7-15 周一

报道日

上午熟悉项目、公司地形，体验了x20产品，与周围同事聊聊天

下午搭建电脑的开发环境，熟悉使用公司电脑，查看网易入职相关注意事项及实习生培养计划，熟悉网易内部app、km、wiki的使用，下载相关电脑app。

7-16 周二

上午看了看工具部分的wiki，看得不是很明白，遂搭建ue环境；学习codemaker。

下午学习ue的相关基础知识，熟悉了编辑器界面、项目建立等等；steam跑测一小时

7-17 周三

上午学习perforce的相关知识。跟着开了个早会。靶场训练了会

下午查看工具Wiki，查看了前几届实习生的实习报告，技能开发等等；xbox跑测一小时；学习ue的资产、关卡知识；晚饭后学习y3相关知识，写了会策划案

7-18 周四

上午steam跑测

git和perforce区别

git

·分布式版本控制系统

·适合管理文本型文件

·提供开源的托管仓库

perforce

·集中式版本控制系统

·适合管理二进制文件

·需要自己搭设服务器

以资源文件为主要工作对象的开发流程。虚幻项目的文件是二进制的格式，不支持并行编辑。 集成性。在虚幻编辑器中，对于资源的工作流程基本上都是遵循了Perforce的流程。 大型文件管理能力。在游戏开发过程中，我们常常会涉及很多体积较大的文件，有些文件甚至以GB为单位。

pyimgui

即时模式（Immediate Mode）和保留模式（Retained Mode）是两种不同的GUI渲染方式。 即时模式是一种基于命令的渲染方式，它的工作方式是在每一帧中重新绘制整个GUI。在即时模式中，每个GUI元素都是一个独立的对象，每一帧都需要重新计算和绘制。即时模式的优点是简单易用，适合于小型项目和快速原型开发。缺点是性能较低，因为每一帧都需要重新计算和绘制整个GUI。 保留模式是一种基于状态的渲染方式，它的工作方式是在GUI元素发生变化时更新状态，然后在每一帧中只绘制发生变化的部分。在保留模式中，GUI元素被组织成一个层次结构，每个元素都有自己的状态。当一个元素的状态发生变化时，只需要更新该元素及其子元素的状态，然后在下一帧中只绘制发生变化的部分。保留模式的优点是性能较高，因为只需要绘制发生变化的部分，而且可以支持复杂的GUI元素和动画效果。缺点是实现较为复杂，需要管理大量的状态信息。

其他

路径最好别用纯字符串，会丢失数据的格式信息，用数组比较好 -fPIC与-fpic都是在编译时加入的选项，用于生成位置无关的代码(Position-Independent-Code)。这两个选项都是可以使代码在加载到内存时使用相对地址，所有对固定地址的访问都通过全局偏移表(GOT)来实现。-fPIC和-fpic最大的区别在于是否对GOT的大小有限制。-fPIC对GOT表大小无限制，所以如果在不确定的情况下，使用-fPIC是更好的选择。 -fPIE与-fpie是等价的。这个选项与-fPIC/-fpic大致相同，不同点在于：-fPIC用于生成动态库，-fPIE用与生成可执行文件。再说得直白一点：-fPIE用来生成位置无关的可执行代码。

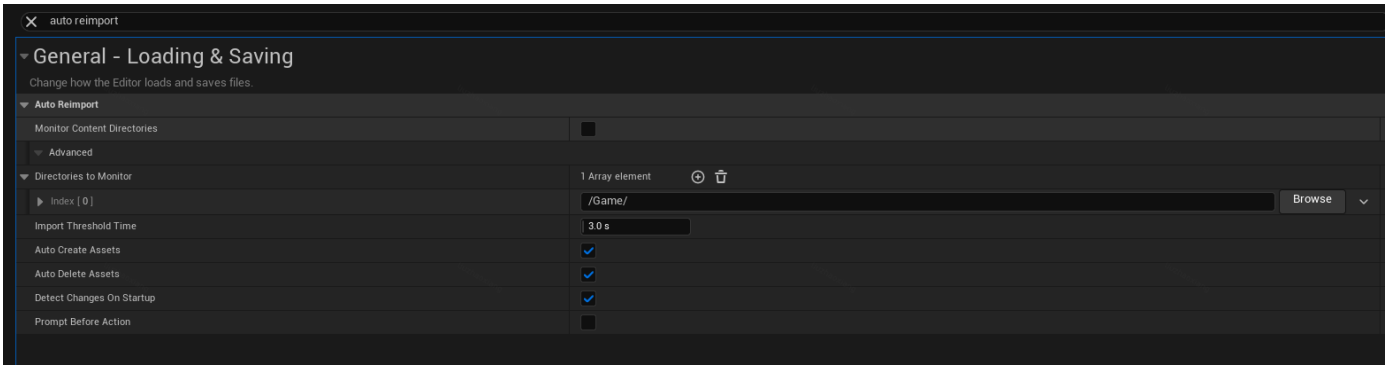
cython enscripten websocket pybind11 cmake pyodide

程序基础认知

引擎/脚本 引擎是游戏逻辑无关的，各种游戏通用的部分，如内存管理，网络同步，渲染等，因为外放后需求相对稳定，运算量大所以一般用C/C++ 脚本是用脚本语言来写游戏逻辑，游戏逻辑需求变化频繁，开发效率高，易热更 一部分游戏逻辑因为运算量大，可能会移到引擎部分实现 重构/迭代 不满足现在的性能和新功能的需求 需求变化

UE

ctrl + b：文件夹定位 ctrl + e：打开物体对应的蓝图 end：贴合地面 alt + 鼠标左键：取消连线 ctrl + p: 找文件 通过使用 监视内容目录（Monitor Content Directories）复选框，可以完全启用和禁用自动重新导入功能。



UImGuiSubsystem::Draw() ->存储该窗口的打开情况，供cpp使用，pyimgui的UImGuiSubsystem::DrawPython() -> PyImGui_Draw(GetWorld())

```python
def imgui_module(self, pyimgui, module_name, world):
    if pyimgui.module_name == module_name:
        if self.windows[module_name]:
            pyimgui.enable = True
            pyimgui.draw(world)
            self.windows[module_name] = pyimgui.enable
            self.ImGuiWindows = self.windows

def PyImgui_Draw(self, world:World):
    self.windows = self.ImGuiWindows
    for module_name in self.windows:
        if self.windows[module_name]:
            self.imgui_module(self.pyimgui_demo, module_name, world) #Template
            self.imgui_module(self.pyimgui_ai, module_name, world) #Template
            self.imgui_module(self.pyimgui_level, module_name, world)
            self.imgui_module(self.pyimgui_wood_pile, module_name, world)
            self.imgui_module(self.pyimgui_player_view, module_name, world)
            self.imgui_module(self.pyimgui_anti_cheat, module_name, world)
            self.imgui_module(self.pyimgui_database, module_name, world)
            self.imgui_module(self.pyimgui_destructible_actor, module_name, world)
```

PyMarvelPlayerController继承自MarvelPlayerController反射于cpp的AMarvelPlayerController 所以PyMarvelPlayerController调用的ParseWizcmd其实是调用AMarvelPlayerController的ParseWizcmd函数 热更python时，修改类的init函数不起作用，需要重启游戏