Writing Games with Pygame

Wrestling with Python

UNIVERSITY OF Hull

Making a Game

- Drawing Images
- Making Images Move
- A game as an object
- Creating a “game loop”
- Giving the Player Control of the Game

Drawing an Image

Drawing Cheese

```
def drawCheese(self):
    pygame.init()
    width=800
    height=600
    size = (width,height)
    surface = pygame.display.set_mode(size)
    cheeseImage = pygame.image.load("cheese.png")
    cheesePos = (40,60)
    surface.blit(cheeseImage, cheesePos)
    pygame.display.flip()
```

- This is what we finished with last week
- It draws some cheese on the screen

Drawing Cheese

```
def drawCheese(self):
    pygame.init()
    width=800
    height=600
    size = (width,height)
    surface = pygame.display.set_mode(size)
    cheeseImage = pygame.image.load("cheese.png")
    cheesePos = (40,60)
    surface.blit(cheeseImage, cheesePos)
    pygame.display.flip()
```

- This is the part that actually draws the cheese

Drawing Cheese

```
cheesePos = (40,60)
surface.blit(cheeseImage, cheesePos)
```

- The first statement sets the position of the draw

Drawing Cheese

```
cheesePos = (40,60)
surface.blit(cheeseImage, cheesePos)
```

- The first statement sets the position of the draw
- The second statement does the drawing

Cheese Position

```
cheeseX = 40
cheeseY = 60
cheesePos = (cheeseX,cheeseY)
surface.blit(cheeseImage, cheesePos)
```

- This code does exactly the same as the previous statements
- But the X and Y positions of the cheese are now variables

Cheese Position Variables

```
cheeseX = 40
cheeseY = 60
cheesePos = (cheeseX,cheeseY)
surface.blit(cheeseImage, cheesePos)
```

- This code does exactly the same as the previous statements
- But the X and Y positions of the cheese are now variables

Moving Cheese

```
for i in range(1,50):
    cheeseX = cheeseX + 1
    cheeseY = cheeseY + 1
    cheesePos = (cheeseX,cheeseY)
    surface.blit(cheeseImage, cheesePos)
    pygame.display.flip()
```

- What does this code do?

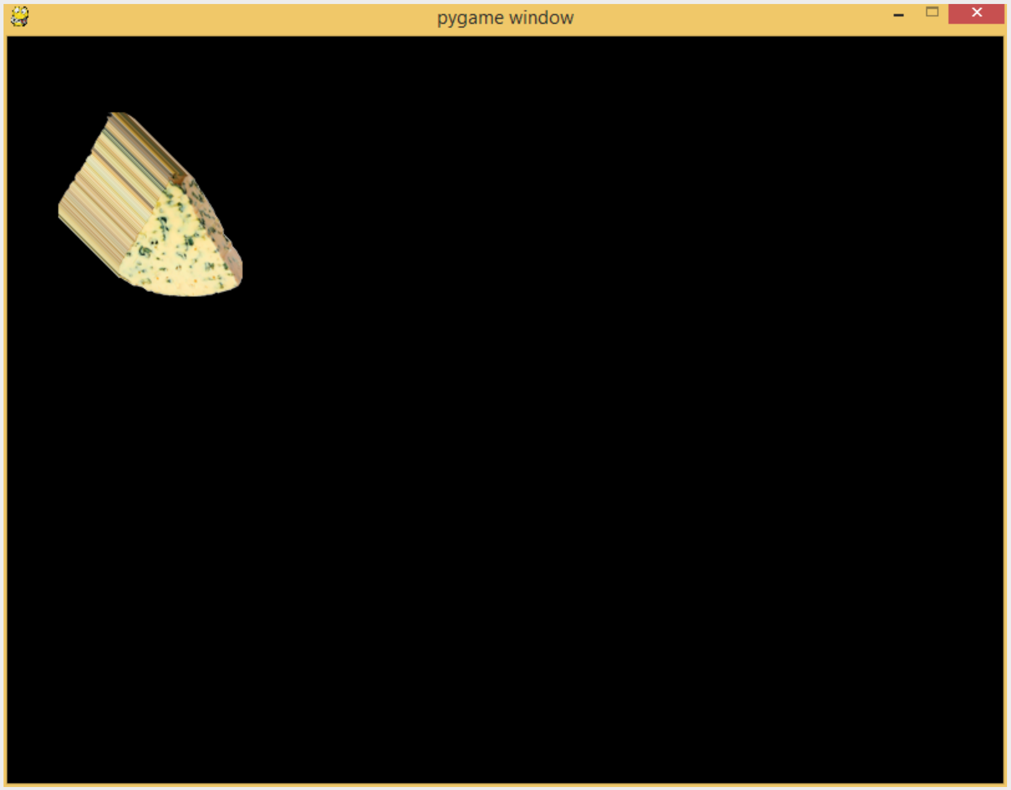
Moving Cheese

```
for i in range(1,50):
    cheeseX = cheeseX + 1
    cheeseY = cheeseY + 1
    cheesePos = (cheeseX,cheeseY)
    surface.blit(cheeseImage, cheesePos)
    pygame.display.flip()
```

