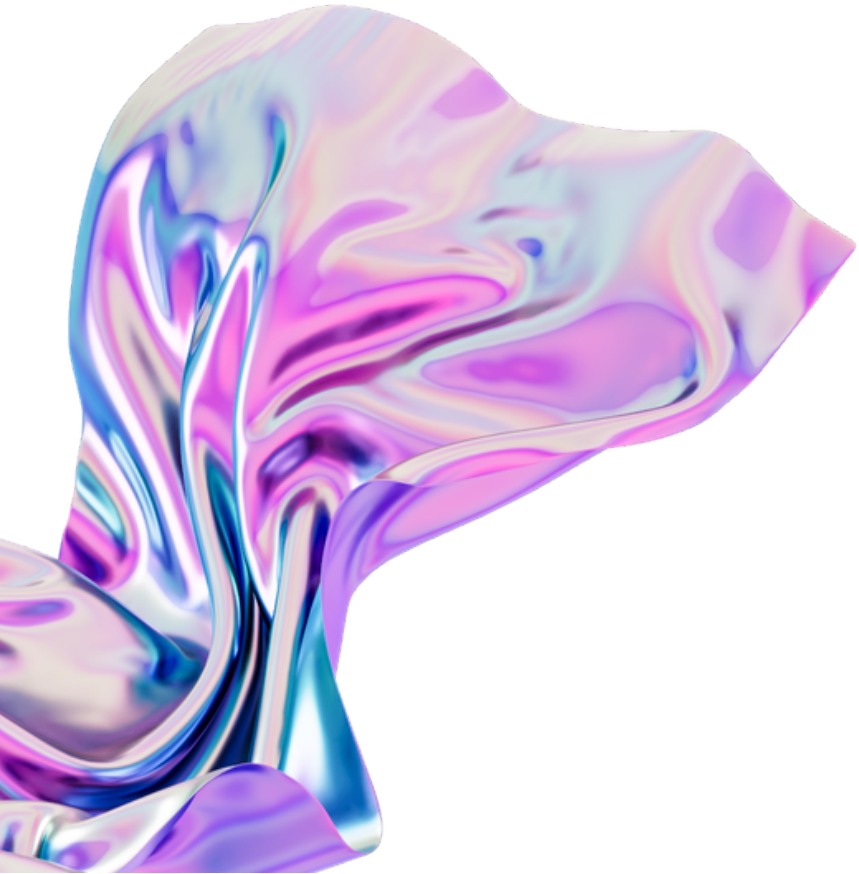
Snake game

Project Proposal

Artificial Intelligence



Riham Mohammed	2110278
Enas Abdulaziz	2110329
Roaa Adel	2111015

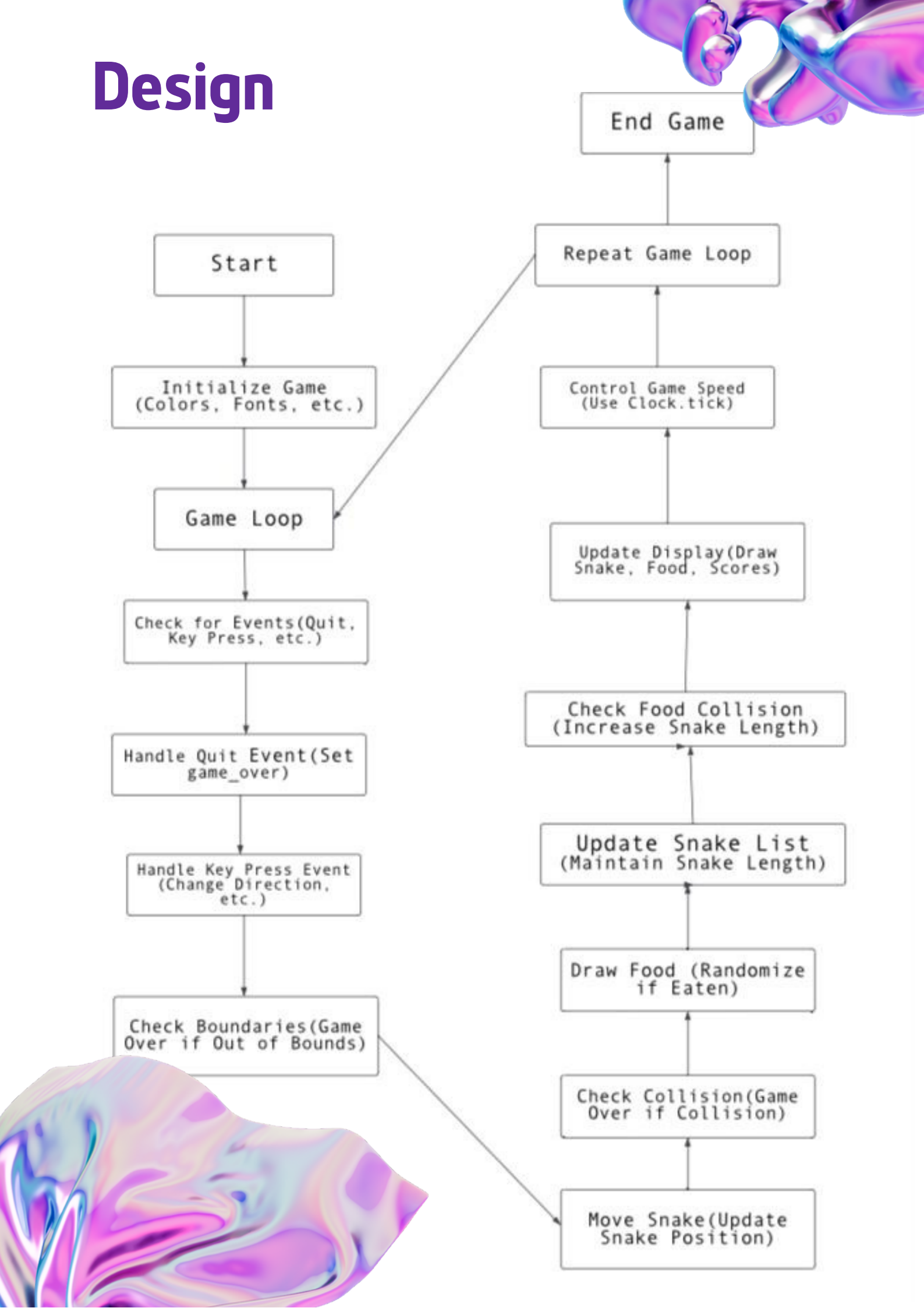
Introduction

A beloved video game, Snake lets players take control of a snake as it moves through a grid, eats food, and gets longer. The goal of implementing artificial intelligence (AI) in Snake games is to develop intelligent agents capable of making strategic decisions while playing the game on their own. This has proven to be an interesting challenge. The concept, implementation, and methodology of an AI-based Snake game are examined in this article.

Objective

Creating an adaptive agent with the ability to make wise judgments in order to strategically consume food, prevent self-collision, and maneuver