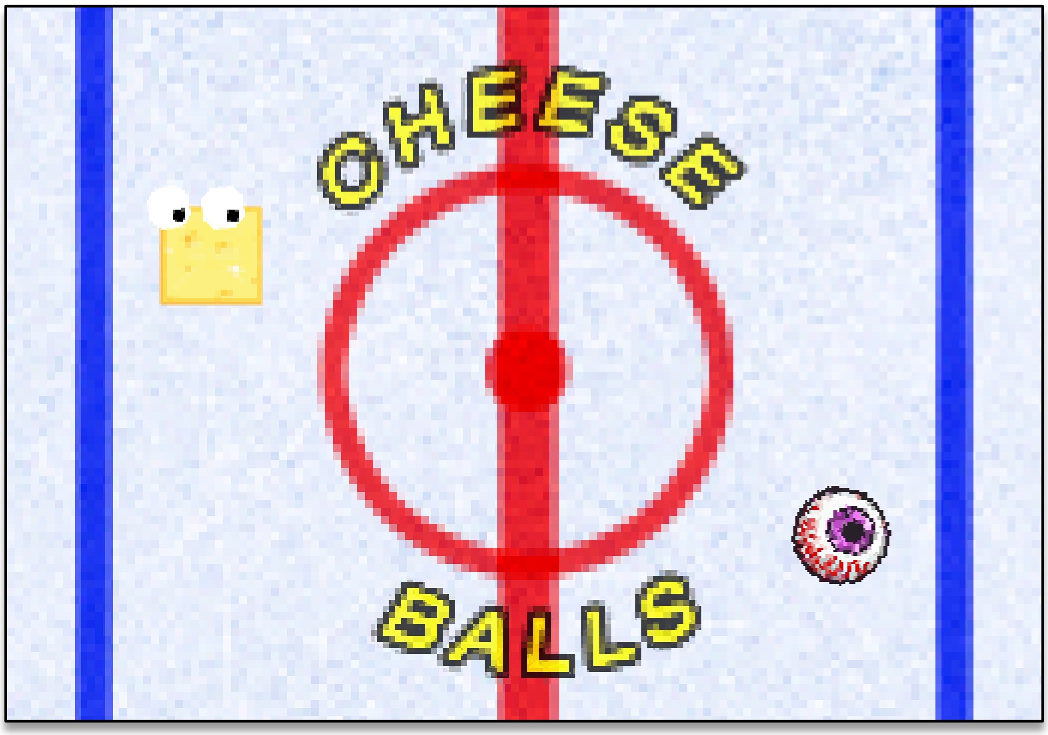
DEVELOPER UNIVERSITY

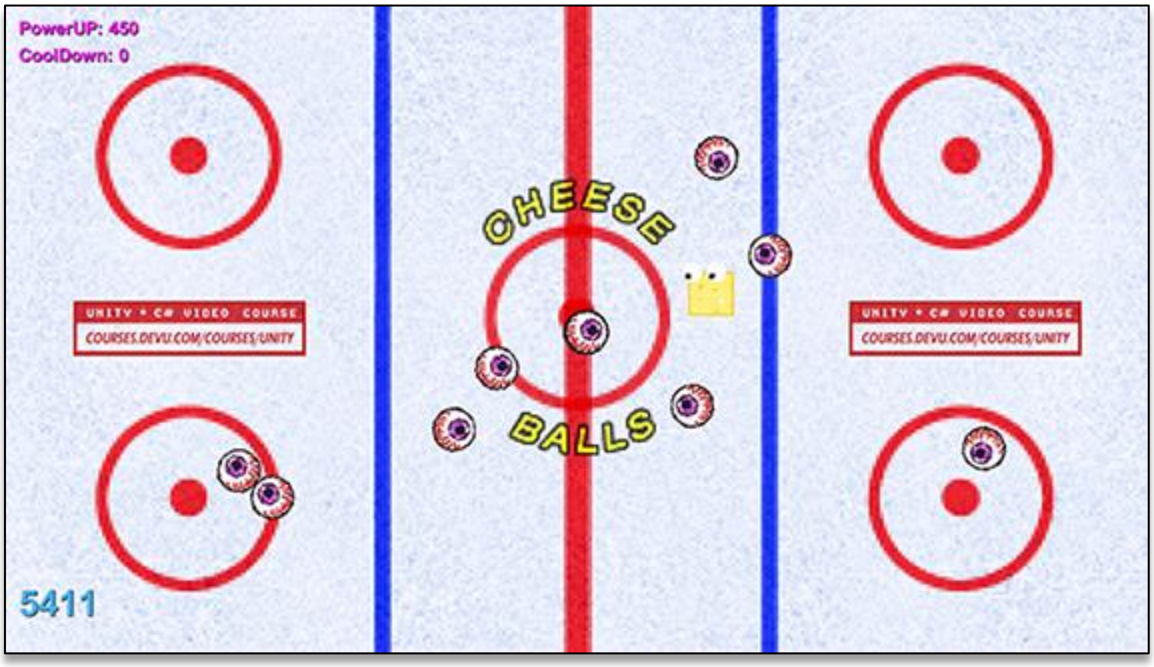
Presents



Unity2D Demonstration Project
- FREE Asset Store Package -

