Wormy Game
3/22/2020

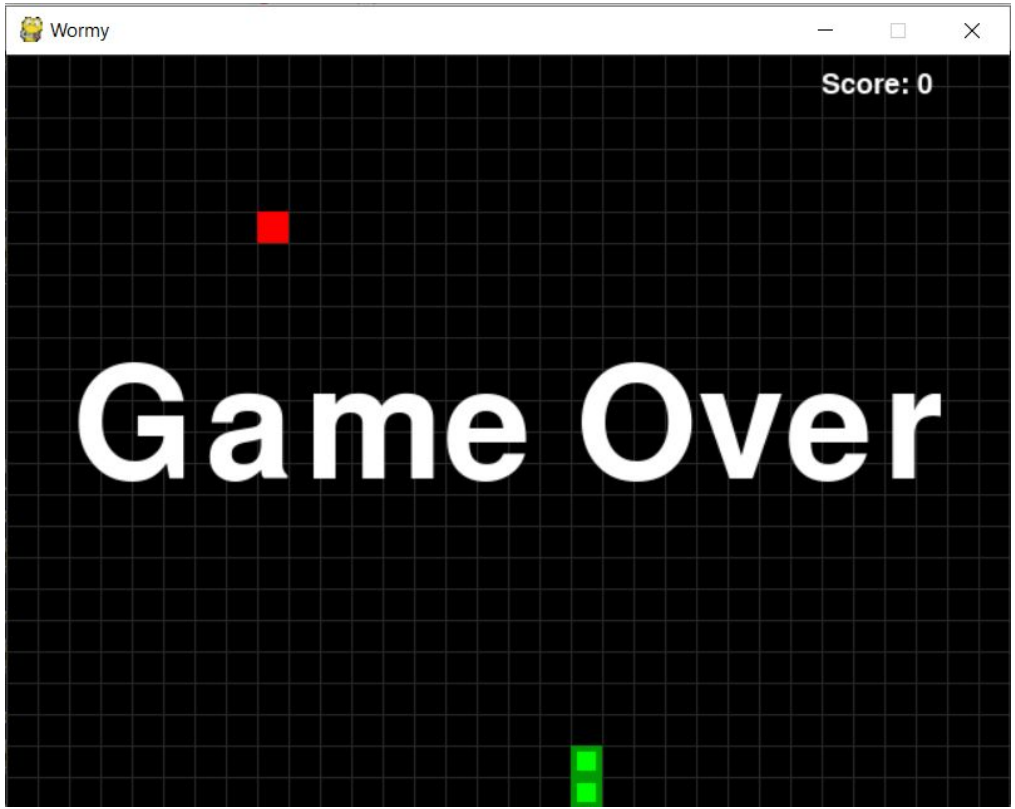
Exercise

```
apple = {
    'color': 'red',
    'Size': 20
    }

banana = {
    'color': 'yellow',
    'size': 10
    }

fruit = [ apple, banana ]

print(fruit)

print(fruit[0])

print(fruit[-1])

print(fruit[0]['color'])

print(fruit[1]['size'])
```

Game rules

1. Eat the apple
2. Don’t hit the wall
3. Don’t hit worm body
4. Get the highest score

Game over



Game setup

```
import random, pygame, sys
from pygame.locals import QUIT, KEYDOWN, KEYUP, K_LEFT, K_RIGHT, K_UP, K_DOWN

FPS = 5
WINDOWWIDTH = 640
WINDOWHEIGHT = 480
CELLSIZE = 20
assert WINDOWWIDTH % CELLSIZE == 0, "Window width must be a multiple of cell size."
assert WINDOWHEIGHT % CELLSIZE == 0, "Window height must be a multiple of cell size."
CELLWIDTH = int(WINDOWWIDTH / CELLSIZE)
CELLHEIGHT = int(WINDOWHEIGHT / CELLSIZE)

#               R    G    B
WHITE     = (255, 255, 255)
BLACK     = (  0,   0,   0)
RED       = (255,   0,   0)
GREEN     = (  0, 255,   0)
DARKGREEN = (  0, 155,   0)
DARKGRAY  = ( 40,  40,  40)
BGCOLOR = BLACK
```

Grid

Window width

Cell size

Window height

Game setup and main function

```
UP = 'up'
DOWN = 'down'
LEFT = 'left'
RIGHT = 'right'

HEAD = 0 # syntactic sugar: index of the worm's head

def main():
    global FPSCLOCK, DISPLAYSURF, BASICFONT

    pygame.init()
    FPSCLOCK = pygame.time.Clock()
    DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))
    BASICFONT = pygame.font.Font('freesansbold.ttf', 18)
    pygame.display.set_caption('Wormy')

    while True:
        runGame()
        showGameOverScreen()
```

Run game function

```
def runGame():
    # Set a random start point.
    worm = Worm(CELLWIDTH, CELLHEIGHT, CELLSIZE)
    # Start the apple in a random place.
    apple = Apple(CELLWIDTH, CELLHEIGHT, CELLSIZE)

    while True: # main game loop
        if worm.hit_edge() or worm.hit_self():
            return

        for event in pygame.event.get(): # event handling loop
            if event.type == QUIT:
                terminate()
            elif event.type == KEYDOWN:
                if (event.key == K_LEFT) and worm.direction != RIGHT:
                    worm.direction = LEFT
                elif (event.key == K_RIGHT) and worm.direction != LEFT:
                    worm.direction = RIGHT
                elif (event.key == K_UP) and worm.direction != DOWN:
                    worm.direction = UP
                elif (event.key == K_DOWN) and worm.direction != UP:
                    worm.direction = DOWN

        worm.update()
```

