游戏引擎架构理解

岳苡萱 2020214430

游戏引擎通常由工具套件、运行时组件两部分组成。游戏引擎以软件层构建，上层依赖下层，下层不依赖上层。（软件中应尽量避免循环依赖）

本作业结合Unity，对课程内容、《游戏引擎架构第二版》书中内容学习、理解、提炼。

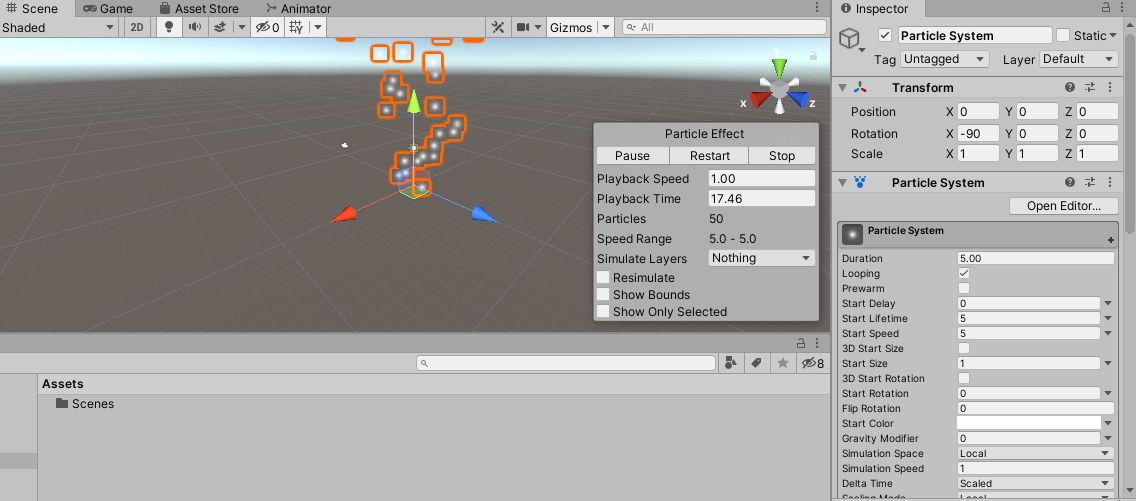
1. 游戏引擎运行时架构
2. 游戏专用子系统
3. 游戏相机：固定型、跟随型。
4. 专用渲染：地形、流体模拟渲染。
5. 玩家机制：移动、控制、状态机&动画。
6. 人工智能：目标决策、动作、视线追踪&感知、寻路
7. 其他：武器、道具、载具、技能等。
8. 前端等主要引擎模块
9. 引擎渲染：流水线式、分层架构。
10. 低阶渲染器：高速渲染几何图元，包含图形设备接口、shader、光照、相机、字体、几何图元、骨骼网格渲染等。低阶渲染器在每个视区中将摄像机结合至世界坐标系；使用材质系统、动态光照系统管理图形硬件的状态和各种着色器。

·图形设备接口：使用图形SDK，初始化设备、建立渲染表面等。

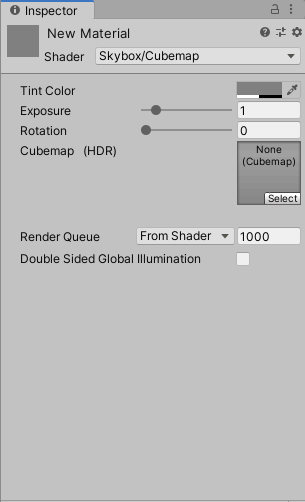
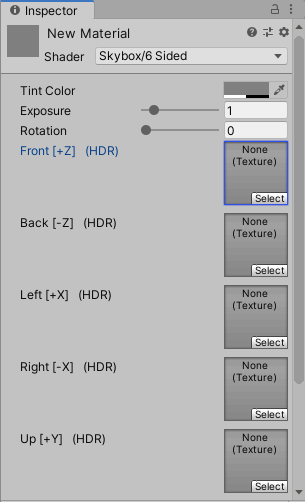
·其他渲染器组件：收集并渲染网格、线表、点表、粒子、地形块等几何图元。

1. 场景图/剔除优化：空间细分（BSP Tree、quadtree、octree等）、遮挡及潜在可见集、LOD系统等。
2. 视觉效果：粒子系统（烟雾、火焰、水等）、贴花系统（弹孔、脚印等）、光照贴图&环境贴图、动态阴影。
3. post-processing：HDR、tone mapping、bloom、FSAA、色彩校正&色彩偏移等。

