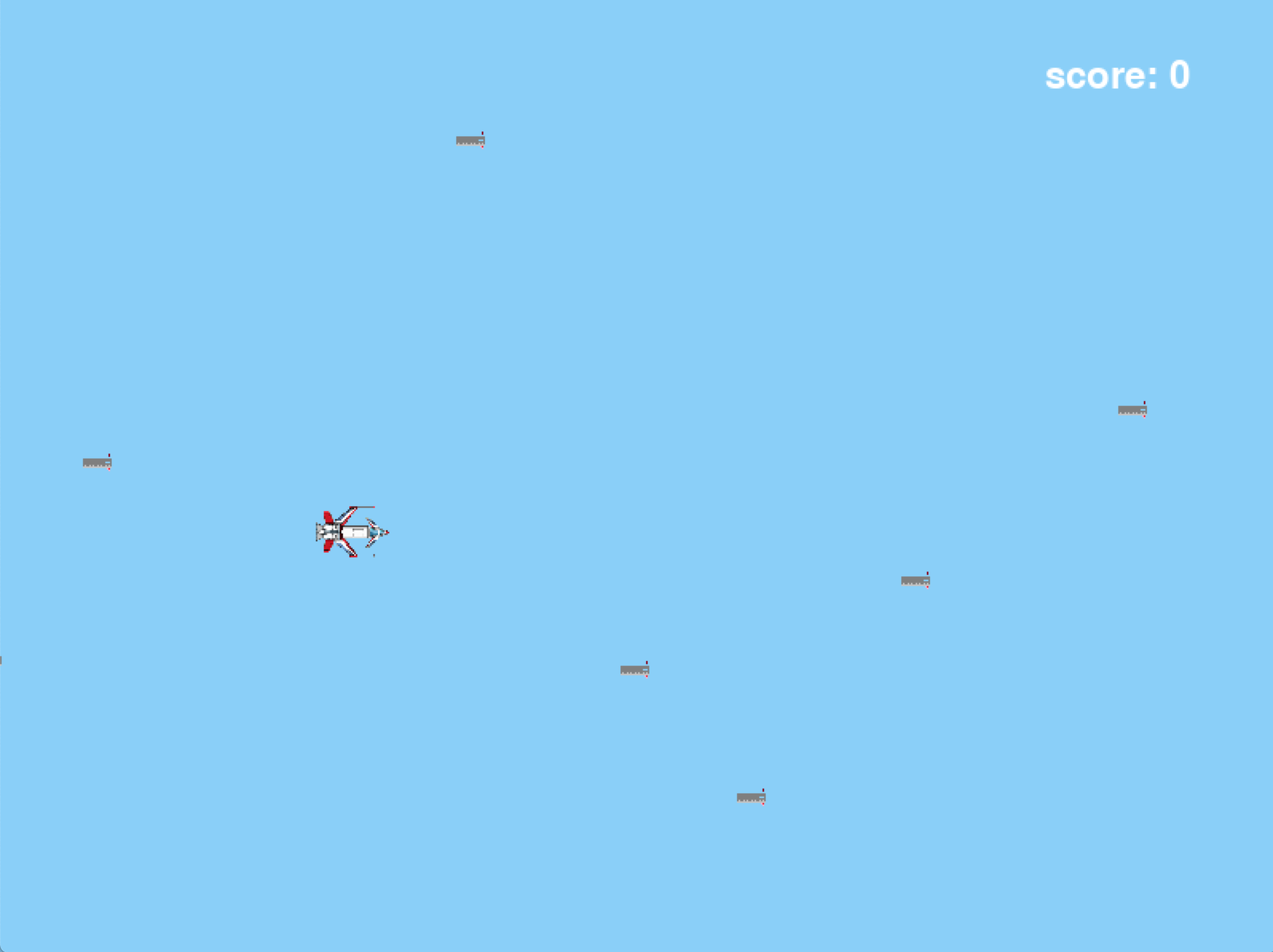
Игра 5: Истребитель



# Повторение

Как обычно, наши первые строчки в игре будут повторять предыдущие проекты - это правильно в программировании использовать уже имеющиеся наработки и изменять их от проекта к проекту. В этом модуле мы рассмотрим уже знакомые нам конструкции, некоторые из них немного модифицируем.

