Writing Games with Pygame

Wrestling with Python

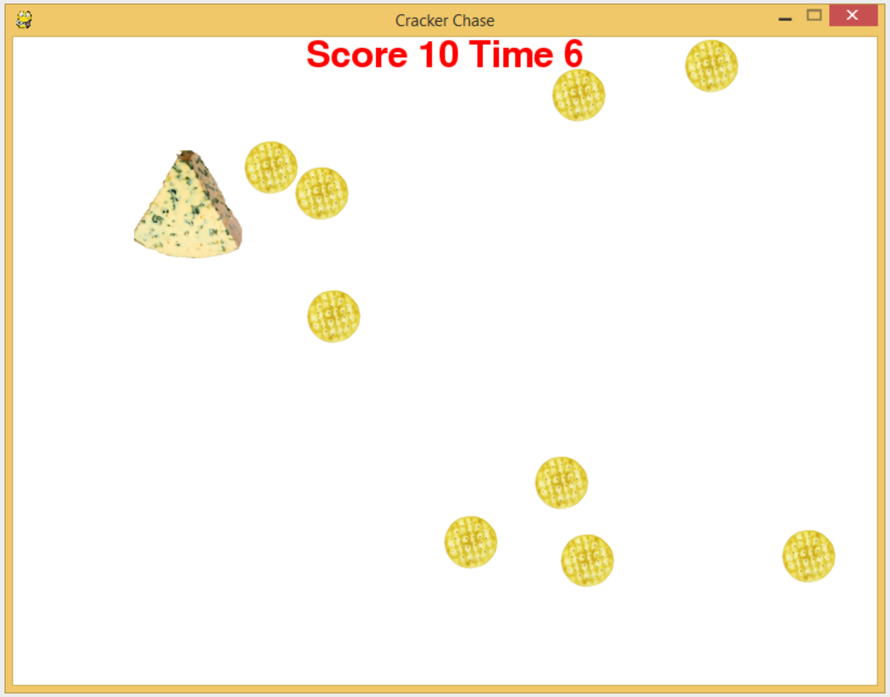
Games and Classes

- A Complete Game
- Games and Objects

CrackerChase

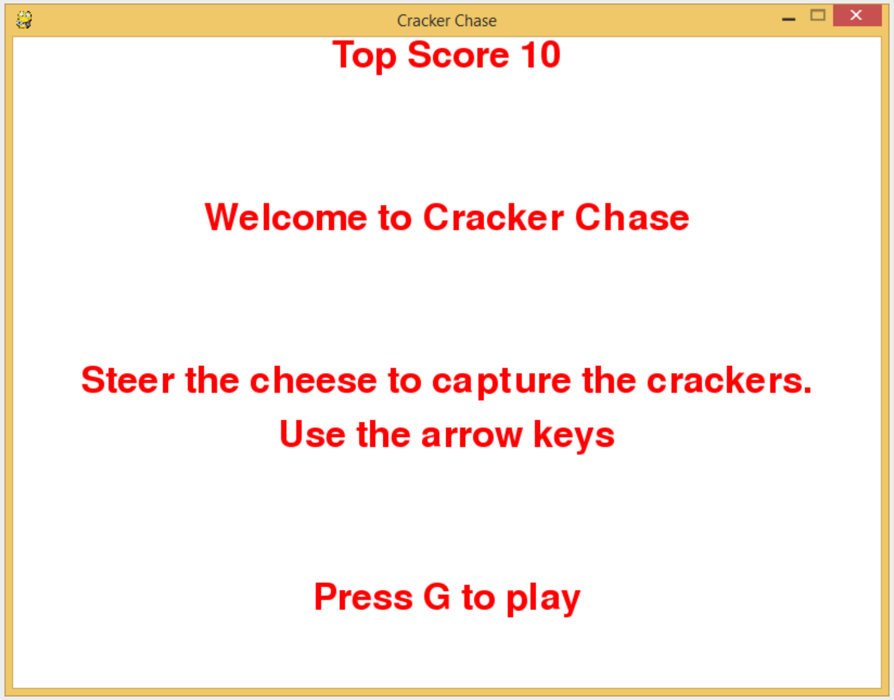
CrackerChase Game

- This is a very simple game
- Players must steer the cheese around the screen and "eat" the crackers



CrackerChase Game

- The game has a start screen and a game screen
- When the game ends the player is returned to the start screen



