开发报告

1 开发约定

1.1 环境约定

本游戏项目的开发环境要求如下：

- 游戏引擎：Unreal Engine 4.26.2，蓝图工程
- C++编译环境：Visual Studio 2019 Enterprise
- .Net Framework 4.8
- 版本管理：Git + Git LFS

本游戏项目的运行平台环境与推荐配置如下：

- 目标平台：Windows 10-64bit
- CPU：Intel Core 8400 或以上
- 显卡：NVIDIA 1070 或以上
- 内存：8GB 或以上
- 硬盘：10GB 或以上

仅保证在上述环境下本项目可以正确编译、运行

1.2 开发规范

我们尽量地遵守了以下规则进行开发

- 需要多次被继承的功能抽象为 BPI（蓝图接口）
- 利用蓝图继承分层次开发蓝图
- 尽量不直接出现常数（Magic Number），提升为变量
- 将函数与变量默认设置为 private，防止接口膨胀

2 主要技术

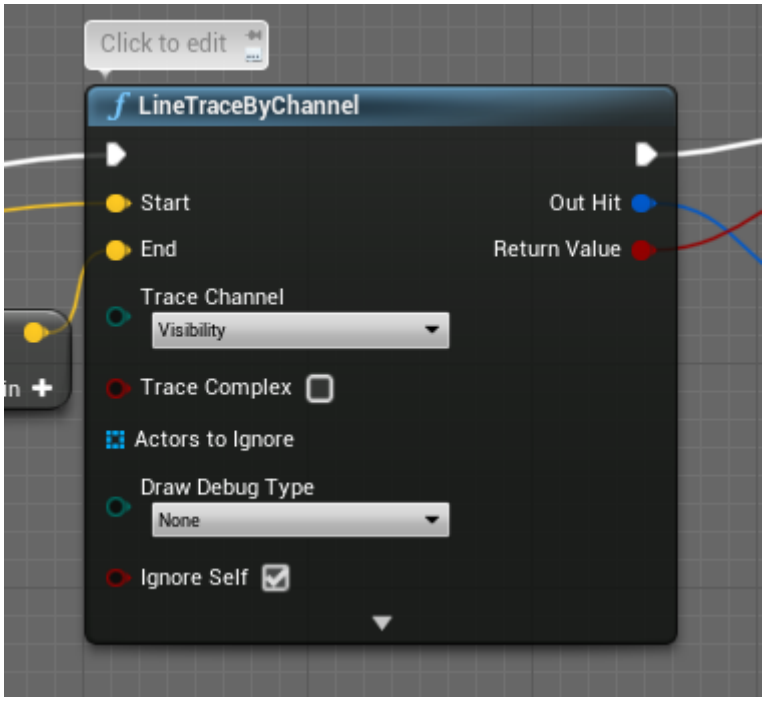
2.1 抽象交互类

需求：

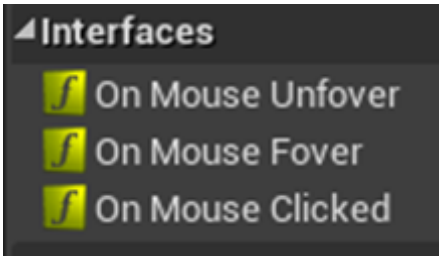
- 具有鼠标悬浮、点击等接口
- 具有相同的父类，在同一个顶层中处理鼠标事件
- 派生出高亮类，悬浮效果被实现为轮廓勾边闪光

实现方式：

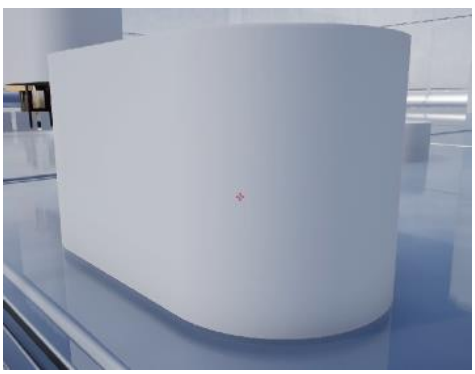
在 PostEffect 处理中，从 Camera 发出一条射线进行物体击中计算，Camera 坐标系下 Z 值最小的物体被选中，关键函数为：

