第 10 章　游戏时间到了

学习编程有一种惯常的做法，就是先键入一些代码，尽管你可能完全不理解这些代码。确实是这样！

有时仅仅键入代码就能让你对程序如何工作找到一点“感觉”，虽然并不是每一行或每一个关键字都理解。我们在第 1 章就是这么做的，就是那个猜数游戏。现在还是用这个老办法建立一个程序，不过这个程序更长也更有意思。

您将创建一个简单的游戏叫做**英雄兔**，在这里作为英雄的兔子要保卫城堡，它必须能够移动和射击反击敌人（关于敌人我们在 X 章再添加，那时你已经能看懂大部分代码了）。

本章您先

为使用 Python 写游戏你需要安装 PyGame。

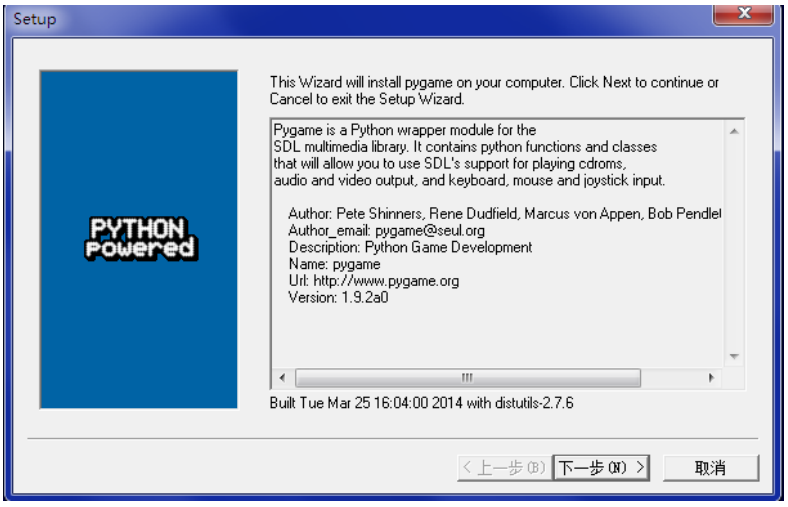
PyGame 是一个使写游戏变简单的 Python 库，它提供如图像处理、声音回放等你可以很容易添加都游戏中的功能。

10.1 安装 PyGame

为使用 Python 写游戏你需要安装 PyGame。PyGame 是一个使写游戏变简单的 Python 库，它提供如图像处理、声音回放等你可以很容易添加都游戏中的功能。

安装方法如下：

- 安装包:依据不同操作系统的版本(32 位/64)选择 32 或者 64 安装包。
- 安装步骤:采用默认安装即可(Next->Next…)，如下图所示：



10.2 游戏时间

Step by step 完成游戏

第 0 步：下载资源

创建游戏前，需要一些图像和声音资源。

资源下载后，在你的硬盘上创建一个游戏使用的文件夹(mygame)，将 resources 文件夹解压到这个文件夹中，这样你的游戏文件夹有了名为 resources 的子文件夹，资源见 QQ 共享：

第一步：（编写简单的）Hello Bunny（嘿，兔子）

运行 IDLE 开发环境，打开一个新的文本编辑窗。在编辑窗键入以下代码：

```
# -*- coding: cp936 -*-
# encoding = utf-8
# 1 - 导入 PyGame 库
import pygame
from pygame.locals import * #导入命名空间

# 2 - 初始化 PyGame 并设置显示窗口
pygame.init()
width, height = 640, 480
screen=pygame.display.set_mode((width, height))

# 3 - 加载你想要给 bunny 使用的图片
player = pygame.image.load("resources/images/dude.png")

# 4 - 循环执行以下缩进的代码
while 1:
    # 5 - 每次绘图前，将屏幕填充成黑色
    screen.fill(0)
    # 6 - 以 100*100 的大小显示图片
    screen.blit(player, (100,100))
```