- It draws the cheese 50 times, each time moving the position by 1 pixel

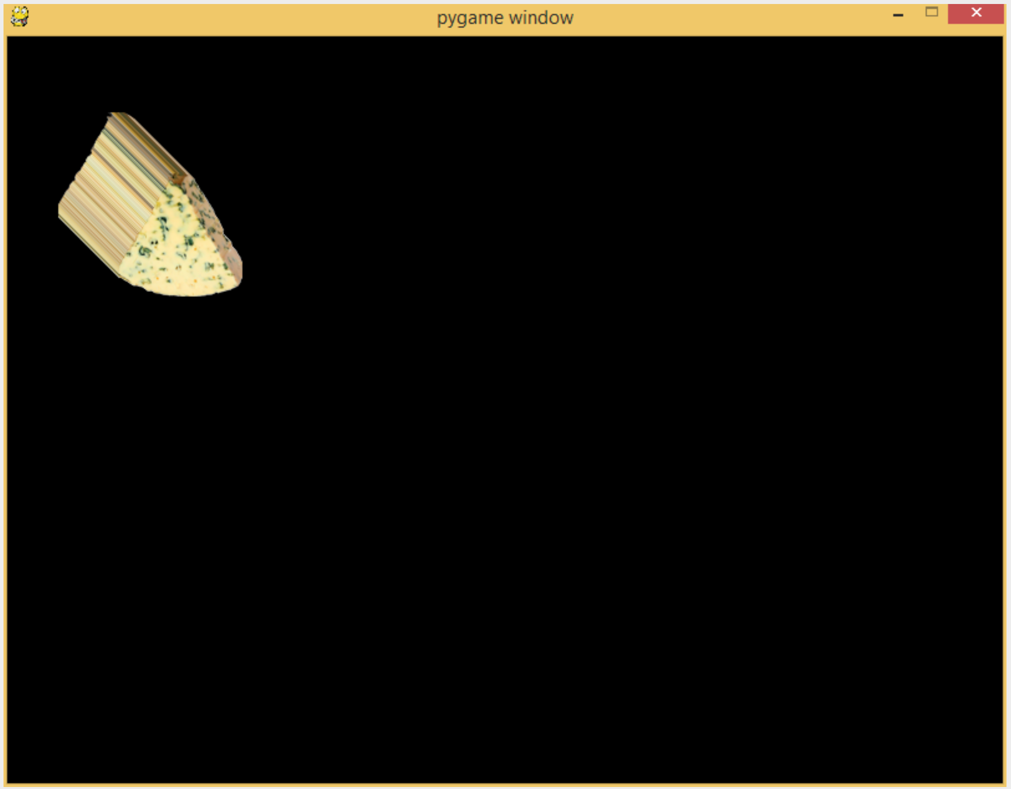
Unsuccessful Cheese Movement 1

- This didn't work very well
- Why?



