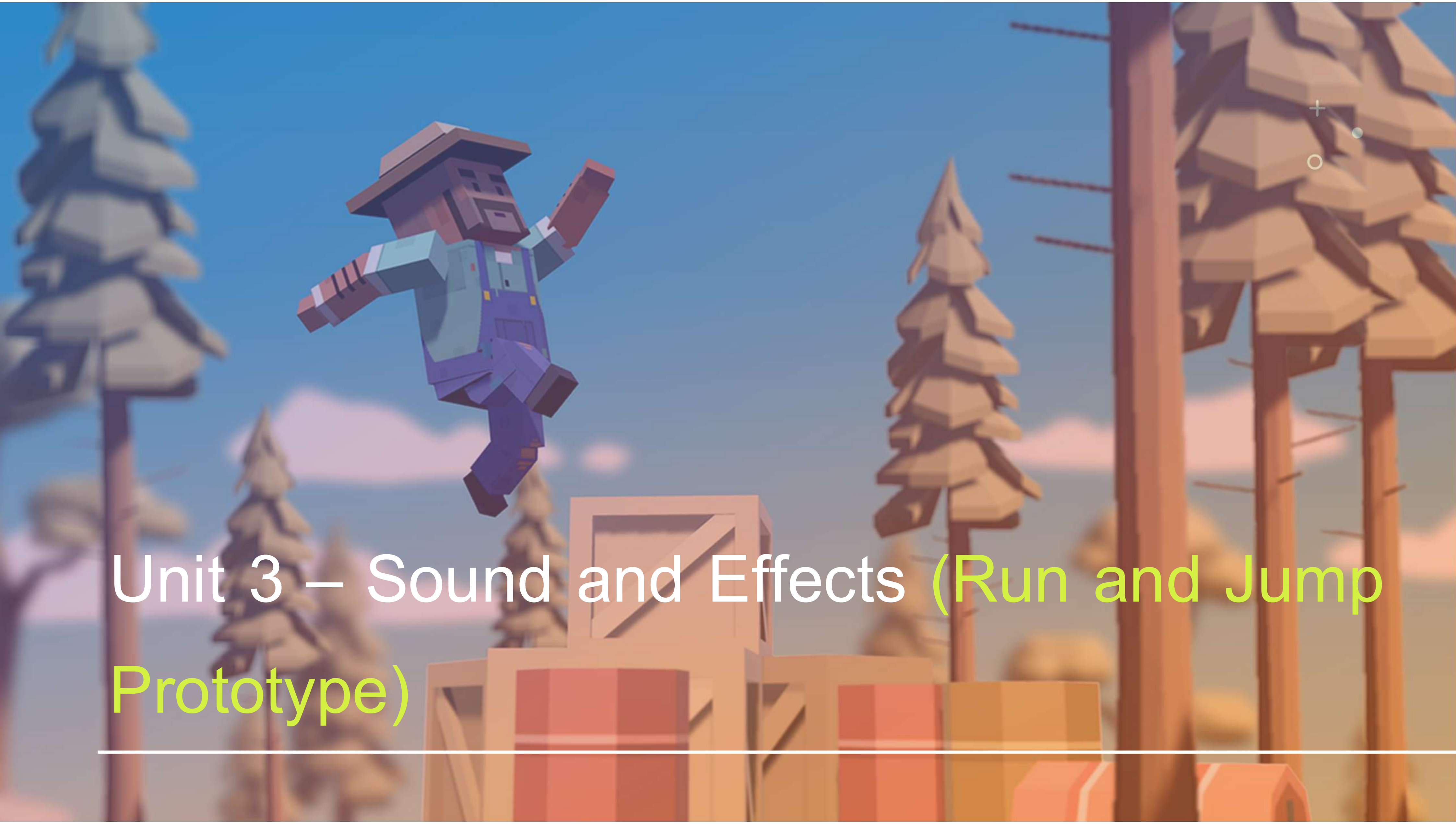
การเขียนโปรแกรมคอมพิวเตอร์ขั้นสูงเพื่อควบคุมอุปกรณ์

Advance Computer Programming

[สัปดาห์ที่ 7]

สอนโดย พงศธร เกียรติเจริญพร (มิว)

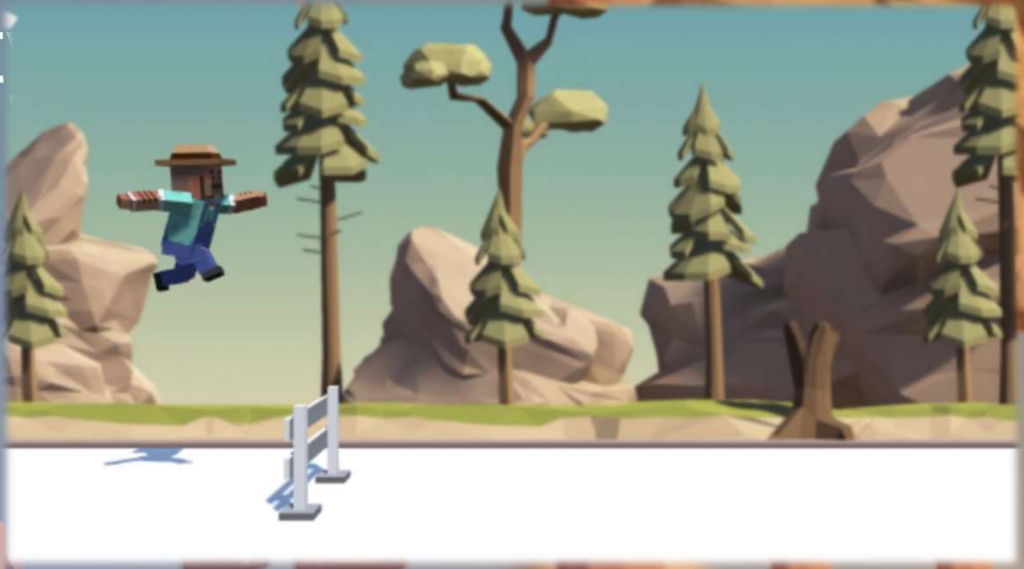
Update Progress (ต่อ)- มาโชว์ผลงานกัน



Unit 3 – Sound and Effects (Run and Jump Prototype)

