北京郵電大學

程序设计实践实验报告

题　　目： **简单绘图程序**

姓　　名：张　　阳

学　　院：信息与通信工程学院

专　　业：通信工程

班　　级：2016211111

学　　号：2016210318

班内序号：20

指导教师：黄平牧老师

2018 年 5 月

目录

摘要与前言

本实验用 C++与 Python 写了一个简单的绘图程序，在老师的指导下写出了一个用于描述图形的 shape 类和用于绘制图形的 Graphicwindow 类。并自己完成了复杂图形——弹球和使用 python 绘制了一个小猪佩奇。（由于担心字体不兼容问题，所以同时生成了 pdf 版本的实验报告）本次课程所写的的代码我同时备份到我的 GitHub 网页上了：

https://github.com/zyzisyz/Programming-Practice

关键词

C++ windows 程序开发 绘画

一、 程序设计的任务与要求

1. 功能要求

设计实现一个简单绘图程序，支持下列菜单功能：

（1） 文件
- 参数设置
- 可设计成对话框形式
- 退出

（2） 绘图
- 简单图形
- 点和线
- 三角形
- 抛物线
- 正弦曲线
- 时钟
- 复杂图形（动画）

  比如：弹球，随机数分布，。。。。。。

2. 成绩评定

  占 50 分：验收（25）+实验报告（含源代码）（25）

3. 实验报告

  开发思路，主要技术，相关算法，主要编程点（书上的尽量少写，主要写自己的东西）

4. 案例提交

（1） 实验报告（含源代码）提交至：

  homework_bupt@163.com

（2） 文件名命名格式

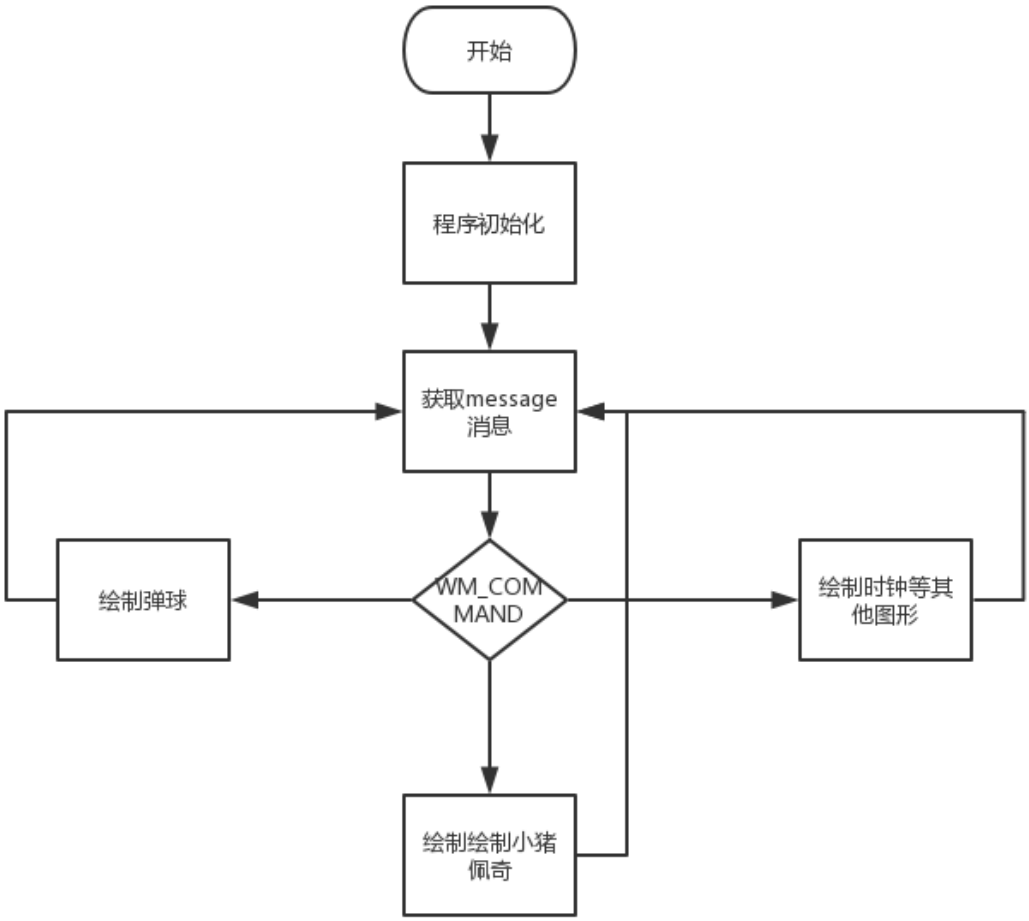
  班级_姓名_小班序号_题目.rar

二、 程序设计的开发思路与结构框图

1. 开发思路:

   首先创建一个用于描述图形的 shape 类和一个用于绘制图形的 Graphicwindow 类，其中大部分绘图函数主要通过调用图形设备接口 GDI 函数实现，然后通过这两个类实现绘图。windows 程序在响应 VM_PAINT 消息时，使用 hdc=BeginPaint（）函数和 EndPaint（hwnd，&ps）函数获取设备环境，然后调用所写好的绘图程序进行图形绘制。

