MiniGame 文档说明

周威威

1. 概述

1.1游戏概述

我做的 MiniGame 是一个 3D 塔防游戏，玩家要做一名合格的外星带路党，代表外星人来入侵地球，需要面对人类一波又一波的反击。

因为时间有限，总共只花了 5 天的时间，还有很多细节没有完善，但是大致还是满足了一般塔防所必要的游戏性。

1.2玩法概述

游戏中有敌对的两方:外星人和地球人。玩家需要扮演外星人，在规定的时间内通过建造不同类型的飞船来对抗反击的地球人。

1） 外星人方面

地图上分布着若干基地，玩家扮演的外星人方面可以在固定的基地上建造两种不同的飞船来打击地球人。

创新点：除了有一般塔防游戏中的固定基地，我还设计了一种可以不需要基地的飞船：移动型飞船，它可以建造在任何位置。它可以吸收范围内的敌人，并把他们沿原路径折返，有冷却时间和持续时间。

下面是外星人方面游戏角色的概述：

a) 防御型飞船：鼠标左键点击基地可以建造。有固定的防御范围，建造后会释放 3 个外星小兵，他们会在飞船的防御范围内巡逻。
b) 外星小兵：攻击方式为激光枪。防御型飞船建造后自动释放，他们会根据设计的 AI 在飞船防御范围内智能的巡逻，当有敌人进入飞船的防御范围时，他们会移动到敌人的位置并攻击最近的敌人。
c) 攻击型飞船：攻击方式为炮弹。鼠标右键点击基地可以建造。有固定的防御范围，建造后探测攻击范围内的敌人并攻击。
d) 移动型飞船：

2） 地球人方面

创新点：不同于以往塔防游戏敌人路线固定，每个地图上的道路都是可以行走的，游戏设计者可以配置敌人按既定路线或者让敌人自由行走。

地球人作为攻击方，暂时设计了三种不同的游戏角色：

a) 普通大兵：灵感来源于红警里的美国大兵，移动速度和生命值都比较低，是炮灰型角色。
b) SWAT 士兵：移动速度和生命都比普通士兵高。
c) Tank：移动速度最慢，但是可以攻击外星人小兵，是防御型飞船的克星。

1.3 交互概述

玩家视角:可以通过方向键和 WASD 控制摄像机的移动，鼠标滑轮可以控制

游戏画面的缩放。

建造飞船：通过鼠标左右键点击基地，可以建造不同的飞船。此外，通过拖拽图标到指定位置释放，可以建造移动型飞船到该位置。

拆除飞船：鼠标右键点击飞船可以拆除飞船。

2. 逻辑系统的设计思路

游戏的逻辑系统分为三大部分，分别是主框架，管理器模块和 AI 逻辑模块。下面分别就这三个模块讲述下我的大致设计思路。

2.1 主框架模块

Manager 文件夹下的 GameKernel 脚本。

主要功能是：

1. 管理游戏中的各种管理器，负责存储游戏时间，玩家血量等初始数据，通过这些数据初始化各种管理器。
2. 更新 UI 显示，根据游戏进度，实时更新各种 UI 内容。
3. 大部分 UI 触发函数，负责响应 UI 中的菜单中的各种按钮，控制游戏主要进度。
4. Update()中更新上述所有管理器。

2.2 管理器模块

2.2.1 金钱管理系统

金钱管理系统下面包括两个脚本：MoneySystem 和 PricePair

PricePair: 游戏中的商品类，成员有商品类型和价格。表示基本的出售单位。

MoneySystem：管理游戏中的金钱系统，有初试资金和当前资金等主要数据成员。主要功能是设置初试资金，以及飞船的买卖和消灭敌人后的奖励加成。此外，在 GameKernel 中将它设置为静态对象。

2.2.2 输入管理器

InputSystem：主要负责游戏中的输入检查，主要成员函数响应鼠标点击事件。在 GameKernel 中将它设置为静态对象

2.2.3 攻击逻辑管理器

AttackLogic: 提供基本的攻击逻辑，比如：判断一定范围内的敌人，获取最近的敌人等等，提供给其他角色使用。

WeaponControl: 武器的控制脚本，可以绑定到游戏中的各类武器上，主要数据成员有：武器速度，伤害值，和目标 tag。可以攻击制定 tag 的敌人。

2.2.4 敌人管理系统

敌人管理系统包括两个脚本：ArmyControl 和 EnemyControl

ArmyControl: 军队管理器，准确说不属于敌人管理系统，此后如果扩展了本方的军队功能，也可以使用。用户可以通过它在面板上配置敌人的行军路径和士兵种类，士兵数量等一系列属性。

