**Описание проекта**

Игра представляет собой космический 2D шутер на основе старой игры Asteroids 1979, но при этом не является полным клоном старой игры. Новый проект разработан в программе Unity2020.3.2 для платформы PC, управление рассчитано на клавиатуру, кнопки меню нажимаются мышкой. Оригинальная игра, судя по управлению, была для приставки. Мобильную версию этой игры для Android или iOS сделать проблематично, из-за разнообразия элементов управления, которые неудобно и не эргономично размещать на сенсорном экране. Возможна адаптация для браузерной версии на платформе WebGL.

**Описание** **игры**

Задачей для игрока является набрать как можно баллов, которые даются за уничтожение астероидов космическим кораблем. За уничтожение крупных и средних астероидов начисляется 25 баллов, за уничтожение мелких – 50, за уничтожение летающей тарелки- 150. При столкновении с астероидом игрок теряет одну жизнь, когда жизни заканчиваются, игра завершается. Дополнительными сложностями являются: при уничтожении крупных астероидов из обломков образуются более мелкие, периодически появляется летающая тарелка, при столкновении с которой также теряется одна жизнь. В отличие от астероидов, траектория полета летающей тарелки не прямолинейна. В моем проекте уничтожение летающей тарелки дает бонус в виде добавления одной жизни.

### Управление

В ходе игры задействованы кнопки клавиатуры: кнопки A, D, W, S служат для перемещения вперед-назад, ArrowLeft, ArrowRight для поворота, ArrowUp для стрельбы «неуправляемой не самонаводящейся инерционной ракетой класса воздух-воздух». Кнопка пробел позволяет ставить игру на паузу, снимать с паузы. После окончания игры появляются меню, с помощью кнопок которого можно выйти (для автономной версии) или начать заново.

### Настройка и первый запуск

Игра не имеет специальных настроек, запускается либо из редактора Unity, либо нажатием на файл Asteroids1979.exe для скомпилированной версии.

### FAQ

• Почему картинки игровых объектов и анимация сделаны так примитивно? Это стилизация под исходную версию Asteroids, так же как шрифт элементов интерфейса и черный фон экрана.

• Как можно улучшить приложение? Можно сделать ввод имени игрока и сохранение результата (так было в оригинальной игре).

### Поддержка.

Данная игра является тестовым заданием для Qmobil, поэтому публикация и поддержка игры не планируется. По вопросам прохождения обращайтесь к разработчику.

### Архитектура проекта

Приложение состоит из единственной сцены SampleScene. В ней имеются следующие объекты:

MainCamera;

Canvas с подчиненными элементами интерфейса – картинками, надписями и кнопками; EventSystem со стандартными настройками;

ShipeActive – игровой объект космического корабля, который управляется пользователем;

AsteroidManager – игровой объект, который програмно управляет добавлением и удалением префабов на сцену по ходу игры.

### Интерфейсы.

В приложение реализовано три интерфейса: стартовая заставка с объяснением управления, основное окно приложения и итоговая заставка с объявлением результатов (Рис. 1).