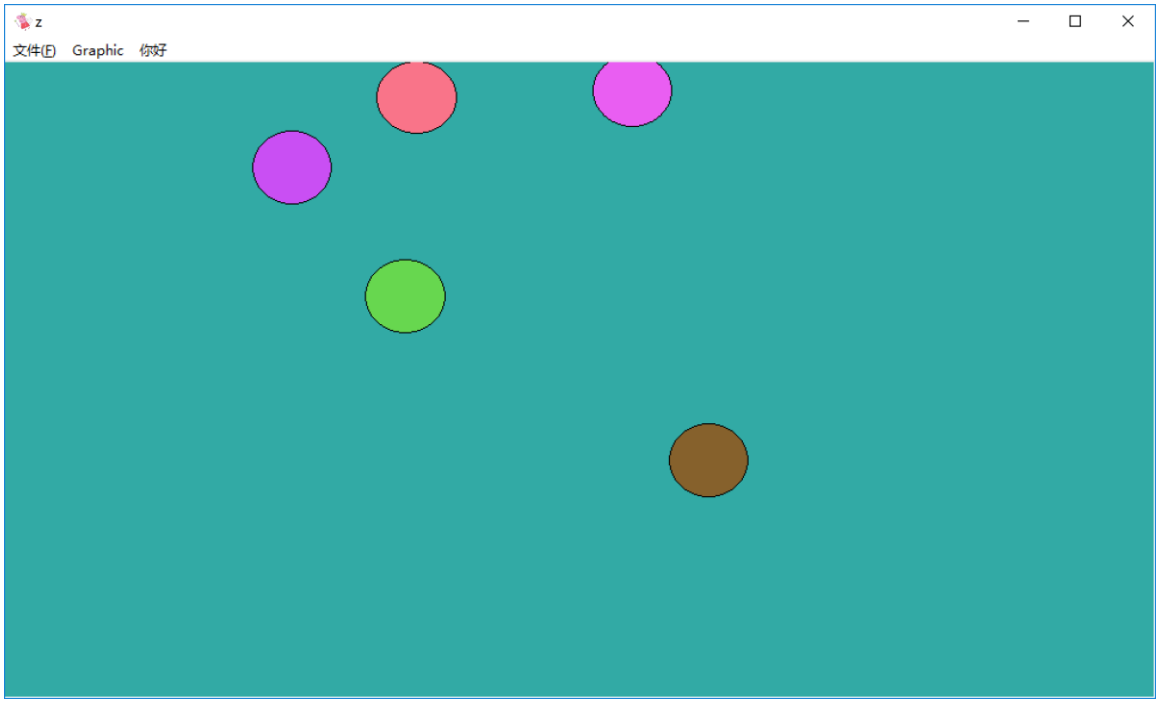
2. 结构图:



三、 主要编程点：源码算法分析与函数功能说明

1. 复杂图形——弹球

在窗口中生成 5 个不同方向、不同颜色、不同移动速度的弹球，当弹球碰壁时将自动改变他的颜色和运动轨迹。背景颜色做成蓝色的呼吸灯效果，在不同的色度的蓝色之间随着时间的变换而变换。



算法分析与说明:

1) 小球的初始化

   a) 算法分析:

      定义了多个全局变量，分别对应小球生成的坐标、颜色、和移动方向。使用随机数初始化 5 个小球的初始坐标、入射角度、颜色等变量。其中两个 bool 类型的变量用于 goleft 和 goup 用于判断是否碰壁需要改变运动轨迹。

   b) 时间复杂度： O（n）

   c) 代码实现为:

```
int n = 5;                                          //球的个数
int r[5], g[5], b[5];                    //颜色
```

```
double x[5], y[5];                          //位置
double w[5];                                //倾斜角

int Ccount;                                 //计数器
Point p[5];
Circle c[5];
bool GoLeft[5];
bool GoUp[5];
//该函数使用随机数初始化小球的各种参数
void init()
{
    Ccount = 0;
    srand((unsigned)time(NULL));
    for (int i = 0; i < n; ++i)
    {

        r[i] = rand() % 256;
        g[i] = rand() % 256;
        b[i] = rand() % 256;
        x[i] = (rand() / 100) % 20 - 10;
        y[i] = (rand() / 100) % 12 - 6;
        p[i].setPoint(x[i], y[i]);
        c[i].setCircle(p[i], 0.7);
        w[i] = PI / 2 * (rand() % 200) / 200;
        if ((PI / 4) > w[i] || w[i] > 0)
        {
            GoUp[i] = true;
            GoLeft[i] = true;
        }
        else if (2 * (PI / 4) > w[i] || (PI / 4) < w[i])
        {
```

```
                GoUp[i] = true;
                GoLeft[i] = false;
            }
            else if (1.5*(PI / 4) > w[i] || 2 * (PI / 4) < w[i])
            {
                GoUp[i] = false;
                GoLeft[i] = false;
            }
            else if (2 * (PI / 4) > w[i] || 1.5 * (PI / 4) < w[i])
            {
                GoUp[i] = false;
                GoLeft[i] = true;
            }
            else if (PI / 4 == w[i])
            {
                GoLeft[i] = true;
            }
            else if (PI / 2 == w[i])
            {
                GoUp[i] = true;
            }
            else if (3 * PI / 4 == w[i])
            {
                GoLeft[i] = false;
            }
            else if (2 * PI == w[i])
            {
                GoUp[i] = false;
            }
        }
    }
```

2) 小球的运动轨迹算法

a) 算法分析

当小球运动到逻辑坐标的边界时候，两个 bool 类型其中有一个会改变，从而控制小球运动方向的改变。例如，当小球圆心+半径=逻辑坐标边界 y=6 时，bool 类型 goup 变为 false，当 goup==false 为真时，y 方向的速度变为负数，从而实现了小球撞壁后轨迹的改变，x 方向的也同理。同时在碰壁时重新生成随机数，实现了小球颜色的改变。

