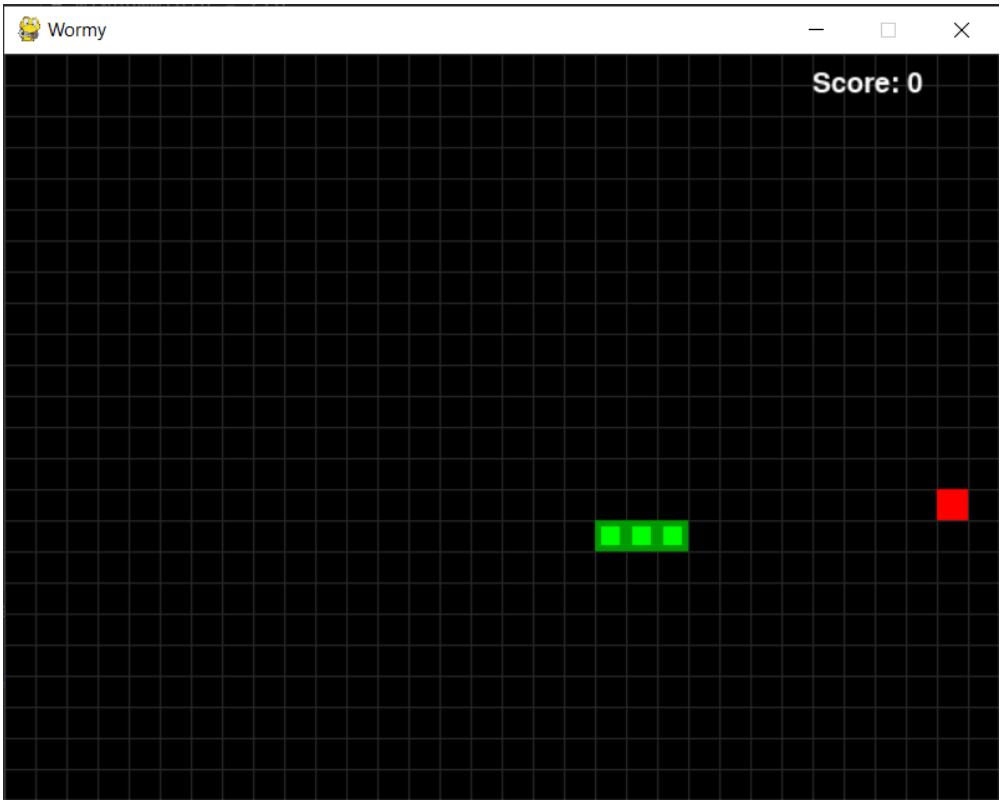
Работа на час wormly

1. Почетен екран

2. *Change the size of the window so that it contains bigger number of cells.*

   Тоа ќе го постигнеме со менување на вредноста на константите WINDOWWIDTH и WINDOWHEIGHT дополнително треба да се внимава нивната вредност да биде делива со големината на полето односно со 20

   WINDOWWIDTH = 800
   WINDOWHEIGHT = 600



3. *Include a yellow apple in the game that will appear only for a limited amount of time. It is aimed to be eaten by the worm and to decrease its length by 1.*

Најпрво ќе дефинираме некои константи кои ќе ја означуваат бојата на јаболкото, колку време ќе биде прикажано јаболкото и времето на повторно појавување.

```
GOLD_COLOR = (255, 215, 0)
GOLD_APPLE_TIMEOUT = 5  # the golden apple will last for 5s
GOLD_APPLE_COOLDOWN = 15  # the golden apple will reappear in 30s
```

Потоа во функцијата rungame на јаболкото му доделуваме случајни локации и го зачувуваме времето на појавување на јаболкото во променлива

```
golden_apple = getRandomLocation();
golden_apple_appearing_time = time.time()
```

Следно во main loop додаваме код кој ќе проверува дали јаболкото треба да исчезне или пак да се појави и ќе се справи со тие настани.

```
if golden_apple is None:
        if time.time() - GOLD_APPLE_COOLDOWN > golden_apple_appearing_time:
                golden_apple = getRandomLocation()
                golden_apple_appearing_time = time.time()
elif time.time() - GOLD_APPLE_TIMEOUT > golden_apple_appearing_time:
        golden_apple = None
```

