Министерство образования и науки Республики Казахстан

Северо-Казахстанский университет им. М. Козыбаева

СЕМЕСТРОВЫЙ ПРОЕКТ 1

На тему:

«Объектно-ориентированное программирование и модуль turtle.»

Группа №4

|  |  |
| --- | --- |
| Выполнили студенты  группы АПО-19 | Шамсудинов В.В.  Сахаров А.В.  Буряк Р.С.  Сыздыков Р.С. |
| Научный руководитель  доцент, PhD | Астапенко Н.В. |

Петропавловск, 2022

ОГЛАВЛЕНИЕ

[1. Постановка задачи 3](#_Toc95422315)

[2. Распределение работ 3](#_Toc95422316)

[3. Архитектура программы 4](#_Toc95422317)

[4. Описание используемых алгоритмов 5](#_Toc95422318)

[5. Тестирование и анализ результатов 5](#_Toc95422319)

[Заключение 8](#_Toc95422320)

[Используемая литература 9](#_Toc95422321)

[ПРИЛОЖЕНИЕ А 10](#_Toc95422322)

## 1. Постановка задачи

В данной групповой работе ставятся следующие задачи:

1) Разработать два класса фигур, расположенных в отдельных модулях. Минимальные требования к каждому классу: наличие конструктора, выполняющего отрисовку фигуры (например, изображение смайлика или мухи); метод, выполняющий отрисовку фигуры в другом положении (например, появление улыбки у смайлика или взмах крыльев мухи).

2) Разработать внешний вид основного окна, содержащего название проекта, фамилии исполнителей, фон.

3) Разработать алгоритм создания одного или нескольких объектов первого класса, движущихся по заранее заданной траектории (например, одна или несколько мух, передвигающихся по синусоиде, меняющие взмах крыльев через определенный интервал).

4) Разработать алгоритм создания объекта другого класса, передвижения его по нажатию клавиш, вызов метода при определенных условиях (например, передвижение смайлика при нажатии клавиши и появление улыбки через определенное число перемещений).

5) Создать исполняемый файл.

## 2. Распределение работ

На первом этапе работы была поставлена задача выбора метода реализации требований проекта. После формирования общего видения и концепции продукта, мы приступили к распределению ролей и обязанностей.

Работа над проектом в составе группы должна иметь и содержать распределение работ, то есть распределение функциональных обязанностей между участниками группы.

Участниками группы являются:

* Алексей Сахаров – тимлид;
* Роман Буряк;
* Раимбек Сыздыков;
* Владислав Шамсутдинов;

Тимлид как роль в проекте занимается непосредственным распределением задач между участниками проекта. Распределение работ представлено в соответствии с таблицей 1.

|  |  |
| --- | --- |
| Участник | Задача |
| Алексей Сахаров | Создание архитектуры программы  Реализация участков кода |
| Роман Буряк | Создание архитектуры программы  Реализация участков кода |
| Раимбек Сыздыков | Знакомство с Git |
| Владислав Шамсутдинов | Знакомство с модулем Turtle |

Таблица 1. Главные задачи и их исполнитель:

## 3. Архитектура программы

Архитектура программы отображает как программа была реализована и из чего она состоит. Архитектура программы представлена в соответствии с рисунком 1.

|  |
| --- |
|  |

Рисунок 1. Архитектура программы.

Использование выбранной архитектуры проекта гарантирует работоспособность готового продукта.

## 4. Описание используемых алгоритмов

Программа будет состоять из следующих функциональных частей:

* Основное тело программы, которое будет отвечать за работоспособность всей программы
* Класс, описывающий тело змейки
* Класс, описывающий наличие краёв карты
* Класс, описывающий смайлик
* Класс, описывающий объект черепашки на поле

А так же содержит методы:

* Метод, описывающий движение черепашки по полю
* Метод, описывающий появление грусти у смайлика
* Метод, описывающий поедание змеёй еды
* Метод, описывающий смерть змейки

Программа использует следующие алгоритмы:

* Использование классов Python
* Программа полностью зависит от методов отдельных классов
* Использование класса smile
* Класс в отдельном модуле, созданный в ходе работы над групповым проектом. Он появляется в самом начале игры и меняет свою структуру в случае проигрыша. В начале игры смайлик находится в верхней части экрана и если змея проигрывает, то запускается отдельный алгоритм, занимающийся рисованием грусти у смайлика
* Использование класса border
* Класс в отдельном модуле, созданный в ходе работы над групповым проектом
* Класс Turtle
* Класс змейки
* Класс ball

## 5. Тестирование и анализ результатов

При тестировании программы был произведен подробный анализ каждой подпрограммы.