b) 时间复杂度： O（n）

c) 代码实现为：

```
//位置变化
for (int i = 0; i < n; ++i)
{
        //判断有没有有到边界
        if (x[i] + 0.5 >= 10)
        {
                GoLeft[i] = false;
                r[i] = rand() % 256;
                g[i] = rand() % 256;
                b[i] = rand() % 256;

        }
        if (x[i] - 0.5 <= -10)
        {
                GoLeft[i] = true;
                r[i] = rand() % 256;
                g[i] = rand() % 256;
                b[i] = rand() % 256;

        }
        if (y[i] + 0.5 >= 6)
```

```
{
        GoUp[i] = false;
        r[i] = rand() % 256;
        g[i] = rand() % 256;
        b[i] = rand() % 256;

}
if (y[i] - 0.5 <= -6)
{
        GoUp[i] = true;
        r[i] = rand() % 256;
        g[i] = rand() % 256;
        b[i] = rand() % 256;

}
if (GoLeft[i])
{
        x[i] += (0.5 + 2 * double(i) / 10)*cos(w[i]);
}
else
{
        x[i] -= (0.5 + 2 * double(i) / 10)*cos(w[i]);
}

if (GoUp[i])
{
        y[i] += (0.5 + 2 * double(i) / 10)*sin(w[i]);
}
else
{
        y[i] -= (0.5 + 2 * double(i) / 10)*sin(w[i]);
}
```

```
    }
```

3) 背景的颜色的变化

a) 算法分析

该算法参考了数电实验实验呼吸灯的实现思路，随时间的变换，背景色蓝色的色度改变。电脑的颜色由 RGB 三色控制，随着时间的改变，改变 RGB 的小范围数值后复原，即可实现呼吸灯的效果。全局变量 Ccount 用于记录时间的变换。

b) 时间复杂度： O（1）

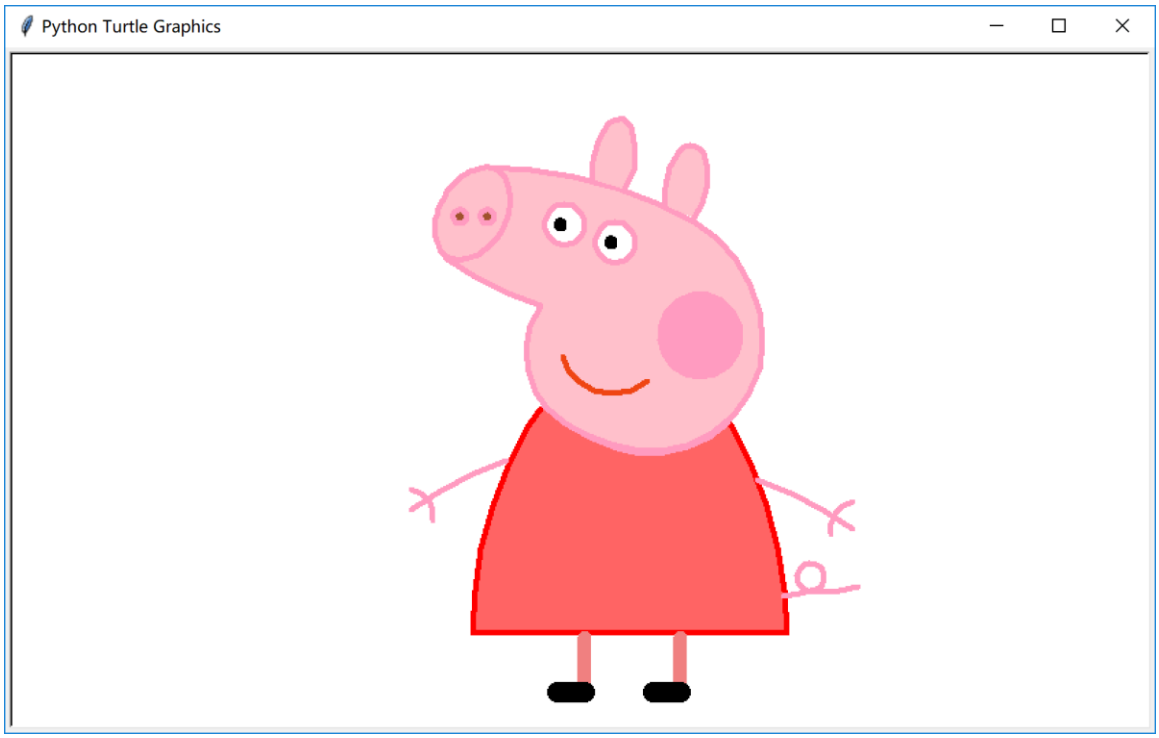
c) 代码实现为：

```
cwin.setBrush(RGB(0 + Ccount, 120 + Ccount, -Ccount + 215));
DrawBall(cwin);
Ccount += 10;
if (abs(Ccount) >= 55)
{
    Ccount = 0;
}
```

2. 使用 Python 绘制小猪佩奇

在电脑中安装 Python 的解释器，并配置环境变量，在 C++中使用该语用 windows 命令调用 python 脚本：system("python pig.py");

