**Игра «Вертолётик»**

Проект состоит из пяти файлов.

1. Файл main.py:

from map import Map

from helicopter import Helicopter as helico

from clouds import Clouds

import time

import os

from pynput import keyboard

import json

TIME\_SLEEP = 0.05

TREE\_UPDATE = 50

FIRE\_UPDATE = 100

MAP\_W, MAP\_H = 10, 10

CLOUDS\_UPDATE = 150

# слой облаков

clouds = Clouds(MAP\_W, MAP\_H)

# вертолёт

helico = helico(MAP\_W, MAP\_H)

# карта игры

game\_map = Map(MAP\_W, MAP\_H)

# генерация элементов игры

game\_map.generate\_forest(3, 10)

game\_map.generate\_rivers(25)

game\_map.generate\_upgrade\_shop()

game\_map.generate\_hospital()

# матрица смещения вертолёта

MOVES = {'w': (-1, 0), 'd': (0, 1), 's': (1, 0), 'a': (0, -1)}

# обработка клавиш

def process\_key(key):

global game\_map, clouds, helico, tick

c = key.char.lower()

# перемещение вертолёта

if c in MOVES.keys():

helico.move(MOVES[c][0], MOVES[c][1])

# сохранение игры в файл

if c == 'n':

data = {'helicopter': helico.export\_data(),

'map' : game\_map.export\_data(),

'clouds' : clouds.export\_data(),

'tick': tick}

with open('level.json', 'w') as lvl:

json.dump(data, lvl)

# восстановление игры из файла

if c == 'm':

with open('level.json', 'r') as lvl:

data = json.load(lvl)

tick = data['tick']

game\_map.import\_data(data['map'])

clouds.import\_data(data['clouds'])

helico.import\_data(data['helicopter'])

else:

None

# обработчик клавиатуры

listener = keyboard.Listener(on\_release=process\_key)

listener.start()

tick = 1

# бесконечный цикл игры

while True:

os.system('clear')

print('TICK= ', tick)

game\_map.process\_helicopter(helico, clouds)

helico.print\_stats()

game\_map.print\_map(helico, clouds)

tick += 1

time.sleep(TIME\_SLEEP)

# обновление игровых элементов

if tick % TREE\_UPDATE == 0:

game\_map.generate\_tree()

if tick % FIRE\_UPDATE == 0:

game\_map.update\_fires()

if tick % CLOUDS\_UPDATE == 0:

clouds.update()

2. Файл map.py:

from utils import randbool

from utils import randcell

from utils import randcell2

from clouds import Clouds

import os

# 0- поле

# 1 - дерево

# 2 - река

# 3 - госпиталь

# 4 - upgrade shop

# 5 - огонь

CELL\_TYPES="🟩🌲🌊🏥🔧🔥"

TREE\_BONUS = 100

UPGRADE\_COST = 500

LIFE\_COST = 100

class Map:

def \_\_init\_\_(self, w, h):

self.w = w

self.h = h

self.cells = [[0 for i in range(w)] for j in range(h)]

self.generate\_forest(5,10)

self.generate\_rivers(10)

self.generate\_rivers(10)

self.generate\_upgrade\_shop()

self.generate\_hospital()

self.clouds = Clouds(w, h)

# проверка координат внутри поля

def check\_bounds(self, x, y):

if (x < 0 or y <0 or x>= self.h or y >= self.w):

return False

return True

# вывод карты

def print\_map(self, helico, clouds):

# верхняя окантовка поля

print("⬛" \* (self.w+2))

for ri in range(self.h):

print("⬛", end="")

for ci in range(self.w):

cell = self.cells[ri][ci]

# обычное облако

if (clouds.cells[ri][ci] == 1):

print("⬜", end="")

# грозовое облако

elif (clouds.cells[ri][ci] == 2):

print("⚡", end="")

# вертолётик

elif (helico.x == ri and helico.y == ci):

print("🚁", end= "")

elif (cell >= 0 and cell < len(CELL\_TYPES)):

print(CELL\_TYPES[cell], end="")

# окантовка игрового поля

print("⬛")

print("⬛" \* (self.w+2))