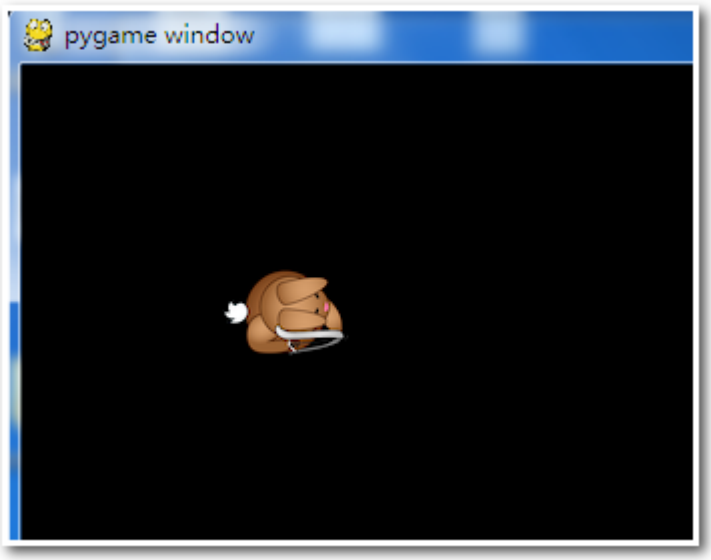
```
    # 7 - 更新屏幕
    pygame.display.flip()
    # 8 - 检查是否有新事件
    for event in pygame.event.get():
        if event.type==pygame.QUIT:
            pygame.quit()
            exit(0)
```

把它保存到你的游戏目录下（即资源子目录）并命名为 itgame.py。让我们逐段分析以上代码：

1. 导入 PyGame 库。这一步让你在你的程序中使用来自库中的函数。
2. 初始化 PyGame 并设置显示窗口。
3. 加载你想要给 bunny 使用的图片。
4. 循环执行以下缩进的代码。
5. 在绘图前，将屏幕填充成黑色。
6. 将之前加载进来的 bunny 图片以 100*100 的大小显示在屏幕上。
7. 更新屏幕。
8. 检查是否有新事件,如果有的话，否则转到退出命令，退出程序。

注意：根据 PyGame 的文档，你不需要调用 pygame.quit()因为解析器关闭时会自动调用它。

如果现在运行这段代码（在 Idle 菜单栏点击 “Run\Run Module"），你应该可以看到一个如下所示的屏幕：



耶，兔子就显示在屏幕上了，并准备做动作！

但是只有一只兔子显示在一个黑漆漆的屏幕上，这个游戏看起来很吓人且很孤单。接下来要做的就是稍微美化一下咯。

第二步：添加布景

我们首先给游戏场景添加背景图片，可以通过调用一组 screen.blit()来完成背景添加。

在代码的#3 小节，载入角色图像之后添加下面的代码(**加粗的部分**)：

```
# 3 - 加载你想要给bunny使用的图片
player = pygame.image.load("resources/images/dude.png")
grass = pygame.image.load("resources/images/grass.png")
castle = pygame.image.load("resources/images/castle.png")
```

这些代码载入图片然后赋给指定的变量，然后把它们画到屏幕上。但如果你检查草地的图片，你会发现它没有覆盖整个 640 x 480 的屏幕，所以你必须平铺使草地的图片完全覆盖屏幕。

