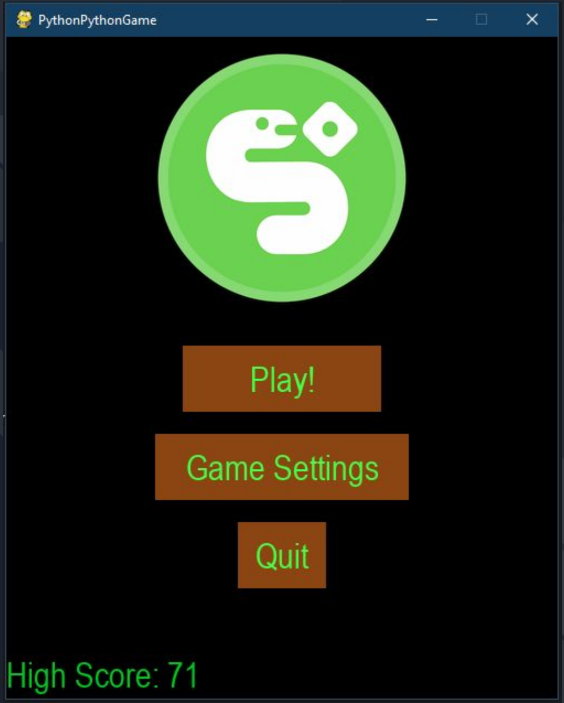
CIS 434 Software Engineering Python Python Game Group 12

Aidan Zapotechne
Derek Woods
Toral Zaveri

Abstract

- Recreated the fun and simplicity of the classic snake game
- Used Pygame framework in python
- Added 2 player game modes and additional game setting
- Focused on github, source control, teamwork and time management

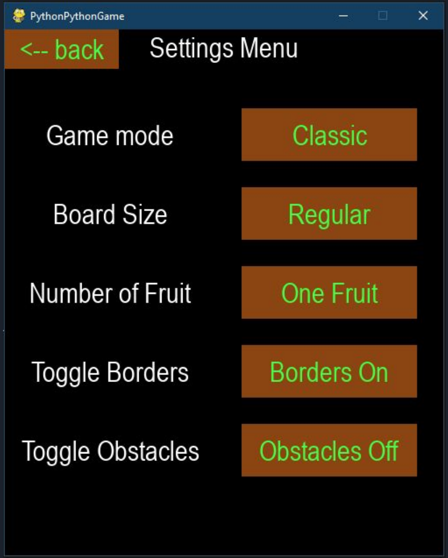


High Score: 71

Contributions

Aidan Zapotechne

- Software
  - Settings menu (buttons and changing game settings based on button clicks)
  - 2-P snake collision handling
  - Fruit spawns based on amount specified
  - 2-P race timer
  - Obstacle implementation
- Final Report
  - Abstract
  - Project Description
  - Professional Awareness

