# SET Game 设计说明

#### 综述

此为SET Game项目的设计说明文档，说明了SET Game项目开发中使用的设计框架、概念、代码结构与组成等。其主要目的在于对代码阅读和审查者进行导引，提高代码可读性。

特别指出，本文并非项目设计文档，仅为项目迭代完成时的设计说明文档。

项目完整代码位于<https://github.com/curno/SetCardGame/tree/master>

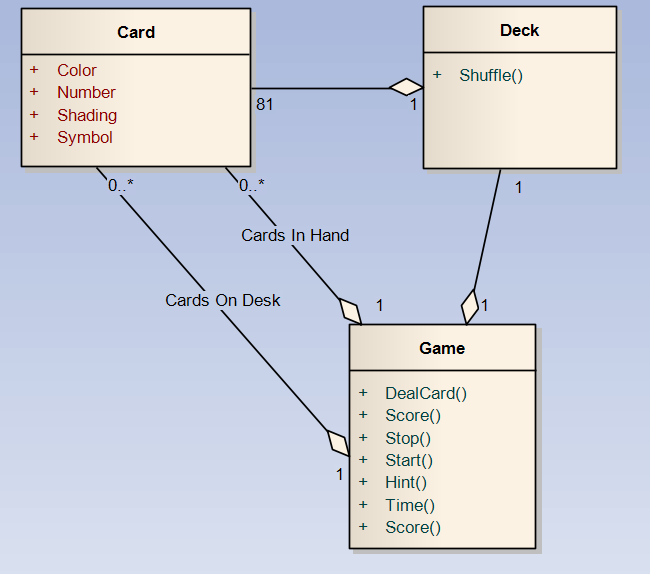
#### 总体设计

根据需求，本项目使用MFC作为窗口基本框架库进行开发。为了获得良好显示效果，使用OpenGL渲染游戏场景。使用Apache 2.2搭建私有服务器，使用python2.7 开发CGI程序，并使用MySQL5.5作数据存储。

下面几小节将针对项目开发中的多个方面进行逐一的设计介绍。总体来讲，项目设计采用MVC架构，将模型（逻辑设计）和显示的耦合降到最低。

#### 逻辑设计

逻辑设计主要负责设计游戏中的核心部件。类图如下：

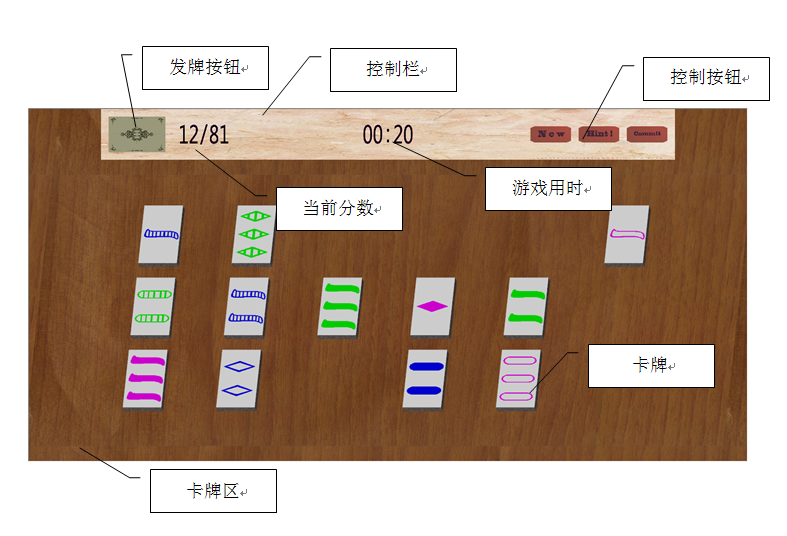


本部分由三个类组成，Card为一张卡牌，有自己的四个属性；Deck为一副牌，由81张卡牌组成，拥有洗牌方法；Game为一次游戏，包含一副卡牌，同时，Game还记录了当前游戏进度下桌面上的卡牌和尚未发出的卡牌，游戏计时和分数等，Game还负责游戏开始、结束、提示等逻辑。

