Inspiring Excellence

CSE 423 : Computer Graphics
Project Title : Snake Game

Section:02 **Group: 03** **Session: Fall 23**	
Name	**ID**
Aurchi Roy	20341010
Anika Islam	21101298
Sreya Majumder	21201742

Main Features

1. Snake feeds on 2 types of foods with different scores
    (a) Food Type 1 : Yellow colored circles => Score = 10
    (b) Food Type 2 : Changing colored squares => Score = 30

2. Length of snake increases with increase in food intake
    (a) Length of the snake from its back.

3. Head and tail collision ends the game
    (a) Head and tail (any part of the body) collides at a point, the game terminates.

Other features:

1. Game maintained by play-pause button, restart button, program termination button
    (a) On clicking on the play button, the game resumes and replaces the play button with a pause button.
    (b) On clicking on the pause button, pauses the game and the play button appears again.
    (c) On clicking on the restart button, restarts the game.
    (d) On clicking on the program termination button, the window screen closes.

2. Display mode controlled by keyboard
    (a) “m” key on the keyboard changes the display to Light Mode.
    (b) “n” key on the keyboard changes the display to Dark Mode.

3. Snake movement controlled by both keyboard and arrow keys.
    (a) Upward movement is controlled by the “u” key on the keyboard and “up” arrow key.
    (b) Downward movement is controlled by the “d” key on the keyboard and “down” arrow key.
    (c) Leftward movement is controlled by the “l” key on the keyboard and “left” arrow key.
    (d) Rightward movement is controlled by the “r” key on the keyboard and “right” arrow key.