the snake is the main goal of integrating AI into the Snake game. In order to adjust to shifting game dynamics, the AI should have learning skills.

Design


Start
Initialize Game (Colors, Fonts, etc.)
Game Loop
Check for Events(Quit, Key Press, etc.)
Handle Quit Event(Set game_over)
Handle Key Press Event (Change Direction, etc.)
Check Boundaries(Game Over if Out of Bounds)
Move Snake(Update Snake Position)
Check Collision(Game Over if Collision)
Draw Food (Randomize if Eaten)
Update Snake List (Maintain Snake Length)
Check Food Collision (Increase Snake Length)
Update Display(Draw Snake, Food, Scores)
Control Game Speed (Use Clock.tick)
Repeat Game Loop
End Game

How AI Approch Works in Our Program ?

1-Start Game:

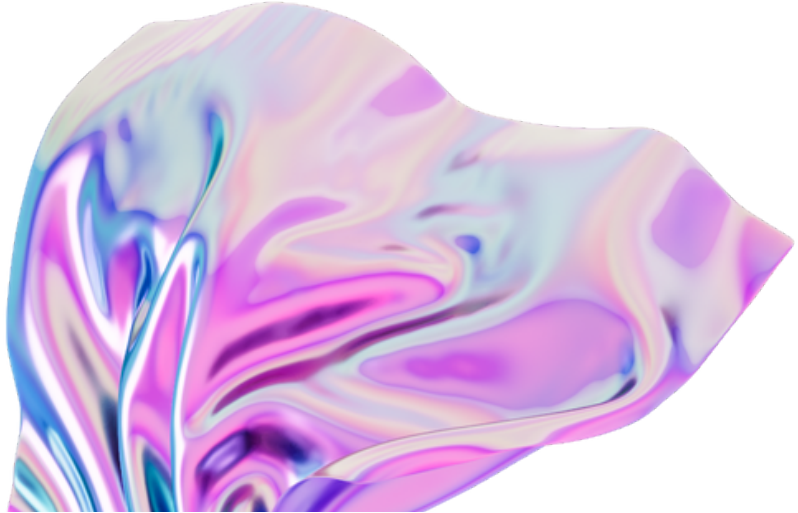
The Snake game begins at this point in the program..

