Programming concepts PR1
Snake game - Python

Denzel Eshun

Fudailahad Khan

Overview

- You control a snake, the goal is to collect 5 lucky blocks. Making direct contact with its own body or the wall, will kill the snake immediately.
- You can attain additional lives by collecting Regular blocks.
- To win the game you have to collect 5 lucky blocks

Game Design

- No classes were pre-identified as in python turtle the turtle object is a premade object with characteristics that are user defined.
- However we knew we would need 4 turtle objects

Mamba, Mamba Extensions, Lucky Blocks & Regular Blocks

- We ended up needing a 5th to create game metrics

Mamba Object

- Attributes – position, heading, colour, shape, cursor visibility
- Behaviours – Forward, backward, right, left, goto, etc

Programming process

Libraries & Modules

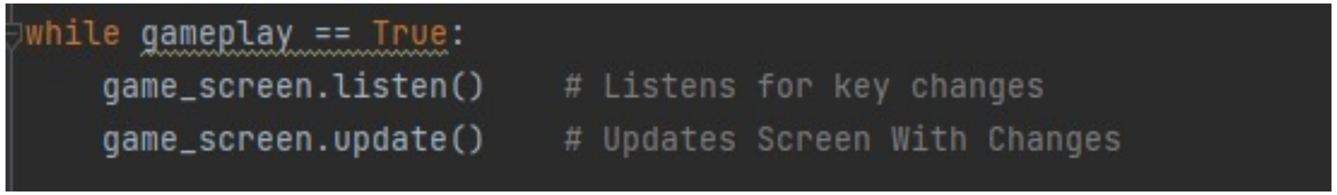
- We used Python Turtle Graphics to run the Game
- Random Module To allow for generating random coordinates
- OS module to allow sound effects to play using system
- Time module to delay execution of code

Stages of Development:

1. Creating Window (Lead:Fudail)
2. Creating Mamba Object (Lead: Fudail)
3.1 Adding Keyboard Functionality (Lead: Fudail)
4. Creating Regular Blocks (Lead: Denzel)
5. Creating Mamba Impacts (Lead: Denzel)
6. Adding Mamba Extensions (Lead: Denzel)
7. Lucky Blocks & Timer (Lead: Denzel)
8. Border Impacts (Lead: Fudail)
9. Game Lost (Lead: Fudail)
10. Sound Effects(Lead: Fudail)
11. Game Metrics (Lead: Denzel)
12. Game Won(Lead: Denzel)
13. Author Names (Lead: Fudail)

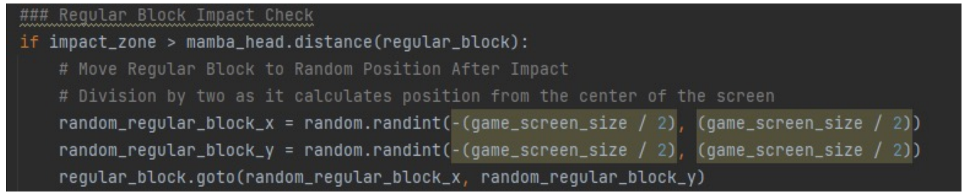
Concepts & Techniques

- Iteration



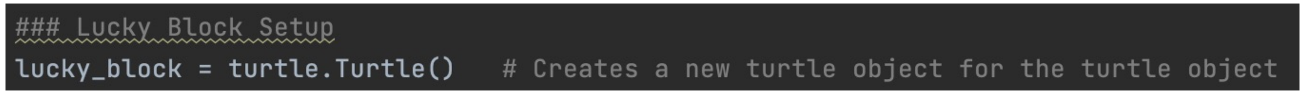
```
while gameplay == True:
    game_screen.listen()    # Listens for key changes
    game_screen.update()    # Updates Screen With Changes
```

- If Statements



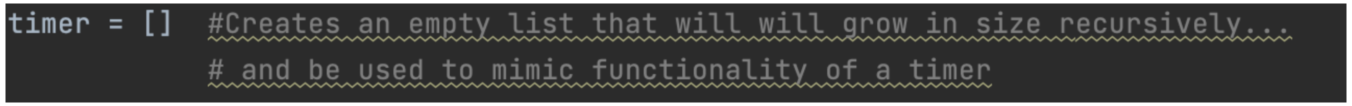
```
### Regular Block Impact Check
if impact_zone > mamba_head.distance(regular_block):
    # Move Regular Block to Random Position After Impact
    # Division by two as it calculates position from the center of the screen
    random_regular_block_x = random.randint(-(game_screen_size / 2), (game_screen_size / 2))
    random_regular_block_y = random.randint(-(game_screen_size / 2), (game_screen_size / 2))
    regular_block.goto(random_regular_block_x, random_regular_block_y)
```

