МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники (ФИТР)

Кафедра программного обеспечения информационных систем и технологий

**КУРСОВАЯ РАБОТА**

по дисциплине: ”Языки программирования”

на тему: ***”***Spaсe invaders***”***

Выполнил**:** ст. гр. 10701320 Ковалёв Н. Д.

Приняла**:** преподаватель Стальцова Е. А.

Минск 2021

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра программного обеспечения информационных систем и технологий

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовой работе**

**по дисциплине**

**«Языки программирования»**

Тема: *«Spaсe invaders» («Космические захватчики»)*

**Исполнитель**: Н. Д. Ковалёв

(подпись)

**Студент 2 курса 10701320 группы**

**Руководитель**: Е. А. Стальцова

(подпись)

Минск 2021

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 6](#_Toc71133898)

[1. Техническое задание на предмет разработки 7](#_Toc71133899)

[2. Обоснование импользуемых языков и технологий 8](#_Toc71133902)

[3. Моделирование и проектирование программного обеспечения 9](#_Toc71133906)

[4. Реализация программного обеспечения 1](#_Toc71133908)3

[5. Развертывание и тестирование программного обеспечения 1](#_Toc71133909)6

[ЗАКЛЮЧЕНИЕ 20](#_Toc71133913)

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 20

ПРИЛОЖЕНИЕ 21

**ВВЕДЕНИЕ**

В ходе выполнения курсовой работы необходимо использовать знания работы с языком программирования Python. Эти знания позволяют выполнять большое количество задач, которые без этих знаний решить невозможно.

Целью данной курсовой работы является закрепление и углубление знаний, полученных при изучении курса «Языки программирования» посредством разработки оконного приложения для компьютера.

**1. Техническое задание на предмет разработки**

# В предмете разработки необходимо организовать игру «Space invaders», целью которой является набор очков путем уничтожения противников

Игру можно разделить на несколько пунктов:

1. Главное меню
2. Игра
3. Меню паузы
4. Экран проигрыша

В главном меню находится кнопка начала игры и кнопка выхода из игры.

Действие игры происходит в космосе. Наша задача - это расстрел пришельцев. Ну, и какой же расстрел пришельцев без оружия. К сожалению, оружие в игре всего одно, но это сделано с учетом общей динамики игры, которая будет только увеличиваться от волны к волне. Инопланетяне тоже могут дать отпор, поэтому стоит уклоняться от их выстрелов или отразить чужой огонь своим, иначе минус одна ценная жизнь. Дабы сохранить такой хрупкий объект перед нашим кораблем находятся четыре крейсера беспилотника, которые берут на себя удар не только со стороны противника, но и с нашей.

Так же во время игры есть возможность поставить её на паузу. Для этого необходимо нажать на клавишу ESC. При этом появляется меню паузы, в котором есть кнопки для продолжения игры, начала её заново, а также выхода из игры.

На экране проигрыша можно начать игру заново или же выйти из нее.

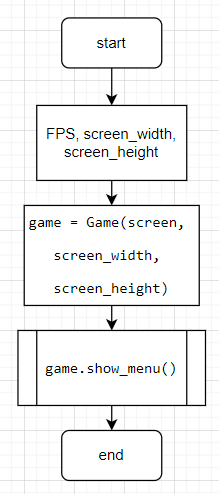
# 2. Обоснование используемых языков и технологий

Для разработки приложения был выбран один из ведущих языков программирования Python, из-за его простоты, лаконичности, логичности, кроссплатформенности и наличия большого количества библиотек.

Также был выбран игровой движок (библиотека) Pygame. Наличия всех необходимых классов и функций, для работы с ними. Например, в этой библиотеке есть классы и функции для работы с изображением, музыкой.

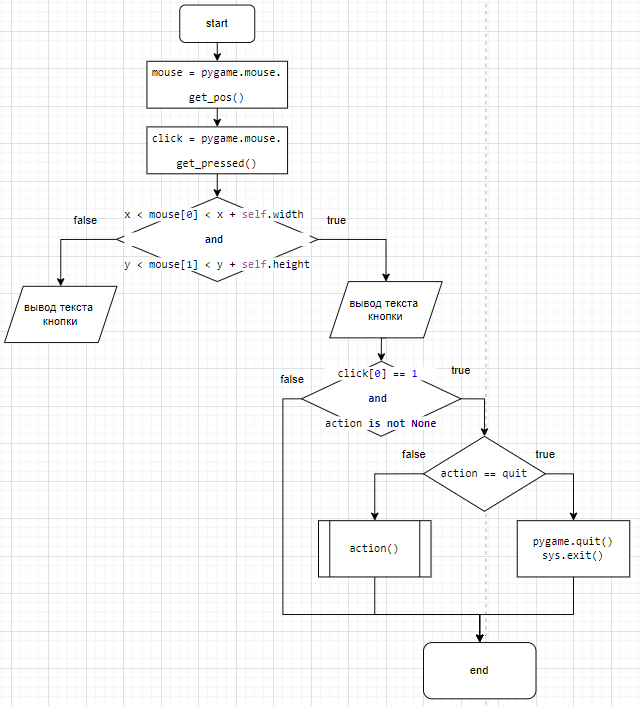
# 3. Моделирование и проектирование программного обеспечения

