|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«Московский технологический университет»**  (МГУПИ) |

Институт ИКБСП специальность (направление) 09.03.02

Кафедра КБ4 «Автоматизированные системы управления»

Дисциплина «Технология программирования в среде Python»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовой работе на тему:**

**Настольный теннис с использованием библиотеки tkinter.**

Студент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Мельничук А.В.

подпись, дата

Группа БСБО-02-15

Работа защищена на оценку\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель проекта (работы)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лебедев А.С.

подпись, дата

Члены комиссии\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

инициалы и фамилия

подпись, дата

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата инициалы и фамилия

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата инициалы и фамилия

МОСКВА 2017 г.

|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«Московский технологический университет»**  (МГУПИ) |

Институт ИКБСП специальность (направление) 09.03.02

Кафедра КБ4 «Автоматизированные системы управления»

Дисциплина «Технология программирования в среде Python»

**ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ (РАБОТУ)**

Студент 3 курса группа: БСБО-02-15

1 Тема:

**Настольный теннис с использованием библиотеки tkinter**

2 Срок представления проекта (работы) к защите 23.12.2017 г.

3 Исходные данные для разработки

ОС Linux, интерпретатор Python

4 Содержание пояснительной записки:

Титульный лист

Задание

Содержание

Введение

1 Требования к программе или программному изделию

2 Стадии и этапы разработки

3 Составляющие проекта

Заключение

Список использованных источников

Руководитель проекта (работы)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лебедев А.С.

подпись, дата

Задание принял к исполнению\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Мельничук А.В.

подпись, дата

МОСКВА 2017 г

**Содержание**

[Введение 4](#_Toc501747472)

[Цель 5](#_Toc501747473)

[Задачи 6](#_Toc501747474)

[Основания для разработки 7](#_Toc501747475)

[Назначение разработки 7](#_Toc501747476)

[Требования к программе или программному изделию 7](#_Toc501747477)

[Требования к функциональным характеристикам 7](#_Toc501747478)

[Требования к составу и параметрам технических средств. 8](#_Toc501747479)

[Требования к информационной и программной совместимости 9](#_Toc501747480)

[Стадии и этапы разработки 10](#_Toc501747481)

[Модули приложения. 11](#_Toc501747482)

[Заключение 12](#_Toc501747483)

[Приложение 1 14](#_Toc501747484)

[P-P.py 14](#_Toc501747485)

# **Введение**

**Аркада** — жанр компьютерных игр, характеризующийся коротким по времени, но интенсивным игровым процессом.

В строгом смысле, аркадной считается игра для аркадных игровых автоматов. Также аркадными называют игры, которые портированы с аркадного автомат.

Наиболее простейшие аркады можно реализовать на языке программирования python c помощью библиотеки tkinter.

**Tkinter** - кросс-платформенная графическая библиотека на основе средств Tk (широко распространённая в мире GNU/Linux и других UNIX‐подобных систем, портирована также и на Microsoft Windows), Входит в стандартную библиотеку Python.

В составе библиотеки присутствуют общие графические компоненты:

• Toplevel

Окно верхнего уровня (корневой виджет)

• Tk

• Frame

Рамка. Содержит в себе другие визуальные компоненты

• Label

Этикетка. Показывает некоторый текст или графическое изображение

Canvas

Холст. Может использоваться для вывода графических примитивов, например, для построения графиков

• Button

Кнопка. Простая кнопка для выполнения команды и других действий

• Menubutton

Кнопка-меню. Используется для организации pulldown-меню

• Text

Форматированный текст. Позволяет показывать, редактировать и форматировать текст с использованием различных стилей, а также внедрять в текст рисунки и окна.

• Scrollbar

Полоса прокрутки. Может использоваться вместе с некоторыми другими компонентами для их прокрутки

• OptionMenu

• Spinbox

• LabelFrame

• PanedWindow

• Menu

# **Цель**

Разработать игру «Настольный теннис» с возможностью управления двумя игроками.