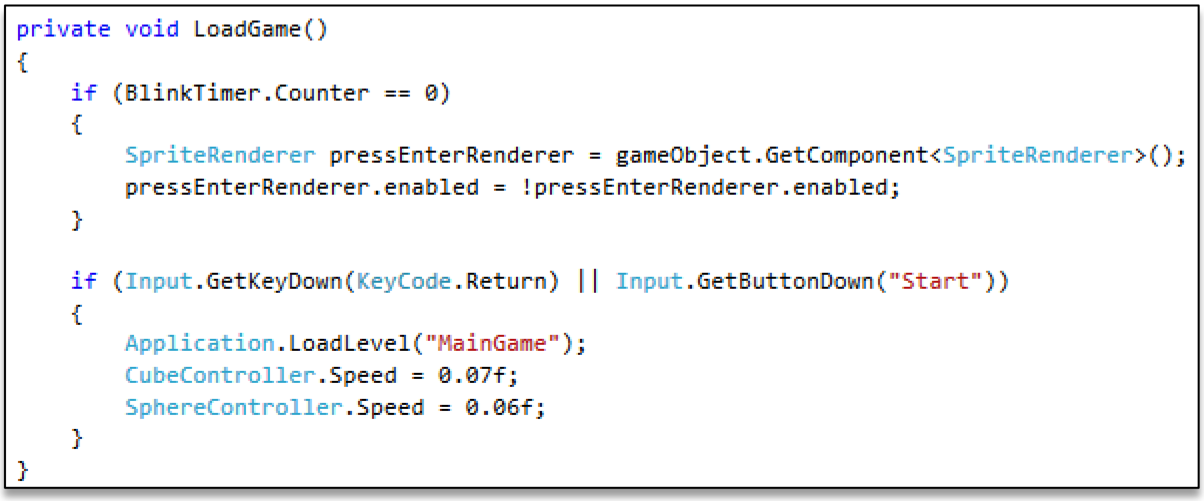
For More Info: http://www.DevU.com/Unity

Take a Peek Behind the Curtain and See How to Make a Fun and Simple 2D Action Game with Unity and C#!



