Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторным работам №3-4

Выполнил:
студент группы ИУ5-35Б
Серопегин Я.Р.
Подпись и дата:

Проверил:
Гапанюк Ю.Е.
Подпись и дата:

Москва, 2024 г

Описание:
Разработаны программы для работы и создания компьютерных игр, а именно: создан игровой движок и несколько игр на его основе для проверки работоспособности. Данная лабораторная работа была оценена как выполнение нескольких лабораторных работ.

Принцип работы:
Первый файл «app.py» - это игровой движок, созданный с использованием библиотеки pygame, однако полностью упрощён и улучшен, чтобы разработчику игры было легко и комфортно создавать продукт.
Второй файл «game.py» - это тестовая игра, которая использует движок app.py и реализует всю логику игры. В данном случае тестируется управление игроком, а также полностью эмулируется управление автомобилем. При этом происходит расчёт физики и коллизии (столкновений). Создана плавная камера, которая следит за игроком и тайлмап (плитка-фон).

Программа (движок):
```
import pygame
import pygame_gui
import pygame_widgets
from pygame_widgets.slider import Slider
import math as math_

import os
import time as times
import random
import datetime
import webbrowser
import copy

# GlobalValues
Name = ''
AppW = 1366
AppH = 768
FPS = 60
scene = None
copy_scene = None
screen = None
manager = None
clock = None
```

```
event1 = None
glob = {}
copy_glob = {}
app_ico = None

nowscene = 0
time = 0
layout = None
MonW = 0
MonH = 0
getTicksLastFrame = 0
MouseX = 0
MouseY = 0
modeApp = 0
gotoscene = 0
ScrollX = 0
ScrollY = 0
EventsKeys = {}
Sounds = []
uid = 0

def init_sprites(scene_):
    for one_scene in scene_:
        for key in one_scene[1].keys():
            j = one_scene[1][key]
            if j[1] == 'sprite' or j[1] == 'tile':
                if j[13]:
                    for k in range(len(j[3])):
                        for l in range(len(j[3][k])):
                            one_scene[1][key][3][k][l] =
PIL.Image.open(j[3][k][l])
                else:
                    for k in range(len(j[3])):
                        for l in range(len(j[3][k])):
                            one_scene[1][key][3][k][l] =
pygame.image.load(j[3][k][l]).convert_alpha()
    return scene_

def init(Name_, AppW_, AppH_, FPS_, scene_, glob_,
modeApp_=0, icon_=None):
    global Name, AppW, AppH, FPS, scene, copy_scene, screen,
manager, clock, event1, layout, MonW, MonH, glob, copy_glob,
modeApp, app_ico, uid
    Name = Name_
    AppW = AppW_
    AppH = AppH_
    FPS = FPS_
    scene = scene_

    modeApp = modeApp_
    pygame.init()
    app_ico = icon_
    if icon_:
        pygame.display.set_icon(pygame.image.load(icon_))
    pygame.display.set_caption(Name)
    MonW = pygame.display.Info().current_w
    MonH = pygame.display.Info().current_h
    screen = pygame.display.set_mode((AppW, AppH),
pygame.RESIZABLE)
    manager = pygame_gui.UIManager((AppW, AppH),
'theme.json', starting_language='ru')
    clock = pygame.time.Clock()
    clock.tick(FPS)
    pygame.mixer.init()

    event1 = pygame.event.get()
    # Инициализация UI и сцен
    for key in scene[nowscene][1].keys():
        obj = scene[nowscene][1][key]
        if obj[1] == 'ui_button':
            scene[nowscene][1][key][2] =
pygame_gui.elements.UIButton(relative_rect=pygame.Rect((obj[
6], obj[7]), (-1, -1)), text=obj[5], manager=manager)
        elif obj[1] == 'ui_textinput':
            scene[nowscene][1][key][2] =
pygame_gui.elements.UITextEntryLine(relative_rect=pygame.Rec
t((obj[6], obj[7]), (200, -1)), manager=manager)
        elif obj[1] == 'ui_toggle':
            scene[nowscene][1][key][3] = Slider(screen,
obj[4], obj[5], obj[6], obj[7], handleRadius=obj[8])
```

```
    layout = scene[nowscene]
    copy_glob = copy.deepcopy(glob_)
    glob = glob_

    # Инициализация спрайтов
    copy_scene = copy.deepcopy(scene)
    scene = init_sprites(scene)
    uid = len(layout[1])

def update_sprites():
    for key in layout[1].keys():
        loop = layout[1][key]
        if loop[1] == 'sprite':
            posx2 = loop[5]-(loop[7]*loop[9])
            posy2 = loop[6]-(loop[8]*loop[10])
            posx1 = loop[5]+(loop[7]*loop[9])
            posy1 = loop[6]+(loop[8]*loop[10])
            xywh = [loop[5], loop[6], loop[7], loop[8]]
            if modeApp == 1:  # Если нужно выравнивать
картинки по ширине монитора
                xywh[0] = (xywh[0]/AppW)*sys.getappw()
                xywh[1] = (xywh[1]/AppW)*sys.getappw()
                xywh[2] = sys.getappw()*(xywh[2]/AppW)
                xywh[3] = sys.getappw()*(xywh[3]/AppW)
            elif modeApp == 2:  # Если нужно выравнивать
картинки по  монитора
                xywh[0] = (xywh[0]/AppH)*sys.getapph()
                xywh[1] = (xywh[1]/AppH)*sys.getapph()
                xywh[2] = sys.getapph()*(xywh[2]/AppH)
                xywh[3] = sys.getapph()*(xywh[3]/AppH)
            elif modeApp == 3:  # Если нужно выравнивать
картинки заполняемости
                xywh[0] = (xywh[0]/AppW)*sys.getappw()
                xywh[1] = (xywh[1]/AppW)*sys.getappw()
                xywh[2] = sys.getappw()*(xywh[2]/AppW)
                xywh[3] = sys.getappw()*(xywh[3]/AppW)
            img = loop[3][loop[4]][loop[17]]
            if loop[13] == False:
                img = pygame.transform.scale(img, (xywh[2],
xywh[3]))
            layout[1][key][20] = img
```

```
class sys():
    def bg(color):
        return screen.fill(color)
