# CircleWar

## 一、项目简介

CircleWar是一个用python语言编写，基于Tkinter库的一个简单的小游戏。

**项目作者：沈小洲 (学号5142609052)**

运行所需环境：

1. Python 2.7.x
2. Tkinter模块
3. Windows下需设置.pyw与python的文件关联

运行方法：

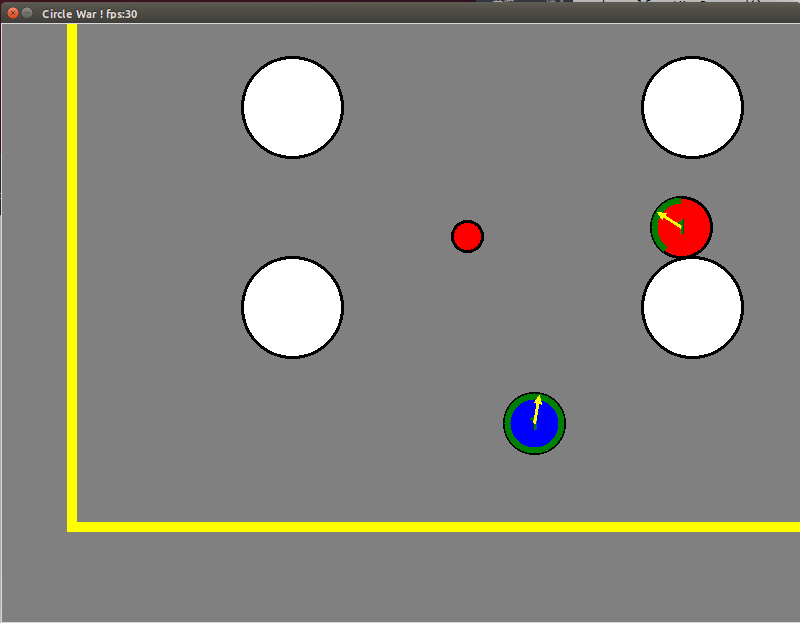
1. 在Ubuntu 14.04中的终端下运行如下代码



其中<path-to-this-project>是整个项目文件夹的目录

1. 在Windows中打开项目文件夹下的src文件夹，双击circlewar.pyw运行

Demo界面



Demo游戏目标与操作方法：

操作蓝色人物通过射击打败红色人物，不可移出地图边界，不可从障碍物中穿过。

W 向上移动

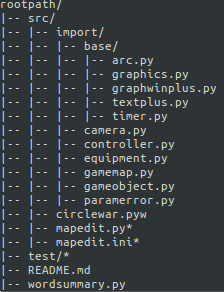
S 向下移动

A 向左移动

D 向右移动

鼠标左键 射击出一颗炮弹

## 二、文件结构

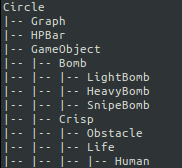


### /src/import/base基础模块

1. 模块graphics.py为第三方图形库
2. 模块arc.py ，参照graphics.py模块中图形类的定义方式，定义了Arc和CircleArc两个类，用于绘制弧形(轮廓形状对应graphics.py中的Oval和Circle)。实现是用底层的Tkinter的canvas的create\_arc函数。
3. 模块TextPlus.py，定义了TextPlus类，继承自graphics.py中的Text，实现了设置文字锚点的功能。
4. 模块graphwinplus.py，定义了GraphWinPlus类，继承自graphics.py中的GraphWin，是对其的扩展，添加的主要功能分为三类：窗体方法、鼠标方法、键盘方法。窗体方法主要实现了设置标题(setTitle)、图标(setIcon)、大小(setWidth, setHeight)；鼠标方法主要实现了获取鼠标按下的不同的键(getMousePressed, isMousePressed)，获得鼠标当前位置(getMouseCurrentPoint)，检查上一次鼠标事件(checkMouse)，获得鼠标滚轮(checkMouseWheel, getMouseWheel)；键盘方法主要实现了获得按下的键(getKeyPressed, isKeyPressed, checkKey)。详细信息见代码中注释，或者可以在python中使用help(GraphWinPlus)输出帮助信息。
5. 模块timer.py定义了Timer类，作用是每隔给定的时间调用一次给定的回调函数，详细用法见代码中注释，或者在python中使用help(Timer)来查看。