Потоа во main loop во делот каде проверуваме дали црвот изел црвено јаболко додаваме код кој ќе проверува дали изел жолто јаболко и доколку изел и има должина поголема од три ќе ја намалиме должиината за еден

```
elif golden_apple and wormCoords[HEAD]['x'] == golden_apple['x'] and \
wormCoords[HEAD]['y'] == golden_apple['y']:
        golden_apple = None
        if len(wormCoords) > 3:
                del wormCoords[-1]
        del wormCoords[-1]
```

Потоа во main loop доколку има јаболко повикуваме функција за исцртување на јаболкото

```
if (golden_apple):
        drawApple(golden_apple, GOLD_COLOR)
```

Дополнително во функцијата drawApple правиме промена и додаваме нов аргумент на функцијата кој ќе се однесува на бојата на јаболкото што го цртаме со предефинирана вредност RED.

```
def drawApple(coord, color=RED):
        …
        pygame.draw.rect(DISPLAYSURF, color, appleRect)
        …
```

4. *Firstly decrease the speed of the game. Then every 30 seconds increase the speed of the game.*

Првин ја тргаме константата FPS таа сега ќе биде глобална променлива и додаваме нови константи кои означуваат за колку ќе се забрзува играта и на колку време ќе се забрзува

```
# FPS = 15
FPS_INCREMENT = 3
FPS_INTERVAL = 30


def main():

        global FPS

        ...

        while True:

        # Change for requirement 4:

        FPS = 7
```

```
runGame()
showGameOverScreen()
```

Потоа во run game додаваме променлива која ќе го чува времето кога се забрзала играта

```
speed_up_timestamp = time.time()
```

Следно во main loop додаваме код со кој што проверуваме дали треба да се зголеми брзината на играта а ако треба ја забрзуваме

```
If time.time() - FPS_INTERVAL > speed_up_timestamp:
    FPS += FPS_INCREMENT
    speed_up_timestamp = time.time()
```

5. *Add a blue apple that will appear only for a limited amount of time. It is aimed to be eaten by the worm and to decrease the speed of the game to the previous speed value.*

Слично како и за третото барање најпрво ќе дефинираме некои константи кои ќе ја означуваат бојата на јаболкото колку време ќе биде прикажано јаболкото и времето на повторно појавување.

```
BLUE = (0, 0, 255)
BLUE_APPLE_TIMEOUT = 3  # the blue apple will last for 3s
BLUE_APPLE_COOLDOWN = 25  # the golden apple will reappear in 25s
```

Потоа во функцијата rungame на јаболкото му доделуваме случајни локации и го зачувуваме времето на појавување на јаболкото во променлива.

```
blue_apple = getRandomLocation()
blue_apple_appearing_time = time.time()
```

Следно во main loop додаваме код кој ќе проверува дали јаболкото треба да исчезне или пак да се појави и ќе се справи со тие настани.

```
if blue_apple is None:
    if time.time() - BLUE_APPLE_COOLDOWN > blue_apple_appearing_time:
        blue_apple = getRandomLocation()
        blue_apple_appearing_time = time.time()
```