- Variable Declaration



```
### Lucky Block Setup
lucky_block = turtle.Turtle()   # Creates a new turtle object for the turtle object
```

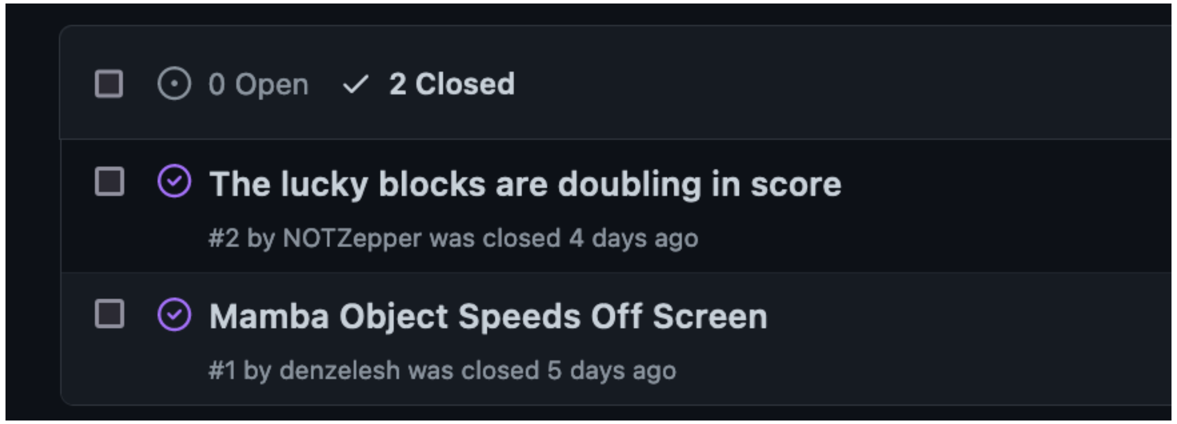
- Lists



```
timer = []  #Creates an empty list that will will grow in size recursively...
            # and be used to mimic functionality of a timer
```

Programming Issues

- Lucky block speeding off uncontrollably (lack of waiting)
- Lucky blocks doubling In size (fast code execution)
- Background Music (lack of async – alternative)



0 Open ✓ 2 Closed

The lucky blocks are doubling in score
#2 by NOTZepper was closed 4 days ago

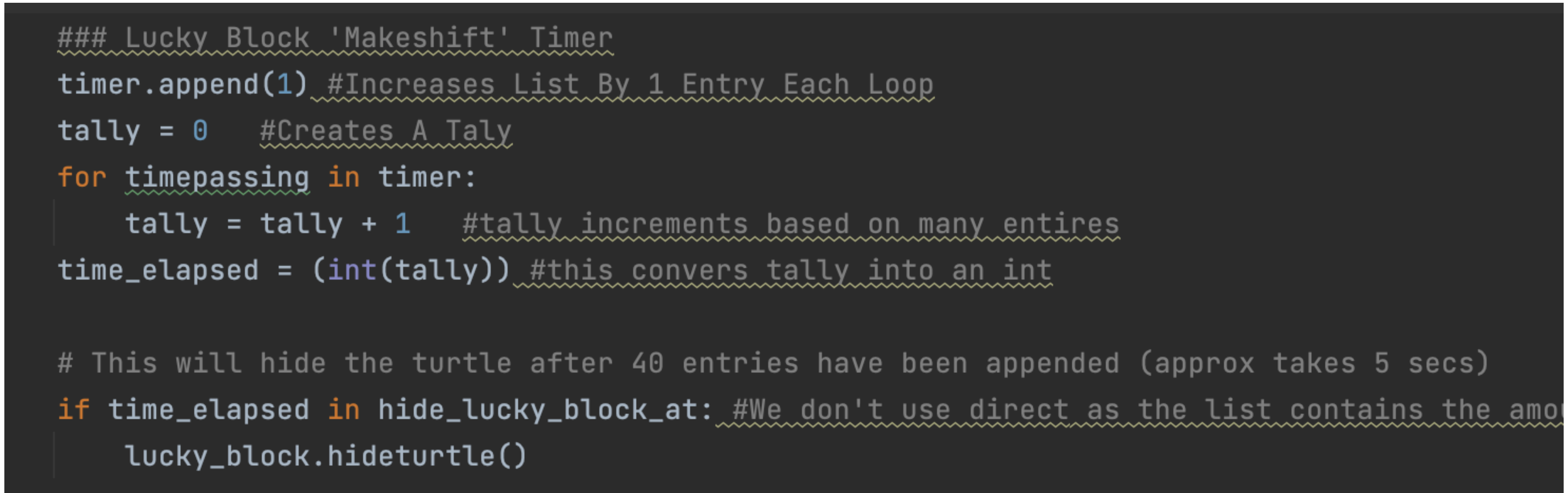
Mamba Object Speeds Off Screen
#1 by denzelesh was closed 5 days ago

