

«Space invaders»

Формулировка задания

Реализовать классическую игру « **Space invaders** »

Постановка задачи

Реализовать алгоритм игры « **Space invaders** »

Содержание

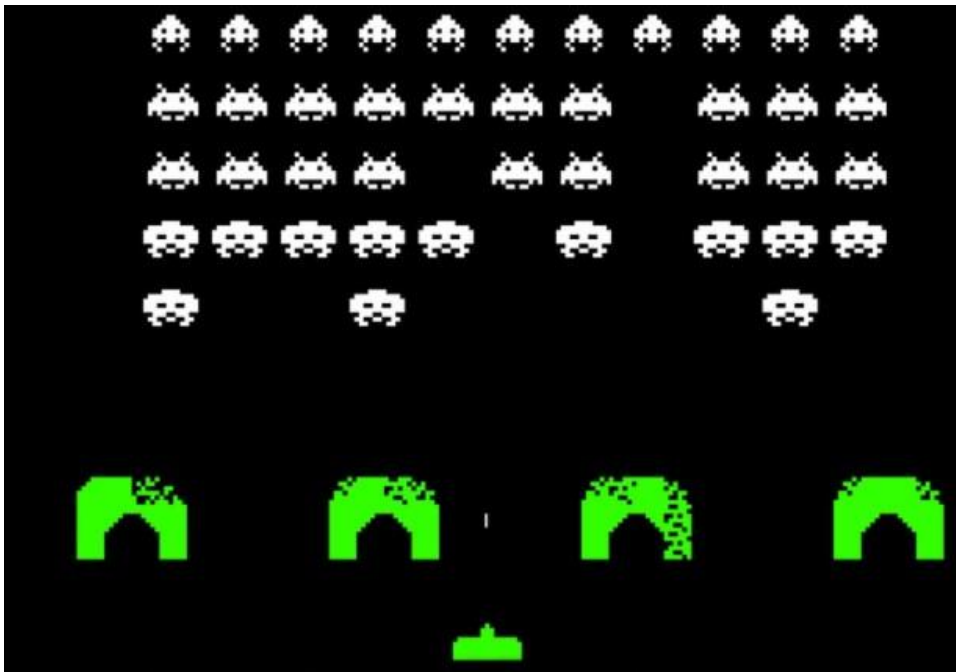
1.Постановка задачи	2
1.1. Формулировка задания	4
1.2. Алгоритмическое разрешение задачи	5
1.3. Контрольные примеры	6
2.Решение задачи	8
2.1 Выбор средств реализации	8
2.2. Описание основных классов	9
2.3. Интерфейс приложения	9
3. Тестирование приложения	9
3.1. Работа приложения на контрольных тестах	11
3.2. Результаты работы	9

1.Постановка задачи

1.1. Формулировка задания

Написать код реализующий игру « **Space invaders** »

Пример программной области:



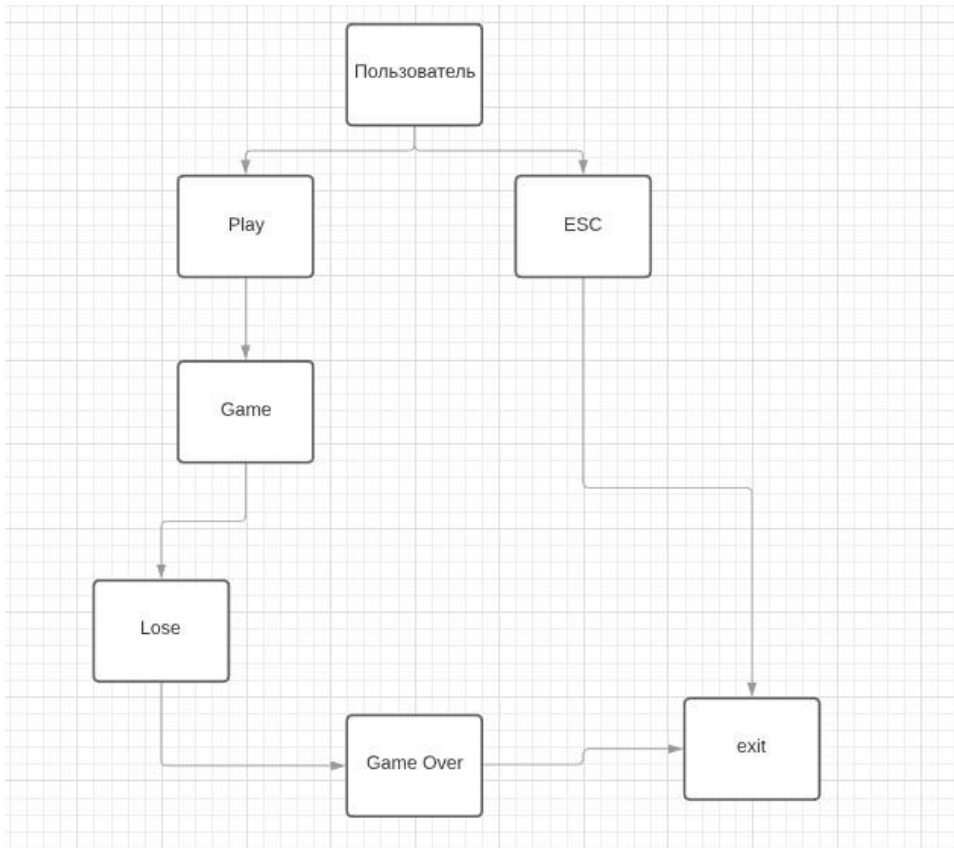
Постановка задачи

Реализовать алгоритм игры « *Space invaders* »

1.2. Алгоритмическое разрешение задачи

Алгоритм проверяет какие клавиши нажимает пользователь , на основе этих данных исполняет движение и выстрелы

Алгоритм пользователя, чтобы играть в игру



1.3. Контрольные примеры

Пример №1

Выбираем режим “Play”

Нажимаем “space” для выстрела

Нажимаем “RiGTH_K” чтобы доска передвинулась направо

Нажимаем “LEFT_K” чтобы доска передвинулась налево

Изменяется кол-во очков

Изменяется уровень

Корабль имеет определенное кол-во очков

Пример №2

Регистрация попадание по нескольким enemy

Проверка работы, прогрессии в очках за одного enemy

Проверка правильности работы жизней в игре

2. Решение задачи

2.1. Выбор средств реализации

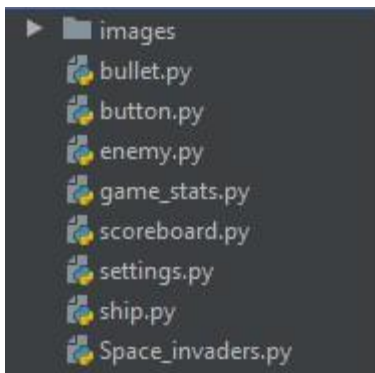
В качестве языка программирования для решения этой задачи был выбран язык Python.

Python это язык программирования общего назначения, нацеленный в первую очередь на повышение продуктивности самого программиста, нежели кода, который он пишет. Говоря простым человеческим языком, на Python можно написать практически что угодно (веб-/настольные приложения, игры, скрипты по автоматизации, комплексные системы расчёта, системы управления жизнеобеспечением и многое другое) без ощутимых проблем.

Также я использовал библиотеку pygame, для более удобной реализации данного проекта

2.2. Описание основных классов

При решении задачи было разработано 8 классов, показанных ниже:



Главный цикл реализующий все действие в игре под названием
“Space_indavers”

Разберем его по подробнее:

1. Добавление основных библиотек и импортирование различных классов

```
import sys
from time import sleep

import pygame

from settings import Settings
from game_stats import GameStats
from scoreboard import Scoreboard
from button import Button
from ship import Ship
from bullet import Bullet
from enemy import Enemy
```

2. Класс показывающий все объекты в игре

```
class SpaceIndavers:

    def __init__(self):

        pygame.init()
        self.screen = pygame.display.set_mode([800, 600])
        pygame.display.set_caption("SpaceIndavers")

        self.settings = Settings()
```

```

self.stats = GameStats(self)
self.sb = Scoreboard(self)
self.ship = Ship(self)
self.bullets = pygame.sprite.Group()
self.enemy = pygame.sprite.Group()

self._create_fleet()
# create button play
self.play_button = Button(self, "Play")

```

3. Функция, которая создает и вычитывает мах кораблей в рядах

```

def _create_fleet(self):
    # create enemy
    enemy = Enemy(self)
    enemy_width, enemy_height = enemy.rect.size
    enemy_width = enemy.rect.width
    available_space_x = self.settings.screen_width - (2 * enemy_width)
    number_enemy_x = available_space_x // (2 * enemy_width)

    ship_height = self.ship.rect.height
    available_space_y = (self.settings.screen_height -
                        (3 * enemy_height) - 2 * ship_height)
    number_rows = available_space_y // (2 * enemy_height)

    # create first enemy fleet
    for row_number in range(number_rows):
        for enemy_number in range(number_enemy_x):
            self._create_enemy(enemy_number, row_number)

def _create_enemy(self, enemy_number, row_number):
    # create enemy
    enemy = Enemy(self)
    enemy_width, enemy_height = enemy.rect.size
    enemy.x = enemy_width + 2 * enemy_width * enemy_number
    enemy.rect.x = enemy.x
    enemy.rect.y = enemy.rect.height + 2 * enemy.rect.height * row_number
    self.enemy.add(enemy)

```

4. Функция, которая определяет края экрана и заставляет енему двигаться в другом направлении

```

def _change_fleet_edges(self):
    for enemy in self.enemy.sprites():
        if enemy.check_edges():
            self._change_fleet_direction()
            break