该图形使用 python 自带的的 turtle 库绘制



代码为：

```
# coding:utf-8
import turtle as t
t.pensize(4)
t.hideturtle()
t.colormode(255)
t.color((255,155,192),"pink")
t.setup(840,500)
t.speed(200)
#鼻子
t.pu()
t.goto(-100,100)
```

```
t.pd()
t.seth(-30)
t.begin_fill()
a=0.4
for i in range(120):
    if 0<=i<30 or 60<=i<90:
        a=a+0.08
        t.lt(3) #向左转 3 度
        t.fd(a) #向前走 a 的步长
    else:
        a=a-0.08
        t.lt(3)
        t.fd(a)
t.end_fill()
t.pu()
t.seth(90)
t.fd(25)
t.seth(0)
t.fd(10)
t.pd()
t.pencolor(255,155,192)
t.seth(10)
t.begin_fill()
t.circle(5)
t.color(160,82,45)
t.end_fill()
t.pu()
t.seth(0)
t.fd(20)
t.pd()
```

```
t.pencolor(255,155,192)
t.seth(10)
t.begin_fill()
t.circle(5)
t.color(160,82,45)
t.end_fill()
#头
t.color((255,155,192),"pink")
t.pu()
t.seth(90)
t.fd(41)
t.seth(0)
t.fd(0)
t.pd()
t.begin_fill()
t.seth(180)
t.circle(300,-30)
t.circle(100,-60)
t.circle(80,-100)
t.circle(150,-20)
t.circle(60,-95)
t.seth(161)
t.circle(-300,15)
t.pu()
t.goto(-100,100)
t.pd()
t.seth(-30)
a=0.4
for i in range(60):
    if 0<=i<30 or 60<=i<90:
```

```
        a=a+0.08
        t.lt(3) #向左转 3 度
        t.fd(a) #向前走 a 的步长
    else:
        a=a-0.08
        t.lt(3)
        t.fd(a)
t.end_fill()
#耳朵
t.color((255,155,192),"pink")
t.pu()
t.seth(90)
t.fd(-7)
t.seth(0)
t.fd(70)
t.pd()
t.begin_fill()
t.seth(100)
t.circle(-50,50)
t.circle(-10,120)
t.circle(-50,54)
t.end_fill()
t.pu()
t.seth(90)
t.fd(-12)
t.seth(0)
t.fd(30)
t.pd()
t.begin_fill()
t.seth(100)
```

```
t.circle(-50,50)
t.circle(-10,120)
t.circle(-50,56)
t.end_fill()
#眼睛
t.color((255,155,192),"white")
t.pu()
t.seth(90)
t.fd(-20)
t.seth(0)
t.fd(-95)
t.pd()
t.begin_fill()
t.circle(15)
t.end_fill()
t.color("black")
t.pu()
t.seth(90)
t.fd(12)
t.seth(0)
t.fd(-3)
t.pd()
t.begin_fill()
t.circle(3)
t.end_fill()
t.color((255,155,192),"white")
t.pu()
t.seth(90)
t.fd(-25)
t.seth(0)
```

```
t.fd(40)
t.pd()
t.begin_fill()
t.circle(15)
t.end_fill()
t.color("black")
t.pu()
t.seth(90)
t.fd(12)
t.seth(0)
t.fd(-3)
t.pd()
t.begin_fill()
t.circle(3)
t.end_fill()
#腮
t.color((255,155,192))
t.pu()
t.seth(90)
t.fd(-95)
t.seth(0)
t.fd(65)
t.pd()
t.begin_fill()
t.circle(30)
t.end_fill()
#嘴
t.color(239,69,19)
t.pu()
t.seth(90)
```

```
t.fd(15)
t.seth(0)
t.fd(-100)
t.pd()
t.seth(-80)
t.circle(30,40)
t.circle(40,80)
#身体
t.color("red",(255,99,71))
t.pu()
t.seth(90)
t.fd(-20)
t.seth(0)
t.fd(-78)
t.pd()
t.begin_fill()
t.seth(-130)
t.circle(100,10)
t.circle(300,30)
t.seth(0)
t.fd(230)
t.seth(90)
t.circle(300,30)
t.circle(100,3)
t.color((255,155,192),(255,100,100))
t.seth(-135)
t.circle(-80,63)
t.circle(-150,24)
t.end_fill()
#手
```

```
t.color((255,155,192))
t.pu()
t.seth(90)
t.fd(-40)
t.seth(0)
t.fd(-27)
t.pd()
t.seth(-160)
t.circle(300,15)
t.pu()
t.seth(90)
t.fd(15)
t.seth(0)
t.fd(0)
t.pd()
t.seth(-10)
t.circle(-20,90)
t.pu()
t.seth(90)
t.fd(30)
t.seth(0)
t.fd(237)
t.pd()
t.seth(-20)
t.circle(-300,15)
t.pu()
t.seth(90)
t.fd(20)
t.seth(0)
t.fd(0)
```

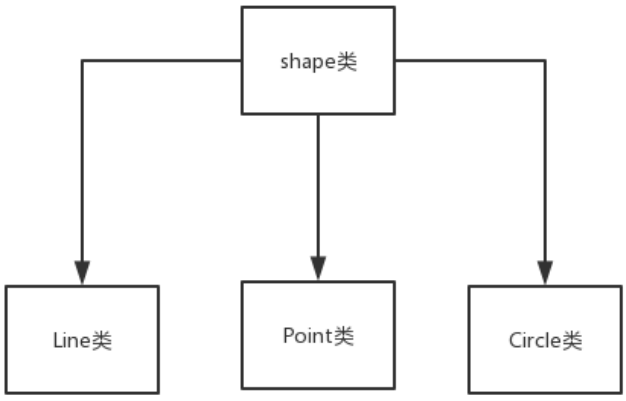
```
t.pd()
t.seth(-170)
t.circle(20,90)
#脚
t.pensize(10)
t.color((240,128,128))
t.pu()
t.seth(90)
t.fd(-75)
t.seth(0)
t.fd(-180)
t.pd()
t.seth(-90)
t.fd(40)
t.seth(-180)
t.color("black")
t.pensize(15)
t.fd(20)
t.pensize(10)
t.color((240,128,128))
t.pu()
t.seth(90)
t.fd(40)
t.seth(0)
t.fd(90)
t.pd()
t.seth(-90)
t.fd(40)
t.seth(-180)
t.color("black")
```

```
t.pensize(15)
t.fd(20)
#尾巴
t.pensize(4)
t.color((255,155,192))
t.pu()
t.seth(90)
t.fd(70)
t.seth(0)
t.fd(95)
t.pd()
t.seth(0)
t.circle(70,20)
t.circle(10,330)
t.circle(70,30)
t.done()
```