4. Scoreboard is displayed on the terminal.

Code

```
# Import
from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *
import random
import math

#Necessary Initialization
snake_state = [(10, 20), (10, 21), (10, 22)]
food_coordiate = (random.randint(0, 49), random.randint(0, 49))
food_type = random.choice(["circle", "square"])
direction = (1, 0)
score = 0
bg_color = (0.0,0.0,0.0,0.0)
is_game_paused = False
game_over = False


# Find Zone of the points
def findZone(x1, y1, x2, y2):
   dx = x2 - x1
   dy = y2 - y1
   zone = None
   if (abs(dx) >= abs(dy) and dx >= 0 and dy >= 0):
       zone = 0
   elif (abs(dy) > abs(dx) and dx >= 0 and dy >= 0):
       zone = 1
   elif (abs(dy) > abs(dx) and dx <= 0 and dy >= 0):
       zone = 2
   elif (abs(dy) <= abs(dx) and dx <= 0 and dy >= 0):
       zone = 3
   elif (abs(dx) <= abs(dy) and dx <= 0 and dy <= 0):
       zone = 4
   elif (abs(dy) > abs(dx) and dx <= 0 and dy <= 0):
       zone = 5
   elif (abs(dy) >= abs(dx) and dx >= 0 and dy <= 0):
       zone = 6
   elif (abs(dx) >= abs(dy) and dx >= 0 and dy <= 0):
```

```
        zone = 7
    return zone


# Other Zones --> Zone 0
def forConvertZone(x, y, zone):
    if zone == 0:  # Zone 0 --> Zone 0
        return x, y
    elif zone == 1:  # Zone 1 --> Zone 0
        return y, x
    elif zone == 2:  # Zone 2 --> Zone 0
        return y, -x
    elif zone == 3:  # Zone 3 --> Zone 0
        return -x, y
    elif zone == 4:  # Zone 4 --> Zone 0
        return -x, -y
    elif zone == 5:  # Zone 5 --> Zone 0
        return -y, -x
    elif zone == 6:  # Zone 6 --> Zone 0
        return -y, x
    elif zone == 7:  # Zone 7 --> Zone 0
        return x, -y


# Zone 0 --> Other zones
def backConvertZone(x, y, zone):
    if zone == 0:  # zone 0 --> zone 0
        return x, y
    elif zone == 1:  # zone 0 --> zone 1
        return y, x
    elif zone == 2:  # zone 0 --> zone 2
        return -y, x
    elif zone == 3:  # zone 0 --> zone 3
        return -x, y
    elif zone == 4:  # zone 0 --> zone 4
        return -x, -y
    elif zone == 5:  # zone 0 --> zone 5
        return -y, -x
    elif zone == 6:  # zone 0 --> zone 6
        return y, -x
```

```
    elif zone == 7:  # zone 0 --> zone 7
        return x, -y



# Midpoint Line Algorithm
def drawLines(x1, y1, x2, y2):
    zone = findZone(x1, y1, x2, y2)
    conv_x1, conv_y1 = forConvertZone(x1, y1, zone)
    conv_x2, conv_y2 = forConvertZone(x2, y2, zone)
    dx = conv_x2 - conv_x1
    dy = conv_y2 - conv_y1
    d = (2 * dy) - dx
    incrNE = 2 * (dy - dx)
    incrE = 2 * dy


    while conv_x1 < conv_x2:
        if d < 0:
            conv_x1 += 1
            d += incrE
        else:
            conv_x1 += 1
            conv_y1 += 1
            d += incrNE
        temp_x, temp_y = backConvertZone(conv_x1, conv_y1, zone)
        glVertex2f(temp_x, temp_y)



##############################################################################
############################



# Midpoint Circle Algorithm
def drawCircle(cx, cy, r):
    x = 0
    y = r
    d = 1 - r
```

```
  while x <= y:
      glVertex2f(cx + x, cy + y)
      glVertex2f(cx - x, cy + y)
      glVertex2f(cx + x, cy - y)
      glVertex2f(cx - x, cy - y)
      glVertex2f(cx + y, cy + x)
      glVertex2f(cx - y, cy + x)
      glVertex2f(cx + y, cy - x)
      glVertex2f(cx - y, cy - x)

      incrE = 2 * x + 3
      incrSE = 2 * (x - y) + 5

      if d <= 0:
          d = d + incrE
      else:
          y -= 1
          d = d + incrSE
      x += 1




##############################################################################
############################


# Draw circle
def allCircles(cx, cy, r):
 n = 400
 r = r
 drawCircle(cx, cy, r)
 a = r
 r = a // 2
 theta = (math.pi * 2) / n
 i = 0
 while i <= n:
     x = r * math.cos(theta * i)
```

```
    y = r * math.sin(theta * i)
    drawCircle(cx + x, cy + y, r)
    i += 1


################################################################################
##############################


# Draw  square
def showSquare(x, y, size):
   width = size//2

   # Top Edge
   drawLines(x - width, y + width, x + width, y + width)

   # Bottom Edge
   drawLines(x - width, y - width, x + width, y - width)

   # Left Edge
   drawLines(x - width, y - width, x - width, y + width)

   # Right edge
   drawLines(x + width, y - width, x + width, y + width)


   # Fill the square
   for i in range(x - width, x + width):
       drawLines(i, y - width, i, y + width)


################################################################################
##############################