При запуске программы появилось окно программы с самой игрой.

Окно рекламы представлено в соответствии с рисунком 2.

|  |
| --- |
| https://sun2.dataix-kz-akkol.userapi.com/impg/SPi_ZZFZHgsUYwYUEHrEVedq9jepNsOb0vvg_Q/e_LgNxcQUKM.jpg?size=649x727&quality=96&sign=ffe4cfa2024bbb732edde46e6eb0c4a0&type=album |

Рисунок 2. Главное окно змейки.

Главное окно змейки появляется сразу после запуска программы.

После смерти появляется окно, показанное на рисунке 3. Это окно означает, что игра была проиграна игроком.

|  |
| --- |
| https://sun2.dataix-kz-akkol.userapi.com/impg/FP4gwb9DlZ0C5EYGwjAKwLSKtb79TZ838OlLGg/R1XTf3H01eg.jpg?size=648x729&quality=96&sign=7b9711e1c6608d4c79e12da6125e7c21&type=album |

Рисунок 3. Смерть змейки.

Недоработкой проекта является прохождение змейки сквозь саму себя, как показано на рисунке 4. В дальнейшем эту недоработку планируется полностью исправить, тогда игра будет считаться проигранной если змея съест сама себя.

|  |
| --- |
| https://sun9-59.userapi.com/impg/hTSnS4Ktv0hJp46N-c4Myto3BDA3pz07y-_9zw/goH1Y6nbhik.jpg?size=645x728&quality=96&sign=53312810945be6a03efb3bd25ebf3d7e&type=album |

Рисунок 4. Прохождение змейки сквозь себя.

## Заключение

В ходе выполнения курсовой работы были изучены основы ООП на языке Python, а так же изучен порядок работы с GitHub. Написание программы способствовало закреплению теоретического материала членов группы на практике.

Игра "Змейка" является логически завершенной игрой. Также возможны изменения и добавления некоторых моментов в геймплей, такие как обработка поедания змейкой самой себя. Прохождение змейки через край карты так, чтобы она появилась в этом же месте на противоположной стороне.

## Используемая литература

1. GitHub для самых маленьких #1 | Основы. Режим доступа: URL: https://www.youtube.com/watch?v=Rke\_Z1-nvUM (01.02.2022);
2. Github для самых маленьких #2 | Практика – GitHub Desktop | Приложение Гитхаб для компьютера. Режим доступа: URL: https://www.youtube.com/watch?v=hyUBMmL0WtA (01.02.2022);
3. Github для самых маленьких #3 | Практика – GitHub terminal | Работа с Git в терминале. Режим доступа: URL: https://www.youtube.com/watch?v=fYFiQ7lpfiE (01.02.2022);
4. Изучение GitHub в одном видео уроке за 15 минут! Режим доступа: URL: https://www.youtube.com/watch?v=JfpCicDUMKc (01.02.2022);
5. Как опубликовать проект на GitHub из PyCharm / уроки Python. Режим доступа: URL: https://www.youtube.com/watch?v=1xkWYCJaBAU (01.02.2022);
6. Уроки Python для начинающих | #17 - Основы ООП Python. Режим доступа: URL: https://www.youtube.com/watch?v=CWSgQcIF--8&list=PL0lO\_mIqDDFXgfuxOEDTCwsWmKezOaDTu&index=17&ab\_channel=ГошаДударь (01.02.2022);
7. Уроки Python для начинающих | #18 - Наследование, инкапсуляция, полиморфизм. Режим доступа: URL: https://www.youtube.com/watch?v=jA5ZTZNfnt0&list=PL0lO\_mIqDDFXgfuxOEDTCwsWmKezOaDTu&index=18 (01.02.2022);
8. Уроки Python для начинающих | #19 - Конструкторы, переопределение методов. Режим доступа: URL: https://www.youtube.com/watch?v=zIN94aYs5O0&list=PL0lO\_mIqDDFXgfuxOEDTCwsWmKezOaDTu&index=19 (01.02.2022);
9. ООП на питоне от egoroff. Режим доступа: URL: <https://stepik.org/lesson/361889/step/1?unit=346427> (01.02.2022);
10. Мэтиз Э. Изучаем Python. Программирование игр, визуализация данных, веб-приложения. — СПб.: Питер, 2017. — 496 с.: ил. — (Серия «Библиотека программиста») (01.02.2022);
11. Федоров Д.Ю. Основы программирования на примере языка Python: учебное пособие. – СПб.: Питер, 2019. – 152 с. (01.02.2022);
12. <https://stepik.org/lesson/330013/step/1?unit=313364> (01.02.2022);
13. Рядченко, В.П. Программирование на языке высокого уровня Python: учебно-методическое пособие / В.П. Рядченко, Л.М. Эльканова, Л.М. Шавтикова. – Черкесск: БИЦ СевКавГГТА, 2018. –144с. (01.02.2022);
14. Как создать exe файл Python Режим доступа: URL: <https://www.youtube.com/watch?v=IQY6WgVwQ80> (01.02.2022);