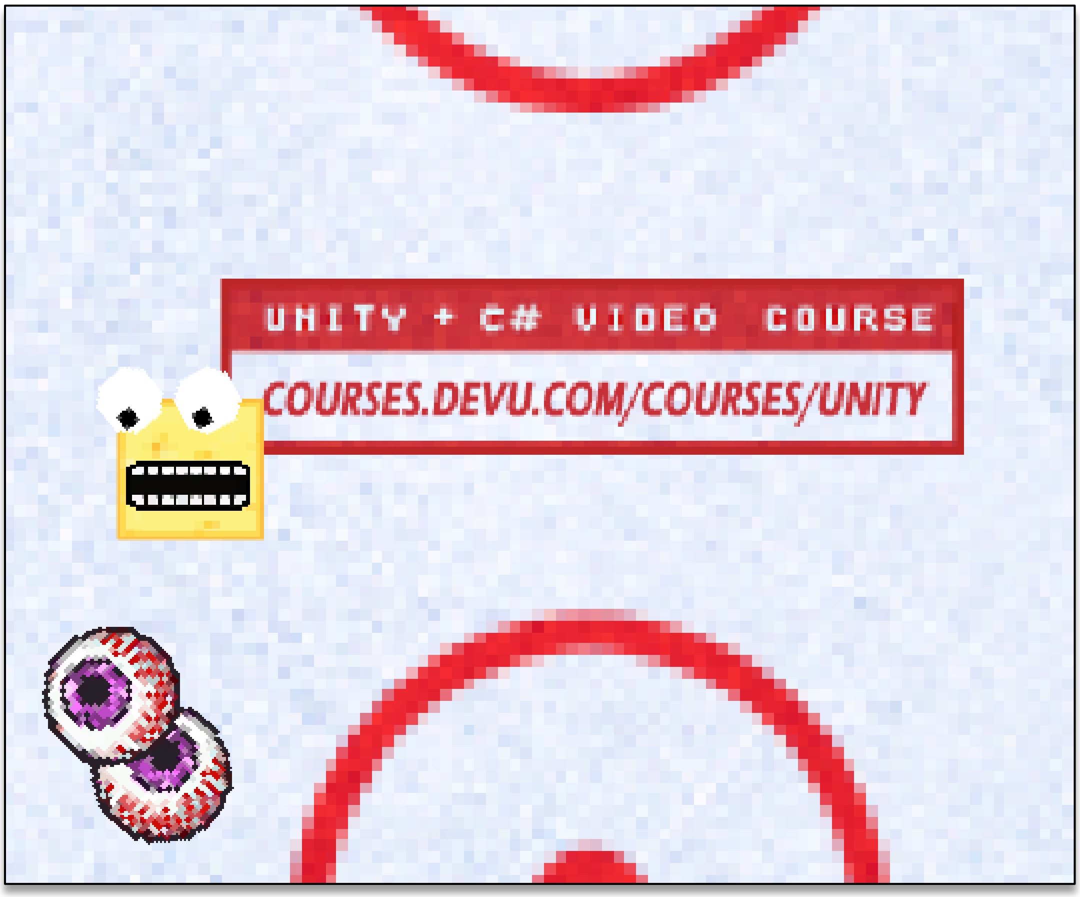
This Project Demonstrates How To…

- **Load a level (or, “Scene”) From another Level/Scene:**

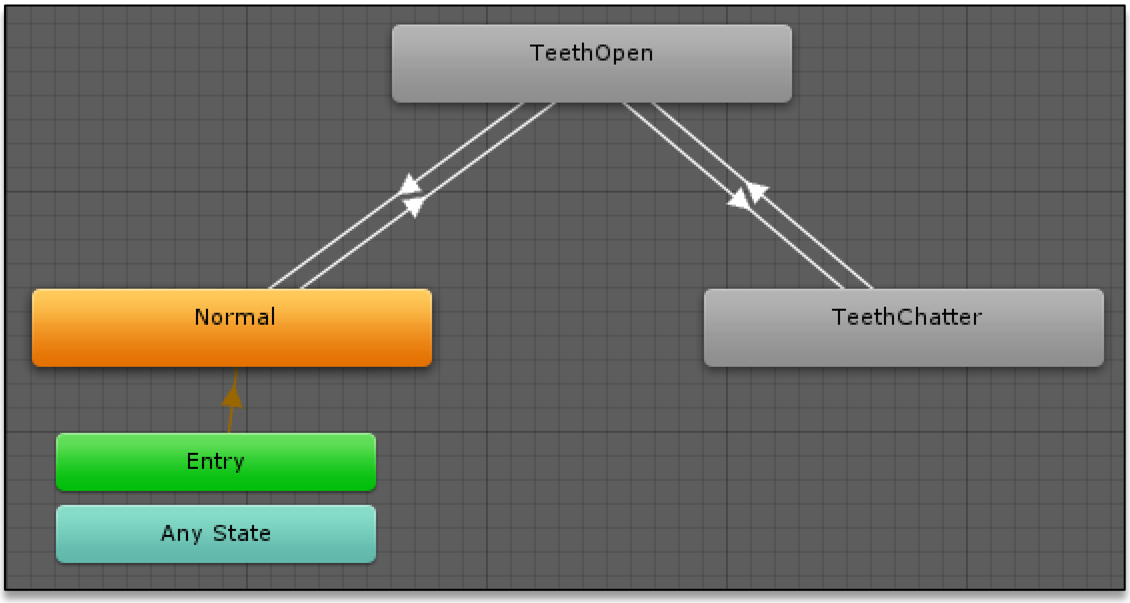


```
private void LoadGame()
{
    if (BlinkTimer.Counter == 0)
    {
        SpriteRenderer pressEnterRenderer = gameObject.GetComponent<SpriteRenderer>();