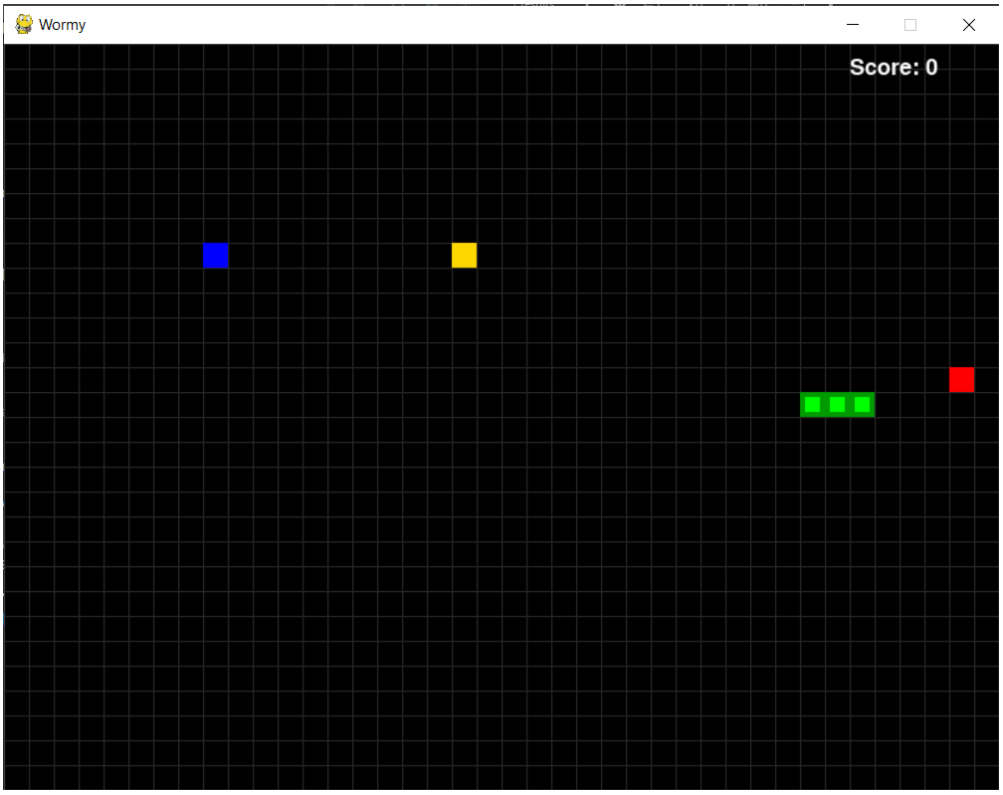
```
elif time.time() - BLUE_APPLE_TIMEOUT > blue_apple_appearing_time:
    blue_apple = None
```

Потоа во main loop во делот каде проверуваме дали црвот изел црвено или жолто јаболко додаваме код кој ќе проверува дали изел сино јаболко и доколку изел и брзината на играта не е преспора ќе ја намалиме брзината на играта.

```
elif blue_apple and wormCoords[HEAD]['x'] == blue_apple['x'] and \
wormCoords[HEAD]['y'] == blue_apple['y']:
    blue_apple = None
    if FPS > 6:
        FPS -= FPS_INCREMENT
    del wormCoords[-1]
```

Потоа во main loop доколку има јаболко повикуваме функција за исцртување на јаболкото

```
if (blue_apple):
    drawApple(blue_apple, BLUE)
```



6. Change the color of the worm each time the speed of the game is changed.

Во runGame додаваме променлива во која ќе се чува бојата на црвот.

```
worm_color = DARKGREEN
```

Потоа секаде каде што го зголемуваме или намалуваме FPS односно ја променуваме брзината на играта додаваме код со кој што ја менуваме бојата на црвот.

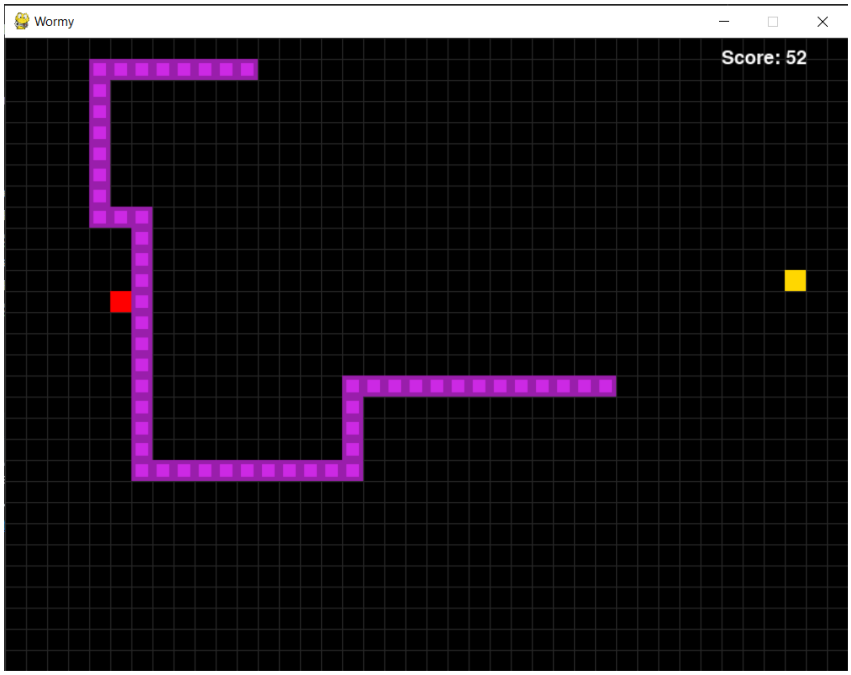
```
worm_color = getRandomColor()
```

За генерирање на нова боја користиме функција која случајно генерира боја

```
def getRandomColor():
    return random.randint(0, 255), random.randint(0, 255), \
    random.randint(0, 255)
```