本部分代码位于工程的 Include/Model 下。

#### 界面设计

界面设计说明如下图所示：



关于卡牌的图案，使用手工绘制的图片贴图到卡牌上。需要指出的是，虽然游戏中有3\*3\*3\*3共81中不同的卡牌，但是不同数目的卡牌可以通过多次贴图完成，不同颜色的卡牌可以通过图像重新着色完成，因此，需要准备的卡牌为3\*3共9张即可。

关于图像纹理重新着色的细节位于:

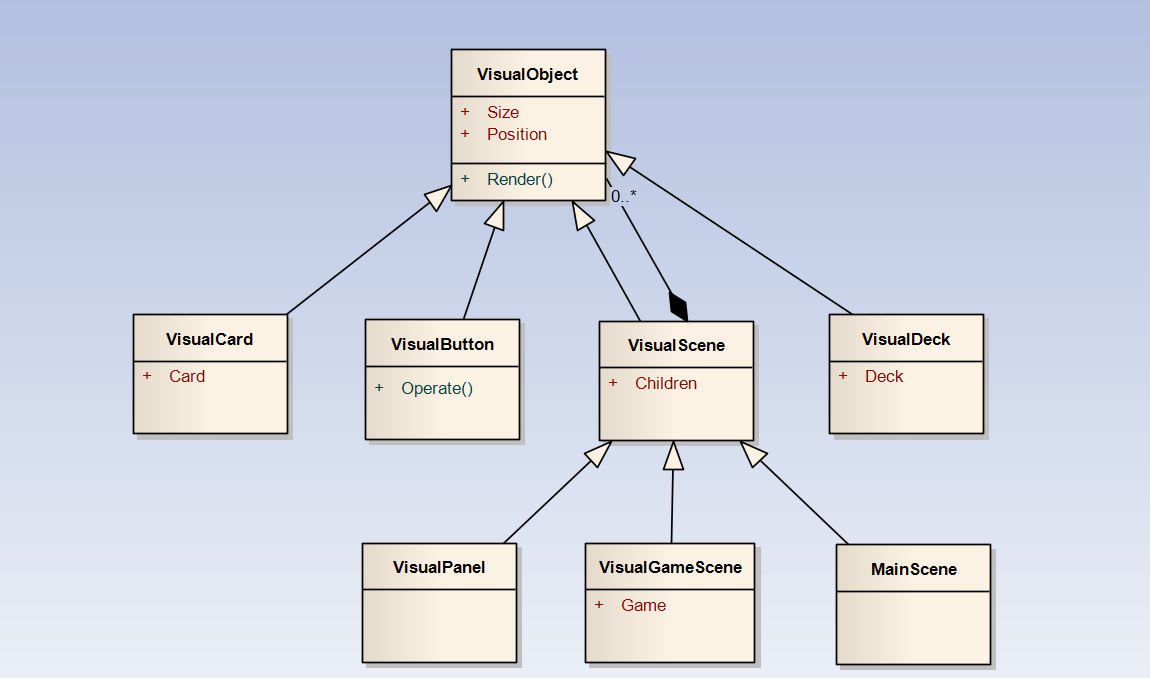
