Galaxy

1. Player:

Скрипт на игроке PlyerScript. Он отвечает за передвижение персонажа (далее Игрок), здоровье Игрока, подсчет очков.

* 1. Перемещение Игрока происходит путем нажатия на кнопки Horizontal и Vertical ( по умолчанию стрелки на клавиатуре и кнопки WASD )

public void FixedUpdate()

{

MovementLogic();

}

private void MovementLogic()

{

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

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

Vector3 movement = new Vector3(moveHorizontal, moveVertical , 0.0f);

rb.AddForce(movement \* Speed); //даем силу толчка

}

* 1. Поворот в сторону мышки в Update:

var mousePosition = Input.mousePosition;

mousePosition = Camera.main.ScreenToWorldPoint(mousePosition);

var angle = Vector2.Angle(Vector2.right, mousePosition - transform.position);

transform.eulerAngles = new Vector3(0f, 0f, transform.position.y < mousePosition.y ? angle : -angle);

* 1. Ограничиваем движение Игрока экраном:

Сначала в методе Start задаем параметры экрана:

float dist = Vector3.Distance(pos, Camera.main.transform.position);

leftBorder = Camera.main.ViewportToWorldPoint(new Vector3(0, 0, dist)).x;

rightBorder = Camera.main.ViewportToWorldPoint(new Vector3(1, 0, dist)).x;

topBorder = Camera.main.ViewportToWorldPoint(new Vector3(0, 1, dist)).y;

bottomBorder = Camera.main.ViewportToWorldPoint(new Vector3(0, 0, dist)).y;

Далее в Update проверяем:

pos = transform.position;

transform.position = new Vector3(Mathf.Clamp(pos.x, leftBorder, rightBorder), Mathf.Clamp(pos.y, bottomBorder, topBorder), pos.z);

* 1. Фиксируем повреждения в Damageplayer();
  2. Добавляем очки за убитого противника UpdateScore();
  3. Включаем звук взрыва, когда убиваем противника Playvzriv();

1. Выстрелы.

Игроку добавляем пустой GameObject (в нашем случае добавил Sprite). Располагаем его перед Игроком. Из этой точки будут вылетать снаряды. И вешаем на него скрипт Shoot.

* 1. В Start создается пул объектов из снарядов. Задать размер можно в pool\_count;
  2. В Update при нажатии на левую кнопку мыши создаются снаряды на сцене (если зажать кнопку мыши, то снаряды будут создаваться с интервалом TimeSpawn). При создании снаряда проигрывается звук выстрела.

if (Input.GetMouseButton(0) == true)

{

if (Time.time > nextTime)

{

audioSource.PlayOneShot(shootClip);

nextTime = Time.time + TimeSpawn;

GameObject obj = pool\_parent.GetChild(pool\_element\_ID).gameObject;

obj.SetActive(true);

if (obj.activeInHierarchy)

{

obj.transform.position = new Vector3(point.transform.position.x, point.transform.position.y);

obj.transform.parent = null;

}

pool\_element\_ID++;

if (pool\_element\_ID > pool\_parent.childCount - 1) pool\_element\_ID = 0;

}

}

* 1. pool\_parent - позиция точки в которой будет появляться снаряд ;

pool\_element\_ID - Id элемента в пуле объектов;

Prefab – префаб снаряда;

pool\_count – размер пула объетов;

point - позиция точки в которой будет появляться снаряд, во время создания снаряда в Update();

TimeSpawn – время задержки создания снарядов;

audioSource – объект звука выстрела на сцене;

shootClip – аудио дорожка выстрела;

1. Скрипт Bullet

Его необходимо повесить на префаб снаряда.

Он включает в себя движение снаряда в Update().

А так же находит своего родителя и через определенное время в иерархии объектов, переносится к родителю и отключается.

1. Создание противников (разделены на 2 категории: 1. Большие камни 2. Мелкие камни и летающие тарелки).

Противники создаются по принципу снарядов.

Большие камни (tag – Rock1 и далее Rock1) создаются в скрипте CreateRock, мелкие камни и летающие тарелки (tag - Rock2 и tag – NLO , далее по tag)

* 1. CreateRock – необходимо повесить на объект в котором будут создаваться Rock1

Он таккой же как и в скрипте создания выстрелов Shoot. Так же создается пулл объетов. Отличается только тем, что передает cоздаваемым объектам тэг родителя. Это необходимо, что бы после уничтожения, объект нашел своего родителя и стал дочерним.

public EnemyScript en; // сюда в инспекторе необходимо закинуть префаб Rock1 со скриптом EnemyScript;

* 1. Скрипт EnemyScript – помещаем на префаб Rock1

Отвечает за перемещение Rock1, за столкновения со снарядами и игроком, за создание Rock2, возвращение к родителю и отключение, если его не уничтожили

Метод CreateminiRock()

После уничтожения Rock1 при помощи Random определяет сколько необходимо создать Rock2. Создает Rock2, изменяет угол поворота Rock2 на случайный угол от 180 до -180 градусов.

KolvaminiRock = Random.Range(0, 4); // количество Rock2

prefabMiniRock = new GameObject[KolvaminiRock];

for (int i = 0; i < prefabMiniRock.Length; i++)

{

prefabMiniRock[i] = Instantiate(prefabMini, transform.position, transform.rotation, parentMiniRock);

prefabMiniRock[i].SetActive(true);

prefabMiniRock[i].transform.rotation = Quaternion.Euler(0, 0, Random.Range(-180, 180)); // Изменяет угол поворота Rock2 от 180 до -180 градусов

prefabMiniRock[i].transform.position = new Vector3(this.transform.position.x, this.transform.position.y, this.transform.position.z); //определяет позицию в которой активирует Rock2

prefabMiniRock[i].transform.parent = null;

Метод OnTriggerEnter2D()

Отвечает за столкновения с Игроком и снарядами

* 1. Скрипт CreateNLO – необходимо повесить на объект в котором будут создаваться NLO.

Идентичен Shoot.

* 1. Скрипт MinyEnemy – помещаем на префабы Rock2 и NLO.

Этот скрипт отвечает за столкновения с Игроком и снарядами.