МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

ЛУГАНСКОЙ НАРОДНОЙ РЕСПУБЛИКИ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

ЛУГАНСКОЙ НАРОДНОЙ РЕСПУБЛИКИ

«ЛУГАНСКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ

ИМЕНИ ТАРАСА ШЕВЧЕНКО»

Институт физики, математики и информационных технологий

(полное название института / факультета)

Кафедра информационных технологий и систем

(полное название кафедры)

**КУРСОВАЯ РАБОТА**

по Объектно-ориентированному программированию

(название дисциплины)

на тему: «2D игра «Space Invaders»»

Студента 2 курса, группы 2-ПИ

специальности «Программная инженерия»

(код, название без кавычек)

форма освоения ООП очная

(очная, заочная)

Донцова Владимира Витальевича

(фамилия, имя, отчество)

Руководитель:

старший преподаватель Шкандыбин Ю.А.

(должность, ученое звание, научная степень, фамилия и инициалы)

Национальная шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Количество баллов: \_\_\_\_\_\_ Оценка: ECTS \_\_\_\_

Члены комиссии:

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

(подпись) (фамилия и инициалы)

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

(подпись) (фамилия и инициалы)

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

(подпись) (фамилия и инициалы)

Луганск – 2019 год

Оглавление

[ВВЕДЕНИЕ 3](#_Toc41827249)

[**1.** ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 5](#_Toc41827250)

[1.1 Задание и предметная область. 5](#_Toc41827251)

# ВВЕДЕНИЕ

На сегодняшний день игровая индустрия вошла почти в каждый дом, где есть компьютер или смартфон, и доступ в Интернет. Люди играют в видеоигры, когда скучно и убить время, интерактивно развить какие-либо навыки, или же чтобы прославиться в киберспортивных дисциплинах.

История игровой индустрии начинается с 29 ноября 1972 года, когда компания Atari выпустила аркадный автомат Pong, который стал первым коммерчески успешным аркадным автоматом. Игровой процесс представлял собой игру пинг-понг с видом сверху, где было два белых пикселя – игроков, и один белый пиксель – мяч.

Спустя почти два года, 5 ноября 1974 года, компания Kee Games выпустила Pong-подобную игру Tank, где игроки уже не перебрасывают «мячи-пиксели», а выстреливают «пикселями-снарядами» в соперника.

И уже в июне 1978 года компания Taito выпускает аркадный автомат Space Invaders («Космические захватчики»), основоположник современного жанра «Shoot 'em up» («перестреляй их всех»). Игра становится настолько популярной, что вызывает временный дефицит монет номинала, принимаемого автоматом. Цель игры буквально описана названием жанра, игрок управляет лазерной пушкой, передвигая её горизонтально, в нижней части экрана, а также отстреливая инопланетян, надвигающихся сверху экрана. Целью игры является уничтожение пяти рядов по одиннадцать инопланетян, которые двигаются горизонтально, а также вертикально, по направлению к низу экрана. Игрок имеет бесконечное количество патронов. Попадая в инопланетянина, игрок уничтожает его, за что получает очки. При уничтожении инопланетян увеличивается скорость движения оставшихся, а также ускоряется темп звуковых эффектов. При уничтожении всех инопланетян появляется новая, ещё более сильная волна, а игрок получает одно дополнительное очко жизни. Количество новых волн инопланетян неограниченно, что делает игру бесконечной.

Программа курсовой работы построена на языке программирования Python с помощью библиотеки Pygame. В процессе разработки использовалась интегрированная среда разработки PyCharm (1. PyCharm: IDE для профессиональной разработки на Python [Электронный ресурс]. Режим доступа: https://www.jetbrains.com/ru-ru/pycharm/).

Цель: разработать аркадную игру Space Invader на языке программирования Python, опираясь на современные вкусы геймеров.

Задачи:

1. Подготовить пространство, для разработки игры применяя интегрированная среда разработки PyCharm.
2. Написать код на языке программирования Python с помощью библиотеки Pygame.
3. Нарисовать спрайты для текстур и персонажей.

