import pygame, sys, random

from pygame.math import Vector2

class SNAKE:

    def \_\_init\_\_(self):

        self.body = [Vector2(5,10),Vector2(4,10),Vector2(3,10)]

        self.direction = Vector2(0,0)

        self.new\_block = False

        self.head\_up = pygame.image.load('Graphics/head\_up.png').convert\_alpha()

        self.head\_down = pygame.image.load('Graphics/head\_down.png').convert\_alpha()

        self.head\_right = pygame.image.load('Graphics/head\_right.png').convert\_alpha()

        self.head\_left = pygame.image.load('Graphics/head\_left.png').convert\_alpha()

        self.tail\_up = pygame.image.load('Graphics/tail\_up.png').convert\_alpha()

        self.tail\_down = pygame.image.load('Graphics/tail\_down.png').convert\_alpha()

        self.tail\_right = pygame.image.load('Graphics/tail\_right.png').convert\_alpha()

        self.tail\_left = pygame.image.load('Graphics/tail\_left.png').convert\_alpha()

        self.body\_vertical = pygame.image.load('Graphics/body\_vertical.png').convert\_alpha()

        self.body\_horizontal = pygame.image.load('Graphics/body\_horizontal.png').convert\_alpha()

        self.body\_tr = pygame.image.load('Graphics/body\_tr.png').convert\_alpha()

        self.body\_tl = pygame.image.load('Graphics/body\_tl.png').convert\_alpha()

        self.body\_br = pygame.image.load('Graphics/body\_br.png').convert\_alpha()

        self.body\_bl = pygame.image.load('Graphics/body\_bl.png').convert\_alpha()

        self.crunch\_sound = pygame.mixer.Sound('Sound/crunch.wav')

    def draw\_snake(self):

        self.update\_head\_graphics()

        self.update\_tail\_graphics()

        for index,block in enumerate(self.body):

            x\_pos = int(block.x \* cell\_size)

            y\_pos = int(block.y \* cell\_size)

            block\_rect = pygame.Rect(x\_pos,y\_pos,cell\_size,cell\_size)

            if index == 0:

                screen.blit(self.head\_right,block\_rect)

            elif index == len(self.body) - 1:

                screen.blit(self.tail,block\_rect)

            else:

                previous\_block = self.body[index + 1] - block

                next\_block = self.body[index - 1] - block

                if previous\_block.x == next\_block.x:

                    screen.blit(self.body\_vertical,block\_rect)

                elif previous\_block.y == next\_block.y:

                    screen.blit(self.body\_horizontal,block\_rect)

                else:

                    if previous\_block.x == -1 and next\_block.y == -1 or previous\_block.y == -1 and next\_block.x == -1:

                        screen.blit(self.body\_tl,block\_rect)

                    if previous\_block.x == -1 and next\_block.y == 1 or previous\_block.y == 1 and next\_block.x == -1:

                        screen.blit(self.body\_bl,block\_rect)

                    if previous\_block.x == 1 and next\_block.y == -1 or previous\_block.y == -1 and next\_block.x == 1:

                        screen.blit(self.body\_tr,block\_rect)

                    if previous\_block.x == 1 and next\_block.y == 1 or previous\_block.y == 1 and next\_block.x == 1:

                        screen.blit(self.body\_bl,block\_rect)

    def update\_head\_graphics(self):

        head\_position = self.body[1] - self.body[0]

        if head\_position == Vector2(1,0): self.head = self.head\_left

        elif head\_position == Vector2(-1,0): self.head = self.head\_right

        elif head\_position == Vector2(0,1): self.head = self.head\_up

        elif head\_position == Vector2(0,-1): self.head = self.head\_down

    def update\_tail\_graphics(self):

        tail\_position = self.body[-2] - self.body[-1]

        if tail\_position == Vector2(1,0): self.tail = self.tail\_left

        elif tail\_position == Vector2(-1,0): self.tail = self.tail\_right

        elif tail\_position == Vector2(0,1): self.tail = self.tail\_up

        elif tail\_position == Vector2(0,-1): self.tail = self.tail\_down

    def move\_snake(self):

        if self.new\_block == True:

            body\_copy = self.body[:]

            body\_copy.insert(0,body\_copy[0] + self.direction)

            self.body = body\_copy[:]

            self.new\_block = False

        else:

            body\_copy = self.body[:-1]

            body\_copy.insert(0,body\_copy[0] + self.direction)

            self.body = body\_copy[:]

    def add\_block(self):

        self.new\_block = True

    def play\_crunch\_sound(self):

        self.crunch\_sound.play()

    def reset(self):

        self.body = [Vector2(5,10),Vector2(4,10),Vector2(3,10)]

        self.direction = Vector2(0,0)

class FRUIT:

    def \_\_init\_\_(self):