# **Задачи**

1. Создание игрового поля размером 900\*300.

2. Реализовать движение мячика.

3. Разработать возможность управления ракетками клавишами w, a, s, d и стрелочками.

4. Реализовать возможность случайных отскоков мяча.

5. Реализовать подсчет очков и появление мяча в центре поля.

# **Основания для разработки**

Учебный план по дисциплине «Технология программирования в среде Python», направление бакалавриата «09.03.02», МТУ, 2017.

# **Назначение разработки**

Программа предназначена для моделирования игры «Настольный теннис» с возможностью управления двумя игроками.

# **Требования к программе или программному изделию**

## **Требования к функциональным характеристикам**

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

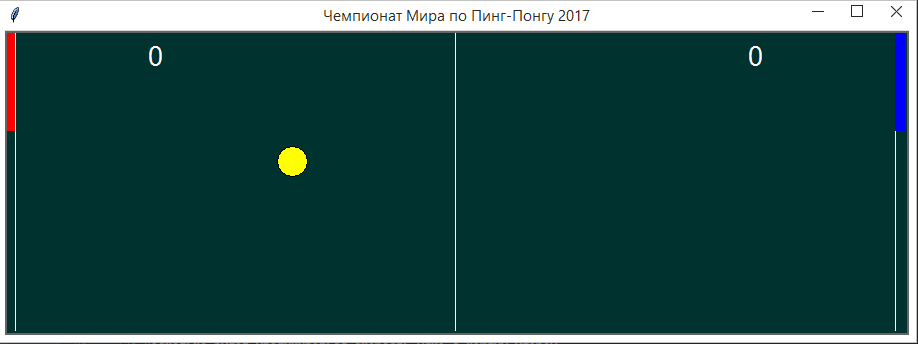
1. Реализация подсчёта очков.

2. Управление ракетками на клавишах w, a, s, d и стрелочках.

3. После завершения игры у пользователя должна появиться возможность сыграть ещё раз.

4. По окончанию игры создать возможность выхода.

При запуске программы пользователю сразу же становится доступна игра. (рисунок 1)



*Рисунок 1- Окно процесса игры*

Во время игры ведётся учёт очков. (рисунок 2)

**

*Рисунок 2- Окно процесса игры*

В результате завершения игры пользователю выводится результат, а также появляется возможность начать игру заново или выйти из приложения.

(рисунок 3)



*Рисунок 3-окно завершения игры*

## **Требования к составу и параметрам технических средств.**

* Процессор: Intel Core i3-4340 / AMD FX-6300
* Оперативная память: 1 GB ОЗУ
* Сеть: Широкополосное подключение к интернету
* Место на диске: 1 GB
* Наличие монитора

## **Требования к информационной и программной совместимости**

В состав технических средств должен входить персональный компьютер на базе операционной системы Windows 7 и далее, или семейства ОС UNIX.

Установленный интерпретатор Python 3.6.2, а также библиотека tkinter.

#### **Tkinter**

Tkinter – это графическая библиотека на основе Tk, которая входит в стандартную библиотеку Python. Tk является библиотекой базовых элементов графического интерфейса для языка Tcl. Tkinter реализован как оболочка Python для Tcl интерпретатора, встроенного в интерпретатор Python.

# **Стадии и этапы разработки**

1.Анализ предметной области.

2.Разработка схемы базы данных.

3.Разработка технического задания на разработку Программы.

4.Разработка структур данных, предназначенных для представления сущностей в оперативной памяти.

5.Разработка программного кода, реализующего функциональные требования к программе.

Для разработки данного программного обеспечения (ПО) используется интерпретатор Python 3.6.2 и интегрированная среда разработки (IDE) PyCharm 2017, а также библиотеку tkinter.