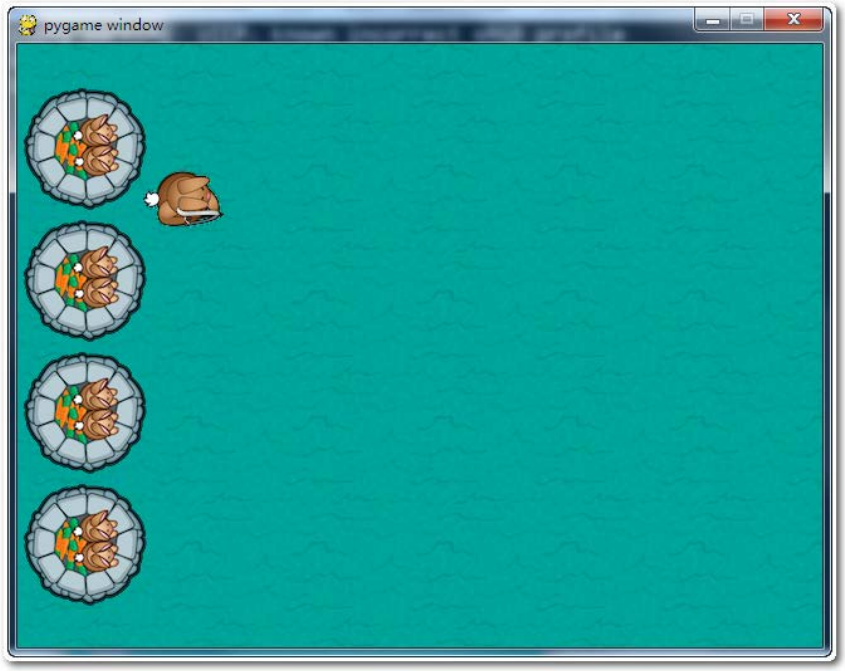
在#6 小节的开始(把兔子画在屏幕上之前)，添加以下代码到 game.py(**加粗的部分**)：

```
# 6 - 以100*100的大小显示图片
for x in range(width/grass.get_width()+1):
    for y in range(height/grass.get_height()+1):
        screen.blit(grass,(x*100,y*100))
        screen.blit(castle,(0,30))
        screen.blit(castle,(0,135))
```

```
screen.blit(castle,(0,240))
screen.blit(castle,(0,345 ))
screen.blit(player, (100,100))
```

正如你所见，首先把 x 坐标通过 for 循环递增，在这个循环中再把 y 坐标循环递增，并且将草地画在使用循环生成的 x、y 坐标上。紧接着的一组代码只是将城堡画在屏幕上。从现在开始，变得好看点了！

你现在运行这个程序，你会看到像如下的结果：



第三步：让兔子动起来

下面你需要添加一些真正的游戏元素，比如让兔子响应键盘的按键。

要做到这一点，你需要实现好一个方法记录哪一个键在某一时刻被按下。你可以简单的使用一个数组保存在游戏需要使用的键的按下状态。

添加如下代码到 game.py 的#2 小节结束(在你设了屏幕高和宽度之后) **加粗的部分**：

```
# 2 - 初始化PyGame并设置显示窗口
...
screen=pygame.display.set_mode((width, height))
keys = [False, False, False, False]
playerpos=[100,100]
```

这段代码是非常明了。数组 keys 按 WASD 的顺序记录它们的状态。数组的每个元素对应一个键，第一个是 W,第二个是 A 等等。

playerpos 变量定义程序开始绘制游戏角色的起始位置。因为这个游戏将移动游戏角色到不同的位置，设置一个储存角色位置的变量，然后便可以简单地将角色绘制到这个位置。

现在你需要修改现有的代码来绘制角色，使用新的 playerpos 变量，将# 6 的程序:

```
screen.blit(player, (100,100))
```

改为：

```
screen.blit(player, playerpos)
```

接下来，基于哪些按键被按下更新键数组，PyGame 通过添加 event.key 事件使检测按键很容易实现。

在# 8 检测 event.type==pygame.QUIT 之后，添加这些代码(如果有块缩进的话使用与 pygame.QUIT 相同的缩进级) **加粗的部分**：

```
# 8 - 循环游戏、检查是否有新事件
for event in pygame.event.get():
    # check if the event is the X button
    if event.type == pygame.QUIT:
        # if it is quit the game
        pygame.quit()
        exit(0)
    if event.type == pygame.KEYDOWN:
        if event.key == K_w:
            keys[0] = True
        elif event.key == K_a:
            keys[1] = True
        elif event.key == K_s:
            keys[2] = True
        elif event.key == K_d:
            keys[3] = True
    if event.type == pygame.KEYUP:
        if event.key == pygame.K_w:
            keys[0] = False
        elif event.key == pygame.K_a:
```

```
            keys[1] = False
        elif event.key == pygame.K_s:
            keys[2] = False
        elif event.key == pygame.K_d:
            keys[3] = False
```