## Импорт и инициализация

* Импортируем библиотеки pygame и random:

|  |
| --- |
| import pygame import random |

* Из модуля pygame импортируем атрибуты, которые понадобятся нам в программ, все наименования кроме первого нам уже известны:

|  |
| --- |
| from pygame.locals import (  RLEACCEL,  K\_UP,  K\_DOWN,  K\_LEFT,  K\_RIGHT,  K\_ESCAPE,  KEYDOWN,  QUIT, ) |

* Инициализируем модуль pygame:

|  |
| --- |
| pygame.init() |

## 

## Экран

Как и в прошлые разы создадим объект экрана с заданными параметрами ширины и высоты:

|  |
| --- |
| SCREEN\_WIDTH = 800 SCREEN\_HEIGHT = 600  screen = pygame.display.set\_mode((SCREEN\_WIDTH, SCREEN\_HEIGHT)) |

## Цвета

Обозначим два цвета в формате RGB, которые будем использовать в этой игре:

|  |
| --- |
| LIGHT\_BLUE = (135, 206, 250) WHITE = (255, 255, 255) |

## 

## ФПС

Добавим ограничение кадров в секунду для того, чтобы наша игра работала одинаково на компьютерах с разными процессорами. Для этого используем модуль Clock() внутри time в pygame:

|  |
| --- |
| clock = pygame.time.Clock() running = True  while running:  for event in pygame.event.get():  if event.type == pygame.QUIT:  running = False  clock.tick(30) |

Оставим стандартное ограничение в 30 кадров.

## Функция put\_text()

Используем нашу функцию put\_text() из прошлых игр, на этот раз будем писать белым цветом по черному экрану:

|  |
| --- |
| font = pygame.font.Font(None, 36)   def put\_text(message, x, y):  text = font.render(message, True, WHITE)  place = text.get\_rect(center=(x, y))  screen.blit(text, place) |

# Цель игры



В этой игре мы будем управлять истребителем. Его цель - долететь до пункта назначения (правый край экрана), но все не так просто: вражеские ракеты будут стремиться уничтожить истребитель.

# План игры

В этот раз мы снова прибегнем к магии программирования - логическое решение задачи до того, как начать писать код. Давайте подумаем об архитектуре нашей игры.

## Объекты

Тут все просто, объектов будет всего два:

* Истребитель - наш главный объект, им будем управлять игрок.
* Ракеты - враги, от них мы будем уворачиваться.

Для обоих объектов мы инициализируем изображение. Мы будем использовать картинки, которые создают разработчики специально для аркадных игр вроде нашей. Исходные файлы разработчики выкладывают на соответствующие сайты. Один из удобных ресурсов - это <https://opengameart.org/> . Оттуда мы и возьмем изображения для нашей новой игры, скачайте их перед тем, как приступать к написанию кода:

1. [Изображение истребителя](https://drive.google.com/file/d/1bMBSaaAkNPn-Ze4h46u6eZzQdiys5AI_/view?usp=sharing)
2. [Изображение ракеты](https://drive.google.com/file/d/1uZi_iAYp4PaSK9-6Esj4gZxxHUjxAW2C/view?usp=sharing)

Оба файла с изображениями необходимо поместить в папку, в которой мы создадим файл с новой игрой. Теперь для запуска игры нам понадобится 3 файла (сам код, написанный на python, и два изображения).

## Процесс игры

Сам процесс игры очень простой:

* Игрок управляет истребителем с помощью стрелочек на клавиатуре.
* Через равный промежуток времени в правой части экрана появляются ракеты.
* Игрок пытается зайти за правую часть экрана, чтобы получить победные очки, уворачиваясь при этом от ракет.

# Реализация

## Класс Player

Для создания класса игрока, мы также наследуем параметры класса Sprite из pygame, после чего создадим прямоугольную область вокруг подгруженного изображения истребителя:

### Инициализация

|  |
| --- |
| class Player(pygame.sprite.Sprite):   def \_\_init\_\_(self):  super().\_\_init\_\_()  self.img = pygame.image.load("jet.png").convert()  self.surf = pygame.transform.scale(self.img, (50, 40))  self.surf.set\_colorkey((0, 0, 0), RLEACCEL)  self.rect = self.surf.get\_rect() |

Давайте разберемся с каждой строчкой:

1. Инициализируем наш класс, как класс спрайта с помощью метода super().
2. Загрузим изображение истребителя и сохраним объект поверхности в отдельную переменную. Метод convert() рекомендован к использованию - он увеличивает скорость отображения пикселей объекта на экране.
3. Для того, чтобы истребитель на экране у нас был не в полный размер, изменим размеры изображения с помощью методов transform.scale(), аргументами будут исходное изображение и кортеж из значений ширины и высоты.
4. Осталась одна проблема - вокруг истребителя не исчез фон. С помощью метода set\_colorkey(), мы сможем исправить это. Передадим туда кортеж из цвета фона (черный) и обозначим нужные нам флаги, RLEACCEL также помогает ускорить отображение.
5. Создадим объект прямоугольной области игрока с помощью метода get\_rect().