За да може да го исцртаме црвот со различна боја ја менуваме функцијата за цртање на црвот така што додаваме аргумент за боја кој е предефиниран на GREEN потоа темната нијанса на бојата ја добиваме така што доделуваме 0.75 од секој канал од оригиналната боја и потоа конечно ги исцртуваме квадратите на црвот.

```
def drawWorm(wormCoords, color=GREEN):
    # Change for requirement 6:
    r, g, b = color
    dark_color = (int(r * 0.75), int(g * 0.75), int(b * 0.75))
    for coord in wormCoords:
        x = coord['x'] * CELLSIZE
        y = coord['y'] * CELLSIZE
        wormSegmentRect = pygame.Rect(x, y, CELLSIZE, CELLSIZE)
        pygame.draw.rect(DISPLAYSURF, dark_color, wormSegmentRect)
        wormInnerSegmentRect = pygame.Rect(x + 4, y + 4, CELLSIZE - 8, \
CELLSIZE - 8)
        pygame.draw.rect(DISPLAYSURF, color, wormInnerSegmentRect)
```



7. *Make a change of your choice.*

Во играта ќе додадеме препреки на случајни локации кои што ако играчот ги удри играта ќе заврши.

За таа цел најпрво ќе додадеме константи кои ќе ги означуваат боите на препреките и бројот на препреки

```
OBSTACLE_COLOR = (133, 127, 40)
OBSTACLE_INER_COLOR = (179, 170, 39)
NUM_OBSTACLES = int(CELLHEIGHT * CELLWIDTH * 0.02)
# 2% of the board will be obstacles
```

Потоа во runGame ќе додадеме променлива во која ќе се чуваат препреките

```
obstacles = get_obstacles()
```

Потоа во main loop ќе провериме дали главата на црвот се судрила со некоја од препреките во потврден случај играта ќе заврши

```
for obstacle in obstacles:
    if wormCoords[HEAD]['x'] == obstacle['x'] and wormCoords[HEAD]['y'] == \
obstacle['y']:
        return  # game over
```

Потоа во main loop препреките ги исцртуваме

```
drawObstacles(obstacles)
```

За да ги земеме препреките користиме функција која ги генерира препреките на случајни локации

```
def get_obstacles():
    obstacles = []
    for i in range(NUM_OBSTACLES):
        obstacles.append(getRandomLocation(obstacles))
    return obstacles
```

За да ги исцртаме препреките користиме функција која ги црта препреките на сличен начин како што се цртаат полињата на црвот

```
def drawObstacles(obstacles):
    for obstacle in obstacles:
        x = obstacle['x'] * CELLSIZE
        y = obstacle['y'] * CELLSIZE
        obstacleSegmentRect = pygame.Rect(x, y, CELLSIZE, CELLSIZE)
        pygame.draw.rect(DISPLAYSURF, OBSTACLE_COLOR, \
obstacleSegmentRect)
```

```
                    obstacleInnerSegmentRect = pygame.Rect(x + 4, y + 4, CELLSIZE - 8, \
CELLSIZE - 8)
                    pygame.draw.rect(DISPLAYSURF, OBSTACLE_INER_COLOR, \
obstacleInnerSegmentRect)
```

Дополнително за правилна функционалност на играта потремна е измена во функцијата гоја генерира нови локации со цел зафатена локација да не може да биде генерирана. Па така додаваме аргумент кој има предефинирана вредност празна низа а во функцијата генерираме низа на полиња кои што не се зафатени (нема препреки на нив) и од нив случајно избираме едно поле кое го враќаме како резултат

```
def getRandomLocation(obstacles = []):
    tiles = []
    for i in range (CELLWIDTH):
        for j in range (CELLHEIGHT):
            tiles.append({'x': i,'y': j})
            for obstacle in obstacles:
                if obstacle['x']==i and obstacle['y']==j:
                    del tiles[-1]
                    break
    return random.choice(tiles)
```