Include/Rendering/TextureManager.h

项目中使用的纹理文件及所有其他资源位于：

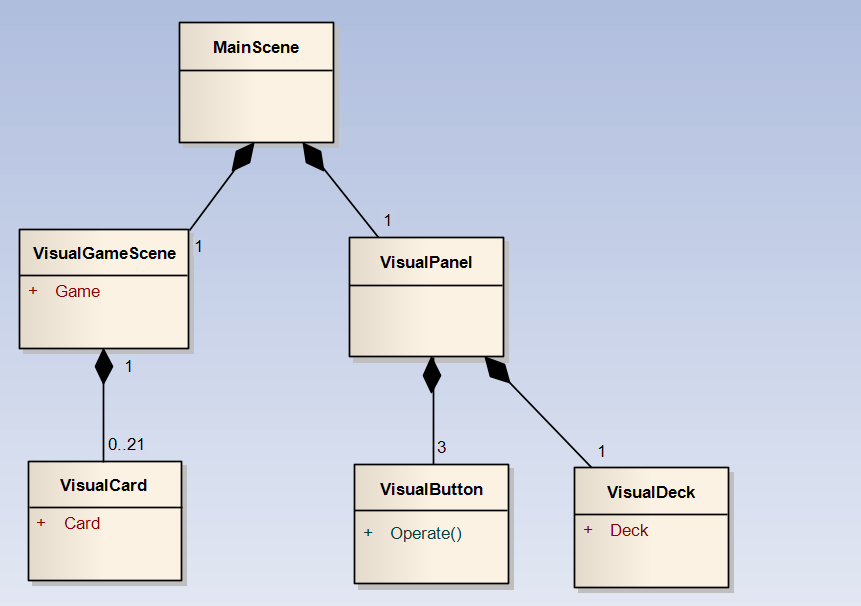
Res/

1. OpenGL显示框架

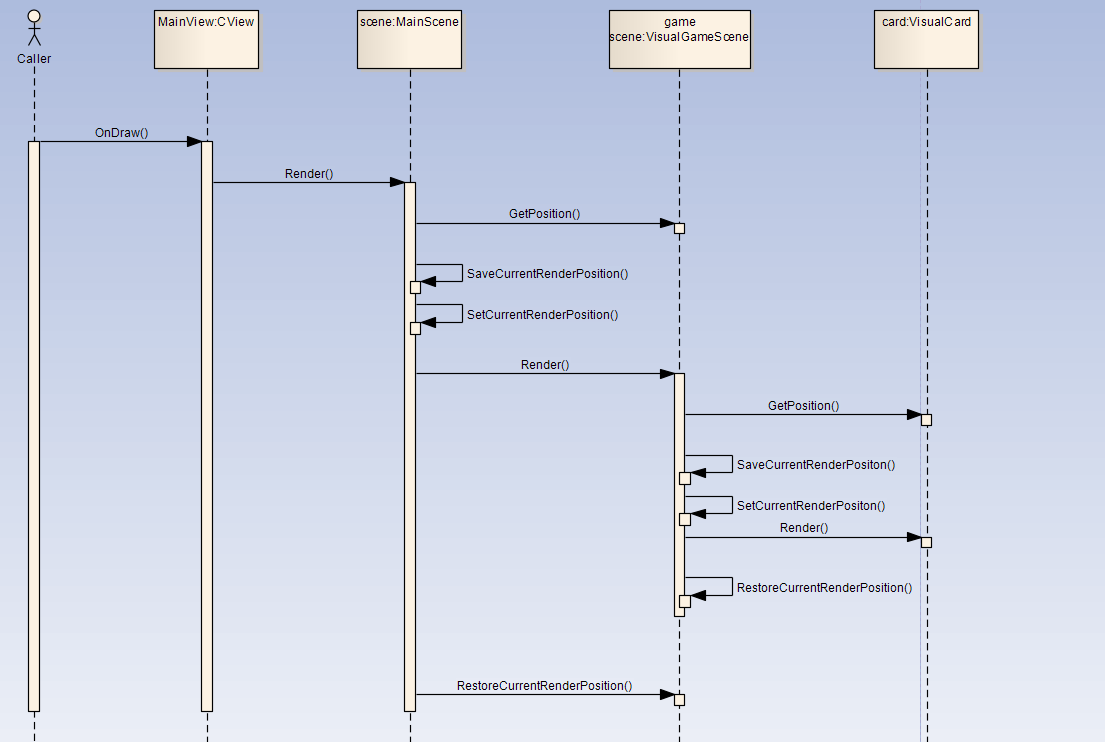
本项目使用OpenGL进行游戏渲染，为游戏中的物体如卡牌、按钮、面板等设计了专门的3D显示框架。相关类图如下：



