Creating the Multiplayer Game ‘Red Light, Green Light’ in Python

Alex Slocombe

University of the Arts, London

Aims and game logic

With the project to make a multiplayer game, I aim to consolidate the knowledge of Python that I have learnt so far with practical usage of the language. Using a range of different tools learnt in the Coding 1 module, I aspire for a game that is dynamic and challenging, but fun for the player. I chose to make the game Red Light, Green Light. The game is a race to the finish line between two players, in which the players mustn't move on the red light or get collide with moving guards, or they would be sent back to the beginning. The first player to achieve 3 points wins the game.

Design and creation process

I began by learning the basics of Pygame and applying this knowledge through the creation of the basic game board. I created a simplistic design that would allow the player to easily understand the happenings on screen. I soon understood the significance of the following two loops:

```python
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
```

These two loops allow for a continuous refresh of all the events happing on screen, whilst the game is still open, allowing for instances such as moving sprites. The for loop allows us to register all events occurring within the game and ends all activity should the player quit.

The next key step was creating the sprites. Key to the main function of the game were the two sprites the player's control. A persistent issue that I experienced with these sprites was a dragging effect, leaving a trail behind any movement. This was amended by changing the order in which my code was written, highlighting to me the importance of order execution.

```
#Background colour
    screen.fill((0,0,0))

    #Drawing game board
    borderHorizontal = pygame.draw.rect(screen, (255, 255, 255),(0, 49, 400,
2))
    borderVertical = pygame.draw.rect(screen, (255, 255, 255),(199, 50, 2,
450))
```

Moving the above code to outside the for loop, above the registration of player key presses allowed me to resolve this issue, as the background then was not influenced by the event of player input. Next, I added the enemy sprites into the game, an important aspect of the project's function as a difficult and competitive game. I chose to randomize the location of these four sprites, to add a change to the game with each round of the game, creating a more dynamic design.

The creation of the traffic lights was crucial to the function of the game, but not without great difficulty. I found an issue with the randomization aspect within the for loop. If the order of execution was not correct, a constant change of the values needed occurred which created an undesirable strobe effect of the lights. I decided to randomize the speed at which the red and green light persisted in order to create surprise and enhance the difficulty. Rather than creating a rhythmic change, a variable speed diverts the player focus from other factors, such as the enemies, to the lights causing more frequent player error. The final code which allowed for good execution is as follows:

```
greenColor = brightGreen
    amberColor = dimAmber
    redColor = dimRed
    if count >= greenTime:
        greenColor = dimGreen
        amberColor = brightAmber
        redColor = dimRed
    if count >= greenTime + 1:
        greenColor = dimGreen
        amberColor = dimAmber
        redColor = brightRed
    if count >= redTime:
        count = 0
        greenTime = random.randint(1,6)
        redTime = greenTime + 1 + random.randint(1,6)
    count += 0.0006
```

Finally, my work needed more of a seamless flow to it. I added a start screen, informing the player of the rules and controls, and an end screen, displaying the result of the game and allowing the players to replay. I also added music that varied in volume with the events of the game, lowering should the lights turn red. This created a more immersive effect within the game and allowed the player another signal, furthering their understanding of how the rules apply in the game. The following conditional statements allowed me to do so:

```
    if redColor == brightRed:
        pygame.mixer.music.set_volume(0.2)
    else:
        pygame.mixer.music.set_volume(1)
```