哇！这里有很多行代码，如果你把它按 If 语句拆分也不是很复杂的。

首先你要检测一是否有键被按下或释放，然后你需要检测哪个键被按下或者释放，如果被按下或者释放的键是你要使用的键，根据键值更新相应的键变量。

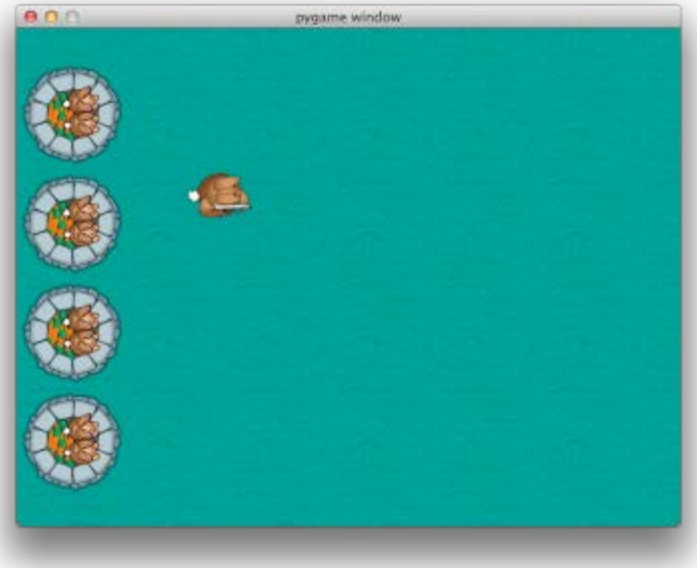
最后，你需要更新 playerpos 变量作为键按下的响应，这太简单了。

将下附代码添加到 game.py 的末尾（使用和 for 循环相同的缩进级）

```
# 9 - Move player
if keys[0]:
    playerpos[1] -= 5
elif keys[2]:
    playerpos[1] += 5
if keys[1]:
    playerpos[0] -= 5
elif keys[3]:
        playerpos[0] += 5
```

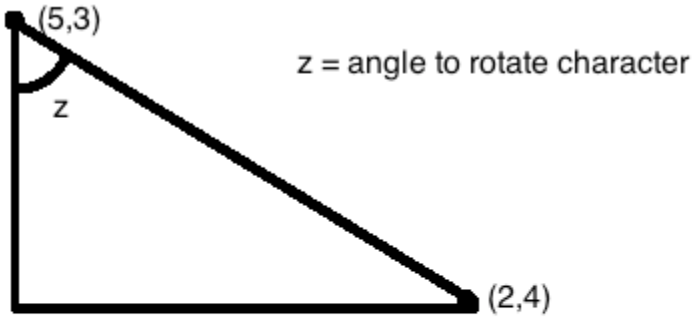
这段代码只是检查哪个键被按下，然后添加或减去游戏角色的 x 或 y 位置(取决于按下的键)来移动游戏角色。

运行这个游戏，你会得到根以前一样的角色。试一下按 WASD 键，哈哈，起效了！



第四步：旋转兔子

是的，你的兔子现在可以随着你的按键移动，但用鼠标来旋转兔子朝向到你选择的方向岂不更酷，所以它不是所有时间都朝向一个方向。使用三角函数来实现，看一下下面的图解：



atan2(diff x, diff y) = z
atan2(5-2, 3-4) = z
atan2(3, -1) = z

上图中，如果(5,3)是兔子的位置，(2,4)是当前鼠标的位置，你可以通过对两点间距离的不同应用 atan2 三角函数获得旋转角度(z)。当然，一旦你知道旋转角，你可以轻松相应地旋转兔子。