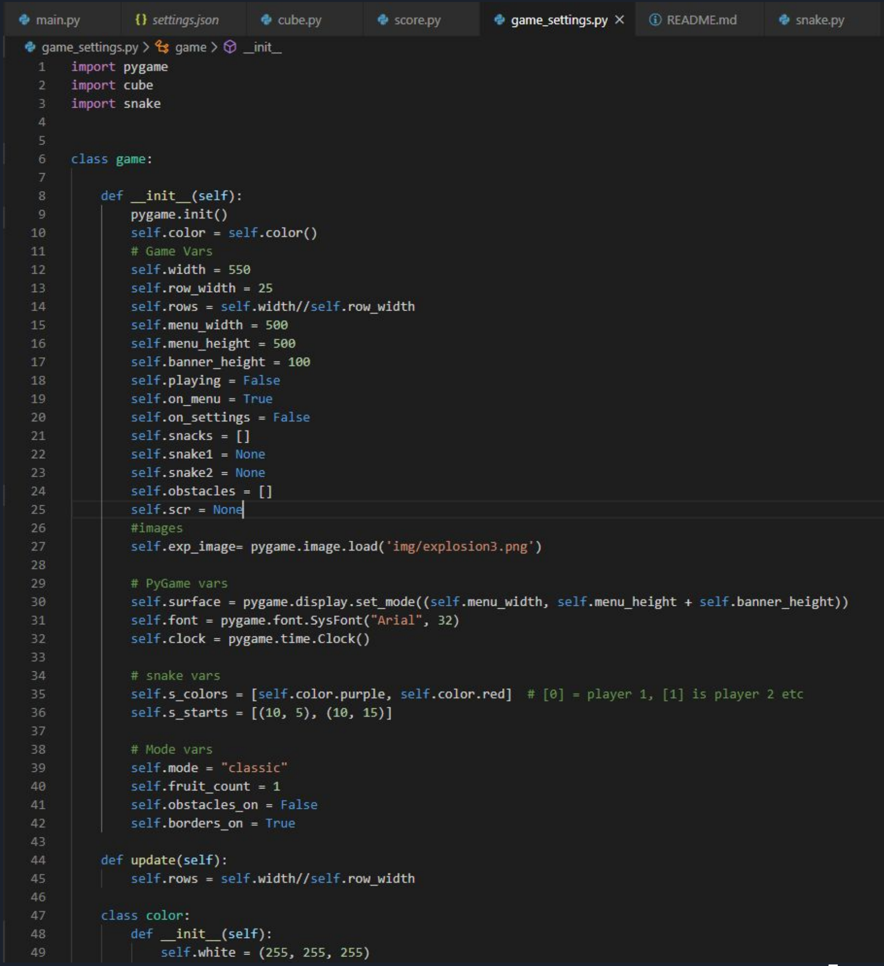


Settings Menu

Game mode	Classic
Board Size	Regular
Number of Fruit	One Fruit
Toggle Borders	Borders On
Toggle Obstacles	Obstacles Off

Derek Woods

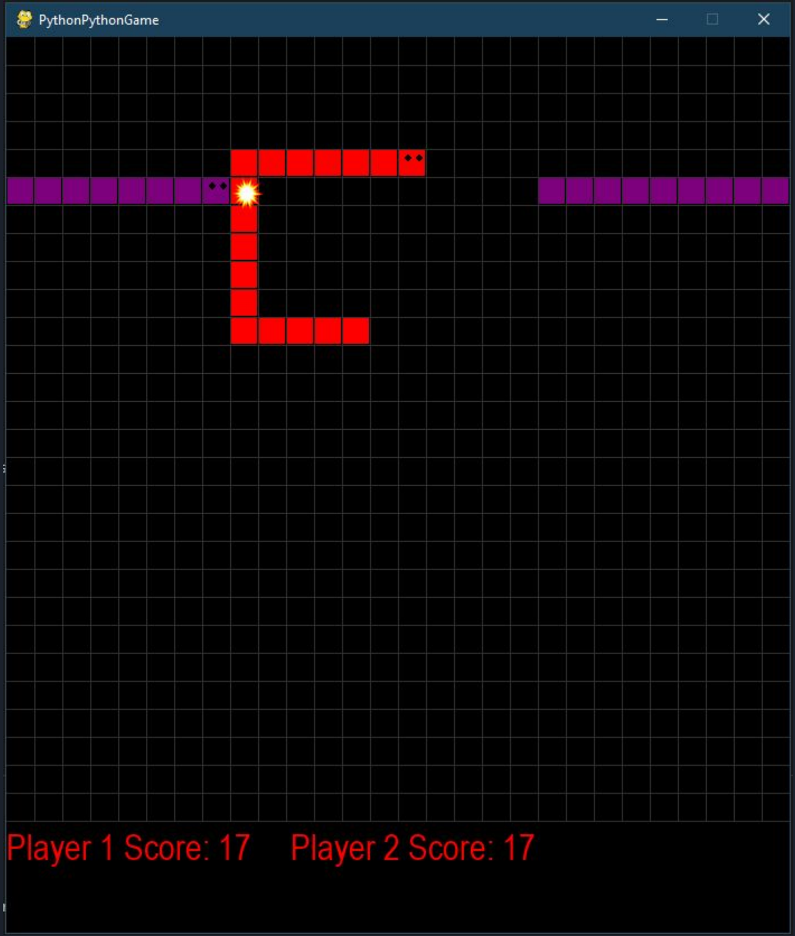
- Software:
  - Base Game (V1.0.0)
    - Simple framework of a single snake with movement, growth and collision detection to kickoff the project
  - Game Settings class
  - Various bug fixes and UI improvements
- Final Report:
  - Game manual in ‘README.md’
  - Partial Objectives
  - Project Timelines



```python
import pygame
import cube
import snake


class game:

    def __init__(self):
        pygame.init()
        self.color = self.color()
        # Game Vars
        self.width = 550
        self.row_width = 25
        self.rows = self.width//self.row_width
        self.menu_width = 500
        self.menu_height = 500
        self.banner_height = 100
        self.playing = False
        self.on_menu = True
        self.on_settings = False
        self.snacks = []
        self.snake1 = None
        self.snake2 = None
        self.obstacles = []
        self.scr = None
        #images
        self.exp_image= pygame.image.load('img/explosion3.png')

        # PyGame vars
        self.surface = pygame.display.set_mode((self.menu_width, self.menu_height + self.banner_height))
        self.font = pygame.font.SysFont("Arial", 32)
        self.clock = pygame.time.Clock()

        # snake vars
        self.s_colors = [self.color.purple, self.color.red]  # [0] = player 1, [1] is player 2 etc
        self.s_starts = [(10, 5), (10, 15)]

        # Mode vars
        self.mode = "classic"
        self.fruit_count = 1
        self.obstacles_on = False
        self.borders_on = True

    def update(self):
        self.rows = self.width//self.row_width

    class color:
        def __init__(self):
            self.white = (255, 255, 255)
```