# Draw the snake
def drawSnake(snake):
    if (not game_over) :
        glColor3f(0.0,0.8,0.8)
    else :
        glColor3f(1.0,1.0,1.0)
```

```
    head_x, head_y = snake[-1]
    allCircles(head_x * 10 + 5, head_y * 10 + 5, 7)
    if (not game_over) :
        glColor3f(0.5,0.0,0.5)
    else :
        glColor3f(1.0,0.8,0.8)
    for seg in snake[:-1]:
        x, y = seg
        showSquare(x * 10 + 5, y * 10 + 5, 10)


################################################################################
##############################

# Draw the food (food type : circle and square)
def showFood(food, food_type):
   x, y = food
   glBegin(GL_POINTS)
   if food_type == "circle": # score = 10
       glColor3f(1.0,1.0,0.0)
       allCircles(x * 10 + 5, y * 10 + 5, 4)

   elif food_type == "square": # score = 30
       glColor3f(random.random(),random.random(),random.random())
       showSquare(x * 10 + 5, y * 10 + 5, 6)
   glEnd()


################################################################################
##############################

#Restart Button(Left Arrow Button)
def drawRestartButton():
   glColor3f(0.0, 1.0, 1.0)
   drawLines(20, 460, 60, 460)
   drawLines(20, 460, 40, 440)
   drawLines(20, 460, 40, 480)
```

```
#############################################################################
#############################

#Play Button
def drawPlayButton():
   glColor3f(1.0, 0.75, 0.0)
   drawLines(245, 440, 245, 480)
   drawLines(245, 440, 275, 460)
   drawLines(245, 480, 275, 460)

#############################################################################
#############################

#Pause Button
def drawPauseButton():
   glColor3f(1.0, 0.75, 0.0)
   drawLines(245, 440, 245, 480)
   drawLines(255, 440, 255, 480)


#############################################################################
#############################

#Cross Button (Red Cross Button)
def drawExitButton():
   glColor3f(1.0, 0.0, 0.0)
   drawLines(440, 440, 480, 480)
   drawLines(440, 480, 480, 440)

#############################################################################
#############################


# Score Calculation + Game Update
def animate(_):
   global snake_state, food_coordiate, food_type, direction, score,
game_over

   if not game_over:
```

```
        if not is_game_paused:
            head_x, head_y = snake_state[-1]
            new_head_x = (head_x + direction[0]) % 50
            new_head_y = (head_y + direction[1]) % 50


            if (new_head_x, new_head_y) in snake_state: # head-body
collision  ---> game over
                game_over = True
                glutPostRedisplay()
                return


            if (new_head_x, new_head_y) == food_coordiate: # length
increases with food intake + food types with different scores
                if food_type == "square":
                    score += 30
                else:
                    score += 10
                food_coordiate = (random.randint(0, 49), random.randint(0,
49))
                food_type = random.choice(["circle", "square"])
                print(f"Score:{score}\n======")
            else:
                snake_state.pop(0)


            snake_state.append((new_head_x, new_head_y))
            glutPostRedisplay()
            glutTimerFunc(150, animate, 0)
        else:
            glutPostRedisplay()



##############################################################################
##############################


# Keyboard input function (Background Color Change)
```

```
def keyboardListener(key, x, y):
   global direction,bg_color

   if (not is_game_paused) :

       #snake movement using key
       if key == b"u":
           if direction[1] == 0:
               direction = (0, 1)
       elif key == b"d":
           if direction[1] == 0:
               direction = (0, -1)
       elif key == b"l":
           if direction[0] == 0:
               direction = (-1, 0)
       elif key == b"r":
           if direction[0] == 0:
               direction = (1, 0)


       # bg color change
       if key == b"m" :
           bg_color = (1.0,1.0,1.0,1.0)
           print(f"Light mode activated\n=====\n")
       elif key == b"n" :
           bg_color = (0.0,0.0,0.0,0.0)
           print(f"Dark mode activated\n=====\n")

       glutPostOverlayRedisplay()


##########################################################################
############################

# Arrow Keys For Snake Movement
def specialKeyListener(key, x, y):
 global direction
 if not is_game_paused :
    if key == GLUT_KEY_UP:
        if direction[1] == 0:
```