        pressEnterRenderer.enabled = !pressEnterRenderer.enabled;
    }

    if (Input.GetKeyDown(KeyCode.Return) || Input.GetButtonDown("Start"))
    {
        Application.LoadLevel("MainGame");
        CubeController.Speed = 0.07f;
        SphereController.Speed = 0.06f;
    }
}
```

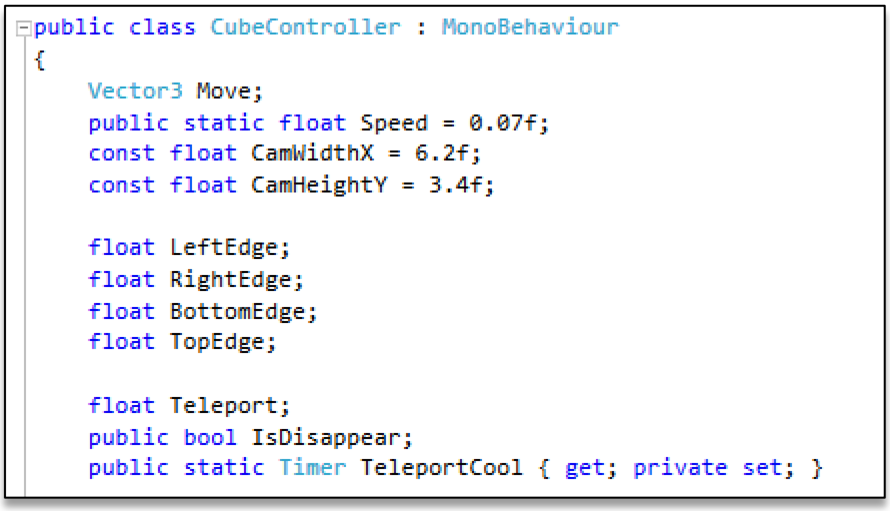
- **Create Animations with Spritesheets:**



