unity

3.4 Particles and Sound Effects

Steps:

Step 1: Customize an explosion particle

Step 2: Play the particle on collision

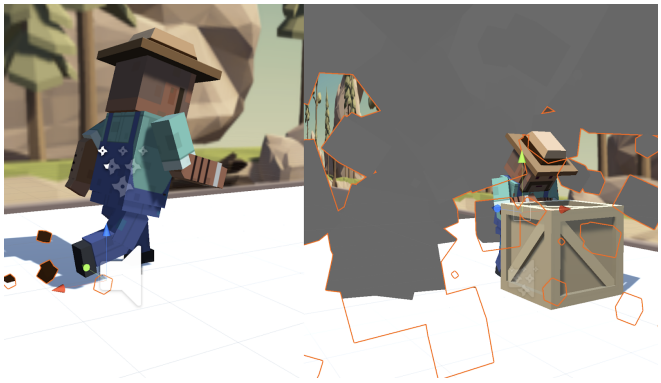
Step 3: Add a dirt splatter particle

Step 4: Add music to the camera object

Step 5: Declare variables for Audio Clips

Step 6: Play Audio Clips on jump and crash

Example of project by end of lesson



Length: 60 minutes

Overview: This game is looking extremely good, but it's missing something critical: Sound effects and Particle effects! Sounds and music will breathe life into an otherwise silent game world, and particles will make the player's actions more dynamic and eye-popping. In this lesson, we will add cool sounds and particles when the character is running, jumping, and crashing.

Project Outcome: Music will play as the player runs through the scene, kicking up dirt particles in a spray behind their feet. A springy sound will play as they jump and a boom will play as they crash, bursting in a cloud of smoke particles as they fall over.

Learning Objectives: By the end of this lesson, you will be able to:

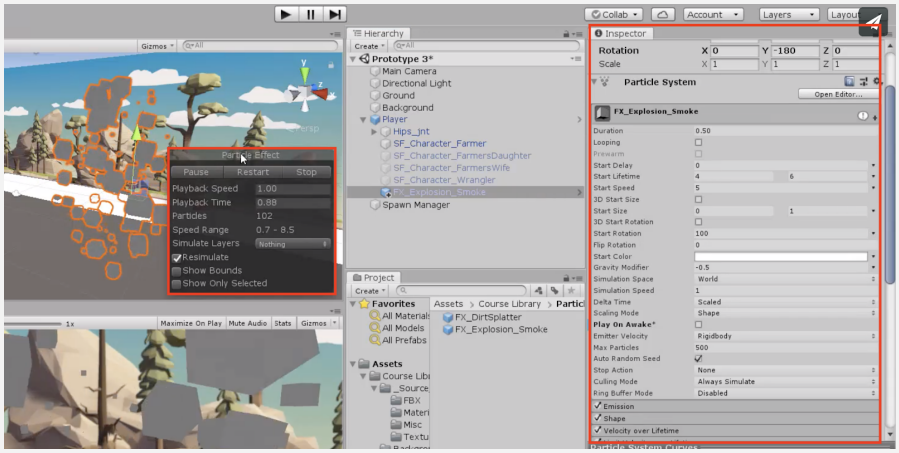
- Attach particle effects as children to game objects
- Stop and play particle effects to correspond with character animation states
- Work with Audio Sources and Listeners to play background music
- Add sound effects to add polish to your project

Step 1: Customize an explosion particle

The first particle effect we should add is an explosion for when the player collides with an obstacle.

1. From the *Course Library > Particles*, drag **FX_Explosion_Smoke** into the hierarchy, then use the **Play** / **Restart** / **Stop** buttons to preview it
2. Play around with the **settings** to get your **particle system** the way you want it
3. Make sure to **uncheck** the **Play on Awake** setting
4. Drag the **particle** onto your player to make it a **child object**, then position it relative to the player

- **New Concept:** Particle Effects
- **Warning:** Don't go crazy customizing your particle effects, you could easily get sidetracked
- **New Concept:** Child objects with relative positions
- **Tip:** Hovering over the settings while editing your particle provides great tool tips



Step 2: Play the particle on collision

We discovered the particle effects and found an explosion for the crash, but we need to assign it to the Player Controller and write some new code in order to play it.