Toral Zaveri

- Software:
  - Adding live score for classic game
  - Adding live score for 2-player
- Final Report:
  - Conclusion

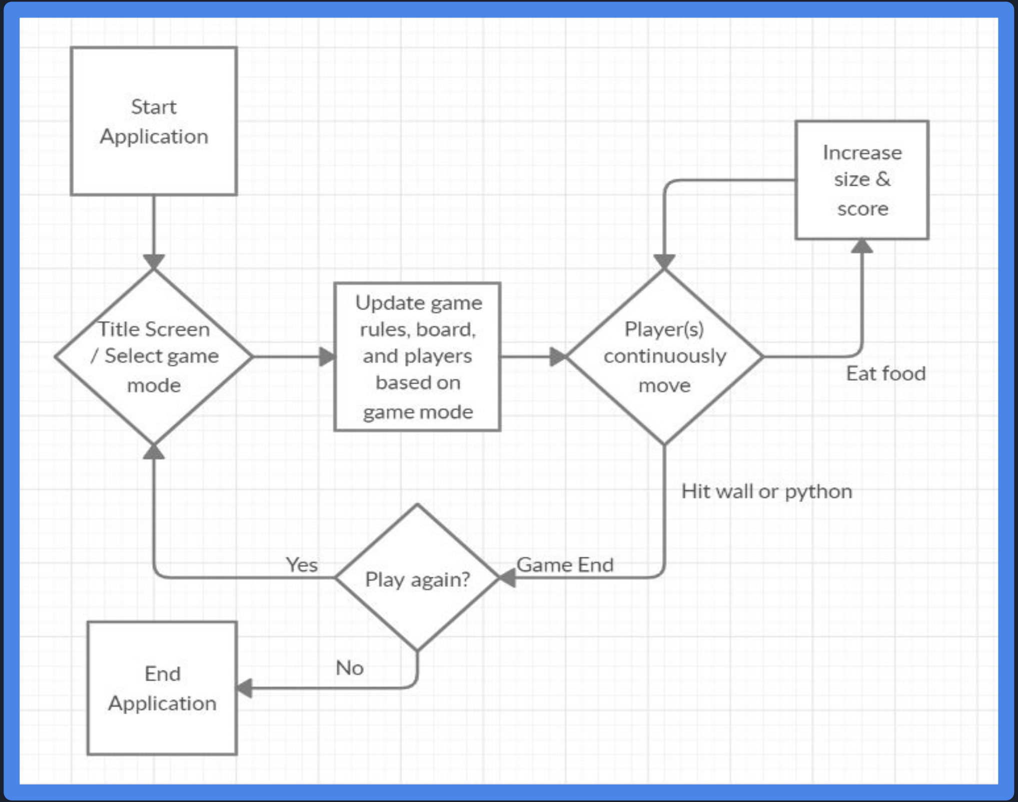


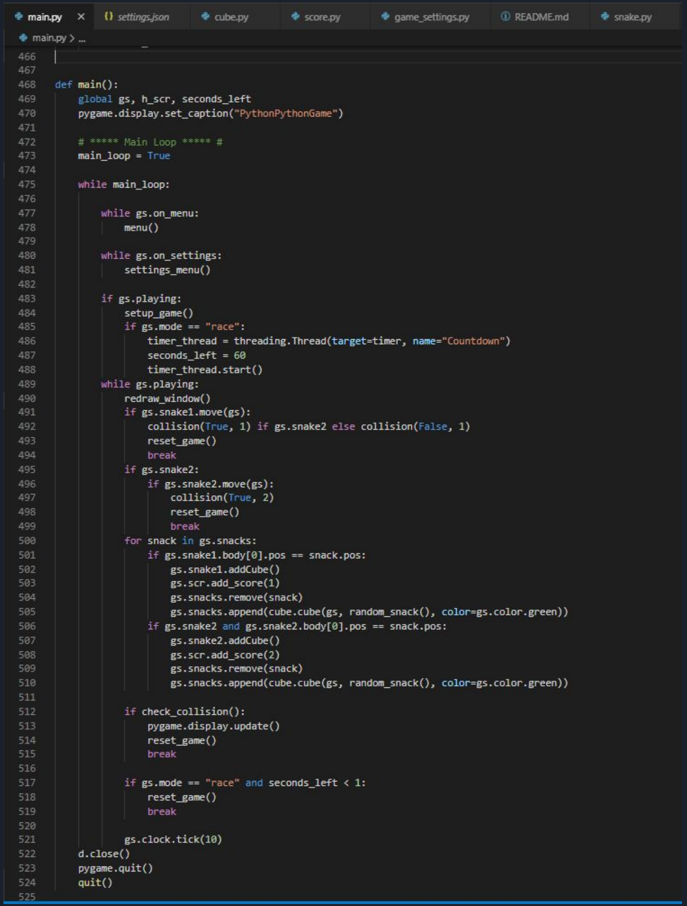
Project Objectives

- Utilize the Pygame framework in Python to create an iteration of the snake game.
- Have all the basic snake game aspects.
- Make sure the game has an enjoyable user experience
- Allow the difficulty to change during the game.
- Follow engineering methodologies discussed in the course
- Work effectively in a software engineering team

Project Description

Software Design


Start Application
Title Screen / Select game mode
Update game rules, board, and players based on game mode
Player(s) continuously move
Increase size & score
Eat food
Hit wall or python
Game End
Play again?
Yes
No
End Application



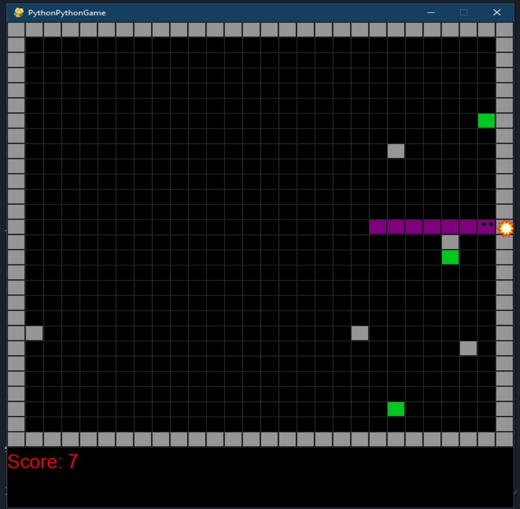
```python

def main():
    global gs, h_scr, seconds_left
    pygame.display.set_caption("PythonPythonGame")

    # ***** Main Loop ***** #
    main_loop = True

    while main_loop:

        while gs.on_menu:
            menu()

        while gs.on_settings:
            settings_menu()

        if gs.playing:
            setup_game()
            if gs.mode == "race":
                timer_thread = threading.Thread(target=timer, name="Countdown")
                seconds_left = 60
                timer_thread.start()
        while gs.playing:
            redraw_window()
            if gs.snake1.move(gs):
                collision(True, 1) if gs.snake2 else collision(False, 1)
                reset_game()
                break
            if gs.snake2:
                if gs.snake2.move(gs):
                    collision(True, 2)
                    reset_game()
                    break
            for snack in gs.snacks:
                if gs.snake1.body[0].pos == snack.pos:
                    gs.snake1.addCube()
                    gs.scr.add_score(1)
                    gs.snacks.remove(snack)
                    gs.snacks.append(cube.cube(gs, random_snack(), color=gs.color.green))
                if gs.snake2 and gs.snake2.body[0].pos == snack.pos:
                    gs.snake2.addCube()
                    gs.scr.add_score(2)
                    gs.snacks.remove(snack)
                    gs.snacks.append(cube.cube(gs, random_snack(), color=gs.color.green))

            if check_collision():
                pygame.display.update()
                reset_game()
                break

            if gs.mode == "race" and seconds_left < 1:
                reset_game()
                break

            gs.clock.tick(10)
    d.close()
    pygame.quit()
    quit()