Запуск игры будет происходить в функции main, алгоритм которой представлен на рисунке 3.1



**Рисунок 3.1** – Блок-схема алгоритма функции main()

Для проверки нажатия кнопки используется функция draw(), алгоритм которой представлен на рисунке 3.2



**Рисунок 3.2** – Блок-схема алгоритма нажатия кнопки

Спрайт игрока и космических захватчиков будут представляться в виде изображения и ректов на рисунках 3.3 и 3.4 соответственно

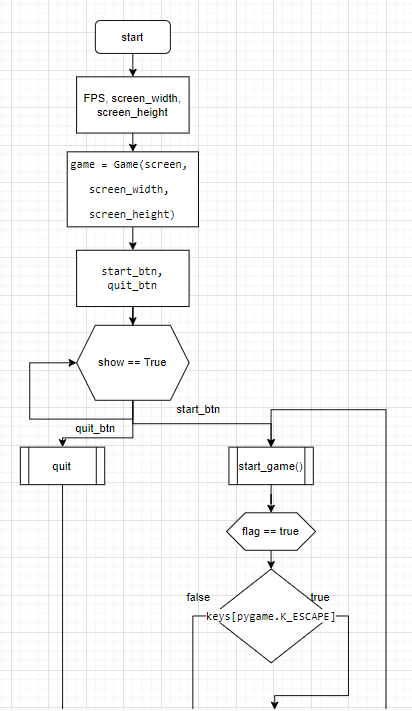
D:\2курс\КУРСОВАЯ_Python\images\player.png

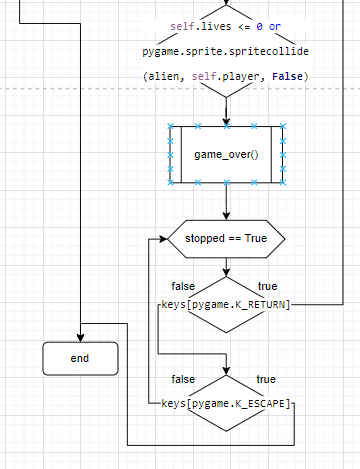
**Рисунок 3.3** – Пример изображения спрайта игрока

D:\2курс\КУРСОВАЯ_Python\images\extra.pngD:\2курс\КУРСОВАЯ_Python\images\green.pngD:\2курс\КУРСОВАЯ_Python\images\red.pngD:\2курс\КУРСОВАЯ_Python\images\yellow.png

**Рисунок 3.4** – Примеры изображения спрайтов космических захватчиков

В алгоритме игры присутствует перемещение главного героя (с помощью клавиш left и right), осуществление выстрела (с помощью клавиши space). Изменение общего счета (зависит от количества убитых захватчиков), происходит покадровая отрисовка главного героя, захватчиков, текста. Алгоритм игры представлен на рисунке 3.5

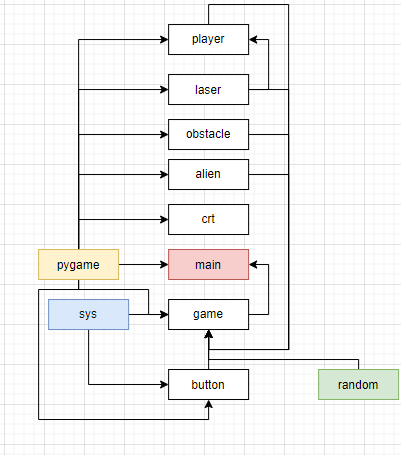




**Рисунок 3.5** – Блок-схема алгоритма игры

**4. Реализация программного обеспечения**

С помощь языка программирования Python и движка pygame было реализовано всё, что было описано выше. В сумме получилось 8 модулей, 1 из которых является модулем для подготовки к запуску программы. Архитектура приложения представлена ниже на рисунке 4.1.