```

```

def _change_fleet_direction(self):
    for enemy in self.enemy.sprites():
        enemy.rect.y += self.settings.fleet_drop_speed
    self.settings.fleet_direction *= -1

    self.screen = pygame.display.set_mode(
        (self.settings.screen_width, self.settings.screen_height))

```

5. Основная функция запускающая игру

```

def run_game(self):
    # start main round game
    while True:
        self._update_enemy()
        self._check_events()
        if self.stats.game_active:
            self.ship.update()
            self._update_bullets()

        self._update_screen()

```

6. Удаление пуль вышедших за края карты, и проверка дошла ли пуля до цели, также добавление level и обновление enemy

```

def _update_bullets(self):
    self.bullets.update()
    # delete bullet
    for bullet in self.bullets.copy():
        if bullet.rect.bottom <= 0:
            self.bullets.remove(bullet)
    print(len(self.bullets))
    # delete enemy
    self._check_bullet_enemy_collisions()

def _check_bullet_enemy_collisions(self):
    # delete ,bullet and enemy
    collisions = pygame.sprite.groupcollide(
        self.bullets, self.enemy, True, True)
    if collisions:
        for enemy in collisions.values():
            self.stats.score += self.settings.enemy_points * len(enemy)
            self.sb.prep_score()
    if not self.enemy:
        # create new fleet
        self.bullets.empty()
        self._create_fleet()
        self.settings.increase_speed()

        # + level
        self.stats.level += 1
        self.sb.prep_level()

def _update_enemy(self):

```



```

self._change_fleet_edges()
self.enemy.update()
if pygame.sprite.spritecollideany(self.ship, self.enemy):
    self._ship_hit()
self._check_enemy_bottom()

def _ship_hit(self):
    if self.stats.ships_left > 0:
        self.stats.ships_left -= 1
        self.sb.prep_ships()

        self.enemy.empty()
        self.bullets.empty()

        self._create_fleet()
        self.ship.center_ship()

        # pause
        sleep(0.5)
    else:
        self.stats.game_active = False
        pygame.mouse.set_visible(True)

def _check_enemy_bottom(self):
    screen_rect = self.screen.get_rect()
    for enemy in self.enemy.sprites():
        if enemy.rect.bottom >= screen_rect.bottom:

            self._ship_hit()
            break

```

7. действие с клавиатурой

```

def _check_events(self):
    # history screen and keyboard

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
        elif event.type == pygame.MOUSEBUTTONDOWN:
            mouse_pos = pygame.mouse.get_pos()
            self._check_play_button(mouse_pos)
        elif event.type == pygame.KEYDOWN:
            self._check_keydown_events(event)
        elif event.type == pygame.KEYUP:
            self._check_keyup_events(event)

def _check_play_button(self, mouse_pos):

    if self.play_button.rect.collidepoint(mouse_pos):
        # delete static
        button_clicked = self.play_button.rect.collidepoint(mouse_pos)

```

```

        if button_clicked and not self.stats.game_active:
            self.settings.initialize_dynamic_settings()
            self.stats.reset_stats()
            self.stats.game_active = True
            self.sb.prep_level()
            self.sb.prep_score()

            pygame.mouse.set_visible(False)

            self.enemy.empty()
            self.bullets.empty()

            self._create_fleet()
            self.ship.center_ship()

def _check_keydown_events(self, event):
    # reaction on keystroke button
    if event.key == pygame.K_RIGHT:
        self.ship.moving_right = True
    elif event.key == pygame.K_LEFT:
        self.ship.moving_left = True
    # exit with button esc
    elif event.key == pygame.K_ESCAPE:
        sys.exit()
    elif event.key == pygame.K_SPACE:
        self._fire_bullet()

def _check_keyup_events(self, event):
    # reaction on don't keystroke button
    if event.key == pygame.K_RIGHT:
        self.ship.moving_right = False
    elif event.key == pygame.K_LEFT:
        self.ship.moving_left = False

```

8. Создание пули

```

def _fire_bullet(self):
    # create new bullet
    new_bullet = Bullet(self)
    self.bullets.add(new_bullet)

```

9. Постоянное обновление экрана и объектов

```

def _update_screen(self):

    self.screen.fill(self.settings.bg_color)
    self.ship.blitme()
    for bullet in self.bullets.sprites():
        bullet.draw_bullet()
    self.enemy.draw(self.screen)
    self.sb.show_score()
    if not self.stats.game_active:
        self.play_button.draw_button()

```

```
pygame.display.flip()
```

10. Запуск игры

```
if __name__ == '__main__':  
    # create and start game  
    ai = SpaceInvaders()  
    ai.run_game()
```

Также можно было бы рассмотреть и другие классы

2.3. Интерфейс приложения

Кнопка “play” - начало игры

Кнопка “space” – выстрел

Кнопки “Right, Left” – движение

Кнопка “ESC” - выход

Также на экране есть ряд показателей:

Кол-во жизней

Уровни

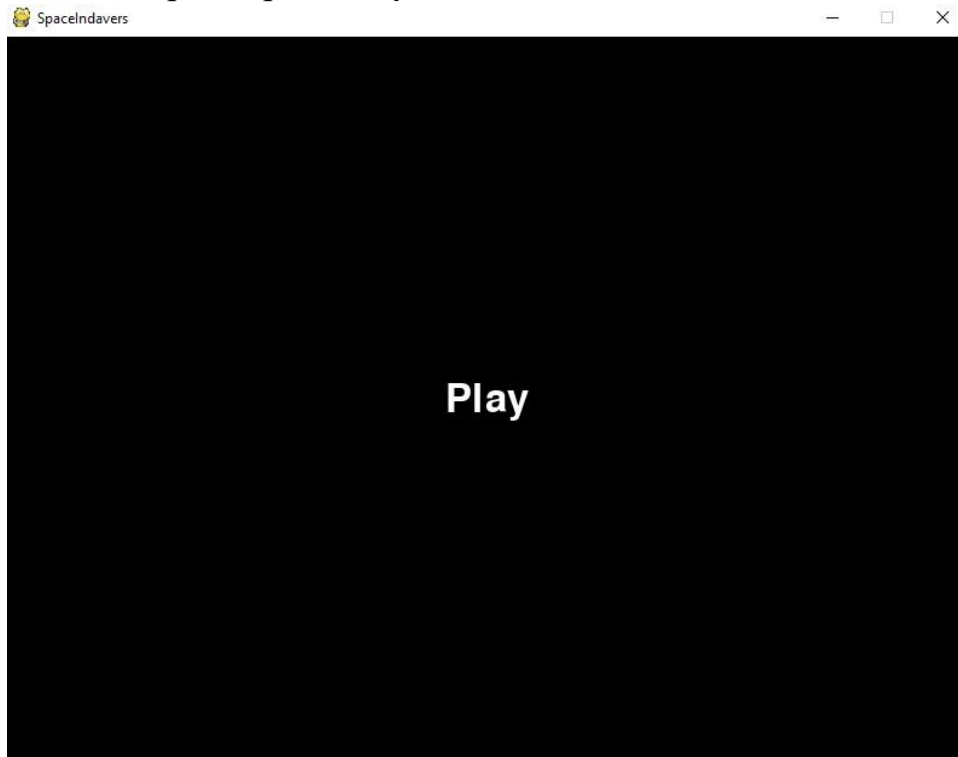
Кол-во очков

3. Тестирование приложения

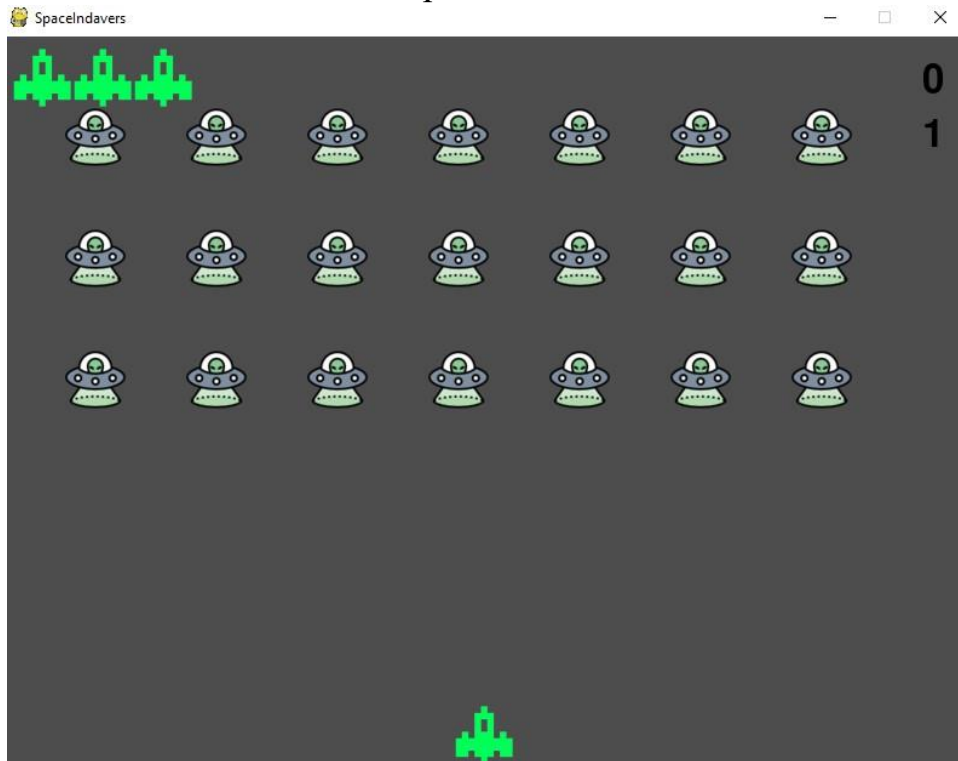
3.1. Работа приложения от начало и до конца

Полный цикл работы игры

1. Начало игры Экран “Play”.