    def fffor(value, start=0, finish=1):
        for value in range(start, finish+1):
            return value
    def exit():
        pygame.quit()
        return
    def getappw():
        return pygame.display.get_surface().get_size()[0]
    def getapph():
        return pygame.display.get_surface().get_size()[1]
    def getmonw():
        return MonW
    def getmonh():
        return MonH
    def goscene(num):
        global gotoscene
        gotoscene = num
    def getscene():
        return nowscene
    def time():
        return time
    def tick():
        return pygame.time.get_ticks()
    def time2():
        return times.time()
    def every(sec):
        sec = sec*1
        df = int(times.time()*1000)
        if (df - (df//sec)*sec) < int(sys.dt()*1000):
            return True
        return False
    def dt():
        return times.time() - getTicksLastFrame
    def setviscursor(val):
        pygame.mouse.set_visible(val)
    def modeapp(mode, vx=AppW, vy=AppH):
        global screen
        if mode == 0:
            pygame.display.quit()
            pygame.init()
            screen = pygame.display.set_mode((AppW, AppH), 
pygame.RESIZABLE)
            curs()
            pygame.display.set_caption(Name)

pygame.display.set_icon(pygame.image.load(app_ico))
            update_sprites()
        elif mode == 1:
            screen = pygame.display.set_mode((MonW, MonH), 
pygame.FULLSCREEN)
            curs()
            update_sprites()
    def rand(s, e):
        return random.randint(s, e)
    def nowtime():
        return str(datetime.datetime.now())
    def fps():
        return clock.get_fps()
    def icon(sprite):
        global app_ico
        app_ico = sprite
        pygame.display.set_icon(pygame.image.load(sprite))
        return
    def restart():
        global scene, layout, ScrollX, ScrollY, glob
        scene = init_sprites(copy.deepcopy(copy_scene))
        glob = copy.deepcopy(copy_glob)
        layout = scene[nowscene]
        ScrollX = 0
        ScrollY = 0
    def scrollx(val):
        global ScrollX
        ScrollX = val
    def scrolly(val):
        global ScrollY
        ScrollY = val
    def getscrollx():
        return ScrollX
    def getscrolly():
        return ScrollY
class sprite():
    global layout
    def setx(name, x):
        layout[1][name][5] = x
        return
    def sety(name, y):
        layout[1][name][6] = y
        return
    def setw(name, w):
        layout[1][name][7] = abs(w)
        layout[1][name][19].add('3-s')
        return
    def seth(name, h):
        layout[1][name][8] = abs(h)
        layout[1][name][19].add('3-s')
        return
    def setangle(name, angle):
        layout[1][name][14] = angle
        layout[1][name][19].add('4-a')
        return
    def setanim(name, num):
        if layout[1][name][4] != num:
            layout[1][name][17] = 0
        layout[1][name][4] = num
        layout[1][name][19].add('2-anim')
        return
    def getx(name):
        return layout[1][name][5]
    def gety(name):
        return layout[1][name][6]
    def getw(name):
        return layout[1][name][7]
    def geth(name):
        return layout[1][name][8]
    def getangle(name):
        return layout[1][name][14]
    def getanim(name):
        return layout[1][name][4]
    def countanim(name):
```

```
    return len(layout[1][name][3])
def setkadr(name, val):
    layout[1][name][17] = int(val)
def getkadr(name):
    return layout[1][name][17]
def setval(name, val, x):
    layout[1][name][12][val] = x
    return
def getval(name, val):
    try:
        return layout[1][name][12][val]
    except:
        return None
def vis(name, val):
    layout[1][name][11] = val
    return
def isvis(name):
    return layout[1][name][11]
def add(name, x, y):
    global uid
    orig_dict = layout[1]
    new_dict = {}
            obj = (layout[1][name]).copy()
    obj[3] = (layout[1][name][3]).copy()
    obj[12] = (layout[1][name][12]).copy()
    obj[19] = copy.deepcopy(layout[1][name][19])
    try:
        obj[20] = (layout[1][name][20]).copy()
    except:
        obj[20] = copy.deepcopy(layout[1][name][20])

    obj[5] = x
    obj[6] = y
    new_key = str(uid)
    uid += 1
    for key in orig_dict.keys():
        if key == name:
            new_dict[new_key] = obj

        new_dict[key] = orig_dict[key]
    layout[1] = new_dict
```

```
            return
        def delete(name):
            layout[1].pop(name)
            for layer in range(len(layout[0])):
                for elem in range(len(layout[0][layer])):
                    if layout[0][layer][elem] == name:
                        del layout[0][layer][elem]
                        return
        def count():
            return len(layout[1])
        def getall():
            return list(layout[1].keys())[1:]
        def alias(name, s):
            layout[1][name][13] = s
        def setop(name, val):
            if val > 255:
                val = 255
            elif val < 0:
                val = 0
            layout[1][name][15] = val
            layout[1][name][19].add('1-o')
        def getop(name):
            opac = layout[1][name][15]
            if opac >= 255:
                return 255
            elif opac <= 0:
                return 0
            return opac
        def collide(obj1, obj2):
            obj1 = layout[1][obj1]
            obj2 = layout[1][obj2]

            col1 = obj1[3][obj1[4]][obj1[17]].get_rect()