- **Control Animations Using the Animator Component:**

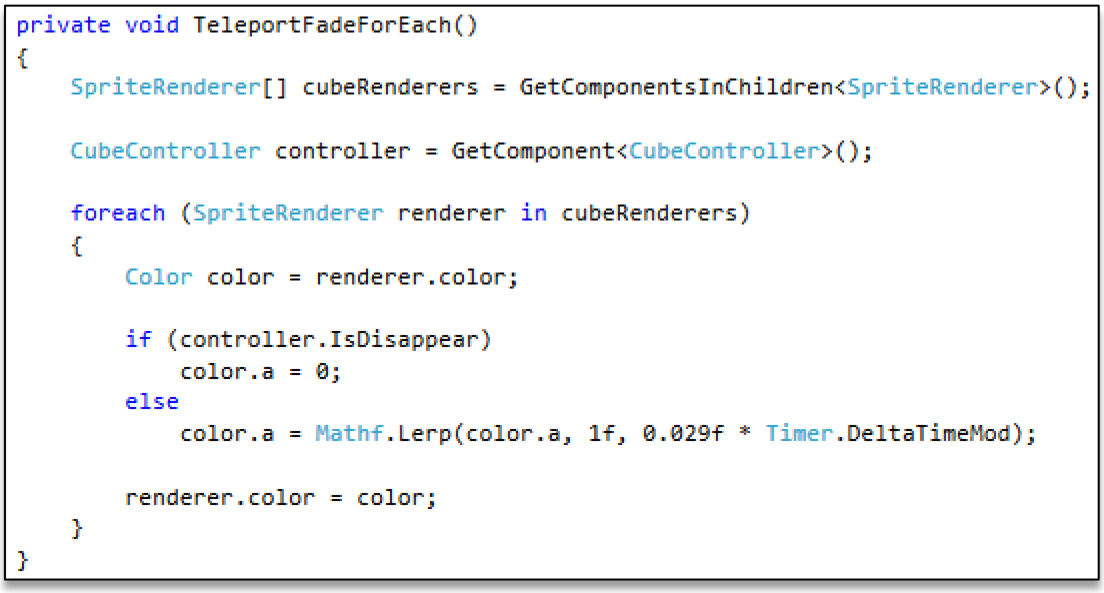


- **Control Player Movement and Behavior without Requiring Physics:**



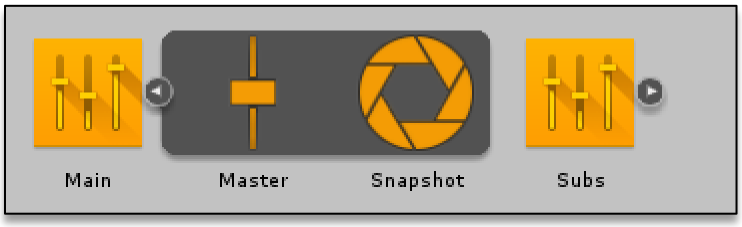
```
public class CubeController : MonoBehaviour
{
    Vector3 Move;
    public static float Speed = 0.07f;
    const float CamWidthX = 6.2f;
    const float CamHeightY = 3.4f;

    float LeftEdge;
    float RightEdge;
    float BottomEdge;
    float TopEdge;

    float Teleport;
    public bool IsDisappear;
    public static Timer TeleportCool { get; private set; }
```

- **Loop Through Components in Parent/Child GameObjects and Apply a Process to Each Component:**

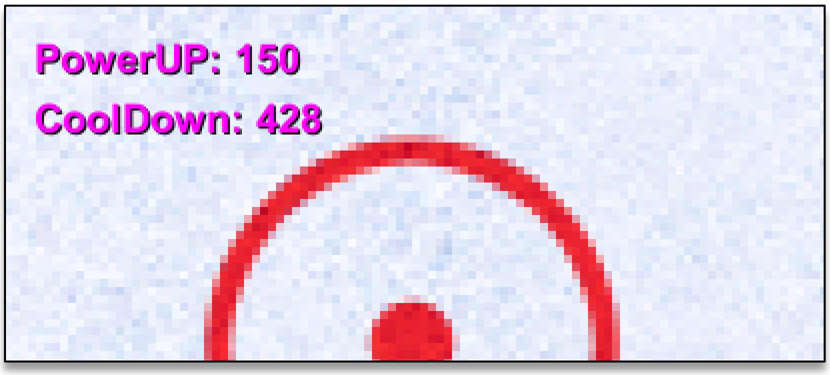


```
private void TeleportFadeForEach()
{
    SpriteRenderer[] cubeRenderers = GetComponentsInChildren<SpriteRenderer>();

    CubeController controller = GetComponent<CubeController>();

    foreach (SpriteRenderer renderer in cubeRenderers)
    {
        Color color = renderer.color;

        if (controller.IsDisappear)
            color.a = 0;
        else
            color.a = Mathf.Lerp(color.a, 1f, 0.029f * Timer.DeltaTimeMod);

        renderer.color = color;
    }
}
```

