1. **PROJECT PROPOSAL(FILE).**

* PROJECT DESCRIPTION

This is a project that looks to focus on students in ALU who are having a lot of work that they can’t handle and this might lead them to have a mental breakdown. So this project is a 2d game that tends to help the ALU students escape from the pressure of the work to maintain their mental health. Since games are important they help people escape the problems of the real world and just have fun when playing the game. That is why games were created.

* PROBLEM STATEMENT

Jacob is a new first-year student at the African Leadership University who has no idea of how the ALU learning curriculum is. After a few weeks, Jacob has been panicking due to a lot of assignments given, so he had to attend therapy for some weeks to keep him mentally safe. As a Computer science student, how can I help the first-year students to keep them mentally ok so that they have any mental issues?

* PROCEDURES AND METHODS

As soon as the snake game app opens, a nice background will be displayed and a snake will also appear on the screen which the user will be able to control. The main goal of the game will be for the snake to eat the food and get the snake as long as possible without hitting the body of the snake Every time a user eats the food a score will be added to the user’s current scores. It’s going to be an endless game since the majority of people enjoy endless games

* ANTICIPATED OUTCOMES

To create a Snake game that allows users to control the movement of a snake on a screen, to get points for eating food and avoiding running into the snake itself.

1. **MAIN: LIST OF CLASSES(FILE).**

‘’’ Our first class is the snake class. This class contains objects of our snake, its properties, and functionalities. For instance; it handles the movement of our snake, the intersection of our game whenever the snake touches its tail, and also draws our rectangle(snake). The object on the surface of the game screen. And whenever this class is called upon, it will not only display our snake but also call our different objects with its functionalities.’’’

class Snake:

def \_\_init\_\_(self):

self.length = 1

self.positions = [((screen\_width / 2), (screen\_height / 2))]

self.direction = random.choice([up, down, left, right])

self.color = (17, 24, 47)

self.score = 0

‘’’ Our second class is a food class. This class contains objects that are the functionalities of our food. For instance, we need to place our food in different positions whenever our game resets. This class also draw our rectangle(food) object on the surface of our game screen. This class also relates to the other classes as it does only display our food on the screen but also position it randomly’’’

class Food(Snake):

def \_\_init\_\_(self, position, color, randomize):

super().\_\_init\_\_()

self.position = (0, 0)

self.color = (223, 163, 49)

self.randomize\_position()

1. **Pseudocode**

Pseudocode:

Create snake Class

Define init to declare self,snake\_length,snake\_position

Declare self.length

Declare self.position

Declare self.direction

Declare self.color

Declare self. Score

1. **SNAKE FOOD CLASS(FILE).**

Pseudocode:

Create Food Class

Define init to declare self

Define super init to declare color, position, randomize\_position

Declare self.position

Declare self.color

Declare self.randomize\_position

Tests

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test number | Description | Test data | Expected result | Actual result | Pass/Fail |
| 1 | Snake game reset when it touches its tail | Def reset:  Game reset based on action trigger by the user | Game reset when the snake intersect with its tail | The game reset when the snake touches its tail | Pass |
| 2 | When a user presses a key on their keyboard, the program should respond to the key pressed | Handle keys: quit, keydown, keyup, key left, key right, | The game should respond to user inputs when triggered | Quit game, snake move based on keys pressed | Pass |
| 3 | Def draw: draw a rectangle on the screen surface. | Draw snake: draws a rectangle on the game screen | It should draw the only rectangle and not circle or others | Draw rectangle when the snake eats a piece of food | Pass |
| 4 | Def turn, turns snake in a different direction | Def turn: Checks for snake position and length size before the snake move | Should not result in an error when moving the snake in a different position | Move snake head based on the snake’s direction and position | pass |
| 5 | Add a rectangle to snake tail whenever snake eats a new food | Def move: self.position.insert | The program should add a new rectangle based on the exact position of the snake. | Adds a new rectangle of the snake’s current position. | Pass |
| 6 | Spread Out the exact size of rectangles on the game surface | Screen: We want a specific length and height of our game screen for the convenience of players | whenever pygame.display.set\_mode is called in a while loop, the set screen length and height should be return | Return specific length and height of the game screen when called in a while loop. | Pass |
| 7 | Display writing on the game screen | My\_font: we want our game to display “score” and score counting on the screen | this function should return a monospace 16 size font on the screen when the game is running | Whenever this internal function is called in a while loop, score and scouting are display on the screen | Pass |
| 8 | Delay the screen when the player loses the game. | Clock: We want our game to keep running for a set time when the game ended | The program should not abruptly quit when user lose, instead it should stay for set time to show user’s score | Whenever this internal function is called in a while loop, it should keep running until the set time is reached. | Pass |
| 9 | Update and return head position when the game reset | Def get head position: Whenever this function is called, the snake current head position is updated and called upon | The program should be able to get the snakehead current position which in return help us to restart the game | When the game ended, we were able to update and call the last status or length of the snake when the user ended or didn’t win. Which in this case is self.positions[0] | Pass |
| 10 | Exit Game when a player press the ‘X’ button | Quit Game: Exit the game whenever user click the exit(X) button | The program should not result in an error whenever user click the ‘X’ button or exit the game | When user press the ‘X’ button, the game ended without crashing | Pass |