2-Initialize Game (Colors, Fonts, etc.):

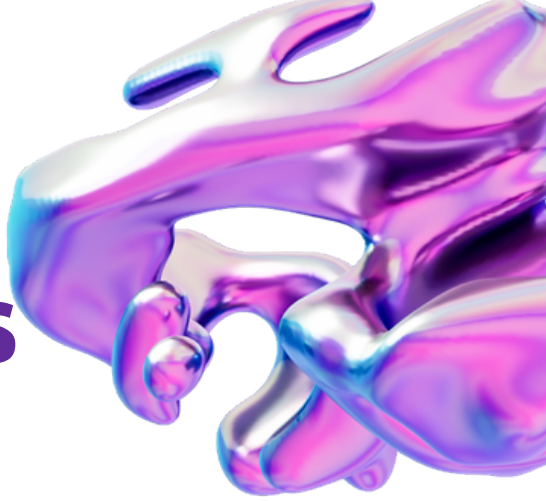
Set the game's initial parameters, including text typefaces and element colors.

3-Game Loop:

the main loop, which keeps going throughout the game. It manages events, modifies the state of the game, and refreshes the screen.



How AI Approch Works in Our Program ?



4-Check for Events (Quit, Key Press, etc.):

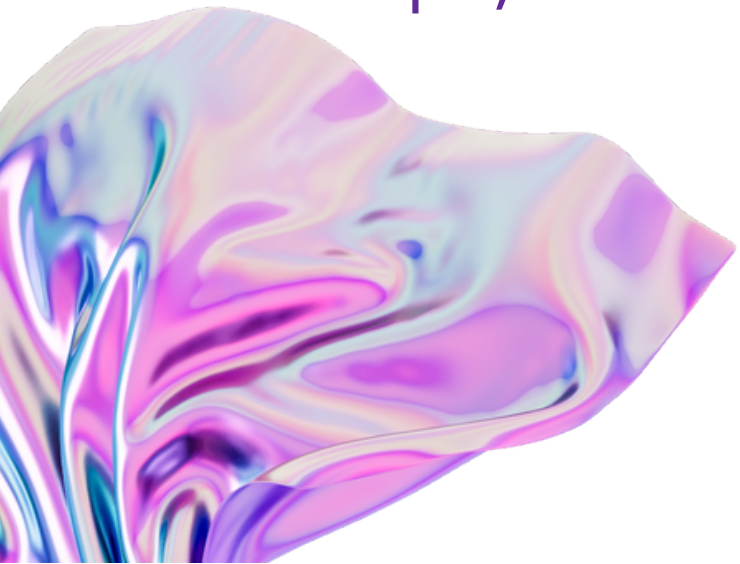
This step in the game loop monitors for events such as hitting keys or ending the game.

5-Handle Quit Event (Set game_over):

Set the game_over flag to True to end the main game loop if the player chooses to end the session.

6-Handle Key Press Event (Change Direction, etc.):

When a key is pushed, do the associated action—for example, reversing the snake's direction.



How AI Approch Works in Our Program ?

7-Check Boundaries (Game Over if Out of Bounds):

Check to see if the snake has crossed the game window's borders. In that case, the game is over.

8-Move Snake (Update Snake Position):

Adapt the snake's position according to its current direction.

9-Check Collision (Game Over if Collision):

Verify whether the snake has struck itself. In that case, the game is over.

10-Draw Food (Randomize if Eaten):

On the screen, sketch the food. Change the snake's location if it eats the food.

How AI Approch Works in Our Program ?

11-Update Snake List (Maintain Snake Length):

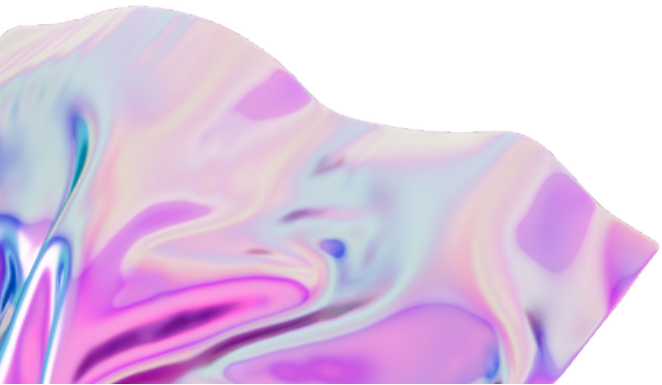
Refresh the list to reflect the snake's body while preserving its length.

12-Check Food Collision (Increase Snake Length):

Verify whether the head of the snake has struck any food. If that's the case, lengthen the snake.

13-Update Display (Draw Snake, Food, Scores):

To see the new snake, food, and score positions, redraw the game window.



How AI Approch Works in Our Program ?