## ПРИЛОЖЕНИЕ А

Код программы

Основное окно:

#хххххххххххххххххххххххххххххх ИМПОРТИРОВАНИЕ БИБЛИОТЕК хххххххххххххххххххххххххххххх

import time

import turtle

import border

import smile

import zmeyka

import ball

from random import randrange

#хххххххххххххххххххххххххххххх СОЗДАНИЕ ОКНА ПРОГРАММЫ хххххххххххххххххххххххххххххх

screen = turtle.Screen()

screen.title('Змейка')

screen.bgcolor('gold')

screen.setup(650, 700)

screen.tracer(0)

#хххххххххххххххххххххххххххххх СОЗДАНИЕ НЕОБХОДИМЫХ ПЕРЕМЕННЫХ хххххххххххххххххххххххххххххх

bord = border.Border(311, 311)

smile = smile.Trey()

snake = zmeyka.zmeyka()

cherepaha = ball.cherepaha(-300, 0)

#хххххххххххххххххххххххххххххх СОЗДАНИЕ ПЕРВОГО ТИПА ЕДЫ хххххххххххххххххххххххххххххх

ball1 = turtle.Turtle()

ball1.shape('circle')

ball1.color('blue')

ball1.penup()

dx = randrange(0, 10)

dy = randrange(0, 10)

randx = randrange(-300, 300)

randy = randrange(-300, 300)

ball1.goto(randx, randy)

#хххххххххххххххххххххххххххххх СОЗДАНИЕ ВТОРОГО ТИПА ЕДЫ хххххххххххххххххххххххххххххх

food = turtle.Turtle()

food.shape('circle')

food.penup()

food.goto(randrange(-300, 300), randrange(-300, 300))

screen.onkeypress(lambda: snake.snake[0].setheading(90), 'Up') # обработка события, по нажатию клавиш с клавиатуры для змейки (вверх)

screen.onkeypress(lambda: snake.snake[0].setheading(270), 'Down') # обработка события, по нажатию клавиш с клавиатуры для змейки (вниз)

screen.onkeypress(lambda: snake.snake[0].setheading(180), 'Left') # обработка события по нажатию клавиш с клавиатуры для змейки (влево)

screen.onkeypress(lambda: snake.snake[0].setheading(0), 'Right') # обработка события по нажатию клавиш с клавиатуры для змейки (вправо)

screen.listen() # считывание нажатия на клавишу

k = -300

score = 0

while True: # бесконечный массив с обработкой случаев выхода из него

x, y = ball1.position() # позиция

if x + dx >= 295 or x + dx <= -295: # обработка отскока

dx = -dx # переворачиваем прирост в обратное направление

if y + dy >= 295 or y + dy <= -295: # так же только по координате У

dy = -dy #

ball1.goto(x + dx, y + dy) # перемещение шарика

# движение тела змеи

for i in range(len(snake.snake) - 1, 0, -1): # перебираем все элементы змейки

x = snake.snake[i - 1].xcor() # определяем новую координату Х

y = snake.snake[i - 1].ycor() # определяем новую координату У

snake.snake[i].goto(x, y) # двигаем на новую координату

# обработка движения головы змеи

snake.snake[0].forward(20) # голова змеи двигается на 20 пикс

screen.update() # обновляем экран

# обработка положения головы змеи относительно краёв карты

x\_cor = snake.snake[0].xcor() # переменная с координатой Х головы змеи

y\_cor = snake.snake[0].ycor() # переменная с координатой У головы змеи

if x\_cor > 300 or x\_cor < -300: # если уходит за края карты

smile.lose(1, 40, 55, 80, 95, 120, 180, 100)

snake.die(screen, score)

break

if y\_cor > 300 or y\_cor < -300: # если уходит за края карты

snake.die(screen, score)

smile.lose(1, 40, 55, 80, 95, 120, 180, 100)

break

# обработка поедания змеёй еды

if snake.snake[0].distance(food) < 15: # если расстояние от головы до еды меньше 20:

food.goto(randrange(-300, 300), randrange(-300, 300)) # еда переходит в новое место

snake.eat(1, score)

score = score + 1 # добавляем одно очко в результат

if snake.snake[0].distance(ball1) < 15:

ball1.goto(randrange(-300, 300), randrange(-300, 300)) # еда переходит в новое место

snake.eat(3, score)

score = score + 3

# вернёмся к свободной черепашке и задаю ее движение

cherepaha.pos(k) # перемещение свободный черепашки

if k >= 300:

k = -300

k = k + 5

cherepaha.dvig(screen)

time.sleep(0.1) # остановка времени работы программы (так можно настроить сложность игры)

