# Лабораторная работа №5 – SDD

# Выполнил - Угарин Никита Александрович

**1. Введение**

**1.1 Цель документа:** Данный документ описывает архитектуру, проектные решения и структуру программного обеспечения для разработки 2D аркадной игры “Инопланетное вторжение”. Он служит руководством для разработчиков и обеспечивает понимание общей структуры системы.

**1.2 Область применения:** Этот документ определяет границы системы и описывает все компоненты, входящие в состав проекта игры “Инопланетное вторжение”.

**1.3 Ссылки:**

* [GDD] [Документ игрового дизайна "Инопланетное вторжение"](https://github.com/Nikitos1408/UCHEBA/blob/main/TexDoc/LR5/ЛР5GDDУгаринНА.docx)
* [SRS] [Спецификация программных требований "Инопланетное вторжение"](https://github.com/Nikitos1408/UCHEBA/blob/main/TexDoc/LR5/ЛР5SRSУгаринНА.docx)

**2. Общее описание системы**

**2.1 Обзор системы:** “Инопланетное вторжение” - это 2D аркадный платформер, в котором игрок управляет космическим кораблем, перемещая его влево и вправо, и стреляет по инопланетным противникам. Игра состоит из нескольких уровней с возрастающей сложностью. Основная цель – уничтожить всех врагов на каждом уровне.

**2.2 Архитектурные ограничения:**

* Используется язык программирования Python 3.12.2
* Используется библиотека Pygame 2.6.1
* Игра должна работать в разрешении 800x600 пикселей.
* Проект должен быть завершен в течение 3 недель.

**3. Архитектурный дизайн**

**3.1 Архитектурный подход:** Применяется объектно-ориентированный подход. Игра разделена на отдельные классы, представляющие основные игровые сущности (космический корабль, противники, пули и т.д.). Это позволяет обеспечить модульность, читаемость и возможность повторного использования кода.

**4. Детальное описание компонентов**

**4.1 Класс Game**

**Описание:** Главный класс, управляющий общим состоянием игры, игровым циклом, и отрисовкой.

**Атрибуты:**

* ai\_settings: Объект класса Settings, содержащий игровые настройки.
* screen: Экран Pygame.
* stats: Объект класса GameStats, содержащий статистику игры.
* ship: Объект класса Ship, представляющий корабль игрока.
* bullets: Объект Group (Pygame), содержащий объекты класса Bullet.
* aliens: Объект Group (Pygame), содержащий объекты класса Alien.

**Методы:**

* \_\_init\_\_(self):
* Инициализирует Pygame (pygame.init(), pygame.mixer.init()).
* Создает объект Settings.
* Создает экран Pygame (pygame.display.set\_mode()).
* Устанавливает заголовок окна (pygame.display.set\_caption()).
* Создает объект GameStats.
* Создает объект Ship.
* Создает группы спрайтов bullets и aliens.
* Создает объект UI. (Предполагается, исходя из SRS и предыдущих корректировок)
* run(self): Основной игровой цикл.
* Запускает бесконечный цикл (while True)
* Вызывает gf.check\_events() для обработки событий.
* Если игра активна (stats.game\_active):
* Обновляет состояние корабля (ship.update()).
* Обновляет состояние пуль (gf.update\_bullets()).
* Обновляет состояние пришельцев (gf.update\_aliens()).
* Обновляет экран (gf.update\_screen()).
* Иначе (игра не активна):
* Отображает главное меню (gf.draw\_menu()).
* Обновляет экран (pygame.display.flip()).
* load\_level(self): Загрузка данных уровня (например, создание пришельцев, начальные позиции).
* game\_over(self): Обработка завершения игры.
* save\_game(self): Сохранение текущего состояния игры.
* load\_game(self): Загрузка сохраненного состояния игры.

**4.2 Класс Settings**

**Описание:** Класс для хранения настроек игры, таких как скорость, цвет, размер объектов, и т.д.

**Атрибуты:**

* screen\_width: Ширина экрана (int, по умолчанию 800).
* screen\_height: Высота экрана (int, по умолчанию 600).
* bg\_color: Цвет фона (tuple, по умолчанию (230, 230, 230)).
* ship\_speed\_factor: Скорость перемещения корабля (float, по умолчанию 1.5).
* ship\_limit: Количество жизней корабля (int, по умолчанию 0).
* bullet\_speed\_factor: Скорость пули (int, по умолчанию 3).
* bullet\_width: Ширина пули (int, по умолчанию 3).
* bullet\_height: Высота пули (int, по умолчанию 15).
* bullet\_color: Цвет пули (tuple, по умолчанию (60, 60, 60)).
* bullets\_allowed: Максимальное количество пуль на экране (int, по умолчанию 5).
* alien\_speed\_factor: Скорость перемещения пришельцев (int, по умолчанию 1).
* fleet\_drop\_speed: Скорость снижения флота пришельцев (int, по умолчанию 10).
* fleet\_direction: Направление движения флота пришельцев (-1 или 1).
* speedup\_scale: Коэффициент ускорения игры при переходе на следующий уровень (float, по умолчанию 1.1).
* laser\_sound: Звук выстрела лазера (pygame.mixer.Sound).
* explosion\_sound: Звук взрыва (pygame.mixer.Sound).
* ship\_hit\_sound: Звук попадания по кораблю (pygame.mixer.Sound).
* game\_over\_sound: Звук окончания игры (pygame.mixer.Sound).