```

Software Features

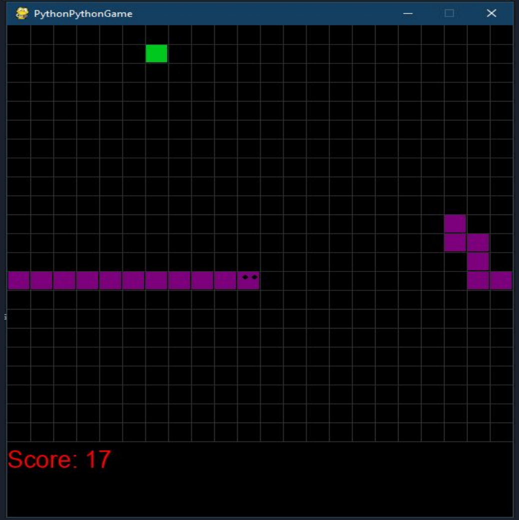
- Several ways for the user to customize their experience via game settings
- Main menu where player can start the game with default settings, or open the settings menu
- In the settings menu the user can change five different attributes:
    - Gamemode - can be classic, two-player race, or two-player melee
    - Board size - player selects a range of five different boards that vary the amount of rows and columns traversable in game
    - Number of Fruit - player specifies how many fruits are spawned on screen at one time
    - Borders - player specifies if borders should either be obstacles, or pass through borders
    - Obstacles - user specifies if obstacles spawn, in which five grey tiles will spawn randomly on screen that the player must avoid

Single player

- Classic
  - This is the only mode that supports high score
  - All settings can still be modified
  - Play until player crashes into obstacle



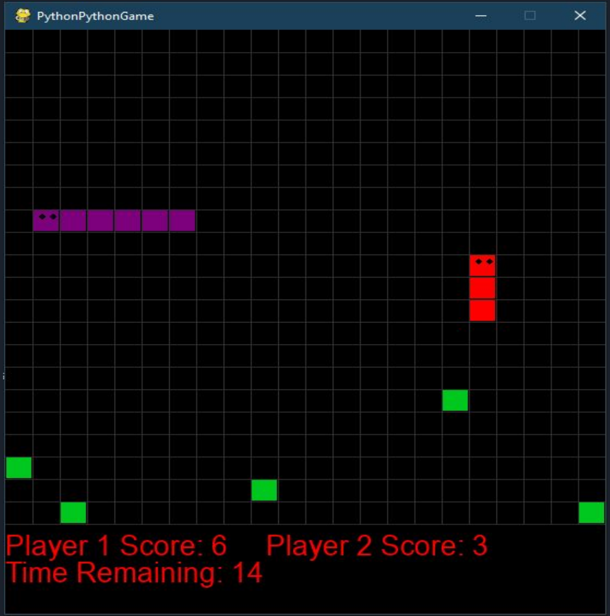
Score: 7



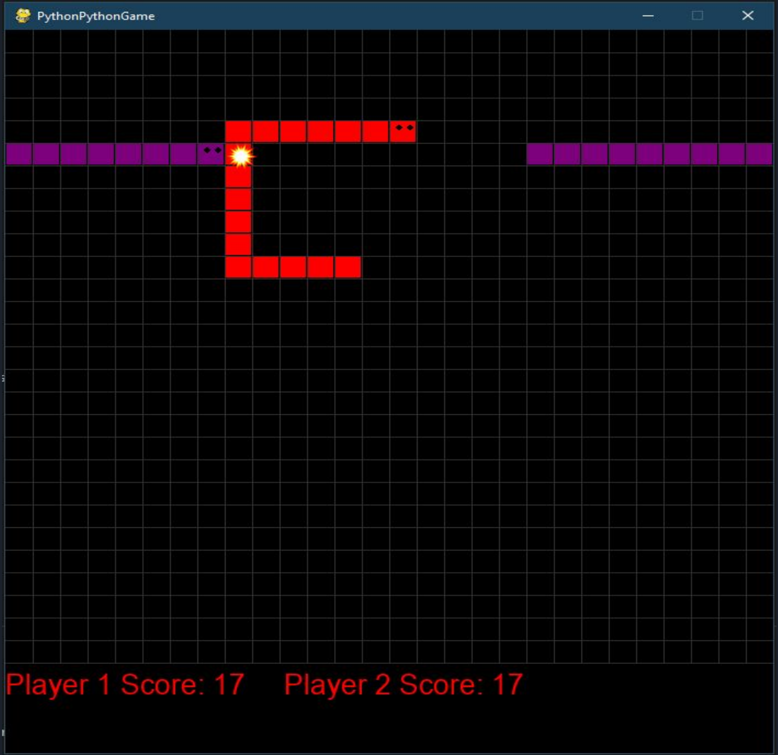
Score: 17

Two player

- Race mode
  - 60 second timer



Player 1 Score: 6   Player 2 Score: 3
Time Remaining: 14



Player 1 Score: 17   Player 2 Score: 17

- Melee Mode
  - No fruits and growth every 10 moves

Project Difficulties and Solutions

- How to implement snake movement, specifically the snake’s segments (long body) ?
- How to implement different game modes?
- How to maintain the high score?