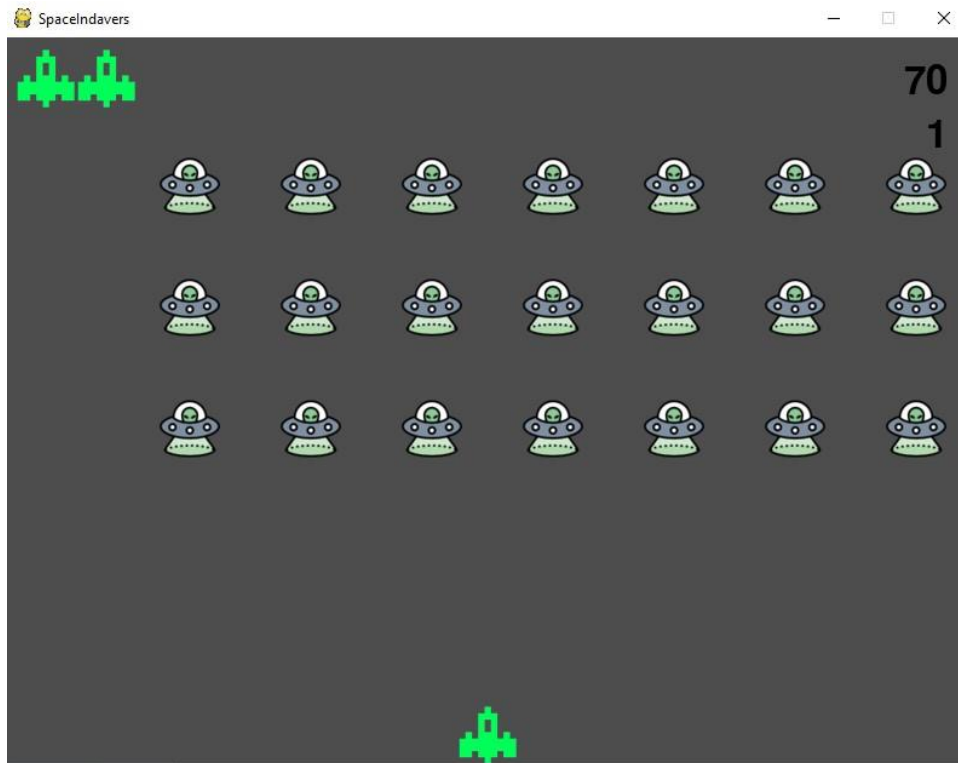
2. Пользователь начинает играть.



3. Когда ship унижтовает епему ему начисляют очки.



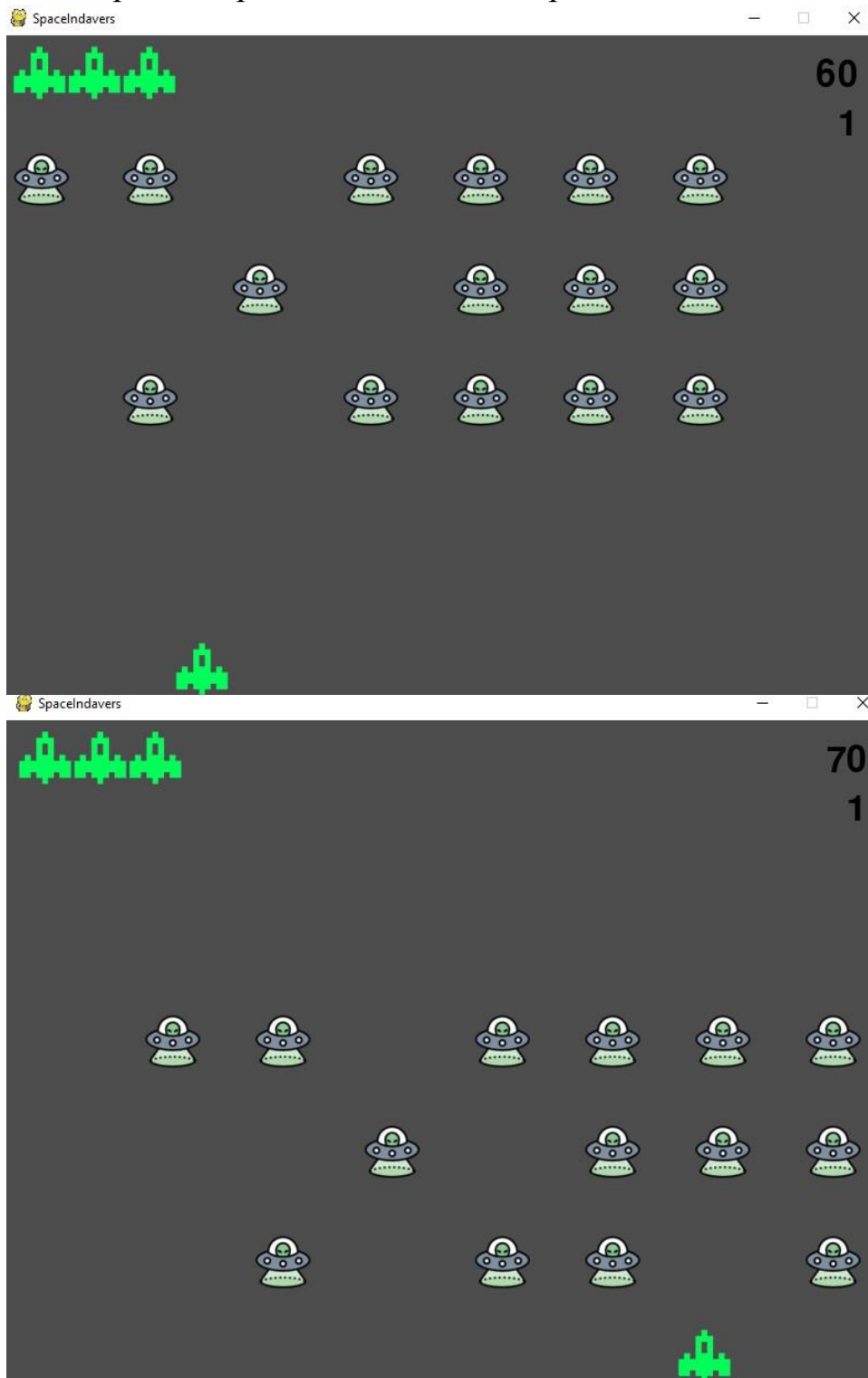
4. Если enemy уничтожают ship, то кол-во жизней уменьшается, если жизни не остается игра заканчивается.