3. Shape 类

（Shape 类写在 GraphicWindows.h 中）

基本图原类，包括 Point 类、Line 类、Circle 类和 Message 类，Shape 类有纯虚函数，是 Point 类、Line 类、Circle 类共有的基类，它是抽象类。Message 用于显示文字。



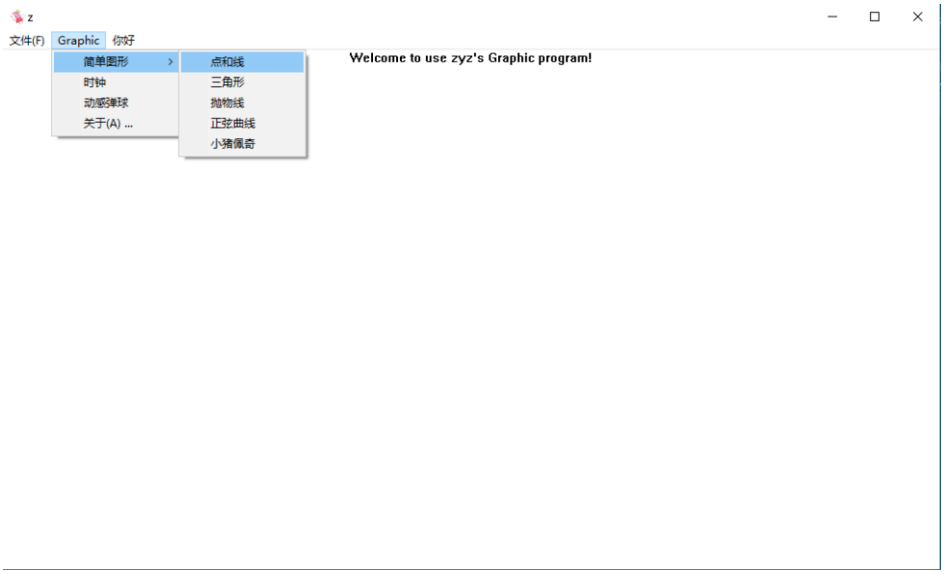
4. GraphicWindows 类

该类用于描述绘图行为。实现了获取窗口信息，设备与逻辑坐标转换，绘制点、线圆等图形和重载运算符号<<等功能。完成了具体绘图的操作。

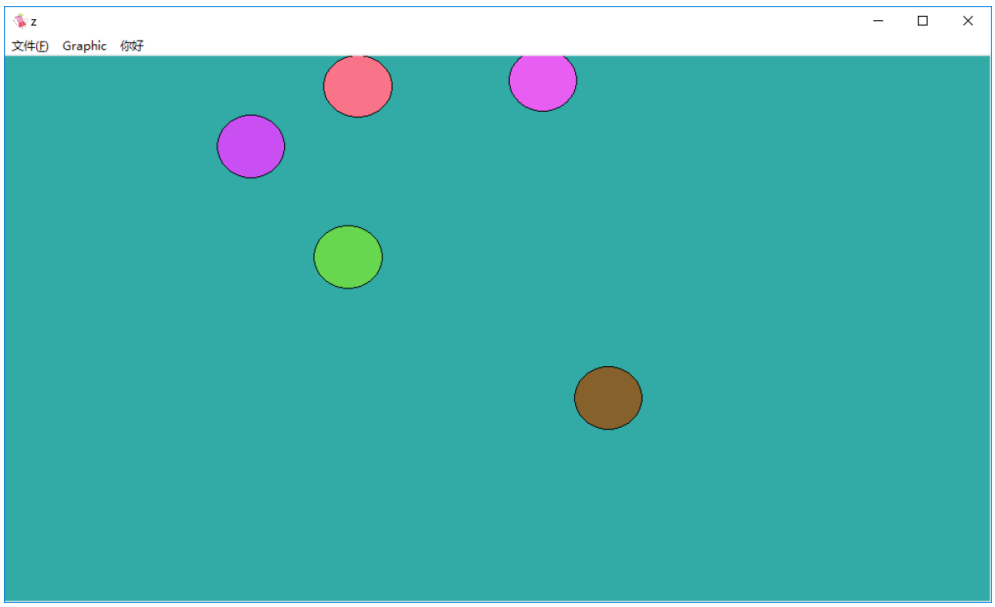
其中 X 逻辑坐标范围是[-10,10]，y 我与课本的不同设置为[-6,6]。