整个结构利用组合设计模式。VisualObject是场景中所有可见物体的抽象基类，所有的可见类均有自己的位置和大小，位置和大小均为三维度量，物体也可以渲染自己。VisualCard和VisualButton、VisualDeck为具体的可见物体，分别为卡牌、按钮和牌堆。VisualScene是VisualObject的组合，通过逐个渲染每个子物体来渲染自己。VisualPanel、VisualGameScene和MainScene分别表示游戏控制面板、游戏主面板和整个游戏界面。上述各个类的包含关系如下：



下面的顺序图近似了渲染游戏场景中一个卡牌的过程：



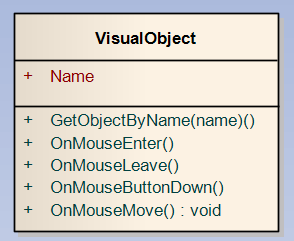
为了方便控制，并提高可扩展性，本游戏并非采用事件驱动，而采用Timer驱动渲染，由主Timer定期发送渲染消息（20ms），刷新界面。

OpenGL显示框架定义在：

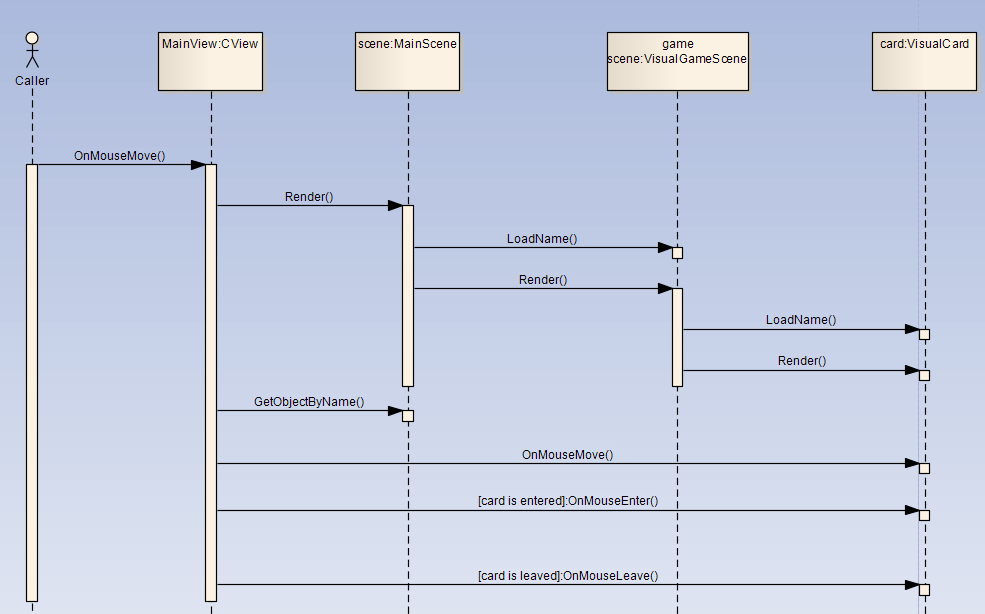
Include/Rendering/

#### OpenGL点选设计

为了完成鼠标选择、悬停和点击的效果，本项目使用OpenGL内置点选功能来进行鼠标击中测试。为此，将OpenGL的点选功能集成到上述的OpenGL框架中。具体来讲,为VisualObject增加几个成员：



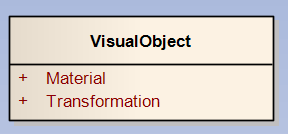
同时，将OpenGL加载名称的调用整合到渲染过程中，下面的顺序图简要说明了这一点：