# генерация водоёмов

def generate\_rivers(self, l):

rc = randcell(self.w, self.h)

rx,ry = rc[0], rc[1]

self.cells[rx][ry] = 2

while l > 0:

rc2 = randcell2(rx,ry)

rx2, ry2 = rc2[0], rc2[1]

if self.check\_bounds(rx2,ry2):

self.cells[rx2][ry2] = 2

rx,ry = rx2, ry2

l -= 1

# генерация леса

def generate\_forest(self, r, mxr):

for ri in range(self.h):

for ci in range(self.w):

if randbool(r,mxr):

self.cells[ri][ci] = 1

# генерация дерева

def generate\_tree(self):

c = randcell(self.w, self.h)

cx, cy = c[0], c[1]

if (self.cells[cx][cy] == 0):

self.cells[cx][cy] = 1

# генерация апгрейда

def generate\_upgrade\_shop(self):

c = randcell(self.w, self.h)

cx, cy = c[0], c[1]

self.cells[cx][cy] = 4

# генерация больницы

def generate\_hospital(self):

c = randcell(self.w, self.h)

cx, cy = c[0], c[1]

if self.cells[cx][cy] != 4:

self.cells[cx][cy] = 3

else:

self.generate\_hospital()

# генерация огня

def add\_fire(self):

c = randcell(self.w, self.h)

cx, cy = c[0], c[1]

if self.cells[cx][cy] == 1:

self.cells[cx][cy] = 5

# обновление огня

def update\_fires(self):

for ri in range(self.h):

for ci in range(self.w):

cell = self.cells[ri][ci]

if cell == 5:

self.cells[ri][ci]= 0

for i in range(10):

self.add\_fire()

# обработка состояния вертолёта

def process\_helicopter(self, helico, clouds):

c = self.cells[helico.x][helico.y]

d = clouds.cells[helico.x][helico.y]

if c == 2:

helico.tank = helico.mxtank

if c == 5 and helico.tank > 0:

helico.tank -= 1

helico.score += TREE\_BONUS

self.cells[helico.x][helico.y] = 1

if (c == 4 and helico.score >= UPGRADE\_COST):

helico.mxtank += 1

helico.score -= UPGRADE\_COST

if (c == 3 and helico.score >= LIFE\_COST):

helico.lives += 10

helico.score -= LIFE\_COST

if (d == 2):

helico.lives -= 1

if (helico.lives == 0):

helico.game\_over()

# данные для эксорта сохранения

def export\_data(self):

return{"cells": self.cells}

# восстановление данных из сохранения

def import\_data(self, data):

self.cells = data["cells"] or [[0 for i in range(self.w)] for j in range(self.h)]

3. Файл helicopter.py:

from utils import randcell

import os

# класс Вертолёт

class Helicopter:

# инициализация класса

def \_\_init\_\_(self, w, h):

rc = randcell(w, h)

rx, ry = rc[0], rc[1]

self.h = h

self.w = w

self.tank = 0

self.mxtank = 1

self.score = 0

self.lives = 20

if (rx >= 0 or ry >= 0 or rx < self.h or ry < self.w):

self.x = rx

self.y = ry

# обработка смещения

def move(self, dx, dy):

nx, ny = dx + self.x, dy + self.y

if (nx >= 0 and ny >= 0 and nx < self.h and ny < self.w):

self.x, self.y = nx, ny

# вывод статистики игры

def print\_stats(self):

print("🥡 ", self.tank, '/', self.mxtank, sep="", end=" | ")

print("🏆 ", self.score, end=" | ")

print("💛 ", self.lives)

# конец игры и вывод очков

def game\_over(self):

os.system("cls")

print('XXXXXXXXXXXXXXXXXXXXXXXXXXXXX')

print(' ')

print("GAME OVER, YOUR SCORE IS", self.score)

print(' ')

print('XXXXXXXXXXXXXXXXXXXXXXXXXXXXX')

exit(0)

# экспорт данных

def export\_data(self):

return {"score": self.score,

"lives": self.lives,

"x": self.x, "y": self.y,

"tank": self.tank, "mxtank": self.mxtank}

# импорт данных

def import\_data(self, data):

self.x = data["x"] or 0

self.y = data["y"] or 0

self.tank = data["tank"] or 0

self.mxtank = data["mxtank"] or 1

self.lives = data["lives"] or 3

self.score = data["score"] or 0

4. Файл clouds.py:

from utils import randbool

# класс слоя облаков

class Clouds:

# инициализация класса

def \_\_init\_\_(self, w, h):

self.w = w

self.h = h

self.cells = [[0 for i in range(w)] for j in range(h)]

# обновление слоя

def update(self, r=1, mxr=10, g=1, mxg=10):

for i in range(self.h):

for j in range(self.w):

if randbool(r, mxr):

self.cells[i][j] = 1

if randbool(g, mxg):

self.cells[i][j] = 2

else:

self.cells[i][j] = 0

# подготовка данных на экспорт

def export\_data(self):

return{"cells": self.cells}

# загрузка данных

def import\_data(self, data):

self.cells = data["cells"] or [[0 for i in range(self.w)] for j in range(self.h)]

