末日之战设计说明书

课程: 软件工程 2019 秋

软件名称:"末日之战"塔防对战游戏

组名: 挑战不熬夜

组员: 邬军、桂延智、葛丛钦、王宇昂、杨祖超、胡啸、朱泽宇、卢烈、徐迎港

1. 简介

1.1 产品概况

塔防游戏种类繁多,受众广泛,游戏模式简单,娱乐性强,植物大战僵尸、保卫萝卜、明日方舟等游戏都是基于塔防模型进行创新,取得了巨大的成功。但是目前市面上稳定的、平衡性好的塔防对战游戏相对较少,并且质量较低,主要问题在于攻守双方玩家的平衡性难以保证,同质化严重,操作空间小,不同水平玩家游戏体验存在很大差异。

为了保证更好体现游戏的策略性与游戏体验,我们基于 PC WINDOWS 开发一款的塔防游戏,可以自行选择地图与攻守阵营,游戏开始阶段主要是角色的放置、道具的使用与及时的反馈,游戏全程攻守双方具有各自明确的目的,同时操作上具有较大差异,玩家全程充分参与,避免出现挂机等快餐元素,满足塔防爱好者的需求。

1.2 产品特色

- 1) 具有多个地图、角色可供选择。
- 2) 采用二维地图,进攻方与防守方的行动与攻击不局限于直线,可以转折。

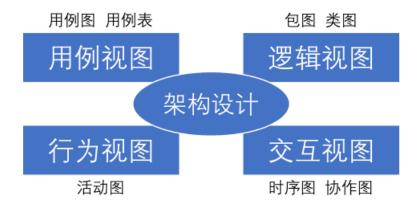
1.3 用户类别与特征

该产品主要面向青少年,不同年龄段玩家均可参与。

2. 游戏规则设计

- 1. 用户在选择完游戏模式(进攻/防守),地图和出战角色后,进入战斗阶段。
- 2. 单机进攻模式中,玩家需要扮演进攻方,放置进攻角色和使用道具来突破电脑部署的防线(每过一段时间电脑会重新部署一次防线,没有费用与冷却限制,采用人为设计关卡,不具备 AI; 技能则根据冷却自动开启),到达目标地点角色超过一定数量即可获胜。
- 3. 单机防守模式中,玩家需要扮演防守方,在可以放置的地块上放置防守角色,使用技能和道具来抵御电脑的进攻,同样电脑按照设计的时间节点从进攻据点生成固定的进攻角色,技能根据冷却自动开启,不具备 AI,防守一定时间即可获胜。

3. 程序架构设计

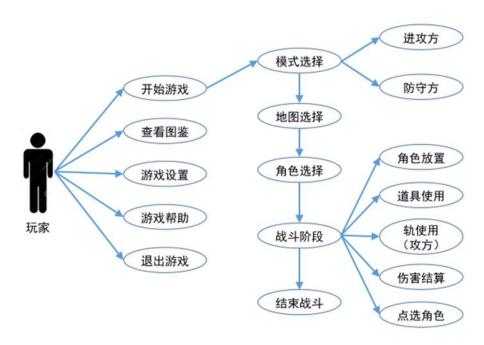


本文通过以上视图阐述程序的架构设计,其中用例视图、行为试图与交互视图相关内容已在需求分析文档中详细说明,这里只给出用例图、包图与类图。

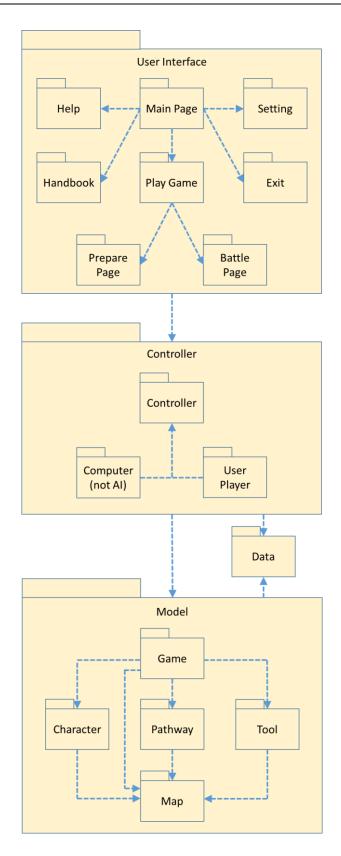
3.1 开发平台

本程序编写语言为 Python3.7,是基于 pygame 平台进行开发,可以在 WINDOWS8.1 以上版本电脑内运行。

3.2 系统用例图



3.3 UML 包图



3.4 类图

MODEL GameController +START_GAME_STATE: Integer = 1 +DEX_STATE: Integer = 2 +GAME_HELP_STATE: Integer = 3 +GAME_SETTINGS_STATE: Integer = 4 +EXIT_STATE: Integer = 5 +GAME_MODE_SELECT_STATE: Integer = 6 +SINGLE_MODE_PREPARE_STATE: Integer = 7 +MULTI_MODE_PREPARE_STATE: Integer = 8 +GAME_END_STATE: Integer = 10 +DEFENDER_SET_STATE: Integer = 11 +ATTACKER_SET_STATE: Integer = 12 +TRAIL_CHANGE_STATE: Integer = 13 CharacterModel +CHARA_SPECIAL_STATE: Integer = 14 +ITEM_USE_STATE: Integer = 15 +MOVEMENT[4][4]: Integer +last_created_time: Integer = 0 +ATTACK_HIT_STATE: Integer = 16 +HP: Integer = 0 +attackerAdded(self, attacker): void Tool +ATTACK_POWER: Integer = 0 +defenderAdded(self, defender): void +reach_model: Integer = None +repertory: Integer = 0 +reloadLevel(self, level): void +addCharacter(self, character): void +type: String _init__(self, position, direction): void +cool_down_time: Integer = 0 +setVolume(self, vol): void +init_image(self): void +update(self): void +initCharaDex(self): void +__init__(self): void +initItemDex(self): void +attack(self): void +init_image(self): void +attacked(self, loss, attacker): void +update(self): void +die(self): void +use (self,target): void +special(self, character): void +get_coordinate(self): void levelLoad(i) +reachable(reaching_char, reached_char, reach_model): void mapDisplay(map) dataLoad(i) Block Map +rowNumber: Integer = 1 +canPlantOn: Bool =True +canZombieOn: Bool = True +columnNumber: Integer = 2 +maps[]: Block +isPlantOn: Bool = False +isZombieOn: Bool = False +__init__(self,m,n): void +isHome: Bool = False +setSomeBlock(self,i,j,styleNumber): void +isBornPoint: Bool = False +setHome(self,i,j): void +setBlock(self,styleNum): void +setBornPoint(self,i,j): void