Разделы:

1. В первом разделе описывается постановка задачи разработки игры, описание среды разработки, алгоритмы программы.
2. Во втором разделе применяется практическое решение задач, поставленных в первом разделе, описание функций программы, диаграмма функций.
3. В заключении подводиться вывод о проделанной работе.

# ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

## 1.1 Задание и предметная область.

В основе каждой игры лежит идея оригинальной игры Space Invaders 1978 года, где игровая механика изменена. В противном случае, итог будет прямолинейным клоном, и это служит достаточно большим препятствием для распространения.

Задачей данной курсовой работы было создание аркадной игры, в которой люди могу посоревноваться на наибольше набранное количество очков. Если разбить задачу, на пункты разработки игры, то мы получим следующее:

1. Первый самый важный шаг – определение аудитории проекта. Как правило, определяется возрастная и социальная группы. Небольшие изменения направленности проекта могут повлечь за собой кардинальное изменение потенциальной аудитории игры.
2. Спроектировать игровую механику(геймплей) – цель игры, способы её достижения, препятствия.
3. Определить платформу, на которой будет запускаться игра. Классические десктопы – проверенный, но слегка сужающийся рынок. Мобильные платформы – наиболее перспективные, но в то же время накладываются определенные платформенные ограничения, и нужно быть к ним готовыми. Консоли – еще один классический рынок сбыта для игр, который сейчас снова в зоне роста из-за выхода консолей нового поколения. Как и на весьма ограниченной в техническом плане платформе, в случае выбора консолей нужно также быть готовыми к ряду ограничений. Последний из доступных рынков – браузерные игры, не требующие установки.

Аудитория данного проекта рассчитана на людей малого возраста, а точнее на детей.

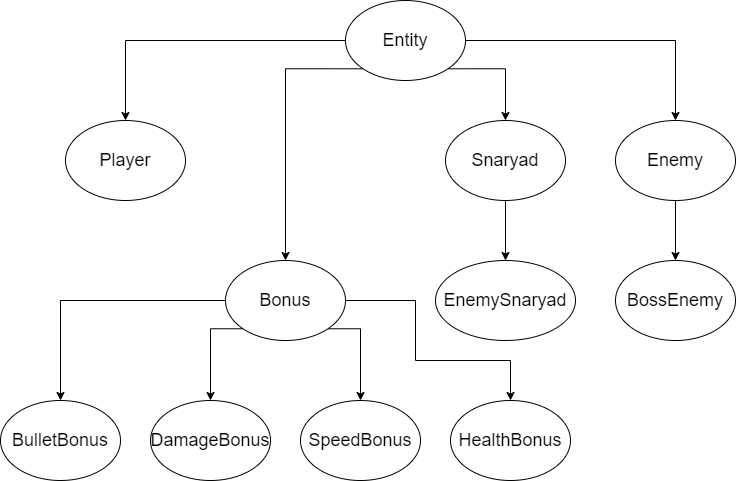
Самый ранний возраст доступа в игру начинается с 6 лет и старше.  
Особенность приложения является то, что оно имеет довольно небольшой размер занимаемого дискового пространства и оптимизировано под современные компьютеры. За основу дизайна приложения взята графика оригинальной игры Space Invaders с небольшими корректировками.

## 1.2 Эскизный проект.

Приложение разработано на языке программирования Python с помощью библиотеки Pygame. Библиотека Pygame используется для создания десктопных 2D приложений. Для взаимодействия пользователю с приложением, достаточно запустить файл game.exe. При загрузке приложения, на полный экран откроется игровое окно, где пользователя встретит главное меню. Для взаимодействия с главным меню пользователь должен использовать клавиши перемещения («стрелки») и «Ввод». Для взаимодействия с игровым процессом, пользователь должен использовать клавиши перемещения (Левая и правая «стрелки») или же клавиши «A» и «D», для выстрела - клавиша «Е», для перезапуска игры – клавиша «F», для вызова меню паузы – клавиша «ESC». Таким образом, пользователь управляет героем самой игры, выполняет игровые цели и старается набрать большее количество очков.

## 1.2 Алгоритмы программы.

Для более грамотного проектирования приложения используется Объектно-ориентированное программирование(ООП).