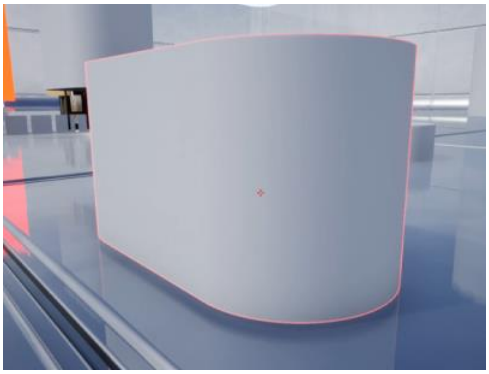


选中之后触发悬浮、离开悬浮、点击等特效，封装为 BPI 接口：



定义一个没有物体的 BP_Actor_Pickupable 类，之后的实际物体 BP 类只需要继承该类即可使用上述接口进行选中操作。例如，下图为选中物体出现红框高亮描边特效（该实现将在 2.2 节中叙述）：





2.2 轮廓勾边

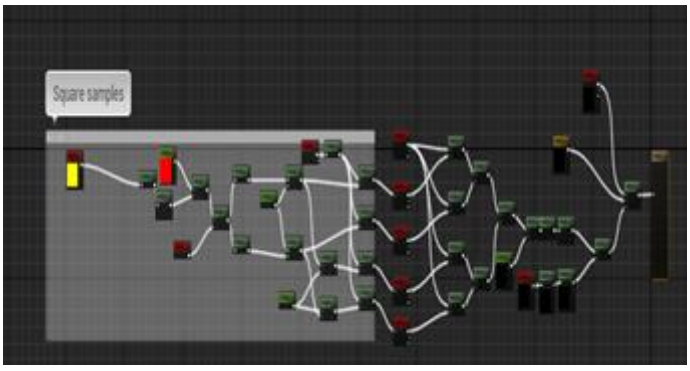
需求：根据摄像机位置，对任意形状的 Actor 使用红色高亮方式描出轮廓

实现方式：

利用一些图形学知识，考虑当前摄像机位置下，屏幕空间坐标的每个像素点对应到目标物体的 Z-Buffer 列表。情形如下：

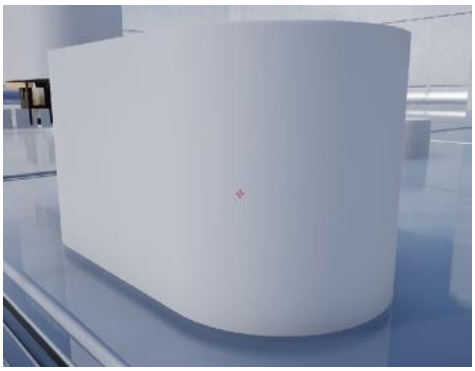
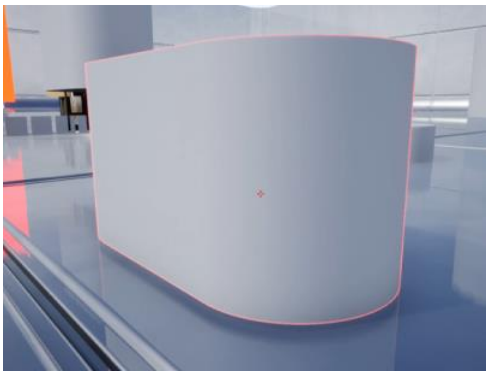
- 当某像素点具有 0 个 Z 值时，说明反推回世界坐标时该像素点是没有该物体的
- 当某像素点大于等于 2 个 Z 值时，说明反推回世界坐标时该像素点多次穿过该物体，不属于轮廓（例如前面与后面）
- 当某像素点恰有 1 个 Z 值时，说明反推回世界坐标时该像素点恰穿过该物体一次，因此属于视野轮廓，在后处理特效中添加一个红色圆点

