电子科技大学 计算机 学院

实 验 报 告

课程名称 C++程序设计

实验题目 精灵游戏

难度级别 4

提交时间 2020.12.21

姓　　名 岳子豪

学　　号 2018051404015

1 软件说明

1.1 游戏使用说明

（1） 背景介绍：

（注：本游戏设定纯属虚构、娱乐，无吹捧或抹黑之意。）

a) 四川大学是一所历史悠久、实力雄厚的中国知名高水平大学，目前在我国的四川省排名第 100；

b) 它可以合并其它学校，比如成都科技大学和华西医科大学，提升自己的实力，与此同时排名也会得到提升；

c) 遇到不合适的学校，如果它尝试去合并，则会遭遇很多麻烦，比如电子科大成都学院；

d) 如果遇到强劲对手电子科技大学，则会受到挫败，多次挫败之后，就会一蹶不振。

四川大学有着宏伟的目标，希望能成为四川第一高校，因此希望尽可能多地合并学校，使自己的排名成为第 1。如果四川大学能在一蹶不振之前，顺利登顶四川高校榜，则可喜可贺；否则，四川大学将依然有很长的一段路要走。

（2） 基本功能：

本游戏的基本功能类似于大鱼吃小鱼，用户通过方向键控制角色移动，可以与界面中的其它对象进行互动，如吃掉、被吃掉、道具效果等，相应的得分和生命值的变化会实时显示在页面中。游戏能正常运行，并根据用户游戏情况给出游戏结束时的提示。

（3） 游戏规则：

a) 游戏运行即开始，可直接进行操控；

b) 游戏开始时，用户有 5 次机会（生命值），初始排名 100；

c) 用户通过计算机的四个方向键“↑”、“↓”、“←”、“→”可以控制角色“四川大学”在画面中按照相应方向移动；

d) 游戏刚开始时画面中会不断出现“成都科技大学”，“四川大学”接触到“成都科技大学”时将会吃掉它，同时排名前进 3 名；

e) 当排名进入前 85 名时，画面中不断出现“华西医科大学”，它们会主动躲闪。“四川大学”接触到“华西医科大学”时将会吃掉它，同时排名前进 5 名；

f) 当排名进入前 70 名时，会出现 1 个“电子科技大学”，并开始随机出现道具。“四川大学”接触到“电子科技大学”时，将会被吃掉并随机复活在另一个地方，同时生命值减 1；

g) 共有 4 种道具：

“123 周年校庆”：令人振奋的校庆可以让“四川大学”恢复 1 个生命值；

“电子科大成都学院”：“四川大学”吞并失败元气大伤，停滞 3s；

“science”：在 Science 上发表论文，可以使移动速度变为原来的 2 倍，持续时间 5s；

“nature”：隔壁成电发表 nature 封面论文，移动速度变为原来的 1/2，持续时间 3s；

(“science”和“nature”同时生效时效果叠加)

h) 若“四川大学”在生命值降为 0 之前排名上升到第 1，则游戏结束，玩家获得成功；若游戏过程中生命值降为 0，则游戏结束，玩家失败。

1.2 源码编译说明

可以通过直接运行源代码文件中的 SCU 大冒险.exe 文件；也可以在 VS 中打开项目并运行；如果依然不行，可以新建项目、拷贝代码重新编译生成可执行文件运行该游戏。接