### /src/import 游戏组件模块

1. 模块gamemap.py定义了GameMap类，用于存储地图信息以及对地图的处理，包括从文件中加载地图(load)，将地图保存到文件(save)，往地图中增减游戏物品(addPlayer, addObstacle, removePlayer, delObstacle)，检查地图合法性(check) 。
2. 模块gameobject.py定义了GameObject类，用于存储游戏物品的信息。继承关系：



Circle是最底层的类，有位置(x, y, center)和半径(r)信息，有移动操作(move, moveTo)，判断相交的方法(intersect)。

Graph继承自Circle，不过底层利用graphics.Circle来存储圆形的图形信息，同时负责将图形绘制到窗口上(draw, undraw)。

HPBar则用来显示环形血条，可以设置填充部分的百分比(setPercentage)。

Graph和HPBar底层都是利用graphics里的图形类来维护绘图功能，利用adjust方法来保持自身继承自Circle父类的信息和图形类信息的同步。

GameObject是游戏物品的基类，拥有队伍(team)、速度(speed)、图形(graph, 一个Graph实例)属性。

Bomb和Crisp继承自GameObject。

Bomb表示游戏中的炸弹，有伤害属性(attack)，飞行距离(dis, 只在初始化中设定)，速度(speed, 继承自GameObject)，负责移动的则是直线飞行的fly方法。

Crisp则是游戏中有血量，能被破坏的物品，有防御力(defense)，装甲(armor)，装甲数量(layer)，增加装甲(addArmor)，基础血量(hpOut)，伤害计算(hurtBy)，摧毁计算(destroy)。

Bomb派生出不同的炸弹种类(LightBomb, HeavyBomb, SnipeBomb)，拥有不同的属性。

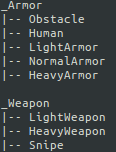
Crisp又派生出Obstacle和Life，分别是障碍物和生物。

Obstacle不可移动，有把撞击他的物品推开的方法(pushAway)，即碰撞处理方法。

Life可以移动和攻击，有武器(weapon)，有血量(hpbar, 一个HPBar的实例)，用run方法处理移动，用turnTo方法处理武器转向，有正在射击的判定(isShooting, setShoot, 用于按住按键不放进行连续射击)，可以即时生成炸弹对象(makeBomb, 实现是调用了weapon的makeBomb方法)。

从Life最后派生出Human，用于表示玩家，初始化自带装甲，和初始化可选的武器。

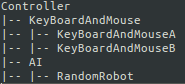
1. 模块equipment.py定义了各种不同种类的Armor和Weapon，如下



\_Armor为装甲基础类，有最大血量(max\_hp)，当前血量(hp)，血量增减(addHP, subHP)，重量(weight, 影响移动速度)，防御力(defense)。Obstacle为gameobject.Obstacle的装甲类型，Human为gameobject.Human的基础装甲类型。其余为附加可选装甲。

\_Weapon为武器基础类，有炸弹类型(bomb\_class, 仅在初始化时设定)，重量(weight)，用于图形化显示的属性(width, color)，标记射击状态的属性(shoot)，生成对应炸弹的方法(makeBomb)。

1. 模块controller定义了Controller和AI。继承结构如下：



Controller用于从初始化时给定game的camera获取用户输入或者自定义输入，是一个抽象类，仅仅定义了获取输入的接口，可以用统一的方式从不同的输入方式得到输入，包括获取射击状态(getShoot)，获取武器指向(getDirection)，获取移动方向(getMoveFourDirection)。Controller类用start方法来初始化(可为空)，用stop方法来停止(可为空)。