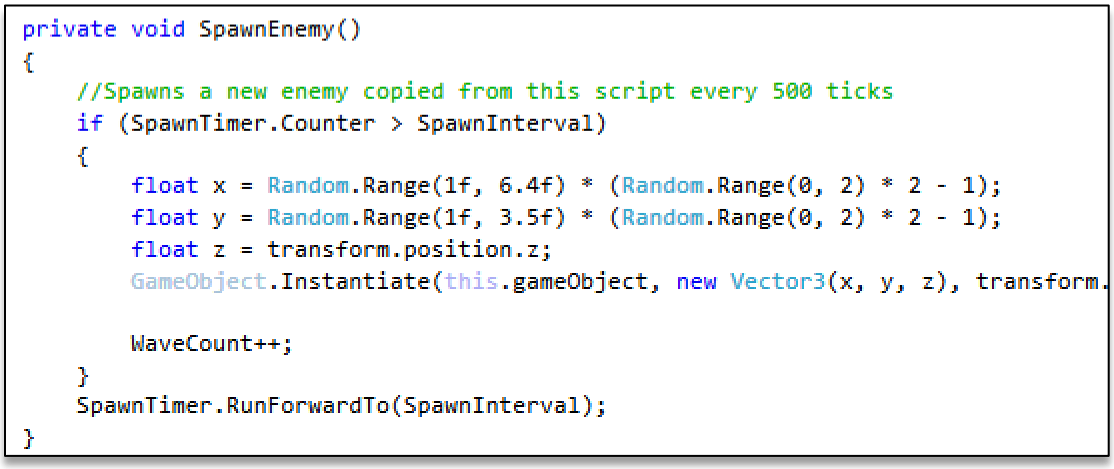
- **Work with Audio, Mixers and Effects:**



- **Display Text that Scales to the Size of the Screen:**

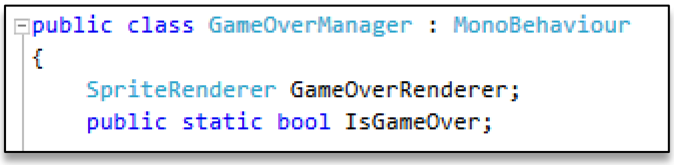


- **Spawn Enemies and Control their Behavior:**



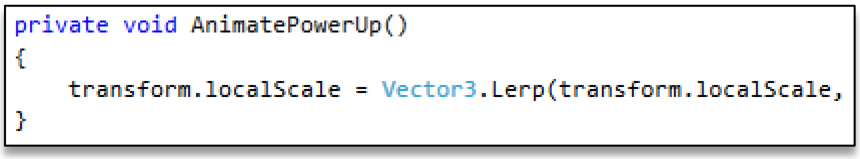
```
private void SpawnEnemy()
{
    //Spawns a new enemy copied from this script every 500 ticks
    if (SpawnTimer.Counter > SpawnInterval)
    {
        float x = Random.Range(1f, 6.4f) * (Random.Range(0, 2) * 2 - 1);
        float y = Random.Range(1f, 3.5f) * (Random.Range(0, 2) * 2 - 1);
        float z = transform.position.z;
        GameObject.Instantiate(this.gameObject, new Vector3(x, y, z), transform.

        WaveCount++;
    }
    SpawnTimer.RunForwardTo(SpawnInterval);
}
```