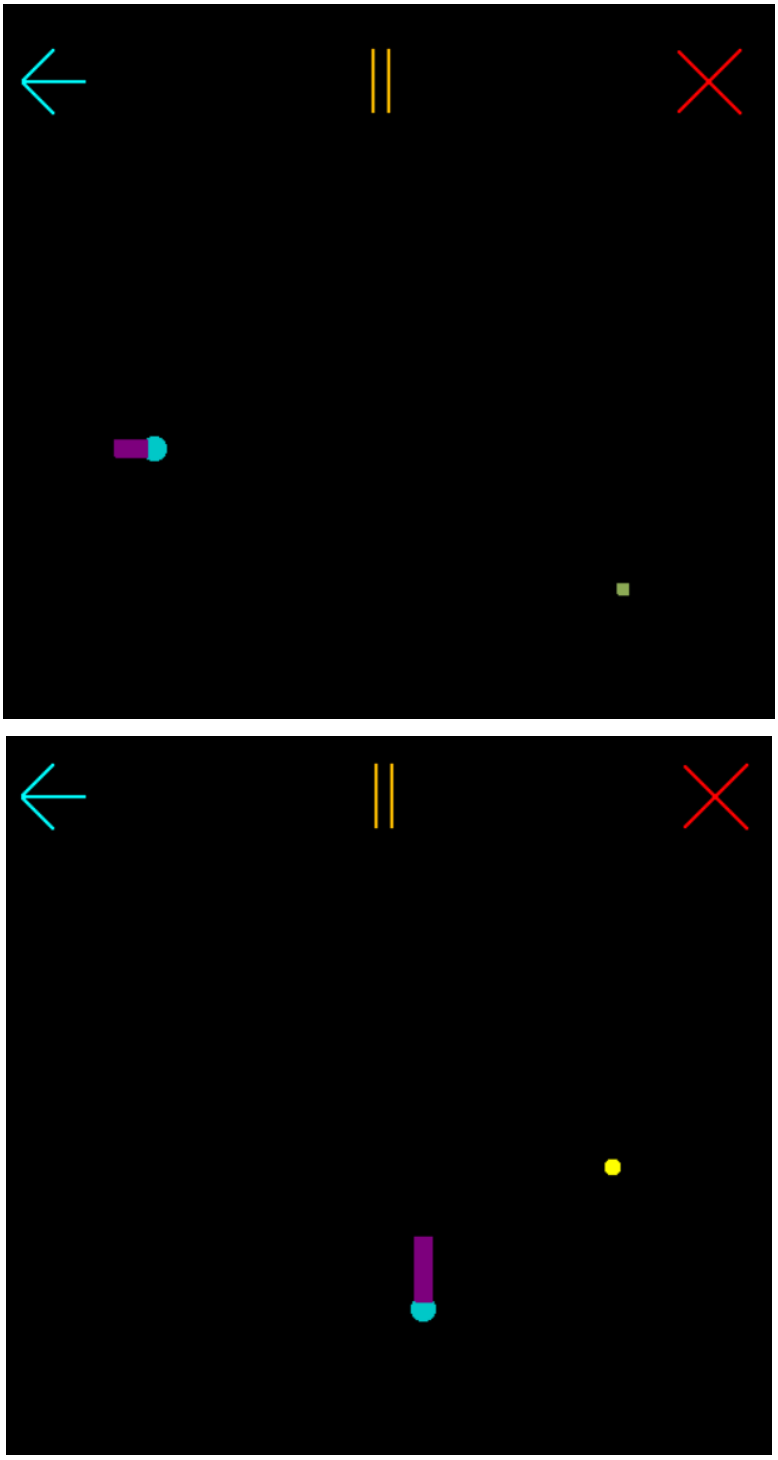
```
            direction = (0, 1)
    elif key == GLUT_KEY_DOWN:
        if direction[1] == 0:
            direction = (0, -1)
    elif key == GLUT_KEY_LEFT:
        if direction[0] == 0:
            direction = (-1, 0)
    elif key == GLUT_KEY_RIGHT:
        if direction[0] == 0:
            direction = (1, 0)


################################################################################
############################



#Restart Game Function
def restartGame():
   global score, snake_state, food_coordiate, direction, game_over,
is_game_paused
   snake_state = [(10, 20), (10, 21), (10, 22)]
   food_coordiate = (random.randint(0, 49), random.randint(0, 49))
   direction = (1, 0)
   if game_over:
       game_over = False
       glutTimerFunc(150, animate, 0)


#Restart, Play, Pause, Exit Button Functionality
def mouseListener(button, state, x, y):
  global is_game_paused, wind, game_over,score
  # Convert Y coordinate
  y = 500 - y

  # if 3 buttons are clicked
  if button == GLUT_LEFT_BUTTON and state == GLUT_UP:
      if is_game_paused == False:
          # Check if the click is within the pause button area
           if 245 <= x <= 255 and 440 <= y <= 480 and game_over== False:
```

```
            is_game_paused = True  # Pause the game
            glutTimerFunc(150, animate, 0)
            print(f"Paused\n=====\nTotal Score: {score}\n=====")
        # Check if the click is within the left arrow area
        elif 20 <= x <= 60 and 440 <= y <= 480:
            score = 0
            print(f"Starting Again!!!\n=====\nScore: {score}\n=====")
            restartGame()
        # Check if the click is within the exit button area
        elif 440 <= x <= 480 and 440 <= y <= 480:
            game_over = True
            print(f"Good Bye!!!\n=====\nScore: {score}\n=====\n")
            glutDestroyWindow(wind)


    else:
        # Check if the click is within the play button area
        if 245 <= x <= 275 and 440 <= y <= 480 and game_over == False:
            is_game_paused = False  # Resume the game
            print(f"Resumed\n=====\nScore:{score}\n=====")
            glutTimerFunc(150, animate, 0)



# OpenGL display function
def showScreen():
  global is_game_paused,score
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
  glLoadIdentity()
  iterate()
  glClearColor(*bg_color)  # Black Background
  glClear(GL_COLOR_BUFFER_BIT)


  if is_game_paused:
      glBegin(GL_POINTS)
      drawPlayButton() #Game  Paused --> Play Button
      glEnd()
```