它的主要功能是存储军队的数据，包括：军队成员对象、士兵数量、军队路径和生成时间等成员。主要函数 Init()负责军队的初始化。

EnemyControl: 敌人管理器，用来配置敌人信息。主要成员有：军队列表、存放军队生成时间的数组和循环生成军队的标志。可以把军队列表内的成员按照他

们的属性进行生产，同时，loopArmy 标志位如果为 true，则开启了循环生成的模式。

2.3 AI 模块

2.3.1 状态基类

BaseState 脚本存放在 Manager 文件夹下，表示状态基类，其他所有的状态都继承了它。它提供了两个虚函数 enter()和 execute()，提供状态进入和执行时的逻辑。

2.3.2 士兵的 AI 实现

在游戏中，我把士兵分为两种类型的士兵：在一定范围内移动的士兵 Tower Soldier 和按一定路径行军的士兵 Attack Soldier。分别为他们设计了两种不同的 AI 逻辑。

1. TowerSoldierControl: 为在一定范围内移动的士兵设计的逻辑脚本。

该脚本除了存储士兵的一般属性外，还维护了一个状态列表和当前状态。

下面介绍它的两种状态：

TSPatrolState：巡逻状态

enter()，在巡逻半径内生成一个随机位置，并移动到该位置巡逻。

execute()，角色会在巡逻半径内随机位置进行巡逻，检查巡逻半径内是否有敌人，如果有，转为 TSAttackState 状态。

TSAttackState：进攻状态

enter():判断当前敌人是否为空，如果不为空，判断敌人是否在攻击半径内，如果在，移动到敌人位置并攻击。如果敌人出了攻击半径，重置敌人对象，继续寻找，如果攻击半径内确定找不到敌人了，会转为 TSPatrolState 状态。

2. AttackSoldierControl：为按一定路径行军的士兵设计的逻辑脚本。

该脚本大致思想和上面差不多，也是维护了一个状态列表和当前状态。此外，还维护了一个行军路径列表，用来控制士兵的行军路线。

下面介绍下他的两种状态

ASPatrolState：行军时的巡逻状态

execute(): 检查距离下一个路径点的距离，如果小于 0.5，更新路径点。检查攻击范围内是否有敌人，如果有，转为 ASAttackState 状态。

ASAttackState：行军时的攻击状态

execute(): 也是判断敌人是否为空，如果不为空，判断是否在攻击范围内，不同之处在于，角色不移动到敌人位置，而是继续在路径上。如果攻击范围内找不到敌人，转为 ASPatrolState 状态。

2.3.3 飞船的 AI 实现

TowerFactory：负责基地生成不同的飞船。

UFOArmyControl：防御型飞船的控制脚本，主要功能是批量的生成外星人小兵，并初始化他们。

AttackTowerControl：进攻型飞船的控制脚本，主要是探测敌人是否在攻击范围内，如果存在敌人，则对敌人进行攻击。

MovingTowerControl: 移动型飞船的控制脚本，主要功能有吸收敌人，反转敌

人的行军路径等特殊的功能。

3. UI 和交互的设计思路

3.1 UI 的设计思路

我把 UI 分为三部分，分别是位于游戏上方的玩家状态栏，中间的菜单提示栏和下方的技能栏，下面一一介绍：

1. 玩家状态栏：有剩余金钱、剩余生命值、剩余时间和暂停按钮。前三个是通过 GameKernel 来更新，暂停按钮可以调出中间的菜单。
2. 菜单提示栏：有胜负提示栏、用户得分栏和三个菜单组成。其中胜负提示栏会在游戏结束时提示玩家输赢，得分栏会根据用户的表现不同提示 1-3 颗星星，三个菜单提供重新开始、返回开始界面和恢复游戏。
3. 技能栏：位于画面的左下方，有技能冷却时间倒计时提示，冷却时还有颜色区分。通过它可以建造移动型飞船。

3.2 交互的思路

因为时间有限，没有太过于考虑交互方面的细节，只是让用户完成游戏的同时，不会像一般的塔防一样，局限于固定的视角和位置，因此可以通过键盘改变视角和远近，同时也不会影响到鼠标的操作。

4. 总结

这个游戏从构思到做完大概花了 5 天多的时间，时间紧迫，但是，在一些玩法上，并没有局限于一般塔防的游戏元素，做了一些微创新。比如：

1. 因为是 3D 塔防，因此添加了可以移动视角和变换大小的功能。
2. 飞船不再局限在建造在基地上，添加了可以任意放置的飞船。
3. 敌人的行军局限不再是固定好的，游戏设计的时候可以通过配置选择敌人是按照固定路径行军或超目标自由行军。

在游戏脚本文件夹里，还有一个重构的文件夹，里面的脚本是我后期打算重构时用的，暂时没有被使用。

这个游戏原本设计的是玩家可以选择外星人或者地球人，但是因为时间有限，暂时完成了扮演外星人的部分。后面如果有时间我会通过重构的框架再添加地球人模块和更多的地图关卡。