## “Jipto game”

Автор: Калоев Константин

Руководитель проекта: Макаренко М.Д

**Общее назначение разработки (что и для чего делаем):**

Игра jipto – проект по предмету: «Прикладное программное обеспечение»

**Техническое задание (какие должны быть функции):**

В игре 7 фигур:

5 - зеленых "убегающих", и 2 - красных "преследователей"

Игра состоит из 2 раундов.

Цель игры:

Пять "убегающих" пытаются перейти на противоположную сторону прямоугольного игрового поля

до поимки их "преследователем". Тот находится в начале игры в середине противоположной к "убегающим"

стороне и стремится их поймать как можно дальше от своей стороны.

Ход фигуры:

Шаг индивидуальной фигуры осуществляется нажатием стрелок.

У каждой фигуры зеленых в запасе один шаг.

У каждой фигуры красных в запасе три шага.

Игроки делают ходы своими фигурами поочередно, причем первыми ходят "убегающие".

Поимка:

"Убегающий", которого коснулся "преследователь", считается пойманным и убирается с игрового поля.

Оценка результата игры:

Для оценки результата используются линии начерченные на игровом поле.

Каждая линия имеет по одному очку, "убегающий" достигший одной из линий получает по одному баллу.

В конце партии суммируются очки набранные "убегающими".

После одной партии игроки меняются фигурами, начинается вторая партия игры.

После игры очки сравниваются, у кого больше очков тот является победителем.

**Руководство (как это сделано):**

Вначале скрипта импортируем все что нам понадобится:

1.Pygame — набор модулей языка программирования Python, предназначенный для написания компьютерных игр и мультимедиа-приложений.

2.Модуль time открывает разработчику Python доступ к нескольким связанным со временем функциям:

import pygame

import time

Размеры пикселей для квадратов сетки:

TileWidth = 20

TileHeight = 20

TileMargin = 2

Cколько плиток в любом направлении сетки:

MapSizeRow = 25

MapSizeColumn = 45

Переменная конца раунда / игры:

end\_of\_the\_game = 0

Эта функция позволит инициализировать все игровые модули:

pygame.init()

Переменные для подсчета баллов Героев:

number\_hero\_points\_0 = 0

number\_hero\_points\_1 = 0

number\_hero\_points\_2 = 0

number\_hero\_points\_3 = 0

number\_hero\_points\_4 = 0

Используется для сумма баллов Героев в конце раунда:

first\_round\_points = 0

second\_round\_points = 0

Используется в конце игры для вывода победителя:

winner = ""

Задаем цвет:

BLACK = (0, 0, 0)

WHITE = (255, 255, 255)

GREEN = (0, 255, 0)

RED = (255, 0, 0)

BLUE\_GREEN = (0, 150, 150)

BlUE\_WHITE = (215, 215, 255)

COLOR\_FIRST\_LINE = (100, 50, 100)

COLOR\_SECOND\_LINE = (50, 100, 100)

COLOR\_THIRD\_LINE = (100, 100, 50)

Для отслеживания, открыто ли окно:

Done = False

Создаем объект, который поможет отслеживать время:

clock = pygame.time.Clock()

Создаем окно:

Screen = pygame.display.set\_mode([(TileWidth+TileMargin)\*MapSizeColumn,

(TileWidth+TileMargin+5)\*MapSizeRow])

Класс для неподвижных вещей, которые находятся на сетке:

class MapTile(object):

def \_\_init\_\_(self, Name, Row, Column):

self.Name = Name

self.Column = Column

self.Row = Row

Класс персонажей, которые могут передвигаться на сетке:

class Character(object):

def \_\_init\_\_(self, Name, HP, Row, Column, Type):

self.Name = Name

self.HP = HP

self.Row = Row

self.Column = Column

self.Type = Type

self.Points = 0

self.First\_Line = False

self.Second\_Line = False

self.Third\_Line = False

Функция для добавления баллов Героям:

def AddPoint(self):

Функция для проверки достижения Героя линий:

def CheckLine(self, LineNumber):

Эта функция позволяет персонажу двигаться в определенном направлении:

def Move(self, Direction):

Эта функция проверяет, есть ли что-нибудь в том направлении, куда персонаж движется. Если есть, то удаляет преграду.

Используется в функции перемещения:

def CollisionCheck(self, Direction):

Функция для удаления персонажа.