# если произошёл выход из цикла, надо чтобы окно не закрывалось сразу же

screen.mainloop() # застывание окна

import turtle

import math

class cherepaha():

cherepaha = turtle.Turtle()

cherepaha.shapesize(1)

X = 0

Y = 0

def \_\_init\_\_(self, x, y):

self.cherepaha.color('black')

X\_cor = self.cherepaha.xcor()

Y\_cor = self.cherepaha.ycor()

self.cherepaha.penup()

self.cherepaha.goto(x, y)

def pos(self, x):

self.X = x

self.Y = math.sin(self.X)

def dvig(self, x):

self.cherepaha.goto(self.X, self.Y)

x.update()

import turtle

class Border():

def \_\_init\_\_(self, x, y):

border = turtle.Turtle()

border.hideturtle()

border.penup()

border.goto(-x, y)

border.pendown()

border.goto(x, y)

border.goto(x, -y)

border.goto(-x, -y)

border.goto(-x, y)

# вторая полоса края для красоты

border.goto(-x-2, y+2)

border.goto(x-2, y+2)

border.goto(x-2, -y+2)

border.goto(-x-2, -y+2)

border.goto(-x-2, y+2)

# третья полоса края для красоты

border.goto(-x-4, y+4)

border.goto(x-4, y+4)

border.goto(x-4, -y+4)

border.goto(-x-4, -y+4)

border.goto(-x-4, y+4)

turtle.penup()

turtle.hideturtle()

turtle.goto(-x+11, y+11)

turtle.write("Групповой проект "

"Сахаров, Буряк, Сыздыков, Шамсутдинов", move=False, font=("Arial", 16, "normal"))

turtle.goto(-x+20, -y-29)

turtle.write("Синяя еда - 3 очка, черная еда - 1 очко. "

"Управление змейкой стрелочками", move=False, font=("Arial", 13, "normal"))

import turtle

class Trey():

def \_\_init\_\_(self):

smile = turtle.Turtle()

smile.hideturtle()

smile.penup()

smile.goto(0, 280)

smile.pendown()

for i in range(360):

smile.right(1)

smile.forward(2)

smile.penup()

smile.goto(40, 120)

smile.left(180)

smile.pendown()

smile.forward(80)

smile.penup()

smile.goto(-40, 250)

smile.left(90)

smile.pendown()

smile.forward(50)

smile.penup()

smile.goto(40, 250)

smile.pendown()

smile.forward(50)

def lose(self, x, y, z, m, u, i, o, p):

smil = turtle.Turtle()

smil.penup()

smil.color("red")

smil.goto(m, i)

smil.right(o)

smil.pendown()

smil.forward(i)

smil.penup()

smil.goto(m, i)

smil2 = turtle.Turtle()

smil2.color("black")

smil2.penup()

smil2.goto(z, p)

smil2.left(u)

smil2.pendown()

for i in range(o):

smil2.forward(x)

smil2.left(x)

import turtle

class zmeyka:

snake = []

len\_snake = len(snake)

n=0

def \_\_init\_\_(self):

for i in range(3):

snake\_n = turtle.Turtle()

snake\_n.shape('square')

snake\_n.color('black')

snake\_n.penup()

if i > 0:

if i % 2 == 0:

snake\_n.color('silver')

elif i % 2 == 1:

snake\_n.color('white')

else:

snake\_n.color('white')

self.snake.append(snake\_n)

def eat(self, x, y):

for i in range(x):

self.n = self.n + x

for i in range(x):

snake\_n = turtle.Turtle()

snake\_n.shape('square')

snake\_n.color("silver")

snake\_n.penup()

if i > 0:

if i % 2 == 0:

snake\_n.color('silver')

elif i % 2 == 1:

snake\_n.color('white')

else:

snake\_n.color('white')

self.snake.append(snake\_n)

def die(self, x, y):

x.bgcolor('red')

turtle.goto(-163, -20)

turtle.color("gold")

turtle.write("ВЫ ПРОИГРАЛИ!", move=False, font=("Arial", 30, "normal"))

print('Ваш результат: ', y)