1 Импортируем ассеты.

2 Добавляем корабль (Space/Prefabs): ShipModel. Позиция в 0,0,0. Масштаб 0.1, 0.1, 0.1

3 Добавляем огонь из сопел (Procedural fire/Prefabs): RightEngine. Поворот в -90, 0, 0.

4 Копированием создаем из RightEngine еще один объект LeftEngine

5 Объединяем эти три объекта в один CreateEmpty. PlayerShip. Позиция в 0,0,0

6 На этом этапе нужно проверить, что и PlayerShip и ShipModel стоят в позициях 0,0,0.

7 Настраиваем камеру. Позиция в 0,25,0. Поворот в 90, 0, 0. Projector – Orthographics, Size – 45. Размер экрана во вкладке Game создаем новый – 600 на 900.

Orthographics – означает, что нет перспективы, то есть объекты одинакового размера независимо от расстояния до камеры.

8 Создаем фон. Create – 3D Object – Plane. Background. Позиция в 0,-10,0. Поворот в 0, 0, 0. Масштаб 10, 1, 10. Перетаскиванием на Plane создаем текстуру Space/NebulaTile

9 Источники освещения. LeftLight. Позиция в -15,9,-1.5. Поворот в 135, -90, 0. Color – красный. Intensity – 1.

10 Копированием создаем еще один источник освещения. RightLight. Позиция в 15,9,-1.5. Поворот в 45, -90, 0. Color – красный. Intensity – 1.5.

11 Движение корабля на кнопки. Добавляем PlayerShip компонент Rigidbody, убираем Use Gravity. Добавляем скрипт PlayerScript.

private Rigidbody Ship;

void Start()

{

Ship = GetComponent<Rigidbody>();

}

void Update()

{

float deltaH = Input.GetAxis("Horizontal");

float deltaV = Input.GetAxis("Vertical");

Ship.velocity = new Vector3(deltaH, 0, deltaV);

}

12 Чтобы корабль летал со скоростью, которая нам нужна:

public float speed;

Ship.velocity = new Vector3(deltaH, 0, deltaV) \* speed;

13 Для реалистичности пространственного полета добавляем повороты. Продемонстрировать в Unity повороты, что когда мы наклоняем нос корабля вперед-назад, то есть двигаемся по вертикали по оси z, то меняется поворот по x. И наоборот, передвигая корабль по оси x, слева направо, меняется угол по оси z.

Таким образом код будет выглядеть так:

public float angle; Задаем в Unity 35.

Ship.rotation = Quaternion.Euler(deltaV \* angle, 0, deltaH \* angle);

14 При тестировании увидим, что по оси x нужно изменить направление следующим образом:

Ship.rotation = Quaternion.Euler(deltaV \* angle, 0, -deltaH \* angle);

15 Накладываем ограничение, чтобы корабль не вылетал за экран. Смотрим в Unity, что корабль влево может быть сдвинут только до -20, а вправо до 20. Вниз мы можем переместит корабль до -35, а вверх до 22, чтобы нос корабля остался виден.

Таким образом мы ставим в коде ограничения на позицию:

public float xMin, xMax, zMin, zMax; Выставляем в Unity -20, 20, -35, 22.

В скрипте:

float xPosition = Mathf.Clamp(Ship.position.x, xMin, xMax);

float zPosition = Mathf.Clamp(Ship.position.z, zMin, zMax);

Ship.position = new Vector3(xPosition, Ship.position.y, zPosition);