            col2 = obj2[3][obj2[4]][obj2[17]].get_rect()
            col1.x = obj1[5]-(obj1[7]*obj1[9])+ScrollX  #
x+(128-128*0.5*2)
            col1.y = obj1[6]-(obj1[8]*obj1[10])+ScrollY
            col1.width = obj1[7]
            col1.height = obj1[8]
            col2.x = obj2[5]-(obj2[7]*obj2[9])+ScrollX  #
x+(128-128*0.5*2)
```

```
        col2.y = obj2[6]-(obj2[8]*obj2[10])+ScrollY
        col2.width = obj2[7]
        col2.height = obj2[8]
        if col1.colliderect(col2):
            return True
        return False
    def oncollide(_obj1, _obj2):
        obj1 = layout[1][_obj1]
        obj2 = layout[1][_obj2]
        col1 = obj1[3][obj1[4]][obj1[17]].get_rect()
        col2 = obj2[3][obj2[4]][obj2[17]].get_rect()
        col1.x = obj1[5]-(obj1[7]*obj1[9])
        col1.y = obj1[6]-(obj1[8]*obj1[10])
        col1.width = obj1[7]
        col1.height = obj1[8]
        col2.x = obj2[5]-(obj2[7]*obj2[9])
        col2.y = obj2[6]-(obj2[8]*obj2[10])
        col2.width = obj2[7]
        col2.height = obj2[8]
        if col1.colliderect(col2):
            if not sprite.getval(_obj1,
'collide_'+str(_obj2)):
                sprite.setval(_obj1, 'collide_'+str(_obj2),
True)
                return True
        else:
            sprite.setval(_obj1, 'collide_'+str(_obj2),
False)
        return False
    def flipx(id):
        layout[1][id][21] = 1
    def flipy(id):
        layout[1][id][21] = 2

    def mouseon(id):
        mx = mouse.x()
        my = mouse.y()
        sx = layout[1][id][5] -
layout[1][id][7]*layout[1][id][9]-ScrollX
        sy = layout[1][id][6] -
layout[1][id][8]*layout[1][id][10]-ScrollY
```

```
        sw = layout[1][id][7]
        sh = layout[1][id][8]
        if mx < sx+sw and mx >= sx and my < sy+sh and my >=
sy:
            return True
        return False
    def setspeed(id, speed):
        layout[1][id][16] = float(speed)
    def effect(id):
        layout[1][id][19].add('5-e')


class key():
    global event1
    def on(name):
        key = pygame.key.get_pressed()
        if key[name]:
            return True
    def down(name):
        if pygame.KEYDOWN in EventsKeys:
            for names in EventsKeys[pygame.KEYDOWN]:
                if names.key == name:
                    return True
        return False
    def up(name):
        if pygame.KEYUP in EventsKeys:
            for key in EventsKeys[pygame.KEYUP]:
                if key == name:
                    return True
        return False
    def click():
        if pygame.KEYDOWN in EventsKeys:
            return True
        return False

class mouse():
    def x():
        if pygame.MOUSEMOTION in EventsKeys:
            global MouseX, MouseY
            MouseX, MouseY =
EventsKeys[pygame.MOUSEMOTION][0].pos
```

```
        if modeApp == 1:
            return MouseX*(AppW/sys.getappw())
        elif modeApp == 2:
            return MouseX*(AppH/sys.getapph())
        else:
            return MouseX
    def y():
        if pygame.MOUSEMOTION in EventsKeys:
            global MouseX, MouseY
            MouseX, MouseY =
EventsKeys[pygame.MOUSEMOTION][0].pos
        if modeApp == 1:
            return MouseY*(AppW/sys.getappw())
        elif modeApp == 2:
            return MouseY*(AppH/sys.getapph())
        else:
            return MouseY
    def nx():
        if pygame.MOUSEMOTION in EventsKeys:
            global MouseX, MouseY
            MouseX, MouseY =
EventsKeys[pygame.MOUSEMOTION][0].pos
        return MouseX
    def ny():
        if pygame.MOUSEMOTION in EventsKeys:
            global MouseX, MouseY
            MouseX, MouseY =
EventsKeys[pygame.MOUSEMOTION][0].pos
        return MouseY
    def pressed(but=0):
        keyd = pygame.mouse.get_pressed()
        return keyd[but]
    def clickdown(but=1):
        if pygame.MOUSEBUTTONDOWN in EventsKeys:
            for key in EventsKeys[pygame.MOUSEBUTTONDOWN]:
                if key.button == but:
                    return True
        return False
    def scroll():
        if pygame.MOUSEWHEEL in EventsKeys:
            return EventsKeys[pygame.MOUSEWHEEL][0].y
```

```
class ui():
    def setx(id, x):
        layout[id][6] = x
    def sety(id, y):
        layout[id][7] = y
    def getx(id):
        return layout[id][6]
    def gety(id):
        return layout[id][7]
    def settext(id, text):
        layout[id][2].set_text(text)
    def setlimit(id, lim):
        layout[id][2].set_text_length_limit(lim)
    def onclick(id):
        if pygame_gui.UI_BUTTON_PRESSED in EventsKeys:
            for elem in
EventsKeys[pygame_gui.UI_BUTTON_PRESSED]:
                if elem.ui_element == layout[id][2]:
                    return True
        return False

class text():
    def setx(id, x):
        layout[1][id][3] = x
    def sety(id, y):
        layout[1][id][4] = y
    def getx(id):
        return layout[1][id][3]
    def gety(id):
        return layout[1][id][4]
    def set(id, text):
        layout[1][id][2] = str(text)
    def get(id):
        return layout[1][id][2]
    def setsize(id, size):
        layout[1][id][5] = size
    def setfont(id, name):
        layout[1][id][8] = False
        layout[1][id][7] = name
    def setmyfont(id, url):
        layout[1][id][8] = True
        layout[1][id][7] = url
    def alias(id, val):
        layout[1][id][10] = val
    def vis(id, val):
        layout[1][id][9] = val
    def isvis(id, val):
        return layout[1][id][9]
    def setcolor(id, val):
        layout[1][id][6] = val