Unsuccessful Cheese Movement 1

- This didn't work very well
- Why?
- We need to clear the screen between each draw



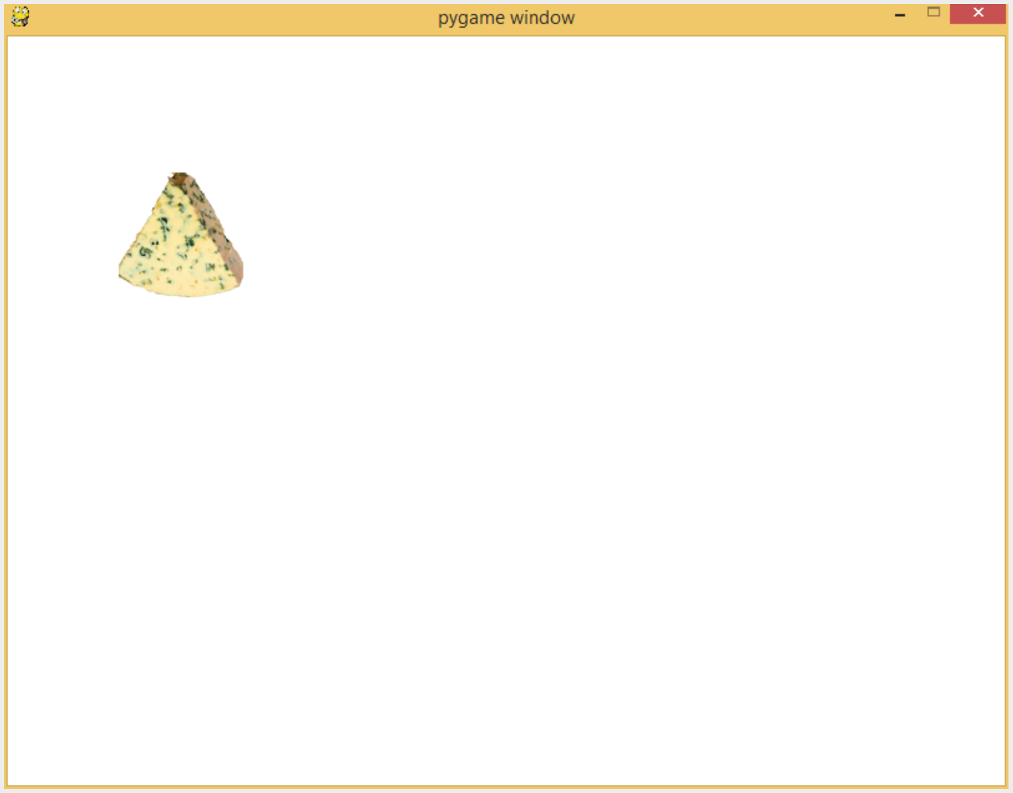
Moving Cheese

```
for i in range(1,50):
    surface.fill((255,255,255))
    cheeseX = cheeseX + 1
    cheeseY = cheeseY + 1
    cheesePos = (cheeseX,cheeseY)
    surface.blit(cheeseImage, cheesePos)
    pygame.display.flip()
```

- This version clears the screen before each draw

Unsuccessful Cheese Movement 2

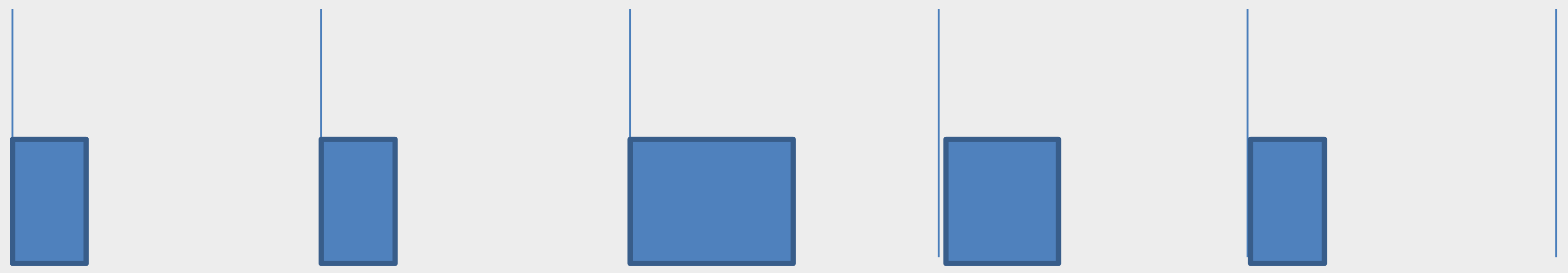
- Now we have the cheese in the final position
- But we didn't see it move there