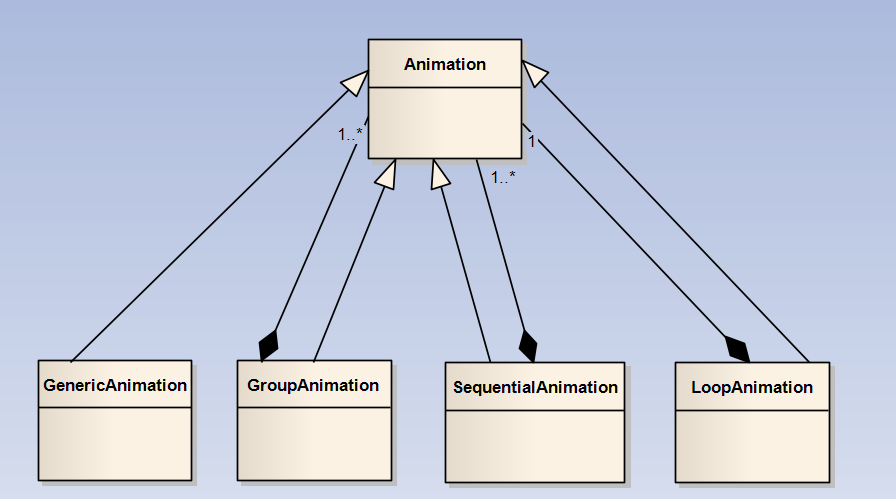
#### 动画框架设计

本项目为游戏中的效果设计了一个简单的动画框架。

首先，为VisualObject添加用于显示的若干属性：



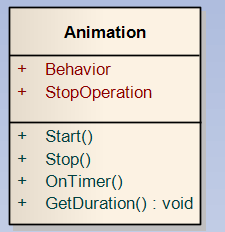
分别为物体的材质和物体的形变。程序中可以通过物体材质的变换、物体的平移、放缩、选择等形变来完成各式各样的动画效果。动画的类结构如下图：



Animation为动画抽象基类，GenericAnimation为泛型动画类，可以接受任何满足特定接口的对象作为其动画行为。项目的Include/Animation/VisualObjectAnimations.h中定义了一系列可以用来构造GenericAnimation的类型，有Rotate、Transform、MoveVisualObject、Shake、Blink等，这些对象通过改变VisualObject的特定属性来完成动画。

GroupAnimation和SequentialAnimation为组合动画类（组合设计模式），分别可以同时执行若干动画或者顺序执行若干动画。LoopAnimation则可以循环执行一个动画，为装饰类（装饰设计模式）。

一个Animation有如下主要的接口：



动画可以开始、结束、获取持续时间，OnTimer()则在每一帧进行调用，进行相应的动画行为。Behavior和StopOperation需要进一步说明。

StopOperation为一个结束行为，即在动画结束时调用的一段代码，可以接受任何可调用的函数子（函数、函数对象、Lambda表达式等）作为行为。为了不在Animation中引入泛型参数，定义了Operation基类和GenericOperation泛型类，将泛型参数引入到GenericOperations中。

Behavior定义了动画的运行行为（或者说轨迹），在动画运行的时候，动画从起始值到中止值不一定是线性变化的，如果需要完成复杂的动画曲线，如正弦、余弦或者震荡等动画运行的行为，可以通过定义特殊的Behavior来定制动画。同StopOperation一样，为了不再Animation中引入泛型参数，定义了AnimationBehavior和GenericAnimationBehavior。

最后，为了让所有的动画能够统一运行并被管理起来，定义全局单例AnimationManager，被启动的所有动画均自动注册到动画管理器，被停止的动画则从动画管理器注销。在每一个渲染周期，动画管理器负责调用所有注册的动画对象的OnTimer函数，以此修改VisualObject的相应属性。属性的变化会立刻在下一次渲染时显示出来，随着逐帧渲染，动画效果得以呈现。

动画框架定义在：

Include/Animation

#### 数据库、服务器设计

数据库使用MySQL5.5，本项目定义两个表，分别是

|  |  |
| --- | --- |
| USER | |
| ID | Integer, Primary Key |
| NAME | Char(20) |

|  |  |
| --- | --- |
| Score | |
| ID | Integer, Primary Key |
| USER\_ID | Integer |
| SCORE | Integer |
| ELAPSED\_TIME | Integer |

USER表记录用户名，SCORE表记录用户ID，分数和计时的记录。

服务器使用Python编写CGI程序，使用Mysql-python包连接数据库。

#### 杂项设计

另外程序还有一些杂项如下：

项目中使用的几何对象、秒表装置、随机数生成器等定义在：

Include/Utils/

项目中出牌带有音效，功能定义在：

Include/Sound/

项目中与服务器通信功能定义在：

Include/Web/