14-Control Game Speed (Use Clock.tick):

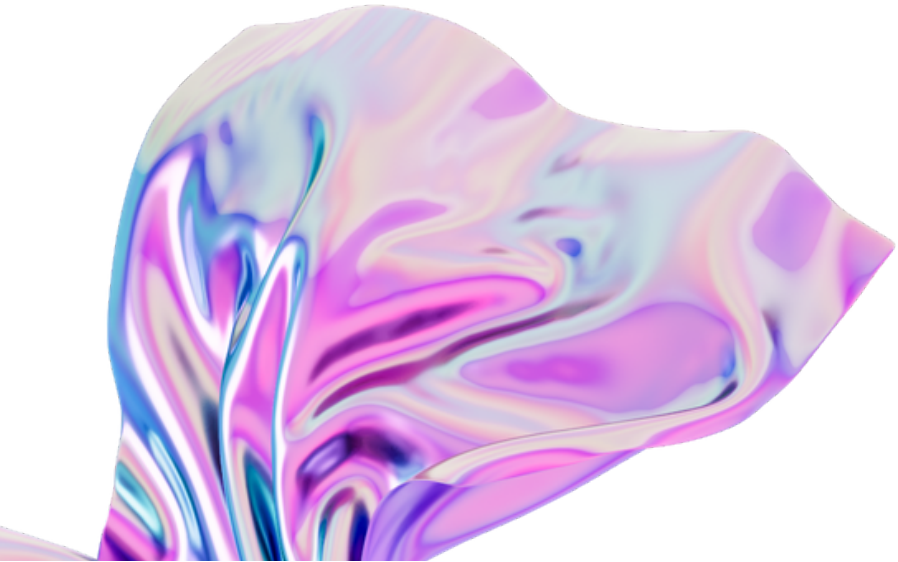
Use the clock.tick function to control the game's speed. This aids in managing the game's overall speed and frame rate.

15-Repeat Game Loop:

Go back to the game loop and carry on this way, letting the game run nonstop until you meet specific requirements (such ending the game or quitting).

16-End Game:

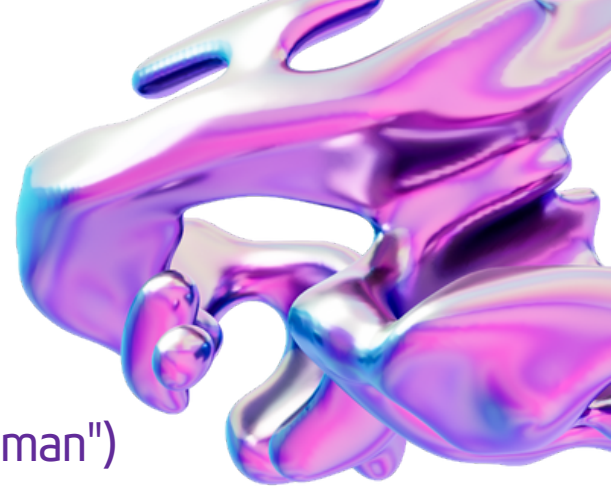
This marks the conclusion of the program and the end of the Snake game.



Code Program

```
#pygame library that helps with the game environment
import pygame
#random library to generate random numbers
import random
#Initializes all of the imported Pygame modules
pygame.init()
#all colors are RGB
#the title for the scores
scores = (82, 68, 255)
#the snake body color
sbody = (60, 179, 113)
#the text for losing the game
lost = (255, 0, 0)
#the color for the square to obtain
food = (255, 165, 0)
#the background color
background = (233, 255, 212)
#Display window height and width
box_w = 600
box_h = 600
#Displaying the window title with the height + width
box = pygame.display.set_mode((box_w, box_h))
pygame.display.set_caption('Ai Project-Snake Game')
#Helps track time for the snake speed
clock = pygame.time.Clock()
#Snake size
s_size = 10
```

Code Program

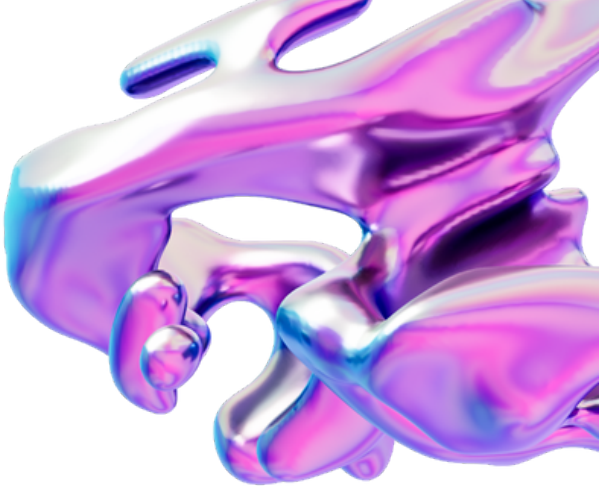


```
#Defining the font for the titles
LT = pygame.font.SysFont("Helvetica", 30,"roman")
ST = pygame.font.SysFont("Helvetica", 30)
#printing the score for the player to view the count
def Scoring(score):
  value = ST.render(str(score), True, scores)
  box.blit(value, [0, 0])
#drawing a recangular with the desired color and size
def TSnake(sblock, snake_list):
  for x in snake_list:
    pygame.draw.rect(box,sbody, [x[0], x[1], sblock, sblock])
#displaying the message for the losing
def messg(msg, color):
  mesg = LT.render(msg, True, color)
  box.blit(mesg, [box_w / 6, box_h / 3])
#intializing the loop
def Loop():
  #intialize them to false
  gOver = False
  gClose = False
  #dividing the width and height by 2
  x1 = box_w/2
  y1 = box_h/2
  #intializing the x and y positions to 0
  x2 = 0
  y2 = 0
```

