
Hebi the Snake Game

Hebi the Snake Game

Report

Group: 25

Introduction:

Hebi the Snake Game is built by using the “pygame” library of python. Our team has put a modern spin on the retro game available in old nokia games using Japanese elements.

Rules:

- The goal of the game is to make “hebi” as long as possible.
- Hebi dies if it crashes in to itself.

Explanation:

The first step in making this game was to install the pygame module. We start the coding process by calling some libraries.

The use of each library shall be explained later.

```
import pygame
import sys  # system
import os
import random
```

To make hebi and the food for hebi we defined two classes. Each classes have multiple functions for smooth running of the game.

Class Hebi ():

The Hebi class consists of the following functions:

```
class Hebi():
    def __init__(self):

    def resetgame(self):

    def head(self):

    def keymovemnet(self):

    def drawhebi(self, surface):

    def turning(self, coordinate):

    def movement(self):
```

- **def __init__(self):**

```
    def __init__(self):
        self.length = 1
        self.position = [(screen_height/2, screen_width/2)]  # middle
        self.direction = left  # initial
        self.score = 0
```

This function defines “hebi” itself. It sets the initial length conditions of hebi.

- **def resetgame(self):**

```
    def resetgame(self):
        FileDir = os.path.dirname(os.path.abspath(__file__))
        end_screen = os.path.join(FileDir, "e2.jpg")
        end = pygame.image.load(end_screen)
        surface.fill((255, 255, 255))
        surface.blit(end, [70, 0])
        pygame.display.flip()
        end_music = os.path.join(FileDir, "omae.mp3")
        pygame.mixer.music.load(end_music)
```

```
        pygame.mixer.music.play(-1)
        while True:
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    pygame.quit()
                    sys.exit()
                elif event.type == pygame.KEYDOWN:
                    if event.key == pygame.K_r:
                        main()
```

This function is called when hebi crashes into itself. This function also displays a new screen and new music. If the “R” is pressed the main loop starts again.

- **def head(self):**

```
    def head(self):
        return self.position[0]
```

This is a fairly simple, but important function. This function returns the position of the head i.e. first “block” of hebi. This function is called when we need to add a new block to hebi or when we need to check if hebi has crashed itself or not.

- **def keymovment(self):**

```
    def keymovemnet(self):
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            elif event.type == pygame.KEYDOWN:
                if event.key == pygame.K_DOWN:
                    self.turning(down)
                elif event.key == pygame.K_RIGHT:
                    self.turning(right)
                elif event.key == pygame.K_UP:
                    self.turning(up)
                elif event.key == pygame.K_LEFT:
```

```
                    self.turning(left)
```

What this function does is that it checks the “events” happening in the pygame window. This function enables the player to exit the game at any time by pressing the “x” button. This function also changes the direction of hebi according to which ever key the player pressed by calling another function which shall be discussed later.

- **def drawhebi(self, surface):**

```
    def drawhebi(self, surface):
        for section in self.position:  # x y width height
            r = pygame.Rect((section[0], section[1]), (gridsize, gridsize))
            pygame.draw.rect(surface, ((34, 67, 89)), r,
                             4)  # where colour area fill
```

This function draws the body of hebi block by block. “Gridsize” is a variable defined in the code that determines the size of each block hebi is made up of. To get the “hollow look” of the snake we set the shape fill to 4.

- **def turning(self, coordinate):**

```
    def turning(self, coordinate):
        if self.length > 1 and (coordinate[0]*-1, coordinate[1]*-
1) == self.direction:
            return
        else:
            self.direction = coordinate
```

This function decides which direction hebi actually moves in. If hebi is only one block long it can move in all four directions but if hebi is longer than that it cannot move back in itself. Say if hebi

is moving upwards and the player presses the down key the hebi will keep going in the same direction i.e upwards, and if the player presses the up key hebi will not change direction as it is already going in that direction.