**Рисунок 1 Иерархия классов**

## 1.3 Алгоритмы программы.

Опишем функции, которые реализуют требования, описанные в эскизном проекте.

Алгоритм - это последовательность команд, предназначенная исполнителю, в результате выполнения которой он должен решить поставленную задачу. Он содержат следующий ряд общих требований:

1. При построении алгоритма прежде всего необходимо задать множество объектов, с которыми будет работать алгоритм. Формализованное (закодированное) представление этих объектов носит название данных. Алгоритм приступает к работе с некоторым набором данных, которые называются входными, и в результате своей работы выдает данные, которые называются выходными. Таким образом, алгоритм преобразует входные данные в выходные.

2. Для работы алгоритма требуется память. В памяти размещаются входные данные, с которыми алгоритм начинает работать, промежуточные данные и выходные данные, которые являются результатом работы алгоритма. Память является дискретной, т.е. состоящей из отдельных ячеек.

3. Алгоритм должен иметь некоторое число входных данных.

4. Алгоритм строится из отдельных шагов (действий, операций, команд). Множество шагов, из которых составлен алгоритм, конечно.

5. После каждого шага необходимо указывать, какой шаг выполняется следующим, либо давать команду остановки.

6. Алгоритм должен завершать работу после конечного числа шагов. При этом необходимо указать, что считать результатом работы алгоритма.

Алгоритм загрузки текстур, персонажей:

1. Загрузка спрайтов из папки
2. Загрузка заднего фона.
3. Загрузка врагов и объединение их в группу
4. Загрузка спрайта героя

Алгоритм размещения врагов на игровом поле:

1. Добавляем врагов в список.
2. Позиционируем врага на игровом поле
3. Добавляем скорость движения
4. Удаляем врага, когда его здоровье оказывается меньше, чем 0.

Алгоритм проигрыша:

1. Удаление игровых элементов с экрана.
2. Очистка списков игровых объектов.
3. Открытие окна с оповещением о проигрыше.

# 2. ПРАКТИЧЕСКОЕ РЕШЕНИЕ

## 2.1 Функции программы

Для написания программы необходимо знание языка программирования Python.Для этого нужно написать функции по описанным в предыдущем разделе алгоритмам.

Функция в программировании — фрагмент программного кода (подпрограмма), к которому можно обратиться из другого места программы. В большинстве случаев с функцией связывается идентификатор, но многие языки допускают и безымянные функции.

Рассмотрим основные функции программы.

Для инициализации всех графических элементов был создан отдельный модуль, где, используются функции библиотеки Pygame.

"""

Resources initialisation

"""

import sys

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

sys.exit()

import pygame

from os import path

pygame.init()

pygame.font.init()

win = pygame.display.set\_mode((1366, 768), pygame.FULLSCREEN)

# инициализируем окошко

ico = pygame.image.load(path.join('sprites/space2.png'))

pygame.display.set\_icon(ico) # ярлык окошка

pygame.display.set\_caption("Space Ivaders") # название окошка

pygame.time.Clock().tick(60) # FPS

win\_w = pygame.display.Info().current\_w # определение ширины экрана

win\_h = pygame.display.Info().current\_h # определение высоты экрана

myfont = pygame.font.SysFont('Comic Sans MS', 15) # малый шрифт (для хп)

bigfont = pygame.font.SysFont('Comic Sans MS', 30) # больший шрифт (для всего)

emptytext = bigfont.render(f'', False, (255, 255, 255)) # инициализация пустого текстового поля

bg = pygame.image.load(path.join('sprites/0.gif')).convert\_alpha()

# инициализация фона и конвертация в правильный формат пикселей

bg = pygame.transform.scale(bg, (win\_w, win\_h)) # растягиваем на весь экран

player\_png = pygame.image.load(path.join('sprites/player.png'))

enemy\_png = pygame.image.load(path.join('sprites/space2.png'))

enemy\_boss\_png = pygame.image.load(path.join('sprites/boss.png'))

bonus\_png = pygame.image.load(path.join('sprites/diam.png'))

bonus\_dmg\_png = pygame.image.load(path.join('sprites/diam\_dmg.png'))

bonus\_speed\_png = pygame.image.load(path.join('sprites/diam\_speed.png'))

bonus\_hp\_png = pygame.image.load(path.join('sprites/diam\_hp.png'))

pygame.mixer.music.load(path.join('sprites/STS.mp3'))