Code Program

```
#creating an empty list for the snake and the length
  sList = []
  LSnake = 1
  #The generated random number is then divided by 10.0
  #and rounded to the nearest multiple of 10 using the round function
  foodx = round(random.randrange(0, box_w - s_size) / 10.0) * 10.0
  foody = round(random.randrange(0, box_h - s_size) / 10.0) * 10.0
#moving the snake with the arrow keybind while the player didnt lose
  while not gOver:
    #closing the game requires the user to press q or r to restart
    while gClose:
      #filling the background
      box.fill(background)
      #printing the message
      messg("Press R-to Restart | Q-to Quit", lost)
      Scoring(LSnake - 1)
      #updating the score and the display
      pygame.display.update()
      #linking the keybinds with the action
      for event in pygame.event.get():
        #q to quit,r to restart
        if event.type == pygame.KEYDOWN:
          if event.key == pygame.K_q:
            gOver = True
            gClose = False
          if event.key == pygame.K_r:
            Loop()
```

Code Program



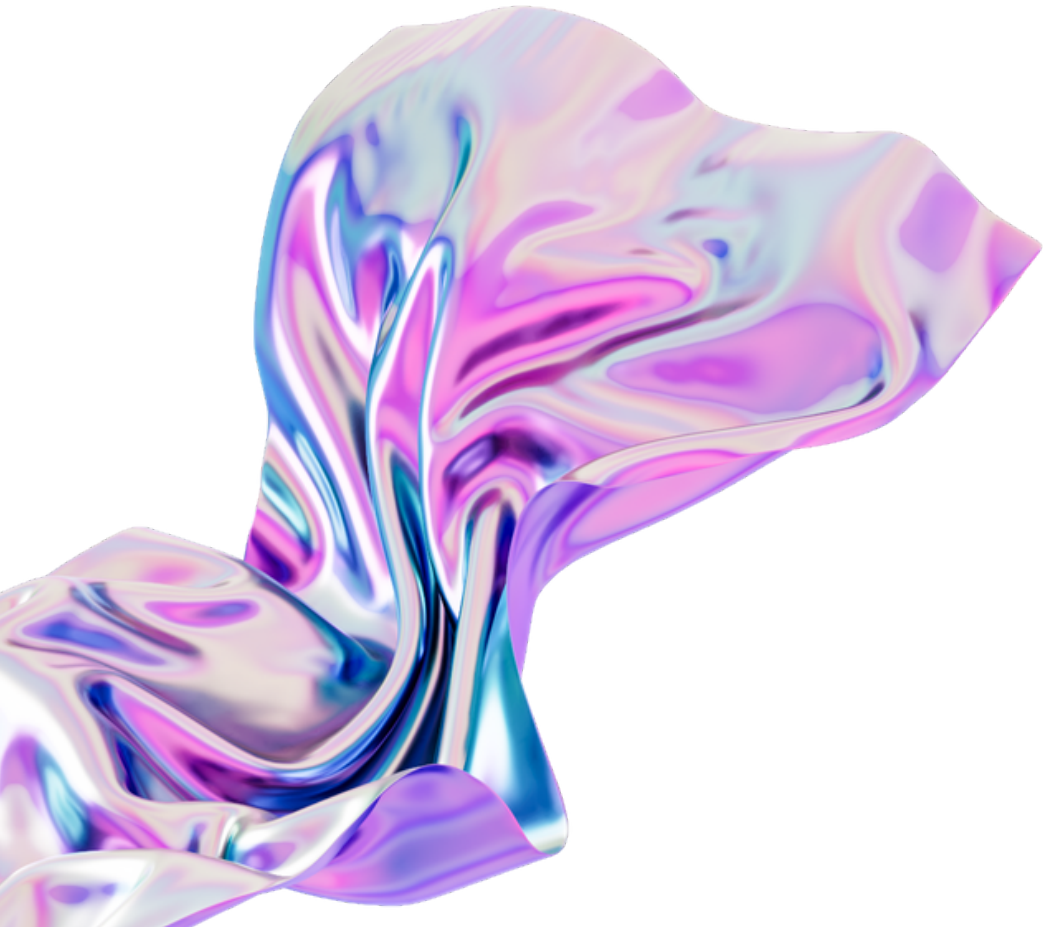
```
#if the event wanted is quit = quit
    for event in pygame.event.get():
      if event.type == pygame.QUIT:
        gOver = True
      #to start the moving it is required to go down
      if event.type == pygame.KEYDOWN:
        #if the event is key left = go left
        if event.key == pygame.K_LEFT:
          x2 = -s_size
          y2 = 0
        #if the event is key right = go right
        elif event.key == pygame.K_RIGHT:
          x2 = s_size
          y2 = 0
        #if the event is key up = go up
        elif event.key == pygame.K_UP:
          y2 = -s_size
          x2 = 0
          #if the event is key down = go down
        elif event.key == pygame.K_DOWN:
          y2 = s_size
          x2 = 0
    #if the player hits the boundaries of the screen, then he loses
    if x1 >= box_w or x1 < 0 or y1 >= box_h or y1 < 0:
      gClose = True
    x1 += x2
    y1 += y2
```

