Unity3D常用组件及分析

- U3D的Component
  - 基本概念
    - 1.Unity3D = GameObject + Components
    - 2.Everything is a component
    - 3.基本(内置) + 扩充
  - 使用
    - 1 编辑器挂载
    - 2 代码控制
      - 增加
      - 删除
      - 查询获取
  - 生命周期
    - Unity程序的任务:override component interfaces
    - Awake(不要使用构造函数)
    - Start
    - Update
    - OnDestroy
    - LifeCycle
- Transform
  - 坐标变换的图形学原理
    - 坐标系
      - 坐标系的作用
      - 坐标系是相对的
      - 左右手系
    - 坐标系间的转换
      - 点 P(xyz)
      - 向量 v(xyz)
      - 矩阵(M)
      - 父子空间
    - 空间运动的基本形式
      - translate
      - rotate
      - scale
  - Transform 组件
    - translate
    - rotation
    - scale
  - 应用
    - local/global
    - Transfrom_xxx
    - InvreseTransfrom_xxx
    - LookAt
    - RotateAround
- Light
  - 类型
    - point light
      - 有位置
      - 球型扩散
      - 电灯泡 有衰减
    - directional light
      - 无位置 有方向
      - 无衰减 太阳
    - spot light
      - 有位置
      - 圆锥形扩散
      - 有衰减
      - 舞台聚光灯
  - 顶点光和像素光
    - 光照计算与法线
    - 逐顶点光照计算和逐法线光照计算
    - 哪些情况一定要像素光？
  - 烘焙与实时
    - 离线渲染和实时渲染
  - 阴影
    - Quality setting 里设置
    - shadow distance
    - shadow type
    - shadow caster
    - shadow receiver
    - shadwo cascade
  - 性能问题
    - 实时光源越少越好,甚至可以不用实时光
    - 利用好离线烘焙
    - spotlight开销较大
    - 实时光源只使用定点光或者限制像素光的数量
    - 谨慎使用实时的阴影
    - 用软影
    - 减少影子距离
    - 调低shadow castcade
    - Cull
- Camera
  - Camera组件
    - 摄像机是可以看到画面的基础
    - Clear Flag
    - Cull
    - 多摄像机与渲染顺序（叠在上面的摄像机一般是只清depth的）
  - 正交摄像机
    - 属性
      （ 视口大小和近远剪裁面 ）
    - 特点 ：无透视
      非真实的
      渲染2D视角
    - 用途 ： UI
  - 透视摄像机
    - 属性
      （ 视椎体 ）
    - 特点 ：透视
      模拟真实的人眼
    - 用途 ：正常3D世界
  - 摄像机空间
    - 定义 ：以摄像机为原点的坐标空间(
      并且位一到一定范围)
    - 范围 0,1
    - 表示 ：在摄像机图像上的位置
    - 获取 ：camera.WorldToViewportPoint
    - 用途 ：获取在视角内的位置 自定义的一些裁剪
  - 渲染到纹理
    - 创建render texture
    - 用途 ：后视镜 小地图。。
    - 多摄制
      开倍大
  - 性能问题
    - 视椎体越小越好
      注意你的远裁剪面
    - 注意摄像机的角度
      FOV这些直接影响到可视物数量的参数
    - 慎用rendertexture
    - 在手机上慎用
      occlusion culling
    - 自己做一些基于摄像机视角的剪裁
    - 利用好culling mask
- Terrain
  - 地形系统简介
    - UNITY提供了一套比较强大的制作野外真实地形的工具
    - terrain在移动设备上的开销较大
    - 地形的主要内容
      - 高度图存储高度信息
      - 地表贴图和区域信息
      - 植物系统
  - 地形的简单制作
    - 拉高地表
    - 拉低地表
    - 平原
    - 绘制地表贴图，最多支持多少通贴图？
    - 树（实体模型）
    - 地表的植物（公告板）
    - 其他设置
    - 可以从外部导入高度图
    - terrain collider
  - 常用的程序接口
    - SampleHeight
    - TerrainData（包含所有的高度图，每层细节图的alpha信息）
- 粒子系统
  - 粒子系统简介与使用
  - 程序中访问和控制
    - 控制播放暂停
    - 查询是否在播放
    - 获取renderer
  - 粒子系统与性能
    - 性能大敌!
    - 发射器的数量（<30）
    - 全屏粒子数量(<200)
    - 不再粒子中使用模型
    - 慎用子发射器
    - 关掉离子的阴影和碰撞
    - 自己实现简单的粒子系统
    - 关于粒子的合批
- 屏幕特效
  - 原理和UNITY中的实现
    - 渲染到纹理后的图像空间像素处理
      - 子主题
    - 全屏逐像素
    - 某些设备可能意味着多次绘制
  - Unity的内置屏幕特效
    - HDR
    - Antialiasing
    - Blur
    - Bloom
    - Glow
    - SSAO
  - 实现自己的屏幕特效
    - OnRenderImage
    - Shader
- Mesh
  - mesh的基本概念
    - 导入（fbx）
    - mesh的构成(顶点和索引)
    - 顶点信息的组成(位置 颜色 法线 切线 uv1 uv2 ...)