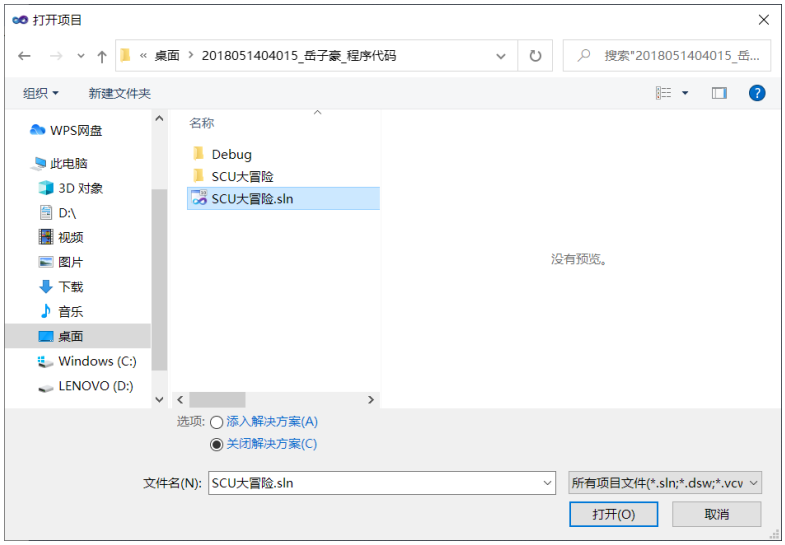
下来对三种方式进行详细介绍。

1. 直接运行.exe 文件

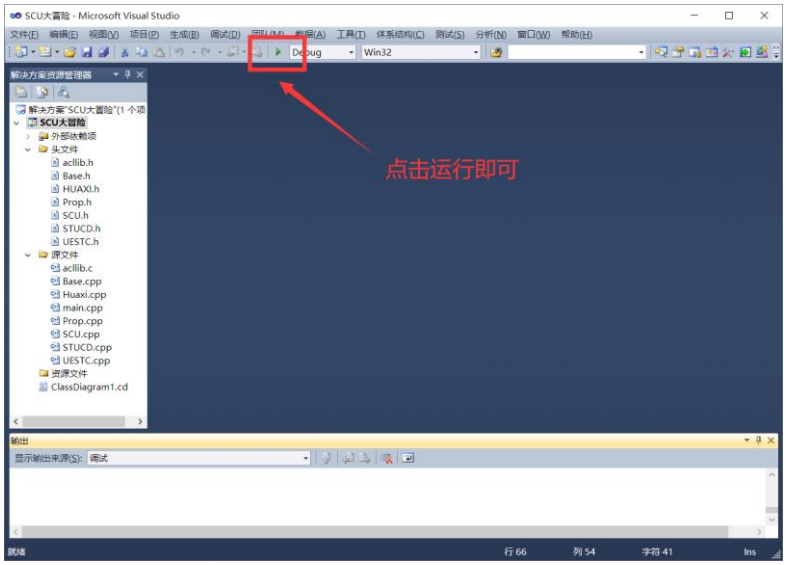
打开“2018051404015_岳子豪\2018051404015_岳子豪_程序代码\SCU 大冒险”路径下的“SCU 大冒险.exe”。（这种方法貌似不太行，我发给三个同学他们都不能在自己的电脑上成功运行……非常抱歉）

2. 打开项目重新编译

在 Visual Studio 2010 中选择“文件 => 打开 => 项目/解决方案”，在弹出的窗口中选择“2018051404015_岳子豪\2018051404015_岳子豪_程序代码”路径下的“SCU 大冒险.sln”，打开。