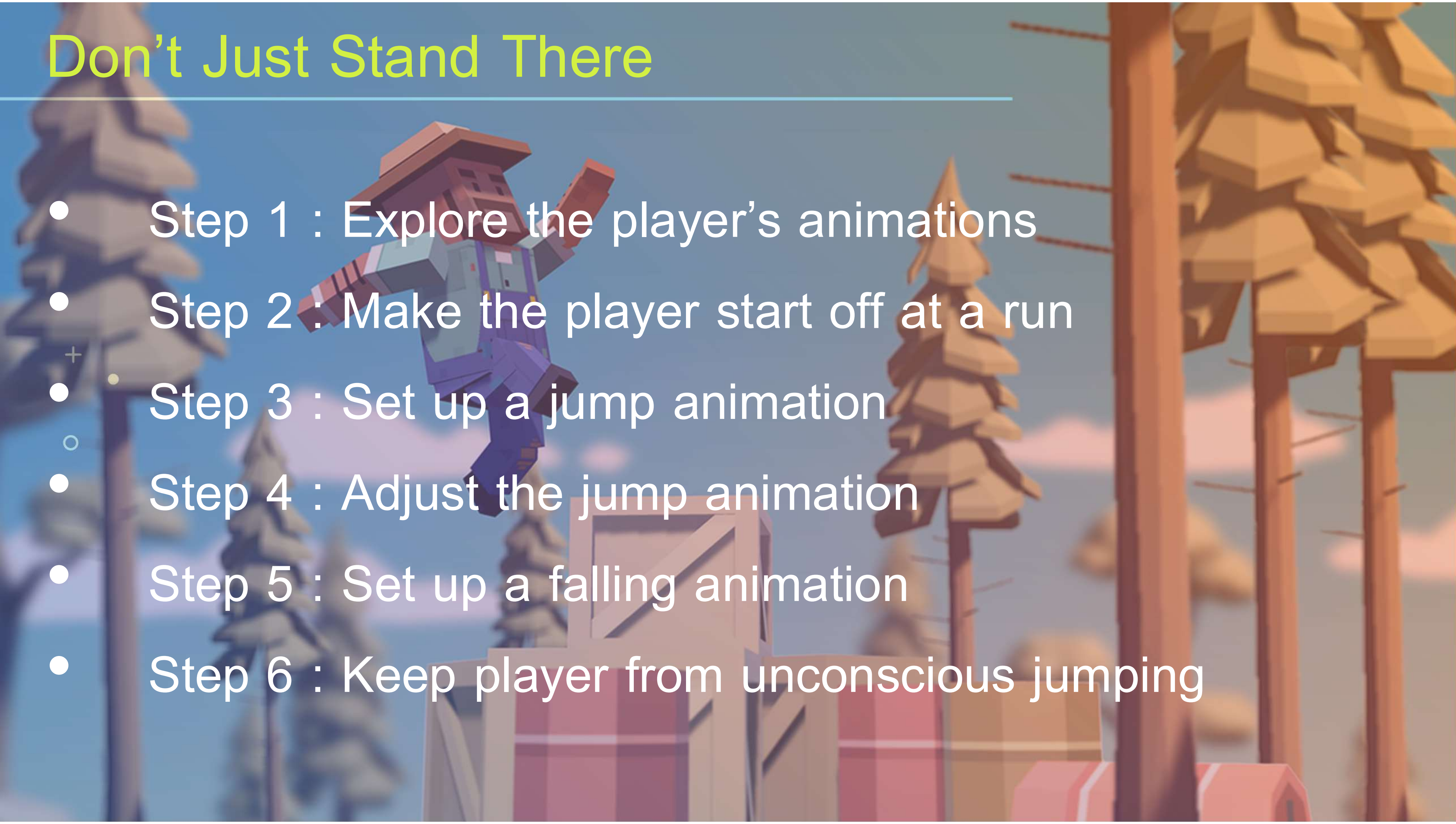
Unit 3 – Animation, Sound and Effects

- Run and Jump Protot
  - Jump Force
  - Make the World Whiz By
  - Don’t Just Stand There
  - Particles and Sound Effects



Don’t Just Stand There – อย่าแค่ยืนเฉยๆ สิ!

Don’t Just Stand There



- Step 1 : Explore the player’s animations
- Step 2 : Make the player start off at a run
- Step 3 : Set up a jump animation
- Step 4 : Adjust the jump animation
- Step 5 : Set up a falling animation
- Step 6 : Keep player from unconscious jumping

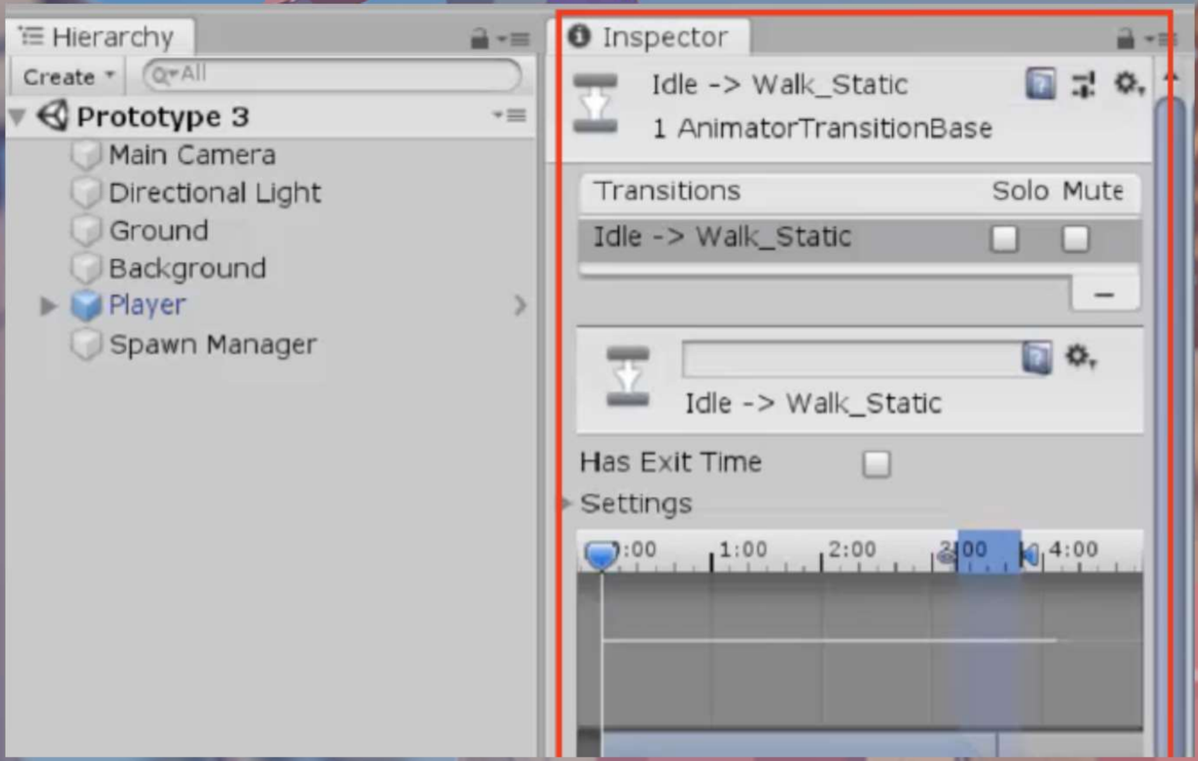
player’s animations

In order to get this character moving their arms and legs, we need to explore the Animation Controller.