Используется в функции «CollisionCheck»:

def DeleteEntity(self, array, del\_object):

Клас сетки:

class Map(object):

global MapSizeRow #Размеры сетки:

global MapSizeColumn

#Массивы персонажей и массив сетки:

array\_of\_heros = []

array\_of\_villains = []

Grid = []

Создаем сетку:

for Row in range(MapSizeRow):

Grid.append([])

for Column in range(MapSizeColumn):

Grid[Row].append([])

for i in range(3):

Grid[Row][Column].append([])

Создаем границы сетки:

for Row in range(MapSizeRow):

for Column in range(MapSizeColumn):

TempTile = MapTile("Rock", Row, Column)

if Row == 0:

Grid[int(Row)][int(Column)][2].append(TempTile)

if Row == MapSizeRow-1:

Grid[int(Row)][int(Column)][2].append(TempTile)

if Column == 0:

Grid[int(Row)][int(Column)][2].append(TempTile)

if Column == MapSizeColumn-1:

Grid[int(Row)][int(Column)][2].append(TempTile)

Создаем Героев и Врагов:

for i in range(5):

array\_of\_heros.append(Character("Hero" + str(i), 1, (i\*4)+4, 35, 0))

array\_of\_villains.append(Character("Villain0", 1, 4, 5, 1))

array\_of\_villains.append(Character("Villain1", 1, 20, 5, 1))

Заносим персонажей в массив сетки:

for Hero in array\_of\_heros:

Grid[int(Hero.Row)][int(Hero.Column)][0].append(Hero)

for Villain in array\_of\_villains:

Grid[int(Villain.Row)][int(Villain.Column)][1].append(Villain)

Эта функция проходит через всю сетку и проверяет, расходятся ли внутренние координаты какого-либо объекта с его текущим положением на сетке.

Если это происходит, то удаляет объекты и помещает их на сетке в соответствии с ее внутренними координатами + проверка достижения Героем линий и добавление баллов:

def update(self):

Создаем сетку:

Map = Map()

Создаем переменные, которые будут нужны для смены хода персонажей:

iter\_heros = 0

iter\_villains = 0

plus\_on\_iter\_villains = 0

Переменная для проверки нахождения координат указателя мышки в пределах: кнопки правил:

rule\_spawn = False

Создаем главный цикл с «поимкой событий»:

while not Done:

for event in pygame.event.get():

if event.type == pygame.QUIT:

Done = True

Если координаты указателя мышки в пределах кнопки — «rule\_spawn = True»

elif event.type == pygame.MOUSEBUTTONDOWN:

Pos = pygame.mouse.get\_pos()

if 400 + 100 > Pos[0] > 400 and 600 + 100 > Pos[1] > 600:

if rule\_spawn == False:

rule\_spawn = True

else:

rule\_spawn = False

Двигаем персонажа, если событие — нажатие на одну из стрелок:

elif not iter\_heros < len(Map.array\_of\_heros) and not iter\_villains < len(Map.array\_of\_villains) + 4:

iter\_heros = 0

iter\_villains = 0

plus\_on\_iter\_villains = 0

elif iter\_heros < len(Map.array\_of\_heros):

if event.type == pygame.KEYDOWN:

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

Map.array\_of\_heros[iter\_heros].Move("LEFT")

iter\_heros += 1

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

Map.array\_of\_heros[iter\_heros].Move("RIGHT")

iter\_heros += 1

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

Map.array\_of\_heros[iter\_heros].Move("UP")

iter\_heros += 1

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

Map.array\_of\_heros[iter\_heros].Move("DOWN")

iter\_heros += 1

elif iter\_villains < len(Map.array\_of\_villains) + 4:

if iter\_villains > 1 and iter\_villains < 3:

plus\_on\_iter\_villains = -2

elif iter\_villains > 3:

plus\_on\_iter\_villains = -4

if event.type == pygame.KEYDOWN:

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

Map.array\_of\_villains[iter\_villains + plus\_on\_iter\_villains].Move("LEFT")

iter\_villains += 1

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

Map.array\_of\_villains[iter\_villains + plus\_on\_iter\_villains].Move("RIGHT")

iter\_villains += 1

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

Map.array\_of\_villains[iter\_villains + plus\_on\_iter\_villains].Move("UP")

iter\_villains += 1

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

Map.array\_of\_villains[iter\_villains + plus\_on\_iter\_villains].Move("DOWN")

iter\_villains += 1

Отрисовка интерфейса:

font = pygame.font.Font(None, 32)

text = font.render("1:" + str(number\_hero\_points\_0) + ' 2:' + str(number\_hero\_points\_1) + ' 3:' + str(number\_hero\_points\_2) + ' 4:' + str(number\_hero\_points\_3) + ' 5:' + str(number\_hero\_points\_4), 1, WHITE)

place = text.get\_rect(center=(200,590))

Screen.blit(text,place)

Отрисовка сетки с границами, линиями, персонажами :

for Row in range(MapSizeRow): #Drawing grid

for Column in range(MapSizeColumn):