实现采用创建一个特殊 Texture 的方式：



当上述 Texture 激活时则添加该轮廓勾边效果

效果：与 2.1 节展示的一致：

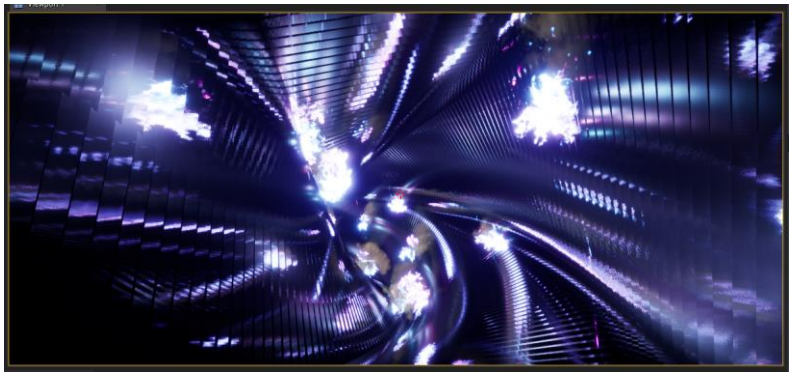



上图中左图为未开启轮廓勾边，右图为开启轮廓勾边。可以发现，该逻辑正确勾出了边框。

2.3 乱重力模拟隧道

需求：

- 整体空间为一个扭曲的隧道，该隧道如下图：



- 玩家在隧道中前进时重力会不断变化，且重力方向与脚下地面保持一致
- 隧道中有一些冷色的火焰团，玩家触碰之后会死亡重生

隧道重力实现方式：

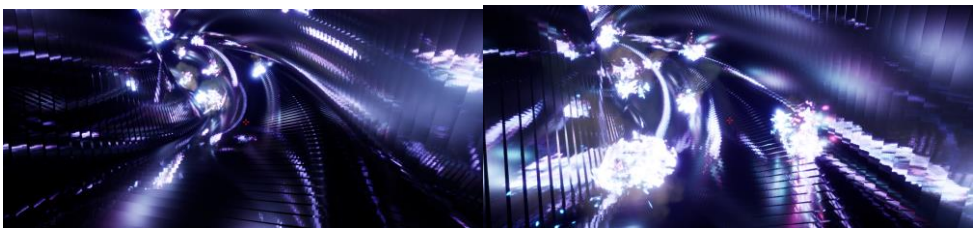
由于原生 UE4 引擎不支持任意角度的落地判定，直接使用加速度修改来模拟重力修改是存在落地判定不可用的技术问题的，这个问题需要引入极其复杂的引擎修改才能解决。

因此，我们使用了一个巧妙的方式来实现隧道旋转的需求：

1. 蓝图切分生成旋转隧道，每个面片根据所在圆心角计算出公转角与自转角，进行细致的坐标计算
2. 据玩家位置计算玩家出玩家在隧道中对应的公转角

3. 据玩家公转角计算出隧道应当额外旋转的自转角，更新隧道的生成相位

隧道重力效果：



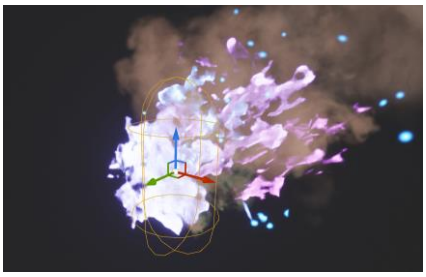
可以发现，当玩家在隧道中行进时隧道将会跟随旋转，保持玩家脚下地面的倾斜角不变化，可以制造出玩家重力改变爬上墙壁的认知感。

冷色火焰与**游戏逻辑**实现方式：

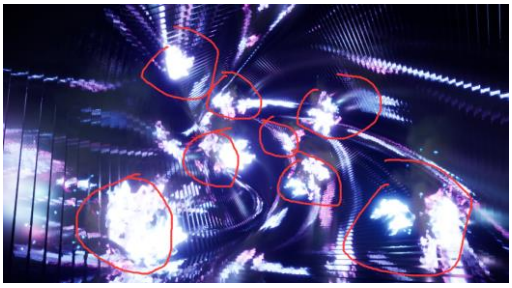
冷色火焰可以对 UE4 自带的火焰粒子特效进行修改，调节粒子群的各部件色温：

上图为调节之后的冷色火焰，主要渐变方式为生成火舌时用蓝白色，火焰接近消失时渐变为紫红色。

由于粒子群本身不具有碰撞体积（直接使用粒子碰撞效率低且判定范围过大），手动添加一个胶囊碰撞箱：



然后使用类似隧道外墙的方式将冷色火焰添加到隧道的四个曲面上，并设置一些伪随机偏移：



上图中的冷色火焰将随着玩家行进跟随隧道旋转，玩家需要找到较好的行进路线才能不被火焰击中。

2.4 复杂状态控制-华容道

需求：令玩家可以操作一系列物体组合，并判断合法性

实现方式：

1. 建立每个 Piece 的蓝图类，预留离散化坐标设置接口
2. 父类作为容器生成 Piece，自动设置贴图、坐标等
3. Piece 被点击时将信息上传父类，父类判断合法性