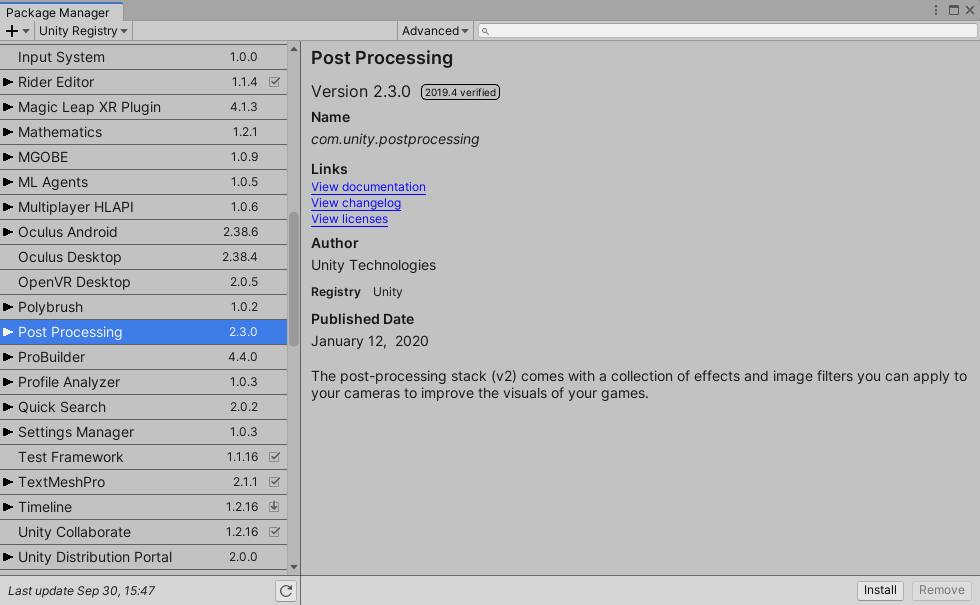
粒子系统、贴花系统常有独立组件，并作为低阶渲染器的输入端；光照及环境贴图、阴影等则常在引擎内部处理；post-processing既可以在渲染器内实现，也可以在独立组件（渲染器输出）内实现。



Unity 粒子系统组件

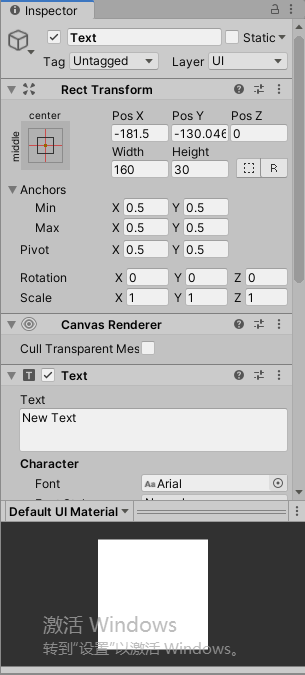
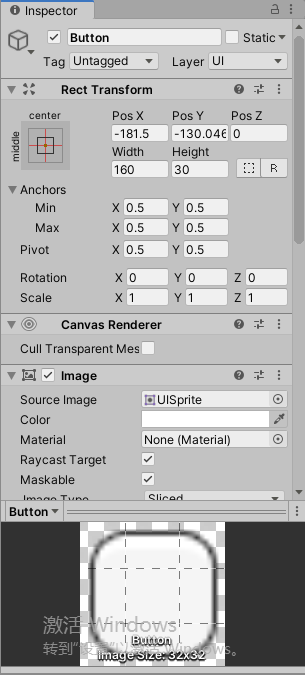
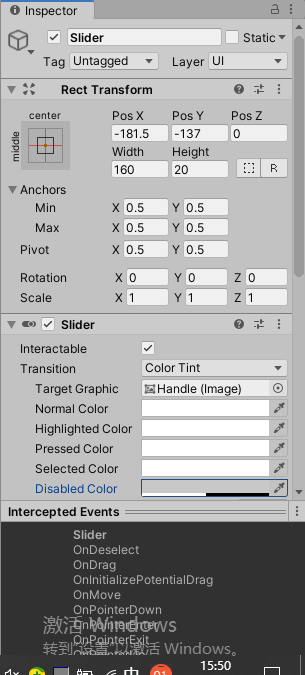