# **Модули приложения.**

Файл P-P.py содержит основной код программы в который входит:

* Окно процесса игры
* Окно завершения игры
* Меню выхода из приложения

# **Заключение**

В данной курсовой работе была разработана игра «Настольный теннис».

Были выполнены такие задачи как создание игрового поля, движение мячика, управление ракетками, реализация отскоков, а также подсчет очков и появление мяча в центре поля .**Список использованных источников**

1. URL: <https://dev.rbcafe.com/python/python-3.5.1-pdf/library.pdf-> документация к библиотеке tkinter. (дата обращения: 11.12.2017)
2. URL: <http://www.tcltk.co.kr/files/TclTk_Introduction_To_Tkiner.pdf>- документация к библиотеке tkinter. (дата обращения: 12.12.2017)
3. Лутц, М. Программирование на Python. Том 1, 4-е издание. Москва: Символ-Плюс 2011.
4. Алексей Горожанов "PyQT для лингвистов". Том 1, 1-е издание. Москва: Символ-Плюс 2015
5. Прохоренок Н., Дронов В. - Python 3 и PyQt 5. Том 1, 2-е издание. Санкт-Петербург: Питер 2016.

# **Приложение 1**

Исходный код программы:

## **P-P.py**

from tkinter import \*

# импортируем библиотеку random

import random

from tkinter.font import Font

# Добавляем глобальные переменные

# глобальные переменные

# настройки окна

WIDTH = 900

HEIGHT = 300

# настройки ракеток

# ширина ракетки

PAD\_W = 10

# высота ракетки

PAD\_H = 100

# настройки мяча

# Насколько будет увеличиваться скорость мяча с каждым ударом

BALL\_SPEED\_UP = 1.05

# Максимальная скорость мяча

BALL\_MAX\_SPEED = 40

# радиус мяча

BALL\_RADIUS = 30

INITIAL\_SPEED = 20

BALL\_X\_SPEED = INITIAL\_SPEED

BALL\_Y\_SPEED = INITIAL\_SPEED

# Счет игроков

PLAYER\_1\_SCORE = 0

PLAYER\_2\_SCORE = 0

# Добавим глобальную переменную отвечающую за расстояние

# до правого края игрового поля

right\_line\_distance = WIDTH - PAD\_W

flag = True

def update\_score(player):

global PLAYER\_1\_SCORE, PLAYER\_2\_SCORE

if player == "right":

PLAYER\_1\_SCORE += 1

c.itemconfig(p\_1\_text, text=PLAYER\_1\_SCORE)

check\_score()

else:

PLAYER\_2\_SCORE += 1

c.itemconfig(p\_2\_text, text=PLAYER\_2\_SCORE)

check\_score()

def check\_score():

global flag,root

if PLAYER\_1\_SCORE == 5 or PLAYER\_2\_SCORE == 5:

flag = False

winner = 0

winner\_str =""

if PLAYER\_1\_SCORE > PLAYER\_2\_SCORE:

winner = PLAYER\_1\_SCORE

los = PLAYER\_2\_SCORE

winner\_str = "Player BLUE"

else:

winner\_str = "Player RED"

winner = PLAYER\_2\_SCORE

los = PLAYER\_1\_SCORE

global t

t = Toplevel()

t.wm\_title("Game Over")

helv36 = Font(family='Helvetica', size=20, weight='bold')

l = Label(t,font=helv36, text=winner\_str+" WIN!!!\n With score: "+str(winner)+" : "+str(los))

l.pack(side="top", fill="both", expand=True, padx=100, pady=100)

b = Button(t, text="Retry", command=restart, bg="white",fg="blue", font=helv36)

b.pack()

c = Button(t, text="Exit", command=exit, bg="white",fg="blue", font=helv36)

c.pack()

# c.unbind("<KeyPress>", movement\_handler)

# c.unbind("<KeyRelease>")

def restart():

global PLAYER\_1\_SCORE, PLAYER\_2\_SCORE, flag

t.destroy()

