软件设计文档

技术选型

开发工具 unity 3d

游戏类型为 3D 游戏

itween 动画库

实现了 move 和 attack 的动画

move 为例

移动一步

```
endPointToReach = path[0].transform.position;

opt.Add("position", endPointToReach);

opt.Add("speed", move_speed);

iTween.MoveTo(gameObject, opt);
```

把 gameobject 移动到 endPointToReach

移动一个路径

```
Vector3[] points = new Vector3[useMoves];

for (int i = 0; i < useMoves; i++) points[i] = path[i].transform.position;

endPointToReach = points[points.Length - 1];

opt.Add("path", points);

opt.Add("speed", ((move_speedMulti * useMoves) + 1f) * move_speed);

iTween.MoveTo(gameObject, opt);
```

把 gameobject 沿着 points 里的节点依次移动

navmap 地图

地图实际上是一个 N*M 的棋盘，棋盘的每个格子是一个 TileNode

物体实际上是通过和 Node 的绑定而决定他的位置

地形则是由 TileNode 决定，TileNode 根据其不同的类型，决定这个位置的地形，实际上也就是有哪些单位可以放置在这个位置上。

主要的函数

public TileNode[] GetPath(TileNode fromNode, TileNode toNode, TileNode.TileType validNodesLayer) 寻路函数

void CreateTileNodes(...) 创建 TileNode　void LinkNodes()　将节点链接起来，表示他们是邻居，也就是可以互相到达

void SetAllNodeMasksTo(TileNode.TileType mask) 设置地形

void AddToTileNodeMasks(TileNode.TileType mask, Transform parent) 增加地形等级 更高的等级可以容纳更多类型的单位

1. 架构设计：

a) 主摄像机部分：

主要由 common 中 CameraMove.cs、CameraOrbit.cs 两个文件进行控制。

CameraMove.cs 和 CameraOrbit.cs 均挂载在主摄像机上：

CameraMove.cs：主要负责控制在非选中状态下，玩家可以根据键盘或图形化接口对游戏内的视野进行移动，以及游戏移动的速度。

并且实现了当用户选中单位后，摄像机跟随着被选中单位进行移动。

CameraOrbit.cs：主要负责控制在非选中状态下，玩家可以调整视野的方向和远近。

b) 游戏流程控制部分

主要由 GameController.cs 进行控制。

在 GameController.cs 内首先对玩家控制单位的状态进行设置，保证基础的回合制的进行

```
private enum State : byte { Init = 0, Running, DontRun } //枚举控制单位的状态
```

之后在 SpawnRandomUnits 函数中随机生成一定数量的玩家控制单位并进行初始化。

并实现了玩家切换之后，所控制的单位的状态全部初始化。

控制玩家选中单位后，被选中单位上出现选择器，并展开可移动范围和攻击范

围，使用 CameraMove.cs 中的跟随接口，实现跟随。

并实现当移动/攻击完成后处理。

c) UI 设计部分

由 CameraMove.cs、startUI.cs 和 SampleGui.cs 进行控制。

在 CameraMove.cs 实现了可视化的移动窗口，在 SampleGui. cs 中实现了玩家回合的显示和切换，并创建了实时小地图。

d) 单位设计部分

通过 Unit.cs、NPCBlood.cs、SampleWeapon.cs 进行控制。

Unit.cs：负责实现单位的生成，攻击，死亡等。

NPCBlood.cs：负责实现可视化的血条。

SampleWeapon.cs：负责实现导弹攻击的模拟。

e) 地图设计部分

通过 MapNav.cs、TileNode.cs、TMNController.cs 进行控制。

MapNav.cs: 负责通过 TileNode.cs 实现地图创建的功能和在 TileNode 节点上的移动。

TileNode.cs：负责设置单位、地图块。

TMNController.cs：综合控制以上两个地图相关，并将其实现与场景中。

f) 拓展部分

TNELinksOnOffSwitch.cs、TNEMovementModifier.cs、SelectionIndicator.cs。

SelectionIndicator.cs：为选择器，控制选中单位实现出被选中的特效

TNELinksOnOffSwitch：扩展打开节点及其邻居之间的链接。

TNEMovementModifier：拓展禁止移动区域的修改

g) 外部依赖。

NaviUnit.cs、iTween.cs 两者共同实现了战棋的移动方式。

一、 图形化界面及视角控制模块

1 玩家视角移动和旋转

目的是实现玩家通过点击虚拟按键进行视角（即屏幕内显示内容）的移动以及旋转，其中旋转是基于一个点进行 360°的视角切换，视角移动可以让玩家通过点击虚拟按键观察此刻不在屏幕内的内容；

虚拟按键以传统的 WSAD 四键控制放置在屏幕右下角，通过点击不同按键来进行视角的不同方向的平移，到达视角极限后将限制其移动功能。

2 视角距离的变化

目的是通过操作来控制视角距离的变化，实际为游戏画面的缩放，并且在缩放过程中游戏画面内容要保持相对不变，即相当于人通过走近或走远目标来实现视野内物体画面的大小和物体的数量，方便玩家即可观看战场全局形式也能注意小范围内战斗单位的状态。通过控制摄像头（camera）距离游戏地图的距离实现视角距离变化，即控制的并非游戏画面的移动而是移动摄像头的位置，以近大远小的原理来体现视角距离的变化。

3 游戏界面切换

目的是实现游戏开始显示一个开始界面，只有当玩家点击“start”按钮才能进入游戏战斗场景。实现方式是通过判定按钮是否被按下，进而通过调用 scene 的加载函数实现场景的切换。