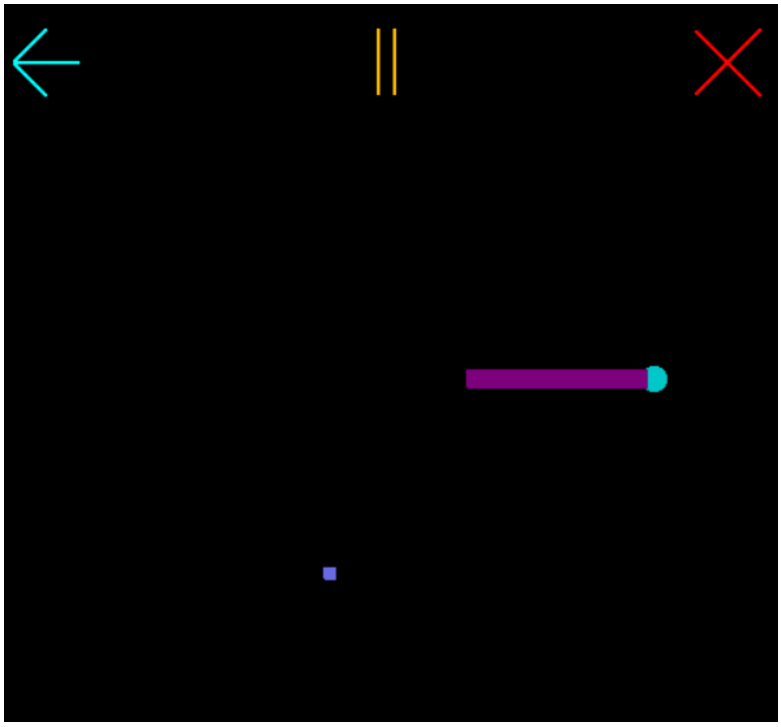
```
  else :
      glBegin(GL_POINTS)
      drawPauseButton() #Game Resume --> Pause Button
      glEnd()


  glBegin(GL_POINTS)
  drawRestartButton() #Restart Button
  glEnd()


  glBegin(GL_POINTS)
  drawExitButton() #Exit Button
  glEnd()


  glPointSize(2)
  glBegin(GL_POINTS)
  drawSnake(snake_state)
  glEnd()
  if (game_over!= True):
      showFood(food_coordiate, food_type)
  else:

       print(f"Game Over\n=====\nTotal Score:{score}\n=====\n")

  glutSwapBuffers()




def iterate():
  glViewport(0, 0, 500, 500)
  glMatrixMode(GL_PROJECTION)
  glLoadIdentity()
  glOrtho(0.0, 500, 0.0, 500, 0.0, 1.0)
  glMatrixMode(GL_MODELVIEW)
  glLoadIdentity()
```

```
glutInit()
glutInitDisplayMode(GLUT_RGBA)
glutInitWindowSize(500, 500)
glutInitWindowPosition(0, 0)
wind = glutCreateWindow(b"Lab Project - Snake Game")
glutDisplayFunc(showScreen)
glutMouseFunc(mouseListener)
glutSpecialFunc(specialKeyListener)
glutKeyboardFunc(keyboardListener)
glutTimerFunc(0, animate, 0)
glutMainLoop()
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
glLoadIdentity()
iterate()
```