**Рисунок 4.1** – Архитектура приложения

Описание классов:

Game – модуль для работы с игрой:

* create\_obstacle – метод, создающий препятствиe.
* create\_multiple\_obstacles – метод, создающий несколько препятствий.
* alien\_setup – метод, создающий пришельцев и располагающий их на поверхности.
* alien\_position\_checker – метод, управляющий движением пришельцев влево и вправо, а также скоростью их движения.
* аlien\_move\_down – метод, управляющий движением пришельцев вниз.
* аlien\_shoot – метод, отвечающий за стрельбу пришельцев.
* extra\_alien\_timer – метод, отвечающий за появление дополнительного пришельца.
* collision\_check – метод, обработки всех столкновений (лазера игрока с препятствиями и пришельцами; лазера пришельцев с препятствиями и игроком; пришельцев с игроком).
* display\_lives – метод, выводящий количество жизней на экран.
* display\_score – метод, выводящий количество очков на экран.
* pause – метод, осуществляющий паузу в игре.
* continue\_game = метод, продолжающий игру.
* victory\_message – метод выводящий сообщение о победе.
* print\_text – метод для вывода сообщений на экран.
* show\_menu – метод, отображающий главное меню.
* start\_game – метод, начинающий игру.
* game\_over – метод, отображающий экран проигрыша.
* game\_run – основной метод класса, который вызывается в главном цикле и содержащий большинство методов, находящихся выше.

Player – модуль для работы с игроком:

* get\_input – метод, управляющий самим игроком.
* recharge – метод, отвечающий за перезарядку оружия игрока.
* constraint – метод, не позволяющий игроку выйти за пределы экрана.
* shoot\_laser – метод, отвечающий за стрельбу игрока.
* update – метод, обновляющий все действия игрока.

Laser – модуль для работы с лазерами игрока и пришельцев:

* destroy – метод, уничтожающий спрайты лазеров, если они покинули территорию экрана.
* update – метод, обновляющий все действия лазера.

Alien – модуль для работы с пришельцами:

* update – метод, обновляющий все действия пришельцев

Extra – модуль для работы с дополнительным пришельцем:

* метод, обновляющий все действия дополнительного пришельца.

Block – модуль для работы с препятствиями.

Button – модуль для работы с кнопками в игре:

* draw – метод, располагающий кнопку на поверхности.
* print\_text – метод, печатающий текст на экран.

CRT – модуль для создания рамки в игре:

* сreate\_crt\_lines – метод, создающий crt-рамку.
* draw – метод, рисующий crt-рамку.

**5. Развертывание и тестирование программного обеспечения**

Тестирования программного обеспечения:

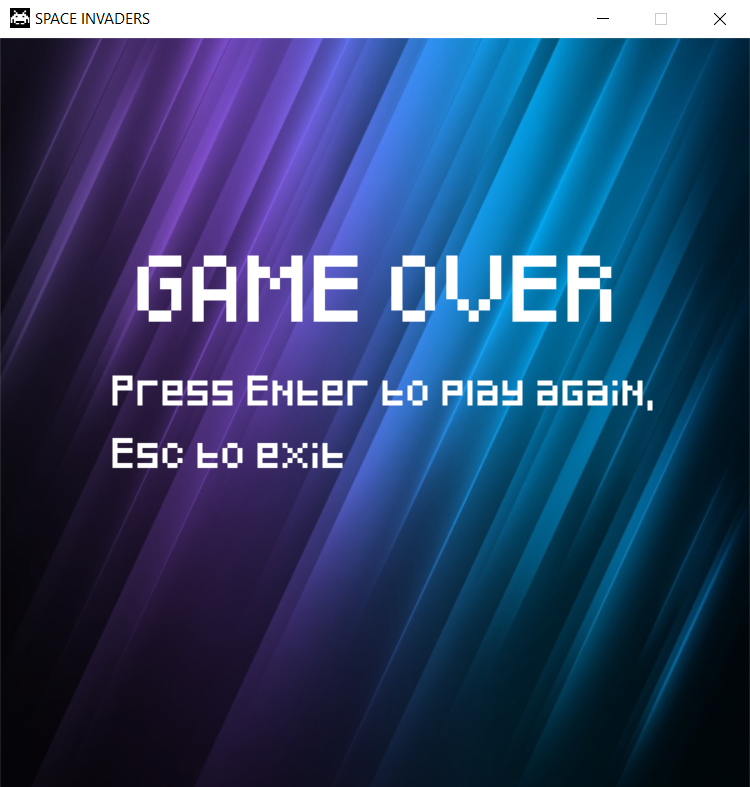
Кнопки главного меню, меню проигрыша, меню паузы были протестированы, они работают без багов. Примеры меню были продемонстрированы на рисунках 5.1, 5.2, 5.3.