    def getcolor(id):
        return layout[1][id][6]
    def bold(id, val):
        layout[1][id][11] = val
    def italic(id, val):
        layout[1][id][12] = val

class val():
    def set(name, z=''):
        glob[name] = z
    def get(name):
        return glob[name]

class file():
    def write(url, info, cod='utf-8'):
        f = open(url, 'w', encoding=cod)
        f.write(info)
        f.close()
    def read(url, cod='utf-8'):
        f = open(url, 'r', encoding=cod)
        info = f.read()
        f.close()
        return info
    def run(url):
        return os.startfile(url)
    def exist(name):
        return os.path.isfile(name)

class math():
    def sin(x):
        return math_.sin(math_.radians(x))
    def cos(x):
        return math_.cos(math_.radians(x))
    def tg(x):
        return math_.tan(math_.radians(x))
    def ctg(x):
        return 1/math_.tan(math_.radians(x))

class box():
    def setx(id, x):
        layout[id][3] = x
    def sety(id, y):
        layout[id][4] = y
    def setw(id, w):
        layout[id][5] = w
    def seth(id, h):
        layout[id][6] = h
    def setop(id, val):
        layout[id][7] = val
    def getop(id):
        return layout[id][7]

class sound():
    try:
        def init():
            pygame.mixer.music.load(name)
        def play(chan, name, arg=-1):

pygame.mixer.Channel(chan).play(pygame.mixer.Sound(name),
arg)
        def stop(chan):
            pygame.mixer.Channel(chan).stop()
        def volume(chan, val):
            pygame.mixer.Channel(chan).set_volume(val)
    except:
        pass

class music():
    try:
        def load(name):
            pygame.mixer.music.load(name)
        def play(arg=-1):
            pygame.mixer.music.play(arg)
        def stop():
            pygame.mixer.music.stop()
        def volume(val):
            pygame.mixer.music.set_volume(val)
    except:
        pass

class web():
    def open(url):
        return webbrowser.open(str(url))

class snow():
    def vis(id, v):
        layout[id][9] = v

def start():
    global event1, EventsKeys
    event1 = pygame.event.get()
    EventsKeys = {}
    for event in event1:
        if event.type in EventsKeys:
            EventsKeys[event.type].append(event)
        else:
            EventsKeys[event.type] = [event]
        manager.process_events(event)

def gotoscene2(num):
    global nowscene, layout, manager, gotoscene, uid
    nowscene = num
    manager = pygame_gui.UIManager((AppW, AppH),
'theme.json', starting_language='ru')
    for key in scene[nowscene][1].keys():
        obj = scene[nowscene][1][key]
        if obj[1] == 'ui_button':
            scene[nowscene][1][key][2] =
pygame_gui.elements.UIButton(relative_rect=pygame.Rect((obj[
6], obj[7]), (obj[8], obj[9])), text=obj[5],
manager=manager)
        elif obj[1] == 'ui_textinput':
            scene[nowscene][1][key][2] = 
pygame_gui.elements.UITextEntryLine(relative_rect=pygame.Rec
t((obj[6], obj[7]), (obj[8], obj[9])), manager=manager)
        elif obj[1] == 'ui_sliderh':
            scene[nowscene][1][key][2] = 
pygame_gui.elements.UIHorizontalSlider(pygame.Rect((obj[6], 
obj[7]), (obj[8], obj[9])), obj[11], (obj[12], obj[13]), 
click_increment=obj[14], manager=manager)
        elif obj[1] == 'ui_toggle':
            scene[nowscene][1][key][3] = Slider(screen, 
obj[4], obj[5], obj[6], obj[7], handleRadius=obj[8])

    layout = scene[nowscene]
    try:
        layout[1][0][4]()
    except:
        pass
    gotoscene = -1
    update_sprites()
    uid = len(layout[1])+1

import PIL
from PIL import Image

time_update = times.time()
def fixed(value):
    global time_update
    timeget = times.time()
    if timeget-time_update > value/FPS:#0.001:
        time_update = timeget
        return True
    return False

def update():
    global event1, time, getTicksLastFrame
    t = times.time()
    getTicksLastFrame = t
    time += 1

    if pygame.QUIT in EventsKeys:
        pygame.quit()
    if pygame.VIDEORESIZE in EventsKeys:
        pass

    # ОБНОВЛЕНИЕ ЭКРАНА (ИГРОВЫХ ОБЪЕКТОВ И UI ЭЛЕМЕНТОВ)
    for key in layout[1].keys():
        loop = layout[1][key]
        if loop[1] == 'sys':
            if loop[3]:
                sys.bg(loop[2])
        elif loop[1] == 'sprite':
            if loop[11]:
                posx1 = loop[5]-(loop[7])*loop[9]  # x+(128-
128*0.5*2)
                posy1 = loop[6]-(loop[8])*loop[10]
                posx2 = loop[5]+loop[7]-(loop[7])*loop[9]  #
x+(128-128*0.5*2)
                posy2 = loop[6]+loop[8]-(loop[8])*loop[10]
                if posx2-ScrollX <= 0 or posy2-ScrollY <= 0
or posx1-ScrollX >= sys.getappw() or posy1-ScrollY >=
sys.getapph():
                    continue
                xywh = [loop[5] -ScrollX, loop[6] -ScrollY,
loop[7], loop[8]]
                if modeApp == 1:  # Если нужно выравнивать
картинки по ширине монитора
                    xywh[0] = (xywh[0]/AppW)*sys.getappw()
                    xywh[1] = (xywh[1]/AppW)*sys.getappw()
                    xywh[2] = sys.getappw()*(xywh[2]/AppW)
                    xywh[3] = sys.getappw()*(xywh[3]/AppW)
                elif modeApp == 2:  # Если нужно выравнивать
картинки по ширине монитора
                    xywh[0] = (xywh[0]/AppH)*sys.getapph()
                    xywh[1] = (xywh[1]/AppH)*sys.getapph()
                    xywh[2] = sys.getapph()*(xywh[2]/AppH)
                    xywh[3] = sys.getapph()*(xywh[3]/AppH)
                elif modeApp == 3:  # Если нужно выравнивать
