



Presents



Unity2D Physics SideScroller Demo Project - FREE Asset Store Package -

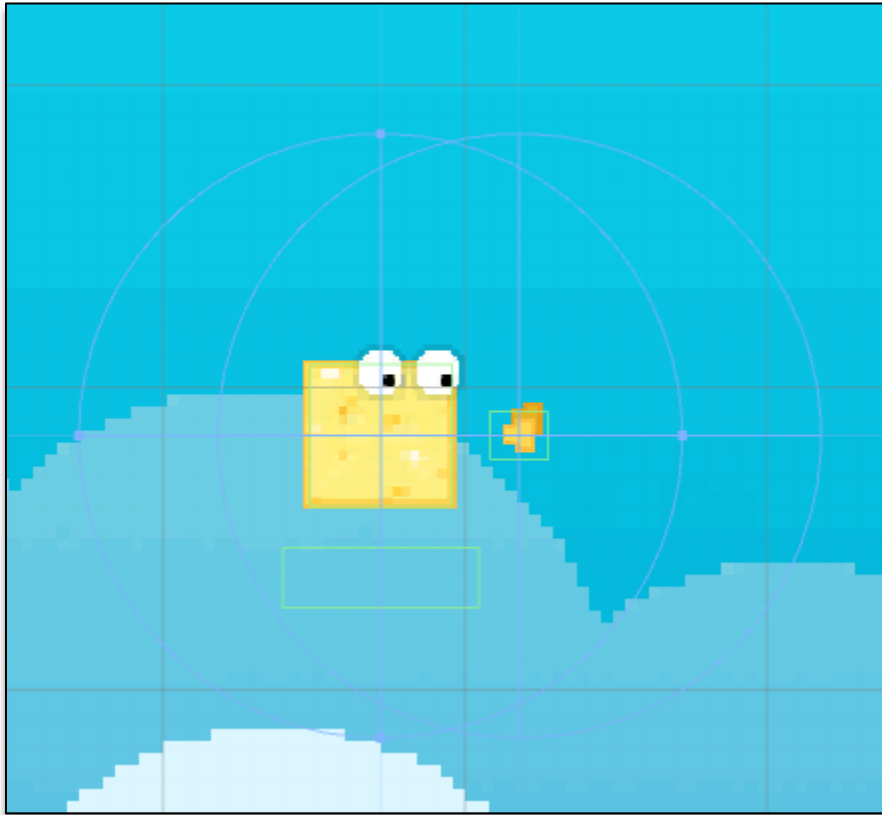
For More Info: <http://www.DevU.com/Unity>

See How to Make a Physics-Based 2D SideScrolling Platformer in Unity with C#!



This Project Demonstrates How To...

- **Work with Physics Colliders:**



- **Create Infinitely Tiling Backgrounds Along with Parallax Scrolling:**



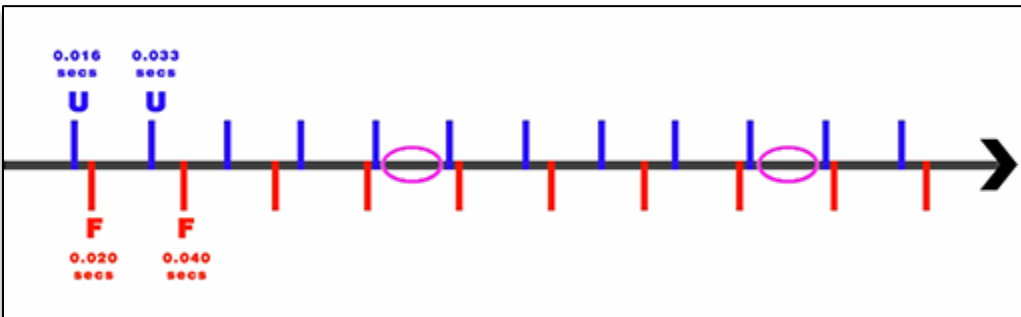
- Access the Physics Engine in Code:

```
public class PlayerController : MonoBehaviour
{
    Rigidbody2D Player;
    bool JumpActivated;

    void Update()
    {
        if (Input.GetButtonDown("Jump"))
            JumpActivated = true;
    }

    void FixedUpdate()
    {
        if (JumpActivated)
        {
            if (PlayerState.Instance.Vertical == Vertical.Grounded)
            {
                PlayerState.Instance.Vertical = Vertical.Airborne;
                Player.AddForce(new Vector2(0, 6), ForceMode2D.Impulse);
            }
            JumpActivated = false;
        }
    }
}
```

- Use FixedUpdate() and Update() in Tandem:



- Use Rigidbody Components that Respond to Forces and Gravity:



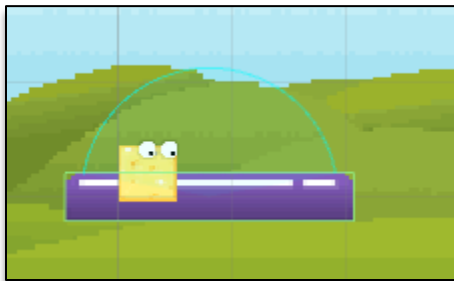
- Create Attacks like Punching, Stomping and Throwing Projectiles:



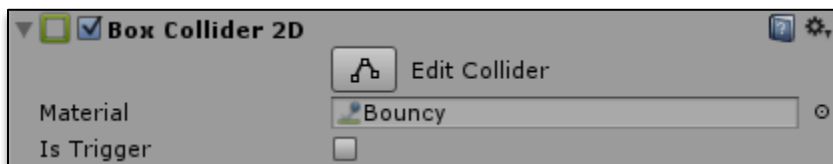
- Procedurally Generate Enemies and Platforms:



- Create Platforms with Physics Properties using Effectors:



- Work with Physics Materials:



- **Create Dynamic Camera Movement:**

```
if (CameraState == CameraState.Recentering)
{
    float x = Mathf.Lerp(transform.position.x, Player, 0.02f * Time.deltaTime * 60);
    transform.position = new Vector3(x, transform.position.y, transform.position.z);

    if (Math.Round(CheeseScreenPosition.x, 1) == 0.5f)
        CameraState = CameraState.Stationary;
}
```

- **Create OOP Enemy Classes:**

```
public class Enemy<T> where T : Enemy
{
    public GameObject GameObject;
    public T ScriptComponent;

    public Enemy(string name)
    {
        GameObject = new GameObject(name);
        ScriptComponent = GameObject.AddComponent<T>();
    }
}

public abstract class Enemy : MonoBehaviour
{
    protected int HP;

    public Rigidbody2D Body;
    public SpriteRenderer Sprite;
    public CircleCollider2D Collider;

    public int Speed;
    public int Direction;
```

- **Instantiate Enemies and Set Their Properties all Within Code:**

```
Enemy<Gigantor> giantGeorge = new Enemy<Gigantor>("GiantGeorge");
giantGeorge.ScriptComponent.Initialize(speed: 1, position: new Vector3(randomX, randomY, 1));
```

- Use Singletons and Enums to Create and Manage States:

```
public class PlayerState : MonoBehaviour
{
    private static PlayerState _instance;
    public static PlayerState Instance
    {
        get
        {
            if (_instance == null)
                _instance = new GameObject("PlayerState").AddComponent<PlayerState>();

            return _instance;
        }
    }

    public Horizontal Horizontal;
    public Vertical Vertical;
    public DirectionFacing DirectionFacing;
    public Attack Attack;
}
```

Also Learn How To...

- Display Text,
- Utilize Collision and Trigger Events,
- Create Scoring Mechanics,
- Reference Outside GameObjects in Code,
- Load Assets Dynamically at Runtime,
- Change Framerates for Testing Purposes,
- Create Animations Entirely with Code,
- Implement a Variety of Game Design Basics using Unity and C#!

For More Information...

See the “**Script Synopsis**” at the top of every script to better understand the purpose of each script used in this project.

For detailed **step-by-step video lesson tutorials** showing how this game was constructed - along with additional beginner-focused C# and Unity instruction - please see the course available at:

<http://courses.devu.com/courses/unity>

For more information on this course check out:

<http://www.devu.com/unity>

See Below for a List of all Lessons Included in the Course:

All Lessons for Introduction to Unity with C#

01. Who This Course is For, Message For Beginners	23. Static Fields and Methods	45. Scene Loading and Game Over Manager
02. Course Outline	24. Method Inputs and Returns	46. Understanding Properties
03. Installation and Getting Started	25. Reference vs Value Types	47. Controller Mapping and Input Manager
04. Starting the First Project	26. Intro to Polymorphism	48. Understanding Enums
05. Prototype Workflow	27. Navigating the Unity API	49. Dealing with Null References
06. Basic Code Review	28. Applying What You Learned and Refactoring	50. Handling Variable Framerates with Time. deltaTime
07. Understanding the Game Loop	29. Constructors, Local Variables in the Update Method	51. Preparing the Project for Final Build
08. Prototyping Continued	30. Creating Collectible Items and PowerUps	52. Final Build and Project Settings
09. C# Fundamentals and Hello World	31. Spawning and Managing Prefab PowerUps	53. Introduction to the Unity Physics Engine
10. Variables and Operations	32. Implementing PowerUp State Logic	54. Understanding FixedUpdate vs Update
11. Variables and Operations Continued, Math Operations	33. Displaying Text, OnGUI, Method Overloading	55. Movement using Physics, Singletons and Enums for Managing States
12. Floats, Booleans and Casting	34. Referencing Instantiated GameObjects, Parenting	56. Attack Script and Collision Events with OnCollisionEnter2D
13. If() Conditional Statements	35. Understanding the Lerp Method	57. Projectiles and Stomping Attack
14. If() Conditional Statements Continued	36. Creating Pseudo Animations in Code	58. Parallax Background and Scrolling Camera
15. Complex Evaluations and States	37. Understanding Generic Classes and Methods	59. Infinitely Tiling Background Sprites
16. Code Structure vs Style, Errors	38. Animations Using SpriteSheets and Animator	60. OOP Enemy Classes
17. Variable Scope	39. Working with Arrays and Loops	61. OOP Enemy Classes Continued
18. Object Oriented Programming Intro	40. Debugging Unity Projects with Visual Studio	62. Trigger Colliders and Dealing Damage
19. OOP, Access Modifiers, Instantiation	41. Camera Movement and LateUpdate	63. Multi-Dimensional Arrays, Procedural Platforms, Materials and Effectors
20. Object Containment and Method Returns	42. Playing Audio Clips	64. Finishing Touches
21. "Has-A" Object Containment	43. Routing Audio, Mixers and Effects	65. Series Wrap
22. "Is-A" Inheritance Containment	44. Add Scoring Mechanics and Enhancements	