# Структура проекта

Изображение выглядит как текст, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

# Описание ключевых классов

1. Класс Player (KKKickers.GameObjects)

Я написал класс для персонажа в 2D-игре: прыжки, скольжение, движение, смерть, отрисовка — всё моё.

* Описание:
  + Синглтон (только один игрок в игре)
  + Поля: Position, MoveSpeed, Gravity, JumpForce, спрайты (\_slideSprite, \_jumpSprite, \_fallSprite, \_deadSprite).
  + Флаги: IsDead, IsSliding, IsFacingRight, JumpsRemaining (до 2 прыжков).
  + Методы:
    - Jump(): Прыжок, второй — с разворотом.
    - Slide(): Скольжение, возвращает прыжки.
    - Die(): Смерть.
    - ApplyGravity(): Падение/подъём, меняет спрайты.
    - Draw():Отрисовка персонажа с поворотом и направлением.
    - Reset(): Сброс на старт.
    - HandleMovement(): Бег вправо/влево.
* Закодил всю логику, физику, анимации и нарисовал спрайты, протестировал геймплей.

1. Класс Wall (KKKickers.GameObjects.Walls)

Я разработал абстрактный класс Wall для представления стен в 2D-игре KKKickers, обеспечивающий базовую функциональность для препятствий.

* Назначение: определяет общую структуру для всех типов стен, включая их позицию, размеры, отображение и взаимодействие с игроком.
* Основные компоненты:
  + Свойства:
    - Bounds (RectangleF): определяет позицию и размер стены.
    - Sprite (Image, абстрактное): Спрайт стены, реализуемый в производных классах.
    - IsActive (bool): указывает, активна ли стена (по умолчанию true).
  + Конструктор:
    - Принимает position (PointF) и size (Size) для инициализации Bounds.
  + Методы:
    - Draw(Graphics g): отрисовывает спрайт стены, если она активна и спрайт не null.
    - HandleCollision(Player player): обрабатывает столкновение с игроком, используя вспомогательный класс WallCollisionHelper. Виртуальный, может переопределяться.
    - Update(): Виртуальный метод для обновления состояния стены (по умолчанию пустой).
* Разработал архитектуру класса, включая абстрактную основу для различных типов стен. Реализовал логику отрисовки и обработки столкновений. Создал базовую структуру, позволяющую легко расширять функциональность через наследование.

1. Классы папки GameLogic (KKKickers.GameLogic)

Я разработал классы для управления основной логикой игрока, стен, фона и игрового процесса.

* Назначение: управляют состоянием игры, включая игрока, препятствия, фон, очки и подсказки.
* Для каждого класса синглтон (экземпляров каждого из классов должно быть по одному)
* Сути классов:
  + BackgrondUpdator – отвечает за состояние заднего фона
  + CollisionHandler – проверяет коллизию Player с Wall
  + ScoreHandler – отвечает за счет, если надо записывает рекорд
  + SpikeActivator – активирует/деактивирует шипы у шипованных стен
  + StructureGenerator – генерирует и хранит структуры
  + UIHandler – отвечает за подсказки-издевалки
* Разработал всю логику управления игрой: координацию игрока, препятствий, фона и счёта. Реализовал генерацию структур стен и динамическое перемещение фона. Создал систему подсказок и обработку столкновений. Настроил механику шипов и обновление игровых объектов.

# Логика работы

Главное меню – нажатие «Играть».

GameManager создает начальную и следущую за ней случайную структуры.

Игрок прыгает, избегая препятствий.

При выходе наивысшей точки текущей структуры за границы поля зрения, она текущая становится следующей, а следующая случайно выбирается (без повторов предыдущей).

При падении игрока за границы экрана – экран с кнопкой «Еще раз», счетом и советом.

# Затраченное время

* На рисование спрайтов ушло около 2 часов,
* На разработку игровой логики ушло около 18 часов,
* На создание и тестирование структур ушло около 3 часов