Unity 环境贴图（6面体、Cubemap）

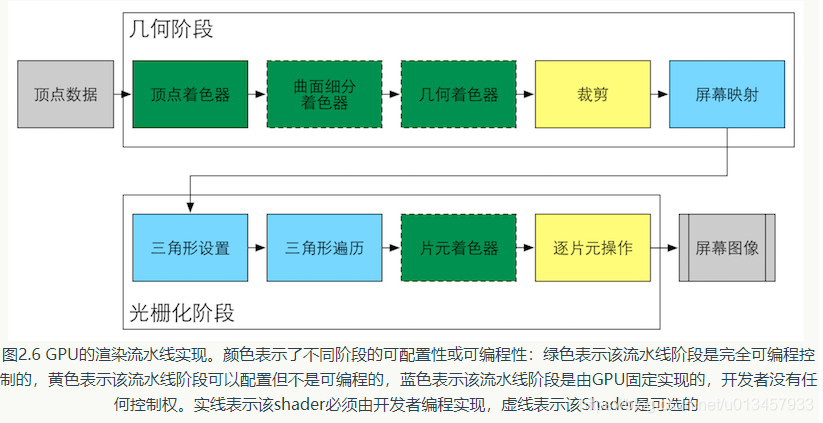


Unity 后处理组件包（适用于shader苦手）

1. 前端、用户界面：平视显示器（HUD）、GUI、游戏内置菜单&主控台&其他开发工具、全动视频（FMV）、游戏内置电影（IGC）等。



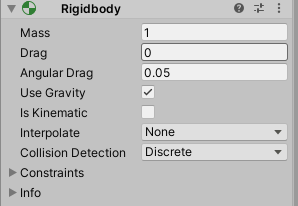
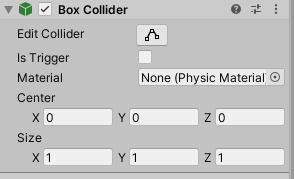
Unity 部分UI控件



GPU渲染流水线

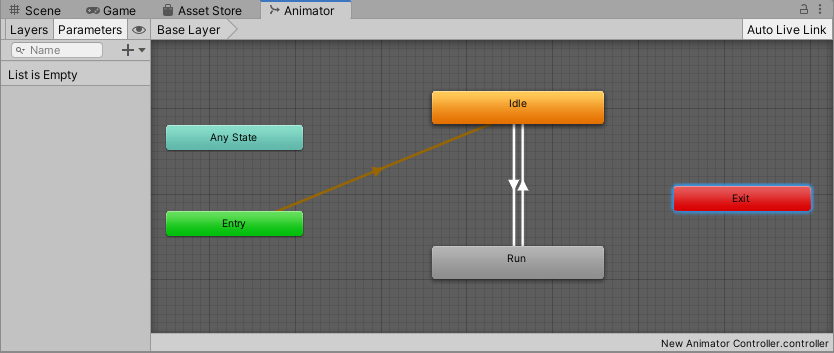
（上图：绿色为可编程控制，黄色为可配置不可编程，蓝色为GPU固定实现即开发者无权限。实现表示shader必须由开发者编程实现，虚线表示shader可选。）

1. 碰撞和物理：真实或半真实动力学模拟。没有碰撞检测，物体会彼此穿透失真。通常使用物理SDK（Havok、PhysX等）。



碰撞包围盒、刚体组件

1. 动画系统：精灵/纹理动画、刚体层次结构动画、顶点动画、骨骼动画（含状态树&层、IK、混合、播放、动画压缩等）、目标变形动画。



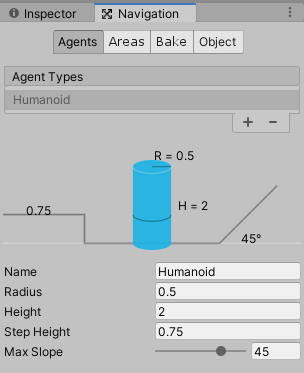
Animator Controller

1. 人体学接口设备：处理玩家输入（控制器底层细节与游戏操作解耦）。包含键鼠、手柄、专用控制器&传感器、触摸屏等。
2. 音频系统：包含DSP/效果、三维音频模型、音频播放/管理等子系统。



Unity声源组件

1. 网络系统：单屏多人、切割屏多人、网络多人、大型多人在线游戏（MMO）。在线多人网络子系统包括：比赛&游戏管理、对象管辖权策略、游戏状态复制。
2. 游戏性基础系统：游戏性指游戏内进行的活动、支配游戏虚拟世界的规则、玩家角色的能力（玩家机制）、其他角色和对象的能力、玩家的长短期目标。
3. 游戏对象：静态场景物体、动态刚体、PC、NPC、武器、抛射物、载具、光源、摄像机。
4. 事件与消息系统：游戏对象间通信。
5. 脚本系统
6. AI：导航网格生成、寻路、物体躲避、弱点识别等。