картинки по заполнению
                    xywh[0] = (xywh[0]/AppH)*sys.getapph()
                    #print(xywh[0])
                    xywh[1] = (xywh[1]/AppH)*sys.getapph()
                    xywh[2] = sys.getapph()*(xywh[2]/AppH)
                    xywh[3] = sys.getapph()*(xywh[3]/AppH)

                if True:
                    newkadr = loop[17]
                    img = None
                    if loop[16] != 0:
                        if times.time()*100 - loop[18] >
loop[16]:
                            loop[17] += 1
                            loop[18] = times.time()*100
                    else:
                        loop[17] = 0
                    if loop[17] >= len(loop[3][loop[4]]):
                        loop[17] = 0

                    if newkadr != loop[17]:
                        img = loop[3][loop[4]][loop[17]]
                        img = pygame.transform.scale(img,
(xywh[2], xywh[3]))
                        loop[20] = img
                    else:
                        if loop[20]:
                            img = loop[20]
                        else:
                            img = loop[3][loop[4]][loop[17]]
                            img =
pygame.transform.scale(img, (xywh[2], xywh[3]))
                            loop[20] = img

                    if loop[13]:
                        loop7 = xywh[2]
                        loop8 = xywh[3]
                        loop7 = abs(int(loop7))
                        loop8 = abs(int(loop8))
                        loop7 = loop7 if loop7 > 0 else 1
                        loop8 = loop8 if loop8 > 0 else 1
                        img1 = img.resize((loop7, loop8),
resample=Image.BILINEAR).tobytes("raw", 'RGBA')
                        image =
pygame.image.frombuffer(img1, (loop7, loop8), 'RGBA')
```

```
                        image =
pygame.transform.scale(image, (xywh[2], xywh[3]))
                        #Angle
                        image_rect = image.get_rect(topleft
= (xywh[0] - xywh[2]*loop[9], xywh[1] - xywh[3]*loop[10]))
                        offset_center_to_pivot =
pygame.math.Vector2((xywh[0], xywh[1])) - image_rect.center
                        rotated_offset =
offset_center_to_pivot.rotate(-loop[14])
                        rotated_image_center = (xywh[0] -
rotated_offset.x, xywh[1] - rotated_offset.y)
                        img = pygame.transform.rotate(image,
loop[14])
                        rotated_image_rect =
img.get_rect(center = rotated_image_center)

                        screen.blit(img,
(*rotated_image_rect.topleft, *img.get_size()))
                    else:
                        typerend = 0
                        for change in sorted(loop[19]):
                            if change == '2-anim':
                                img =
loop[3][loop[4]][loop[17]]
                                img =
pygame.transform.scale(img, (xywh[2], xywh[3]))
                            if change == '3-s':
                                img =
loop[3][loop[4]][loop[17]]
                                img =
pygame.transform.scale(img, (xywh[2], xywh[3]))
                            elif change == '4-a':
                                loop[20] = img
                                typerend = 1
                                image_rect =
img.get_rect(topleft = (xywh[0] - xywh[2]*loop[9], xywh[1] -
xywh[3]*loop[10]))
                                offset_center_to_pivot =
pygame.math.Vector2((xywh[0], xywh[1])) - image_rect.center
                                rotated_offset =
offset_center_to_pivot.rotate(-loop[14])
```

```
                                rotated_image_center = 
(xywh[0] - rotated_offset.x, xywh[1] - rotated_offset.y)
                                img = 
pygame.transform.rotate(img, loop[14])
                                rotated_image_rect = 
img.get_rect(center = rotated_image_center)
                            if change == '5-e':
                                if not '5-e-ok' in loop[19]:
                                    ccolor = 
'#aaaaaa'#pygame.Color(50)
                                    img0 = 
pygame.Surface(img.get_size()).convert_alpha()
                                    img0.fill(ccolor)
                                    img.blit(img0, (0, 0), 
special_flags = pygame.BLEND_MULT)
                                    loop[19].remove('5-e')
                                    loop[19].add('5-e-ok')
                        #Angle
                        if loop[21] == 1:
                            img = pygame.transform.flip(img, 
True, False)
                            loop[21] = 0
                        elif loop[21] == 2:
                            img = pygame.transform.flip(img, 
False, True)
                            loop[21] = 0
                        img.set_alpha(loop[15])

                        if typerend == 0:
                            loop[20] = img
                            screen.blit(img, (xywh[0] - 
xywh[2]*loop[9], xywh[1] - xywh[3]*loop[10]))
                        if typerend == 1:
                            screen.blit(img, 
(*rotated_image_rect.topleft, *img.get_size()))

        elif loop[1] == 'text':
            if loop[9]:
                xyz = [loop[3], loop[4], loop[5]]
                if modeApp == 1:
                    xyz[0] = (xyz[0]/AppW)*sys.getappw()
```

```
                        xyz[1] = (xyz[1]/AppW)*sys.getappw()