5. Есть прогрессия в уровнях(больше очков и скорость enemy).



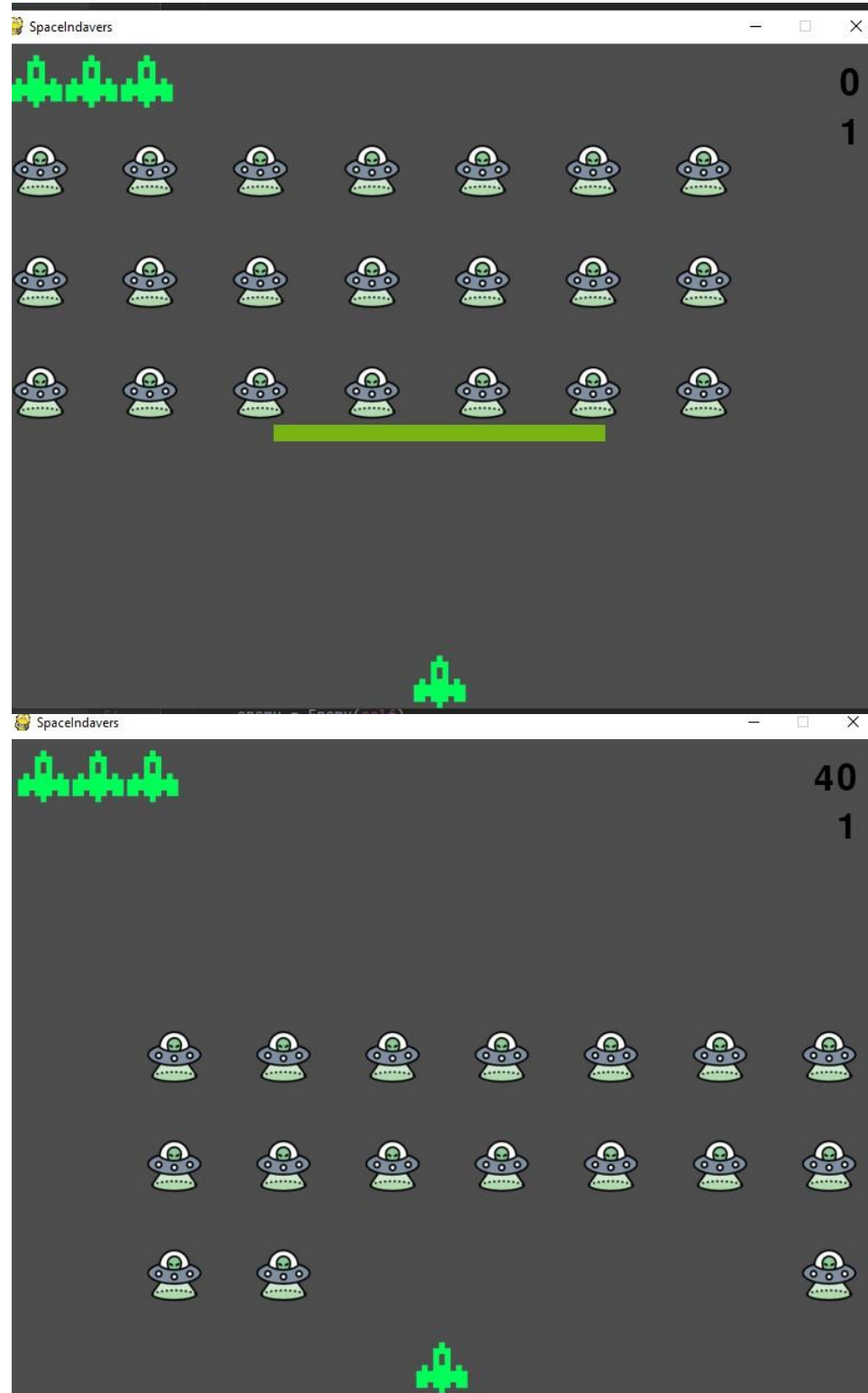
6. И да корабль перемещаться влево и вправо.