    - 顶点数和面数
    - 影响顶点数的因素(光滑组的概念)
  - mesh render
    - mesh
      - bounds
      - vertexCount
      - vertex property
        - 获取顶点属性的开销
        - read/write
        - 子主题
      - material
      - shared 和非shared
      - 动态创建一个mesh
  - skinned mesh renderer
    - mesh
    - material
    - bones
      - 换装
        - 整体模型置换
        - 模型挂接
        - 贴图置换
        - 基于相同骨骼的模型置换
  - unity中的几种renderer
    - mesh filter & mesh renderer
    - skinned mesh renderer
    - particleSystemRenderer
- Physics
  - 碰撞的基本原理
    - 场景的划分
      - 八叉树
    - 线段的相交查询
    - 优化计算与包围盒
    - 产生碰撞的因素
      - 碰撞检测体
      - 包围盒区域(collider)
  - Unity常用的Collider
    - 常见包围盒
      - spherer
      - AABB
      - OBB
      - KDOP
    - Unity 包围盒
      - spherer
      - box
      - capsule
      - character controler
  - Rigid body
    - 刚体的物理概念
    - 重力
    - 它是一个碰撞检测体
  - Charactor Controller
    - 他是一个拥有capsule collider的碰撞检测体
    - 同capsule collider的区别
      - dynamic collider
      - static collider
        - 移动它的代价！
      - charactor collider
        - 不需要刚加rigidbody,刚体有它特殊的物理受力表现，计算量较大
        - 通常用于角色的collider
    - move()/SimpleMove()
      - 检测碰撞的移动距离/速度
  - Tigger
    - 无碰撞的阻挡
    - 可以检测到进入collider的事件
      - OnTriggerEnter/OnTriggerExit
        - 要求进入的物体必须是个碰撞检测体
      - OnCollisionEnter/OnCollisionExit
        - 要求进入的物体必须是个碰撞检测体
    - 用于一些触发器或机关区域的制作
  - Physic Cast
    - 有时我们不需要使用碰撞检测体来判断碰撞，只想计算一些几何体间是否有交叉
    - 常用cast
      - SphereCast/SphereCastAll
        - 球形区域探测
      - RayCast/RayCastAll
        - 射线检测
      - LineCast
        - 线段探测
      - CapsuleCast/CapsuleCastAll
        - 发射胶囊体探测，如探测人性物体是否可行走
    - 在Collider A内无法探测到A！
  - 实际应用（控制你的角色在真实环境中行走）
    - 1.地面有collider
    - 2.角色使用charactor collider
    - 3.角色每个update中中计算位移量
      - 需要获得摄像机坐标系下鼠标移动方向对世界坐标的转换
    - 4.对角色的charactor collider 应用位移量
      - 要保持与地表，因此位移量的y要是够向下
  - 物理系统的性能
    - 物理系统在移动端的性能不佳，可能是cpu上的主要瓶颈
    - 调低Fixed Timestep
    - 只用简单的collider，不用mesh collider ，terrain collider ，particle collider
    - 尽量少用刚体
    - 禁止出现static collider
    - 自己模拟一些简单的物理行为，如重力
    - 只在绝对必要的时候使用physics cast
- Audio&Movie
  - 声音系统简介
    - 内部使用的是Fmod
    - 支持3D声音
    - 支持的音频的压缩
    - 支持的格式 AIFF WAV MP3 OGG XM MOD IT S3M
  - Audio Listenner & Audio Source
    - Listenner是声音的接受者，每个场景只能有一个
    - Source是声音的产生源
    - Audio clip是声音资源
  - 在代码中控制声音
    - AudioSource.Playe/Pause/Stop
    - AudioClip存储音频数据的格式
      - channel
      - frequency
      - short存储的采样数据（short.MinValue - short.MaxValue）
  - Movie
    - PC
      - .mov, .mpg, .mpeg, .mp4, .avi, .asf
      - 编辑器导入时依赖机器安装QuickTime
      - 导入后就是一个动态texture，当texture一样使用（解码器渲染到纹理）
      - MovieTexture.Play
    - IOS/Android
      - 使用Handheld.PlayFullScreenMovie接口
      - 该接口会挂起当前U3D进程，调用操作系统的播放器播放，然后回到游戏
      - 可以使用几种播放结束方式，如点击屏幕
      - 视频文件必须放在StreamingAssets文件下（它在设备上是安装文件夹内的只读区域）