                        xyz[2] = 
int((xyz[2]/AppW)*sys.getappw())
                    elif modeApp == 2:
                        xyz[0] = (xyz[0]/AppH)*sys.getapph()
                        xyz[1] = (xyz[1]/AppH)*sys.getapph()
                        xyz[2] = 
int((xyz[2]/AppH)*sys.getapph())

                    if loop[8]:
                        text_rend = pygame.font.Font(loop[7], 
xyz[2])#.set_italic(False)
                    else:
                        text_rend = pygame.font.SysFont(loop[7], 
xyz[2])#.set_italic(False)
                    if loop[11]:
                        text_rend.set_bold(True)
                    if loop[12]:
                        text_rend.set_italic(True)
                    text_rend_2 = text_rend.render(loop[2], 
loop[10], loop[6])
                    screen.blit(text_rend_2, (xyz[0], xyz[1]))
            elif loop[1] == 'box':
                s = pygame.Surface((loop[5], loop[6]))
                s.set_alpha(loop[8])
                s.fill(loop[7])
                screen.blit(s, (loop[3], loop[4]))
            elif loop[1] == 'snow':
                if loop[9]:
                    if sys.every(loop[6]):
                        pic = pygame.Surface((loop[5], loop[5]))
                        pic.fill(loop[8])
                        loop[10].append([pic, random.randint(10, 
255), random.randint(0, sys.getmonw()), loop[4], 
random.randint(80, loop[7])])
                    for pic in loop[10]:

                        if pic[3] > sys.getmonh():
                            loop[10].pop(loop[10].index(pic))
                        else:
                            pic[0].set_alpha(255-pic[1])
                            pic[1] = abs(math.sin((pic[3] *
    pic[4])/1500))*255
                            pic[3] += sys.dt()*pic[4]
                            screen.blit(pic[0], (pic[2],
    pic[3]))
                else:
                    if len(loop[10]) > 0:
                        for pic in loop[10]:
                            if pic[1] >= 255:

    loop[10].pop(loop[10].index(pic))
                            else:
                                pic[1] += sys.dt()*1000
                                pic[0].set_alpha(255-pic[1])
                                pic[3] += sys.dt()*pic[4]
                                screen.blit(pic[0], (pic[2],
    pic[3]))
            elif loop[1] == 'tile':
                if loop[11]:
                    posx1 = loop[5]-(loop[7])*loop[9]  # x+(128-
    128*0.5*2)
                    posy1 = loop[6]-(loop[8])*loop[10]
                    posx2 = loop[5]+(loop[7])*loop[9]  # x+(128-
    128*0.5*2)
                    posy2 = loop[6]+(loop[8])*loop[10]
                    if posx2-ScrollX <= 0 or posy2-ScrollY <= 0
    or posx1-ScrollX >= sys.getappw() or posy1-ScrollY >=
    sys.getapph():
                        continue
                    xywh = [loop[5] -ScrollX, loop[6] -ScrollY,
    loop[7], loop[8]]
                    if modeApp == 1:  # Если нужно выравнивать
    картинки по ширине монитора
                        xywh[0] = (xywh[0]/AppW)*sys.getappw()
                        xywh[1] = (xywh[1]/AppW)*sys.getappw()
                        xywh[2] = sys.getappw()*(xywh[2]/AppW)
                        xywh[3] = sys.getappw()*(xywh[3]/AppW)
                    elif modeApp == 2:  # Если нужно выравнивать
    картинки по ширине монитора
                        xywh[0] = (xywh[0]/AppH)*sys.getapph()
                        xywh[1] = (xywh[1]/AppH)*sys.getapph()
                        xywh[2] = sys.getapph()*(xywh[2]/AppH)
                        xywh[3] = sys.getapph()*(xywh[3]/AppH)
                    elif modeApp == 3:  # Если нужно выравнивать
    картинки по заполнению
                        xywh[0] = (xywh[0]/AppH)*sys.getapph()
                        #print(xywh[0])
                        xywh[1] = (xywh[1]/AppH)*sys.getapph()
                        xywh[2] = sys.getapph()*(xywh[2]/AppH)
                        xywh[3] = sys.getapph()*(xywh[3]/AppH)
                    if True:
                        newkadr = loop[17]
                        img = None
                        if loop[16] != 0:
                            if times.time()*100 - loop[18] >
    loop[16]:
                                loop[17] += 1
                                loop[18] = times.time()*100
                        else:
                            loop[17] = 0
                        if loop[17] >= len(loop[3][loop[4]]):
                            loop[17] = 0

                        if newkadr != loop[17]:
                            img = loop[3][loop[4]][loop[17]]
                            img = pygame.transform.scale(img,
    (xywh[2], xywh[3]))
                            loop[20] = img
                        else:
                            if loop[20]:
                                img = loop[20]
                            else:
                                img = loop[3][loop[4]][loop[17]]
                                img =
    pygame.transform.scale(img, (xywh[2], xywh[3]))
                                loop[20] = img

                        typerend = 0
                        for change in sorted(loop[19]):
                            if change == '2-anim':
                                img = loop[3][loop[4]][loop[17]]
                                img =
    pygame.transform.scale(img, (xywh[2], xywh[3]))
                            if change == '3-s':