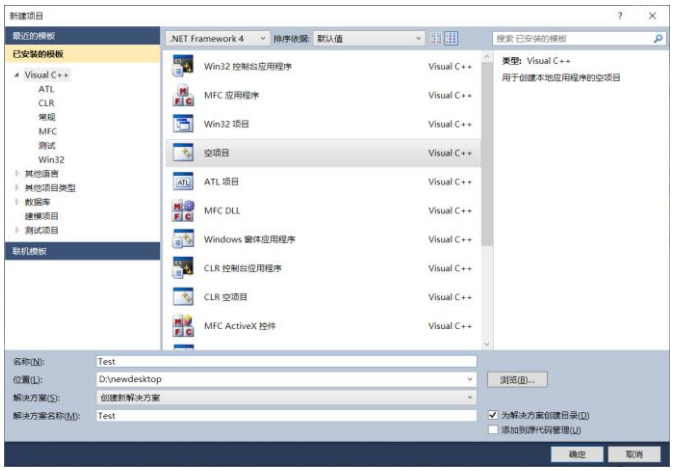


打开后，点击启动调试即可运行该项目。

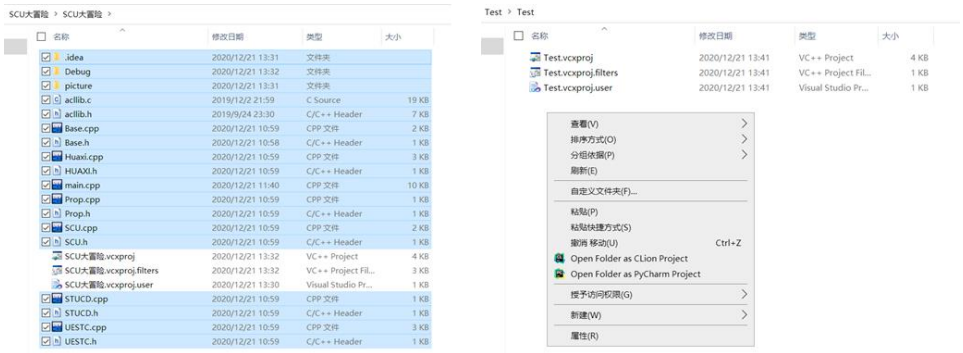


3. 拷贝代码重建项目

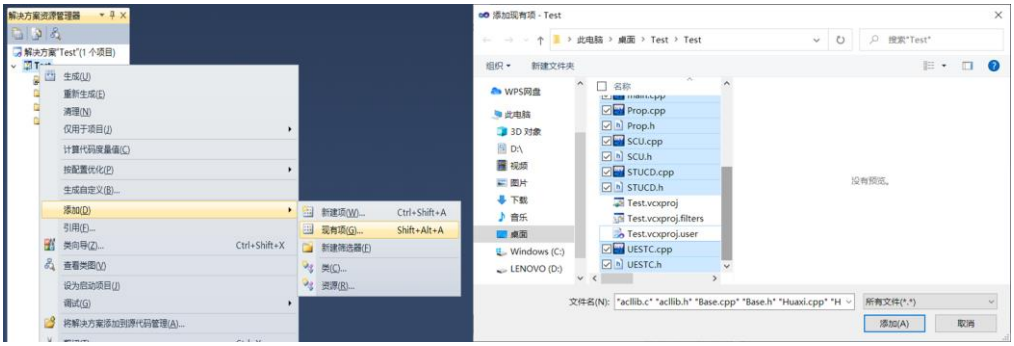
a) 使用 Microsoft Visual Studio 新建一个 C++空项目 Test，并打开项目文件夹；



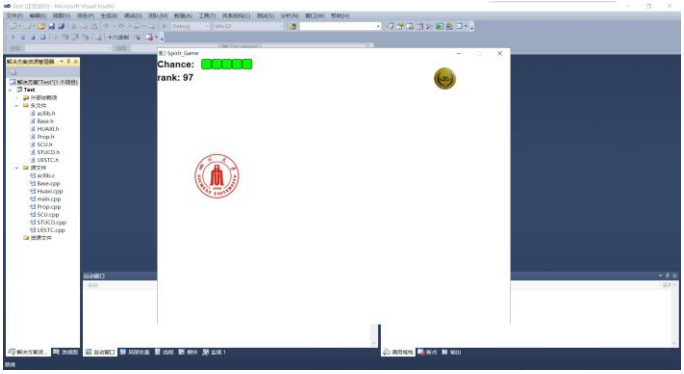
b) 将“2018051404015_岳子豪\2018051404015_岳子豪_程序代码\SCU 大冒险”路径下的所有文件夹、.cpp 文件、.h 文件拷贝到“Test\Test”，如图。



