**Нетиповое муниципальное бюджетное общеобразовательное учреждение**

**Анжеро-Судженского городского округа**

**«Гимназия №11»**

**ПРОЕКТ**

по учебному предмету Информатика

на тему «Разработка игры “Сапер”»

**Выполнил**:

Лищенко Тимофей

НМБОУ «Гимназия №11»

9 класс

**Руководитель проекта**:

Мангазеев А. Ю.

учитель информатики и икт

НМБОУ «Гимназия №11»

Анжеро-Судженск 2020

**Оглавление**

1. Введение
2. Идея проекта
3. О игре
4. Принцип игры
5. Код программы
6. Заключение
7. Список литературы

**Введение**

**Что такое Python?**

Python — активно развивающийся язык программирования, общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Название языка произошло вовсе не от названия семейства пресмыкающихся. Автор назвал язык в честь популярного британского комедийного телешоу 1970-х «Летающий цирк Монти Пайтона». Впрочем, всё равно название языка чаще связывают именно со змеёй, нежели с передачей — пиктограммы файлов в KDE или в Microsoft Windows и даже эмблема на сайте python.org (до выхода версии 2.5) изображают змеиные головы. Важная цель разработчиков Python — создавать его забавным для использования. Это отражено в его названии, которое пришло из Монти Пайтона.

Разработка языка Python была начата в конце [1980-х годов](https://ru.wikipedia.org/wiki/1980-%D0%B5) сотрудником голландского института CWI [Гвидо ван Россумом](https://ru.wikipedia.org/wiki/%D0%93%D0%B2%D0%B8%D0%B4%D0%BE_%D0%B2%D0%B0%D0%BD_%D0%A0%D0%BE%D1%81%D1%81%D1%83%D0%BC). Для распределённой ОС [Amoeba](https://ru.wikipedia.org/wiki/Amoeba_(%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0)" \o "Amoeba (операционная система)) требовался расширяемый [скриптовый язык](https://ru.wikipedia.org/wiki/%D0%A1%D1%86%D0%B5%D0%BD%D0%B0%D1%80%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA), и Гвидо начал писать Python на досуге, позаимствовав некоторые наработки для языка [ABC](https://ru.wikipedia.org/wiki/ABC_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)) (Гвидо участвовал в разработке этого языка, ориентированного на обучение программированию). В феврале [1991 года](https://ru.wikipedia.org/wiki/1991_%D0%B3%D0%BE%D0%B4) Гвидо опубликовал исходный текст в [группе новостей](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D1%83%D0%BF%D0%BF%D0%B0_%D0%BD%D0%BE%D0%B2%D0%BE%D1%81%D1%82%D0%B5%D0%B9) alt.sources. С самого начала Python проектировался как [объектно-ориентированный язык](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F). [3 декабря](https://ru.wikipedia.org/wiki/3_%D0%B4%D0%B5%D0%BA%D0%B0%D0%B1%D1%80%D1%8F) [2008 года](https://ru.wikipedia.org/wiki/2008_%D0%B3%D0%BE%D0%B4), после длительного тестирования, вышла первая версия Python 3000 (или Python 3.0, также используется [сокращение](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D0%B1%D1%80%D0%B5%D0%B2%D0%B8%D0%B0%D1%82%D1%83%D1%80%D0%B0) Py3k). В Python 3000 устранены многие недостатки архитектуры с максимально возможным (но не полным) сохранением совместимости со старыми версиями Python. На сегодня поддерживаются обе ветви развития (Python 3.x и 2.x), но поддержка Python 2.7 заканчивается в 2020 году.

**Идея проекта**

Написать игру “Сапер” на языке программирования Python с использованием библиотеки Tkinter.

**О игре**

«Сапёр» ( англ. Minesweeper) — компьютерная игра-головоломка.