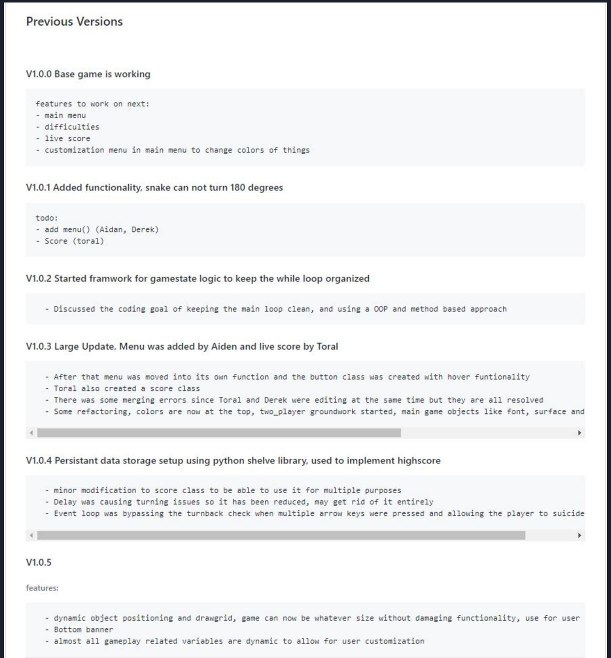
- Utilize python data structures in clever way: One list contains body segment positions and another contains the turns the snake takes
- Two lists hold settings buttons, one is active settings and one is inactive. Based on which are pressed, the respective settings are updated
- The high score is saved to the user’s local machine through the python shelve library

Project Timeline

Project Tasks

- PyGame setup and draw game board were the first tasks completed.
- Python sprite, user input, movement and food spawning were the next tasks tackled. The team was able to handle all collisions, growth, movement and rendering so the python was never transformed into a PyGame sprite.
- After the base game was operational is when the menu was focused on. The menu was postponed until later since it was tied into the game settings, modes and UI improvements.
- The final task was additional game modes and is where our team spent most of our time, additional game modes and settings required code refactoring and the gs object previously mentioned. The end result was the ability to add game features with ease.

Version Control



Previous Versions

V1.0.0 Base game is working

```
features to work on next:
- main menu
- difficulties
- live score
- customization menu in main menu to change colors of things
```

V1.0.1 Added functionality, snake can not turn 180 degrees

```
todo:
- add menu() (Aidan, Derek)
- Score (toral)
```

V1.0.2 Started framwork for gamestate logic to keep the while loop organized

```
  - Discussed the coding goal of keeping the main loop clean, and using a OOP and method based approach
```

V1.0.3 Large Update, Menu was added by Aiden and live score by Toral

```
  - After that menu was moved into its own function and the button class was created with hover funtionality
  - Toral also created a score class
  - There was some merging errors since Toral and Derek were editing at the same time but they are all resolved
  - Some refactoring, colors are now at the top, two_player groundwork started, main game objects like font, surface and
```

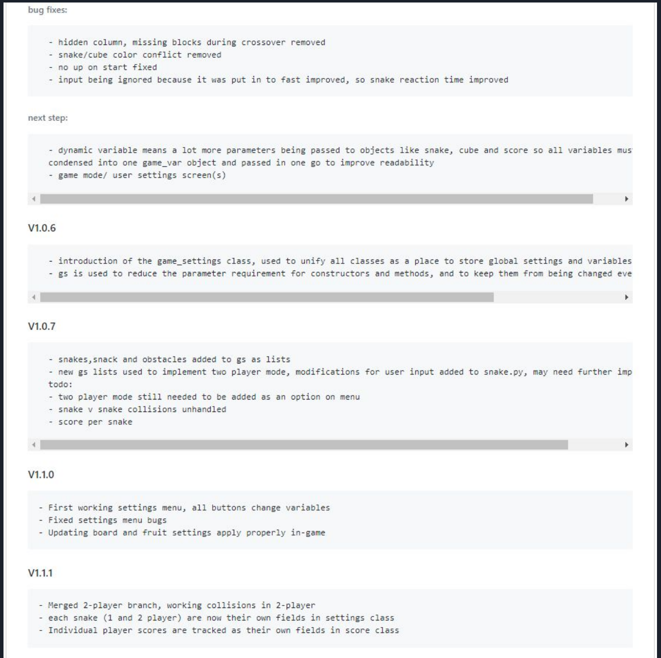
V1.0.4 Persistant data storage setup using python shelve library, used to implement highscore

```
  - minor modification to score class to be able to use it for multiple purposes
  - Delay was causing turning issues so it has been reduced, may get rid of it entirely
  - Event loop was bypassing the turnback check when multiple arrow keys were pressed and allowing the player to suicide
```