Logical Issue

- How to make the lucky block appear & disappear after a set time?
- Could not create a timer loop within the main game loop as this would require running nested while loops which will delay the execution of code
- Could not get rid of ‘While True’ loop or else the game would not be able to run indefinitely without a condition such as ‘game won’ or ‘game lost’
- Could not use threading or parallel processing as this would create two separate windows, independent of each other.

Created a ‘Makeshift Timer’

- ’Makeshift’ because it’s a list (not a timer 😢) that grows in size per every loop
- Based on the size of the list, it would hide/show luckyblock
- Not perfect but works with limitations



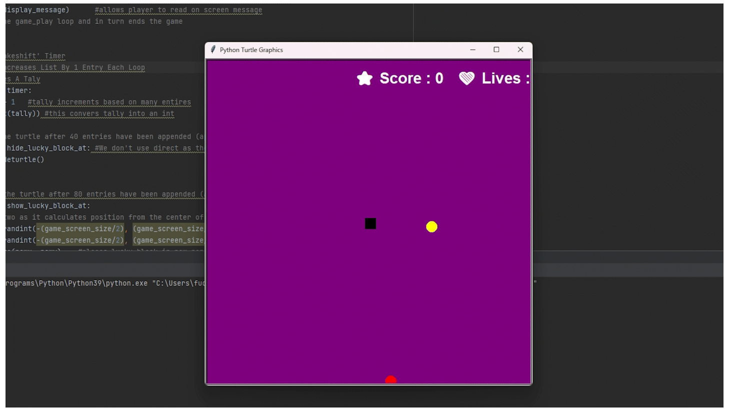
```
### Lucky Block 'Makeshift' Timer
timer.append(1) #Increases List By 1 Entry Each Loop
tally = 0    #Creates A Taly
for timepassing in timer:
    tally = tally + 1    #tally increments based on many entires
time_elapsed = (int(tally)) #this convers tally into an int

# This will hide the turtle after 40 entries have been appended (approx takes 5 secs)
if time_elapsed in hide_lucky_block_at: #We don't use direct as the list contains the amo
    lucky_block.hideturtle()
```

Testing

Test description	Expected outcome	Actual Outcome
Main window	A window opens with the correct dimensions	A window opens with a purple background
Keyboard Functionality	Snake moves in the direction inputted	Snake moves in the direction inputted
Regular Blocks	Regular boxes spawn in random locations	Regular boxes spawn in random locations
Mamba Extensions	Mamba extends when a regular block is consumed	Mamba grows when a regular block is consumed
Lucky Blocks & Timer	Lucky blocks spawn and despawn after the time limit to collect them runs out	Lucky block spawns and despawns with the time limit to consume it
Border Impacts & Game Lost	Snake collides with the wall, the game is lost if the snake has no lives left	The feature works as expected
Sound Effects	All sounds work	Sounds work as expected

Testing Footage

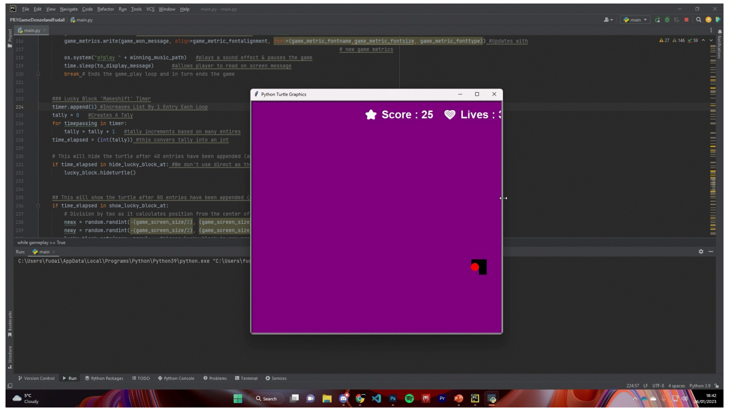
Game Lost & Sound Effect



```
isplay_message)      #allows player to read on screen message
e game_play loop and in turn ends the game


keshift' Timer
creases List By 1 Entry Each Loop
s A Taly
timer:
 1   #tally increments based on many entires
(tally)) #this convers tally into an int

e turtle after 40 entries have been appended (a
hide_lucky_block_at: #We don't use direct as th
eturtle()


he turtle after 80 entries have been appended (
show_lucky_block_at:
wo as it calculates position from the center of
andint(-(game_screen_size/2), (game_screen_size
andint(-(game_screen_size/2), (game_screen_size
```

```
rograms\Python\Python39\python.exe "C:\Users\fu
```