**Рисунок 5.1** – Главное меню



**Рисунок 5.2** – Меню паузы



**Рисунок 5.3** – Меню проигрыша

Были успешно протестированы перемещение игрока, все коллизии, а также все случаи проигрышей. Демонстрация игры приведена на рисунке 5.4.



**Рисунок 5.4** – Демонстрация игры

Вывод тестирования: при тестировании игры багов замечено не было. Приложение является работоспособным.

**ЗАКЛЮЧЕНИЕ**

В ходе данной курсовой работы была написана игра «Space invaders» на языке программирования Python с использованием библиотеки Pygame, а также изучена специальная документация по программированию на языке Python, закреплены и углублены знания, полученные при изучении курса «Языки программирования».

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1 Python. PyGame and Raspberry Pi Game Development (2016)

2 Beginning Python Games Development, With PyGame, 2-е издание (2015)

3. Program Arcade Games: With Python and Pygame, 4-е издание (2016)

4. К вершинам мастерства. Лучано Рамальо (2015)

**ПРИЛОЖЕНИЕ Листинг исходного кода**

**Файл alien.py**

**import** pygame  
  
**class** Alien(pygame.sprite.Sprite):  
 **def** \_\_init\_\_(self, color, x, y):  
 super().\_\_init\_\_()  
 file\_path = **'../images/'** + color + **'.png'** *#file\_path = '../images/artemp2.jpg'* self.image = pygame.image.load(file\_path).convert\_alpha()  
 self.rect = self.image.get\_rect(topleft=(x, y))  
  
 **if** color == **'red'**:  
 self.value = 100  
 **elif** color == **'green'**:  
 self.value = 200  
 **else**:  
 self.value = 300  
  
 **def** update(self, direction):  
 self.rect.x += direction  
  
**class** Extra(pygame.sprite.Sprite):  
 **def** \_\_init\_\_(self, side, screen\_width):  
 super().\_\_init\_\_()  
 self.image = pygame.image.load(**'../images/extra.png'**).convert\_alpha()  
  
 **if** side == **'right'**:  
 x = screen\_width + 50  
 self.speed = -3  
 **else**:  
 x = -50  
 self.speed = 3  
  
 self.rect = self.image.get\_rect(topleft = (x, 80))  
  
 **def** update(self):  
 self.rect.x += self.speed

**Файл game.py**

**import** pygame, sys  
  
**from** player **import** Player  
**import** obstacle  
**from** alien **import** Alien, Extra  
**from** random **import** choice  
**from** random **import** randint  
**from** laser **import** Laser  
**from** button **import** Button  
  
  
  
**class** Game:  
 **def** \_\_init\_\_(self, screen, width, height):  
 *# main surface* self.screen = screen  
 self.screen\_width = width  
 self.screen\_height = height  
  
 *# Player setup* player\_sprite = Player((self.screen\_width / 2, self.screen\_height), self.screen\_width, 5)  
 self.player = pygame.sprite.GroupSingle(player\_sprite)  
  
 *# health and score setup* self.lives = 3  
 self.live\_surf = pygame.image.load(**'../images/player.png'**).convert\_alpha()  
 self.live\_x\_start\_pos = self.screen\_width - (self.live\_surf.get\_size()[0] \* 2 + 20)  
 self.score = 0  
 self.font = pygame.font.Font(**'../fonts/Pixeled.ttf'**, 20)  
  
 *# game setup* self.FPS = 60  
 self.clock = pygame.time.Clock()  
  
 *# obstacle setup* self.shape = obstacle.shape  
 self.block\_size = 6  
 self.blocks = pygame.sprite.Group()  
 self.obstacle\_amount = 4  
 self.obstacle\_x\_positions = [num \* (self.screen\_width / self.obstacle\_amount) **for** num **in** range(self.obstacle\_amount)]  
 self.create\_multiple\_obstacles(\*self.obstacle\_x\_positions, x\_start=self.screen\_width / 15, y\_start=480)  
  
 *# alien setup* self.aliens = pygame.sprite.Group()  
 self.alien\_lasers = pygame.sprite.Group()  
 self.alien\_setup(rows=6, cols=8)  
 self.alien\_direction = 1  
  
