Snake game with pygame

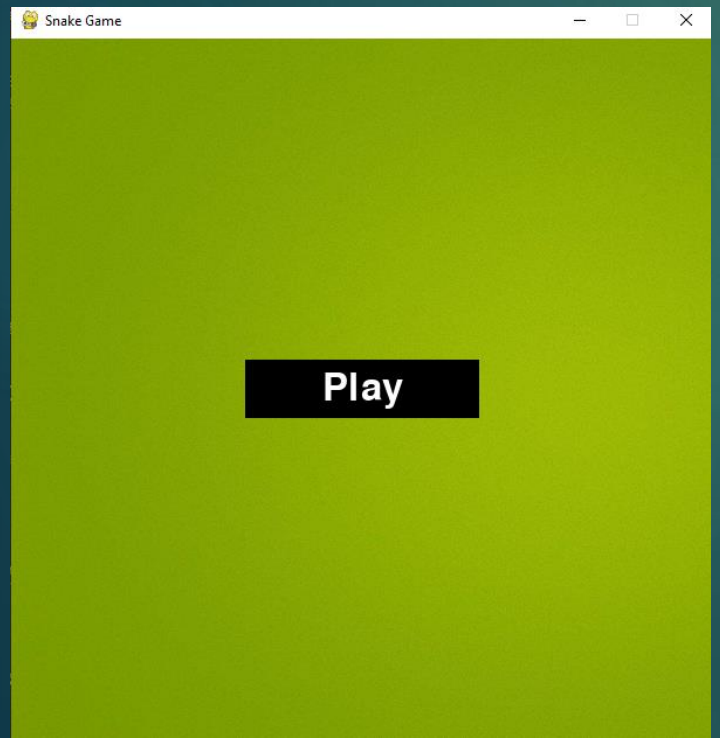
The play button

- Създаваме клас Button, който приема като аргументи, екрана върху, който ще бъде имплементират, както и какво да бъде изписано върху него като втори аргумент.
- Вътре в init метода оказваме ширината и дължината на нашия бутон. Избираме още цвят на бутона и на текста вътре, а ако искаме може и шрифт.
- След това инициализираме кой екран ще използваме и използваме функцията get_rect(), за да вземем местоположението на нашия екран, за да може след това да сложим нашия бутон по средата, като преди това разбира се създадем обект за нашия бутон.

```
import pygame.font

class Button():
    def __init__(self, screen, msg):
        """Initialize button attributes."""
        self.width, self.height = 200, 50
        self.screen = screen
        self.screen_rect = screen.get_rect()
        # Set the dimensions and properties of the button. self.wid
        self.button_color = (0, 0, 0)
        self.text_color = (255, 255, 255)
        self.font = pygame.font.SysFont(None, 48)
        # Build the button's rect object and center it.
        self.rect = pygame.Rect(0, 0, self.width, self.height)
        self.rect.center = self.screen_rect.center
        # The button message needs to be prepped only once.
        self.prep_msg(msg)
```

- Класът ще има два метода – prep_msg, който ще показва избрания надпис върху бутона ни, в случая ‘Play’ и draw_button, който „рисува“ бутона върху екрана ни.
- В prep_msg използваме font.render за да създадем нова повърхност със съобщението и цветовете, които сме избрали по-рано, втория аргумент е за ъглите на буквите. Създаваме обект от повърхността и го слагаме по среда -та на бутона.

```
def prep_msg(self, msg):
    """Turn msg into a rendered image and center text on the button."""
    self.msg_image = self.font.render(msg, True, self.text_color,
    self.button_color)
    self.msg_image_rect = self.msg_image.get_rect()
    self.msg_image_rect.center = self.rect.center

def draw_button(self):
    # Draw blank button and then draw message.
    self.screen.fill(self.button_color, self.rect)
    self.screen.blit(self.msg_image, self.msg_image_rect)
```



Накрая с фунцкията draw_button, ‚изрисуваме‘ бутона върху екрана. С fill попълваме цвета на бутона, а с blit вече поставяме бутона върху екрана.