KeyBoardAndMouse从给定的camera获取按键和鼠标输入，实现了上述三个方法，KeyBoardAndMouseA，KeyBoardAndMouseB则是两套不同的按键方案。

AI则是对Controller的包装，各种输入可以由程序内部产生。AI类主要在初始化时可以添加额外的信息，供复杂的AI使用。由于时间关系，目前只实现了随机AI，即RandomRobot类，随机的移动和射击。

1. 模块camera.py定义了Camera类，继承自GraphWinPlus，一是管理窗口，二是负责图像正确地在屏幕上显示。顾名思义，屏幕范围即相机拍摄范围，当屏幕范围小于地图大小时，相机会有选择的移动，显示地图内合适的区域。

Camera类用startup和stop来控制运行，refresh来刷新画面内容，isRunning来判断是否在运行中。

Camera类用toCamera和toMap两个方法来保持地图坐标系和屏幕坐标系的映射，用Move和MoveTo来移动相机拍摄范围，用catch方法来计算一个物品是否在相机范围内。

Camera类用bindObject和unbindObject维护一个物品列表，只有绑定的物品会被相机正确显示。通过lockObject来锁定镜头到一个物品上，镜头会跟随该物品移动，通过unlock来解锁，通过objectLocked获得被锁定的物品。

1. 模块paramerror.py定义了ParamError继承自Exception，在一些函数中当参数不正确时引发该异常。

### /src 游戏主体

1. 该目录下的circlewar.pyw是游戏主程序，定义了CircleWar游戏类，用来初始化和运行一场游戏，通过loadMap来加载地图，addPlayer和emptyPlayer来管理玩家(维护一个\_human列表)，用start来初始化相机以及进入游戏主循环，用stop来中止游戏，用isGameover来判断游戏是否结束。特别要注意的是\_doEveryFrame方法，是每帧要执行的流程，将在第二部分细说。
2. 该程序有一个main函数，用来实例化一个游戏。由于时间原因，只做了一个简单的demo。
3. 该目录下的mapedit.py是地图编辑器程序，目前未完成，mapedit.ini是其配置文件。

### / 根目录

1. 该目录下的test目录为测试socket连接代码，将来可用于游戏联机对战，当前未完成。
2. 该目录下的wordsummary.py用于统计项目规模大小。
3. 该目录下的README.md是引导用的markdown标记文档，用于指引用户运行程序。

## 三、运行原理和流程

### 1. main()

1. 利用GameMap类创建地图(用代码创建或者从文件中载入，src目录的test.cwmap是一个样例)

2. 用该地图创建一个CircleWar游戏实例，其中CircleWar初始化的w,h参数为窗口宽高(摄像机范围)。

3. 创建玩家(Human类)并设定组别(Team)和装备(Armor和Weapon)，然后通过CircleWar的addPlayer方法把玩家对象、控制器(一个Controller对象，可以是获取用户输入的KeyBoardAndMouse对象，也可以是AI对象)、出生点位(在地图中设置)关联起来。

4. 最后调用CircleWar对象的start函数，同时把一个玩家作为参数，使得相机锁定在该玩家上，展示跟随视角。

### 2. game.start(lockPlayer)

1. 从地图中提取出障碍物信息，初始化一个障碍物对象的列表(\_obstacle)，初始化一个炸弹对象的空列表(\_bomb)。

2. 启动相机，并把障碍物、炸弹、玩家对象都与相机绑定(Camera对象的bindObject方法)，并将相机锁定到指定玩家(Camera对象的lockObject方法)。

3. 启动所有玩家的控制器对象(Controller对象的start方法)