效果：

当玩家点击合法方块时，方块将被平滑地移动到空缺位置；点击不合法方块时无动作；正确解密之后，将触发下一步剧情

2.5 3DCapture 传送门

需求：

- 创建可以显示远处画面的物体

- 对玩家进行传送

实现方式：

1. 创建一个 3DCapture，使用 Capture_Cube 摄像机对目标位置的画面进行捕捉
2. 生成 Capture 纹理，使用镜面 UV 进行计算，得到反镜面 UV 坐标，使玩家可以有“透过”传送门看世界的观感
3. 玩家靠近传送门时将玩家传送到指定位置

效果：

可以发现，红框中的内容并不是场景的直接内容，而是远处房间的摄像内容；当玩家移动视角时，该传送门显示的内容将会改变，并且该内容是实时渲染的；当玩家进入传送门时，会被传送到远处的房间，但画面感和传送之前十分相似，会有较为平滑的过渡感

问题：

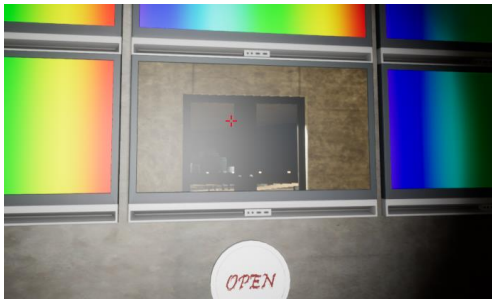
Cube_Capture 实时刷新具有巨大的开销，多个使用可能会严重影响性能

解决方式：

由于我们并不太需要实时更新渲染画面的能力，可以将捕捉频率改低来实现减少开销。实际上在改低频率之后开销是可以忽略不计的。

更多效果：

通过设置传送门的传送能力以及 capture 偏移角度以及 UV-map，传送门可用来实现其他内容：



上图为监控室效果，可以让玩家操作其他房间的远处物体并及时获得反馈

上图为实时反射镜面效果，可以增添恐怖气氛并用于某些特殊的镜面解密。该技术修改了 UV-map，使得摄像机视角与物体的 UV-map 产生镜面感。

2.6 其他技术

以下是一些实现较为简单的技术，因此仅给出需求与效果

2.6.1 画面黑白化

需求：实现画面全屏逐渐黑白化的过程

实现方式：在后处理特效果调整参数即可

效果：

2.6.2 任意长度密码锁

需求：实现任意长度的密码锁方便解密关卡设计

实现方式：自动计算密码方块相对坐标以及密码正确判定

效果：

3 UI 技术

3.1 开始界面

需求：显示游戏的开始界面，玩家可与界面上的 UI 交互进行开始游戏、更改设置等操作。

实现方式：

1.建立开始场景，在固定位置创建正交相机。

2.建立开始菜单 UserWidget 蓝图类，创建所需组件并为按钮绑定事件和动画

3.将玩家视角绑定在正交相机上，并在玩家窗口上显示开始菜单

效果：

3.2 游戏暂停界面

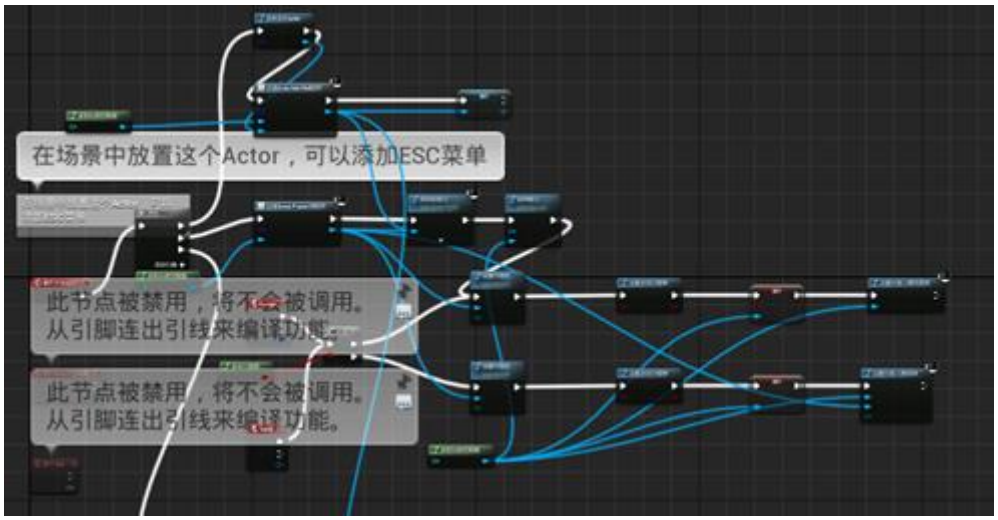
需求：

玩家可在游戏场景中暂停游戏，呼出交互菜单，进行保存、更改设置、退出游戏等操作。