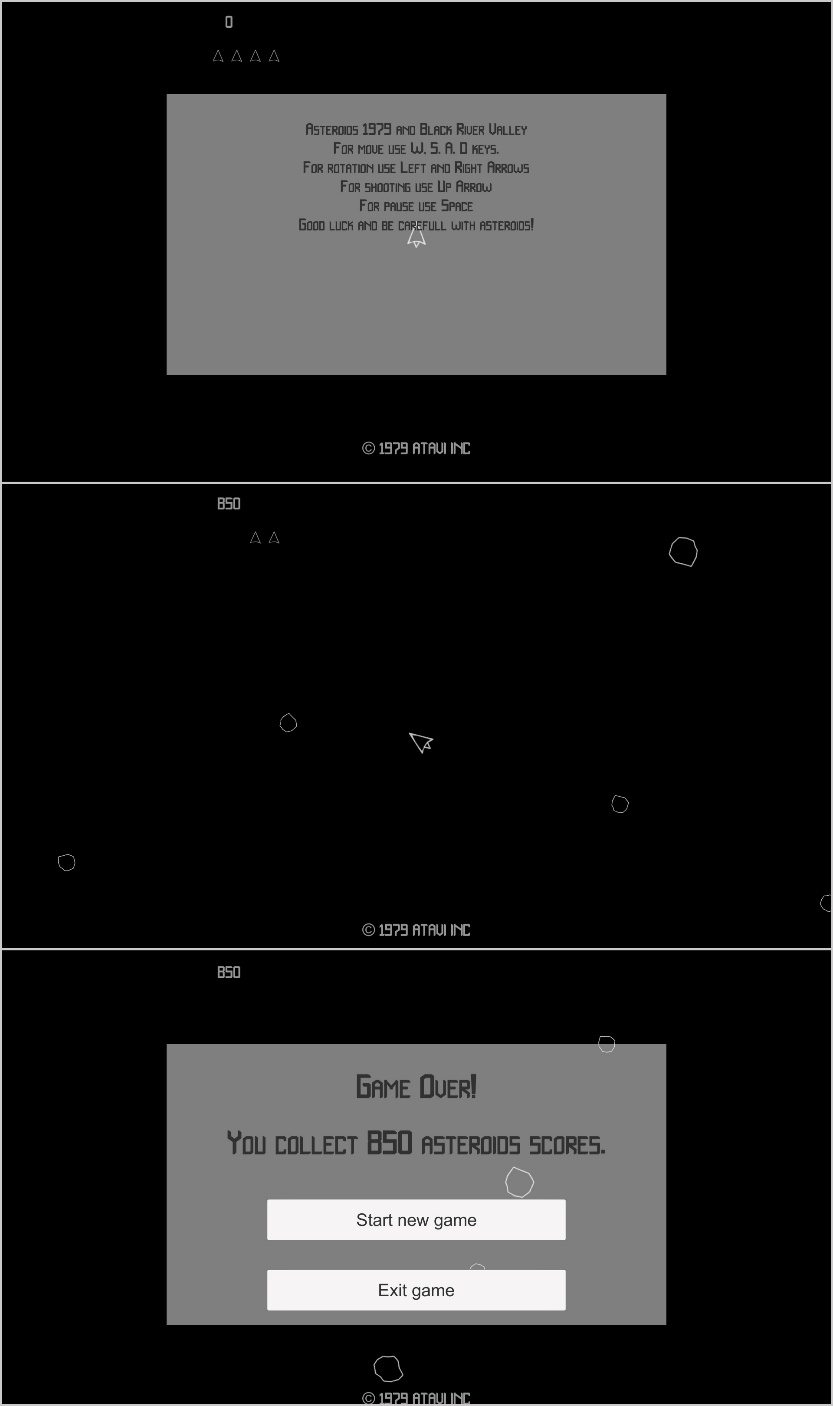


Рис. 1 Интерфейсы Asteroids 1979.

### Опорные классы проекта.

В проекте создано 9 пользовательских классов, все наследуются от MonoBehavior.