 *# Extra setup* self.extra = pygame.sprite.GroupSingle()  
 self.extra\_spawn\_time = randint(40, 80)  
  
 *# Audio* self.laser\_sound = pygame.mixer.Sound(**'../audio/audio\_laser.wav'**)  
 self.laser\_sound.set\_volume(0.1)  
 self.explosion\_sound = pygame.mixer.Sound(**'../audio/audio\_explosion.wav'**)  
 self.explosion\_sound.set\_volume(1.5)  
 self.music = pygame.mixer.Sound(**'../audio/music.wav'**)  
  
 *# Menu* self.menu\_background = pygame.image.load(**'../images/gradient.jpg'**)  
  
 self.paused = **False  
  
 def** create\_obstacle(self, x\_start, y\_start, offset\_x):  
 **for** row\_index, row **in** enumerate(self.shape):  
 **for** col\_index, col **in** enumerate(row):  
 **if** col == **'x'**:  
 x = x\_start + col\_index \* self.block\_size + offset\_x  
 y = y\_start + row\_index \* self.block\_size  
 block = obstacle.Block(self.block\_size, (241, 79, 80), x, y)  
 self.blocks.add(block)  
  
 **def** create\_multiple\_obstacles(self, \*offset, x\_start, y\_start):  
 **for** offset\_x **in** offset:  
 self.create\_obstacle(x\_start, y\_start, offset\_x)  
  
 **def** alien\_setup(self, rows, cols, x\_distance=60, y\_distance=48, x\_offset=70, y\_offset=100):  
 **for** row\_index, row **in** enumerate(range(rows)):  
 **for** col\_index, col **in** enumerate(range(cols)):  
 x = col\_index \* x\_distance + x\_offset  
 y = row\_index \* y\_distance + y\_offset  
  
 **if** row\_index == 0:  
 alien\_sprite = Alien(**'yellow'**, x, y)  
 **elif** 1 <= row\_index <= 2:  
 alien\_sprite = Alien(**'green'**, x, y)  
 **else**:  
 alien\_sprite = Alien(**'red'**, x, y)  
 self.aliens.add(alien\_sprite)  
  
 **def** alien\_position\_checker(self):  
 all\_aliens = self.aliens.sprites()  
 **for** alien **in** all\_aliens:  
 count\_aliens = len(all\_aliens)  
 **if** count\_aliens < 25:  
 **if** alien.rect.right >= self.screen\_width:  
 self.alien\_direction = -2  
 self.alien\_move\_down(2)  
 **elif** alien.rect.left <= 0:  
 self.alien\_direction = 2  
 self.alien\_move\_down(2)  
 **elif** count\_aliens < 11:  
 **if** alien.rect.right >= self.screen\_width:  
 self.alien\_direction = -5  
 self.alien\_move\_down(2)  
 **elif** alien.rect.left <= 0:  
 self.alien\_direction = 5  
 self.alien\_move\_down(2)  
 **elif** count\_aliens < 6:  
 **if** alien.rect.right >= self.screen\_width:  
 self.alien\_direction = -10  
 self.alien\_move\_down(2)  
 **elif** alien.rect.left <= 0:  
 self.alien\_direction = 10  
 self.alien\_move\_down(2)  
 **else**:  
 **if** alien.rect.right >= self.screen\_width:  
 self.alien\_direction = -1  
 self.alien\_move\_down(2)  
 **elif** alien.rect.left <= 0:  
 self.alien\_direction = 1  
 self.alien\_move\_down(2)  
  
 **def** alien\_move\_down(self, distance):  
 **if** self.aliens:  
 **for** alien **in** self.aliens.sprites():  
 alien.rect.y += distance  
  
 **def** alien\_shoot(self):  
 **if** self.aliens.sprites():  
 random\_alien = choice(self.aliens.sprites())  
 laser\_sprite = Laser(random\_alien.rect.center, self.screen\_height, 6)  
 self.alien\_lasers.add(laser\_sprite)  
 self.laser\_sound.play()  
  
 **def** extra\_alien\_timer(self):  
 self.extra\_spawn\_time -= 1  
 **if** self.extra\_spawn\_time <= 0:  
 self.extra.add(Extra(choice([**'right'**, **'left'**]), self.screen\_width))  
 self.extra\_spawn\_time = randint(400, 800)  
  
 **def** collision\_check(self):  
  
 *# player lasers* **if** self.player.sprite.lasers:  
 **for** laser **in** self.player.sprite.lasers:  
 *# obstacle collisions* **if** pygame.sprite.spritecollide(laser, self.blocks, **True**):  
 laser.kill()  
  