1. Double-click on the Player’s Animation Controller, then explore the different Layers, double-clicking on States to see their animations and Transitions to see their conditions

New Concept : Animator Controller

New Concept : States and Conditions

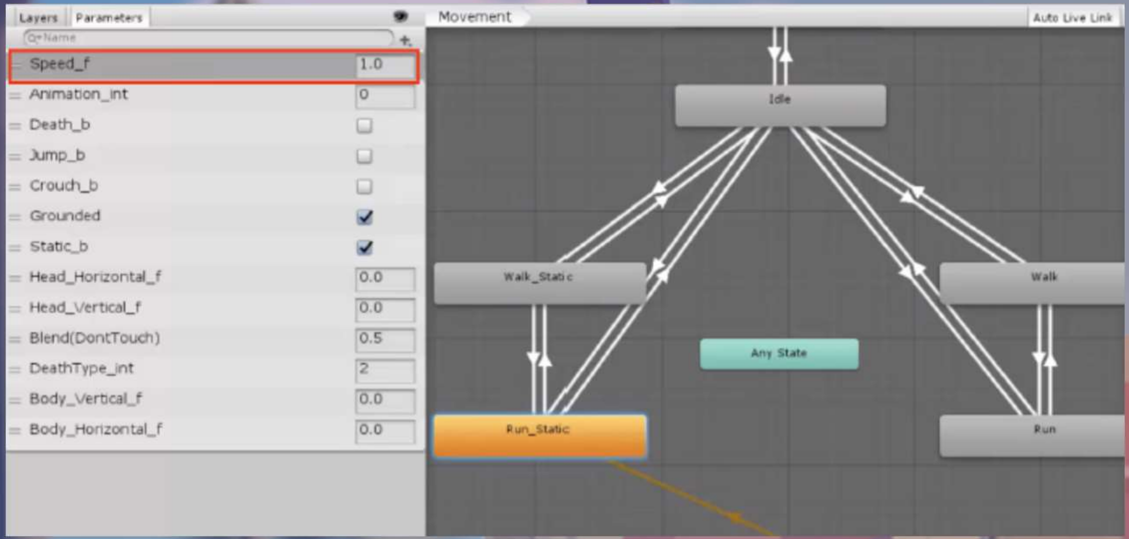

Hierarchy
Create
Prototype 3
Main Camera
Directional Light
Ground
Background
Player
Spawn Manager

Inspector
Idle -> Walk_Static
1 AnimatorTransitionBase
Transitions Solo Mute
Idle -> Walk_Static
Idle -> Walk_Static
Has Exit Time
Settings

start off at a run

Now that we're more comfortable with the animation controller, we can tweak some variables and settings to make the player look like they're really running.

1. In the **Parameters tab**, change the **Speed_f** variable to 1.0
2. Right-click on Run_Static > Set as Layer Default State
3. Single-click the the Run_Static state and adjust the Speed value in the inspector to match the speed of the background



Speed_f	1.0
Animation_int	0
Death_b	☐
Jump_b	☐
Crouch_b	☐
Grounded	✓
Static_b	✓
Head_Horizontal_f	0.0
Head_Vertical_f	0.0
Blend(DontTouch)	0.5
DeathType_int	2
Body_Vertical_f	0.0
Body_Horizontal_f	0.0

Tip: Notice how it transitions from idle to walk to Run - looks awkward - that's why need to make run default

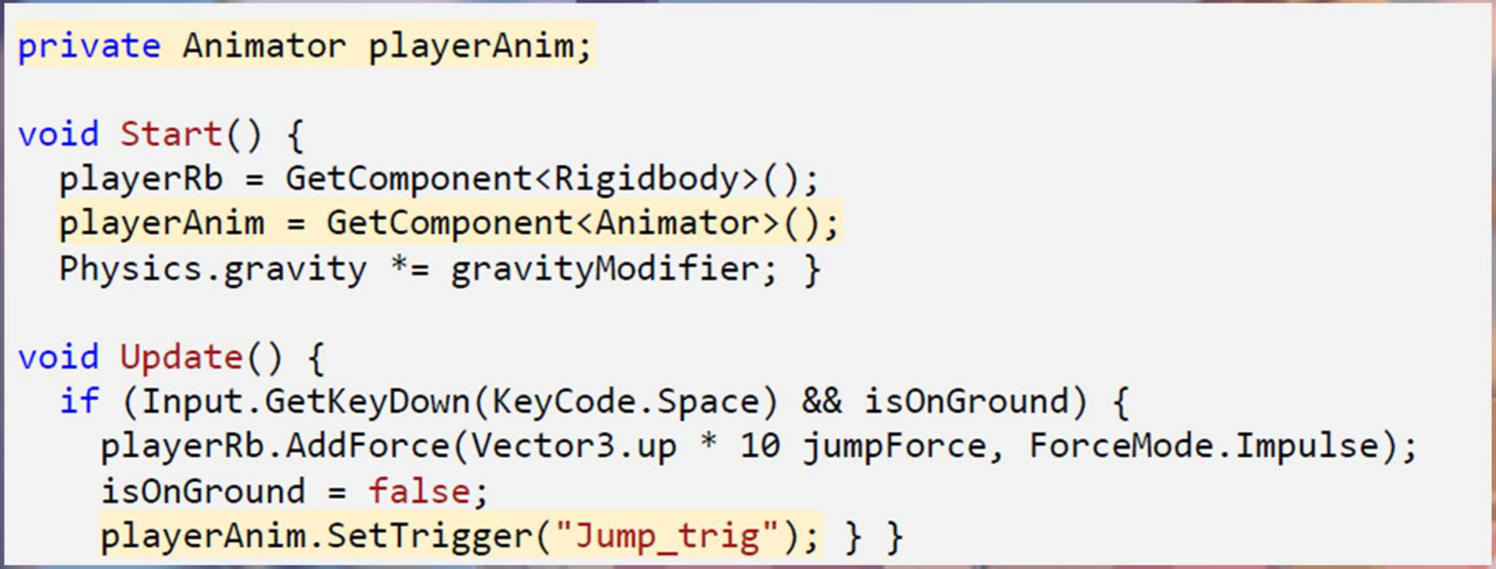
animation

The running animation looks good, but very odd when the player leaps over obstacles. Next up, we need to add a jumping animation that puts a real spring in their step.

1. In PlayerController.cs, declare a new private Animator playerAnim;
2. In Start(), set playerAnim = GetComponent<Animator>();
3. In the if-statement for when the player jumps, trigger the jump:
   - animator.SetTrigger("Jump_trig");

New Function: anim.SetTrigger

Tip: SetTrigger is helpful when you just want something to happen once then return to previous state (like a jump animation)

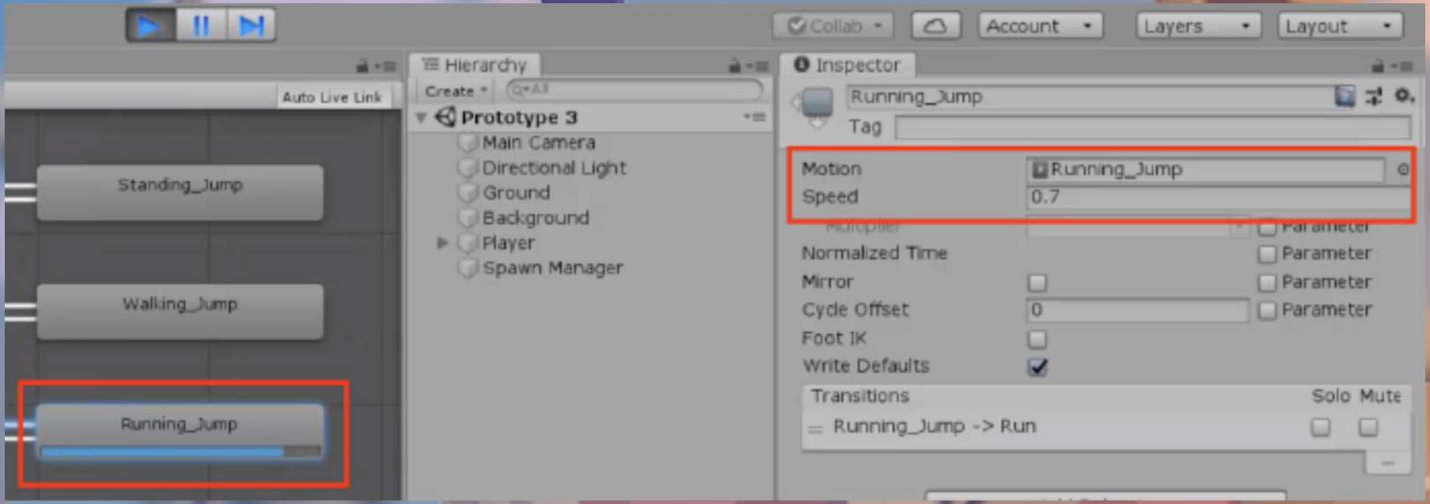


```
private Animator playerAnim;

void Start() {
  playerRb = GetComponent<Rigidbody>();
  playerAnim = GetComponent<Animator>();
  Physics.gravity *= gravityModifier; }

void Update() {
  if (Input.GetKeyDown(KeyCode.Space) && isOnGround) {
    playerRb.AddForce(Vector3.up * 10 jumpForce, ForceMode.Impulse);
    isOnGround = false;
    playerAnim.SetTrigger("Jump_trig"); } }
```