四、 总体程序实现功能说明与展示

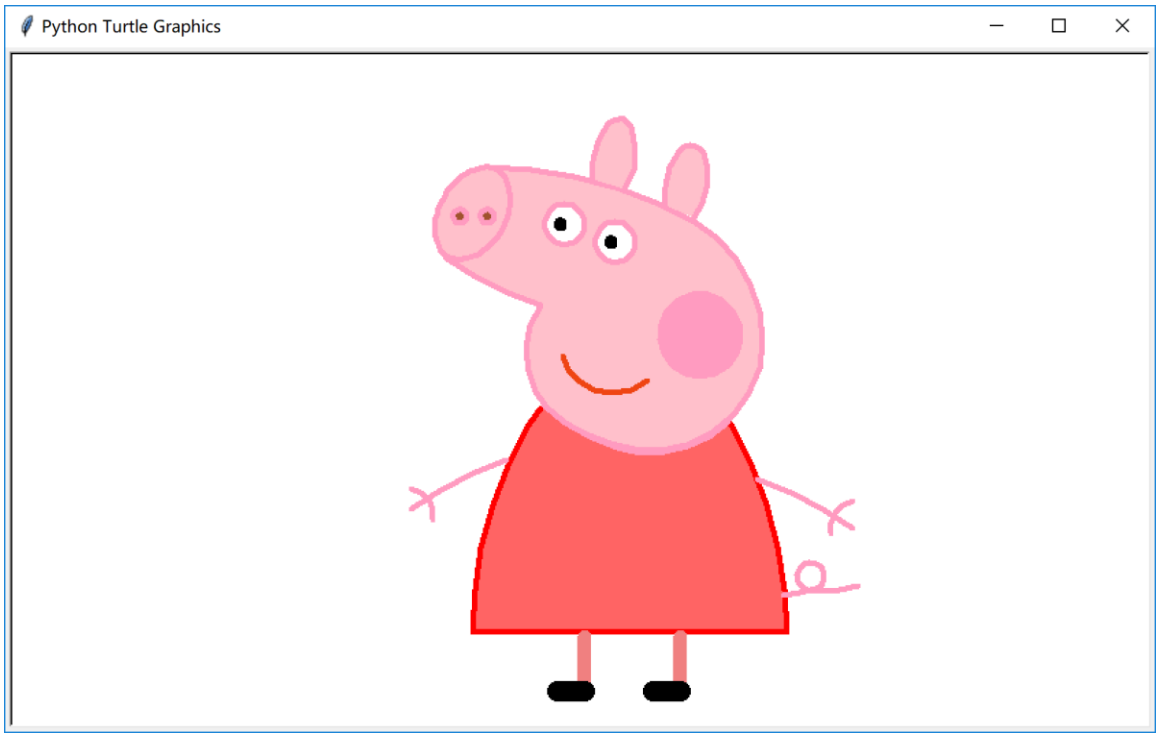
1. 初始界面：



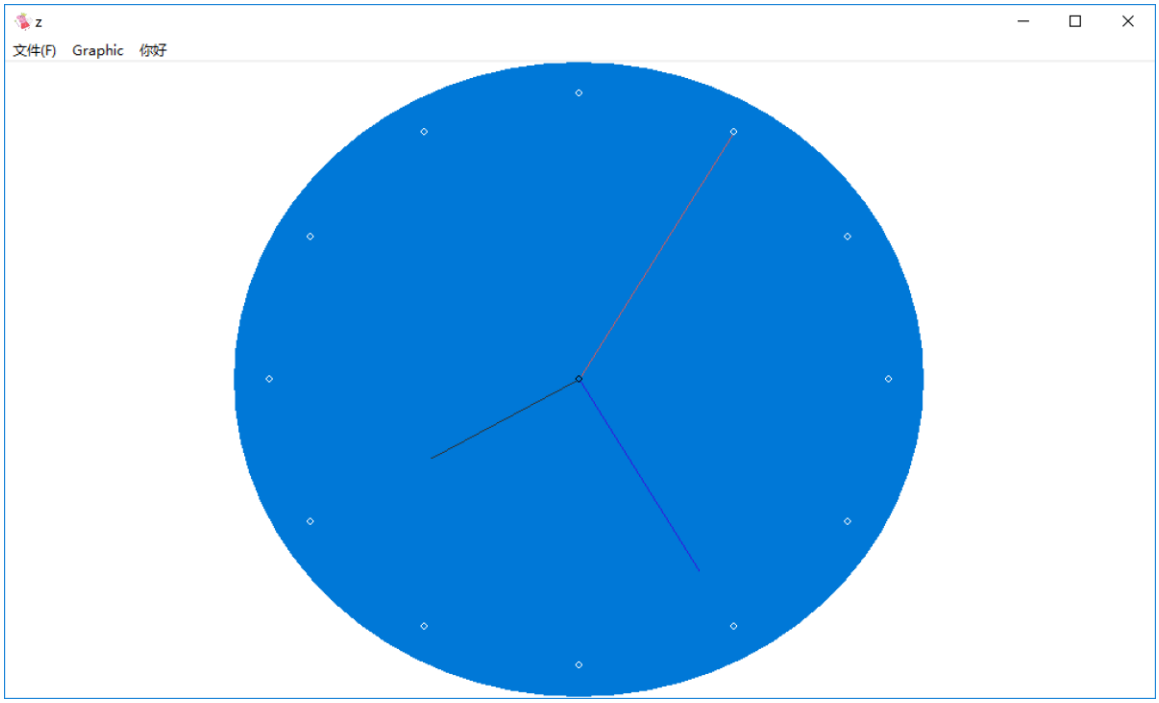
