Game Programming Using Unity 3D

Lesson 16: Pick-up Items

Ferdinand Joseph Fernandez

Chief Technological Officer, Dreamlords Digital Inc.

Admin and Co-founder, Unity Philippines Users Group

December 2011

Overview

We'll add a simple functionality in our game: the zombies will drop medkits on the ground by chance when you kill them. Running over one will heal your health.

Creating The Medikit

Our medkit will be a simple cube. We'll make a prefab for this so we can easily instantiate it later.

The idea is we check if the player collided with the medkit. If it did, then we heal the player.

Create a new C# script. Name it “Medkit”. Add this code:

```
using UnityEngine;
using System.Collections;

public class Medkit : MonoBehaviour
{
	void OnCollisionEnter(Collision c)
	{
		if (c.transform.root.tag == "Player")
		{
			Debug.Log("collided with the player!");
		}
	}
}
```

We just check if we collided with the player for now. Its important to use transform.root and not just transform. Transform.root gives the topmost parent of a game object. It could happen that one of the child ragdoll colliders were the one that collided with the medkit, but we need the topmost Player game object. So we use transform.root.

Test this code.

Open your MainScene.unity. Create a cube (GameObject > Create Other > Cube). Place it nearby the player. Attach the Medkit to it. Attach a rigidbody to the cube as well.

Run the game. Let yourself get hurt, then run over to the cube. It should display the message in the console.

Now we need a function that will heal the player. Like we have a function for damaging, we'll add a function for healing.

Add this code to the Health script:

```
using UnityEngine;
using System.Collections;

public class Health : MonoBehaviour
{
	[SerializeField]
	int _maximumHealth = 100;

	int _currentHealth = 0;

```

```
    override public string ToString()
    {
        return _currentHealth + " / " + _maximumHealth;
    }

    public bool IsDead { get{ return _currentHealth <= 0; } }

    Renderer _renderer;

    PlayerStats _playerStats;

    [SerializeField]
    AudioClip[] _hitSounds;

    [SerializeField]
    AudioClip _deathSound;

    void Start()
    {
        _renderer = GetComponentInChildren<Renderer>();
        _currentHealth = _maximumHealth;

        GameObject player = GameObject.FindGameObjectWithTag("Player");
        _playerStats = player.GetComponent<PlayerStats>();
    }

    public void Heal(int healAmount)
    {
        _currentHealth += healAmount;

        if (_currentHealth > _maximumHealth)
        {
            _currentHealth = _maximumHealth;
        }
    }

    public void Damage(int damageValue)
    {
        _currentHealth -= damageValue;

        if (_currentHealth < 0)
        {
            _currentHealth = 0;
        }
        else
        {
            if (_hitSounds != null && _hitSounds.Length > 0)
            {
                AudioClip soundToUse = _hitSounds[Random.Range(0, _hitSounds.Length)];
                audio.clip = soundToUse;
                audio.Play();
            }
        }

        if (_currentHealth == 0)
        {
            if (_deathSound != null)
            {
                audio.clip = _deathSound;
                audio.Play();
            }

            Animation a = GetComponentInChildren<Animation>();
            a.Stop();

            if (tag == "Player")
            {
                Destroy(GetComponent<PlayerMovement>());
                Destroy(GetComponent<PlayerAnimation>());
                Destroy(GetComponent<RifleWeapon>());
            }
            else // its an enemy
            {
                _playerStats.ZombiesKilled++;
                EnemySpawnManager.OnEnemyDeath();
                Destroy(GetComponent<EnemyMovement>());
                Destroy(GetComponentInChildren<EnemyAttack>());
            }

            Destroy(GetComponent<CharacterController>());

            Ragdoll r = GetComponent<Ragdoll>();
            if (r != null)
            {
                r.OnDeath();
            }
        }
    }

    void Update()
    {
        if (IsDead && !_renderer.isVisible)
        {
            Destroy(gameObject);
        }
    }
}
```

Now go back to the Medkit script and we'll use that new function:

```
using UnityEngine;
using System.Collections;

public class Medkit : MonoBehaviour
{
    [SerializeField]
    int _healAmount = 50;

    void OnCollisionEnter(Collision c)
    {
        if (c.transform.root.tag == "Player")
        {
            Health playerHealth = c.transform.root.GetComponent<Health>();
            playerHealth.Heal(_healAmount);
```

```
                Destroy(gameObject);
            }
        }
}
```

As well as healing the player, we delete the medkit afterwards.

Test the game again. Our medkit should work properly now.

Creating The Medkit When Killing Enemies

Now all that's left is to instantiate this medkit by chance when killing the enemy.

We'll need to turn the cube medkit into a prefab first. Rename the cube into “Medkit”. Drag it into yoru Project View's Prefabs folder to turn it into a prefab.

Create a new C# script. Name it “EnemyDrops”. This script will handle making the medkit when the zombie dies.

Add this code:

```
using UnityEngine;
using System.Collections;

public class EnemyDrops : MonoBehaviour
{
    [SerializeField]
    GameObject _dropItemPrefab;

    public void OnDeath()
    {
        Instantiate(_dropItemPrefab, transform.position, transform.rotation);
    }
}
```

We have a public function OnDeath, which is meant to be called when this enemy dies.