c) 在 VS 中为 Test 项目添加现有项，并选择刚刚拷贝的文件，点击添加，如图。



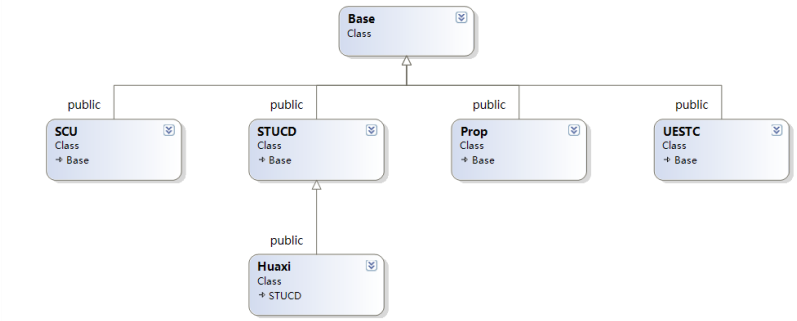
d) 添加完成后，点击运行即可生成.exe 可执行文件，游戏窗口将自动弹出。



e) 游戏结束后，再次点击运行可重新生成，并再次进行游戏。

2 系统设计

本实验主要利用 C++的封装、继承与多态思想实现游戏主要功能(即各类精灵的创建)，类视图如下。



首先设计了一个基类 Base，包括获取位置、大小、移动、碰撞检测以及绘制等基本功能，以及窗口类用于保存图形界面的窗口数据。包含 Base、collision、getheight、getwidth、getx、gety、move、paint 等方法、height、width、iconimage、speedx、speedy 等字段。类详细信息如下：

名称	类型	修饰符
方法		
~Base		public
Base		public
Base		public
collision	bool	public
getheight	int	public
getwidth	int	public
getx	int	public
gety	int	public
move	void	public
move	void	public
paint	void	public
字段		
heigth	int	protected
iconimage	string	protected
img	ACL_Image	protected
speedx	int	protected
speedy	int	protected
width	int	protected
Winsize	WINSIZE	protected
x	int	protected
y	int	protected

SCU 类由 Base 派生，继承了基类的属性，并增加了设置出现位置、键盘控制移动、生命值、速度、是否停滞等属性，包含 SCU、getrank、setrank、getchance、move、addchance、reducechance 等方法和 chance、isbreak 等字段。类详细信息如下：

名称	类型	修饰符	摘要
方法			
~SCU		public	
addchance	void	public	增加机会（生命值）
getbreak	int	public	是否停滞
getchance	int	public	
getrank	int	public	
move	void	public	
move	void	public	
reducechance	void	public	减少机会（生命值）
SCU		public	
SCU		public	
setbreak	void	public	设置停滞
setrank	void	public	排名改变（增量为负）
setspeed	void	public	速度提升/降低multiple倍
setxy	void	public	设置位置
字段			
chance	int	protected	
get_rank	int	protected	当前排名
isbreak	int	protected	是否停滞标志

STUCD 类由 Base 派生，继承了基类的属性，并增加了移动、增益、生存状态等功能和属性，包含 STUCD、setlife、islife、move 等方法和 life、rank 等字段。类详细信息如下：

名称	类型	修饰符	摘要
方法			
~STUCD		public	
getrank	int	public	
islife	bool	public	判断是否存活
move	void	public	
move	void	public	
setlife	void	public	
STUCD		public	
STUCD		public	
字段			
life	int	protected	存活标志
rank	int	protected	增益

Huaxi 类由 STUCD 类派生，继承了 STUCD 类的属性，包含基类和 STUCD 类中新增的属性，并增加了判断与用户精灵 SCU 的距离和自动变速躲避等功能，包含 Huaxi、move、distance 等方法和 is_speedup 和 scu_distance 等字段。类详细信息如下：

名称	类型	修饰符	摘要
方法			
~Huaxi		public	
distance	void	protected	计算与用户精灵的距离
Huaxi		public	
Huaxi		public	
move	void	public	
move	void	public	
move	void	public	
overboundary	void	protected	越界处理
字段			
is_speedup	int	protected	是否加速
scu_distance	int	protected	与用户（川大）距离

Prop 类（道具）由 Base 派生，继承了基类的属性，并增加了移动、是否存在判断等功能，包含 Prop、getexist、move 等方法和 isexist 等字段。类详细信息如下：

名称	类型	修饰符	摘要
方法			
~Prop		public	
getexist	int	public	
move	void	public	
move	void	public	
Prop		public	
Prop		public	
setexist	void	public	
字段			
isexist	int	protected	是否存在

UESTC 类由 Base 派生，继承了基类的属性，并增加了移动、判断距离、追捕等功能，包含 UESTC、move、islife 等方法和 scu_distance、is_speedup 等字段。类详细信息如下：

名称	类型	修饰符	摘要
方法			
~UESTC		public	
distance	void	protected	计算距离
islife	int	public	
move	void	public	
move	void	public	
move	void	public	
overboundary	void	protected	越界处理
setlife	void	public	
UESTC		public	
UESTC		public	
字段			
is_speedup	int	protected	追捕速度
life	int	protected	是否出现
scu_distance	double	protected	与用户（川大）距离