jump animation

The running animation plays, but it’s not perfect yet, we should tweak some of our character’s physics-related variables to get this looking just right.

1. In the Animator window, click on the Running_Jump state, then in the inspector and reduce its Speed value to slow down the animation
2. Adjust the player’s mass, jump force, and gravity modifier to get your jump just right



Running_Jump
Tag
Motion: Running_Jump
Speed: 0.7
Multiplier: Parameter
Normalized Time: Parameter
Mirror: Parameter
Cycle Offset: 0 Parameter
Foot IK
Write Defaults ✓

Transitions	Solo	Mute
Running_Jump -> Run	☐	☐

animation

The running and jumping animations look great, but there’s one more state that the character should have an animation for. Instead of continuing to sprint when it collides with an object, the character should fall over as if it has been knocked out.

1. In the condition that player collides with Obstacle, set the Death bool to true
2. In the same if-statement, set the DeathType integer to 1

New Function:

anim.SetBool

New Function:

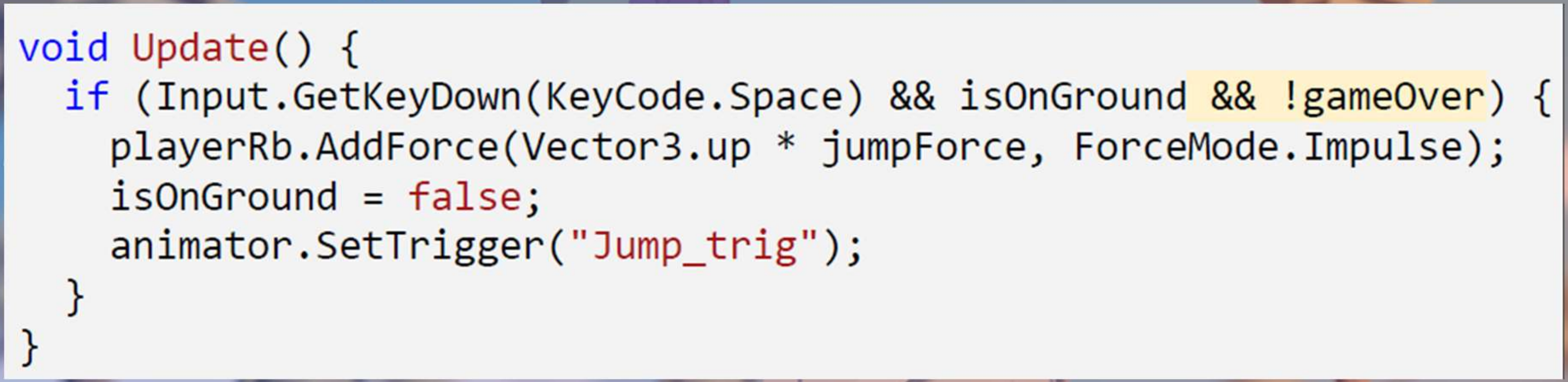
anim.SetInt

```
public bool gameOver = false;

private void OnCollisionEnter(Collision collision) {
  if (collision.gameObject.CompareTag("Ground")) {
        isOnGround = true;
  } else if (collision.gameObject.CompareTag("Obstacle")) {
        Debug.Log("Game Over")
        gameOver = true;
        playerAnim.SetBool("Death_b", true);
        playerAnim.SetInteger("DeathType_int", 1);
  }
}
```

unconscious jumping

Everything is working perfectly, but there’s one small disturbing bug to fix: the player can jump from an unconscious position, making it look like the character is being defibrillated.

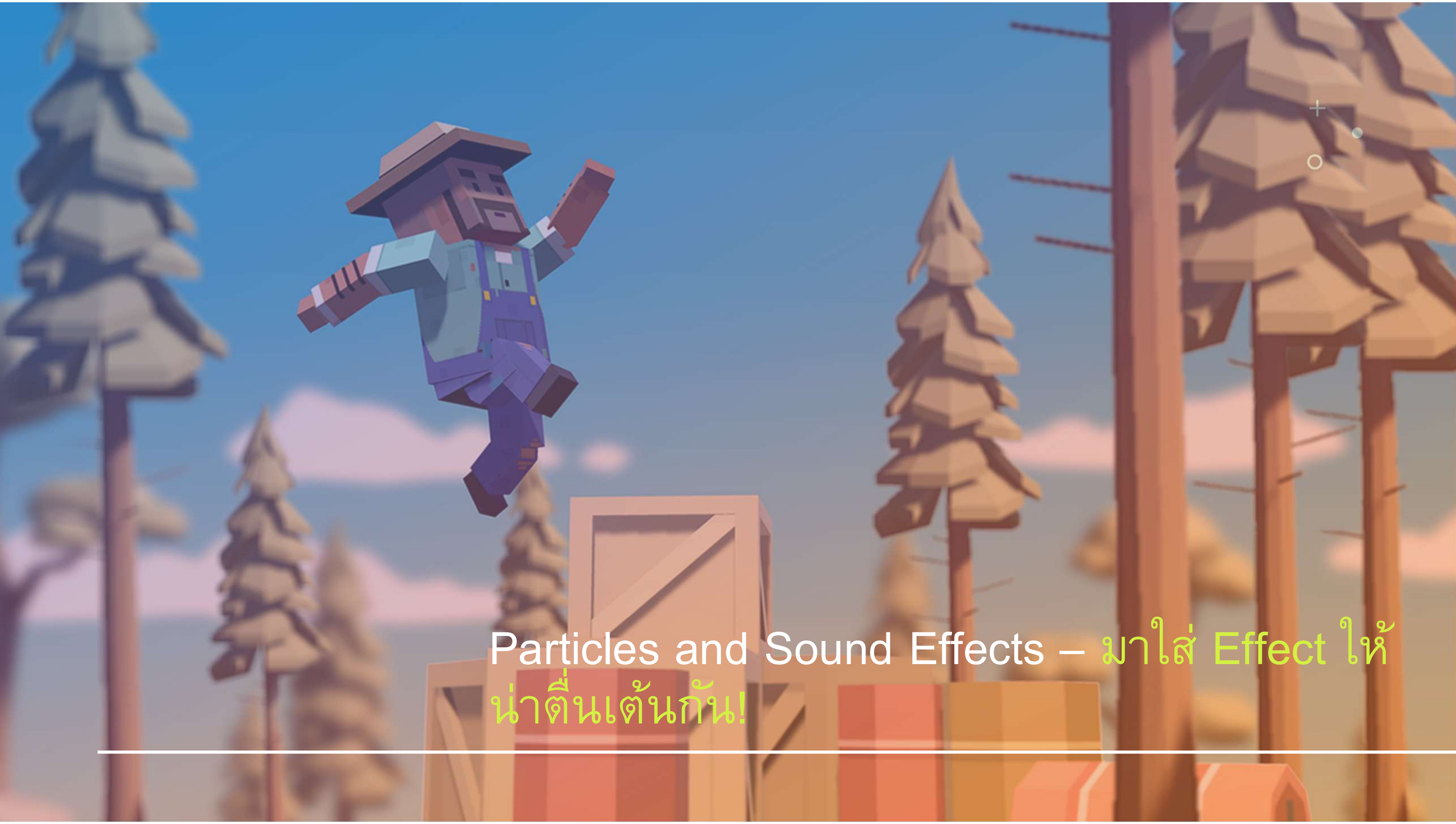
1. To prevent the player from jumping while unconscious, add && !gameOver to the jump condition



```
void Update() {
  if (Input.GetKeyDown(KeyCode.Space) && isOnGround && !gameOver) {
    playerRb.AddForce(Vector3.up * jumpForce, ForceMode.Impulse);
    isOnGround = false;
    animator.SetTrigger("Jump_trig");
  }
}
```