Цель игры состоит в том, чтобы очистить прямоугольную доску, содержащую скрытые "мины" или бомбы, не взорвав ни одной из них, с помощью подсказок о количестве соседних мин в каждом поле. Игра берет свое начало с 1960-х годов и была написана для многих компьютерных платформ, используемых сегодня. Она имеет множество вариаций и ответвлений.Некоторые версии тральщика установят доску, никогда не ставя мину на первый обнаруженный квадрат. Minesweeper для версий Windows защищает первый обнаруженный квадрат; начиная с Windows 7, игроки могут выбрать для воспроизведения доску, в которой игра ведется путем выявления квадратов сетки, нажав или иным образом указав каждый квадрат. Если обнаруживается квадрат, содержащий мину, игрок проигрывает игру. Если мина не обнаружена, вместо этого в квадрате отображается цифра, указывающая, сколько соседних квадратов содержат мины; если рядом нет мин, квадрат становится пустым, и все соседние квадраты будут рекурсивно раскрыты. Игрок использует эту информацию, чтобы вывести содержимое других квадратов, и может либо безопасно раскрыть каждый квадрат, либо пометить квадрат как содержащий мину.

**Принцип игры**

Плоское или объёмное игровое поле разделено на смежные ячейки (квадраты, шестиугольники, кубы и т. п.), некоторые из которых «заминированы»; количество «заминированных» ячеек известно. Целью игры является открытие всех ячеек, не содержащих мины.

Игрок открывает ячейки, стараясь не открыть ячейку с миной. Открыв ячейку с миной, он проигрывает. Мины расставляются после первого хода, поэтому в новых версиях проиграть на первом же ходу невозможно. В старых версияхдовольно частая ситуация, что под первой открытой ячейкой оказывалась мина. Если под открытой ячейкой мины нет, то в ней появляется число, показывающее, сколько ячеек, соседствующих с только что открытой, «заминировано» (в каждом варианте игры соседство определяется по-своему); используя эти числа, игрок пытается рассчитать расположение мин, однако иногда даже в середине и в конце игры некоторые ячейки всё же приходится открывать наугад. Если под соседними ячейками тоже нет мин, то открывается некоторая «не заминированная» область до ячеек, в которых есть цифры. «Заминированные» ячейки игрок может пометить, чтобы случайно не открыть их. Открыв все «не заминированные» ячейки, игрок выигрывает.

**Код программы**

from tkinter import \*

from tkinter import messagebox

import random

from collections import deque

class Minesweeper:

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

self.tile\_plain = PhotoImage(file="images/tile\_plain.png")

self.tile\_clicked = PhotoImage(file="images/tile\_clicked.png")

self.tile\_mine = PhotoImage(file="images/tile\_mine.png")

self.tile\_flag = PhotoImage(file="images/tile\_flag.png")

self.tile\_wrong = PhotoImage(file="images/tile\_wrong.png")

self.tile\_no = []

for x in range(1, 9):

self.tile\_no.append(PhotoImage(file="images/tile\_"+str(x)+".png"))

frame = Frame(master)

frame.pack()

self.label1 = Label(frame, text="Сапер")

self.label1.grid(row=0, column=0, columnspan=10)

self.flags = 0

self.correct\_flags = 0

self.clicked = 0

self.buttons = dict({})

self.mines = 0

x\_coord = 1

y\_coord = 0

for x in range(0, 100):

mine = 0

gfx = self.tile\_plain

if random.uniform(0.0, 0.5) < 0.1:

mine = 1

self.mines += 1

self.buttons[x] = [Button(frame, image=gfx),

mine,

0,

x,

[x\_coord, y\_coord],

0]

self.buttons[x][0].bind('<Button-1>', self.lclicked\_wrapper(x))

self.buttons[x][0].bind('<Button-3>', self.rclicked\_wrapper(x))

y\_coord += 1

if y\_coord == 10:

y\_coord = 0

x\_coord += 1

for key in self.buttons:

self.buttons[key][0].grid(

row=self.buttons[key][4][0], column=self.buttons[key][4][1])

for key in self.buttons:

nearby\_mines = 0

if self.check\_for\_mines(key-9):

nearby\_mines += 1

if self.check\_for\_mines(key-10):

nearby\_mines += 1

if self.check\_for\_mines(key-11):

nearby\_mines += 1

if self.check\_for\_mines(key-1):

nearby\_mines += 1

if self.check\_for\_mines(key+1):

nearby\_mines += 1

if self.check\_for\_mines(key+9):