3 程序实现

在功能的实现中，统一采用函数思想，如 move()函数控制精灵的移动，distance()函数用来计算其他精灵与用户精灵 SCU 的距离等。由于篇幅有限，本部分内容仅在每一个类中选择一个函数进行详细介绍，包括 Base 类的碰撞检测函数 Base::collision(Base *base)、Huaxi 类的边缘检测函数 Huaxi::overboundary()、Prop 类的道具类构造函数 Prop::Prop(Prop &prop)、SCU 类的得分管理函数 SCU::setrank(int addrank)、STUCD 类的移动函数 STUCD::move()、UESTC 类的距离判断函数 UESTC::distance(SCU &scu)等。

函数通过两层嵌套的条件判断，覆盖了两个对象从不同方向上碰撞的四种可能，通过边缘检测判断是否碰撞。

以基类的碰撞检测函数为例，实现代码如下：

```
bool Base::collision(Base *base)
{
     if (x < base->x && x + width > base->x){
              if (y < base->y && y + heigth > base->y) return true;
              if (y > base->y && y < base->y + base->heigth) return true;
     }
     if (x > base->x && x < base->x + base->width){
              if (y < base->y && y + heigth > base->y) return true;
              if (y > base->y && y < base->y + base->heigth) return true;
     }
     return false;
}
```

部分对象移动的过程中，需要对游戏界面的边缘进行检测，保证始终在界面内移动，因此，需要设计边缘检测函数 overboundary()进行判断。而该功能是部分类有、部分类不需要的，如道具类随机飘过，需要用户自己抓住机会去捕捉，并不需要边缘越界检测，而 Huaxi 等类的移动则需要限制在界面内。该函数的设计需要获取对象位置和当前移动方向，通过四个条件判断，覆盖了上下左右四种越界情况。本实验选取的越界处理方式为循环传送，即左出右进，上出下进。以左边界为例，当对象位置处于左边界之外，且依然有左向的速度分量时，则重新设置其位置，通过界面窗口尺寸和当前位置计算出新位置，并设置位置属性，实现循环传送。

以 Huaxi 类为例，边缘检测函数的代码如下：

```
void Huaxi::overboundary()
{
     if (x + width < 0 && speedx < 0){
              x = Winsize.getwidth();
              y = Winsize.getheight() - heigth - y;
     }
     if (x > Winsize.getwidth() && speedx > 0){
              x = -width;
              y = Winsize.getheight() - heigth - y;
     }
     if (y + heigth < 0 && speedy < 0){
              y = Winsize.getheight();
              x = Winsize.getwidth() - width - x;
     }
     if (y > Winsize.getheight() && speedy > 0){
              y = -heigth;
              x = Winsize.getwidth() - width - x;
     }
}
```

游戏中一共涉及四种道具，种类较多，因此需要注意构造函数的设计。函数针对不同的应用场景，提供了两种构造函数的调用方式，可以赋值调用，也可以通过类指针传入，并提供了一个函数用于外部读取道具是否存在的标志变量 isexist。

对应代码如下：

```
    Prop::Prop(int x, int y, int speedx, int speedy, int width, int heigth, WINSIZE
winsize, string iconimage)
    :Base(x,y,speedx,speedy,width,heigth,winsize,iconimage)
    {
         isexist = 0;
    }
    Prop::Prop(Prop &prop)
    :Base(prop.x,prop.y,prop.speedx,prop.speedy,prop.width,prop.heigth,prop.Wins
ize,prop.iconimage)
    {
         this->isexist = prop.isexist;
    }
    int Prop::getexist()
    {
         return isexist;
    }
    void Prop::setexist(int exist)
    {
         this->isexist = exist;
         if (exist == 1)
         {
                  this->x = rand() % (Winsize.getwidth() - width);
                  this->y = 0 - heigth;
         }
    }
```