如果你对这部分有点困惑，不要担心，你依然可以继续。但这就是你应该在数学课上用功的原因 !:] 在游戏编程时你会一直使用这个东西。

现在你需要将这个原理应用到你的游戏上，为做到这一点，你可以使用 PyGameSurface.rotate(度数)函数， 顺便说一下，记住，Z 值是弧度。

atan2 函数在 Python 的 math 库总，所有先在#1 结尾处增加：

```
import math
```

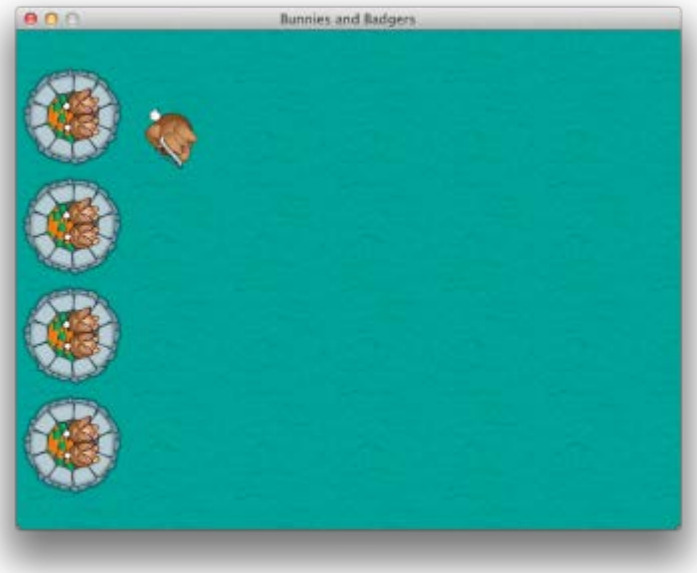
然后，将#6 最后一行（screen.blit(player， playerpos) 这行）替换成下面的代码：

```
# 6.1 - Set player position and rotation
position = pygame.mouse.get_pos()
angle =
math.atan2(position[1]-(playerpos[1]+32),position[0]-(playerpos[0]+26))
playerrot = pygame.transform.rotate(player, 360-angle*57.29)
playerpos1 = (playerpos[0]-playerrot.get_rect().width/2,
              playerpos[1]-playerrot.get_rect().height/2)
screen.blit(playerrot, playerpos1)
```

让我们梳理一下上面代码的基本流程。首先获取鼠标和游戏角色的位置，然后你对两个位置应用 atan2 函数，之后，你将 atan2 返回的弧度转化成度数(将弧度乘以近似 57.29 或 360/2π)。

因为兔子会旋转，它的位置也将会改变。所以现在计算兔子的新位置并把它显示在屏幕。

再次运行这个游戏。如果你只按了“WASD”键，那么这个游戏应该和之前完全一样，但如果你移动你的鼠标兔子也会随之旋转，酷！



第五步：射击吧，兔子!

现在你的兔子已经可以自由活动了，是时候给它添加更多的动作了。让兔子可以用剑射击敌人怎么样？它可不是只温顺的兔斯基！

这一步稍微有点复杂，因为你必须记录所有射出的剑，更新它们的位置、旋转并且在它们飞出屏幕后删除它们。首先，在初始化小节(#2 小节)的结束添加必要的变量：

```
acc=[0,0]
arrows=[]
```

第一个变量记录玩家的射击精度，第二个变量记录所有的剑。精度变量 acc 实际上是一个包含射击数量和命中獾次数的列表。最后我们可以使用这些信息来计算精度的百分比。

接下来，在 section #3 的末尾加载弓箭图像：

```
arrow = pygame.image.load("resources/images/bullet.png")
```

现在当玩家点击鼠标，弓箭就要射出。在 section #8 的末尾加入以下代码作为新的事件句柄

```
if event.type == pygame.MOUSEBUTTONDOWN:
            position = pygame.mouse.get_pos()
            acc[1] += 1
            arrows.append(
                [math.atan2(position[1] - (playerpos1[1] + 32), position[0] -
(playerpos1[0] + 26)), playerpos1[0] + 32,
                 playerpos1[1] + 32])
```

这些代码检查是否有鼠标点击，如果有它会读取鼠标位置，并根据玩家的旋转和指针的位置计算出弓箭的旋转。这个旋转储存在 arrows 数组里。

接下来你需要真正在屏幕上画出弓箭了。在 section #6.1 之后插入以下代码：

```
# 6.2 - Draw arrows
for bullet in arrows:
    index=0
    velx=math.cos(bullet[0])*10
    vely=math.sin(bullet[0])*10
    bullet[1]+=velx
    bullet[2]+=vely
    if bullet[1]<-64 or bullet[1]>640 or bullet[2]<-64 or bullet[2]>480:
        arrows.pop(index)
    index+=1
    for projectile in arrows:
        arrow1 = pygame.transform.rotate(arrow, 360-projectile[0]*57.29)
        screen.blit(arrow1, (projectile[1], projectile[2]))
```

运用基本的三角函数可以计算出 vely 和 velx。10 是弓箭的速度。if 语句检查弓箭是否飞出边界，如果是则删除该弓箭。第二个 for 语句循环过 arrows 数组并画出相应旋转的弓箭。

试试运行程序。你应该有一只兔兔在你点击鼠标时发射弓箭了。

第六步:拿起武器!獾!