2. 绘制复杂图形——弹球



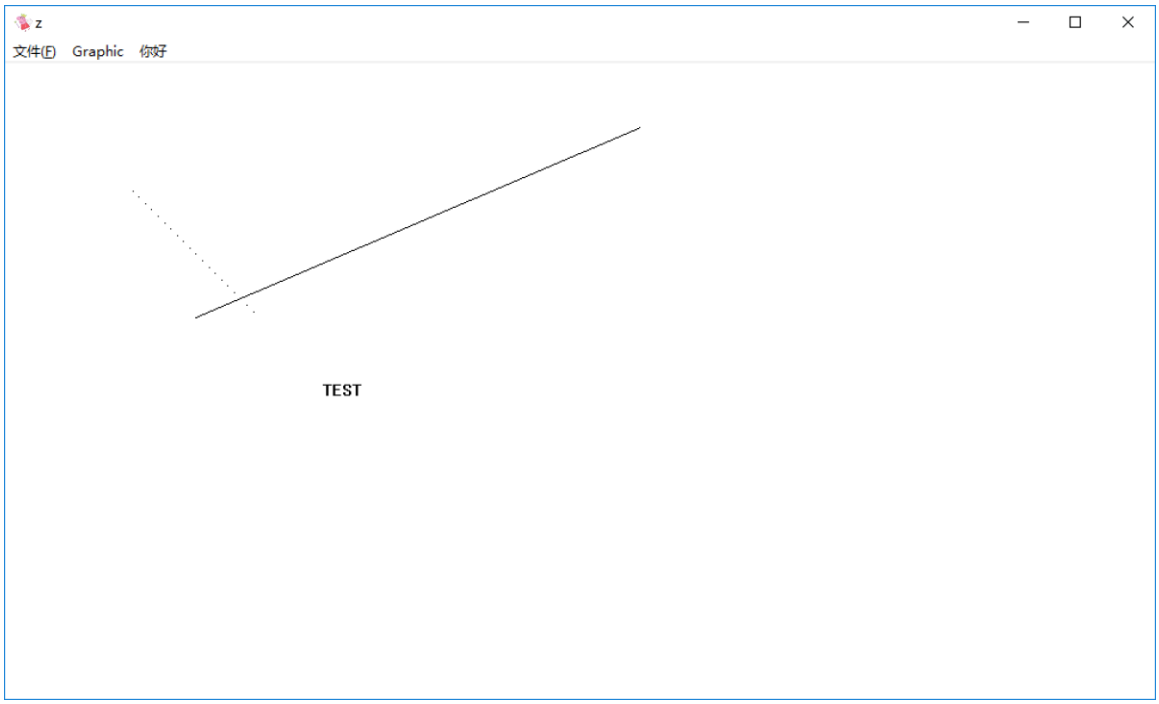
3. 绘制小猪佩奇



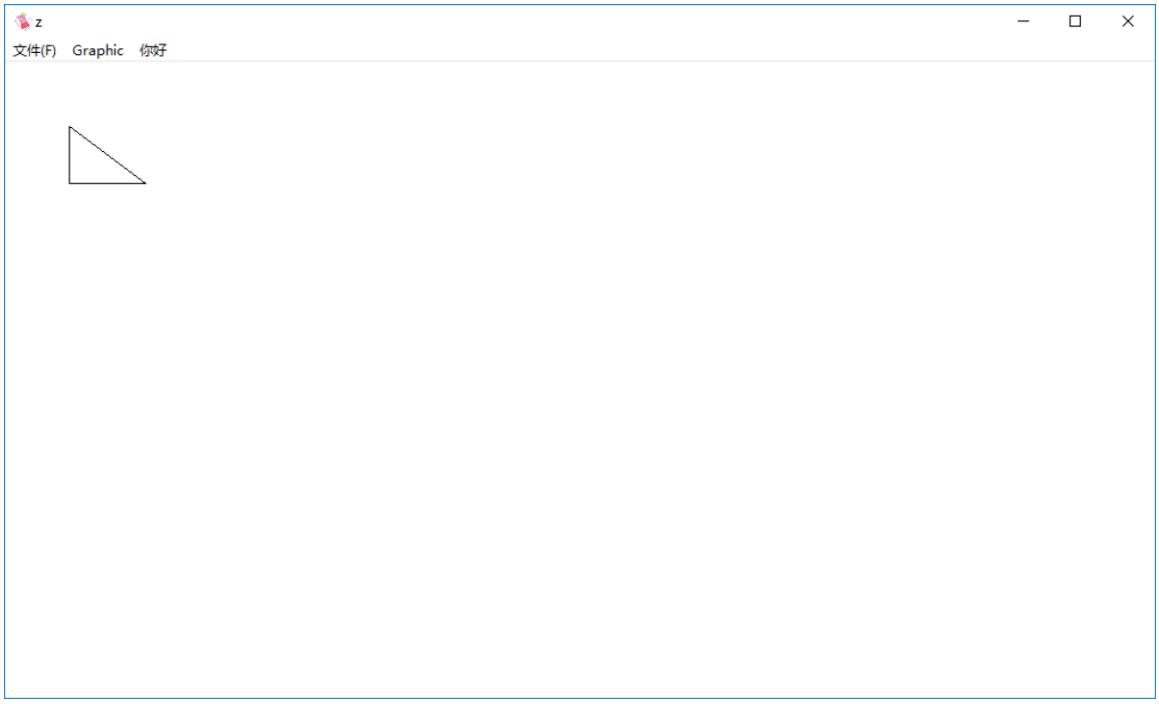
4. 绘制动态时钟:



