钟添芸 3180103009

设计思路

MVVM框架搭建及App层实现

首先学习老师的demo已经网络上已有的MVVM模型代码

最后在各层初步完成后在App层完成绑定工作

Common层基础实现

按照向量类→线段类→多边形类→刚体类的设计顺序对Common层的数学物理模型进行建立并提供稳定可靠的接口，完成除了碰撞以外的大部分功能

Command及Parameter实现

根据用户需求设计命令（Command），再设计相应的命令参数（Parameter）

ViewModel层实现

根据Model层提供的接口与基本逻辑设计程序的业务逻辑，并与View层进行沟通（此处并不是好做法）

解决方案

MVVM框架搭建及App层实现

以下均在App/App.h与App/App.cpp中实现

1. 在app层的构造函数中创建各层实体并传指针到app层
2. 将ViewModel与Model、View、WIndows层绑定，并将View与Windows层绑定（由于使用了控件）
3. 在app层将命令的set与get函数进行绑定
4. 在App::stratWorld()中进行窗口初始化及UI逻辑展示

Common层基础实现

以下均在Common/CommonBase.h与Common/CommonBase.cpp中实现

1. **向量类Vec设计**

   构造析构函数

   运算符重载（向量加减、数乘、内积、外积）

   get/set方法（包括x-y式与极坐标式）

   绕轴旋转函数

2. **线段类Segment设计**

   构造析构函数（参数Vec）

   get/set方法

3. **线段与向量交互计算**

点到点距离计算

点到线段（而非直线）距离计算

三点张角计算

4. **多边形类Poly设计**

构造析构函数（参数vector)

get/set方法

5. **匀质多边形面积、质心及刚体转动惯量计算**

多边形面积计算：通过拆分为多个三角形求和实现

多边形质心计算：通过拆分为多个三角形并以面积加权平均实现

刚体转动惯量计算：阅读文献后，通过拆分为多个三角形求解各转动惯量并结合平行轴定律实现

6. **几何接触关系计算**

点与多边形接触情况判断：通过计算张角和实现

多边形与多边形接触情况判断：通过点的情形遍历延拓实现

获得多边形与多边形接触点：返回遍历时有接触的点

获得多边形与多边形接触边：返回接触点最近的边

Command及Parameter实现

以下均在Common/Common.h、Command/XXX.h、Command/XXX.cpp以及Common/Parameter.h中实现

actionMode枚举常量

在Common层定义，有CreateMode,AdjustMode,DeleteMode三种

Parameter设计

类名	基类名	数据	方法	功能
Parameter	(None)	(None)	(None)	基类
IntParameter	Parameter	int	getInt()	存储整形数据
VecParameter	Parameter	Vec	getVec()	存储向量数据
ShapeParameter	Parameter	id, actionMode	getId(), getActionMode()	存储形状数据，也是基类
PolyParameter	ShapeParameter	id, actionMode, Poly	getId(), getActionMode(), getPoly()	存储多边形数据
RigidBodyParameter	ShapeParameter	id, actionMode, RigidBody	getId(), getActionMode(), getRigidBody()	存储刚体数据

Command设计

类名	基类名	命令参数类型	功能
Command	(None)	(None)	基类
AddForceFieldDataCommand	Command	VecParameter	向Model添加力场
ClearUserRigidBodyCommand	Command	(None)	清除用户刚体数据
RefreshViewCommand	Command	(None)	触发图像更新
SimulateTimeFlyDataCommand	Command	IntParameter	触发物理时间轴模拟
UpdatePolyViewCommand	Command	PolyParameter	向View发出刚体更新命令
UpdateRigidBodyDataCommand	Command	RigidBodyParameter	向Model发出刚体更新命令

ViewModel层实现

以下均在ViewModel/ViewModel.h与ViewModel/ViewModel.cpp中实现

1. **ViewModel构造函数**

   在构造函数中对各种命令的实体进行空间申请，并使用共享指针管理

   值得一提的是，命令中各参数的空间申请是在命令发出时申请的

2. **bind绑定函数**

   通过该函数可以实现ViewModel到Model、View、WIndows的绑定

3. **getCommand函数**

   该类函数提供ViewModel与其他层的Command操作对象绑定的接口，具体形式与功能类似于“Command设计”中的表格

4. **execCommand函数**

   该类函数提供ViewModel的Command操作对象的实际执行内容，具体形式与功能类似于“Command设计”中的表格

运行效果图

由于本人负责内容均是框架结构或者底层代码，因此难以有一个较好的运行展示方式，因而本处不展示运行效果图，程序的最终运行效果图可以参照总报告

课程心得体会

首先，要有良好的工程结果果然是需要一个优秀的开发模式。我们在理解MVVM模式时产生了一定的误解，使得整体框架偏向于MVC模式。因此也确实在开发的过程中发生了小部分的耦合，拖延了一些开发时间。多学多看多想，一个好的开头可能就已经离成功不远了。

其次，程序设计课与现代工程开发确实存在一定代沟。比如git，jenkins等现代工具就不会在常规的程序设计课上被要求使用，而这实际上是会影响到学生对于现代开发框架的理解的。诚然，不能推进现代化的工具也有着各种各样的原因和阻力，但是推进一定是有好处的。因此，这一点也是本课程所教会我的重要内容。

最后，本课程确实并不太适合大一学生选修。我们小组的前半段时间基本都在恶补oop知识，这可能也是我们理解MVVM模式不够充分的原因之一吧。

课程建议

- 可以在课程介绍中加上比较详尽的预修要求（尤其是oop）
- 可以加入一些签到机制（点名可能有些困难，但是签到比较容易被接受），鼓励同学们来机房现场合作