nearby\_mines += 1

if self.check\_for\_mines(key+10):

nearby\_mines += 1

if self.check\_for\_mines(key+11):

nearby\_mines += 1

self.buttons[key][5] = nearby\_mines

self.label2 = Label(frame, text="Мины: "+str(self.mines))

self.label2.grid(row=11, column=0, columnspan=5)

self.label3 = Label(frame, text="Флажки: "+str(self.flags))

self.label3.grid(row=11, column=4, columnspan=5)

def check\_for\_mines(self, key):

try:

if self.buttons[key][1] == 1:

return True

except KeyError:

pass

def lclicked\_wrapper(self, x):

return lambda Button: self.lclicked(self.buttons[x])

def rclicked\_wrapper(self, x):

return lambda Button: self.rclicked(self.buttons[x])

def lclicked(self, button\_data):

if button\_data[1] == 1:

for key in self.buttons:

if self.buttons[key][1] != 1 and self.buttons[key][2] == 2:

self.buttons[key][0].config(image=self.tile\_wrong)

if self.buttons[key][1] == 1 and self.buttons[key][2] != 2:

self.buttons[key][0].config(image=self.tile\_mine)

self.gameover()

else:

if button\_data[5] == 0:

button\_data[0].config(image=self.tile\_clicked)

self.clear\_empty\_tiles(button\_data[3])

else:

button\_data[0].config(image=self.tile\_no[button\_data[5]-1])

if button\_data[2] != 1:

button\_data[2] = 1

self.clicked += 1

if self.clicked == 100 - self.mines:

self.victory()

def rclicked(self, button\_data):

if button\_data[2] == 0:

button\_data[0].config(image=self.tile\_flag)

button\_data[2] = 2

button\_data[0].unbind('<Button-1>')

if button\_data[1] == 1:

self.correct\_flags += 1

self.flags += 1

self.update\_flags()

elif button\_data[2] == 2:

button\_data[0].config(image=self.tile\_plain)

button\_data[2] = 0

button\_data[0].bind(

'<Button-1>', self.lclicked\_wrapper(button\_data[3]))

if button\_data[1] == 1:

self.correct\_flags -= 1

self.flags -= 1

self.update\_flags()

def check\_tile(self, key, queue):

try:

if self.buttons[key][2] == 0:

if self.buttons[key][5] == 0:

self.buttons[key][0].config(image=self.tile\_clicked)

queue.append(key)

else:

self.buttons[key][0].config(

image=self.tile\_no[self.buttons[key][5]-1])

self.buttons[key][2] = 1

self.clicked += 1

except KeyError:

pass

def clear\_empty\_tiles(self, main\_key):

queue = deque([main\_key])

while len(queue) != 0:

key = queue.popleft()

self.check\_tile(key-9, queue)

self.check\_tile(key-10, queue)

self.check\_tile(key-11, queue)

self.check\_tile(key-1, queue)

self.check\_tile(key+1, queue)

self.check\_tile(key+9, queue)

self.check\_tile(key+10, queue)

self.check\_tile(key+11, queue)

def gameover(self):

messagebox.showinfo("Ты проиграл", "Попоробуй еще раз!")

global root

root.destroy()

def victory(self):

messagebox.showinfo("Конец игры", "Ты победил!")

global root

root.destroy()

def update\_flags(self):

self.label3.config(text="Флажки: "+str(self.flags))

def main():

global root

root = Tk()

root.title("Сапер")

minesweeper = Minesweeper(root)

root.mainloop()

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Заключение**

Мы написали полноценно функциональную игру “Сапер”. Использовали библиотеку Tkinter, как основной графический интерфейс. В итоге можем радовать себя и своих близких данной головомкой.

**Список литературы**

1. <https://ru.wikipedia.org/wiki/%D0%A1%D0%B0%D0%BF%D1%91%D1%80_(%D0%B8%D0%B3%D1%80%D0%B0)>
2. <https://github.com/TimmmofeyD/mine>
3. <https://www.python.org/>
4. <https://git-scm.com/book/ru/v2/%D0%92%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5-%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D1%8B-Git>
5. <https://docs.python.org/3/library/tkinter.html>