 *# alien collisions* aliens\_hit = pygame.sprite.spritecollide(laser, self.aliens, **True**)  
 **if** aliens\_hit:  
 **for** alien **in** aliens\_hit:  
 self.score += alien.value  
 laser.kill()  
 self.explosion\_sound.play()  
  
 *# extra collision* **if** pygame.sprite.spritecollide(laser, self.extra, **True**):  
 self.score += 500  
 laser.kill()  
  
 *# alien lasers* **if** self.alien\_lasers:  
 **for** laser **in** self.alien\_lasers:  
 *# obstacle collisions* **if** pygame.sprite.spritecollide(laser, self.blocks, **True**):  
 laser.kill()  
  
 **if** pygame.sprite.spritecollide(laser, self.player, **False**):  
 laser.kill()  
 self.lives -= 1  
 **if** self.lives <= 0:  
 **if** (self.game\_over()):  
 self.start\_game()  
 **else**:  
 pygame.quit()  
 sys.exit()  
 *# pygame.quit()  
 # sys.exit()  
  
 # aliens* **if** self.aliens:  
 **for** alien **in** self.aliens:  
 pygame.sprite.spritecollide(alien, self.blocks, **True**)  
  
 **if** pygame.sprite.spritecollide(alien, self.player, **False**):  
 **if** (self.game\_over()):  
 self.start\_game()  
 **else**:  
 pygame.quit()  
 sys.exit()  
  
 **def** display\_lives(self):  
 **for** live **in** range(self.lives - 1):  
 x = self.live\_x\_start\_pos + (live \* (self.live\_surf.get\_size()[0] + 10))  
 self.screen.blit(self.live\_surf, (x, 8))  
  
 **def** display\_score(self):  
 score\_surf = self.font.render(**f'score: {**self.score**}'**, **False**, **'white'**)  
 score\_rect = score\_surf.get\_rect(topleft=(10, -10))  
 self.screen.blit(score\_surf, score\_rect)  
  
 **def** pause(self):  
 pygame.mixer.pause()  
  
 continue\_btn = Button(222, 56, self.screen)  
 restart\_btn = Button(218, 56, self.screen)  
 exit\_btn = Button(108, 56, self.screen)  
  
 self.paused = **True  
 while** self.paused:  
 **for** event **in** pygame.event.get():  
 **if** event.type == pygame.QUIT:  
 pygame.quit()  
 sys.exit()  
  
 self.print\_text(**"Pause"**, 165, 20, 52)  
  
 pygame.display.update()  
 self.clock.tick(self.FPS)  
  
 self.screen.fill((30, 30, 30))  
  
 continue\_btn.draw(178, 180, **'CONTINUE'**, self.continue\_game)  
 restart\_btn.draw(180, 260, **'RESTART'**, self.start\_game)  
 exit\_btn.draw(235, 340, **'EXIT'**, quit)  
  
 pygame.mixer.unpause()  
  
 **def** continue\_game(self):  
 self.paused = **False  
  
 def** victory\_message(self):  
 **if not** self.aliens.sprites():  
 self.print\_text(**'You won'**, self.screen\_width / 2 - 50, self.screen\_height / 2, 20)  
  
  
 **def** print\_text(self, message, x, y, font\_size=20, font\_color=(255, 255, 255), font\_type=**'../fonts/Pixeled.ttf'**):  
 font\_type = pygame.font.Font(font\_type, font\_size)  
 text = font\_type.render(message, **False**, font\_color)  
 self.screen.blit(text, (x, y))  
  
 **def** show\_menu(self):  
  
 menu\_background = pygame.image.load(**'../images/gradient.jpg'**)  
  
 start\_btn = Button(295, 56, self.screen)  
 quit\_btn = Button(108, 56, self.screen)  
  
 show = **True  
 while** show:  
 **for** event **in** pygame.event.get():  
 **if** event.type == pygame.QUIT:  
 pygame.quit()  
 sys.exit()  
  
 self.screen.blit(menu\_background, (0, 0))  
 *# screen.fill((0, 0, 0))* start\_btn.draw(150, 150, **'START GAME'**, self.start\_game)  
 quit\_btn.draw(245, 260, **'QUIT'**, quit)  
  
  
 pygame.display.update()  
 self.clock.tick(self.FPS)  
  
 **def** start\_game(self):  
 game = Game(self.screen, self.screen\_width, self.screen\_height)  
  
 *# music = pygame.mixer.Sound('../audio/music.wav')* self.music.set\_volume(0.05)  
 self.music.play(loops=-1)  
  