Color = WHITE

if Column == 25: #Первая лин

Color = COLOR\_FIRST\_LINE

if Column == 20: #Вторая лин

Color = COLOR\_SECOND\_LINE

if Column == 15: #Третья лин

Color = COLOR\_THIRD\_LINE

for i in range(0, len(Map.Grid[Row][Column])):

for j in range(0, len(Map.Grid[Row][Column][i])):

if Map.Grid[Row][Column][i][j].Name == "Rock":

Color = BLUE\_GREEN

for Hero in Map.array\_of\_heros:

if Map.Grid[Row][Column][i][j].Name == Hero.Name:

Color = GREEN

if Map.Grid[Row][Column][i][j].Name == "Villain0" or Map.Grid[Row][Column][i][j].Name == "Villain1":

Color = RED

for i in range(0, len(Map.Grid[Row][Column])):

for j in range(0, len(Map.Grid[Row][Column][i])):

if iter\_heros < len(Map.array\_of\_heros):

if Map.Grid[Row][Column][i][j].Name == Map.array\_of\_heros[iter\_heros].Name:

Color = BLACK

elif iter\_villains < len(Map.array\_of\_villains) + 4:

if iter\_villains > 1 and iter\_villains < 3:

plus\_on\_iter\_villains = -2

elif iter\_villains > 3:

plus\_on\_iter\_villains = -4

if Map.Grid[Row][Column][i][j].Name == Map.array\_of\_villains[iter\_villains + plus\_on\_iter\_villains].Name:

Color = BLACK

pygame.draw.rect(Screen, Color, [(TileMargin + TileHeight) \* Column + TileMargin,

(TileMargin + TileWidth) \* Row + TileMargin,

TileWidth,

TileHeight])

Функция для отрисовки текста с переносом строк:

def blit\_text(surface, text, pos, font, color=pygame.Color('black')):

Отрисовка кнопки правил + ее выделение:

mouse = pygame.mouse.get\_pos()

if 400 + 100 > mouse[0] > 400 and 600 + 100 > mouse[1] > 600:

pygame.draw.rect(Screen, BLUE\_GREEN, (400, 600, 100, 100))

else:

pygame.draw.rect(Screen, RED, (400, 600, 100, 100))

text = font.render("Правила", 1, WHITE)

place = text.get\_rect(center=(400+(100/2), 600+(100/2)))

Screen.blit(text,place)

Вывод правил на на экран:

if rule\_spawn == True:

pygame.draw.rect(Screen, WHITE, (0, 0, 1000, 500))

text ="""Правила"""

font\_rule = pygame.font.Font(None, 22)

blit\_text(Screen, text, (0,0), font\_rule)

Подсчет баллов в конце раунда и респавн персонажей:

if not Map.array\_of\_heros: #Конец раунда

if end\_of\_the\_game == 0:

first\_round\_points = (number\_hero\_points\_0 +

number\_hero\_points\_1 +

number\_hero\_points\_2 +

number\_hero\_points\_3 +

number\_hero\_points\_4)

else:

second\_round\_points = (number\_hero\_points\_0 +

number\_hero\_points\_1 +

number\_hero\_points\_2 +

number\_hero\_points\_3 +

number\_hero\_points\_4)

Конец второго раунда — выйти из игры и отобразить счет на интерфейсе:

if end\_of\_the\_game == 2: #два раунда - выход из игры

########################### -GUI

text = font.render(str(first\_round\_points) + "- Первый раунд " + str(second\_round\_points) + "-Второй раунд ", 1, WHITE)

if first\_round\_points > second\_round\_points:

text\_result = font.render("Победитель: " + str("1 игрок") , 1, WHITE)

elif first\_round\_points < second\_round\_points:

text\_result = font.render("Победитель: " + str("2 игрок") , 1, WHITE)

else:

text\_result = font.render("Победитель: " + str("Ничья") , 1, WHITE)

place = text.get\_rect(center=(600,590))

place\_result = text\_result.get\_rect(center=(600, 610))

Screen.blit(text, place)

Screen.blit(text\_result, place\_result)

###########################

Done = True

При вызове Clock.tick(60) один раз в секунду, программа никогда не будет работать со скоростью более 60 кадров в секунду:

clock.tick(60)

Обновляем всю поверхность на дисплее:

pygame.display.flip()

Обновляем сетку:

Map.update()

Количество секунд, в течение которых программа Python должна будет приостановиться, а затем выйти:

time.sleep(5)

pygame.quit()

**Руководство пользователя (как там чего нажимать):**

Шаг индивидуальной фигуры осуществляется нажатием стрелок.

У каждой фигуры зеленых в запасе один шаг.

У каждой фигуры красных в запасе три шага.

Игроки делают ходы своими фигурами поочередно, причем первыми ходят "убегающие".

**Литература/статьи в инете, которые вам помогли:**

<https://www.pygame.org/docs/> - официальная документация по pygames