5. Файл utils.py:

from random import randint as rand

# условная генерация

def randbool(r, mxr):

t = rand(0, mxr)

return (t <= r)

# случайная ячейка игрового поля

def randcell(w,h):

tw = rand(0, w - 1)

th = rand(0, h - 1)

return (th, tw)

def randcell2(x, y):

moves = [(-1, 0), (0, 1), (1, 0), (0, -1)]

# 0 - вверх, 1 - направо, 2 - вниз, 3 - налево

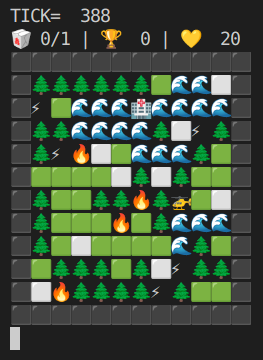
t = rand(0,3)

dx, dy = moves[t][0], moves[t][1]

return (x + dx, y + dy)

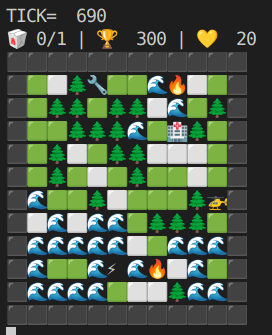
**Испытания игры**

1. После запуска файла main.py формируется игровое поле:

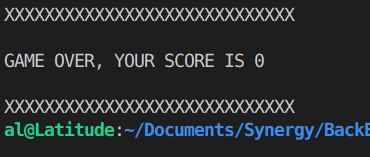


На картинке отображён текущий кадр (TICK=388), статистика по ёмкости бака, жизням, ниже в окантовке представлено игровое поле с лесом, деревьями, водными объектами, горящими деревьями. Отдельным слоем, который находится «выше», нанесены облака.

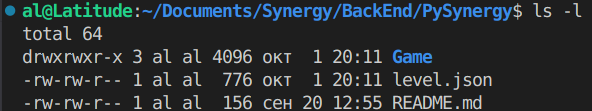
Перемещая «вертолётик» клавишами w-a-s-d пролетаем над водой, наполняем бак, тушим огонь и набираем очки:



Игра заканчивается, когда количество жизней «сердечек» становится равным нулю, выводится соответствующее сообщение:



Нажав клавишу n можно сохранить игру — текущее состояние игры сохраняется в файл level.json вот такого содержания:



{"helicopter": {"score": 0, "lives": 20, "x": 2, "y": 7, "tank": 0, "mxtank": 1}, "map": {"cells": [[2, 2, 0, 1, 0, 0, 1, 0, 1, 0], [2, 1, 1, 1, 1, 1, 1, 0, 0, 1], [2, 2, 1, 1, 1, 1, 0, 1, 1, 4], [2, 2, 1, 1, 1, 1, 1, 0, 1, 1], [2, 2, 2, 1, 1, 1, 0, 1, 1, 1], [2, 2, 0, 1, 1, 3, 1, 1, 1, 0], [1, 1, 1, 1, 1, 0, 1, 1, 4, 1], [1, 0, 1, 1, 0, 1, 1, 0, 0, 0], [0, 2, 1, 2, 1, 1, 0, 1, 1, 1], [1, 2, 1, 1, 1, 0, 1, 1, 0, 0]]}, "clouds": {"cells": [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]}, "tick": 46}

Чтобы восстановить состояние игры нужно нажать клавишу m. Проверим работу восстановления состояния игры и вручную исправим счёт score на 777, количество жизней lives на 888 и восстановим игру, нажав клавишу m:

{"helicopter": {"score": 777, "lives": 888, "x": 2, "y": 7, "tank": 0, "mxtank": 1}, "map": {"cells": [[2, 2, 0, 1, 0, 0, 1, 0, 1, 0], [2, 1, 1, 1, 1, 1, 1, 0, 0, 1], [2, 2, 1, 1, 1, 1, 0, 1, 1, 4], [2, 2, 1, 1, 1, 1, 1, 0, 1, 1], [2, 2, 2, 1, 1, 1, 0, 1, 1, 1], [2, 2, 0, 1, 1, 3, 1, 1, 1, 0], [1, 1, 1, 1, 1, 0, 1, 1, 4, 1], [1, 0, 1, 1, 0, 1, 1, 0, 0, 0], [0, 2, 1, 2, 1, 1, 0, 1, 1, 1], [1, 2, 1, 1, 1, 0, 1, 1, 0, 0]]}, "clouds": {"cells": [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]}, "tick": 46}

Смотрим состояние игрового поля — игра правильно восстановлена из файла!