Code Program

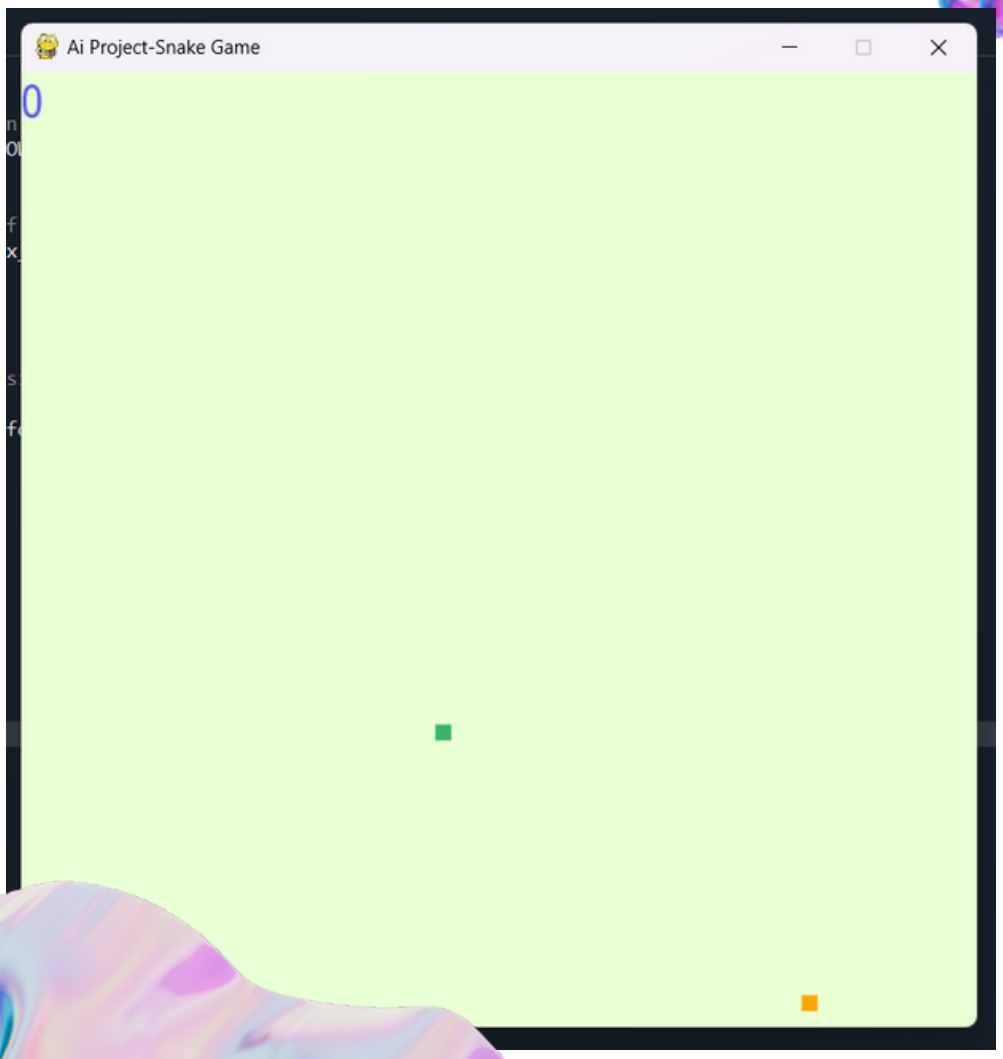
```
 #drawing the game(snake, food) with sizes
    box.fill(background)
    pygame.draw.rect(box, food, [foodx, foody, s_size, s_size])
    #increasing the body of the snake after eating the food with the
intial size
    sHead = []
    sHead.append(x1)
    sHead.append(y1)
    sList.append(sHead)
    #if the length of the list is greater than the value of the length
    #delete the first element in the list
    if len(sList) > LSnake:
      del sList[0]
    for x in sList[:-1]:
      #represents the head or starting point of the snake. If there is a
match
      #the code inside the if statement will be executed
      if x == sHead:
        gClose = True
    #The function takes the size and the list
    TSnake(s_size, sList)
    #This function call invokes the Scoring function and passes the
argument LSnake - 1
    Scoring(LSnake - 1)
    pygame.display.update()
    #two variables foodx and foody are being assigned random values
within a
    #specified range and then rounded to the nearest multiple of 10
```