Run game function (cont)

```
worm.update()

# check if worm has eaten an apply
if worm.Coords[HEAD] == apple.Coord:
    apple.update()
else:
    worm.remove_tail() # remove worm's tail segment

DISPLAYSURF.fill(BGCOLOR)
drawGrid()
worm.draw()
apple.draw()
drawScore(len(worm.Coords) - 3)
pygame.display.update()
FPSCLOCK.tick(FPS)
```

Show game over screen function

```
def showGameOverScreen():
    gameOverFont = pygame.font.Font('freesansbold.ttf', 100)
    gameSurf = gameOverFont.render('Game Over', True, WHITE)
    gameRect = gameSurf.get_rect()
    gameRect.midtop = (int(WINDOWWIDTH/2), int(WINDOWHEIGHT/2)-50)
    DISPLAYSURF.blit(gameSurf, gameRect)

    pygame.display.update()
    pygame.time.wait(500)

    while True:
        keypressed = False
        for event in pygame.event.get(): # event handling loop
            if event.type == QUIT:
                terminate()
            elif event.type == KEYUP:
                keypressed = True
                break
        if keypressed:
            return
```

Terminate, draw score functions

```
def terminate():
    pygame.quit()
    sys.exit()

def drawScore(score):
    scoreSurf = BASICFONT.render('Score: %s' % (score), True, WHITE)
    scoreRect = scoreSurf.get_rect()
    scoreRect.topleft = (WINDOWWIDTH - 120, 10)
    DISPLAYSURF.blit(scoreSurf, scoreRect)
```

Draw Grid

```
def drawGrid():
    for x in range(0, WINDOWWIDTH, CELLSIZE): # draw vertical lines
        pygame.draw.line(DISPLAYSURF, DARKGRAY, (x, 0), (x, WINDOWHEIGHT))
    for y in range(0, WINDOWHEIGHT, CELLSIZE): # draw horizontal lines
        pygame.draw.line(DISPLAYSURF, DARKGRAY, (0, y), (WINDOWWIDTH, y))
```

Apply class

```
class Apple(object):
    def __init__(self, cell_width, cell_height, cell_size):
        self.cell_width = cell_width
        self.cell_height = cell_height
        self.cell_size = cell_size
        self.update()

    def draw(self):
        x = self.Coord['x'] * self.cell_size
        y = self.Coord['y'] * self.cell_size
        appleRect = pygame.Rect(x, y, self.cell_size, self.cell_size)
        pygame.draw.rect(DISPLAYSURF, RED, appleRect)

    def update(self):
        self.Coord = {'x': random.randint(0, self.cell_width - 1), 'y': random.randint(0, self.cell_height - 1)}
```

Worm class

```
class Worm(object):
    def __init__(self, cell_width, cell_height, cell_size):
        self.cell_width = cell_width
        self.cell_height = cell_height
        self.cell_size = cell_size
        self.direction = RIGHT
        # Set a random start point.
        margin = 5
        startx = random.randint(margin, cell_width - margin)
        starty = random.randint(margin, cell_height - margin)
        self.Coords = [{'x': startx,     'y': starty},
                       {'x': startx - 1, 'y': starty},
                       {'x': startx - 2, 'y': starty}]

    def draw(self):
        for coord in self.Coords:
            x = coord['x'] * self.cell_size
            y = coord['y'] * self.cell_size
            wormSegmentRect = pygame.Rect(x, y, self.cell_size, self.cell_size)
            pygame.draw.rect(DISPLAYSURF, DARKGREEN, wormSegmentRect)
            wormInnerSegmentRect = pygame.Rect(x + 4, y + 4, self.cell_size - 8, self.cell_size - 8)
            pygame.draw.rect(DISPLAYSURF, GREEN, wormInnerSegmentRect)
```

0	1	2	3	4

Worm class (cont)

new	0	1	2	3	4

```
def update(self):
    if self.direction == UP:
        newHead = {'x': self.Coords[HEAD]['x'], 'y': self.Coords[HEAD]['y'] - 1}
    elif self.direction == DOWN:
        newHead = {'x': self.Coords[HEAD]['x'], 'y': self.Coords[HEAD]['y'] + 1}
    elif self.direction == LEFT:
        newHead = {'x': self.Coords[HEAD]['x'] - 1, 'y': self.Coords[HEAD]['y']}
    elif self.direction == RIGHT:
        newHead = {'x': self.Coords[HEAD]['x'] + 1, 'y': self.Coords[HEAD]['y']}
    self.Coords.insert(0, newHead)

def remove_tail(self):
    del self.Coords[-1]

def hit_edge(self):
    if self.Coords[HEAD]['x'] == -1 or self.Coords[HEAD]['x'] == self.cell_width \
        or self.Coords[HEAD]['y'] == -1 or self.Coords[HEAD]['y'] == self.cell_height:
        return True
    else:
        return False

def hit_self(self):
    if self.Coords[HEAD] in self.Coords[1:]:
        return True
    else:
        return False
```

Start the game

```
if __name__ == '__main__':
    main()
```