1. In **PlayerController.cs**, declare a new ***public ParticleSystem explosionParticle;***
2. In the Inspector, assign the **explosion** to the **explosion particle** variable
3. In the **if-statement** where the player collides with an obstacle, call ***explosionParticle.Play();***, then test and tweak the **particle properties**

- **New Function:** particle.Play()

```
public ParticleSystem explosionParticle;

private void OnCollisionEnter(Collision collision other) {
  if (other.gameObject.CompareTag("Ground")) {
    isOnGround = true;
  } else if (other.gameObject.CompareTag("Obstacle")) {
    ... explosionParticle.Play(); } }
```

Step 3: Add a dirt splatter particle

The next particle effect we need is a dirt splatter, to make it seem like the player is kicking up ground as they sprint through the scene. The trick is that the particle should only play when the player is on the ground.

1. Drag **FX_DirtSplatter** as the Player's **child object**, reposition it, rotate it, and edit its settings
2. Declare a new ***public ParticleSystem dirtParticle;***, then **assign** it in the Inspector
3. Add ***dirtParticle.Stop();*** when the player jumps or collides with an **obstacle**
4. Add ***dirtParticle.Play();*** when the player lands on the **ground**

- **New Function:** particle.Stop()

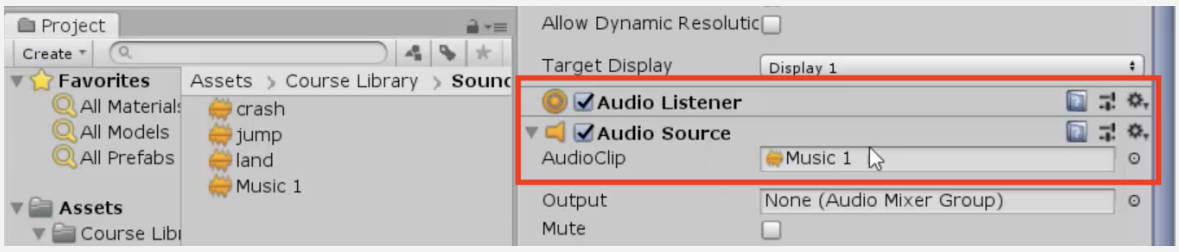
```
public ParticleSystem dirtParticle

void Update() {
  if (Input.GetKeyDown(KeyCode.Space) && isOnGround && !gameOver) {
    ... dirtParticle.Stop(); } }

private void OnCollisionEnter(Collision collision other) {
  if (other.gameObject.CompareTag("Ground")) { ... dirtParticle.Play();
  } else if (other.gameObject.CompareTag("Obstacle")) {  ... dirtParticle.Stop(); } }
```

Step 4: Add music to the camera object

Our particle effects are looking good, so it's time to move on to sounds! In order to add music, we need to attach sound component to the camera. After all, the camera is the eyes AND the ears of the scene.

1. Select the Main **Camera** object, then *Add Component > Audio Source*
2. From *Course Library > Sound*, drag a **music clip** onto the **AudioClip** variable in the inspector
3. Reduce the **volume** so it will be easier to hear **sound effects**
4. Check the **Loop** checkbox

- **New Concept:** Audio Listener and Audio Sources
- **Tip:** Music shouldn't appear to come from a particular location in 3D space, which is why we're adding it directly to the camera