Функция добавления врагов:

def spawn(chislo):

global vragi, level, bosslvl

level += 1

bosslvl = True if level % 5 == 0 else False

if not bosslvl:

for i in range(int(chislo)):

vragi.append(Enemy(i))

else:

vragi.append(BossEnemy())

Обработка проигрыша:

for item in vragi:

if molodec.health <= 0:

vragi.clear()

bullets.clear()

proigral = True

Полный листинг кода расположен в Приложении А.

## 2.2 Интерфейс программы и руководство пользователя.

В результате работы было получено приложение для игры на компьютере.

После запуска программы будет выведено полноэкранное окно, где пользователя встретит главное меню.

В целях более динамичного игрового процесса была изменена механика боя с врагами. В отличие от оригинала, где одновременно на экране может находится только один снаряд, в данной игре это число неограниченно. В следствие этого, для восстановления игрового баланса, у врагов появилось определённое количество здоровья, которое увеличивается экспоненциально с каждым уровнем. Была добавлена механика выпадения бонусов, которые должны увеличить характеристики главного героя.

Для управления героем, пользователь должен использовать клавиши перемещения (Левая и правая «стрелки») или же клавиши «A» и «D», для выстрела - клавиша «Е». С помощью такого управления герой может перемещаться по игровому полю и стрелять снарядами во врагов.

Случайным образом враги выпускают вражеские снаряды. При прикосновении героя к вражескому снаряду у него отнимется одно очко здоровья. Если здоровье главного героя окажется меньше либо рано 0, то игра завершится.

# ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы была получено приложение, которым можно воспользоваться в любой момент через ПК.

Приложения использует игровой процесс старой игры и переосмысливает его, добавляя популярные игровые механики современной игровой индустрии.

Во время выполнения курсовой работы были изучены принципы Объектно-ориентированного программирования в языке программирования Python, способы создания функций, изучены функции работы с файлами, работа со сторонними библиотеками.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Python 3.8.3 documentation [Электронный ресурс]. Режим доступа: https://www.<https://docs.python.org/3/>
2. Pygame v2.0.0.dev7 documentation [Электронный ресурс]. Режим доступа: https://www. pygame.org/docs/
3. Head First Python, 2nd Edition by Paul Barry, 2016
4. Онлайн-курс Complete Python Developer in 2020: Zero to Mastery [Электронный ресурс]. Режим доступа: <https://www.udemy.com/course/complete-python-developer-zero-to-mastery/>

# ПРИЛОЖЕНИЕ А

import random

from colorsys import hsv\_to\_rgb

from pause import \*

from math import e

class Entity: # инициализации главного класса

wight = height = speed = x = y = 0

def move(self, direction):

if direction == 'left':

self.x -= self.speed

else:

self.x += self.speed

class Player(Entity):

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

self.wight = 50

self.height = 30

self.x = 50

self.y = win\_h - self.height - 10

self.speed = 5

self.dmg = 1

self.bullet\_count = 1

self.health = 100

self.player = player\_png

class Enemy(Entity):

alive = True

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

global level

self.sprite = enemy\_png

self.type = "Enemy"

self.x = 5 + 80 \* poryadoc

self.y = 70

self.wight = 50

self.height = 50

self.speed = level % 20

self.go = 'right'

self.health = int(level \* e \*\* 0.5) \* 8

self.hhealth = self.health # неизменяемое здоровье

self.text\_hp = myfont.render(f'{self.health}', False, (255, 255, 255))

def move(self):

def switch():

if self.go == 'right':

self.go = 'left'

else:

self.go = 'right'

self.y += 10

if self.alive:

if self.go == 'right':

self.x += self.speed

else:

self.x -= self.speed

if self.go == "lefty":

pass

elif self.x >= win\_w - self.wight - 5:

switch()

if self.x <= 4:

switch()

elif self.y >= win\_h - self.height - 5:

self.alive = False

return False

return True

class BossEnemy(Enemy):