在游戏运行过程中，需要对用户的得分（即 SCU 排名）进行实时的更新与反馈。为了实现此功能，故设计函数 SCU::setrank(int addrank)，通过外部传入的 addrank（即用户控制 SCU 吃掉一个 STUCD 或 Huaxi 之后带来的排名增益，为负数），来更新当前 SCU 对象的排名属性 get_rank。由于排名的数值最低为 1，因此需要判断加上增以后的新排名是否大于 1，若不大于 1，则需强制设定为 1，否则排名有可能变为 0 或负数。

```
    void SCU::setrank(int addrank)
    {
         if(getrank()+addrank>1) get_rank += addrank;
         else get_rank=1;
    }
```

STUCD 类是 Huaxi 的父类，具有一定的普遍性。移动是 STUCD 类及其子类 Huaxi 类的一个必不可少的功能，并且调用及其频繁，设计 move()函数如下，首先根据对象的 life 判断是否还存活，如果被吃掉则不再继续运动，否根据位置判断是否需要将速度反向，规则为纵横方向任意方向越界，则将该方向速度取反。之后是运动过程，每次调用均使 x 和 y 的位置改变一个 speed 大小，通过离散的变化模拟运动。

对应代码如下：

```
void STUCD::move()
 {
     if (this->life == 0) return;
     if (x < 0 || x + width > Winsize.getwidth()) speedx *= -1;
     if (y < 0 || y + heigth > Winsize.getheight()) speedy *= -1;
     x = x + speedx;
     y = y + speedy;
}
```

UESTC 类由于有自动追捕功能的需求，因此 UESTC 类对象需要时刻对自己与 SCU 类对象的距离进行计算，程序才能根据计算值进行后续的处理。距离采用两点间的直线距离计算法，计算两个对象的正中心的距离，并写入 UESTC 类的对象的 scu_distance 里，以便程序的其它模块调用。

```
void UESTC::distance(SCU &scu)
{
     double dx, dy;
     dx = pow(x + width / 2.0 - (scu.getx() + scu.getwidth() / 2.0), 2);
     dy = pow(y + heigth / 2.0 - (scu.gety() + scu.getheight() / 2.0), 2);
     this->scu_distance = sqrt(dx + dy);
}
```

项目中还使用了状态检测触发函数，用于判断用户进行到游戏的哪一阶段，以及对游戏中的各类事件进行统一的检测与处理。由于本函数内容较多，选取与游戏进度管理有关的代码段进行详细描述。根据游戏设置，当排名进入前 85 名时，开始出现 Huaxi 类精灵，条件判断中的 huaxi_mark 与 huaxinum 分别为出现个数和初始设置的个数，用于约束其数量，当 scu 的 get_rank 小于或等于 85 时，将 huaxi[huaxi_mark]的 life 属性置 1，创建 Huaxi 类对象，并在之后定时产生新的 huaxi。同理，当 get_rank 小于或等于 70 时，创建 UESTC 对象 uestc，并为创建 Prop 对象做好准备。与 huaxi 不同的是，uestc 只有一个，因此不需要重复创建。

对应代码如下：

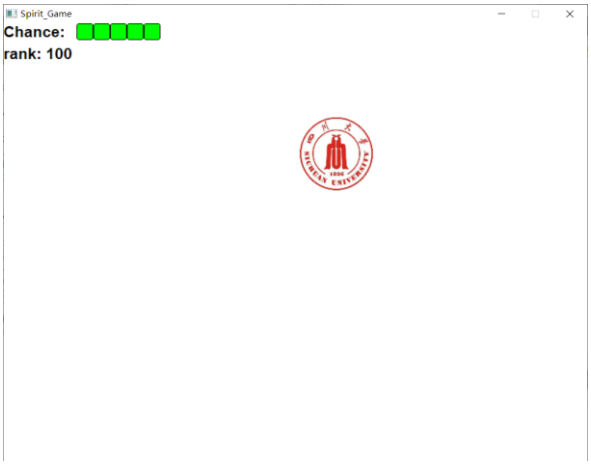
```
//排名进入前 85，开始出现华西
if (scu->getrank() <=85 && huaxi_mark < huaxinum){
        huaxi[huaxi_mark]->setlife(1);
        if (huaxi_mark == 0) startTimer(2, 100);
        huaxi_mark++;
}
//排名进入前 70，开始出现 UESTC 和道具
if (scu->getrank() <=70)
        if(uestc_mark == 0){
                uestc_mark = 1;
                uestc->setlife(1);
                startTimer(3, 150);
        }
        if(prop_mark == 0){
                startTimer(4, 5000);
```

```
                startTimer(5, 100);
                prop_mark = 1;
        }
if (stucd_mark == stucdnum && huaxi_mark == huaxinum && uestc_mark == 1 && prop_mark
== 1) cancelTimer(0);
```