二、 游戏角色控制模块

1 游戏角色的控制

回合判定后选中、取消选中角色，被选中角色的状态变化（移动、攻击等）。首先通过回合这一变量控制玩家 1 和玩家 2 的回合，每个玩家只能在它的回合内才能操纵游戏角色。通过点击游戏角色可以选中，点击其他角色则取消上一选中对象并选中最新选择的攻击对象。

2 游戏角色的状态

攻击单位的血量控制：包括血量初始化，随被攻击而减少等，血量数值通过血条的形式表现，每个攻击单位初始化血条相同并且都是满值，在收到攻击后会定量减少血量；武器控制：攻击单位武器为炮弹，实现攻击时炮弹发射及打中对象的动画及状态影响，通过在自己回合内点击攻击单位显示出移动、攻击范围，再点击攻击范围内的另一玩家的游戏角色进行攻击。

3 切换游戏控制回合

通过点击切换按钮实现游玩双方的攻守回合交换，只有在进攻环节才能进行移动、攻击、设置防御等行为；点击切换按钮后，按钮上的信息显示会变换显示当前为哪个玩家的回合。

三、 地图及游戏角色模块

1 地图格子控制

在地图模型上生成方格，并控制其可视化与否。计算移动路径，同时判断路径上的不可移动位置。每个方格分配节点用于设置位置可放置的对象类型（比如战斗单位和障碍物等），主要显示为点击选中游戏角色后，会在该角色周边以透明格子的形式显示出其可移动或攻击的范围，再通过点击这些格子或格子内的其他玩家的游戏角色实现移动或攻击。而一旦移动或攻击完成，或者取消选中该单位之后，这些格子会消失不可见。

2 战斗单位（即游戏角色）

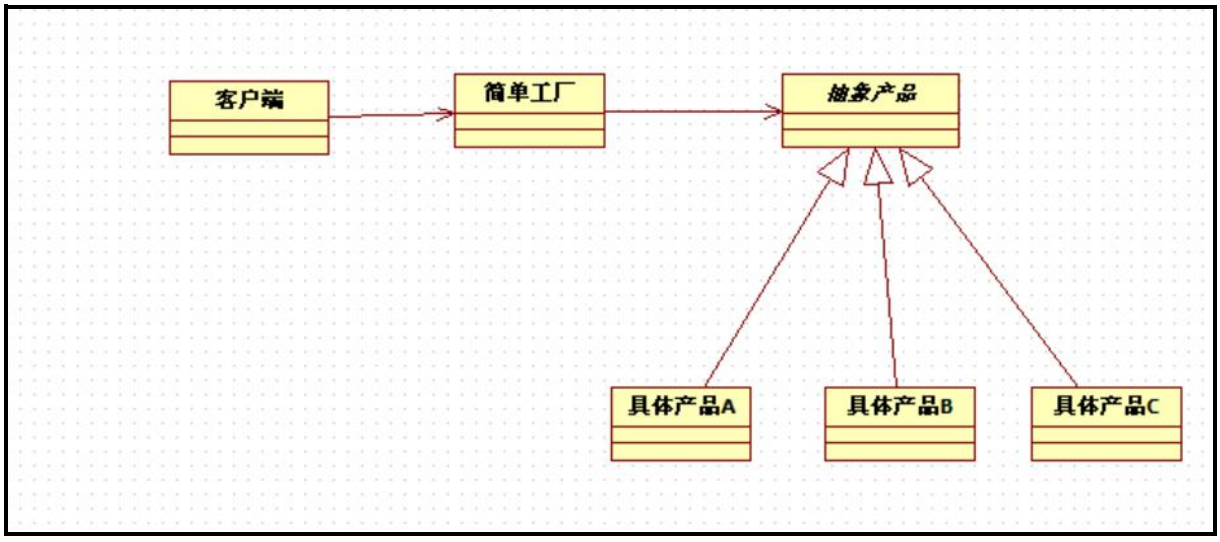
在可放置战斗单位的格子处（格子不可见）非固定位置生成战斗单位，并通过与节点链接的断开重连和位置变化实现战斗单位的移动；其中节点与各格子相对应，画面表现为格子，而其中实现是以节点来操作的。战斗单位会在游戏开始时在地图上生成，不同玩家的初始战斗单位数量相同。

3 节点间的链接控制

控制链接信息，新建链接等来控制游戏区域是否为可移动区域，通过节点间的链接信息来进行邻近角色、障碍物判断、攻击对象等的识别，以及移动路径的生成和移动的实现。

软件设计技术

简单工厂模式



游戏界面为客户端

gamecontroller 为简单工厂

unit，tilenode 等为产品。

用户在游戏界面的发出请求，由 gamecontroller 处理，分别调用相应的 unit 的 move，

attack 事件，select 事件。以及 camera 的移动事件

面向对象程序设计

所有代码均采用了面向对象的程序设计。

玩家单位，摄像头，选取点，都有相应的对象。

玩家单位对应 unit。

unit 主要包含 HP， 能否攻击，所属位置（所绑定的 node）等属性

start() 初始化，实例化一个 weapon

bool CanAttack(Unit target) 判断是否可以攻击

bool Attack(Unit target) 攻击动作

void UnderAttack(int del) 被攻击动作

血条对应 NPCBlood

NPCBlood 主要包含

update() 更新血条信息，大小，血量等

OnGUI() 生成血条

武器对应 SampleWeapon

SampleWeapon 主要包含

void Play(Unit target) unit 调用他相对应的 weapon 的 play 来实现攻击

NaviUnit

Unit 的基类。 用于绑定所有单位到 node 上，单位的寻路，以及运动。