3.2. Работа приложения на контрольных тестах

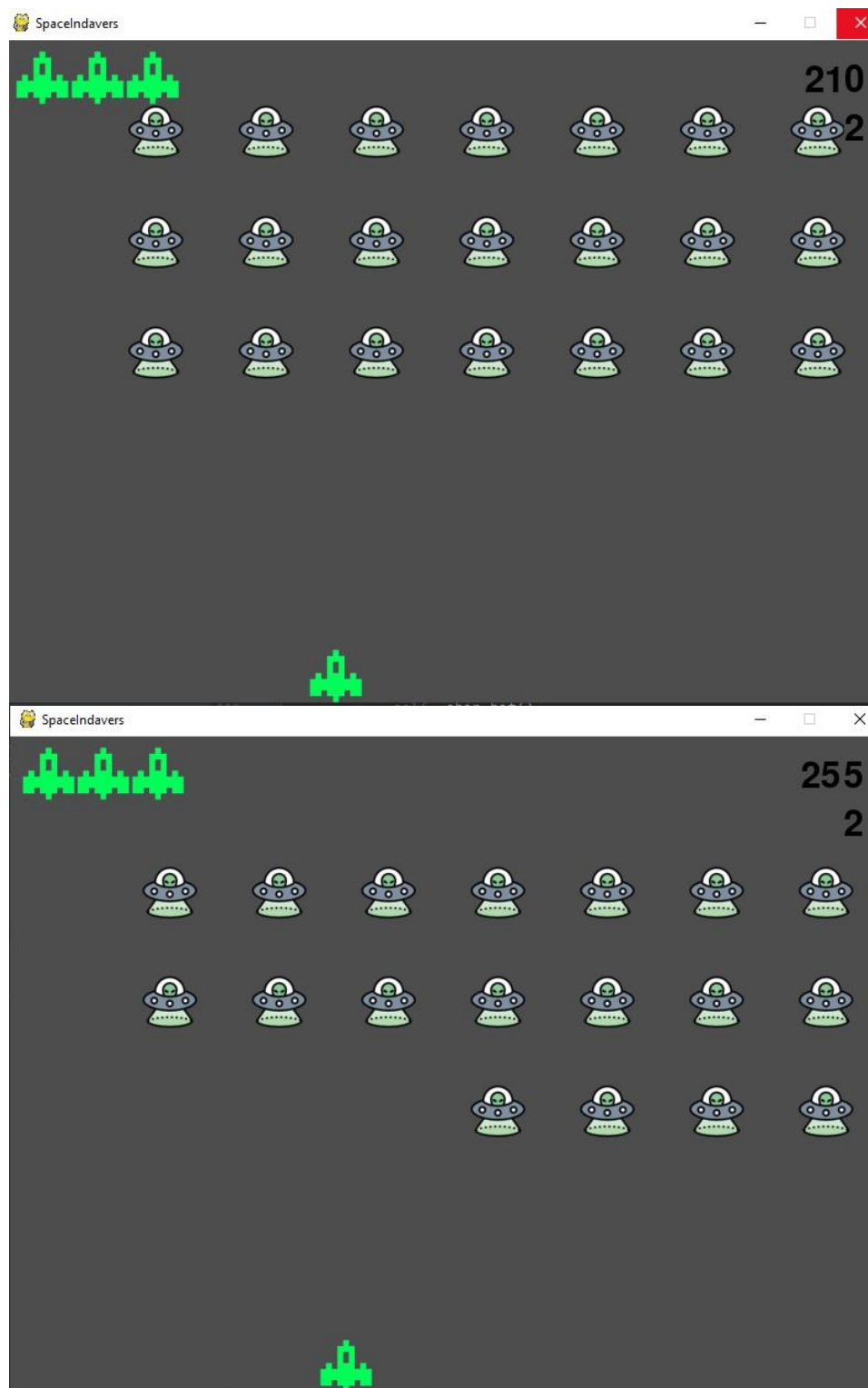
1. Тест: регистрация попадания по нескольким енему. Для этого увеличим наш снаряд и выстрелим в несколько енему сразу.

```
self.bullet_width = 300
```