Speed and Frames

- Python will normally run our programs as fast as it can – this is usually what we want
- However, for a game we need a way to make the program pause between frames
- Movies run at 24 frames per second
- Video games run at 60 frames per second

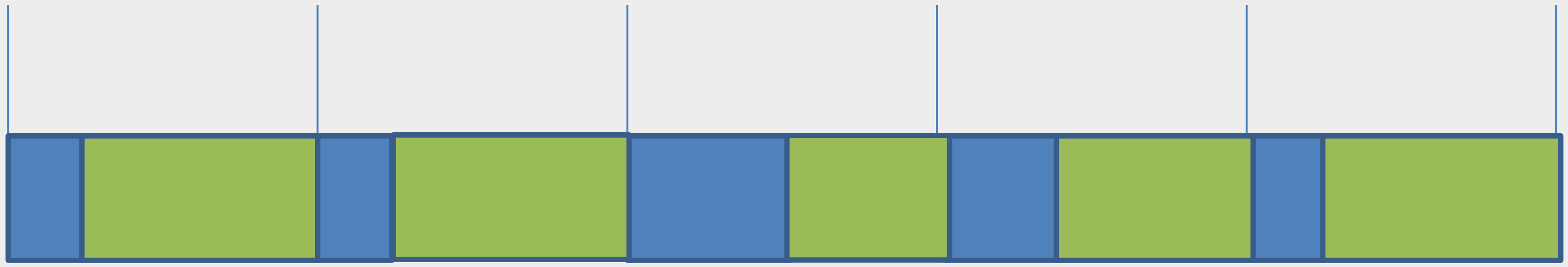
Games and Ticks



- Above you can see the clock ticks which are at regular intervals as the game runs
- The shaded blocks represent the time that the game spends drawing the screen
- Not all the blocks are the same size

UNIVERSITY OF Hull

Games and Ticks



- The green areas show the time that the game must wait after drawing each frame before drawing the next one
- This is so that the frame rate the user sees is constant

The Pygame Clock

- The Pygame system has a clock built into it
- It can use this clock to manage timing for the game when it runs
- It provides a tick method that will work out how long it needs to wait (the length of the green bit) to provide a particular frame rate

Moving Cheese

```
for i in range(1,50):
    surface.fill((255,255,255))
    cheeseX = cheeseX + 1
    cheeseY = cheeseY + 1
    cheesePos = (cheeseX,cheeseY)
    surface.blit(cheeseImage, cheesePos)
    pygame.display.flip()
    pygame.time.Clock().tick(30)
```

- This is the statement that makes the game pause between frames

Moving Cheese

```
for i in range(1,50):
    surface.fill((255,255,255))
    cheeseX = cheeseX + 1
    cheeseY = cheeseY + 1
    cheesePos = (cheeseX,cheeseY)
    surface.blit(cheeseImage, cheesePos)
    pygame.display.flip()
    pygame.time.Clock().tick(30)
```

- The `time` member of the pygame class contains a method called `Clock()`

Moving Cheese

```
for i in range(1,50):
    surface.fill((255,255,255))
    cheeseX = cheeseX + 1
    cheeseY = cheeseY + 1
    cheesePos = (cheeseX,cheeseY)
    surface.blit(cheeseImage, cheesePos)
    pygame.display.flip()
    pygame.time.Clock().tick(30)
```