**Методы:**

* \_\_init\_\_(self): Инициализация настроек.
* initialize\_dynamic\_settings(self): Инициализация динамически изменяемых настроек (например, скорость).
* increase\_speed(self): Увеличение скорости игры (корабля, пуль, пришельцев) при повышении уровня.

**4.3 Класс GameStats**

**Описание:** Класс для хранения статистики игры, такой как количество оставшихся жизней, счет и уровень.

**Атрибуты:**

* ai\_settings: Объект класса Settings.
* ships\_left: Количество оставшихся жизней корабля (int).
* score: Текущий счет игрока (int).
* level: Текущий уровень игры (int).
* game\_active: Флаг, указывающий, идет ли игра в данный момент (boolean).

**Методы:**

* \_\_init\_\_(self, ai\_settings): Инициализация статистики игры.
* reset\_stats(self): Сброс статистики к начальным значениям.

**4.3 Класс Ship**

**Описание:** Представляет космический корабль игрока.

**Атрибуты:**

* screen: Экран Pygame.
* ai\_settings: Объект класса Settings.
* image: Изображение корабля.
* rect: Прямоугольник, определяющий положение корабля.
* screen\_rect: Прямоугольник экрана.
* center: Горизонтальная координата центра корабля (float).
* moving\_right: Флаг, указывающий, что корабль движется вправо.
* moving\_left: Флаг, указывающий, что корабль движется влево.

**Методы:**

* \_\_init\_\_(self, ai\_settings, screen): Инициализация корабля.
* update(self): Обновление положения корабля на основе его движения.
* blitme(self): Отрисовка корабля на экране.
* center\_ship(self): Установка корабля в центр экрана.

**4.4 Класс Alien**

**Описание:** Представляет инопланетного противника.

**Атрибуты:**

* screen: Экран Pygame.
* ai\_settings: Объект класса Settings.
* image: Изображение пришельца.
* rect: Прямоугольник, определяющий положение пришельца.
* x: Горизонтальная координата пришельца (float).

**Методы:**

* \_\_init\_\_(self, ai\_settings, screen): Инициализация пришельца.
* blitme(self): Отрисовка пришельца на экране.
* update(self): Обновление положения пришельца в соответствии с его скоростью и направлением.
* check\_edges(self): Проверка, достиг ли пришелец края экрана.

**4.5 Класс Bullet**

**Описание:** Представляет пулю, выпущенную кораблем.

**Атрибуты:**

* screen: Экран Pygame.
* ai\_settings: Объект класса Settings.
* rect: Прямоугольник, определяющий положение и размеры пули.
* y: Вертикальная координата пули (float).
* color: Цвет пули.
* speed\_factor: Скорость пули.

**Методы:**

* \_\_init\_\_(self, ai\_settings, screen, ship): Инициализация пули.
* update(self): Обновление положения пули.
* draw\_bullet(self): Отрисовка пули.

**4.8 Модуль gf (game functions)**

**Описание:** Модуль, содержащий функции, связанные с игровой логикой, обработкой событий, обновлением состояний игры, отрисовкой и управлением игровым процессом.