PLAYER\_1\_SCORE = -1

PLAYER\_2\_SCORE = -1

update\_score("left")

update\_score("right")

flag = True

def exit():

t.destroy()

root.destroy()

def spawn\_ball():

global BALL\_X\_SPEED

# Выставляем мяч по центру

c.coords(BALL, WIDTH / 2 - BALL\_RADIUS / 2,

HEIGHT / 2 - BALL\_RADIUS / 2,

WIDTH / 2 + BALL\_RADIUS / 2,

HEIGHT / 2 + BALL\_RADIUS / 2)

# Задаем мячу направление в сторону проигравшего игрока,

# но снижаем скорость до изначальной

BALL\_X\_SPEED = -(BALL\_X\_SPEED \* -INITIAL\_SPEED) / abs(BALL\_X\_SPEED)

# функция отскока мяча

def bounce(action):

global BALL\_X\_SPEED, BALL\_Y\_SPEED

# ударили ракеткой

if action == "strike":

BALL\_Y\_SPEED = random.randrange(-10, 10)

if abs(BALL\_X\_SPEED) < BALL\_MAX\_SPEED:

BALL\_X\_SPEED \*= -BALL\_SPEED\_UP

else:

BALL\_X\_SPEED = -BALL\_X\_SPEED

else:

BALL\_Y\_SPEED = -BALL\_Y\_SPEED

# устанавливаем окно

root = Tk()

root.title("Чемпионат Мира по Пинг-Понгу 2017")

# область анимации

c = Canvas(root, width=WIDTH, height=HEIGHT, background="#003330")

c.pack()

# элементы игрового поля

# левая линия

c.create\_line(PAD\_W, 0, PAD\_W, HEIGHT, fill="white")

# правая линия

c.create\_line(WIDTH - PAD\_W, 0, WIDTH - PAD\_W, HEIGHT, fill="white")

# центральная линия

c.create\_line(WIDTH / 2, 0, WIDTH / 2, HEIGHT, fill="white")

# установка игровых объектов

# создаем мяч

BALL = c.create\_oval(WIDTH / 2 - BALL\_RADIUS / 2,

HEIGHT / 2 - BALL\_RADIUS / 2,

WIDTH / 2 + BALL\_RADIUS / 2,

HEIGHT / 2 + BALL\_RADIUS / 2, fill="yellow")

# левая ракетка

LEFT\_PAD = c.create\_line(PAD\_W / 2, 0, PAD\_W / 2, PAD\_H, width=PAD\_W, fill="red")

# правая ракетка

RIGHT\_PAD = c.create\_line(WIDTH - PAD\_W / 2, 0, WIDTH - PAD\_W / 2,

PAD\_H, width=PAD\_W, fill="blue")

p\_1\_text = c.create\_text(WIDTH - WIDTH / 6, PAD\_H / 4,

text=PLAYER\_1\_SCORE,

font="Arial 20",

fill="white")

p\_2\_text = c.create\_text(WIDTH / 6, PAD\_H / 4,

text=PLAYER\_2\_SCORE,

font="Arial 20",

fill="white")

# добавим глобальные переменные для скорости движения мяча

# по горизонтали

BALL\_X\_CHANGE = 20

# по вертикали

BALL\_Y\_CHANGE = 0

def move\_ball():

global flag

if flag:

# определяем координаты сторон мяча и его центра

ball\_left, ball\_top, ball\_right, ball\_bot = c.coords(BALL)

ball\_center = (ball\_top + ball\_bot) / 2

# вертикальный отскок

# Если мы далеко от вертикальных линий - просто двигаем мяч

if ball\_right + BALL\_X\_SPEED < right\_line\_distance and \

ball\_left + BALL\_X\_SPEED > PAD\_W:

c.move(BALL, BALL\_X\_SPEED, BALL\_Y\_SPEED)