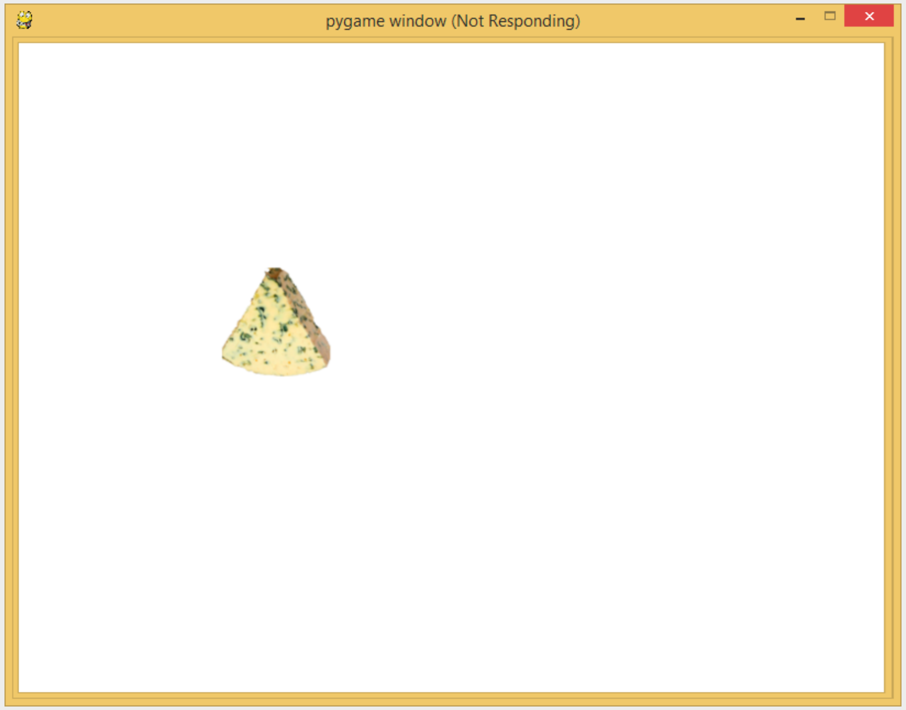
- The `Clock()` method returns a reference to the pygame clock

Movement and Frame Rate

- If every tick is the same size we can move the game objects by the same amount in each frame and they will appear to move at a fixed speed
- If ticks are different sizes (perhaps we are drawing as fast as we can) then the amount an object moves in each frame will depend on the time since the last tick

Pygames that get stuck

- You might have noticed that your game “gets stuck” sometimes
- This is because it is not handling events properly



UNIVERSITY OF Hull

Input and Programs

- Up until now every program that we have written has stopped and waited for the user when input is required
- Now we are going into a situation where the outside world will “attract the attention” of our program by raising events

University of Hull

Events

- While the Pygame is running it is managing incoming events
  - Mouse movement and buttons
  - Keys pressed and released
  - Joystick movement
- Each of these events is placed in a queue
- If the queue gets too large the program may stick

Using Events

```
for e in pygame.event.get():
    if e.type == pygame.KEYDOWN and
                e.key == pygame.K_ESCAPE:
    return
```

- This will exit the game method and stop the game if the user presses the escape key when the game is running

Using Events

```
for e in pygame.event.get():
    if e.type == pygame.KEYDOWN and
                    e.key == pygame.K_ESCAPE:
    return
```

- The `for` loop is going to work through a list of events that `pygame` has detected
- The variable `e` will be set to each event in turn – just like for loops we have already seen

Using Events

```
for e in pygame.event.get():
   if e.type == pygame.KEYDOWN and
               e.key == pygame.K_ESCAPE:
   return
```

- The get method returns a list of events that have been detected since it was last called

Using Events

```
for e in pygame.event.get():
   if e.type == pygame.KEYDOWN and
                  e.key == pygame.K_ESCAPE:
   return
```

- An event has a `type` attribute that tells the game what type of event it is
- If the type value is set to `pygame.KEYDOWN` this indicates a key press

Using Events

```
for e in pygame.event.get():
   if e.type == pygame.KEYDOWN and
               e.key == pygame.K_ESCAPE:
       return
```

- If we got a key, check to see if it was the escape key
- If it was the escape key, return from the method

Using the keyboard events

- We can use the keyboard events to steer the cheese around the screen
- This is how a player could control our cheese in the game

Steering the Cheese

```
for e in pygame.event.get():
    if e.type == pygame.KEYDOWN:
        if e.key == pygame.K_ESCAPE:
            return
        if e.key == pygame.K_w:
            cheeseY = cheeseY-1
        if e.key == pygame.K_x:
            cheeseY = cheeseY+1
```

- If the player presses w the cheese moves up
- If the player presses x the cheese moves down

Cheese Steering

PRACTICAL BREAK 2

Next Time

- Next time we will see how to construct create a game class that supports lots of objects on the screen at the same time