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

self.sprite = enemy\_boss\_png

self.type = "Boss"

self.wight = 100

self.height = 50

self.x = win\_w + 150

self.y = 70

self.speed = int(level \* 0.7) % 20

self.go = "lefty"

self.health = int(level \* e \*\* 0.6) \* 10

self.hhealth = self.health

class Snaryad(Entity):

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

global color

self.type = 'friendly'

self.x = x

self.y = y

self.radius = 2

self.collor = (hsv2rgb(random.randint(0, 360) / 100, 1, 1)) # color

self.vel = 20

def live(self):

self.y -= self.vel

class EnemySnaryad(Snaryad):

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

Snaryad.\_\_init\_\_(self, x, y)

self.type = 'enemy'

self.radius = 4

self.collor = hsv2rgb(1, 1, 1)

self.vel = -10

class Bonus(Entity):

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

self.speed = 5

self.alive = True

self.x = x\_vraga + 25

self.y = y\_vraga + 50

self.sprite = vid

def move(self, igrok):

self.y += self.speed

if igrok.x < self.x < igrok.x + igrok.wight and igrok.y < self.y < igrok.y + igrok.height:

self.alive = False

self.check(igrok)

def check(self, igrok):

pass

class BulletBonus(Bonus):

def check(self, igrok):

igrok.bullet\_count += 1

class DamageBonus(Bonus):

def check(self, igrok):

igrok.dmg += 1

class SpeedBonus(Bonus):

def check(self, igrok):

if igrok.speed <= 10:

igrok.speed += 1

class HealthBonus(Bonus):

def check(self, player):

player.health += 10

def hsv2rgb(h, s, v):

return tuple(round(i \* 255) for i in hsv\_to\_rgb(h, s, v))

def do\_restart(igrok):

global proigral, level, textsurface, textsurface\_timer, score

score = proigral = level = 0

igrok.bullet\_count = igrok.dmg = 1

textsurface = textsurface\_timer = emptytext

igrok.speed = 5

igrok.health = 100

igrok.x = 50

bullets.clear()

vragi.clear()

def spawn(chislo):

global vragi, level, bosslvl

level += 1

bosslvl = True if level % 5 == 0 else False

if not bosslvl:

for i in range(int(chislo)):

vragi.append(Enemy(i))

else:

vragi.append(BossEnemy())

def drawwindow(score): # прорисовка графики

global start\_ticks, proigral, level, color

textsurface\_timer = text\_score = textsurface = emptytext

win.blit(bg, (0, 0))

pygame.draw.rect(win, (0, 0, 0), (0, 0, win\_w, 45))

text\_lvl = bigfont.render(

f'level: {level} score: {score} Health: {molodec.health} bullets: {molodec.bullet\_count} dmg:{molodec.dmg}'

f' speed: {molodec.speed}', False, (255, 255, 255))

if color == 360:

color = 0

text\_name = bigfont.render(r"\\ Donts\_'s Space Invaders", False, hsv2rgb(color / 100, 1, 1))

color += 1

if not len(vragi) and not proigral:

if not start\_ticks and level:

start\_ticks = pygame.time.get\_ticks()

seconds = (pygame.time.get\_ticks() - start\_ticks) / 1000

textsurface = bigfont.render(f'КРОСАВА, ВСЕХ УБЫВ!!!', False, (255, 255, 255))

textsurface\_timer = bigfont.render(f'Время на отдохнуть: {round(4 - seconds, 3)}', False, (255, 255, 255))

if seconds > 4:

spawn(17)

start\_ticks = 0

textsurface = bigfont.render('', False, (255, 255, 255))

textsurface\_timer = bigfont.render('', False, (255, 255, 255))

if not proigral:

win.blit(molodec.player, (molodec.x, molodec.y))

for item in vragi:

if item.alive:

item.text\_hp = myfont.render(f'{item.health}', False, (255, 255, 255))

pygame.draw.rect(win, (0, 0, 0), (item.x, item.y - 25, item.wight, 25))

pygame.draw.rect(win, hsv2rgb((item.health \* 100 / item.hhealth) / 333, 1, 1), (

item.x + 3, item.y - 23, (item.wight - 5) - (item.wight - (item.wight \* item.health) / item.hhealth),