实现方式：

1.编辑所需 UserWidget，创建所需组件并为按钮绑定事件和动画，使得点击按钮时能创建存档、设置等功能对应的 UI 面板。

2.定义并在场景中放置 UImanager 类 actor，用于接收键盘消息并管理所有 UI 界面的显示。该 actor 在每个关卡地图中唯一存在。



3.当该 actor 接收到相关键盘消息时，将暂停游戏，呼出游戏暂停菜单，允许用户与菜单交互。

4.当游戏暂停菜单（以及由其打开的其他所有菜单）被关闭时，UImanager 将恢复游戏的运行状态。

效果：

3.3 玩家 HUD 面板

需求：游戏进行时，在玩家窗口上显示各种信息。目前主要实现两种功能：

(1)当玩家注视可交互物品时，在界面上弹出该物品的名称、描述、相关图片（均可选）。

(2)当玩家解开特定谜题时，显示提示文本或剧情字幕。

实现方式：

1.创建 ActorInfoHud 类 UI，绑定在玩家摄像机上，由该类 UI 显示所需信息。

2.扩展开发报告 2.1 中描述的 BP_Actor_Pickupable 类，使得该类物体可以设置名称、描述、图片等属性。

3.通过上文所述 UImanager 类，可以在游戏中即时获取玩家正在注视的 BP_Actor_Pickupable 类对象（即可交互对象），ActorInfoHud 使用如下逻辑进行处理：

(a)若没有注视可交互对象，隐藏所有 UI

(b)若玩家摄像机注视了新的可交互对象，播放 UI 动画，弹出对应信息面板。在动画播放完毕之后，保持弹出的信息可见。

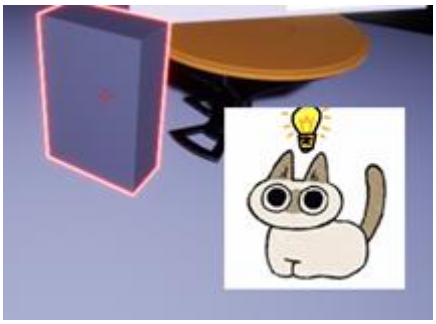
(c)若玩家摄像机不再注视原本的可交互对象，播放 UI 动画，收起对应信息面板

由此可以实现需求（1）。

4.关卡蓝图通过 UImanager 类可以间接调用 ActorInfoHud 类的函数，在画面上以指定位置、时长、内容、字号显示具有淡入淡出效果的字幕，由此实现了需求（2）。

效果举例：

（注视可交互物品显示图片）



（在屏幕上任意位置显示字幕）

3.4 带 UI 面板的场景中物体

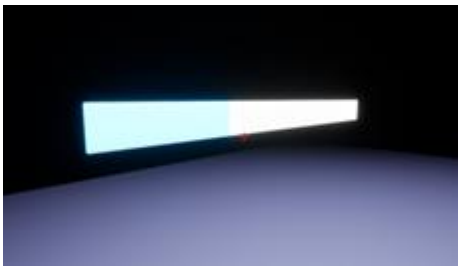
需求：游戏中需要有能动态显示信息的类“显示屏”模型，用于显示关卡进度等。

实现方式：

利用 UE4 原生的 WidgetComponent，可将任何 UserWidget 组件附加到指定 actor 上，因此只需定义所需 UI 面板，再将其挂载到 WidgetComponet 类上即可。此处不再赘述。