### Управление

Для управления нашим кораблем мы будем использовать клавиши направо и налево. Отслеживать нажатия клавиш правильнее в главном цикле игры, добавим эту строчку:

|  |
| --- |
| pressed\_keys = pygame.key.get\_pressed() |

Метод get\_pressed() модуля key возвращает список всех зажатых на данный момент клавиш, добавьте эту строчку в главный цикл после цикла с проверкой event.

В методе нашего класса Player мы используем созданный нами список:

|  |
| --- |
| def update(self):  if pressed\_keys[K\_UP]:  self.rect.move\_ip(0, -5)  if pressed\_keys[K\_DOWN]:  self.rect.move\_ip(0, 5)  if pressed\_keys[K\_LEFT]:  self.rect.move\_ip(-5, 0)  if pressed\_keys[K\_RIGHT]:  self.rect.move\_ip(5, 0)   if self.rect.left < 0:  self.rect.left = 0  if self.rect.left > SCREEN\_WIDTH:  self.rect.left = 0  self.rect.y = SCREEN\_HEIGHT//2  return 1  if self.rect.top <= 0:  self.rect.top = 0  if self.rect.bottom >= SCREEN\_HEIGHT:  self.rect.bottom = SCREEN\_HEIGHT |

* Двигаем объект игрока с помощью метода move\_ip(), со скоростью 5 пикселей/сек в каждом направлении при нажатии соответствующих клавиш.
* Обозначим ограничения для модельки истребителя: за верхний, нижний и левую границы экрана истребитель вылетать не может.
* Если истребитель вылетел за пределы правой границы экрана - вернем 1, это будет означать, что уровень пройден, будем использовать возвращаемое значение в главном цикле. Также в этот момент вернем наш истребитель в левую часть экрана, по центру относительно высоты экрана.

Класс игрока готов, но пока мы не создали экземпляр класса, работа с ним не начнется.

## Класс Enemy

Здесь все проще, создадим класс по аналогии:

1. Инициализируем объект ракеты:
   1. Загрузим изображение.
   2. Выделим область вокруг размером 20 на 20 пикселей.
   3. Место для появления ракетки выберем с помощью метода randint, так наши ракеты будут появляться на разной высоте и на разной удаленности - от 20 до 100 пикселей за экраном.
   4. Обозначим скорость ракеты - рандомное значение в промежутке с 5 до 12 пикселей в секунду.
2. Метод update() у ракеты будет очень простой:

Ракеты летит слева направо, и если вылетела за левый край экрана, удаляем её из всех групп спрайтов.

|  |
| --- |
| class Enemy(pygame.sprite.Sprite):   def \_\_init\_\_(self):  super(Enemy, self).\_\_init\_\_()  self.img = pygame.image.load("missile.png").convert()  self.surf = pygame.transform.scale(self.img, (20, 20))  self.surf.set\_colorkey((0, 0, 0), RLEACCEL)  self.rect = self.surf.get\_rect(  center=(  random.randint(SCREEN\_WIDTH + 20, SCREEN\_WIDTH + 100),  random.randint(0, SCREEN\_HEIGHT),  )  )  self.speed = random.randint(5, 12)   def update(self):  self.rect.move\_ip(-self.speed, 0)  if self.rect.right < 0:  self.kill() |

## 

## Группы спрайтов

Перед тем, как продолжить, создадим группы спрайтов в коде над главным циклом:

|  |
| --- |
| all\_sprites = pygame.sprite.Group()  enemies = pygame.sprite.Group()  player = Player() all\_sprites.add(player) |

Всего у нас будет 2 группы:

1. all\_sprites - все существующие спрайты. Эту группу мы будем использовать для отображения всех спрайтов на экране и их движения.
2. enemies - вражеские корабли. В эту группу мы будем заносить экземпляры классов врагов, использовать эту группу будем для того, чтобы отделить врагов от всех остальных спрайтов, это очень важный момент.

В последних двух строчках мы создали экземпляр класса Player и поместили его в группу all\_sprites.

## 

## Отрисовка спрайтов

Добавим в главный цикл отображение и движение всех спрайтов после строчки с созданием списка зажатых клавиш:

|  |
| --- |
| enemies.update() if player.update():  for enemy in enemies:  enemy.kill()  score += 1  screen.fill(LIGHT\_BLUE)  for entity in all\_sprites:  screen.blit(entity.surf, entity.rect) |

1. Во всех классах наших объектов мы переопределяем метод update(), в нашем случае отвечает за движение объектов:
   1. Сначала вызовем update() для ракет.
   2. Вызовем отдельно update() для игрока и проверим, если метод вернул 1, тогда уничтожим все ракеты на карте и добавим к счетчику очков 1.
2. Заполняем экран небесным цветом для дальнейшей прорисовки на экране по-новому.
3. Проходим циклом по группе all\_sprites и каждую сущность отображаем на ее поверхности с помощью метода blit().