好的,你有了一个城堡和可以移动和射击的英雄。但少了什么东西呢?攻击城堡的敌人!

(炸掉你的城堡！)

在这个步骤中,你将创建几只随机生成并跑向城堡的獾。随着游戏的进展会有越来越多的獾。所以，让我们

做一个列表看看需要做那些工作。

1、把坏家伙们添加到一个数组列表；

2、每帧每帧的更新这个数组并检查他们是否都出现在屏幕上；

3、显示这些坏家伙。

很容易，是吧？:]

首先，把下面这段代码添加到 section #2 后面：

```
badtimer=100
badtimer1=0
badguys=[[640,100]]
```

```
healthvalue=194
```

上面这段代码设置了一个计时器（以及其他一些值）以便每过一段时间在游戏中增加一只新的獾。你每帧每帧的减少 badtimer 直到 0， 然后就生成一个新的獾。

现在把下面这段代码添加到 section #3 后面：

```
badguyimg1  = pygame.image.load("resources/images/badguy.png")
badguyimg=badguyimg1
```

上面代码的第一行类似于前面所有的图片加载代码。第二行代码设置了一个图片副本以便于这个坏家伙能更容易动起来。

下一步，你必须更新并且显示这个坏家伙。把下面代码添加到 section #6.2 之后：

```
# 6.3 - Draw badgers
if badtimer==0:
        badguys.append([640, random.randint(50,430)])
        badtimer=100-(badtimer1*2)
        if badtimer1>=35:
                badtimer1=35
        else:
                badtimer1+=5
index=0
for badguy  in badguys:
        if badguy[0]<-64:
                badguys.pop(index)
        badguy[0]-=7
        index+=1
for badguy  in badguys:
        screen.blit(badguyimg, badguy)
```

仔细的看下这段代码，:] 第一行检查 badtimer 是否为 0，以 badtimer 至今为止已经运行的次数为基础，创建一个新的獾并再次设置 badtimer。第一个 for 循环更新獾的 X 坐标，并检查是否在屏幕上。如果不再屏幕上就删除掉。第二个佛如循环绘制所有獾。

为了在上述代码中使用随机函数，我们必须引入 random 库：

```
    import random
```

最后，在 while 语句后面添加下面一行代码用来让 badtimer 值减 1：

```
    badtimer-=1
```

再次运行游戏来测试上述代码，现在你将看到真正的游戏，你可以发射、移动、转向，而且獾也尝试跑向你。

但等等，为什么不是獾炸毁这座城堡吗？马上搞定！

在＃6.3 小节中每一个循环前的 index += 1 之前添加如下代码：

```
# 6.3.1 - Attack castle
badrect=pygame.Rect(badguyimg.get_rect())
badrect.top=badguy[1]
badrect.left=badguy[0]
if badrect.left<64:
        healthvalue  -= random.randint(5,20)
        badguys.pop(index)
# 6.3.3 - Next bad guy
```

这段代码非常简单，如果獾的 x 坐标值小于 64，则删除这家伙然后随机递减 5 到２０点的城堡的生命值。

（在后面你可以显示生命值到屏幕上）

现在你构建并运行程序，你将看到一堆獾攻击城堡后消失。虽然你不能看到但獾确实降低了城堡的生命值。

第七步：獾与箭的碰撞

现在獾可以攻击你的城堡但你的箭却对它们没有任何效果！兔子该如何保护自己的家园？

是时候让箭可以杀死獾，你才可以保护你的城堡赢得游戏！首先，你必须循环检查每个坏蛋，在检查时以

需要循环检查所有的箭是不是和獾碰撞了。如果箭碰到了獾则删除箭和獾，并且递增命中值。

在#6.3.1 小节后添加如下代码：

```
#6.3.2 - Check for collisions
index1=0
for bullet  in arrows:
        bullrect=pygame.Rect(arrow.get_rect())
        bullrect.left=bullet[1]
        bullrect.top=bullet[2]
        if badrect.colliderect(bullrect):
                acc[0]+=1
                badguys.pop(index)
                arrows.pop(index1)
        index1+=1
```