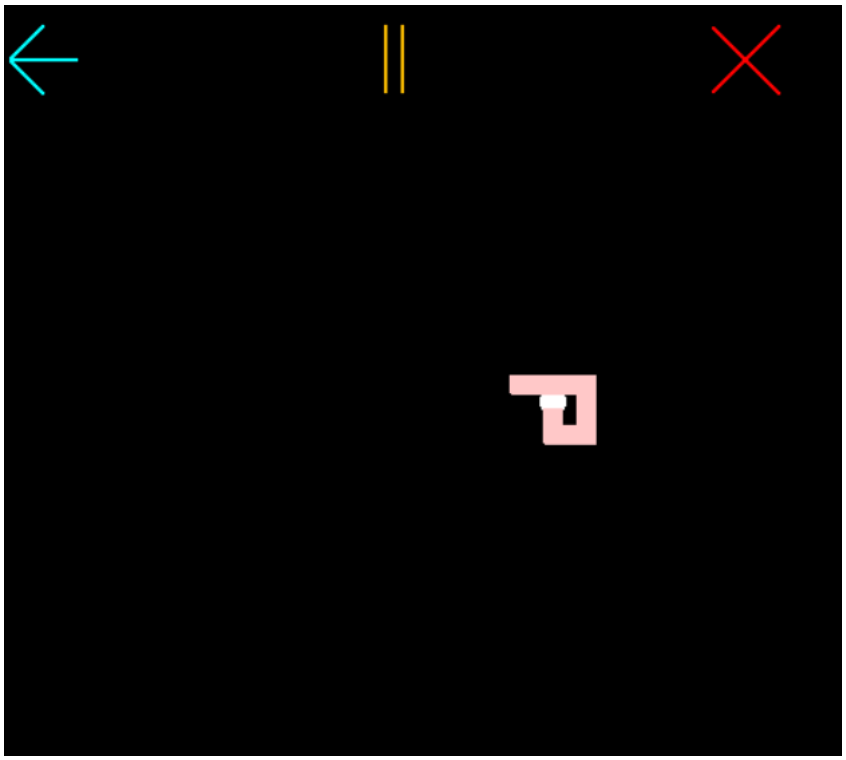
Snapshots of Features



Feature 1 : Two Types of Foods



Feature 2 : Length of snake increases with food intake

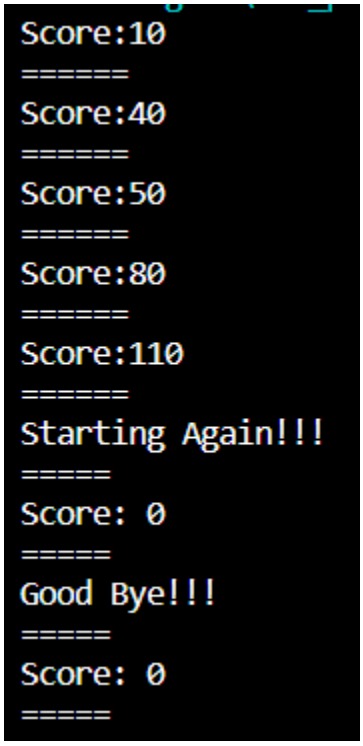


Feature 3 : Head to tail collision ends the game

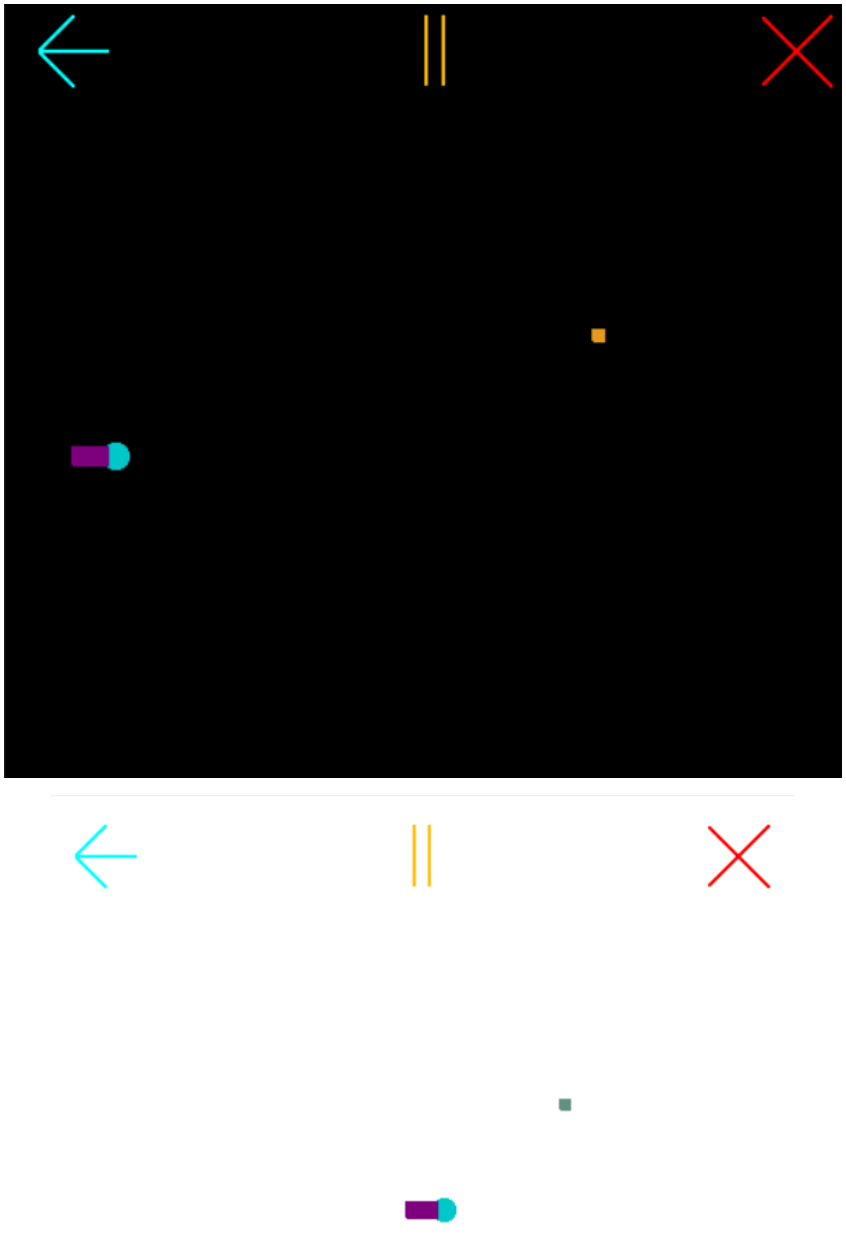
Feature 4 : Game pauses (clicking on pause button)

Feature 5 : Game resumes by “clicking on play button”

Feature 6 : Game restarts using "clicking on arrow button"



Feature 7 : Window screen closes "clicking on cross button"



Feature 8: Light (clicking “m”) & Dark Mode (clicking “n”)