The background

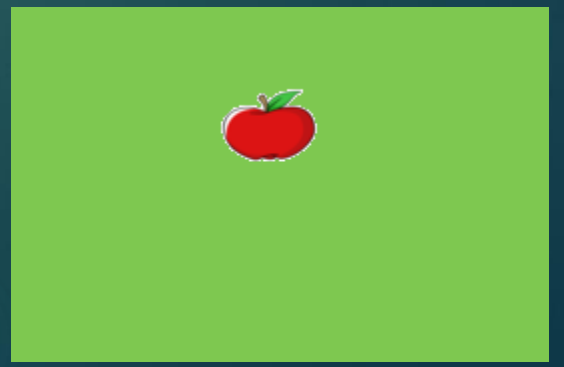
- Създаваме два класа, единия е за фона по време на игра, а другия за фона на нашето меню. Нямаме други методи в класовете, имаме два аргумента, снимката, която искаме да използваме като фон и втора аргумент, къде да се намира фона. Също така наследяваме класа Sprite, който се използва за ‚рисуване‘ върху обект.
- (Всъщност се отказах, да използвам снимка за фон, вътре в играта, защото не намерих подходяща, и използвах просто зелен цвят.)

```python
import pygame

class Background_pause(pygame.sprite.Sprite):
    def __init__(self, image_file, location):
        pygame.sprite.Sprite.__init__(self)  #call Sprite initializer
        self.image = pygame.image.load(image_file)
        self.rect = self.image.get_rect()
        self.rect.left = location
        self.rect.top = location


class Background_game(pygame.sprite.Sprite):
    def __init__(self, image_file, location):
        pygame.sprite.Sprite.__init__(self)  #call Sprite initializer
        self.image = pygame.image.load(image_file)
        self.rect = self.image.get_rect()
        self.rect.left = location
        self.rect.top = location
```

The fruit

- Създаваме клас Fruit с аргументи, settings, които са нашите настройки за играта, както и екрана ни.
- Зареждаме снимката, която искаме да използваме като плод, правим обект от нея, като използваме get_rect върху снимката, за да вземем нейните кординати.
- След това използваме randint за да сложим ябълката на случайни кординати, като гледаме да не излизаме от екрана.
- Накрая изрисуваме плода върху екрана.

```python
import pygame

import random

class Fruit:
    def __init__(self, ai_settings, screen):
        self.ai_settings = ai_settings
        self.screen = screen

        self.image = pygame.image.load('images/apple.bmp')
        self.rect = self.image.get_rect()
        self.rect.x =  random.randint(50, ai_settings.screen_width-50)
        self.rect.y =  random.randint(30, ai_settings.screen_height-30)

        self.x = float(self.rect.x)
        self.y = float(self.rect.y)

    def blitme(self):
        """Draw the apple at its current location."""
        self.screen.blit(self.image, self.rect)
```



The snake

- Създаваме клас Snake с аргумент settings – настройките на нашата игра, и слагаме змията в центъра на екрана, за сега не се движим така че, промяната на положението на змията е 0.
- Създаваме обект, който е нашата змия, с нейното положение и големина
- Създаваме лист, който ще използваме, когато змията се уголемява и задаваме първоначална големина на змията.

```
1  import pygame
2  import time
3  import sys
4  from pygame import mixer
5
6  import game_functions as gf
7
8  class Snake:
9      def __init__(self, ai_settings):
10         self.x = ai_settings.screen_width/2
11         self.y = ai_settings.screen_height/2
12         self.x_change = 0
13         self.y_change = 0
14         self.rect = pygame.Rect(self.x, self.y, ai_settings.snake_width,
15         ai_settings.snake_height)
16         self.snake_list = []
17         self.length_snake = 1
18         #self.apple_sound1 = mixer.Sound("music/mary_sound1.wav")
19
20
```