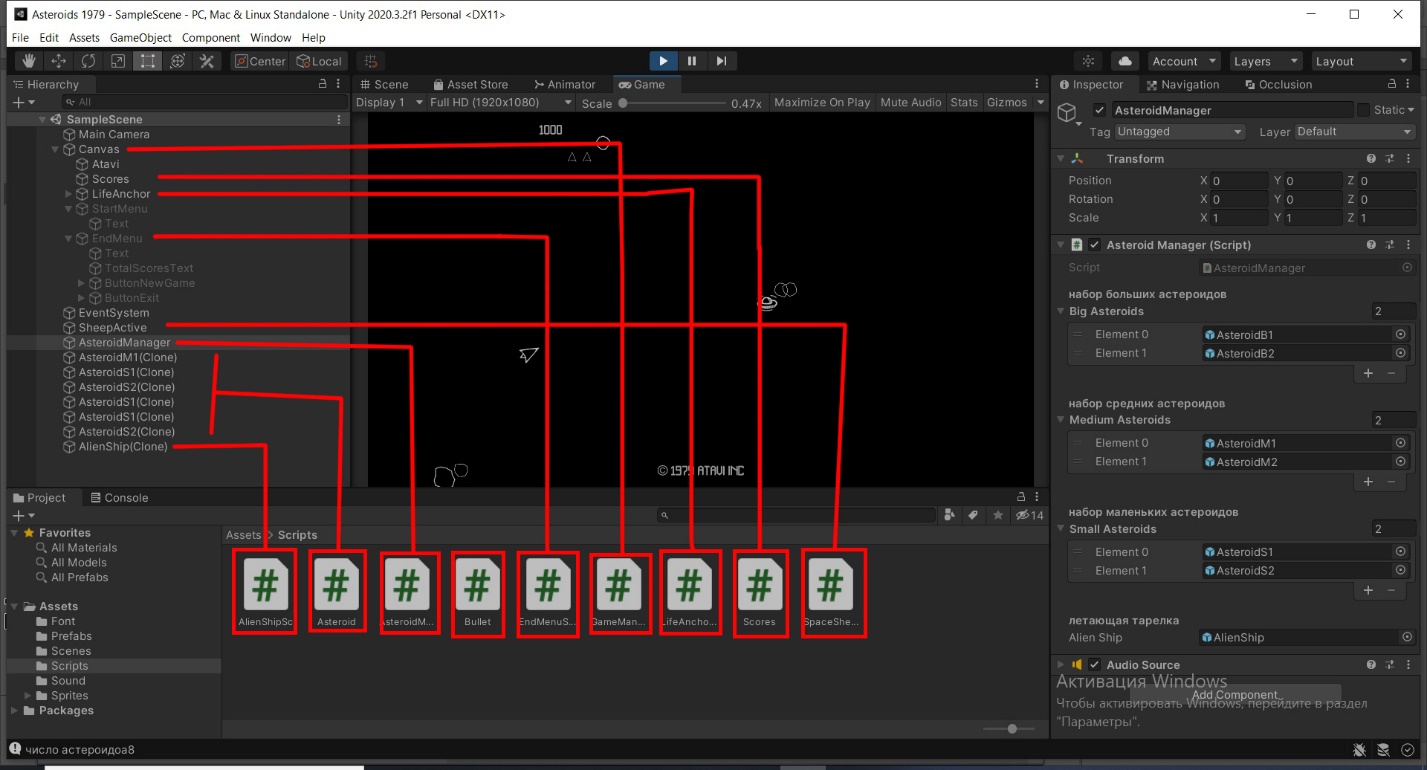


Рис. 2. Размещение классов на игровых объектах при работе приложения.

Таблица классов приложения и основных функций.

|  |  |  |
| --- | --- | --- |
| класс | Основные функции и процедуры класса | назначение |
| **AlienShipSc** | IEnumerator NewDestination() | Корутина, которые через 5 секунд определяет новую точку назначения для движения объекта |
|  | public void AlienDestroy() | Функция разрушения летающей тарелки (запуск анимации, звук и удаление префаба) |
| **Asteroid** | public void TrobleShoot() | Функция разрушения астероида (запуск анимации, звук и удаление префаба) |
|  | private Vector3 LocationNew() | Перемещает астероид на новое место и дает разворот к центру экрана |
| **AsteroidManager** | IEnumerator timeForNewSet() | Корутина которая каждые 5 сек проверяет количество астероидов всех размеров на сцене и если их меньше 5 запускает функцию добавления нового набора астероидов |
|  | IEnumerator TimeForNewAlien() | Корутина которая каждые 15 сек проверяет количество астероидов всех размеров на сцене и если их меньше 10 запускает функцию добавления летающей тарелки |
|  | private void CreateAlien() | Функция создания летающей тарелки |
|  | IEnumerator DestroyAlien() | Корутина которая запускается после создания лет тарелки и через 15 сек проверяет ее наличие, если есть удаляет |
|  | public void CreateNewSet() | Функция создания нового набора астероидов. С каждой итерацией на один астероид больше |
|  | public void CreateNewWreck(int AsteroidCode, Vector3 pos) | Функция создания новых астероидов при уничтожении старого. Принимает значение поля Код и позицию игрового объекта. |
| **Bullet** | private void OnTriggerEnter2D(Collider2D col) | Функция обеспечивает взаимодействие префаба с мишенью. Задействует функцию уничтожения на объекте в зависимости от его тэга |
| **EndMenuScores** | void Update() | Проверка статического поля количества баллов, отображение в виде текста |
| **GameManagerSc** | public void StartGame() | Загрузка сцены – начало игры по нажатию кнопки |
|  | public void ExitGame() | Выход из приложения в автономной версии (в редакторе не работает) |
|  | public void EndGame() | Открывает итоговое меню, ставит игру на паузу |
| **LifeAnchorSc** | void Update() | Отображает число жизней игрока в виде количества префабов- картинок прикрепленных к пустому игровому объекту на канвасе |
| **Scores** | void Update() | Следит за отображением набранных баллов по ходу игры |
| **SpaceSheepSc** | IEnumerator EndGameDelay() | Задерживает на 0.5 с окончание игры для того чтобы проигралась анимация разрушения корабля |
|  | private void OnTriggerEnter2D(Collider2D col) | При столкновении с игровым объектом уменьшает число жизней, которое записано в статичном поле этого класса. |
|  | private void KeyboardInput() | Функция обеспечивающая управление и стрельбу. Вынесено в отдельную функцию из Update, чтобы при необходимости было удобно добавить директивы компиляции и адаптировать игру под другую платформу |