Code Program

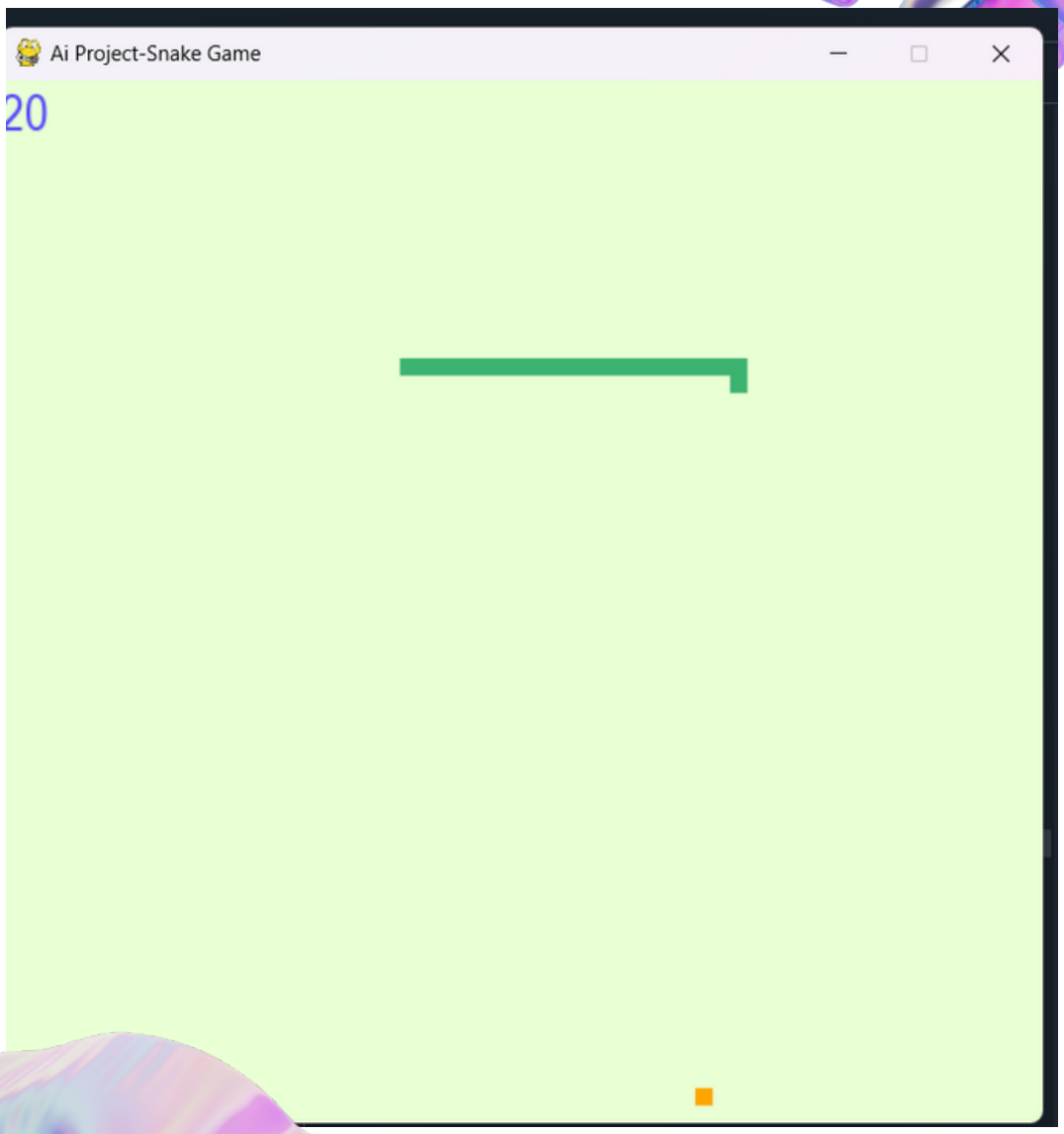
```
#This line generates a random number between
0 and box_w - s_size
 if x1 == foodx and y1 == foody:
 foodx = round(random.randrange(0, box_w -
s_size) / 10.0) * 10.0
 foody = round(random.randrange(0, box_h -
s_size) / 10.0) * 10.0
 #increment +4 for the score and length
 LSnake += 4
 #The tick method is used to regulate the frame
rate
 clock.tick(s_size)
 pygame.quit()
 quit()
Loop()
```