Top Score 10

Welcome to Cracker Chase

Steer the cheese to capture the crackers.
Use the arrow keys

Press G to play

Building CrackerChase

- To create a game that can be extended and customised we need to change the arrangement of the elements
- For this game we will need the cheese for the player to control and some targets
- Now we need to look at how we can create a Player object that the user can steer

Games and Objects

Objects and Players

- An object brings together data and behaviours to manage a part of a system
  - The Player will have a position on the screen
  - It will also have behaviours such as draw, update and reset
- We are going to see how to create a player object

Creating a Player

```
self.gameCheese =
       Player((0,0),self.windowSize,cheeseImage)
```

- The `gameCheese` variable is a member of the game class
- `gameCheese` is constructed at the start of the game
- It is given several setting values when it is constructed

Creating a Player

```
self.gameCheese =
          Player((0,0),self.windowSize,cheeseImage)
```

- This is the initial position of the player on the screen
- We are placing them on the top, left hand corner
- The player will be moved to this position at the start of each game

Creating a Player

```
self.gameCheese =
       Player((0,0),self.windowSize,cheeseImage)
```

- This is the size of the area where the cheese will be drawn
- The cheese will not allow itself to be placed outside this area
- This is a tuple that is set when the game starts and contains an x and a y value

Creating a Player

```
self.gameCheese =
       Player((0,0),self.windowSize,cheeseImage)
```

- This is the image used to draw a picture of cheese on the screen
- If we make different player objects (perhaps for a multi-player game) we can use different images

Creating a Player

```
def __init__(self, position, limit, image):
    self.resetPosition = position
    self.image = image
    ...
```

- The constructor method inside the player takes the settings that are passed into it and uses these to configure this player instance

A Player Object in CrackerChase

PRACTICAL BREAK 1

UNIVERSITY OF Hull

Object Communication

- At the moment we just now how to add the object to our game and use it
- Now we are going to find out how to send commands to the object as the game is played
- We are going to steer the cheese

Steering the Cheese

```
for e in pygame.event.get():
    if e.type == pygame.KEYDOWN:
        if e.key == pygame.K_w:
            cheeseMovingUp = True
    if e.type == pygame.KEYUP:
        if e.key == pygame.K_w:
            cheeseMovingUp = False
if cheeseMovingUp:
    cheeseY = cheeseY-cheeseYSpeed
```

- At the end of the last session we had discovered how to steer cheese around the screen

Steering the Cheese

```
for e in pygame.event.get():
    if e.type == pygame.KEYDOWN:
        if e.key == pygame.K_w:
            cheeseMovingUp = True
    if e.type == pygame.KEYUP:
        if e.key == pygame.K_w:
            cheeseMovingUp = False
if cheeseMovingUp:
    cheeseY = cheeseY-cheeseYSpeed
```

- This loop works through all the pygame events

UNIVERSITY OF Hull

Steering the Cheese

```
for e in pygame.event.get():
    if e.type == pygame.KEYDOWN:
        if e.key == pygame.K_w:
            cheeseMovingUp = True
    if e.type == pygame.KEYUP:
        if e.key == pygame.K_w:
            cheeseMovingUp = False
if cheeseMovingUp:
    cheeseY = cheeseY-cheeseYSpeed
```

- If the event is a KEYDOWN it will move the cheese UP if the key pressed is a w

Steering the Cheese

```
for e in pygame.event.get():
    if e.type == pygame.KEYDOWN:
        if e.key == pygame.K_w:
            cheeseMovingUp = True
    if e.type == pygame.KEYUP:
        if e.key == pygame.K_w:
            cheeseMovingUp = False
if cheeseMovingUp:
    cheeseY = cheeseY-cheeseYSpeed
```

- This flag holds the vertical movement state of the cheese

Steering the Cheese

```
for e in pygame.event.get():
    if e.type == pygame.KEYDOWN:
        if e.key == pygame.K_w:
            cheeseMovingUp = True
    if e.type == pygame.KEYUP:
        if e.key == pygame.K_w:
            cheeseMovingUp = False
if cheeseMovingUp:
    cheeseY = cheeseY-cheeseYSpeed
```

- The flag is set when the key is and cleared when the key is released

Steering the Cheese

```
for e in pygame.event.get():
    if e.type == pygame.KEYDOWN:
        if e.key == pygame.K_w:
            cheeseMovingUp = True
    if e.type == pygame.KEYUP:
        if e.key == pygame.K_w:
            cheeseMovingUp = False
if cheeseMovingUp:
    cheeseY = cheeseY-cheeseYSpeed
```

