基于 UE4 快速实现一款 ARPG 游戏雏形

鲁峻豪 白俊鸽
成都锦城学院计算机与软件学院 四川 成都 611731

【摘要】现多数开发者能接触到的游戏引擎中其大多数不适合开发大作，更别谈 3A 级别。因此本文基于 UE4 引擎实现快速利用 UE4 和 C++ 创建一款 ARPG 类游戏的雏形。

【关键词】UE4；C++；游戏引擎游戏设计

1 引言

现国家已主张大力发展电子游戏产业，游戏本身其能作为文化输出载体，如小说、电影等。但发展道路上仍是任重而道远，因国内游戏厂商多数为不做创新，尽量压榨成本开发。目前国内手游市场虽发展尚好，但除了手机平台，在 PC，XBOX（微软开发的游戏机）PS5（索尼开发的游戏机）Switch（任天堂开发的游戏机）等游戏机平台（针对这些游戏机，统称为游戏主机，文内中简写为主机）我国仍没有话语权。虽现今手机的用户群基数庞大，但手机的性能和操作模式注定了可以发挥的游戏性有限，无法实现在游戏主机和 PC 上的游戏性和物理效果以及优秀的光影效果和画质。而就算等到了云游戏普及，其本质只是换成在云游戏平台提供商的 PC 上运行。

2 系统功能需求

开发一款游戏首先要清楚需要完成怎样的系统和工作，明白如何让玩家快速上手，如让玩家看见角色就条件反射按下 WSAD 来控制移动；如需打造出怎样的世界，之后再进行具体的开发。

2.1 城镇和副本

至少需要一个城镇且城镇地图中有 NPC 可以接受和提交任务，副本地图则提供各种敌人来使玩家完成各种任务和获得道具。

2.2 战斗系统

RPG 游戏自然也需拥有战斗系统，战斗系统需要玩家容易上手。

2.3NPC

NPC 必不可少，游戏和其他媒体不同的便是游戏拥有交互，而 NPC 为交互起到了无可避免的作用。

2.4 不同的敌人

一款 RPG 游戏自然得拥有多种敌人，不然玩家便会觉得单调且不同的敌人得拥有不同的战斗难度。

3 系统功能分析

明确需求功能之后，下一步便针对功能从玩家和开发者两方角度来进行分析，通过对业务流程的梳理画出简单的系统概念图用以帮助开发者快速了解逻辑流转过程。从各个功能的实现分析，以图形方式表达系统的逻辑功能描绘出制作各个功能的流程图。

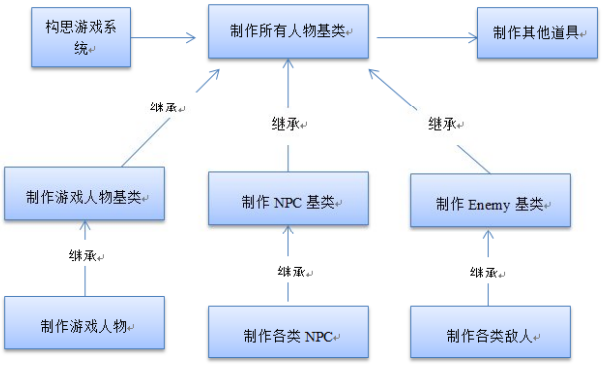

图 1 大致制作流程图

4 系统详细设计及实现

4.1 任务系统制作

4.1.1 任务系统设计

创建三个枚举分别负责任务类别、任务地点、任务目标。

E-Type：
主线任务
支线任务
副本任务

E_Region：
城镇
冰岛
（此处可以根据自己需要的任务地图来创建命名）

E_GoalType：
获取
击杀
寻找
交谈

之后再创建 4 个结构体分别负责任务信息，任务奖励，任务详情，以及任务地点：

S_QuestInfo 结构体成员：Name(字符串)、Kind(E-Type)、Describe(字符串)、Region(E_

Region)、Award(S_Award)、SuggestLevel(整 型)、Difficulty(浮点型)、GoalInfo(S_GoalInfo)。

S_Award 结构体成员：Exp(整型)、Honor(整型)、Money(整形)。

S_GoalInfo 结构体成员：Type(E_GoalType)、CustomGoal(布尔型)、CustomGoalText(字符串)、Target(字符串)、HuntNum(整型)、TargetLocat(S_TargetLocat)、UpdateQuestDescribe(布 尔 值)、UpdateDescribe(字符串)、FollowgoalIndice(整型数组)、Member(布尔值)、TargClass(Actor 类引用)、TargClassID(整型)。