此外，除了上述功能函数，还有注册键盘处理函数等其它函数，用来实现包括接收用户键盘指令在内的功能，相应设计详见源代码，在此不一一赘述。

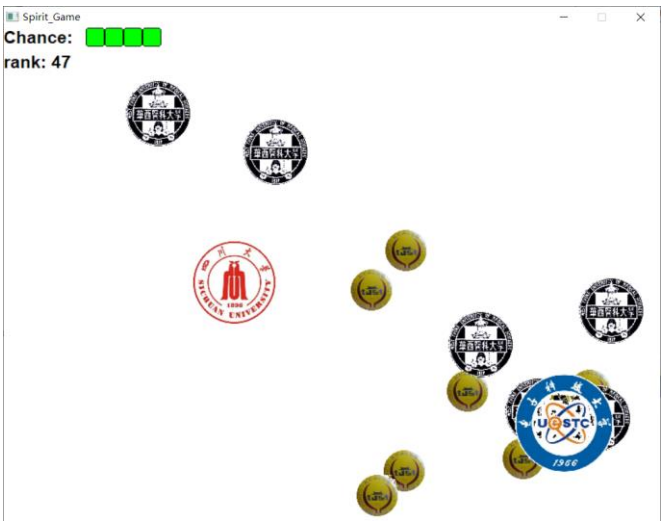
4 测试报告

1. 功能点 1：启动游戏
   测试结果：游戏正常启动，界面如图所示：



2. 功能点 2：方向键控制 scu
   测试结果：分别按下四个方向键，scu 均能向对应方向移动，且支持长按连续移动，功能正常。
3. 功能点 3：控制 scu 接触 stucd
   测试结果：接触之后，stucd 消失，rank 减 3，功能正常。

4. 功能点 4：控制 scu 接触 huaxi
   测试结果：接触之后，huaxi 消失，rank 减 5，功能正常。



5. 功能点 5：控制 scu 接近 huaxi
   测试结果：接近之后，huaxi 躲闪，功能正常。

6. 功能点 6：控制 scu 捕获 science 道具
   测试结果：接触之后，science 道具消失，scu 移动速度加倍，持续 5s，功能正常。

7. 功能点 7：控制 scu 捕获 nature 道具

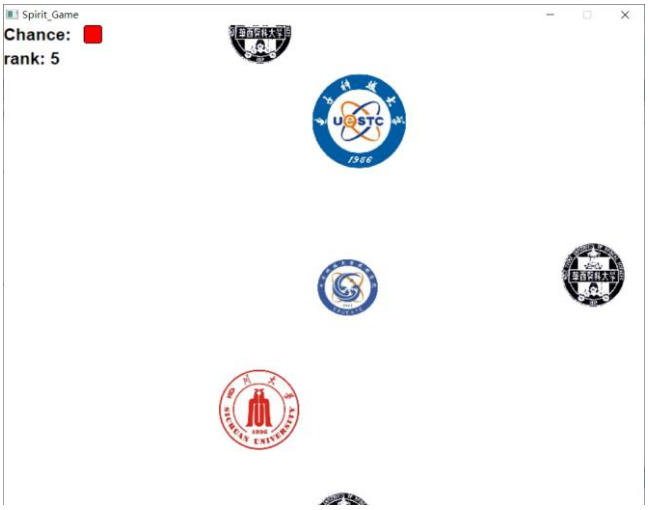
测试结果：接触之后，nature 道具消失，scu 移动速度减半，持续 3s，功能正常。