- **def movement(self):**

```
def movement(self):
    current = self.head()
    x, y = self.direction
    new = (((current[0]+(x*gridsize)) % screen_width),
           (current[1]+(y*gridsize)) % screen_height)
    if len(self.position) > 2 and new in self.position[2:]:
        self.resetgame()
    else:
        self.position.insert(0, new)
        if len(self.position) > self.length:
            self.position.pop()
```

This is the function that adds a new block to hebi's body. It gets the x and y coordinates of the first block and uses the "gridsize" and "screen_width" variables to new blocks. This function enables hebi to go from one side of the screen and appear from the other side.

The dimensions of the pygame window are set as (480 x 480). When either the x or y coordinate becomes 0 this function brings back hebi from the other sie of the screen. Say hebi is moving leftwards and it reaches the edge of the screen,

0 + (-1* 30) % 480 = 450 (the new x-coordinte)

Class foodforhebi():

The foodforhebi class consists of the following functions

```
def __init__(self):

def random_food(self):

def draw(self, surface):
```

- **def __init__(self):**

```
def __init__(self):
    self.position = (0, 0)
    self.color = ((0, 255, 0))
    self.random_food()
```

This sets the initial conditions of the food. This function also calls another function which will be explained next.

- **def random_food(self):**

```
def random_food(self):
    self.position = (random.randint(0, grid_width-1)*gridsize,
                     random.randint(0, grid_height-1)*gridsize)
```

This function utilizes the “random” module we imported at the start. This function randomizes the position of the food to add an element of surprise.

- **def draw(self, surface):**

```
def draw(self, surface):
    r = pygame.Rect(
        (self.position[0], self.position[1]), (gridsize, gridsize))
    pygame.draw.rect(surface, ((255, 255, 255)), r)
```

This function is used to “draw” the block that will be the food. It uses the “Rect” function of pygame.

Base colour function:

```
def basecolour(surface):
    surface.fill((156, 218, 255))
```

This function is a separate function outside of both the classes that, that, when called, colors the screen with a light blue color. This function will be called in the main game loop.

Variables:

```
# the variables

screen_width = 480
screen_height = 480  # dimensions
gridsize = 30
grid_width = int(screen_width/gridsize)
grid_height = int(screen_height/gridsize)
up = (0, -1)
down = (0, 1)
left = (-1, 0)
right = (1, 0)
```

We defined a few variables which will be called multiple times in the classes and functions as well as the main game loop.

Main Game Loop:

```
ef main():
    pygame.init()  # initialization
    global surface
    surface = pygame.display.set_mode((screen_width, screen_height))
    clock = pygame.time.Clock()
    FileDir = os.path.dirname(os.path.abspath(__file__))

    #caption and icon
    pygame.display.set_caption("Hebi the snake game")
    icon_image = os.path.join(FileDir, "hebi.jpg")
    s_icon = pygame.image.load(icon_image)
    pygame.display.set_icon(s_icon)  # for the icon

    # start image
    cover_image = os.path.join(FileDir, "c2.jpg")
    cover = pygame.image.load(cover_image)
    surface.fill((255, 255, 255))
    surface.blit(cover, [70, 0])
    pygame.display.flip()

    # bm music
    bm_music = os.path.join(FileDir, "mario.mp3")
    pygame.mixer.music.load(bm_music)
    pygame.mixer.music.play(-1)
    surface.fill((255, 255, 255))
    surface.blit(cover, [70, 0])
    pygame.display.flip()
    while True:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            elif event.type == pygame.KEYDOWN:
                if event.key == pygame.K_s:
                    hebi = Hebi()
                    food = foodforhebi()
                    basecolour(surface)
                    score_record = pygame.font.SysFont("monospace", 18)

                    while True:
                        clock.tick(10)
                        hebi.keymovemnet()
                        basecolour(surface)
```

```
                        hebi.movement()
                        if hebi.head() == food.position:
                            hebi.length += 1
                            hebi.score += 1
                            food.random_food()
                        hebi.drawhebi(surface)
                        food.draw(surface)
                        surface.blit(surface, (0, 0))
                        text = score_record.render(
                            "Food eaten {0}".format(hebi.score), False, (0, 0, 0)
)
                        surface.blit(text, (5, 10))

                        pygame.display.update()


main()
```

The first thing the main loop does is initialize the pygame module and then defines the “surface”. We also set the caption, starting screen and background music. When the player presses the “S” key the game starts running.