### Логика создания астероидов.

Астероиды это префабы с классом Asteroid, которые различаются значением поля Asteroid .AsteroidCode, размером (в компоненте Transform) и картинкой. По значению этого поля существует три типа астероидов: 1 крупные, 2- средние, 3 –мелкие. При уничтожении астероида после попадания в него ракеты запускается функция класса AsteroidManager. В нее передается код астероида и его положение в мире (глобальные координаты). В зависимости от кода функция добавляет баллы за уничтоженный астероид и создает в переданной позиции два астероида более мелкого класса, либо ничего не создает (если уничтожен мелкий астероид с кодом 3), но баллы все равно начисляет.

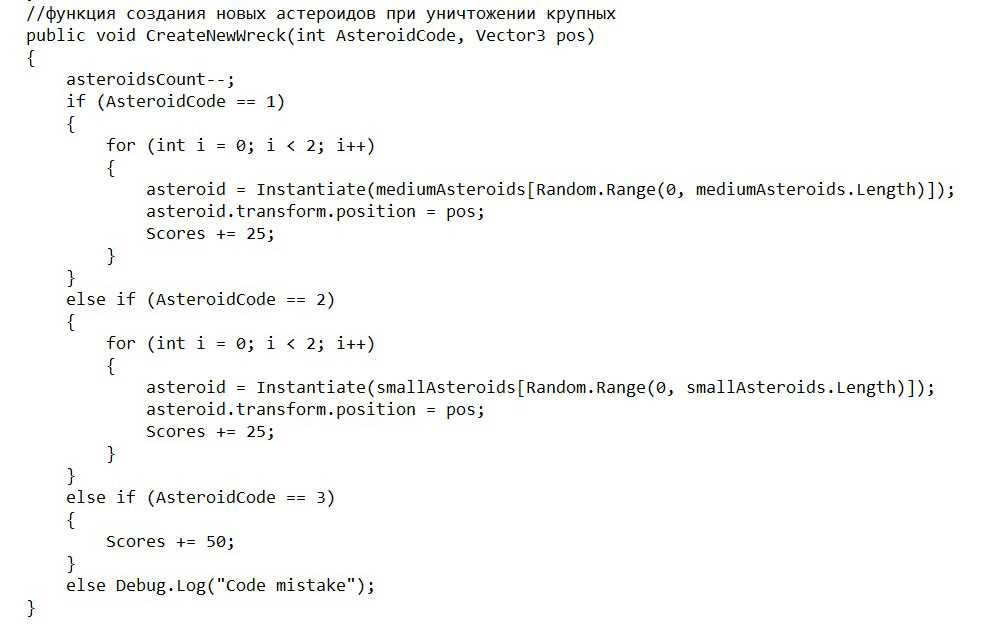


Рис. 3. Код создания астероидов

### Описание внутренней логики игры для разработчика.

### В начале игры AsteroidManager инициализирует на сцене первый набор из 4 крупных астероидов, при появлении каждого из них счетчик астероидов прибавляет 7 (расчет: 1 большой астероид делится на 2 средних которые делятся на 4 маленьких – итого 7 единиц). Игрок начинает стрелять по астероидам ракетами, которые инициализируются классом SpaceSheepSc при нажатии вниз на кнопку СтрелкуВверх. Ракета после появления на сцене управляется собственным скриптом Bullet. Она движется без остановки и либо вылетает за край экрана, после чего удаляется со сцены, либо врезается в игровой объект с коллайдером, запускает на нем функцию OnTriggerEnter и также удаляется со сцены. При запуске функции астероид передает в класс AsteroidManager функцию создания обломков, по которой на его месте создаются дочерние астероиды 2 поколения, из счетчика астероидов вычитается один, сам астероид убирается со сцены. Когда значение счетчика астероидов меньше 10 AsteroidManager добавляет на сцену летающую тарелку, когда меньше 5 – создает новый набор астероидов. Летающая тарелка изначально двигается в сторону игрока, затем через каждые 5 секунд получает новую цель для движения.

### C:\Users\BlackRiver\Desktop\Рис2.tif

Рис. 4. Отношения между классами