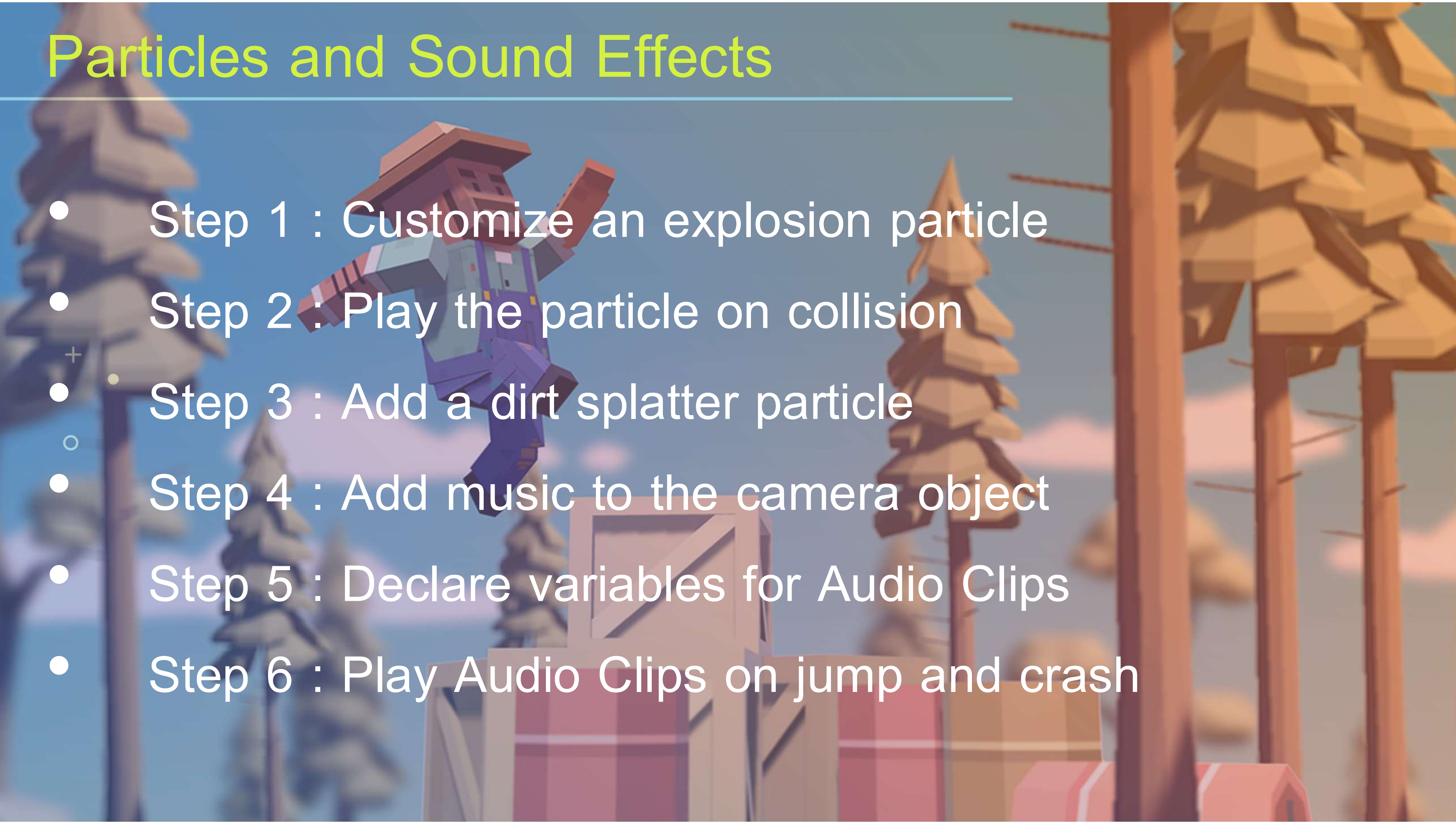
New Concept: ! “Does not” and !=“Does not equal” operators

Tip: gameOver != true is the same as gameOver == false



Particles and Sound Effects – มาใส่ Effect ให้น่าตื่นเต้นกัน!

Particles and Sound Effects



- 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

particle

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

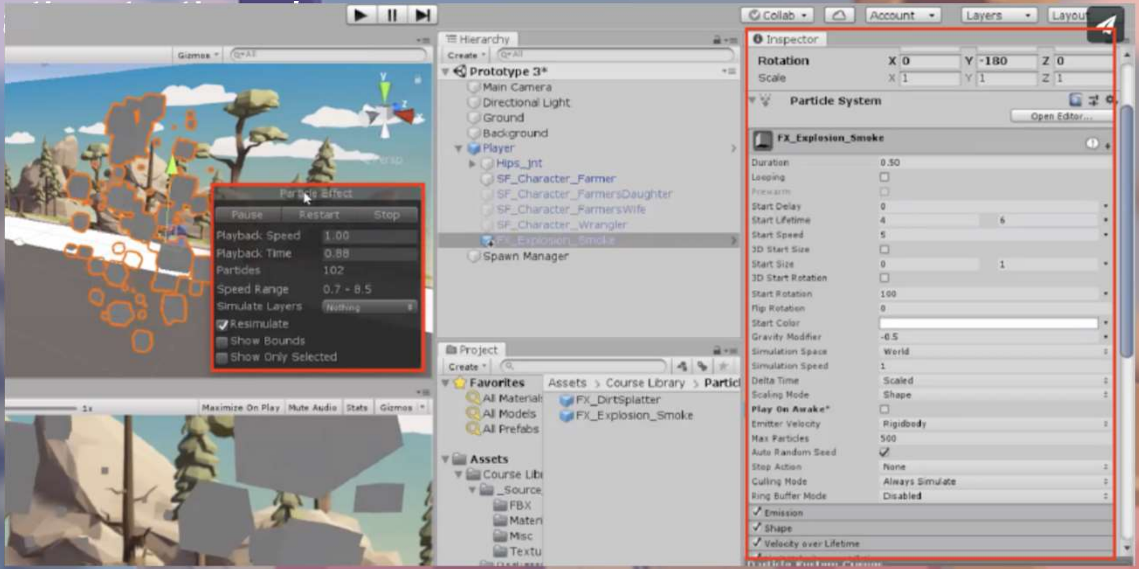
1. From the Course Library > Particles, drag FX_Explosion_Smoke into the hierarchy, then use the Play / Restart / Stop buttons to preview it
2. Play around with the settings to get your particle system the way you want it
3. Make sure to uncheck the Play on Awake setting
4. Drag the particle onto your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