- **Manage Game State with a GameOverManager Script:**



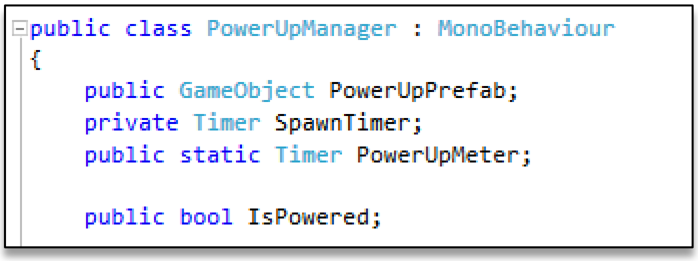
```
public class GameOverManager : MonoBehaviour
{
    SpriteRenderer GameOverRenderer;
    public static bool IsGameOver;
```

- **Use Various “Lerp” Techniques to Produce Change Across Frames:**



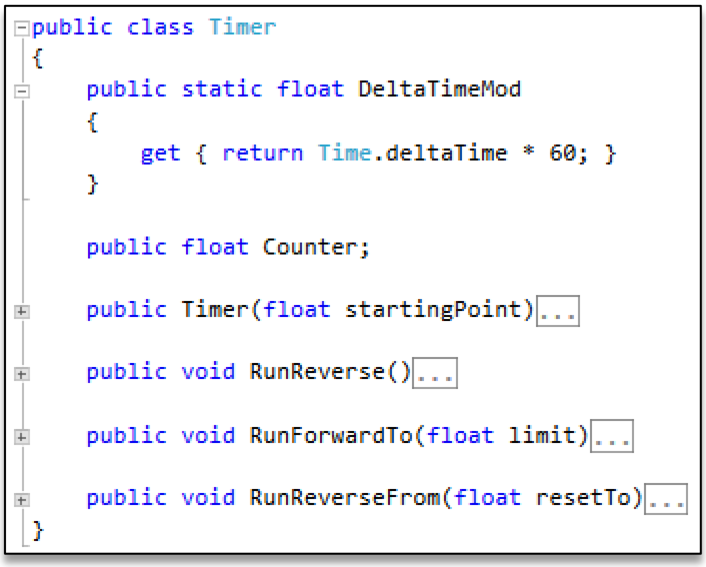
```
private void AnimatePowerUp()
{
    transform.localScale = Vector3.Lerp(transform.localScale,
}
```

- **Create Collectible Item Prefabs and Manage a “PowerUp” State:**



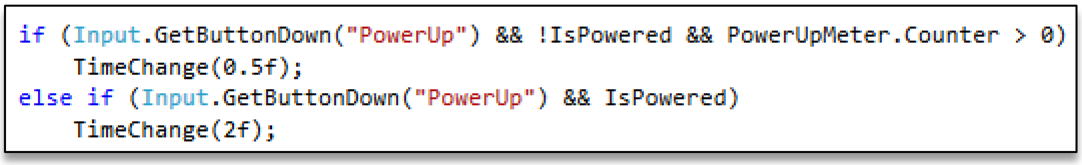
```
public class PowerUpManager : MonoBehaviour
{
    public GameObject PowerUpPrefab;
    private Timer SpawnTimer;
    public static Timer PowerUpMeter;

    public bool IsPowered;
```

- **Create a Custom Timer Class and Implement Time.deltaTime to Make the Game Framerate Independent:**