4. 启动两个计时器(Timer)，分别用于处理每秒的事情，和处理每帧的事情。

5. 进入主循环(终止条件是isGameover())。利用计时器，在每一秒钟统计FPS；每一帧调用\_doEveryFrame

### 3. game.\_doEveryFrame()

1. 先是移动玩家。对于每个玩家，第一步通过对应的控制器获取用户或者AI程序的输入，设置武器方向、移动玩家、设置射击状态、必要时生成炸弹；第二步进行出界判定，若超出地图边界则设置玩家位置(Human对象的move方法)，使其回到界内；第三步进行障碍物碰撞设定，若与Obstacle对象相交，则重新设置玩家位置，使其离开障碍物(通过Obstacle对象的pushAway方法和Human对象的move方法)。

2. 再是移动炸弹。对于每个炸弹，设置一个标记mark，当它飞过预定的飞行距离(初始化给定的dis)时，或者当与障碍物碰撞时(利用\_obstacle列表)，或者当超出地图边界时，或者与玩家碰撞时(利用CircleWar对象维护的一个\_human列表)，把mark记为True，此时需要把它从游戏中删除(先加入到临时的delbomblist列表中，最后一起删除)。其中当炸弹与玩家碰撞时，计算伤害(Human对象的hurtBy方法)。

最后调用相机的refresh方法，刷新画面。

### 4. camera.refresh()

1. 如果锁定了一个物体，则计算相机的位置，使物体始终处在相机长宽的1/3~2/3范围内。

2. 对于绑定的每个物体，先同步每个物体的图像与实际位置(Graph对象的adjust方法)，如果一个物体出现在相机中(Camera对象的catch方法)，则刷新该物品的图像。

## 四、技术领域

### 文件操作

在GameMap类中有save和load方法，可以将游戏地图保存到文件中，以及从文件中读取地图信息。

除游戏外实现了一个wordsummary.py程序，用于统计项目中.py和.pyw代码的总行数和总字符数，用递归进入文件夹实现深度的探查。

### 函数操作

有大量的类封装和方法定义，逻辑层次清晰，调用有序。

基础组件为了扩展graphics.py模块，直接调用了Tkinter的接口和os, sys等库

大量使用类的继承特性，层次逻辑清晰

### 算法实现

自行构想、设计和实现了游戏运行算法、碰撞检测和处理算法、图像渲染绘制算法，整个流程比较复杂。

另外各种实现按照通用的方式来写，可拓展性很强，比如以后可以加入多种多样的武器装备，多种多样的玩家属性，还有不同的游戏玩法。

### 图形界面

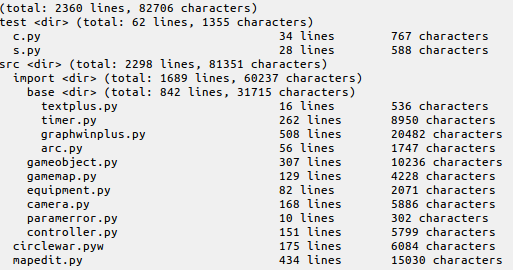
由于时间有限，游戏部分完成度不高，因此只有一个简陋的demo，单单就这个demo来讲，绘制的内容有方框和圆形和文字，其中有自定义的显示的样式，除此之外还有一个自行设计和实现的Arc类，用于在graphics.py提供的环境中绘制弧形，用于血条的显示。

实际上，若给足时间，好好实现游戏部分，可以加入更复杂的游戏界面，特别是开始的选择引导界面，以及额外的地图编辑器程序。

### 项目规模

自己的代码总共2360行，除去未使用的代码(mapedit.py)为1874行。

详细信息如下图：



### 项目逻辑

见第三部分运行原理和流程。逻辑复杂程度很高

### 面向对象

见第二部分文件结构。实现了大量的类的封装，通过继承机制来管理不同的类的关系。类的封装使程序的复杂程度大大降低

### 创意与实用性

这个项目实际上是一个游戏模板，可以在这基础上做出很多有趣的游戏。单单就现在这demo也是一种新式的游戏类型。