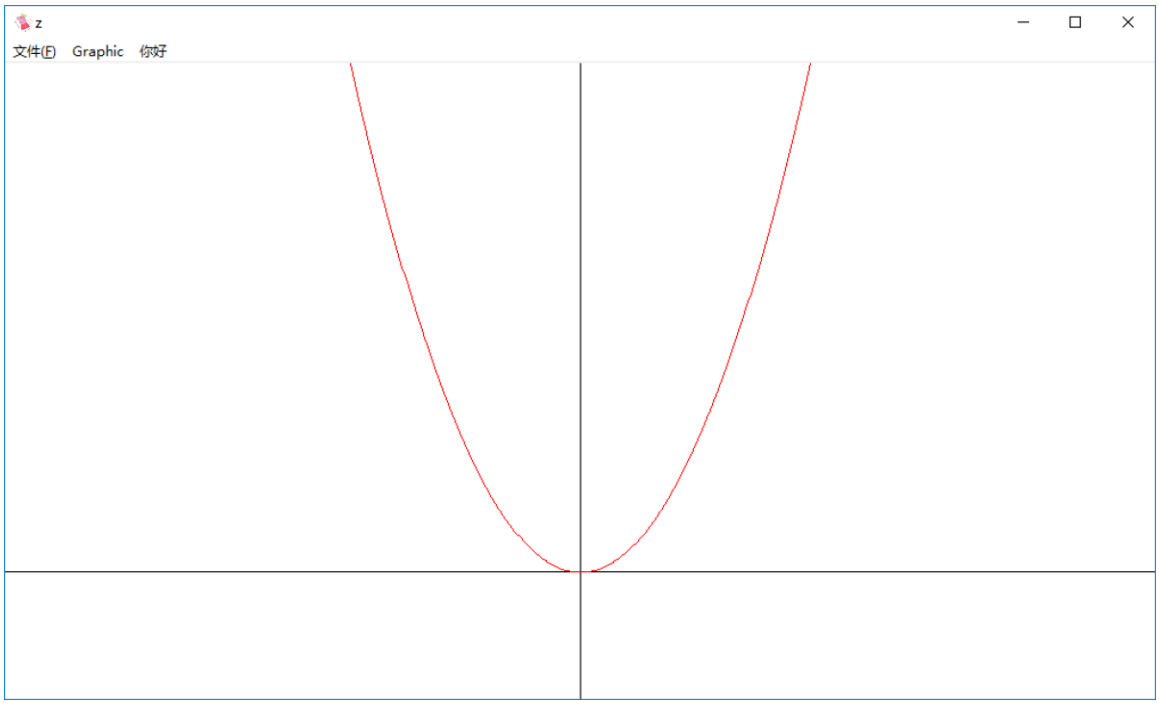
5. 绘制点和线：



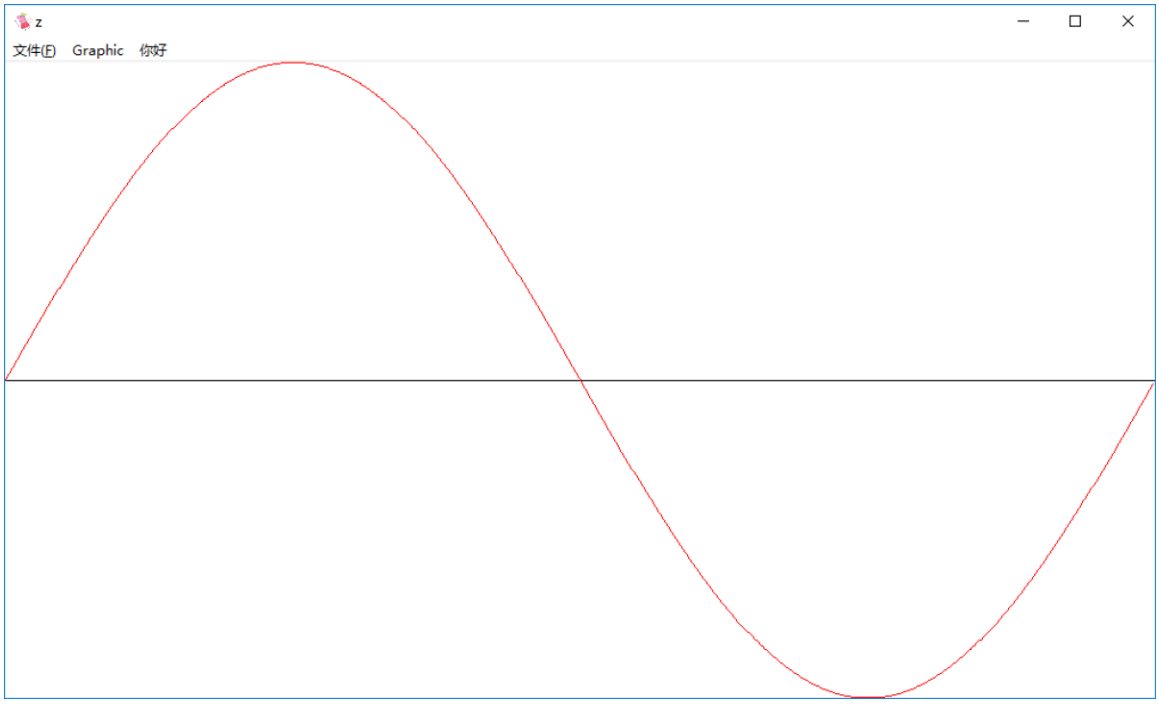
6. 绘制三角形：



7. 绘制抛物线



8. 绘制正弦曲线



五、 故障与问题分析

1. 编译中的问题：
   在源代码编译中，出现了诸多的问题。一开始我把所有的类实现在不同的头文件下，类与类之间互相调用，出现了诸多错误。后面查了很多 C++编译的原理，也看了很多论坛，发现虽然 C++是一们向对象的语言，但是他的编译和链接是面向过程的，由于自己并没有系统的学过编译原理，所以我就把所有类实现的代码按照调用顺序，全部写在了 Graphicwindows.h 一个头文件下，最后顺利的解决了问题。
   尽管我认为这并不是一种好的方法，但由于时间有限，没有太多的时间去学习 C++编译和链接的过程，今后有时间我将好好的重新更改我的代码，使之更具有可读性和结构性。

2. 解决动画无法流畅动的方法：
   在一开始写动画的时候，动画没法正确刷新，最后设置断点调试后发现是自己在每个 message 后没有写 InvalidateRect(hWnd, NULL, true);造成的。

六、 总结与心得体会

1. 本次实验前前后后写了将近快 1200 行代码，充分练习与使用了 C++中的虚函数、重载运算符、面向对象等特性，并在老师的指导下绘制出了复杂的动画图形。让我对程序的开发有了更深刻的理解，同时也更加牢固的掌握了 C++这门强大的程序设计语言。

2. 我学习并使用了分布式版本控制系统 Git 以应对大量的代码编写，在编程的过程中出现种种问题，比如电脑死机等，在一些时候想恢复到曾经的代码，因此在本次编程中，在每次上完课时都及时 Git，并把它们备份到我的 GitHub 网页上。
   （https://github.com/zyzisyz/Programming-Practice）

3. 通过这次开发，使我对 Git 命令更加熟悉，也记住了很多 window 的 CMD 命令，同时更重要的是我了解与学到了很多软件工程的思想。

4. 在此次实验中，我使用 photoshop 制作了全新的图标，并把它应用到了我的程序上，使程序更加具有个性。

graphic.exe