University of Hull
Department of Computer Science

Writing Games with Pygame

Vsn. 1.0 Rob Miles 2014

Week 5: Game Structure

These notes describe the practical elements of the course. They are to be used in conjunction with the slide decks.

Making Crackers Rotate

Last time we made a game called **`ChasedByCrackers`**. This time we are going to look at how we would change the behaviour of one of the elements in the game.

Getting started

You will need to find the blank game framework that contains a game and some classes that represent items in the game. You will be copying and modifying elements in the framework to make the game that you want.

You can download the framework from:

`http://www.robmiles.com/s/Week5Practical1.zip`

Before you go any further; perform the following:

1. Log into your system with the username and password that you know and love from the past.
2. Start the Idle environment.
3. Copy the game framework zip file from the web address above and unzip it into a folder on your machine.
4. Open the file **`ChasedByCrackers.py`** and run the program. You should see the game menu above. If you press the G key the game should change to the game screen.
5. You can move the cheese around the screen by pressing the arrow keys. The cracker will chase the cheese .If it gets close enough to touch the cheese the game ends.

Your boss is quite impressed by your program but she wants the crackers to rotate. Later she might ask for more of them. You are not sure that this will add much to the gameplay, but since she is paying your mortgage you agree to implement the changes.

You are going to do this by modifying the existing objects and changing the way that they work. Something, somewhere, will be using the rotate behaviours provided by PyGame to draw a rotated cracker.

An Important Question

It is very easy to rotate an object in Pygame. But we need to consider who is in charge of the rotation of a cracker. There are two ways to do this:

- The game could rotate the cracker image
- The cracker could rotate the cracker image

Which makes most sense?

Cohesion and Responsibility

The way I see it, the cracker should actually rotate the image. The game should not have to know how the rotation is performed. It should just tell the cracker that it needs to rotate, and the cracker should take care of it. The rotation speed will be given in degrees per second. When the cracker is constructed the game will set an initial rotation speed for that cracker. The cracker will go back to that rotation speed when the game is reset. We did something very similar in the last practical session, when we added a cracker to the game by modifying **`PhysicsMover`** class. Now we are going to make changes to the Cracker class.

Editing the Cracker class

When a cracker is created it is set with some initial data. This happens in the **`__init__`** method of the class:

```
# Initialise the Cracker object
#    position - the initial position of the player
#    limit - a tuple giving the size of the screen
#    speed - a tuple giving the speed of movement in pixels/second
#    image - the image to be drawn
#    width - the x size of the image in pixels
#    target - the object the cracker is chasing
def __init__(self, position, limit, speed, image, width, target):

    # Set the target for the chase
    self.target = target
    ...
```

I always put a comment in before the **`__init__`** method that gives the name and purpose of each parameter. Remember that a parameter feeds information into a method call, in the code above we are setting a cracker up with a particular position, width, image and movement speed.

We need to add another parameter to the **`__init__`** method.

```
# Initialise the Cracker object
#    position - the initial position of the player
#    limit - a tuple giving the size of the screen
#    speed - a tuple giving the speed of movement in pixels/second
#    rotation - a value giving the rotation speed in degrees/second
#    image - the image to be drawn
#    width - the x size of the image in pixels
#    target - the object the cracker is chasing
def __init__(self, position, limit, speed, rotation, image, width, target):

    # Set the rotation speed

    self.rotation = rotation

    # Set the target for the chase
    self.target = target
```

Before you go any further; perform the following:

1. Open **Cracker.py** and find the **__init__** method.
2. Change the method so that it now has a rotation parameter.
3. Save the **Cracker.py** file. (**VERY IMPORTANT)**
4. Run the program. Does it work? Why not?

Constructing a Rotating Cracker instance

The program will not presently build because we have broken a contract. The **__init__** method in the Cracker class is expecting to be given seven pieces of data. However, we are not supplying a rotation value, causing Python to get very confused and upset:

```
self.cracker =
Cracker((500,500),self.windowSize,(50,50),crackerImage,30,
self.gameCheese)
TypeError: __init__() missing 1 required positional argument:
'target'
```

It things that the one that is missing is the one on the end (the target) but actually it is the rotation that is not there. You need to be aware of issues like this (if you aren't already). The system will report an error where it detects it, not where the error is actually located.

We need to add a value to the statement that constructs the cracker:

```
self.cracker = Cracker((500,500),self.windowSize,(50,50),crackerImage,30,
self.gameCheese)
```

This is the original cracker construction code. The cracker is placed at (500,500). It is bounded by the window, it moves at 50 pixels per second in the X and Y directions, it is based on the crackerImage it is 30 pixels wide and it chases the cheese. We need to add the rotation speed to the call of the constructor.

```
self.cracker = Cracker((500,500),self.windowSize,(50,50),360,crackerImage,30,
self.gameCheese)
```

Note that the above call has an additional parameter, the value of 360, which should result in the cracker rotating once per second.

Before you go any further; perform the following:

1. Open **ChasedByCrackers.py** and find the **__init__** method.
2. Change the call of the **Cracker** constructor to include the value of 360 above.
3. Run the program. It should run, but the cracker will not rotate. However, it now has a rotation value that is passed into it when it starts running.

The Cracker class now needs to do the rotating. This is actually quite complex because the way that PyGame rotates things is a bit hard to use. The rotation is performed about a point that varies according to the dimensions of the sprite and the distance being rotated. The result is that when a sprite is rotated it needs to be repositioned on the screen about the original centre.

To save you from having to spend too much time on this I've created a completed version of the lab here:

http://www.robmiles.com/s/Week5Practical1Complete.zip

Before you go any further; perform the following:

1. Download the completed version.
2. Unzip the file and show the rotating cracker.

You show the game to your boss and she is quite impressed, but she would like more crackers and she would like them to rotate in different directions.

Before you go any further; perform the following

1. Use a list of crackers to hold 10 crackers that rotate at different speeds and directions. If you use a negative number for the rotation speed you can make the cheese spin in the opposite direction.

Rob Miles June. 2014