**南京航空航天大学**

《面向对象程序设计语言》课程设计报告

**卡牌游戏 ZooFantasy**

**学号：161610327**

**姓名：周子潇**

**日期：2017.5.7**

1. **需求分析**

随着炉石传说等卡牌游戏的日益火热，卡牌游戏的玩家群体也在逐渐增长。鉴于卡牌游戏不需要大量美工支撑，又能体现算法逻辑，又受大众的喜爱，所以我决定开发一款卡牌游戏。

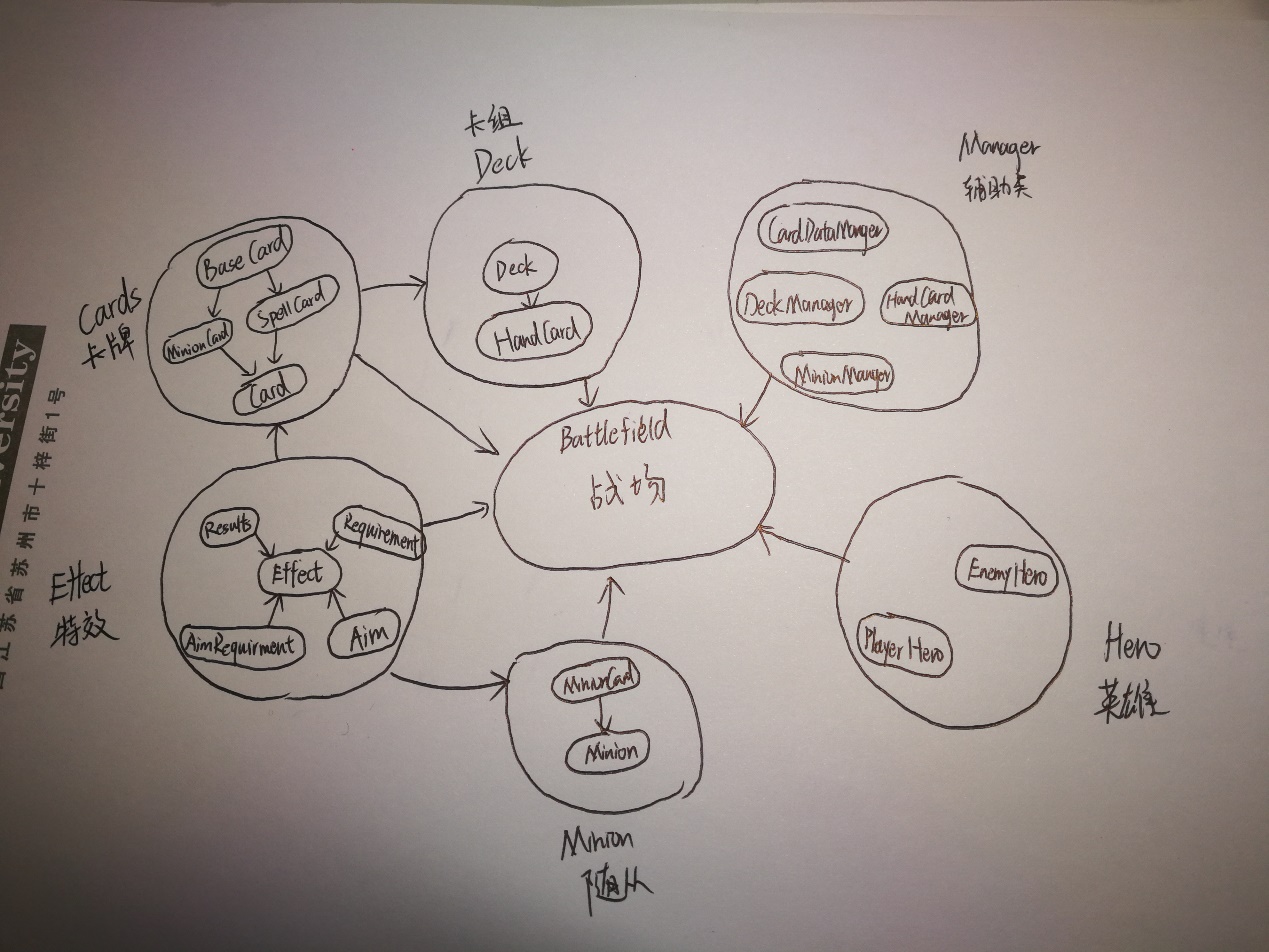
开发早期使用的是C#控制台程序，开发到战场界面时因为控制台的显示效果不能达到预期，无奈之下只得放弃，改用UWP应用。UWP应用的很多控件非常适合卡牌游戏使用，故最终决定用UWP进行开发。