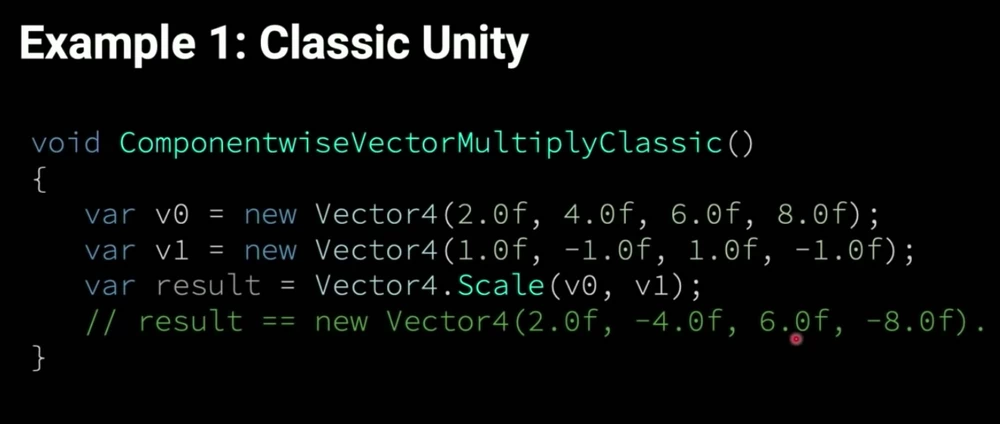
寻路Mesh组件

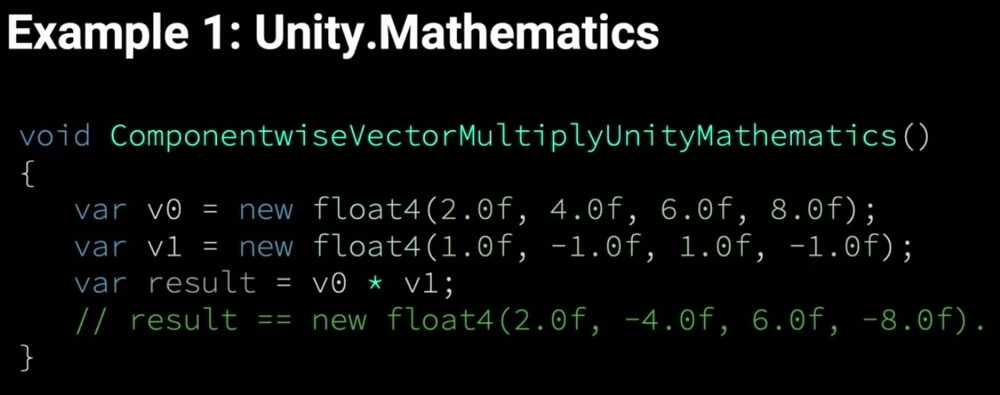
1. 高层次游戏流程系统：状态机、行为树等。
2. 核心系统、资源管理器
3. 核心系统：实用软件、基础功能层，常见功能有：

(1)断言、调试、打印日志：插入代码中捕捉逻辑错误等。

(2)数学库：每个游戏引擎都有一个或多个数学库，提供矢量、矩阵、四元数旋转、直线/光线/球体/平截头体等几何操作、样条线操作、数值积分、解方程组等功能。Unity推出新数学库，具有写法简洁、计算效率高、支持SIMD等特点。如下为旧版-新版数学库运算对比：