Метод за движения на змията

- Ако имаме бутон натиснат надолу, и този бутон е една от стрелкичките отиваме в съответната посока, като правим така, че да избягваме змията да се обръща рязко на 180 градуса и да се самоизяжда.
- Накрая правим съответните промени в положението на змията и го запазваме като флоат, ако бързината на змията не е цяло число.

```
51
52          self.rect.x += self.x_change
53          self.rect.y += self.y_change
54          self.x = float(self.rect.x)
55          self.y = float(self.rect.y)
56
```

```
def move_snake(self, ai_settings):
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_RIGHT and self.x_change >= 0: #

                #self.apple_sound1.play()
                self.x_change = ai_settings.snake_speed
                self.y_change = 0

            elif event.key == pygame.K_LEFT and self.x_change <= 0:
                #apple_sound = mixer.Sound("music/mary_sound1.wav")
                #self.apple_sound1.play()
                self.x_change = -ai_settings.snake_speed
                self.y_change = 0


            elif event.key == pygame.K_UP and self.y_change <= 0:
                self.y_change = -ai_settings.snake_speed
                self.x_change = 0
                #apple_sound = mixer.Sound("music/mary_sound1.wav")
                #apple_sound.play()


            elif event.key == pygame.K_DOWN and self.y_change >= 0:
                #apple_sound = mixer.Sound("music/mary_sound1.wav")
                #apple_sound.play()
                self.y_change = ai_settings.snake_speed
                self.x_change = 0
```

Другите методи на змията

- In_game фунцкията прави, така че, ако ударим една от 4те стени, играта свършва.
- draw_snake изрисува змията и опашката ѝ върху екрана.
- Фунцкията reset рестартира всичко в първоначална позиция, както изписва и издава звука ‘Game Over’.
- Със snake_update ъпдейтваме местоположението на змията, ако змията ‚захапе‘ себе си, играта приключва.
- Също така след като листа стане по-голям от змията, изтриваме от екрана предишното местоположение на змията, за да не стоят черни квадратчета по екрана

```
57    def in_game(self, ai_settings, screen):
58        difficulty = "hard"
59        if difficulty == "hard":
60            #self.x += self.x_change
61            if self.rect.x >= ai_settings.screen_width - 10:
62                #self.rect.x = 0
63                self.reset(ai_settings, screen)
64            elif self.rect.x < 0 - 5:
65                #self.rect.x = ai_settings.screen_width
66                self.reset(ai_settings, screen)
67            #self.y += self.y_change
68            if self.rect.y >= ai_settings.screen_height:
69                #self.rect.y = 0
70                self.reset(ai_settings, screen)
71            elif self.rect.y < 0 - 5:
72                #self.rect.y = ai_settings.screen_height
73                self.reset(ai_settings, screen)
74
75    def draw_snake(self, screen, ai_settings):
76        for i in range(len(self.snake_list)):
77            pygame.draw.rect(screen, ai_settings.snake_color,
78            [self.snake_list[i][0], self.snake_list[i][1], ai_settings.snake_width,
79             ai_settings.snake_height])
80            # x = snake_list[i][0]
81            # y = snake_list[i][1]
82
83        def snake_update(self, ai_settings, screen):
84            self.snake_head = []
85            self.snake_head.append(self.x)
86            self.snake_head.append(self.y)
87            self.snake_list.append(self.snake_head)
88            if len(self.snake_list) > self.length_snake:
89                del self.snake_list[0]
90
91            for x in self.snake_list[:-1]:
92                if x == self.snake_head:
93                    self.reset(ai_settings, screen)
94
```

Настройките на играта

- Създаваме клас Settings, където избираме големината на екрана, цветът на фона, големината на змията и колко бързо се движи.
- Правим ‘score‘, който да следи колко точки сме направили досега в играта.
- Слагаме game_over = True, за да може играта да стартира от менюто.

```python
import pygame

class Settings():
    """A class to store all settings for Alien Invasion."""
    def __init__(self):
        """Initialize the game's settings."""
        # Screen settings
        self.screen_width = 600
        self.screen_height = 600
        self.bg_color = (126,200,80) #Green

        # Snake settings
        self.snake_color = 0, 0, 0 #Black
        self.snake_width = 15
        self.snake_height = 15
        self.snake_speed = 8

        #Score settings
        self.score = 0
        self.score_font = pygame.font.SysFont("comicsansms", 20)
        self.score_color = (255, 255, 255) #White
        self.message_style = pygame.font.SysFont("bahnschrift", 35)
        self.message_color = (255, 0, 0)

        #Menu settings
        self.before_color = (255, 255, 255)


        self.game_over = True