 ALIENLASER = pygame.USEREVENT + 1  
 pygame.time.set\_timer(ALIENLASER, 800)  
  
 flag = **True  
 while** flag:  
 **for** event **in** pygame.event.get():  
 **if** event.type == pygame.QUIT:  
 pygame.quit()  
 sys.exit()  
 **if** event.type == ALIENLASER:  
 game.alien\_shoot()  
  
 keys = pygame.key.get\_pressed()  
 **if** keys[pygame.K\_ESCAPE]:  
 self.pause()  
  
 *# screen.blit(self.menu\_background, (0, 0))* self.screen.fill((30, 30, 30))  
 game.run()  
 *# crt.draw()* pygame.display.update()  
 self.clock.tick(self.FPS)  
  
 **def** game\_over(self):  
 pygame.mixer.pause()  
 **if** self.lives == 0:  
 stopped = **True  
 while** stopped:  
 **for** event **in** pygame.event.get():  
 **if** event.type == pygame.QUIT:  
 pygame.quit()  
 sys.exit()  
  
 self.screen.blit(self.menu\_background, (0, 0))  
 self.print\_text(**'GAME OVER'**, 110, 130, 44)  
 self.print\_text(**'Press Enter to play again,'**, 90, 250)  
 self.print\_text(**'Esc to exit'**, 90, 300)  
  
 keys = pygame.key.get\_pressed()  
 **if** keys[pygame.K\_RETURN]:  
 **return True  
 if** keys[pygame.K\_ESCAPE]:  
 **return False** pygame.display.update()  
 self.clock.tick(self.FPS)  
  
 **def** run(self):  
 self.player.update()  
 self.alien\_lasers.update()  
 self.extra.update()  
  
 self.aliens.update(self.alien\_direction)  
 self.alien\_position\_checker()  
 self.extra\_alien\_timer()  
 self.collision\_check()  
  
 self.player.sprite.lasers.draw(self.screen)  
 self.player.draw(self.screen)  
 self.blocks.draw(self.screen)  
 self.aliens.draw(self.screen)  
 self.alien\_lasers.draw(self.screen)  
 self.extra.draw(self.screen)  
 self.display\_lives()  
 self.display\_score()  
  
 self.victory\_message()

**Файл player.py**

**import** pygame  
**from** laser **import** Laser  
  
**class** Player(pygame.sprite.Sprite):  
 **def** \_\_init\_\_(self, pos, constraint, speed):  
 super().\_\_init\_\_()  
 self.image = pygame.image.load(**'../images/player.png'**).convert\_alpha()  
 self.rect = self.image.get\_rect(midbottom = pos)  
 self.speed = speed  
 self.max\_x\_constraint = constraint  
 self.ready = **True** self.laser\_time = 0  
 self.laser\_cooldown = 600  
  
 self.lasers = pygame.sprite.Group()  
 self.laser\_sound = pygame.mixer.Sound(**'../audio/audio\_laser.wav'**)  
 self.laser\_sound.set\_volume(0.3)  
  
  
 **def** get\_input(self):  
 keys = pygame.key.get\_pressed()  
  
 **if** keys[pygame.K\_RIGHT]:  
 self.rect.x += self.speed  
 **if** keys[pygame.K\_LEFT]:  
 self.rect.x -= self.speed  
  
 **if** keys[pygame.K\_SPACE] **and** self.ready:  
 self.shoot\_laser()  
 self.ready = **False** self.laser\_time = pygame.time.get\_ticks()  
 self.laser\_sound.play()  
  
  
 **def** recharge(self):  
 **if not** self.ready:  
 current\_time = pygame.time.get\_ticks()  
 **if** current\_time - self.laser\_time >= self.laser\_cooldown:  
 self.ready = **True  
  
 def** constraint(self):  
 **if** self.rect.left <= 0:  
 self.rect.left = 0  
 **if** self.rect.right >= self.max\_x\_constraint:  
 self.rect.right = self.max\_x\_constraint  
  
 **def** shoot\_laser(self):  
 self.lasers.add(Laser(self.rect.center, self.rect.bottom, -8))  
  
 **def** update(self):  
 self.get\_input()  
 self.constraint()  
 self.recharge()  
 self.lasers.update()

**Файл laser.py**

**import** pygame  
  
**class** Laser(pygame.sprite.Sprite):  
 **def** \_\_init\_\_(self, pos, screen\_height, speed = -8):  
 super().\_\_init\_\_()  
 self.image = pygame.Surface((4,20))  
 self.image.fill(**'white'**)  
 self.rect = self.image.get\_rect(center = pos)  
 self.speed = speed  
 self.height\_y\_constraint = screen\_height  
  