Open the Health script. That will be where we'll call the OnDeath function. Add this code:

```
                                        .
                                        .
                                        .
    public void Damage(int damageValue)
    {
        _currentHealth -= damageValue;

        if (_currentHealth < 0)
        {
            _currentHealth = 0;
        }
        else
        {
            if (_hitSounds != null && _hitSounds.Length > 0)
            {
                AudioClip soundToUse = _hitSounds[Random.Range(0, _hitSounds.Length)];
                audio.clip = soundToUse;
                audio.Play();
            }
        }

        if (_currentHealth == 0)
        {
            if (_deathSound != null)
            {
                audio.clip = _deathSound;
                audio.Play();
            }

            Animation a = GetComponentInChildren<Animation>();
            a.Stop();

            if (tag == "Player")
            {
                Destroy(GetComponent<PlayerMovement>());
                Destroy(GetComponent<PlayerAnimation>());
                Destroy(GetComponent<RifleWeapon>());
            }
            else // its an enemy
            {
                _playerStats.ZombiesKilled++;
                EnemySpawnManager.OnEnemyDeath();
                Destroy(GetComponent<EnemyMovement>());
                Destroy(GetComponentInChildren<EnemyAttack>());

                EnemyDrops d = GetComponent<EnemyDrops>();
                d.OnDeath();
            }

            Destroy(GetComponent<CharacterController>());

            Ragdoll r = GetComponent<Ragdoll>();
            if (r != null)
            {
                r.OnDeath();
            }
        }
    }

    void Update()
    {
        if (IsDead && !_renderer.isVisible)
        {
            Destroy(gameObject);
        }
    }
}
```

Now attach the EnemyDrops script to your Enemy prefab. Assign the Medkit prefab to its “Drop Item Prefab” slot.

When you kill an enemy, it drops a medkit.

Now we just need to control the chance at which this appears.

Open the EnemyDrops script and add this code:

```
using UnityEngine;
using System.Collections;

public class EnemyDrops : MonoBehaviour
{
	[SerializeField]
	GameObject _dropItemPrefab;

	[SerializeField]
	float _chanceToDrop = 50.0f;

	public void OnDeath()
	{
		if (Random.Range(0.0f, 100.0f) <= _chanceToDrop)
		{
			Instantiate(_dropItemPrefab, transform.position, transform.rotation);
		}
	}
}
```

Our instantiation code is now enclosed in an if-statement controlling when it gets called. The if-statement simply generates a random number from 0 to 100, and compares that to a user-specified treshold. Adjusting this treshold effectively adjusts the chance at which the drop item appears.

Playing A Sound When The Player Is Healed

Playing a sound is a simple matter of changing which AudioClip to play then playing it at the right moment.

Open the Health script and add this code:

```
using UnityEngine;
using System.Collections;

public class Health : MonoBehaviour
{
	[SerializeField]
	int _maximumHealth = 100;

	int _currentHealth = 0;

	override public string ToString()
	{
		return _currentHealth + " / " + _maximumHealth;
	}

	public bool IsDead { get{ return _currentHealth <= 0; } }

	Renderer _renderer;

	PlayerStats _playerStats;

	[SerializeField]
	AudioClip[] _hitSounds;

```

```
    [SerializeField]
    AudioClip _deathSound;

    [SerializeField]
    AudioClip _healSound;

    void Start()
    {
        _renderer = GetComponentInChildren<Renderer>();
        _currentHealth = _maximumHealth;

        GameObject player = GameObject.FindGameObjectWithTag("Player");
        _playerStats = player.GetComponent<PlayerStats>();
    }

    public void Heal(int healAmount)
    {
        _currentHealth += healAmount;

        if (_currentHealth > _maximumHealth)
        {
            _currentHealth = _maximumHealth;
        }

        if (_healSound != null)
        {
            audio.clip = _healSound;
            audio.Play();
        }
    }

    public void Damage(int damageValue)
    {
        _currentHealth -= damageValue;

        if (_currentHealth < 0)
        {
            _currentHealth = 0;
        }
        else
        {
            if (_hitSounds != null && _hitSounds.Length > 0)
            {
                AudioClip soundToUse = _hitSounds[Random.Range(0, _hitSounds.Length)];
                audio.clip = soundToUse;
                audio.Play();
            }
        }

        if (_currentHealth == 0)
        {
            if (_deathSound != null)
            {
                audio.clip = _deathSound;
                audio.Play();
            }

            Animation a = GetComponentInChildren<Animation>();
            a.Stop();

            if (tag == "Player")
            {
                Destroy(GetComponent<PlayerMovement>());
                Destroy(GetComponent<PlayerAnimation>());
                Destroy(GetComponent<RifleWeapon>());
            }
            else // its an enemy
            {
                _playerStats.ZombiesKilled++;
                EnemySpawnManager.OnEnemyDeath();
                Destroy(GetComponent<EnemyMovement>());
                Destroy(GetComponentInChildren<EnemyAttack>());

                EnemyDrops d = GetComponent<EnemyDrops>();
                d.OnDeath();
            }

            Destroy(GetComponent<CharacterController>());

            Ragdoll r = GetComponent<Ragdoll>();
            if (r != null)
            {
                r.OnDeath();
            }
        }
    }

    void Update()
    {
        if (IsDead && !_renderer.isVisible)
        {
            Destroy(gameObject);
        }
    }
}
```

The code here is largely similar to the code for playing death or hit sounds.

You Lesson Assets folder should come with a “health1.wav” that would be suitable as a heal sound.

In Conclusion...

There aren't much new concepts introduced here. Simply using already used ideas (instantiation, random number generation) and using them in various ways can create new functionality.