```

Функции на играта

- Функцията score използваме за да изобразяваме точките ни по време на игра.
- Функцията message служи за да изобразим съобщение върху екрана, като ‚Game Over!' примерно.
- Фунцкията menu създаваме нашето меню, което е изобразено, само докато game_over = True, слагаме снимка за фон, рисуваме бутона и ъпдейтваме екрана.

```
36  def score(ai_settings, screen):
37      #Display score function
38      value = ai_settings.score_font.render("Score: " + str(ai_settings.score), True, ai_settings.score_color)
39      screen.blit(value, [10, 10])
40
41  def message(msg, color,ai_settings, screen,w ,h):
42      #Display message function
43      mesg = ai_settings.message_style.render(msg, True, color)
44      screen.blit(mesg, [ai_settings.screen_width / w, ai_settings.screen_height / h])
45
46  def menu(ai_settings, screen, background_ps, play_button):
47      while ai_settings.game_over == True:
48          #mixer.music.rewind()
49          screen.fill(ai_settings.before_color)
50          screen.blit(background_ps.image, background_ps.rect)
51
52
53          play_button.draw_button()
54          #gf.score(ai_settings, screen)     #Optional Score while in the menu
55          pygame.display.update()
56
57          for event in pygame.event.get():
58              if event.type == pygame.KEYDOWN:
59                  if event.key == pygame.K_q:
60                      sys.exit()
61                  if event.key == pygame.K_p:
62                      ai_settings.game_over = False
63              elif event.type == pygame.MOUSEBUTTONDOWN:
64                  mouse_x, mouse_y = pygame.mouse.get_pos()
65                  if play_button.rect.collidepoint(mouse_x, mouse_y):
66                      ai_settings.game_over = False
67              elif event.type == pygame.QUIT:
```

- snake_and_fruit функцията, служи за това да увеличаваме точките и да уголемяваме змията, когато змията се докосне до плода.
- Издава се звук за ядена на ябълка, всеки път, когато змията докосне плода.
- Всеки път като плода бъде изяден, се появява на ново място
- Увеличаваме големината на змията и точките.