**Функции:**

* check\_events(ai\_settings, screen, ship, bullets, stats, aliens): Проверяет игровые события (нажатие клавиш, события мыши).
* check\_keydown\_events(event, ai\_settings, screen, ship, bullets, stats, aliens): Обрабатывает события нажатия клавиш (движение корабля, стрельба).
* check\_keyup\_events(event, ship): Обрабатывает события отпускания клавиш (остановка движения корабля).
* fire\_bullet(ai\_settings, screen, ship, bullets): Создает и добавляет пулю в список пуль.
* update\_screen(ai\_settings, screen, stats, ship, aliens, bullets): Обновляет отрисовку экрана (фон, корабль, пришельцы, пули, интерфейс).
* draw\_level(screen, stats): Отображает текущий уровень.
* update\_bullets(ai\_settings, screen, stats, ship, aliens, bullets): Обновляет положение пуль, проверяет столкновения пуль с пришельцами, удаляет пули, вышедшие за экран.
* check\_bullet\_alien\_collisions(ai\_settings, screen, stats, ship, aliens, bullets): Проверяет столкновения пуль с пришельцами.
* get\_number\_aliens\_x(ai\_settings, alien\_width): Вычисляет количество пришельцев в одном ряду.
* get\_number\_rows(ai\_settings, ship\_height, alien\_height): Вычисляет количество рядов пришельцев.
* create\_alien(ai\_settings, screen, aliens, alien\_number, row\_number): Создает и добавляет одного пришельца в список пришельцев.
* create\_fleet(ai\_settings, screen, ship, aliens): Создает флот пришельцев.
* update\_aliens(ai\_settings, screen, stats, ship, aliens, bullets): Обновляет положение пришельцев, проверяет их столкновения, проверят достижение нижнего края.
* check\_fleet\_edges(ai\_settings, aliens): Проверяет, достиг ли флот края экрана.
* change\_fleet\_direction(ai\_settings, aliens): Изменяет направление движения флота пришельцев.
* ship\_hit(ai\_settings, screen, stats, ship, aliens, bullets): Обрабатывает столкновение корабля с пришельцем или достижение края экрана.
* check\_aliens\_bottom(ai\_settings, screen, stats, ship, aliens, bullets): Проверяет достижение пришельцами нижней границы экрана.
* save\_game(stats, ship, aliens, bullets, ai\_settings): Сохраняет состояние игры в файл.
* load\_game(stats, ship, aliens, bullets, ai\_settings, screen): Загружает состояние игры из файла.
* draw\_menu(screen, ai\_settings): Отрисовывает главное меню.
* check\_menu\_buttons(ai\_settings, screen, ship, bullets, stats, aliens, mouse\_x, mouse\_y): Проверяет нажатия на кнопки меню.
* start\_new\_game(ai\_settings, screen, ship, bullets, stats, aliens): Начинает новую игру, сбрасывая всю статистику и создавая флот пришельцев.

**5. Дизайн данных**

**5.1 Структуры данных:**

* **Игровые объекты:** Объекты Player, Enemy, Bullet будут храниться в списках (например, enemy\_list, bullet\_list).
* **Уровни:** Данные об уровнях (количество врагов, их позиции) будут храниться в виде словарей или текстовых файлов.

**5.2 Формат данных:**

* **Настройки:** Настройки игры могут храниться в формате JSON или текстовом файле.
* **Ресурсы:** Изображения (спрайты), звуки будут храниться в соответствующих папках.

**6. Интерфейсы системы**

**6.1 Внутренние интерфейсы:**

Взаимодействие между классами Game, Player, EnemyManager, Enemy, Bullet, UI будет осуществляться через вызов методов и передачу необходимых данных.

**6.2 Внешние интерфейсы:**

Пользовательский интерфейс будет обеспечивать взаимодействие с игроком через клавиатуру (стрелки, пробел) и мышь (для меню).

**7. Пользовательский интерфейс**

**7.1 Описание интерфейса:**

* Главное меню: Отображение кнопок “Новая игра” и “Загрузить игру” с возможностью выбора мышью.
* Игровой интерфейс: В верхнем левом углу отображается номер текущего уровня, а также имеется подсказка о кнопке сохранения.
* Игровое окно будет представлять собой 2D вид сверху с персонажем и инопланетянами.

**8. Требования к производительности**

**8.1 Частота кадров:** Игра должна поддерживать стабильную частоту кадров не менее 30 FPS.

**8.2 Потребление памяти:** Потребление оперативной памяти не должно превышать 150 МБ.

**8.3 Скорость загрузки:** Время загрузки уровней должно быть минимальным.

**9. Безопасность**

**9.1 Обработка ошибок:**

* Реализация блоков try-except для обработки возможных исключений.
* Логирование ошибок для упрощения отладки.

**9.2 Валидация данных:**

Проверка корректности загружаемых данных (например, уровней) перед использованием.

**10. Обеспечение качества**

**10.1 Стратегия тестирования:**

* Модульное тестирование отдельных компонентов (классов).
* Интеграционное тестирование для проверки взаимодействия между компонентами.
* Бета-тестирование с привлечением группы пользователей.

**10.2 Критерии приемки:**

* Игра работает без критических ошибок.
* Все заявленные функциональные требования реализованы в соответствии с SRS.
* Игра обеспечивает стабильную производительность и отзывчивое управление.

**11. Требования к среде разработки**

**11.1 Инструменты разработки:**

* VS Codium 1.96.2 (редактор кода)
* Python 3.12.2
* Pygame 2.6.1

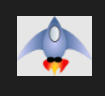
**11.2 Ограничения среды:**

Игра разрабатывается только для операционной системы Windows.

**12. Приложения**

**12.2 Дополнительные материалы:**

* Игровые модельки.

* Эскизы интерфейса.