20))

win.blit(item.sprite, (item.x, item.y))

win.blit(item.text\_hp, ((item.x + item.wight / 3), (item.y - 25)))

for bullet in bullets:

pygame.draw.circle(win, bullet.collor, (bullet.x, bullet.y), bullet.radius)

if len(bonusi):

for item in bonusi:

if item.alive:

win.blit(item.sprite, (item.x, item.y))

item.move(molodec)

if proigral:

dead(score)

do\_restart(molodec)

elif not proigral and len(vragi):

textsurface = bigfont.render('', False, (255, 255, 255))

win.blit(textsurface, (win\_w / 2, 300))

win.blit(textsurface\_timer, (win\_w / 2, 350))

win.blit(text\_lvl, (5, 0))

win.blit(text\_score, (160, 0))

win.blit(text\_name, (win\_w - 410, 0))

pygame.display.update()

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

pygame.mouse.set\_visible(False)

pause()

color = score = start\_ticks = 0 # переменная для таймера

level = 0

bosslvl = False

run = True

proigral = restart = paused = False

vragi = []

bonusi = []

bullets = []

molodec = Player()

music\_play = True

pygame.mixer.music.play(-1)

spawn(17)

while run:

keys = pygame.key.get\_pressed()

if not paused:

for bullet in bullets:

if win\_h > bullet.y > 60:

bullet.live()

for item in vragi:

if item.x < bullet.x < (item.x + item.wight) \

and item.y < bullet.y < (item.y + item.height):

item.health -= molodec.dmg \* molodec.bullet\_count

if item.health <= 0:

if item.type == "Enemy":

score += level

else:

score += level \* 2

item.alive = False

vragi.pop(vragi.index(item))

if random.randint(0, 10) == random.randint(0, 10):

bonusi.append(BulletBonus(item.x, item.y, bonus\_png))

elif random.randint(0, 100) == random.randint(0, 100):

bonusi.append(DamageBonus(item.x, item.y, bonus\_dmg\_png))

elif bosslvl:

bonusi.append(DamageBonus(item.x, item.y, bonus\_dmg\_png))

elif random.randint(0, 110) == random.randint(0, 110):

bonusi.append(HealthBonus(item.x, item.y, bonus\_hp\_png))

elif random.randint(0, 500) == random.randint(0, 500):

bonusi.append(SpeedBonus(item.x, item.y, bonus\_speed\_png))

if bullet in bullets:

bullets.pop(bullets.index(bullet))

if bullet.type == 'enemy' and molodec.x < bullet.x < (molodec.x + molodec.wight) \

and molodec.y < bullet.y < (molodec.y + molodec.height):

molodec.health -= 1

elif bullet in bullets:

bullets.pop(bullets.index(bullet))

if (keys[pygame.K\_LEFT] or keys[pygame.K\_a]) and molodec.x > 5:

molodec.move('left')

elif (keys[pygame.K\_RIGHT] or keys[pygame.K\_d]) and molodec.x < win\_w - molodec.wight - 5:

molodec.move('right')

for item in vragi:

item.move()

if keys[pygame.K\_e] and not proigral:

bullets.append(Snaryad((molodec.x + molodec.wight // 2), molodec.y))

for vrag in vragi:

if len(vragi) != 0 and random.randint(0, 40) == random.randint(0, 35):

bullets.append(EnemySnaryad(vrag.x + vrag.wight//2, vrag.y + 53))

for event in pygame.event.get():

if event.type == pygame.KEYDOWN:

if event.key == pygame.K\_f:

do\_restart(molodec)

elif event.key == pygame.K\_ESCAPE:

pygame.mixer.pause()

pause()

elif event.key == pygame.K\_KP\_PLUS:

level += 1

for item in vragi:

if molodec.health <= 0:

vragi.clear()

bullets.clear()

proigral = True

drawwindow(score)

pygame.quit()

sys.exit()