      - 支持的格式
        - ios ： .mov, .mp4, .mpv, .3gp
        - android ： .mp4 .3gp
  - 录音
    - 使用Microphone类的相关接口
    - Microphone.Start/End
- Animation
  - Unity动画系统简介
    - legacy
    - mechanim
    - 导入
      - 外部的文件（fbx obj max maya c4d）
      - 内部的animation view
  - Legacy animation system
    - 标记为genneric
    - 动作集合
    - 代码中控制播放
    - 控制循环
  - Animator
    - Animator系统简介
      - 标记为genetic或者humanoid
      - 制作animator controller
      - animator组件
      - 代码中控制播放
      - 控制循环 速度
    - 动作状态机
      - 每个clip就是一个状态
      - 控制状态的转移
        - 在编辑器中连线
        - 设置迁移变量
        - 在代码中crossfade
    - 动作融合
      - 不同状态之间的融合
      - 不同身体部分之间的融合
        - 解决了不同身体部分使用不同动作状态驱动的问题
        - 定义layer
        - 定义mask
        - 为每个layer做状态机
    - 动作重定向
      - 解决在不同骨骼间复用动作的问题
      - 方案:将不同骨骼映射到unity内置的一个标准任务骨骼
      - 标记为humanrid
      - 配置avatar
      - 复用controller
    - 根骨骼动画
      - 解决美术直接在max里面为根节点制作位移动画,并能同步到游戏里的逻辑位置的问题
      - 如果没有跟骨骼动画?
      - 切换跟骨骼动画/非跟骨骼动画
      - 设置controller
      - 代码中的接口,接管跟骨骼的位移
    - 动作mask
      - 禁用某些骨骼队动作的响应
    - 动作event
      - 动作系统同代码的接口
      - 动作的播放进度于逻辑的的同步,常用于表现技能动作中的特效音效同步
      - 在动作上打点
      - 在代码中写响应函数
    - 动作event的实际使用:表现一个技能效果
      - 需求
        - 技能由动作 特效 音效组成
        - 点按一个按键能够表现这个动作
      - 实现
        - 准备素材
        - 制作controller
        - 打event点
        - 写响应函数,播放音频 特效
        - 按键响应
    - Animator的性能
      - cpu性能瓶颈
        - 状态机
        - 动作融合
        - ik
      - 优化
        - 限制Animator的数量
        - 关闭跟骨骼动画
        - 限制骨骼数(<=30)
        - base on render
        - 控制动作帧数
        - 用generic
        - 最重要的是:尽量用animation!
    - Animation View
      - 在unity中制作animation clip 的方式
      - 编辑器中暴露的一切数值都可以作为动画
      - 位移动画
      - 纹理动画
      - 过场动画
- 寻路系统
  - 寻路的概念和寻路网格
    - 获取空间内两点间的一条可走路径
    - 游戏中常用的寻路算法
      - A* 算法
        - 场景划分为格子
        - 广度优先的遍历
        - 保存当前代价最小路程最短的路径
        - 有时看起来不够自然
      - Navigation Mesh
        - A* 算法的扩展
        - 从格子划分改为不规则的三角形mesh
        - 可扩展到三维
  - navmesh的生成
    - navmesh static
    - walkable/not walkable
  - 物体的寻路
    - navmeshagent
      - 我们通常需要预先获取某位置navmeh的交点
    - 使用navmesh获取路点 程序控制
    - 寻路与碰撞检测
      - 有时阻挡也是一种实现控制物体移动的方式
      - 静态阻挡
        - 完全置navmesh
        - 对主角可以使用chractor依靠阻挡
      - 动态阻挡
        - navmesh + collider + rigidbody(别不了目的地)
        - navmesh obstacle
        - 经验:对于一个游戏的物体移动系统,可以完全使用寻路,不是用阻挡
  - layer & cost
    - 控制使的场景区域行走的难易度,影响路径的选择
    - 动态控制某快区域是否可走
  - navmesh obstacle
    - 解决动态阻挡的问题
    - carve/not carve:是否更新navmesh
  - 实际应用:控制你的角色在真实环境中寻路
    - 需求A:点击屏幕上一点 ,允许物体寻路到交点
    - 需求B:控制第一人称的移动,但是采用寻路而不是碰撞
- 实际应用
  - 需求:利用Unity常用组件实现一个效的客户端demo
    - 1 搭建一个简易的场景
    - 2.场景中有随机移动寻路的怪物
    - 3.控制一个第三人称2.5D视角的主角
    - 4.怪物和主角都可以施放远程技能
    - 5.技能由动作 特效 音效构成
    - 6怪物或主角被技能击中后表现特殊效果
    - 7.主角可以通过鼠标进行寻路
    - 8 主角可以通过按键进行技能释放