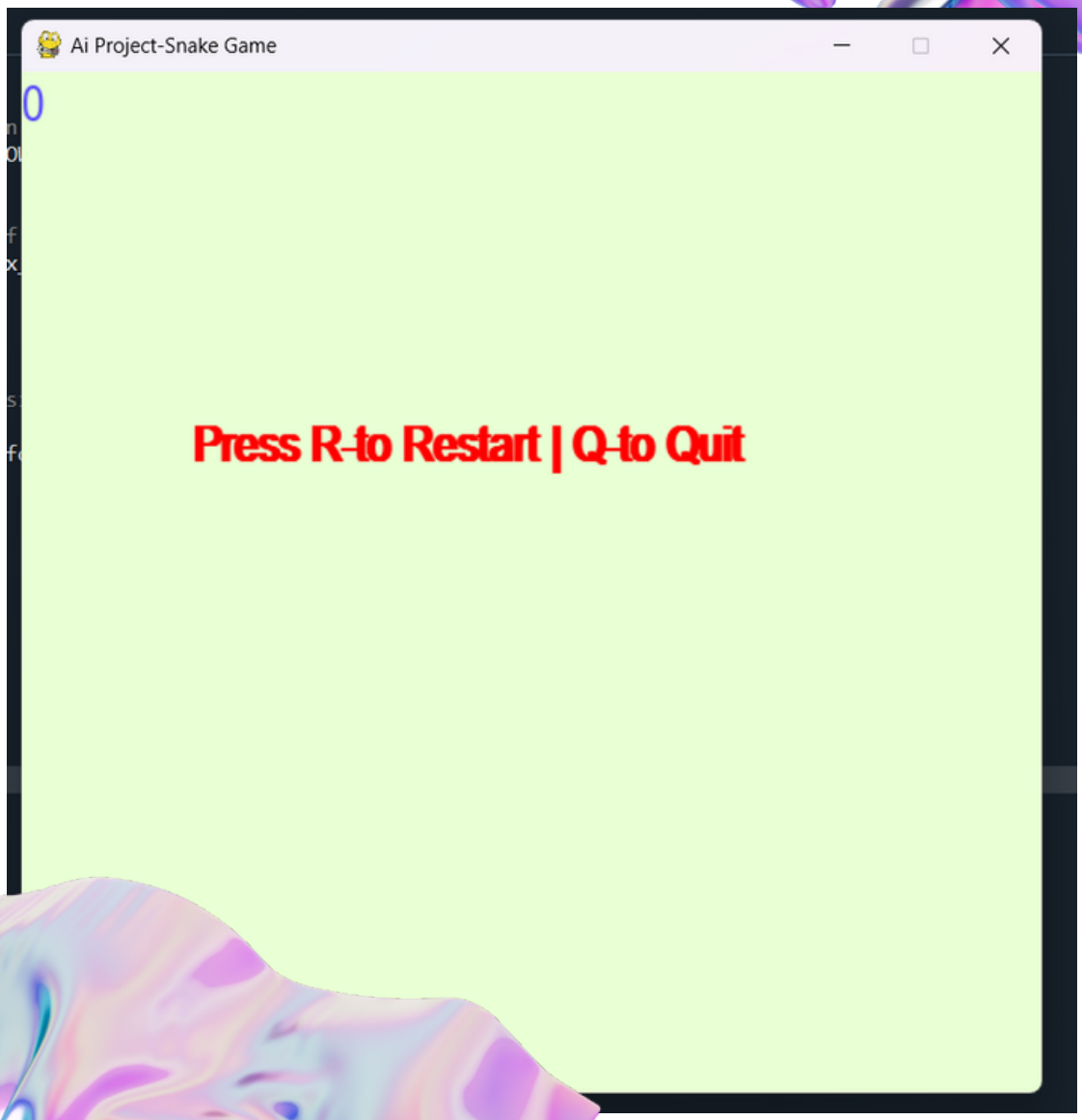
Code Output



Code Output



Code Output



Conclusion

AI implementation in the Snake game necessitates a blend of learning methods and rule-based strategies. The AI agent may learn from its experiences and continuously enhance its performance by utilizing a dynamic and adaptable strategy, demonstrating the promise of AI in classic video games.