V1.0.5

features:

```
  - dynamic object positioning and drawgrid, game can now be whatever size without damaging functionality, use for user
  - Bottom banner
  - almost all gameplay related variables are dynamic to allow for user customization
```



bug fixes:

```
  - hidden column, missing blocks during crossover removed
  - snake/cube color conflict removed
  - no up on start fixed
  - input being ignored because it was put in to fast improved, so snake reaction time improved
```

next step:

```
  - dynamic variable means a lot more parameters being passed to objects like snake, cube and score so all variables mus
  condensed into one game_var object and passed in one go to improve readability
  - game mode/ user settings screen(s)
```

V1.0.6

```
  - introduction of the game_settings class, used to unify all classes as a place to store global settings and variables
  - gs is used to reduce the parameter requirement for constructors and methods, and to keep them from being changed eve
```

V1.0.7

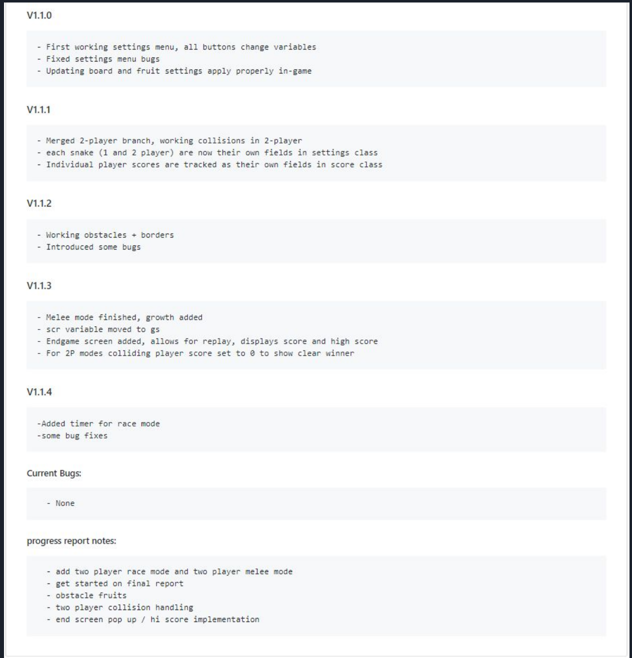
```
  - snakes,snack and obstacles added to gs as lists
  - new gs lists used to implement two player mode, modifications for user input added to snake.py, may need further imp
  todo:
  - two player mode still needed to be added as an option on menu
  - snake v snake collisions unhandled
  - score per snake
```

V1.1.0

```
- First working settings menu, all buttons change variables
- Fixed settings menu bugs
- Updating board and fruit settings apply properly in-game
```

V1.1.1

```
- Merged 2-player branch, working collisions in 2-player
- each snake (1 and 2 player) are now their own fields in settings class
- Individual player scores are tracked as their own fields in score class
```



V1.1.0

```
- First working settings menu, all buttons change variables
- Fixed settings menu bugs
- Updating board and fruit settings apply properly in-game
```

V1.1.1

```
- Merged 2-player branch, working collisions in 2-player
- each snake (1 and 2 player) are now their own fields in settings class
- Individual player scores are tracked as their own fields in score class
```

V1.1.2

```
- Working obstacles + borders
- Introduced some bugs
```

V1.1.3

```
- Melee mode finished, growth added
- scr variable moved to gs
- Endgame screen added, allows for replay, displays score and high score
- For 2P modes colliding player score set to 0 to show clear winner
```