Life added and lost

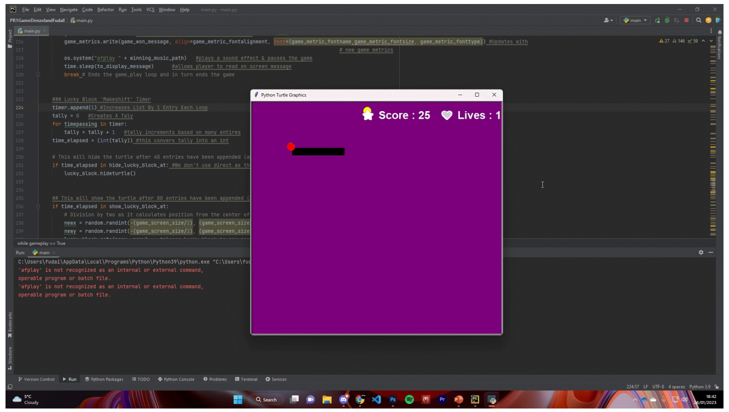


```
            game_metrics.write(game_won_message, align=game_metric_fontalignment, font=(game_metric_fontname,game_metric_fontsize, game_metric_fonttype)) #Updates with
                                                                                                  # new game metrics

            os.system("afplay " + winning_music_path)   #plays a sound effect & pauses the game
            time.sleep(to_display_message)      #allows player to read on screen message
            break # Ends the game_play loop and in turn ends the game


        ### Lucky Block 'Makeshift' Timer
        timer.append(1) #Increases List By 1 Entry Each Loop
        tally = 0   #Creates A Taly
        for timepassing in timer:
            tally = tally + 1   #tally increments based on many entires
        time_elapsed = (int(tally)) #this convers tally into an int

        # This will hide the turtle after 40 entries have been appended (a
        if time_elapsed in hide_lucky_block_at: #We don't use direct as th
            lucky_block.hideturtle()


        ## This will show the turtle after 80 entries have been appended (
        if time_elapsed in show_lucky_block_at:
            # Division by two as it calculates position from the center of
            newx = random.randint(-(game_screen_size/2), (game_screen_size
            newy = random.randint(-(game_screen_size/2), (game_screen_size
```

```
C:\Users\fudai\AppData\Local\Programs\Python\Python39\python.exe "C:\Users\fu
```

Game Won & Sound Effect

Keyboard Input & Score Increments



```
            game_metrics.write(game_won_message, align=game_metric_fontalignment, font=(game_metric_fontname,game_metric_fontsize, game_metric_fonttype)) #Updates with
                                                                                                  # new game metrics

            os.system("afplay " + winning_music_path)   #plays a sound effect & pauses the game
            time.sleep(to_display_message)      #allows player to read on screen message
            break # Ends the game_play loop and in turn ends the game


        ### Lucky Block 'Makeshift' Timer
        timer.append(1) #Increases List By 1 Entry Each Loop
        tally = 0   #Creates A Taly
        for timepassing in timer:
            tally = tally + 1   #tally increments based on many entires
        time_elapsed = (int(tally)) #this convers tally into an int

        # This will hide the turtle after 40 entries have been appended (a
        if time_elapsed in hide_lucky_block_at: #We don't use direct as th
            lucky_block.hideturtle()


        ## This will show the turtle after 80 entries have been appended (
        if time_elapsed in show_lucky_block_at:
            # Division by two as it calculates position from the center of
            newx = random.randint(-(game_screen_size/2), (game_screen_size
            newy = random.randint(-(game_screen_size/2), (game_screen_size
```

```
C:\Users\fudai\AppData\Local\Programs\Python\Python39\python.exe "C:\Users\fu
'afplay' is not recognized as an internal or external command,
operable program or batch file.
'afplay' is not recognized as an internal or external command,
operable program or batch file.
```

Overall Summary

In this project, we successfully programmed a snake game using the Python programming language and the python turtle graphics library.

The game features a snake that moves on the screen, collects blocks, and grows in size. We also implemented collision detection, a scoring system, and a simple user interface.

Throughout the development process, we utilized various programming concepts such as object-oriented programming and event-driven programming to create a functional and engaging game.

The game also passed through a thorough testing phase to ensure that it runs smoothly and is free of bugs. Overall, this project was a great opportunity for us to apply our programming skills and create a fun and interactive game.