

Урок 4 «AddForce и работа со сценами»

Теория по новым командам, которую стоит объяснить и показать на примере.

AddForce – это метод Rigidbody который прикладывает силу к объекту в заданном нами направлении.

Update и FixedUpdate - Update вызывается каждый кадр, а FixedUpdate с фиксированной частотой независимо от FPS. При этом частоту можно изменить в настройках Unity.

Его частота зависит от того, насколько быстро компьютер способен отображать изображения. На более медленном компьютере Update вызывается реже, чем на более быстром. Если вы выполняете расчеты на основе времени, вы можете нормализовать их, используя Time.deltaTime, который скажет сколько времени прошло с момента последнего вызова Update. Обычно его используют для выполнения задач, связанных с отображением (например, обновление элемента пользовательского интерфейса)

FixedUpdate вызывается с фиксированной частотой независимо от FPS. При этом частоту можно изменить в настройках Unity.

Вызывается фиксировано независимо от того, как часто изображение обновляется. Обычно используют для выполнения задач, связанных с игровым процессом (например, обновление физики).

Теги(Tags) - маркеры, которые могут использоваться для идентификации объектов в вашем проекте (для деталей см. эту страницу). Добавить новые теги можно путём ввода нового тега в пустую форму внизу списка тегов или увеличив значение Size. Уменьшение Size удалит теги из конца списка.

Слои (Layers) используются в Unity как способ группирования объектов, которые разделяют определённые характеристики.

Покажи, как их создавать и применять к объекту.

Scenemanager используется для управления сценами в Unity, можно показать его реализацию в проекте про нож, если не разбирали до этого проект или забыли.

Покажи, как создавать новые сцены и перетаскивать их в BuildSettings, поясни что число справа это **buldindex**, для дальнейшей реализации переключения уровней.

Задания

1. Создать новый 2D проект, в нём на панели *Hierarchy* нажать на знак «+» и выбрать из выпадающего меню раздел *2D Object* в нём *Sprites* и в нём *Square*.
Отлично у нас созданся квадрат, переименуй его в «Player», это будет игровой персонаж. Закрепи за ним камеру, если нужно.
2. Создай второй такой же квадрат, покрась его в другой цвет (свойство *Color* на панели *Inspector*) и вытяни его в ширину. Задай имя «Platform».
3. Добавь объекту «Platform» **BoxCollider2D**.
4. Добавь объекту «Player» компонент **Rigidbody2D** и **BoxCollider2D**.
5. Создай скрипт C# с названием **PlayerController** и добавь его к объекту Player.
6. Создай переменные дробного типа **moveForce**, **jumpForce**, и переменную типа **Rigidbody2D playerRb**.
7. Замени Update на FixedUpdate.
8. Вспомни как было реализовано движение на прошлом занятии и реализуй его в лево и право через метод **AddForce** (смотри лист «Материалы»).
9. Реализуй такое же движение для клавиши Space, где у вектора будет задано направление вверх (**Vector2.up**), используй переменную **jumpForce**.
10. Испытай свой проект, игровой персонаж должен двигаться влево и право и прыгать, даже воздухе.
11. Чтобы объект не отталкивался от воздуха нужно создать переменную типа bool для проверки нахождения объекта на земле, назови её **isGrounded**, пусть каждый раз при нажатии пробела ей присваивается значение **false**.
12. Модифицируй условие для прыжка, добавь в **if**, помимо считывания клавиши проверку **isGrounded** равно **true** или **false**(используй оператор **&&** - логическое «и»).
13. Осталось написать функцию проверяющую нахождение на земле. Создай новую функцию **void OnCollisionEnter2D(Collision2D collision)**, за пределами **FixedUpdate**, напиши внутри

```
if (collision.gameObject.layer == LayerMask.NameToLayer("Ground"))  
    isGrounded = true;
```

Теперь при столкновении с объектом у которого слой(Layer) называется «Ground» переменная **isGrounded** будет становиться **true**.

14. Создай объекту «Platform» слой «Ground» и примени его.
15. Отлично, должно всё работать, сейчас твой объект должен уметь прыгать и перемещаться в лево и право, подумай, как можно сделать так чтобы он не перемещался в лево и право в воздухе, попробуй реализовать.
16. Расставь платформы так как тебе хочется, тем самым создав свой уровень.
17. Чего-то не хватает, а именно победы и поражения, создай два новых квадрата, один поставь в конец, он будет точкой, до которой нужно добраться для победы, а второй растяни под всеми платформами, он будет смертельной зоной, при падении на которую уровень будет запускаться. Задай им цвет, если хочешь.
18. Обоим объектам добавь компонент **BoxCollider2D** и включи свойство *Is Trigger*.
19. Создай скрипт «DeathZone» и реализуй в нём перезапуск уровня при соприкосновении с объектом, для этого тебе понадобятся команды **OnTriggerEnter2D** и **SceneManager.LoadScene**, пояснения к ним есть в материалах к уроку. примени скрипт к объекту, который ты растянул под платформами.
20. Теперь создай скрипт WinPoint, этот скрипт будет грузить следующий уровень при соприкосновении с объектом, к которому он применён. Тебе понадобятся те же команды, что и в пункте выше, единственное у **SceneManager.LoadScene** к **buildIndex** будет прибавляться единица, то есть к номеру

текущей сцены будет прибавляться единица, тем самым мы получим следующий уровень(**buildIndex + 1**).

21. Осталось создать новую сцену и добавить её в *Scenes in Build* в разделе меню *Build Settings*.
22. Создай второй уровень и твоя игра готова.

Материалы

Переменные:

public Rigidbody2D playerRb; - переменная для работы с методами Rigidbody2D игрока

public float jumpForce = 500; - Сила прыжка

public float moveForce = 50; - Сила ходьбы

public bool isGrounded = false; - Проверка нахождения на земле

Функции:

private void FixedUpdate() – работает фиксированное количество кадров, которое можно задавать в настройках Unity

if (Input.GetKey(KeyCode.D))

{

playerRb.AddForce(Vector2.right * moveForce);

} – **AddForce** добавляет силу к объекту в направлении направо умноженном на значение силы, **Vector2.right * moveForce**.

private void OnCollisionEnter2D(Collision2D collision)

{

тело программы

} - функция на уровне void start и void update, срабатывает когда объект к которому применен скрипт, сталкивается с другим объектом.

void OnTriggerEnter2D(Collider2D other)

{

тело программы

} - функция на уровне **void start** и **void update**. Исполняется при условии, что коллайдер объекта пересекается с другим коллайдером.

if (other.tag == "Player")

{

тело программы

} - Если тэг объекта, с которым произошло столкновение "Player", сделай что-либо.

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex); - метод класса **SceneManger** загружающий текущую сцену. Для работы нужно вверху скрипта прописать **using UnityEngine.SceneManagement;**