S_TargetLocat 结构体成员：HasLocat(布尔值)、Locat(向量)。

4.1.2 功能点实现与关联

创建一个 Actor 类，其命名为 RulerQuest 并为其添加成员变量：QuestInfo(S_QuestInfo)、GoalIndice(整型数组)、Indice(整型数组)、HuntNum(整型数组)、Info(S_GoalInfo 型数组)、SelectGoalIndex(整型)。

再创建三个函数。

1.Update：

首先清空 Info 的元素，获得 Indice 当前下标做一个循环遍历，输出数组的元素获取为 QuestInfo 里面的 GoalInfo，循环出的结果加上 Info，这便完成了更新。

2.SetupStartingGoals：

首先清除 Indice 的内容把 GoalIndice 的内容赋值给 Indice 然后执行 UpdateInfo 函数最后返回。

3.GoToNextGoal：

获取 Indice 数组的最大值再 +1，若 <=QuestInfo 的 GoalInfo 的最后一项下标，则返回为真，若返回为假，函数直接返回 0。接着执行创建一个临时变量为整型来保存 Indice 最大值 +1 之后的数，然后清除 Indice 数组的内容，再加上临时变量和 Indice，然后执行 Update 函数，之后把返回结果用布尔值来检测是否成功。

最后编译保存。

创建一个组件并为其创建多个需要用到的不同变量。

表 1：QuestManager 组件成员

变量名字	类型
CurQuest	RulerQuest 对象引用
AllQuest	RulerQuest 对象引用数组型
AllQuestClass	RulerQuest 类引用
Player	角色类对象引用
本地变量	类型
LocQuestClass	RulerQuest 类引用
LocQuest	RulerQuest 对象引用

角色类为游戏操控角色的基类，玩家的主要功能写在这里面，当需要建立不同的游戏角色时只需让其继承后再添加需要增加的额外功能即可。

创建函数 AddQuest，为 QuestClass 类型的 RulerQuest 类引用，负责接受任务：

先使用包含方法检测 AllQuestClass 变量然后把传出的布尔值进行非运算，与 QuestClass 进行与运算把输出的布尔值给 LocQuestClass 变量执行判断，若为真则执行 AllQuestClass 和 LocQuestClass 的与运算，然后使用方法生成 Actor 创建 LocQuestClass 并把此方法返回出来的值赋值给 LocQuest 和 UpdateQuest 函数，让 LocQuest 与 AllQuest 相加，接着执行 SetupGoal，函数对象为 LocQuest。最后执行任务更新函数，对象为玩家基类的 UI 对象。

编译保存之后，把自建的任务管理组件添加给玩家基类，这样所有基于玩家基类为父类继承的角色对象都拥有了接受任务，更新任务的功能，需要什么功能时可直接通过添加创建的任务类里面的信息即可。

4.1.3 玩家从 NPC 接受任务

创建新的类继承自创建好的任务类并添加相应的任务信息。

主角类里添加按键映射，为玩家基类添加触发器，当 NPC 进入触发器范围时，按下接任务的映射键，执行自建的任务管理组件里的接受任务函数，需要传入的任务类选择新建的任务类即可。

4.2 战斗系统

4.2.1 敌人制作

创建 Pawn 感应组件。组件可设置 AI 的视野范围，视野侧面角度，听觉范围，是否看到玩家等。因为人眼视野范围大致为 120° 这里便把 AI 的视野侧面角度设置为 60° 即可。