We also added a clock function which refreshes the game screen 10 times per second. In order to record the score of the player we make a variable named “text” which displays the number of “food eaten” by hebi on the top-left corner of the pygame window.

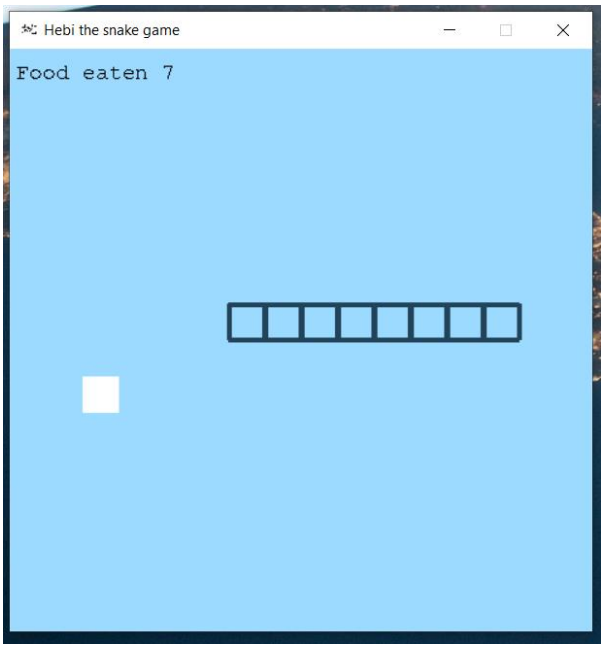
And lastly we call the pygame.display.update() function which keeps updating the game screen.

Demo Outputs:

Starting screen:

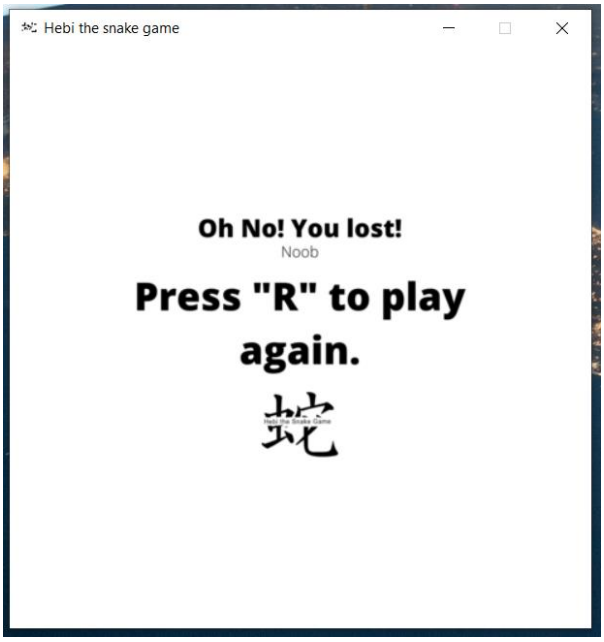
The Mario.mp3 music plays in the background

Game:



The Mario.mp3 music plays in the background

End Screen:



The omae.mp3 music plays in the background

The code:

```
import pygame
import sys  # system
import os
import random


class Hebi():
    def __init__(self):
        self.length = 1
        self.position = [(screen_height/2, screen_width/2)]  # middle
        self.direction = left  # initial
        self.score = 0

    def resetgame(self):
        FileDir = os.path.dirname(os.path.abspath(__file__))
        end_screen = os.path.join(FileDir, "e2.jpg")
        end = pygame.image.load(end_screen)
        surface.fill((255, 255, 255))
        surface.blit(end, [70, 0])
        pygame.display.flip()
        end_music = os.path.join(FileDir, "omae.mp3")
        pygame.mixer.music.load(end_music)
        pygame.mixer.music.play(-1)
        while True:
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    pygame.quit()
                    sys.exit()
                elif event.type == pygame.KEYDOWN:
                    if event.key == pygame.K_r:
                        main()

    def head(self):
        return self.position[0]

    def keymovemnet(self):
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            elif event.type == pygame.KEYDOWN:
                if event.key == pygame.K_DOWN:
                    self.turning(down)
```

```
                elif event.key == pygame.K_RIGHT:
                    self.turning(right)
                elif event.key == pygame.K_UP:
                    self.turning(up)
                elif event.key == pygame.K_LEFT:
                    self.turning(left)

    def drawhebi(self, surface):
        for section in self.position:  # x y width height
            r = pygame.Rect((section[0], section[1]), (gridsize, gridsize))
            pygame.draw.rect(surface, ((34, 67, 89)), r,
                             4)  # where colour area fill

    def turning(self, coordinate):
        if self.length > 1 and (coordinate[0]*-1, coordinate[1]*-
1) == self.direction:
            return
        else:
            self.direction = coordinate

    def movement(self):
        current = self.head()
        x, y = self.direction
        new = (((current[0]+(x*gridsize)) % screen_width),
               (current[1]+(y*gridsize)) % screen_height)
        if len(self.position) > 2 and new in self.position[2:]:
            self.resetgame()
        else:
            self.position.insert(0, new)
            if len(self.position) > self.length:
                self.position.pop()


class foodforhebi():
    def __init__(self):
        self.position = (0, 0)
        self.color = ((0, 255, 0))
        self.random_food()

    def random_food(self):
        self.position = (random.randint(0, grid_width-1)*gridsize,
                         random.randint(0, grid_height-1)*gridsize)

    def draw(self, surface):
        r = pygame.Rect(
```

```
            (self.position[0], self.position[1]), (gridsize, gridsize))
        pygame.draw.rect(surface, ((255, 255, 255)), r)


def basecolour(surface):
    surface.fill((156, 218, 255))


# the variables

screen_width = 480
screen_height = 480  # dimensions
gridsize = 30
grid_width = int(screen_width/gridsize)
grid_height = int(screen_height/gridsize)
up = (0, -1)
down = (0, 1)
left = (-1, 0)
right = (1, 0)


def main():
    pygame.init()  # initialization
    global surface
    surface = pygame.display.set_mode((screen_width, screen_height))
    clock = pygame.time.Clock()
    FileDir = os.path.dirname(os.path.abspath(__file__))

    #caption and icon
    pygame.display.set_caption("Hebi the snake game")
    icon_image = os.path.join(FileDir, "hebi.jpg")
    s_icon = pygame.image.load(icon_image)
    pygame.display.set_icon(s_icon)  # for the icon

    # start image
    cover_image = os.path.join(FileDir, "c2.jpg")
    cover = pygame.image.load(cover_image)
    surface.fill((255, 255, 255))
    surface.blit(cover, [70, 0])
    pygame.display.flip()

    # bm music
    bm_music = os.path.join(FileDir, "mario.mp3")
    pygame.mixer.music.load(bm_music)
    pygame.mixer.music.play(-1)
```

```
    surface.fill((255, 255, 255))
    surface.blit(cover, [70, 0])
    pygame.display.flip()
    while True:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            elif event.type == pygame.KEYDOWN:
                if event.key == pygame.K_s:
                    hebi = Hebi()
                    food = foodforhebi()
                    basecolour(surface)
                    score_record = pygame.font.SysFont("monospace", 18)

                    while True:
                        clock.tick(10)
                        hebi.keymovemnet()
                        basecolour(surface)
                        hebi.movement()
                        if hebi.head() == food.position:
                            hebi.length += 1
                            hebi.score += 1
                            food.random_food()
                        hebi.drawhebi(surface)
                        food.draw(surface)
                        surface.blit(surface, (0, 0))
                        text = score_record.render(
                            "Food eaten {0}".format(hebi.score), False, (0, 0, 0)
)
                        surface.blit(text, (5, 10))

                        pygame.display.update()


main()
```

References and notes:

- We learned to use the pygame module by watching youtube tutorials and studying other programs written by other people.
- We originally used absolute paths to call the images and audios used since we did not know how to use relative paths. We looked up the syntax to use relative paths on stackoverflow.
- The game logo was not taken from the internet and was designed by our teammate.
- Both the audios used were taken from youtube videos.
- This program was written in python 3.9.

Group members:

- Fatima Alvi
- Muhammad Suleiman Qureshi
- Sumaira Khan