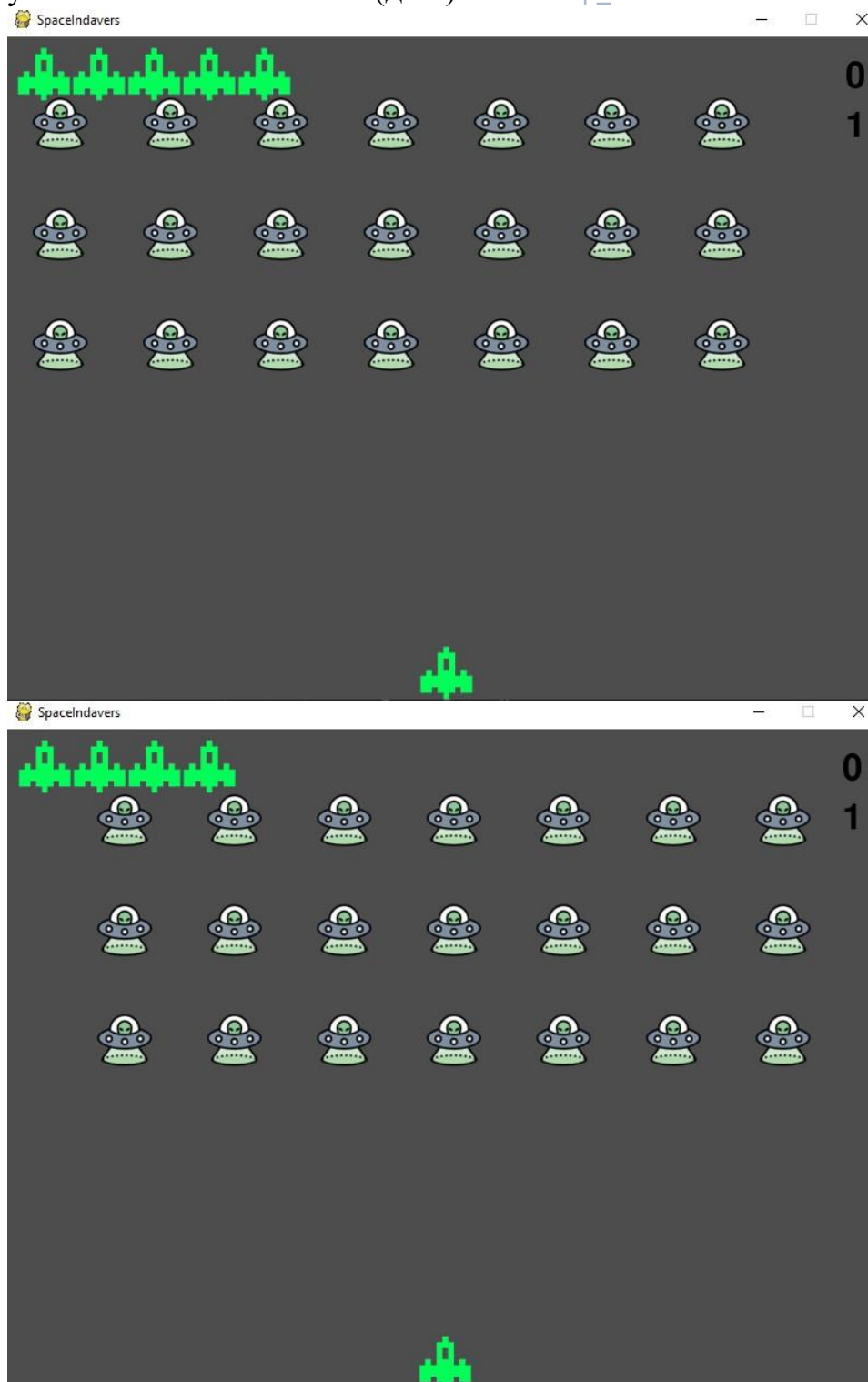
Вывод: все попадание регистрируются

2. Тест: проверим увеличиваются кол-во очков за одного enemy каждый след.уровень. Начальное кол-во очков 10 поставим прогрессию в 1.5 раза. `self.score_scale = 1.5`



Вывод: Как мы видим число некруглое, значит прогрессия очков работает верно.

3. Тест: проверим правильность работы жизней в игре: для этого увеличим кол-во жизней (до 5) `self.ship_limit = 5`



Вывод: Жизни в данном приложении работают корректно.

3.3. Результаты работы приложения

Данная работа предоставила возможность научиться разрабатывать приложения на выбранном мной языке программирования.

Данное приложение позволяет поиграть в игру Space Invaders.