                                img = loop[3][loop[4]][loop[17]]
                                img =
    pygame.transform.scale(img, (xywh[2], xywh[3]))
                            elif change == '4-a':
                                loop[20] = img
                                typerend = 1
                                image_rect =
    img.get_rect(topleft = (xywh[0] - xywh[2]*loop[9], xywh[1] -
    xywh[3]*loop[10]))
                                offset_center_to_pivot =
    pygame.math.Vector2((xywh[0], xywh[1])) - image_rect.center
                                rotated_offset =
    offset_center_to_pivot.rotate(-loop[14])
                                rotated_image_center = (xywh[0]
    - rotated_offset.x, xywh[1] - rotated_offset.y)
                                img =
    pygame.transform.rotate(img, loop[14])
                                rotated_image_rect =
    img.get_rect(center = rotated_image_center)
                            if change == '5-e':
                                if not '5-e-ok' in loop[19]:
                                    ccolor = '#aaaaaa'
                                    img0 =
    pygame.Surface(img.get_size()).convert_alpha()
                                    img0.fill(ccolor)
                                    img.blit(img0, (0, 0),
    special_flags = pygame.BLEND_MULT)
                                    loop[19].remove('5-e')
                                    loop[19].add('5-e-ok')
                        #Angle
                        if loop[21] == 1:
                            img = pygame.transform.flip(img,
    True, False)
                            loop[21] = 0
                        elif loop[21] == 2:
                            img = pygame.transform.flip(img,
    False, True)
                            loop[21] = 0
                        img.set_alpha(loop[15])
                        if typerend == 0:
                            loop[20] = img
                            for x in range(0, loop[9]):
                                for y in range(0, loop[10]):
                                    screen.blit(img,
    (xywh[0]+xywh[2]*x, xywh[1]+xywh[3]*y))
                        if typerend == 1:
                            screen.blit(img,
    (*rotated_image_rect.topleft, *img.get_size()))

        manager.update(clock.tick()/1000)
        manager.draw_ui(screen)

        pygame_widgets.update(event1)

        pygame.display.update()

        if gotoscene != -1:
            gotoscene2(gotoscene)
```

Программа, которая использует движок и демонстрирует работу (сама игра):

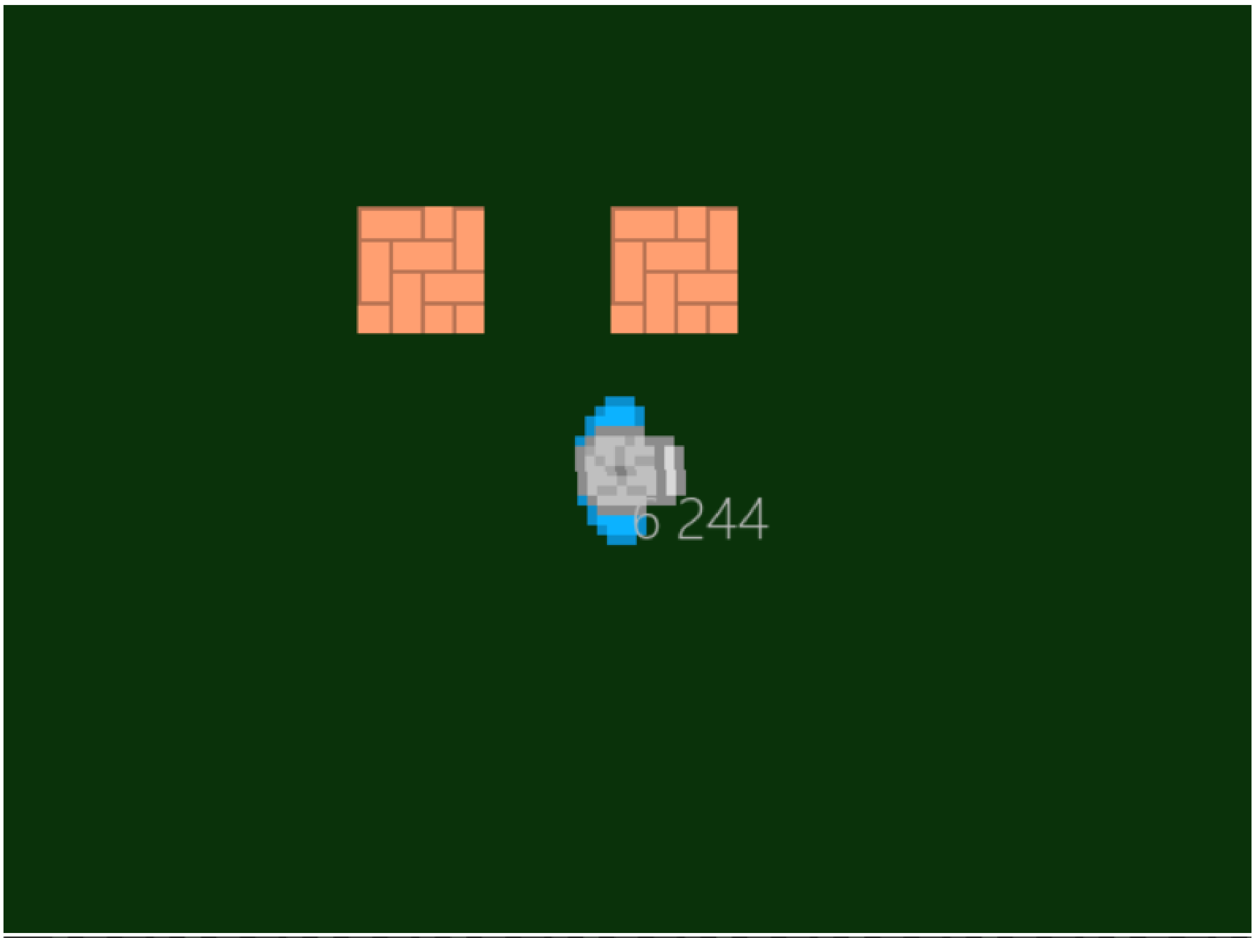
```
import app, pygame
import sys, os

AppW = 640
AppH = 480

def res(relative_path):
    try:
        base_path = sys._MEIPASS + '\\appfiles1'
    except Exception:
        base_path = os.path.abspath(".") + '\\appfiles1'
    return os.path.join(base_path, relative_path)

scene = [
    [
        [],
        {
        '0':[0, 'sys', (10, 50, 10), True],
        '1':[1, 'sprite', 'Name2', [[res('plitka.png')]], 0,
0, 0, 64, 64, 0, 0, True, {}, False, 0, 255, 0, 0, 0, set(),
None, 0],
        '2':[2, 'sprite', 'vilkus', [[res('vilkus.png')],
[res('vilkus1.png')], [res('vilkus.png')],
[res('vilkus2.png')]], 0, AppW//2, AppH//2, 15*5, 11*5, 0.5,
0.5, True, {}, False, 45, 255, 0, 0, 0, set(), None, 0],
        '3':[3, 'text', 'Игрон', AppW//2, AppH//2, 30,
'#bbbbbb', 'Segoe UI', False, True, True, False, False],
        '4':[4, 'snow', '', AppW//2, -10, 10, 100, 100,
'#ffffff', True, []],
        '5':[5, 'sprite', 'Name2', [[res('plitka.png')],
[res('plitka.png')]], 0, 128, 0, 64, 64, 0, 0, True, {},
False, 0, 255, 0, 0, 0, set(), None, 0],
    }],
    [[],
     {
        '0':[0, 'sys', (50, 50, 50), True],
        '3':[2, 'tile', 'Name', [[res('bulijnik.png')]], 0,
0, 0, 17*4, 17*4, 10, 10, True, {}, False, 0, 255, 0, 0, 0,
set(), None, 0],
        '1':[1, 'sprite', 'Name2', [[res('Car-2.png')]], 0,
AppW//2, AppH//2, 43*4, 63*4, 0.5, 0.5, True, {}, False, 0,
255, 0, 0, 0, set(), None, 0],
        '2':[2, 'sprite', 'Name2', [[res('plitka.png')]], 0,
AppW//2+200, AppH//2, 31*4, 31*4, 0.5, 0.5, True, {}, False,
0, 255, 0, 0, 0, set(), None, 0],

    }]
]
```

```
glob = {'set':90, 'speed':0.05, 'x':0, 'y':0, 'trenie':
0.01}

app.init('Nice Jump!', AppW, AppH, 144, scene, glob, 0)

#OnStart()
app.val.set('speed', 0.8)
app.val.set('anim', 0)
app.val.set('rot', 50)
app.val.set('go', 0)
app.val.set('goanim', 0)

def script1():
            full=False
            if app.key.down(pygame.K_F5):
                if full:
                    app.sys.modeapp(0)
                else:
                    app.sys.modeapp(1)
                full = not full
            if app.key.down(pygame.K_1):
                app.sprite.setanim('5', 1)

            if app.key.on(pygame.K_w):
                app.val.set('go', 1)
                if app.key.on(pygame.K_a):
                    app.sprite.setx('2',
app.sprite.getx('2')-app.val.get('speed')*0.7)
                    app.sprite.sety('2',
app.sprite.gety('2')-app.val.get('speed')*0.7)
                    app.val.set('anim', 4)
                elif app.key.on(pygame.K_d):
                    app.sprite.setx('2',
app.sprite.getx('2')+app.val.get('speed')*0.7)
                    app.sprite.sety('2',
app.sprite.gety('2')-app.val.get('speed')*0.7)
                    app.val.set('anim', 5)
                else:
                    app.sprite.sety('2',
