中加入-game参数，可以调试游戏

```
"./Win64/UnrealEditor.exe" "E:\ShooterGame.uproiect" /game/map/mymap -game
"..\Win64\UnrealEditor.exe" "E:\shooterGame.uproject" /Maps/UEDPCDesert_1_1_Root
-game -PIEVIACONSOLE -ResX=1280 -ResY=720 -Multiprocess -messaging -
SessionName="Play in Standalone Game"
".\Win64\UnrealEditor.exe" e:\ShadowTrackerExtra.uproiect /game/Maps/SurviveRoot
-game -featureleveles3 -faketouches
```

GPU Profile

Ctrl + Shift+,在编辑器模式下截取GPU快照，并可视化显示

RenderDoc

使用RenderDoc分析虚幻引擎画面 | 虚幻引擎 5.5 文档 | Epic Developer Community | Epic Developer Community

RenderDoc是一个独立图形调试工具，已经内置于UE，可以对游戏进行单帧捕获和详细分析。

PC平台:

- Visual Studio
- Nsight
- GPA

Mobile:

- Xcode Instrument
- Snapdragon Profiler
- MGD
- adb
- Unreal Insight

编译UE引擎源码

前面已经写过了，这里不再重复

打包安卓包，下载AndroidStudio并安装相应SDK，在项目设置设置好后即可打包，详情查看相应文章