## 

## Создаем объекты ракет

В прошлой игре для создания группы врагов мы использовали цикл, но в данном случае этот способ не подойдет, так как ракеты у нас будут появляться с равной периодичностью на протяжении всей игры.

### Создание ивента

Для этого нам понадобятся кастомные ивенты:

* Создать его очень просто: для начала необходимо присвоить порядковый номер.
* После чего зададим интервал, с которым будет активироваться ивент.

Добавим эти строчки до главного цикла:

|  |
| --- |
| ADDENEMY = pygame.USEREVENT + 1 pygame.time.set\_timer(ADDENEMY, 500) |

ADDENEMY - это имя нашего ивента, который будет происходить каждые 500 миллисекунд (полсекунды).

Первая строчка создает уникальный порядковый номер, рекомендуется использовать + n, для каждого из n созданных в программе ивентов.

### 

### Вызов ивента в цикле

Наш ивент будет программно происходить каждые полсекунды, нам осталось только отловить его, сделаем это в цикле проверки ивентов, добавим необходимые строчки в конец этого маленького цикла:

|  |
| --- |
| while running:  for event in pygame.event.get():  if event.type == KEYDOWN:  if event.key == K\_ESCAPE:  running = False  elif event.type == QUIT:  running = False  elif event.type == ADDENEMY:  new\_enemy = Enemy()  enemies.add(new\_enemy)  all\_sprites.add(new\_enemy) |

Каждый раз, когда будет появляться ивент ADDENEMY, будем создавать экземпляр Enemy(), добавим его в группы спрайтов enemies и all\_sprites. Вот и всё - каждые полсекунды у нас будут появляться новые ракеты. Отличное применение принципов ООП.

## Проверка коллизии

Теперь, когда мы разобрались с группами спрайтов, осталось проверить на то, столкнулся ли наш истребитель с какой-либо ракетой - в таком случае игра будет окончена. Добавим строчки проверки ближе к концу главного цикла:

|  |
| --- |
| if player.rect.right < SCREEN\_WIDTH and pygame.sprite.spritecollideany(player, enemies):  player.kill()  game\_over() |

* Добавим условие, чтобы истребитель не разбивался, если столкнулся с ракетой уже за пределами экрана.
* Если столкновение произошло, удалим игрока из всех групп спрайтов и вызовем функцию game\_over().

## 

## Функция game\_over()

Все просто - выведем на экран конечный счет игрока и оставим возможность выхода из игры по нажатию на крестик или клавиши эскейп.

|  |
| --- |
| def game\_over():  screen.fill(LIGHT\_BLUE)  put\_text("Game over! Final score: " + str(score), SCREEN\_WIDTH // 2, SCREEN\_HEIGHT // 2)  pygame.display.flip()  while True:  for ev in pygame.event.get():  if ev.type == QUIT:  pygame.quit()  exit()  if ev.type == KEYDOWN:  if ev.key == K\_ESCAPE:  pygame.quit()  exit() |

## Переменные

Перед главным циклом обозначим:

* Счет score = 0 в начале игры.
* running для непрерывности цикла.

|  |
| --- |
| score = 0 running = True |

## 

## Конец главного цикла

В конце главного цикла добавим отображение текущего счета игрока и не забудем отобразить все изменения на экране с помощью метода flip(), последние строчки внутри цикла будут выглядеть таким образом:

|  |
| --- |
| put\_text(("score: " + str(score)), SCREEN\_WIDTH - 100, 50) pygame.display.flip()  clock.tick(30) |

# Заключение

Сегодня мы впервые использовали файлы с изображениями, чтобы получилась более красочная с точки зрения визуализации игра. Потренировались в работе с группами спрайтов и коллизией. Получилась интересная игра на “фарм” победных очков. Теперь можно соревноваться со своими прошлыми результатами и улучшать владение истребителем. А можно ли сохранять эти результаты?

# Ссылка на проект

Для того, чтобы увидеть мой вариант решения, переходите по прикрепленной ссылке:

[Игра 5: "Истребитель"](https://drive.google.com/file/d/1-XC8CwxL4sZrJnZCOSZ3ipX-k-RdG842/view?usp=sharing)