8. 功能点 8：控制 scu 接触 uestc

测试结果：接触之后，scu 消失并在另一个地方复活，Chance 数减 1，功能正常。

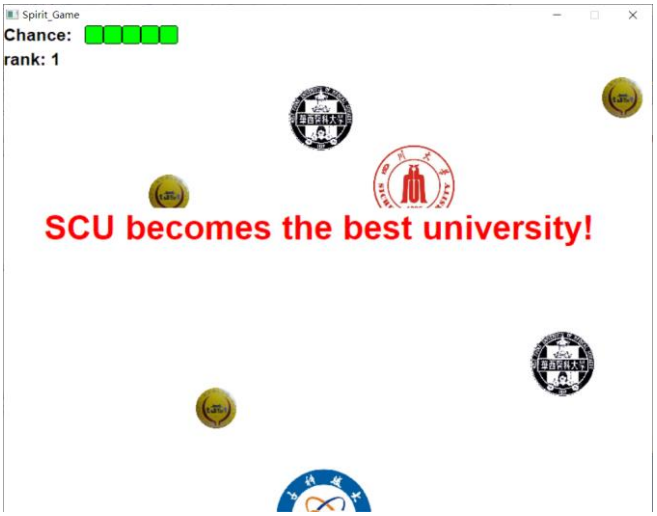
9. 功能点 9：控制 scu 接触 cdxy

测试结果：接触之后，cdxy 消失，用户无法通过键盘控制 scu，持续 3s，功能正常。



10. 功能点 10：控制 scu 接触 celebration
    测试结果：接触之后，celebration 消失，若 Chance 数小于 5，则增加 1，否则不增加。功能正常。

11. 功能点 11：排名为 1，游戏结束
    测试结果：游戏暂停，弹出结束语，如图。功能正常。



12. 功能点 12：生命值为 0，游戏结束
    测试结果：游戏暂停，弹出结束语，如图。功能正常。

经过上述测试，该实验的基本功能均可以较好实现，而新增的功能如道具、自动躲闪等也均正常工作。因此，该程序实现了预期功能。

5 实验心得

作为本科期间为数不多的几个数百行代码大实验之一，本次实验让我颇有心得，受益匪浅。从最开始的无从下手，到从最简单的功能开始着手，不断添砖加瓦，一点一点将所有基本功能全部实现，并在原基础之上增加了一些新的功能。再经过不断地调试、修改，寻找可能存在的 bug，完善细节，直到最后功能基本符合预期，整个过程虽然漫长而艰辛，但结果令人充满成就感。尤其是将能力范围内所有能解决的 bug 全部解决，合上电脑那一瞬间的轻松与愉悦，让我觉得自己的认真对待是完全值得的。

通过自己编码实现精灵游戏，不但加深了我对 C++封装、继承和多态的掌握，让我更加深刻地理解了 C++这一面向对象语言的基本思想，也对我的设计和编码能力有较大的提升，此外，还让我在不断地寻找问题、思考解决方案的过程中锻炼了自己解决问题的能力。

在实验的过程中，我遇到了很多问题。在刚开始进行实验的时候，由于刚学习 C++，并且第一次使用 C++进行软件开发，对继承、多态等语法的掌握并不熟练，导致一开始进度极其缓慢。因此，我决定先阅读课本巩固相关知识，并结合码图进行针对性的练习，之后再次进行实验时，进展明显顺利很多。

此外，在设计实现功能时也遇到过很多问题，这些问题大多都在自己的不断调试和修改，以及同学的帮助下顺利解决，但其中依然存在少数问题至今仍未解决，如 scu 精致不动时，stucd 和 huaxi 由于自己的运动接触到 scu，并不会导致被吃掉，此外，uestc 虽然有追捕功能，但并不理想，需要 scu 静止不动才能得到较好的体现。此外，游戏的视觉效果也并不好，由于能力和时间限制，界面较为粗糙。此外，还有一些同样不影响功能、但是从设计与实现的角度来看还有待优化的其它问题，在此不一一列举。而我之所以没有再进一步去努力解决这些存在的问题，最主要的阻力在于时间与精力有限，作为大三刚转到新专业的学生，一学期上两学期的课，课业压力巨大，也恳请老师与助教学长理解。

总之，本实验过程中确实存在一些值得改进的地方，这些问题将为我后续的学习与工作提供重要参考，成为我在未来的成长道路中的宝贵经验。尽管由于能力有限，这些问题暂时未能得到很好的解决，但我相信随着我知识和经验的进一步积累，在不久的将来我会有能力来把上述存在的问题顺利解决，并把已经实现的部分做得更好。