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



Particle Effect
Pause | Restart | Stop
Playback Speed 1.00
Playback Time 0.88
Particles 102
Speed Range 0.7 - 8.5
Simulate Layers Nothing
✓ Resimulate
Show Bounds
Show Only Selected

Inspector
Rotation X 0 Y -180 Z 0
Scale X 1 Y 1 Z 1
Particle System
Open Editor...
FX_Explosion_Smoke

Duration	0.50
Looping	☐
Prewarm	☐
Start Delay	0
Start Lifetime	4 6
Start Speed	5
3D Start Size	☐
Start Size	0 1
3D Start Rotation	☐
Start Rotation	100
Flip Rotation	0
Start Color	
Gravity Modifier	-0.5
Simulation Space	World
Simulation Speed	1
Delta Time	Scaled
Scaling Mode	Shape
Play On Awake*	☐
Emitter Velocity	Rigidbody
Max Particles	500
Auto Random Seed	✓
Stop Action	None
Culling Mode	Always Simulate
Ring Buffer Mode	Disabled

✓ Emission
✓ Shape
✓ Velocity over Lifetime

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 Make sure to uncheck the Play on Awake setting
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(); } }
```

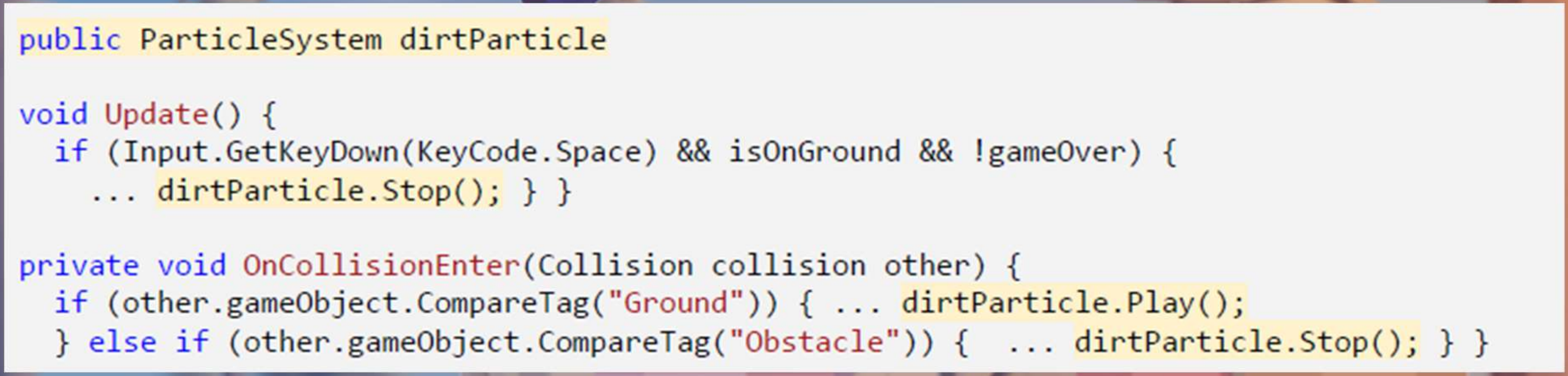
Particles and Sound Effects – Step 3 : Add a dirt splatter particle

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(); } }
```

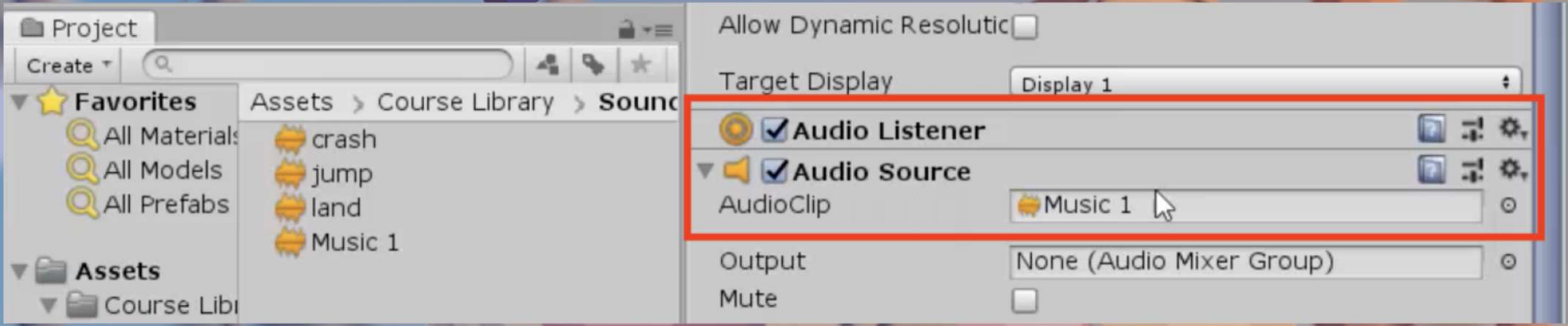
object

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



Allow Dynamic Resolutic ☐
Target Display: Display 1
✓ Audio Listener
✓ Audio Source
AudioClip: Music 1
Output: None (Audio Mixer Group)
Mute ☐

Clips

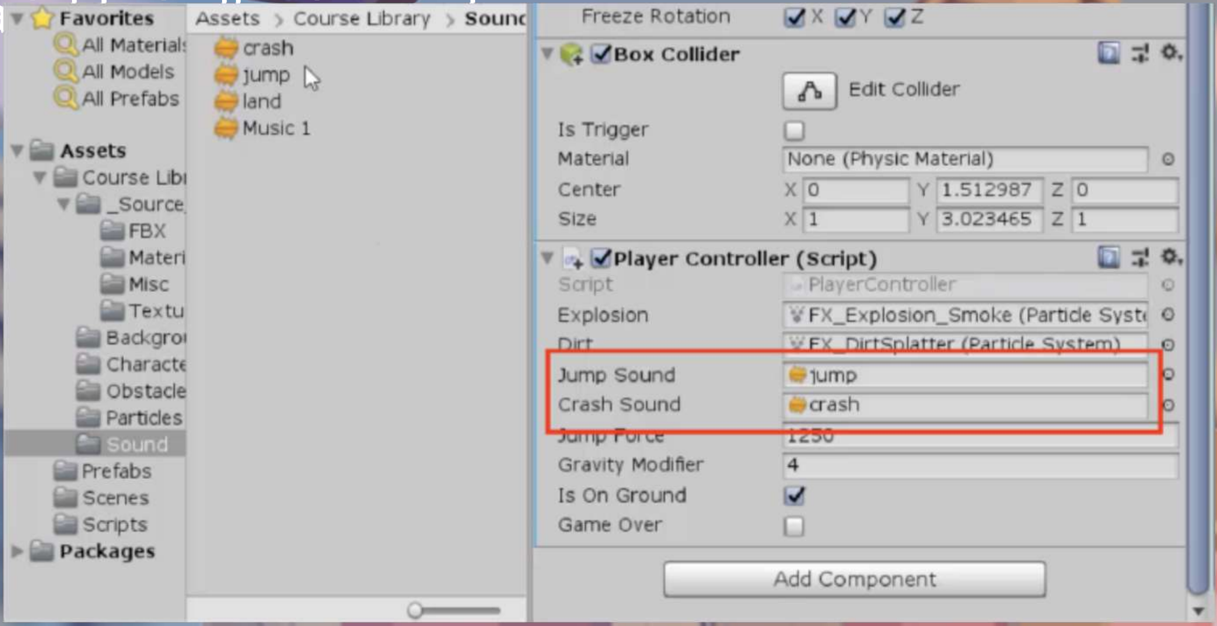
Now that we've got some nice music playing, it's time to add some sound effects. This time audio clips will emanate from the player, rather than the camera itself.