1. **程序的主要功能**
2. 卡牌展示功能：在CardsPage中查看已经写入程序中的所有随从牌以及法术牌，并且通过左键点击卡牌查看他们的详细信息（消耗，攻击力，生命值，名称，分类，描述，特效）
3. 卡牌拓展功能：通过向解决方案的Resouce/CardData文件夹中添加资源文件，实现向程序中添加卡牌的功能。
4. 卡组展示功能：在CardsPage最右侧显示的是玩家的卡组，玩家可以通过单击这些卡组来查看卡组中包含的卡牌。
5. 卡组的添加，删除功能：通过CardsPage最右侧的New Deck按钮，玩家可以新建一个卡组，然后右击卡牌，将卡牌添加进卡组，再点击Creat来完成卡组创建。在卡组详情页，玩家可以通过点击Delete按钮来删除该卡组。
6. 特效功能：在战斗中，法术牌和部分随从牌具有特效。特效分为目标，条件和效果。特效触发时通过判断条件是否满足来确认是否给予目标以效果。目标通常通过单击来选取（选取目标时边框会变绿）。
7. 特效拓展功能：通过添加EffectResult类，EffectRequirement类以及EffectAimRequirement类的代码，实现效果的添加。
8. 战场：在战场中，左下方为手牌，手牌右边为英雄及其生命值，右下方为当前的消耗和回合结束按钮。中间为双方的随从。最上方中间是敌方英雄及其生命值。当敌方英雄的生命值<=0时获胜，玩家英雄的生命值<=0时失败。
9. 故事模式（未完成）：使用预设的卡组游玩一系列的剧情关卡，了解ZooFantasy的背景故事。
10. 挑战模式（未完成）：使用自创的卡组挑战各种Boss并不断冲击最高纪录。
11. 训练场：敌方随从不会攻击，用来测试卡组和卡牌。
12. **程序的运行平台**

IDE：Visual Studio Community 2017

运行环境：Windows 10 （.NET）

1. **系统总框架图**

****

1. **程序类的说明**

**BattlefieldData:**

Battlefield:存放战场数据，消耗值等等。

EnemyHeroData:存放敌方英雄数据并提供相关的函数。

PlayerData:主要用于存放手牌。

PlayerHeroData: 存放我方英雄数据并提供相关的函数。

**CardData:**

BaseCard:卡牌基类。

MinionCard:继承基类的随从牌类。

SpellCard:继承基类的法术牌类。

Card:既可以是随从牌也可以时法术牌的通用牌类，主要用于HandCard。

CardDataManager:卡牌辅助类，包含卡牌的文件读写函数和一些其他的操作函数。

Deck:卡组类，主体为List<MinionCard>和List<SpellCard>。

DeckManager:卡组辅助类，包含卡组的文件读写函数和一些其他的操作函数。

HandCardManager:手牌辅助类，包含抽牌函数，洗牌函数等。

**EffectData:**

Effect:效果类，主体为Aim，AimRequirement，Requirement，Result。

EffectAim:效果目标类，负责描述可以作为效果对象的有哪些目标。

EffectAimRequirement:效果目标要求类，负责检查施放法术的目标是否达到要求。

EffectAimRequirementCheck:触发效果目标要求类。

EffectRequirement:效果要求类，负责检查是否满足施放法术的要求。

EffectRequirementCheck: 触发效果要求类。

EffectResult:效果影响类，负责施放法术特效。

EffectResultDo:触发效果影响类。

**MinionData:**

Minion:随从类，一般通过MinionCard构造，保存了随从的攻击力，生命值等信息，提供随从的相关函数。  
MinionManager:随从辅助类，提供随从相关操作函数。

1. **模块分析**
2. **卡牌&卡组展示模块**

查看和编辑卡牌及卡组的模块，添加的卡组将保存在Decks文件夹以便战斗模块使用。

1. **文件读写模块**

卡牌，卡组信息的文件读写，由后端代码完成（主要由CardDataManager和DeckManager类完成）。

1. **战斗模块**

由BattlefieldPage呈现的战斗模块，其操作大多数使用拖拽完成，方便直观。

1. **比较有特色的函数**

private void CastSpell\_Drop(object sender, DragEventArgs e) //拖拽施放法术（鼠标松开时）

{

MainGrid.DragOver -= CastSpell\_DragOver;

MainGrid.Drop -= CastSpell\_Drop;

MainGrid.DataContext = new MinionCard();

mainBattlefield.CastPlayerSpell(draggedSpellCard);

EffectResultDo.DoResults(PlayerHeroGrid, EnemyHeroGrid, PlayerMinionPanel, EnemyMinionPanel, mainBattlefield, (sender as Grid).DataContext, doneEffect);

mainBattlefield.Player.HandCard.Remove(draggedCard);

UpdateHandCardGridView(HandCardGridView);

UpdateAllMinionGrid();

UpdateHerosHealth();

UpdateCostTextBlock();

CollapseAimForSpell();

}

private void EnemyHeroGrid\_Drop(object sender, DragEventArgs e) //拖拽随从攻击（鼠标松开时）

{

if (draggingType == "PlayerMinion")

{

MinionManager.MinionAttack(draggedGrid, (sender as Grid));

UpdateAllMinionGrid();

UpdateHerosHealth();

}

}

1. **存在的不足和对策**
2. 拓展性不能很好的实施，在添加资源文件的过程中常常因为打错字或者忘记特效名称和参数格式而出现卡牌错误。 对策：以后将在程序中添加卡牌编辑，通过程序来编辑卡牌就不容易出错。
3. 目前游戏内容太少，卡牌只有数十张，法术牌只有三张。剧情和挑战模式未开发。 对策：将会陆续添加卡牌，剧情模式和挑战模式也将逐步开发。
4. 代码繁琐，BattlefieldPage的后台代码包含的函数过多导致修改困难，逻辑不明显。
5. 剧情模式仍是一个很大的挑战。即使是卡牌游戏，剧情是否吸引人也是游戏是否成功的一个重要因素。
6. **程序源代码**

见文件