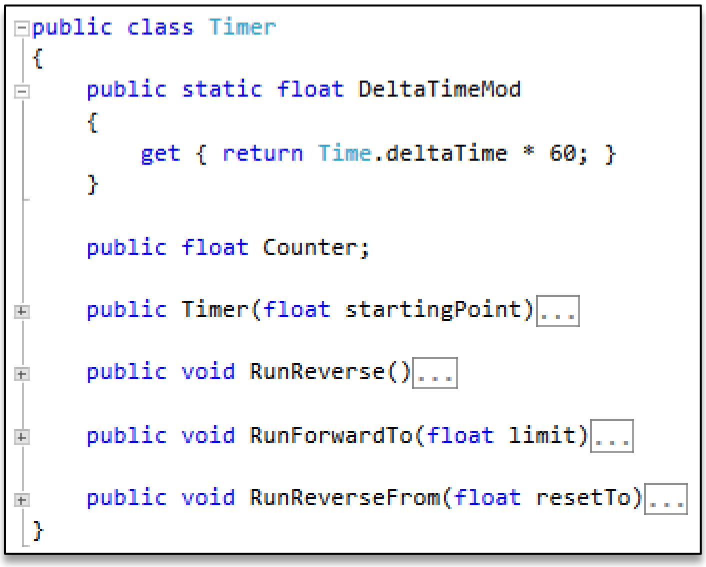
```
public class Timer
{
    public static float DeltaTimeMod
    {
        get { return Time.deltaTime * 60; }
    }

    public float Counter;

    public Timer(float startingPoint)...

    public void RunReverse()...

    public void RunForwardTo(float limit)...

    public void RunReverseFrom(float resetTo)...
}
```

- **Change the Game's Timescale (for example: by creating a slowed-down "bullet-time" effect or to pause the game):**



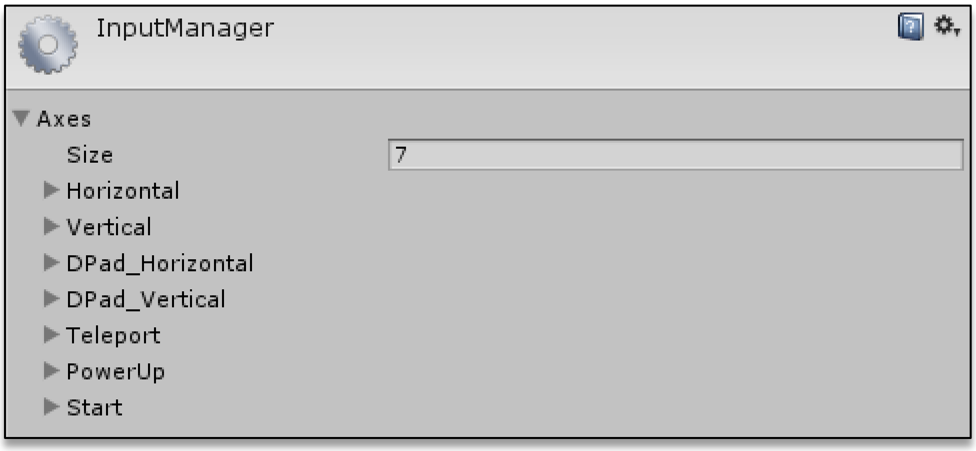
```
if (Input.GetButtonDown("PowerUp") && !IsPowered && PowerUpMeter.Counter > 0)
    TimeChange(0.5f);
else if (Input.GetButtonDown("PowerUp") && IsPowered)
    TimeChange(2f);
```

- **Handle Camera Movement, Including Zooming In and Out of the Action:**



```
public class Timer
{
    public static float DeltaTimeMod
    {
        get { return Time.deltaTime * 60; }
    }

    public float Counter;

    public Timer(float startingPoint)...

    public void RunReverse()...

    public void RunForwardTo(float limit)...

    public void RunReverseFrom(float resetTo)...
}
```

- **Implement Xbox Controller input:**



```
InputManager

▼ Axes
    Size            7
  ▶ Horizontal
  ▶ Vertical
  ▶ DPad_Horizontal
  ▶ DPad_Vertical
  ▶ Teleport
  ▶ PowerUp
  ▶ Start
```

Also Learn How To…

- **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

All Lessons for Introduction to Unity with C#