V1.1.4

```
-Added timer for race mode
-some bug fixes
```

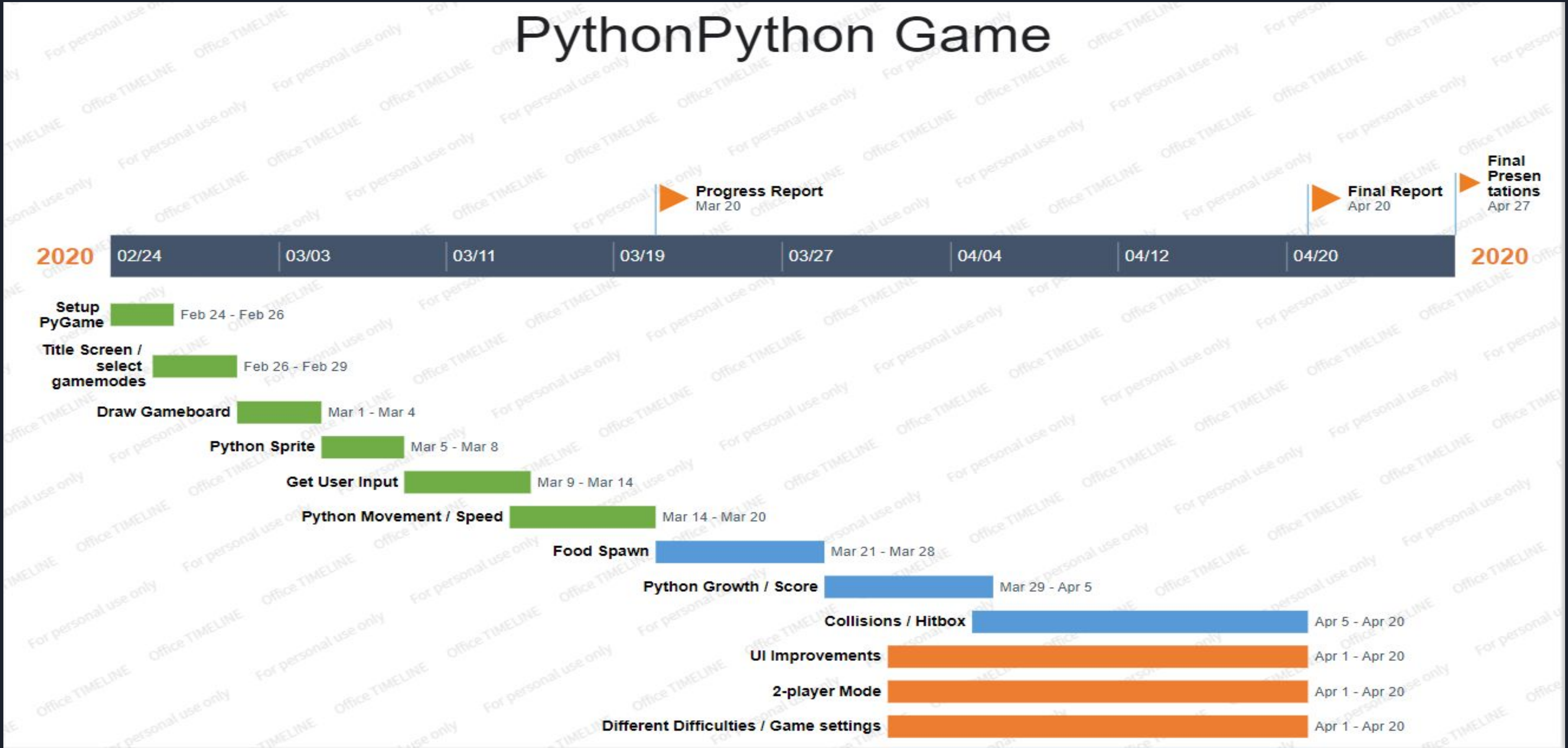
Current Bugs:

```
  - None
```

progress report notes:

```
  - add two player race mode and two player melee mode
  - get started on final report
  - obstacle fruits
  - two player collision handling
  - end screen pop up / hi score implementation
```

Gantt Chart:



Office TIMELINE For personal use only

Conclusion

- For this project, we implemented a fully functional classic Snake game.
- The newly unique version of the snake game has been created in Python.
- By making this project, we learned a new framework, PyGame and learned how to work with the proper software engineering methodologies.
- We faced many ups and downs while doing the project, but eventually we tested, fixed bugs, and updated our version of snake game.

References

“Pygame Front Page.” *Pygame Front Page - Pygame v2.0.0.dev5 Documentation*, www.pygame.org/docs/.

Sommerville, I. *Software Engineering*. 10th ed., Pearson/Addison-Wesley, 2004.

“Python 3.8.2rc2 Documentation.” *3.8.2rc2 Documentation*, docs.python.org/3/.