- When the cheese is moving up the Y position is updated by the speed value

Steering a Player in a game

- When we want to steer a Player object around the screen we need to call methods in that object to start and stop its move behaviour
- The structure of the keyboard management is the same, but what we do when we detect key events must change

Steering a Player object

```
for e in pygame.event.get():
    if e.type == pygame.KEYDOWN:
        if e.key == pygame.K_w:
            self.gameCheese.StartMoveUp()
    if e.type == pygame.KEYUP:
        if e.key == pygame.K_w:
            self.gameCheese.StopMoveUp()
```

- This code steers the player in a game
- It runs inside the game, and controls a player object

Steering a Player object

```
for e in pygame.event.get():
    if e.type == pygame.KEYDOWN:
        if e.key == pygame.K_UP:
            self.gameCheese.StartMoveUp()
    if e.type == pygame.KEYUP:
        if e.key == pygame.K_UP:
            self.gameCheese.StopMoveUp()
```

- In our game the instance of the `Player` object is called `gameCheese`
- The object is created when the game starts running (more on this later)

Steering a Player object

```
for e in pygame.event.get():
    if e.type == pygame.KEYDOWN:
        if e.key == pygame.K_UP:
            self.gameCheese.StartMoveUp()
    if e.type == pygame.KEYUP:
        if e.key == pygame.K_UP:
            self.gameCheese.StopMoveUp()
```

- The method `StartMoveUp` is called when we want the player to start moving up

Steering a Player object

```
for e in pygame.event.get():
    if e.type == pygame.KEYDOWN:
        if e.key == pygame.K_UP:
            self.gameCheese.StartMoveUp()
    if e.type == pygame.KEYUP:
        if e.key == pygame.K_UP:
            self.gameCheese.StopMoveUp()
```

- The method `StartMoveUp` is called when we want the player to start moving up
- The method `StopMoveUp` is called when we want the player to stop moving up

Methods in the Player object

```
def StartMoveUp(self):
    self.movingUp = True
def StopMoveUp(self):
    self.movingUp = False
```

- These are the methods in the `Player` class
- They set flags in the `Player` to tell it what its movement state is

Updating Player position

```
if self.movingUp:
    self.position[1] = self.position[1] -
        (self.movementSpeed[1]*deltaTime)
```

- When the `Player` is asked to update where it is on the screen it will use the flags that have been set to tell it which way to move

THIS IS CONFUSING

THIS IS CONFUSING

..but we do it for a reason

Games and Objects

- When we have a game with hundreds of different kinds of things on the screen we need a way of making sure that we can work with them with out getting confused
- We could put them all in one big lump of code but it would be a nightmare to work with and very hard to reuse the code for other games

Games and Orchestras

- Thinking of the game as an orchestra, with a whole bunch of musicians (objects) being controlled by a conductor (which is also an object)
- I was also thinking of using a Football Team as an analogy, but that might not work so well….

Player and Game Responsibilities

- The `Game` object is in charge of getting the input from the user and deciding what the input actually means
- The `Player` object is in charge of moving around the screen
- The `Game` will tell the `Player` when to start moving, and when to stop

How the Player moves up

1. User presses the Up key
2. Pygame generates a KEYDOWN event for the key
3. The game finds this and calls `StartMoveUp` on the `Player` object
4. Later, when the `Player` is asked to `Update` itself, it will move up

The Player Update method

```
def update(self, deltaTime):
    if self.movingUp:
        self.position[1] = self.position[1]
         - (self.movementSpeed[1]*deltaTime)
```

- While the game is being played it will be updating the Player object
- The Player object will update itself with the movement that was supplied

UNIVERSITY OF Hull

Delta Time

```
def update(self, deltaTime):
    if self.movingUp:
        self.position[1] = self.position[1]
         - (self.movementSpeed[1]*deltaTime)
```

- When the update is called the player is told how long it was since the last update
- It can then use this to control the speed of the player

Position Values

```
def update(self, deltaTime):
    if self.movingUp:
        self.position[1] = self.position[1]
         - (self.movementSpeed[1]*deltaTime)
```

- The player holds coordinates in a list which contains two values, x and y
- The y coordinate is held in element 1, the x coordinate is held in element 0

Controlling the Player Object

PRACTICAL BREAK 2

Next Time

- Next time we will see how we can complete the game by adding sound and a start screen, along with targets to chase