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 va

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



Freeze Rotation ✓X ✓Y ✓Z

✓Box Collider

Edit Collider

Is Trigger	☐
Material	None (Physic Material)
Center	X 0 Y 1.512987 Z 0
Size	X 1 Y 3.023465 Z 1

✓Player Controller (Script)

Script	PlayerController
Explosion	FX_Explosion_Smoke (Particle Syst
Dirt	FX_DirtSplatter (Particle System)
Jump Sound	jump
Crash Sound	crash
Jump Force	1250
Gravity Modifier	4
Is On Ground	✓
Game Over	☐

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

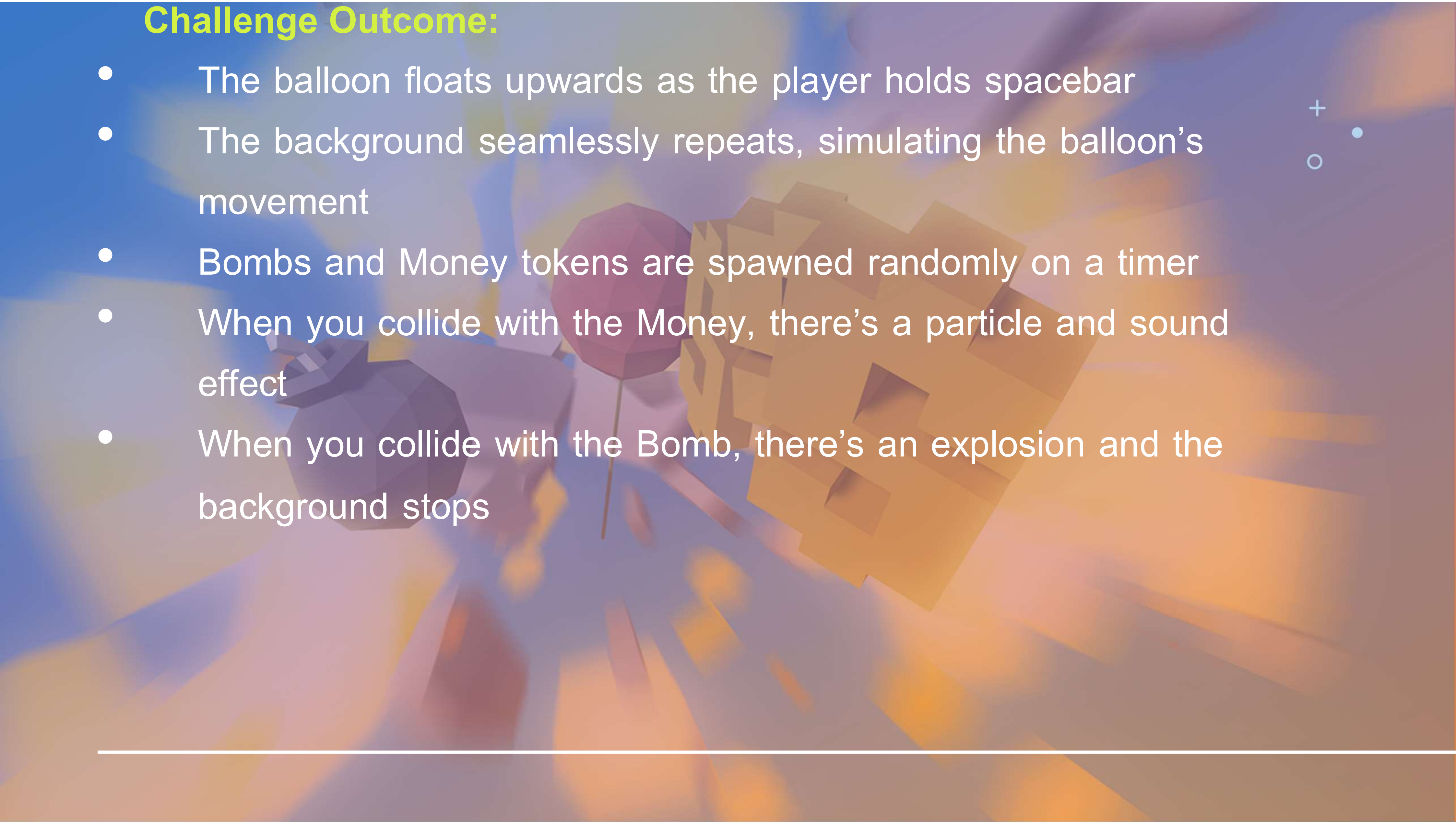
Don't worry: Declaring a new AudioSource variable is just like declaring a new Animator or RigidBody

```
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); } }
```