 **def** destroy(self):  
 **if** self.rect.y <= -50 **or** self.rect.y >= self.height\_y\_constraint + 50:  
 self.kill()  
  
 **def** update(self):  
 self.rect.y += self.speed  
 self.destroy()

**Файл obstacle.py**

**import** pygame  
  
**class** Block(pygame.sprite.Sprite):  
 **def** \_\_init\_\_(self, size, color, x, y):  
 super().\_\_init\_\_()  
 self.image = pygame.Surface((size, size))  
 self.image.fill(color)  
 self.rect = self.image.get\_rect(topleft = (x, y))  
  
shape = [  
**' xxxxxxx '**,  
**' xxxxxxxxx '**,  
**'xxxxxxxxxxx'**,  
**'xxxxxxxxxxx'**,  
**'xxxxxxxxxxx'**,  
**'xxx xxx'**,  
**'xx xx'**]

**Файл button.py**

**import** pygame  
**import** sys  
  
**class** Button:  
 **def** \_\_init\_\_(self, width, height, screen):  
 self.width = width  
 self.height = height  
 self.screen = screen  
 self.inactive\_color = (255, 215, 0, 0) *# дефолтный цвет* self.active\_color = (255, 255, 0, 0) *# цвет при наведении* self.button\_sound = pygame.mixer.Sound(**'../audio/button.wav'**)  
 self.font = pygame.font.Font(**'../fonts/Pixeled.ttf'**, 20)  
  
 **def** draw(self, x, y, message, action = **None**, font\_size = 30):  
 mouse = pygame.mouse.get\_pos()  
 click = pygame.mouse.get\_pressed()  
  
 **if** x < mouse[0] < x + self.width **and** y < mouse[1] < y + self.height:  
 *#pygame.draw.rect(self.screen, self.active\_color, (x, y, self.width, self.height))* self.print\_text(message=message, x=x, y=y - 20, font\_size=font\_size, font\_color = (255, 255, 255))  
 **if** click[0] == 1 **and** action **is not None**:  
 pygame.mixer.Sound.play(self.button\_sound)  
 pygame.time.delay(300)  
 **if** action == quit:  
 pygame.quit()  
 sys.exit()  
 **else**:  
 action()  
 **else**:  
 **pass** *#pygame.draw.rect(self.screen, self.inactive\_color, (x, y, self.width, self.height))* self.print\_text(message = message, x = x, y = y - 20, font\_size = font\_size)  
  
 **def** print\_text(self, message, x, y, font\_size = 20, font\_color = (255, 0, 0), font\_type = **'../fonts/Pixeled.ttf'**):  
 font\_type = pygame.font.Font(font\_type, font\_size)  
 text = font\_type.render(message, **True**, font\_color)  
 self.screen.blit(text, (x, y))

**Файл crt.py**

**import** pygame  
  
**class** CRT:  
 **def** \_\_init\_\_(self, screen, s\_w, s\_h):  
 self.screen = screen  
 self.screen\_width = s\_w  
 self.screen\_height = s\_h  
 self.tv = pygame.image.load(**'../images/tv.png'**).convert\_alpha()  
 self.tv = pygame.transform.scale(self.tv, (self.screen\_width, self.screen\_height))  
  
 **def** create\_crt\_lines(self):  
 line\_height = 3  
 line\_amount = int(self.screen\_height / line\_height)  
 **for** line **in** range(line\_amount):  
 y\_pos = line \* line\_height  
 pygame.draw.line(self.tv, **'black'**, (0, y\_pos), (self.screen\_width, y\_pos), 1)  
  
 **def** draw(self):  
 self.tv.set\_alpha(60)  
 self.create\_crt\_lines()  
 self.screen.blit(self.tv, (0,0))

**Файл main.py**

**import** pygame  
  
**from** game **import** Game  
  
**def** main():  
 pygame.init()  
 FPS = 60  
 screen\_width = 600  
 screen\_height = 600  
 screen = pygame.display.set\_mode((screen\_width, screen\_height))  
 pygame.display.set\_caption(**"SPACE INVADERS"**)  
 pygame.display.set\_icon(pygame.image.load(**"../images/icon2.png"**))  
  
 game = Game(screen, screen\_width, screen\_height)  
 game.show\_menu()  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 main()