        self.randomize()

    def draw\_fruit(self):

        fruit\_rect = pygame.Rect(int(self.pos.x \* cell\_size),int(self.pos.y \* cell\_size),cell\_size,cell\_size)

        screen.blit(apple,fruit\_rect)

        pygame.draw.rect(screen,(126,166,114),fruit\_rect)

    def randomize(self):

        self.x = random.randint(0,cell\_number - 1)

        self.y = random.randint(0,cell\_number - 1)

        self.pos = Vector2(self.x,self.y)

class MAIN:

    def \_\_init\_\_(self):

        self.snake = SNAKE()

        self.fruit = FRUIT()

    def update(self):

        self.snake.move\_snake()

        self.check\_collision()

        self.check\_fail()

    def draw\_elements(self):

        self.draw\_grass()

        self.fruit.draw\_fruit()

        self.snake.draw\_snake()

        self.draw\_score()

    def check\_collision(self):

        if self.fruit.pos == self.snake.body[0]:

            self.fruit.randomize()

            self.snake.add\_block()

            self.snake.play\_crunch\_sound()

        for block in self.snake.body[1:]:

            if block == self.fruit.pos:

                self.fruit.randomize

    def check\_fail(self):

        if not 0 <= self.snake.body[0].x < cell\_number or not 0 <= self.snake.body[1].y < cell\_number:

            self.game\_over()

        for block in self.snake.body[1:]:

            if block == self.snake.body[0]:

                self.game\_over()

    def game\_over(self):

        self.snake.reset()

    def draw\_grass(self):

        grass\_color = (3, 252, 98)

        for row in range(cell\_number):

            if row % 2 == 0:

                for col in range(cell\_number):

                    if col % 2 == 0:

                        grass\_rect = pygame.Rect(col \* cell\_size,row \* cell\_size,cell\_size,cell\_size)

                        pygame.draw.rect(screen,grass\_color,grass\_rect)

            else:

                for col in range(cell\_number):

                    if col % 2 != 0:

                        grass\_rect = pygame.Rect(col \* cell\_size,row \* cell\_size,cell\_size,cell\_size)

                        pygame.draw.rect(screen,grass\_color,grass\_rect)

    def draw\_score(self):

        score\_text = str(len(self.snake.body) - 3)

        score\_surface = game\_font.render(score\_text,False,(56,74,12))

        score\_x = int(cell\_size \* cell\_number - 100)

        score\_y = int(cell\_size \* cell\_number - 50)

        score\_rect = score\_surface.get\_rect(center = (score\_x,score\_y))

        apple\_rect = apple.get\_rect(midright = (score\_rect.left,score\_rect.centery))

        pygame.draw.rect(screen,(167,209,61),bg\_rect)

        screen.blit(score\_surface,score\_rect)

        screen.blit(apple,apple\_rect)

        bg\_rect = pygame.Rect(apple\_rect.left,apple\_rect.top,apple\_rect.width + score\_rect.width +6,apple\_rect.height)

        pygame.draw.rect(screen,(56,74,12),bg\_rect,2)

pygame.mixer.pre\_init(22050, -16, 2, 2048)

pygame.init()

cell\_size = 30

cell\_number = 20

screen = pygame.display.set\_mode((cell\_number \* cell\_size,cell\_number \* cell\_size))

clock = pygame.time.Clock()

apple = pygame.image.load('Graphics/apple.jpg').convert\_alpha()

game\_font = pygame.font.Font('Font/jap.ttf',25)

SCREEN\_UPDATE = pygame.USEREVENT

pygame.time.set\_timer(SCREEN\_UPDATE,150)

main\_game = MAIN()

while True:

    for event in pygame.event.get():

        if event.type == pygame.QUIT:

            pygame.quit()

            sys.exit()

        if event.type == SCREEN\_UPDATE:

            main\_game.update()

        if event.type == pygame.KEYDOWN:

            if event.key == pygame.K\_UP:

                if main\_game.snake.direction.y != 1:

                    main\_game.snake.direction = Vector2(0,-1)

            if event.key == pygame.K\_DOWN:

                if main\_game.snake.direction.y != 1:

                    main\_game.snake.direction = Vector2(0,1)

            if event.key == pygame.K\_LEFT:

                if main\_game.snake.direction.x != 1:

                    main\_game.snake.direction = Vector2(-1,0)

            if event.key == pygame.K\_RIGHT:

                if main\_game.snake.direction.x != 1:

                    main\_game.snake.direction = Vector2(1,0)

        screen.fill(pygame.Color(89, 215, 70))

        main\_game.draw\_elements()

        pygame.display.update()

        clock.tick(60)

[**https://github.com/BossUth/CO452-CW2-Game.git**](https://github.com/BossUth/CO452-CW2-Game.git)