app.sprite.gety('2')-app.val.get('speed'))
                    app.val.set('anim', 0)
            if app.key.on(pygame.K_s):
                app.val.set('go', 1)
                if app.key.on(pygame.K_a):
                    app.sprite.setx('2',
app.sprite.getx('2')-app.val.get('speed')*0.7)
                    app.sprite.sety('2',
app.sprite.gety('2')+app.val.get('speed')*0.7)
                    app.val.set('anim', 6)
                elif app.key.on(pygame.K_d):
                    app.sprite.setx('2',
app.sprite.getx('2')+app.val.get('speed')*0.7)
                    app.sprite.sety('2',
app.sprite.gety('2')+app.val.get('speed')*0.7)
                    app.val.set('anim', 7)
                else:
                    app.sprite.sety('2',
app.sprite.gety('2')+app.val.get('speed'))
                    app.val.set('anim', 1)
            if not app.key.on(pygame.K_w) and not
app.key.on(pygame.K_s):
                if app.key.on(pygame.K_a):
                    app.val.set('go', 1)
                    app.sprite.setx('2',
app.sprite.getx('2')-app.val.get('speed'))
                    app.val.set('anim', 2)
                elif app.key.on(pygame.K_d):
                    app.val.set('go', 1)
                    app.sprite.setx('2',
app.sprite.getx('2')+app.val.get('speed'))
                    app.val.set('anim', 3)
                else:
                    app.val.set('go', 0)

            if app.val.get('anim') == 0:
                if app.sprite.getangle('2') > 360-
app.sprite.getangle('2'):
                    app.sprite.setangle('2',
app.sprite.getangle('2')-360)
                app.sprite.setangle('2',
app.sprite.getangle('2')-
(app.sprite.getangle('2'))/app.val.get('rot'))
```

```
            elif app.val.get('anim') == 1:
                app.sprite.setangle('2', 
app.sprite.getangle('2')-(app.sprite.getangle('2')-
180)/app.val.get('rot'))
            elif app.val.get('anim') == 2:
                app.sprite.setangle('2', 
app.sprite.getangle('2')-(app.sprite.getangle('2')-
90)/app.val.get('rot'))
            elif app.val.get('anim') == 3:
                if app.sprite.getangle('2') < 180:
                    app.sprite.setangle('2', 
app.sprite.getangle('2')+360)
                app.sprite.setangle('2', 
app.sprite.getangle('2')-(app.sprite.getangle('2')-
270)/app.val.get('rot'))
            elif app.val.get('anim') == 4:
                app.sprite.setangle('2', 
app.sprite.getangle('2')-(app.sprite.getangle('2')-
45)/app.val.get('rot'))
            elif app.val.get('anim') == 5:
                if app.sprite.getangle('2') < 180:
                    app.sprite.setangle('2', 
app.sprite.getangle('2')+360)
                app.sprite.setangle('2', 
app.sprite.getangle('2')-(app.sprite.getangle('2')-
305)/app.val.get('rot'))
            elif app.val.get('anim') == 6:
                app.sprite.setangle('2', 
app.sprite.getangle('2')-(app.sprite.getangle('2')-
135)/app.val.get('rot'))
            elif app.val.get('anim') == 7:
                app.sprite.setangle('2', 
app.sprite.getangle('2')-(app.sprite.getangle('2')-
225)/app.val.get('rot'))

            if app.val.get('go') == 1:
                app.val.set('goanim', 
app.val.get('goanim')+0.02)
                if int(app.val.get('goanim')) == 4:
                    app.val.set('goanim', 0)
                app.sprite.setanim('2',
int(app.val.get('goanim')))
            else:
                app.sprite.setanim('2', 0)
            if app.key.down(pygame.K_SPACE):
                app.sys.goscene(1)
            if app.key.on(pygame.K_q):
                app.sprite.add('1', app.sprite.getx('2'),
app.sprite.gety('2'))

            app.text.set('3', str(app.sprite.count())+'
'+str(round(app.sys.fps())))#app.sprite.count())
            app.sys.scrollx(app.sprite.getx('2')-
app.sys.getappw()/2)
            app.sys.scrolly(app.sprite.gety('2')-
app.sys.getapph()/2)

def car(id, id2):
    if app.key.on(pygame.K_w):
        app.val.set('speed',
app.val.get('speed')+app.val.get('trenie'))
    elif app.key.on(pygame.K_s):
        app.val.set('speed', app.val.get('speed')-
app.val.get('trenie'))
    else:
        app.val.set('speed', app.val.get('speed')-
app.val.get('speed')/100)
    if app.key.on(pygame.K_a):
        app.sprite.setangle(id,
app.sprite.getangle(id)+0.2*app.val.get('speed'))
    if app.key.on(pygame.K_d):
        app.sprite.setangle(id, app.sprite.getangle(id)-
0.2*app.val.get('speed'))
    if app.val.get('speed') > 2:
        app.val.set('speed', 2)
    if app.val.get('speed') < -1:
        app.val.set('speed', -1)
    app.sprite.setx(id, app.sprite.getx(id)-
app.val.get('speed')*app.math.sin(app.sprite.getangle(id)))
    app.sprite.sety(id, app.sprite.gety(id)-
app.val.get('speed')*app.math.cos(app.sprite.getangle(id)))
```

```
    if app.sprite.oncollide(id, id2):
        app.val.set('speed', -(app.val.get('speed')/3 +
0.1*app.val.get('speed')/abs(app.val.get('speed'))))
        app.sprite.setangle(id2, 2)
def script2():
    car('1', '2')
    app.sys.scrollx(app.sprite.getx('1')-
app.sys.getappw()/2+app.val.get('speed')*20*app.math.sin(app
.sprite.getangle('1')))
    app.sys.scrolly(app.sprite.gety('1')-
app.sys.getapph()/2+app.val.get('speed')*20*app.math.cos(app
.sprite.getangle('1')))


all_scripts = [script1, script2]

while True:
    if app.fixed(0.5):
        app.start()
        all_scripts[app.nowscene]()
        app.update()
```

Результат работы программ (движок + игра):


6 244