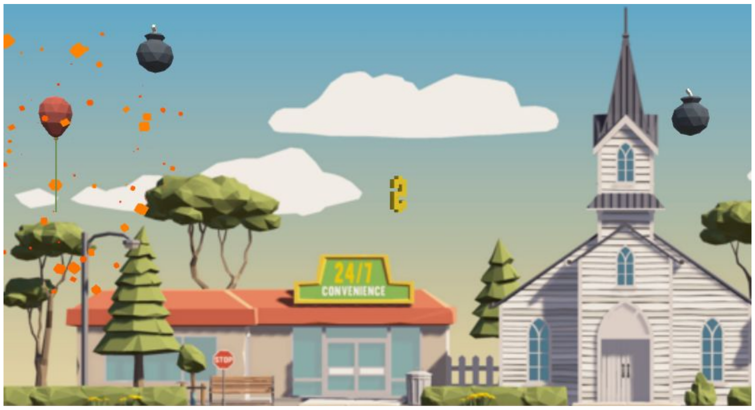
Challenge 3

Balloons & Booleans



Challenge Overview: Apply your knowledge of physics, scrolling backgrounds, and special effects to a balloon floating through town, picking up tokens while avoiding explosives. You will have to do a lot of troubleshooting in this project because it is riddled with errors.

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

Challenge Objectives: In this challenge, you will reinforce the following skills/concepts:

- Declaring and initializing variables with the GetComponent method
- Using booleans to trigger game states
- Displaying particle effects at a particular location relative to a gameobject
- Seamlessly scrolling a repeating background

Challenge Instructions:

- Open your **Prototype 3** project
- **Download** the "Challenge 3 Starter Files" from the Tutorial Materials section, then double-click on it to **Import**
- In the *Project Window > Assets > Challenge 3 >* ***Instructions*** folder, use the "Challenge 3 - Instructions" and Outcome video as a guide to complete the challenge

	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*

	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

Challenge Solution

1 In PlayerControllerX.cs, in Start(), assign ***playerRb*** just like the playerAudio variable:

```
playerAudio = GetComponent<AudioSource>();
playerRb = GetComponent<Rigidbody>();
```

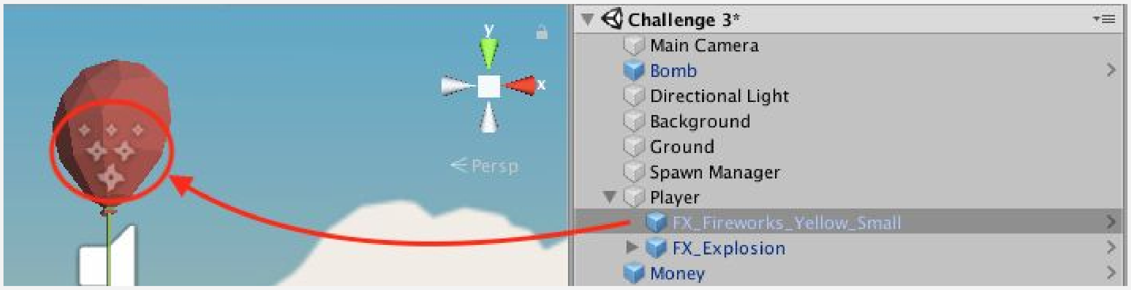
2 In MoveLeftX.cs, the objects should only Translate to the left if the game is NOT over - it's currently checking if the game IS over:

```
if (! playerControllerScript.gameOver) {
  transform.Translate(Vector3.left * speed * Time.deltaTime, Space.World);
}
```

3 In SpawnManagerX.cs, in Start(), the InvokeRepeating method is using an incorrect spelling of "SpawnObjects" - correct the spelling error

```
void Start() {
  InvokeRepeating("~~PrawnsObject~~SpawnObjects", spawnDelay, spawnInterval);
  ...
}
```

4 Select the Fireworks child object and reposition it to the same location as the Player



5 In RepeatBackgroundX.cs, in Start(), the repeatWidth should be dividing the X size (width) of the box collider by 2, not the Y size (height)

```
repeatWidth = GetComponent<BoxCollider>().size.~~y~~ x / 2;
```

Bonus Challenge Solution

X1 In PlayerControllerX.cs create a boolean to track whether the player is low enough to float upwards, then in Update(), set it to ***false*** if the player is above a certain Y value and, else, set it to ***true***

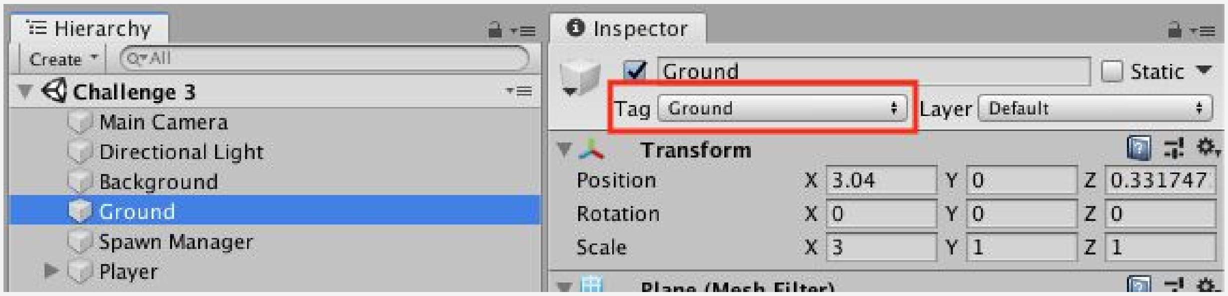
```
public bool isLowEnough;

void Update() {
  if (transform.position.y > 13) {
    isLowEnough = false;
  } else {
    isLowEnough = true;
  }
}
```

X2 In the if-statement testing for the player pressing spacebar, add a condition testing that the ***isLowEnough*** boolean is true:

```
if (Input.GetKey(KeyCode.Space) && isLowEnough && !gameOver) {
  playerRb.AddForce(Vector3.up * floatForce
}
```

Y1 Add a tag to the Ground object so that you can easily test for a collision with it



☑ Ground
Tag Ground
Layer Default

	X	Y	Z
Position	3.04	0	0.331747
Rotation	0	0	0
Scale	3	1	1

Y2 In PlayerControllerX.cs, in the OnCollisionEnter method, add a third else-if checking if the balloon collided with the ground during the game, and if so, to add an impulse force upwards

```
private void OnCollisionEnter(Collision other) {
...
} else if (other.gameObject.CompareTag("Ground") && !gameOver)
{
  playerRb.AddForce(Vector3.up * 10, ForceMode.Impulse);
}
```

Y3 To add a sound effect, declare a new AudioClip variable and assign it in the inspector, then use the PlayOneShot method when the player collides with the ground.

```
public AudioClip moneySound;
public AudioClip explodeSound;
public AudioClip bounceSound;

private void OnCollisionEnter(Collision other) {
...
} else if (other.gameObject.CompareTag("Ground") && !gameOver)
{
  rigidBody.AddForce(Vector3.up * 10, ForceMode.Impulse);
  playerAudio.PlayOneShot(bounceSound, 1.5f);
}
```