这段代码里有一个值得注意的地方，if 语句后是 PyGame 内建的检查两个矩形是否相交的函数。其他语句

只是按上面解释运行。

如果你现在运行程序，你会发现已经可以射击并杀死獾了。

第八步：添加生命值和时钟显示

游戏现在进展良好，有攻击者和守卫者。现在你还需要一种方式显示分数，看看这些兔子干得怎么样。

最简单的方式是添加一个ＨＵＤ(平视显示器)显示当前城堡的健康状况。也可以添加一个时钟显示城堡被保护了多久。

首先添加时钟，在#7 小节前添加如下代码：

```
# 6.4 - Draw clock
font  = pygame.font.Font(None,  24)
survivedtext  = font.render(str((90000-pygame.time.get_ticks())/60000)+":"+str((90000-
textRect  = survivedtext.get_rect()
textRect.topright=[635,5]
screen.blit(survivedtext, textRect)
```

上面代码简单的使用 PyGame 默认字体创建字体并设置尺寸为 24。然后使用字体渲染时间到表面上。之后广西被定位并绘制到屏幕上。

下一步添加生命条。在绘制生命条之前你需要载入它的背景图。添加如下代码到#3 小节结束：

```
healthbar  = pygame.image.load("resources/images/healthbar.png")
health  = pygame.image.load("resources/images/health.png")
```

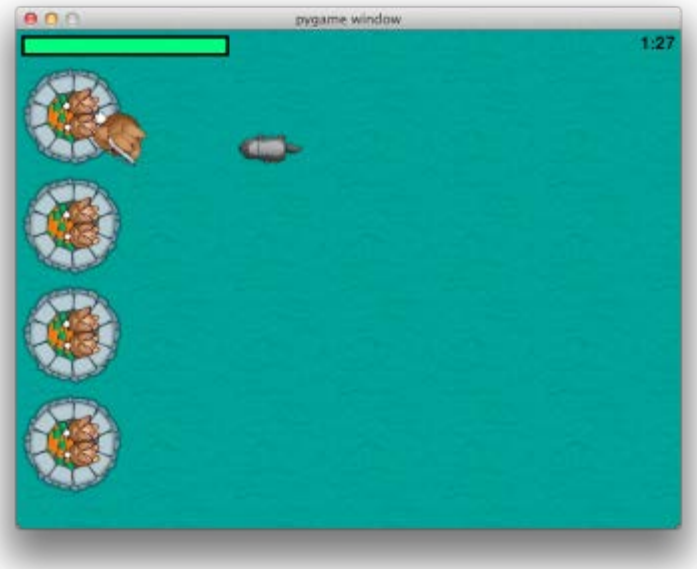
第一个是红色图用来铺满生命条。第二个绿色图显示当前生命值。

现在添加如下代码到#6.4 节（你上一步添加的）之后来将生命条画到屏幕上：

```
# 6.5 - Draw health bar
screen.blit(healthbar, (5,5))
for health1  in range(healthvalue):
        screen.blit(health, (health1+8,8))
```

上面的代码首先画出全红色的生命条，然后根据城堡的剩余的生命绘制绿色生命条。

现在你构建并运行程序，你可以看到时钟和生命条。



第九步：判断输赢

这是什么？如果你玩得足够久，即使你的命令值降到了 0 ，游戏还是继续！不仅仅如此，你也还可以继续射击獾。这应该行不通的，但现在就是这样。你需要添加某种赢/输形式的场景让这个游戏值得玩。