两个Vector4相乘：





新旧数学库对比

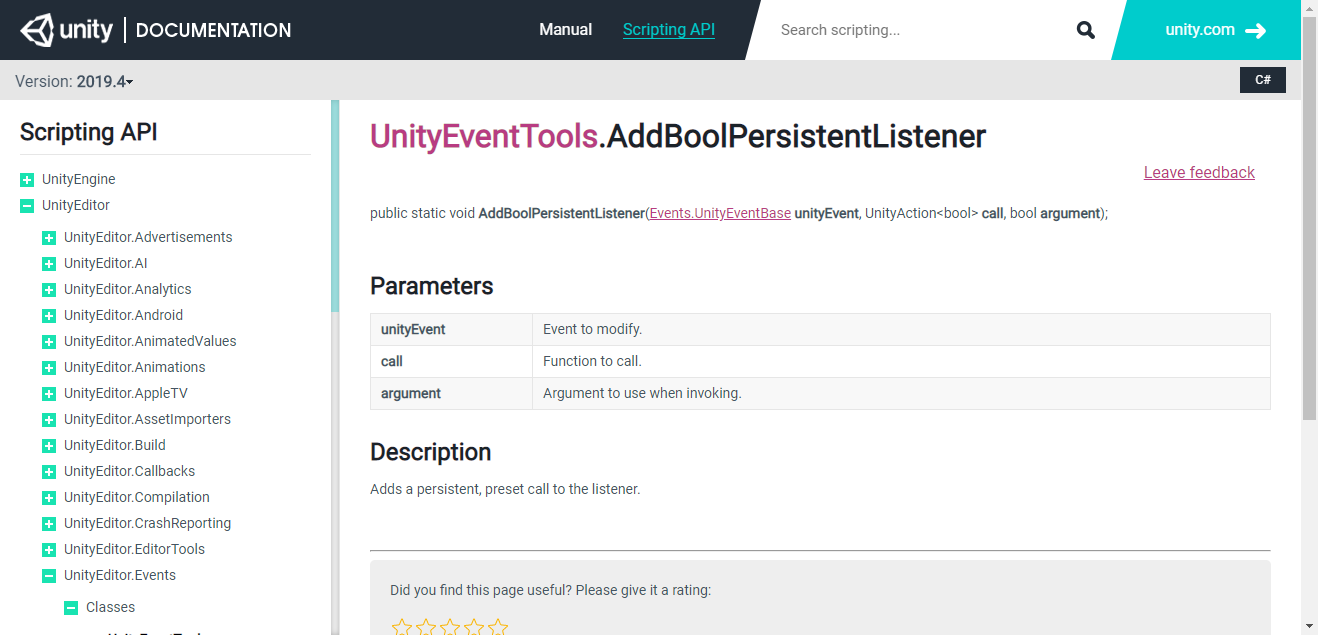
新版数学库可以直接相乘，并且它们是SIMD一条CPU指令同时计算4个结果出来，效率高。

(3)自定义数据结构、算法：引擎需提供一组工具管理基础DS及算法（搜索、排序等）。

1. 资源管理器：提供一个/一组统一接口，管理游戏资源（三维模型、纹理、材质、骨骼、碰撞、物理参数、地图等）及生命周期。Unity支持3种资源读取方式:

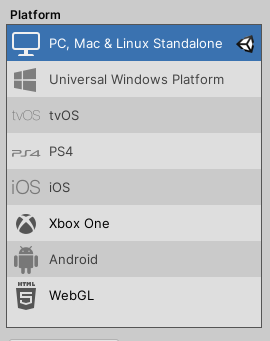
|  |  |
| --- | --- |
| Resources | 支持编辑器模式、发布模式 |
| AssetDatabase | 支持编辑器模式 |
| AssetBundle | 支持编辑器模式、发布模式 |

1. 平台独立层、第三方软件开发包
2. 平台独立层：在硬件、驱动、OS、第三方软件之上，将其余引擎部分与大部分底层隔离。包含平台检测、网络传输层、线程库、图形包类、物理碰撞包类等模块。包装常用标准C语言库、操作系统调用及基础API，确保API在所有硬件平台上都兼容。Unity官方API库示例：



Untiy官方API库

1. SDK、中间件：SDK提供基于函数或类的接口，常见的SDK有：DS/算法（Boost++）、图形（OpenGL、DirectX）、物理碰撞（Havok）、角色动画（Granny）、AI（Kynapse）、生物力学角色模型（Endorphin、Euphoria套件）。如Unity默认用DirectX图形渲染，但可在属性-目标中强行更改为opengl。
2. 系统、驱动与硬件层（底层）
   1. 目标硬件：用来执行游戏的计算机系统或游戏主机。平台包括PC、IPhone/Android智能机和平板电脑、PS/NDS等主机。Unity build出的游戏可执行程序可包含如下平台：



可build平台

* 1. 设备驱动程序：负责管理硬件资源、隔离操作系统及上层引擎，使上层软件不受限于硬件差异。
  2. 操作系统

(1)PC：协调多个执行程序共享硬件（抢占式多任务），游戏程序作为其中一个程序被执行。

(2)游戏主机：一个轻量级库，链接到游戏的执行文档里。在Xbox 360、PS4等新主机、后继主机中，操作系统可中断游戏执行，接管系统资源、允许玩家暂停游戏进入导航菜单等。在操作系统层面，PC与主机区别在减小。