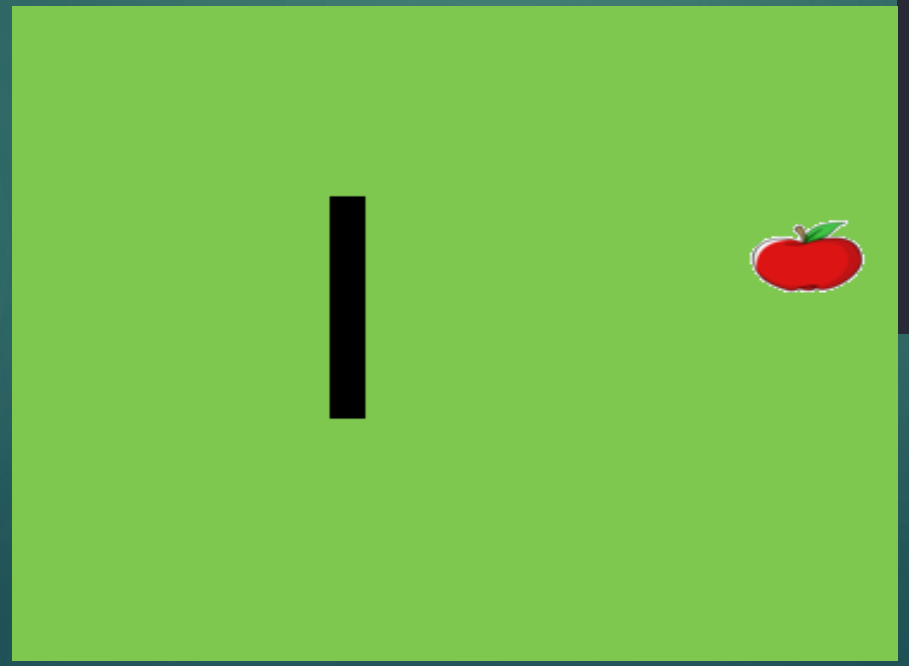
```
8   def snake_and_fruit(snake, fruit, ai_settings):
9       #What happends when tha snake eats the fruit
10
11      if snake.rect.colliderect(fruit.rect):
12          #If there is music I lower it and play eating sound
13          mixer.music.set_volume(0.1)
14          mixer.music.pause()
15          apple_sound = mixer.Sound("music/apple_sound4.wav")
16          apple_sound.play()
17
18          #New spawn point for the fruit
19          fruit.rect.x =  random.randint(30, ai_settings.screen_width-30)
20          fruit.rect.y =  random.randint(30, ai_settings.screen_height-30)
21
22          for x in snake.snake_list[:-1]:
23              while fruit.rect.x == x[0] or fruit.rect.y == x[1]:
24                  fruit.rect.x =  random.randint(30, ai_settings.screen_width-30)
25                  fruit.rect.y =  random.randint(30, ai_settings.screen_height-30)
26
27
28          fruit.x = float(fruit.rect.x)
29          fruit.y = float(fruit.rect.y)
30
31          snake.length_snake += 2  #How fast the snake grows
32          ai_settings.score += 1   #Increase the score
33          mixer.music.unpause()
34          mixer.music.set_volume(0.5)
35
```

Main програмата

- Първо импотираме другите класове, които вече създадохме
- Инициализираме миксера и пайгейм.
- Създаваме обекти от Settings, Snake, Fruit.
- Създаваме pygame екрана, който ще използваме
- Създаваме обкети за фона и бутона
- Зареждаме музикалния файл и стартираме музиката от 5та секунда

```python
import pygame
import game_functions as gf

from settings import Settings
from the_snake import Snake
from the_fruit import Fruit
from button import Button
from background import Background_pause
from background import Background_game
from pygame import mixer

pygame.mixer.pre_init(44100, -16, 1, 512)
pygame.init()
ai_settings = Settings()
screen = pygame.display.set_mode((ai_settings.screen_width, ai_settings.screen_height))
snake = Snake(ai_settings)
fruit = Fruit(ai_settings, screen)

clock = pygame.time.Clock()


#background_game = Background_game("images/grass_bg.png", [0][0])
background_ps = Background_pause("images/snake_bg2.png", [0][0])
play_button = Button(screen, "Play")
pygame.display.set_caption("Snake Game")
#game_over = True

mixer.music.load("music/snake_song.mp3")
mixer.music.play(loops = -1, start = 5)
mixer.music.set_volume(0.5)

```

- Рестартираме музиката отначало, за да може след като ‚умрем‘, музиката в менюто да не започне, от където е била, както и да нямаме музика по време на играта.
- След това повтаряме безкрайния цикъл, докато не ‚умрем‘
- А след като играта приключи има забавяне от 30 секунди, за звуков ефект и за да има време да разберем, че сме умрели, а не да ни изхвърли в менюто направо.



```
36  while True:
37
38      mixer.music.rewind()
39      gf.menu(ai_settings, screen, background_ps, play_button)
40
41      snake.move_snake(ai_settings)
42      snake.in_game(ai_settings, screen)
43
44      screen.fill(ai_settings.bg_color)
45      #screen.blit(background_game.image, background_game.rect)
46      gf.score(ai_settings, screen)
47      fruit.blitme()
48
49
50      snake.snake_update(ai_settings,screen)
51      snake.draw_snake(screen, ai_settings)
52
53
54      pygame.display.update()
55
56      gf.snake_and_fruit(snake, fruit, ai_settings)
57
58
59      clock.tick(30)
60
61
```

КРАЙ
ПАНАЙОТ КОСТОВ
F98280
SNAKE GAME WITH PYGAME