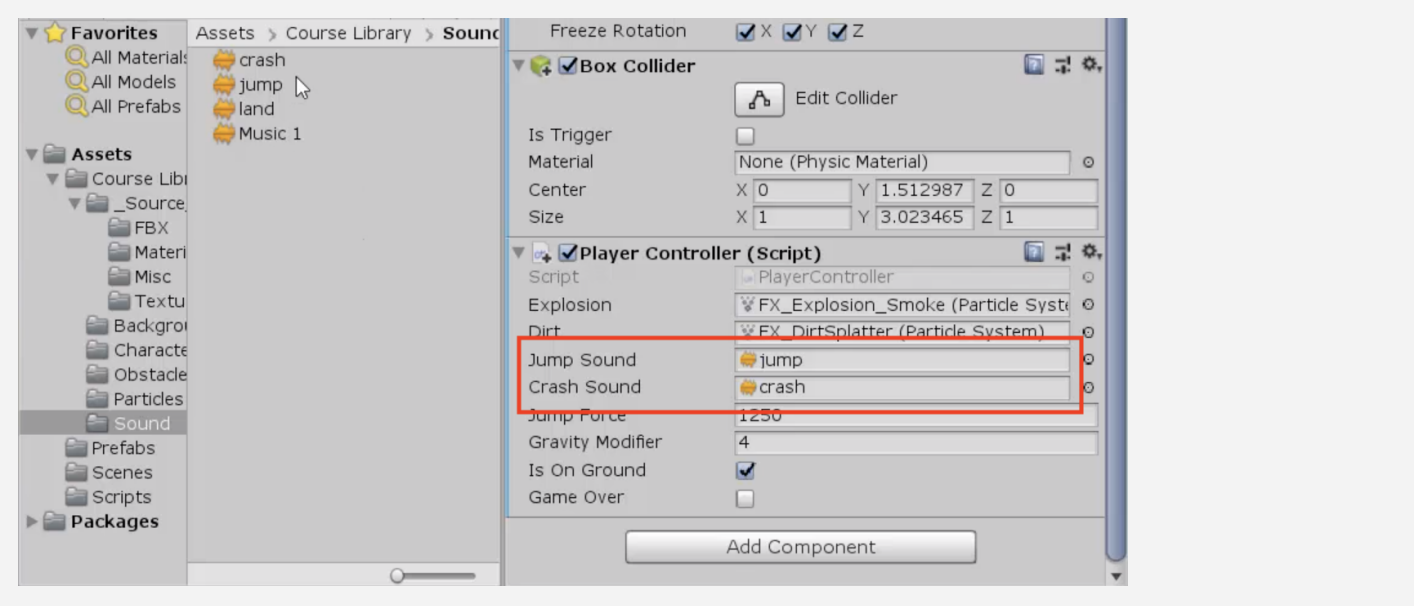


Step 5: Declare variables for Audio Clips

Now that we've got some nice music playing, it's time to add some sound effects. This time audio clips will emanate from the player, rather than the camera itself.

1. In **PlayerController.cs**, declare a new ***public AudioClip jumpSound;*** and a new ***public AudioClip crashSound;***
2. From *Course Library > Sound*, drag a clip onto each new **sound** variable in the inspector

- **Tip:** Adding sound effects is not as simple as adding music, because we need to trigger the events in our code



Step 6: Play Audio Clips on jump and crash

We've assigned audio clips to the jump and the crash in PlayerController. Now we need to play them at the right time, giving our game a full audio experience

1. Add an **Audio Source** component to the **player**
2. Declare a new ***private AudioSource playerAudio;*** and initialize it as ***playerAudio = GetComponent<AudioSource>();***
3. Call ***playerAudio.PlayOneShot(jumpSound, 1.0f);*** when the character **jumps**
4. Call ***playerAudio.PlayOneShot(crashSound, 1.0f);*** when the character **crashes**

- **Don't worry:** Declaring a new AudioSource variable is just like declaring a new Animator or RigidBody

```
private AudioSource playerAudio;

void Start() {
  ... playerAudio = GetComponent<AudioSource>(); }

void Update() {
  if (Input.GetKeyDown(KeyCode.Space) && isOnGround && !gameOver) {
    ... playerAudio.PlayOneShot(jumpSound, 1.0f); } }

private void OnCollisionEnter(Collision collision other) {
  ...
  } else if (other.gameObject.CompareTag("Obstacle"))
  {  ... playerAudio.PlayOneShot(crashSound, 1.0f); } }
```

Lesson Recap

New Functionality	• Music plays during the game • Particle effects at the player's feet when they run • Sound effects and explosion when the player hits an obstacle
New Concepts and Skills	• Particle systems • Child object positioning • Audio clips and Audio sources • Play and stop sound effects