效果：

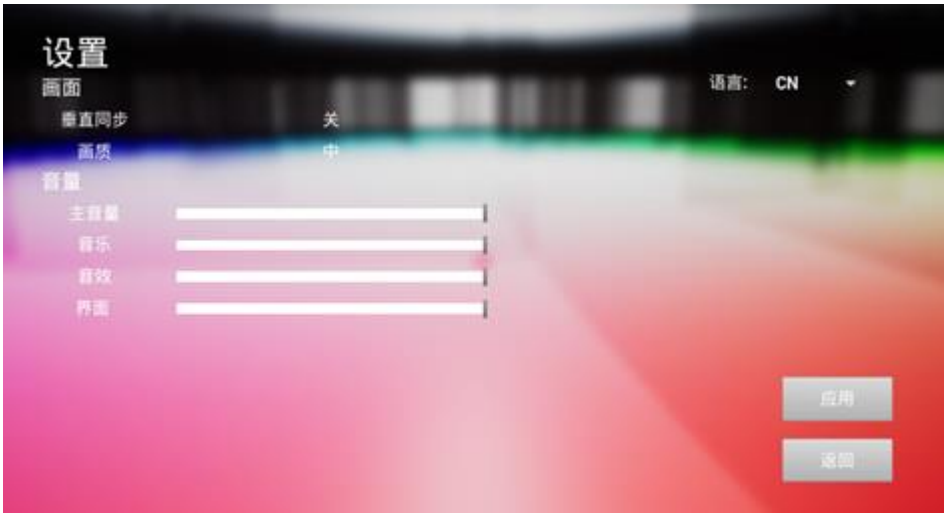
（场景中的进度条）



（显示时间的信息屏）

3.5 设置界面

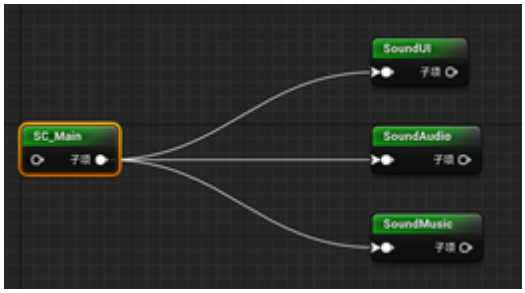
需求：令玩家能更改画质、音量、语言等全局设置。



实现方式：

1.画质的更改只需在 UI 中创建对应的按钮，显示当前设置，并在点击按钮时更改对应设置即可。

2.音量的更改需要提前为游戏中的所有音频文件设置所属音效类。这里主要用四个音效类管控三种不同类型的音频文件：



每个音频文件的所属音效类都要设为背景乐、音效、UI 音效三种之一：

3.之后，只需在更改音量滑条状态时，更改对应音效类的音量乘数属性，即可实现对全局音效的管理。

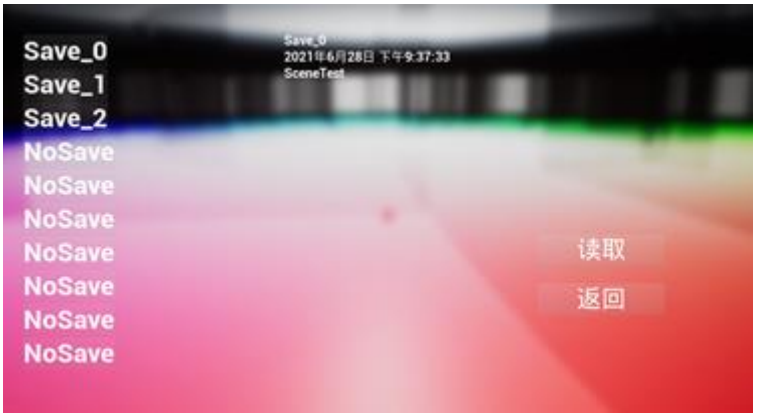
4.语言的切换使用了 UE4 的“本地化（Localization）”功能，详见下文 3.7

3.6 存读档功能

需求：玩家能够在游戏过程中存档、读档，保存或读取所在关卡、收藏品收集进度等。

实现方式：

1.创建存读档界面，显示已有存档、已选中存档信息、存读档按钮等。



2.继承 UE4 的 SaveGame 类，实现自定义的 mSaveGameObj 类，在其中保存需要维护的变量。在点击保存按钮时，可以将该类序列化存为本地文件。读取同理。

效果：玩家可以存档、读档。

3.7 多语言切换

需求：令游戏中的文本可以显示为不同语言。

实现方式：

1.建立字符串表（StringTable），按(key, value)的形式统一维护游戏中所需文本。

2.使用 UE4 的“本地化（Localization）”功能，收集字符串表和 UI 界面中出现的所有“文本(text)”类型变量，设定它们在不同语言环境下的显示内容。

3.在设置界面中定义切换语言环境的相关蓝图。

效果：游戏界面可以以不同语言显示。

（中文）

（英文）