所以我们现在添加输赢条件和各自的画面。你可以退出主游戏循环并进入输赢界面的循环。在输赢循环中，你可以检查用户是输了还是赢了，并显示相应的界面。

下面是一个判断输赢的基本方法：

如果超过了时间(90000 毫秒或 90 秒)：

- 停止运行游戏
- 设置游戏的结果为赢了

如果城堡被摧毁了：

- 停止运行游戏
- 设置游戏结果为输了

计算准确度的方法。

注：使用 acc[0]*1.0 只是将 acc[0]转换成 float 浮点类型，如果你不这么转换，则除法操作将返回一个整数值比如 1 或 2而不是 1.5。

将下面的代码加到 game.py 末尾：

```
        #10 - Win/Lose check
        if pygame.time.get_ticks()>=90000:
                running=0
                exitcode=1
        if healthvalue<=0:
                running=0
                exitcode=0
        if acc[1]!=0:
                accuracy=acc[0]*1.0/acc[1]*100
        else:
                accuracy=0
# 11 - Win/lose display
if exitcode==0:
        pygame.font.init()
        font  = pygame.font.Font(None,  24)
        text  = font.render("Accuracy: "+str(accuracy)+"%",  True, (255,0,0))
        textRect  = text.get_rect()
        textRect.centerx  = screen.get_rect().centerx
        textRect.centery  = screen.get_rect().centery+24
        screen.blit(gameover, (0,0))
        screen.blit(text, textRect)
else:
        pygame.font.init()
        font  = pygame.font.Font(None,  24)
        text  = font.render("Accuracy: "+str(accuracy)+"%",  True, (0,255,0))
        textRect  = text.get_rect()
        textRect.centerx  = screen.get_rect().centerx
        textRect.centery  = screen.get_rect().centery+24
        screen.blit(youwin, (0,0))
        screen.blit(text, textRect)
while 1:
        for event  in pygame.event.get():
                if event.type == pygame.QUIT:
                        pygame.quit()
                        exit(0)
        pygame.display.flip()
```

这是截至目前最长的代码块，但不是最复杂的。

第一个 if 语句检查是否到时间，第二个检查城堡是否被摧毁，第三个计算你的准确率。之后，一个快速的 if 语句检查你是胜利还是失败了，并显示正确的图像。

当然，如果你想显示胜利与失败的屏幕图像，那么这些图像必须先被加载。将下面的代码添加到# 3 结束的地方：

```
gameover  = pygame.image.load("resources/images/gameover.png")
youwin  = pygame.image.load("resources/images/youwin.png")
```

另一点，更改#4 处：

```
# 4 - keep looping through
while 1:
        badtimer-=1
```

更改为：

```
# 4 - keep looping through
running  = 1
exitcode  = 0
while running:
        badtimer-=1
```

running 变量跟踪游戏是否结束，exitcode 变量跟踪玩家是赢了还是输了。

再次运行游戏，现在你可以尝试胜利或死亡！酷！:]

第 10 步: 免费的音乐和声音效果!

这个游戏看起来相当棒，可是有声音吗？是不是太安静了点呢？增加一点点音效会让整个游戏的感觉变得更棒。

PyGame 加载并播放声音的方法非常简单。首先你需要初始化混音器，代码如下：

```
pygame.mixer.init()
```

然后加载要播放的声音，并设置音量大小：

```
# 3.1 - Load audio
hit  = pygame.mixer.Sound("resources/audio/explode.wav")
enemy  = pygame.mixer.Sound("resources/audio/enemy.wav")
shoot  = pygame.mixer.Sound("resources/audio/shoot.wav")
hit.set_volume(0.05)
enemy.set_volume(0.05)
shoot.set_volume(0.05)
pygame.mixer.music.load('resources/audio/moonlight.wav')
pygame.mixer.music.play(-1,  0.0)
pygame.mixer.music.set_volume(0.25)
```

最上代码中大多数是简单加载音频文件和配置的音频音量，但要注意 pygame.mixer.music.load 这一行——这一行加载游戏的背景音乐，下一行设置背景音乐一直重复。

注意音频配置。:] 现在，你所需要做的就是按需要播放各种声音效果。像下面每段注释代码那样做：

```
# section 6.3.1 after if badrect.left<64:
hit.play()
# section 6.3.2 after if badrect.colliderect(bullrect):
enemy.play()
# section 8, after if event.type==pygame.MOUSEBUTTONDOWN:
shoot.play()
```

多次运行游戏，你会发现你现在有碰撞的背景音乐和射击的声音效果，游戏让人感觉更逼真了！:]