自己创建一个新组件，并为其创建变量 PatrolIndex(整型)、ReverDir(布尔值)。

创建行为树命名为 BT_Enemy。

编辑 AI 逻辑，从根节点往下此创建两代 Selector，并为其添加一个黑板，往下再创建两个 Sequence，左边 Sequence 为其创建黑板，向下分别为 FindRandomPoint（如果 AI 只到指定的地点巡逻便会显得不那么智能，所以这里选择随机然后设置一个随机范围即可）、MoveTo、Wait；右边的 Sequence 向下分别为 DistanceCheck、MoveTo 并为其添加黑板、LaunchAttack。

黑板为UE4自带的一个集中行为树数据的设计模式。

创建黑板命名为 BB_Base，为其创建 keys 参数

表 2：BB_Base 行为树参数

键	键类型
SelfActor	Object
Behavior	枚举
Dead	布尔值
Patrolradiu	浮点型
OriginLocat	向量
MoveToLocat	向量
SeenPlayer	布尔值
Player	Object
AttackRange	浮点型
InAttackRange	布尔值

创建一个 AIControl 让其使用 BB_Base，执行 BT_Enemy。

创建角色类命名为 Enemy 作为一切敌人的基类，并为其添加碰撞体检测是否受到攻击，同时为其添加变量 Healthy(浮点型)、SeenPlayer(布尔值)。

当玩家能够造成伤害的物体接触到敌人的碰撞体时，类型强制转换为玩家让敌人造成伤害的对象，获取碰撞时的坐标，把碰撞时的坐标传值给在位置处生成发射器函数和生成 Actor 函数执行，并选择相应的粒子特效，如用粒子特效创建的流血特效，爆炸特效等；和敌人受

到伤害的UI，然后设置生命值的减少。再执行判断，如果生命值<=0再次使用生成Actor函数，此时选择敌人死掉之后掉落的道具，如货币等，最后使用摧毁Actor函数使其在内存中被清理。

制作不同的敌人只需都使其继承自敌人基类，有不同的功能只需单独添加即可。

4.2.2 玩家战斗系统

为玩家基类添加变量：Attack(布尔值)、Count(整型)。

在玩家基类里添加攻击键的映射，然后执行判断是否玩家属性里面的体力值>=需要消耗的体力值，如果为真则再用Attack判断是否为攻击状态，若为真则计算玩家按下攻击键的次数，如总共有三种攻击动作，判断玩家按下的键是第几次，设置Count如果是1则播放第一击人物动作动画，是2则播放第二击人物动作动画等，并在后面都减去需扣掉的体力值。

若玩家是近战攻击单独在武器模型上添加碰撞体即可；若是远程攻击则需添加一个类作为子弹。

创建新的Actor类命名为Magic添加相应的模型，这里因为是作为魔法攻击所以选择添加的粒子特效，再添加发射物移动组件（此组件会每帧一直更新位置，可以实现反弹，前进等运动）在里面设置其初始速度和最大速度，把反弹关掉，重力根据自己的需求设置。

创建一个游戏角色继承自玩家基类作为玩家游玩的角色，覆写父类的攻击函数，在攻击函数里面的最后使用生成Actor函数，选择Magic作为其需要创建的对象实例即可。

5 结语

该项目主供PC端使用，整个项目的模型和美术特效方面均使用EPIC官方免费提供的素材。

该项目只能运行在PC平台上，但移植到索尼平台和微软平台问题不大，只需更改为手柄按键映射。

本系统主要完成功能有任务系统、升级系统、战斗系统、玩家角色腿部IK功能、昼夜变换系统、敌人的行动逻辑、NPC对话、装备系统以及拾取道具等。

若时间充足，且并有影视专业人员提供帮助指导，还可添加镜头动画效果实现电影般的运镜。

【参考文献】

[1] 赵伟明．基于UE4的实时光线追踪技术在展示设计中的应用研究[J]．电子测试，2021(06):43-45.

[2] 吴洪晨．基于UE4的ACT类游戏的设计与实现[J]．产业科技创新，2019,1(02):86-88.