# Если мяч касается своей правой или левой стороной границы поля

elif ball\_right == right\_line\_distance or ball\_left == PAD\_W:

# Проверяем правой или левой стороны мы касаемся

if ball\_right > WIDTH / 2:

# Если правой, то сравниваем позицию центра мяча

# с позицией правой ракетки.

# И если мяч в пределах ракетки делаем отскок

if c.coords(RIGHT\_PAD)[1] < ball\_center < c.coords(RIGHT\_PAD)[3]:

bounce("strike")

else:

# Иначе игрок пропустил - тут оставим пока pass, его мы заменим на подсчет очков и респаун мячика

update\_score("left")

spawn\_ball()

else:

# То же самое для левого игрока

if c.coords(LEFT\_PAD)[1] < ball\_center < c.coords(LEFT\_PAD)[3]:

bounce("strike")

else:

update\_score("right")

spawn\_ball()

# Проверка ситуации, в которой мячик может вылететь за границы игрового поля.

# В таком случае просто двигаем его к границе поля.

else:

if ball\_right > WIDTH / 2:

c.move(BALL, right\_line\_distance - ball\_right, BALL\_Y\_SPEED)

else:

c.move(BALL, -ball\_left + PAD\_W, BALL\_Y\_SPEED)

# горизонтальный отскок

if ball\_top + BALL\_Y\_SPEED < 0 or ball\_bot + BALL\_Y\_SPEED > HEIGHT:

bounce("ricochet")

# зададим глобальные переменные скорости движения ракеток

# скорось с которой будут ездить ракетки

PAD\_SPEED = 20

# скорость левой платформы

LEFT\_PAD\_SPEED = 0

# скорость правой ракетки

RIGHT\_PAD\_SPEED = 0

# функция движения обеих ракеток

def move\_pads():

# для удобства создадим словарь, где ракетке соответствует ее скорость

PADS = {LEFT\_PAD: LEFT\_PAD\_SPEED,

RIGHT\_PAD: RIGHT\_PAD\_SPEED}

# перебираем ракетки

for pad in PADS:

# двигаем ракетку с заданной скоростью

c.move(pad, 0, PADS[pad])

# если ракетка вылезает за игровое поле возвращаем ее на место

if c.coords(pad)[1] < 0:

c.move(pad, 0, -c.coords(pad)[1])

elif c.coords(pad)[3] > HEIGHT:

c.move(pad, 0, HEIGHT - c.coords(pad)[3])

def main():

global flag

# print(flag)

if flag:

move\_ball()

move\_pads()

# вызываем саму себя каждые 30 миллисекунд

root.after(30, main)

# Установим фокус на Canvas чтобы он реагировал на нажатия клавиш

c.focus\_set()

# Напишем функцию обработки нажатия клавиш

def movement\_handler(event):

global LEFT\_PAD\_SPEED, RIGHT\_PAD\_SPEED

if event.keysym == "w":

LEFT\_PAD\_SPEED = -PAD\_SPEED

elif event.keysym == "s":

LEFT\_PAD\_SPEED = PAD\_SPEED

elif event.keysym == "Up":

RIGHT\_PAD\_SPEED = -PAD\_SPEED

elif event.keysym == "Down":

RIGHT\_PAD\_SPEED = PAD\_SPEED

# Привяжем к Canvas эту функцию

c.bind("<KeyPress>", movement\_handler)

# Создадим функцию реагирования на отпускание клавиши

def stop\_pad(event):

global LEFT\_PAD\_SPEED, RIGHT\_PAD\_SPEED

if event.keysym in "ws":

LEFT\_PAD\_SPEED = 0

elif event.keysym in ("Up", "Down"):

RIGHT\_PAD\_SPEED = 0

# Привяжем к Canvas эту функцию

c.bind("<KeyRelease>", stop\_pad)

# запускаем движение

main()

# запускаем работу окна

root.mainloop()