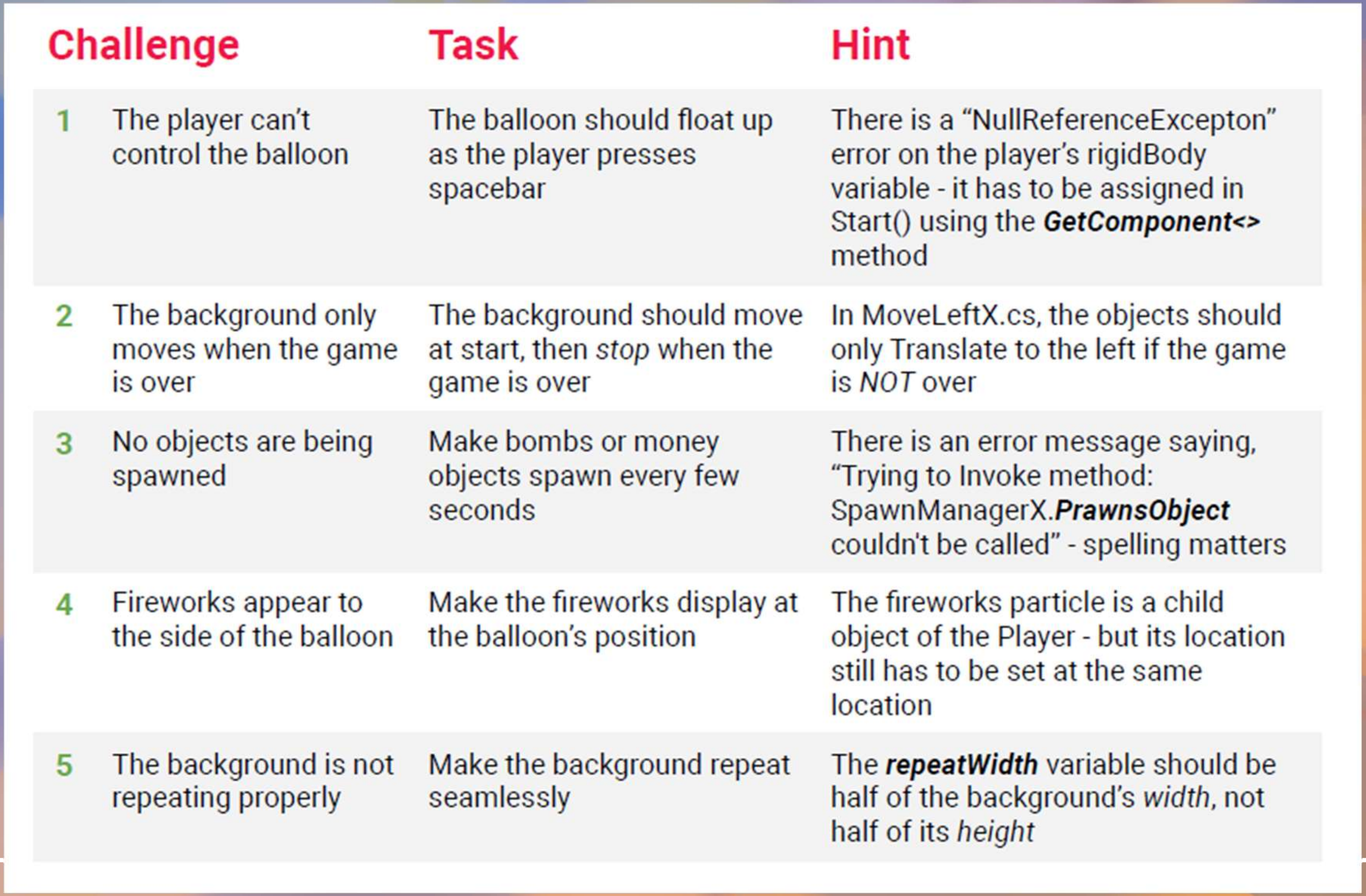
Challenge 3 - Balloons, Bombs, & Booleans

Challenge Outcome:

- The balloon floats upwards as the player holds spacebar
- The background seamlessly repeats, simulating the balloon’s movement
- Bombs and Money tokens are spawned randomly on a timer
- When you collide with the Money, there’s a particle and sound effect
- When you collide with the Bomb, there’s an explosion and the background stops

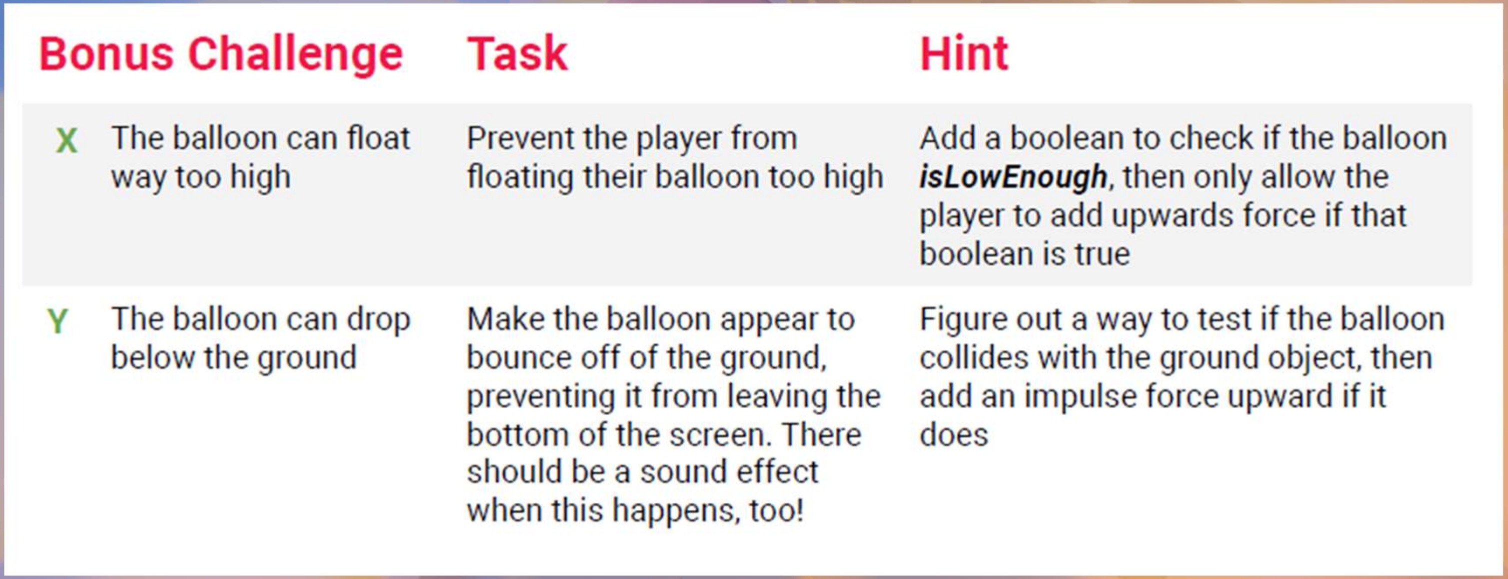


Booleans



	Challenge	Task	Hint
1	The player can't control the balloon	The balloon should float up as the player presses spacebar	There is a "NullReferenceExcepton" error on the player's rigidBody variable - it has to be assigned in Start() using the ***GetComponent<>*** method
2	The background only moves when the game is over	The background should move at start, then *stop* when the game is over	In MoveLeftX.cs, the objects should only Translate to the left if the game is *NOT* over
3	No objects are being spawned	Make bombs or money objects spawn every few seconds	There is an error message saying, "Trying to Invoke method: SpawnManagerX.***PrawnsObject*** couldn't be called" - spelling matters
4	Fireworks appear to the side of the balloon	Make the fireworks display at the balloon's position	The fireworks particle is a child object of the Player - but its location still has to be set at the same location
5	The background is not repeating properly	Make the background repeat seamlessly	The ***repeatWidth*** variable should be half of the background's *width*, not half of its *height*

Booleans



	Bonus Challenge	Task	Hint
X	The balloon can float way too high	Prevent the player from floating their balloon too high	Add a boolean to check if the balloon ***isLowEnough***, then only allow the player to add upwards force if that boolean is true
Y	The balloon can drop below the ground	Make the balloon appear to bounce off of the ground, preventing it from leaving the bottom of the screen. There should be a sound effect when this happens, too!	Figure out a way to test if the balloon collides with the ground object, then add an impulse force upward if it does

Mid Term

Unit 1 - Player Control

Unit 2 - Basic Gameplay

Unit 3 - Sound and Effects

Mid Term

- ## Gameplay Mechanics
  - Watch Where You’re Going
  - Follow the Player
  - PowerUp and CountDown
  - For-Loops For Waves

After Midterm : To Be Continue…