

Bonus Features 4 Solution Walkthrough



Easy - Harder Enemy	2
Medium - Homing Rockets	6
Hard - Smashingly Good	21
Expert - Boss Battle	29

Easy - Harder Enemy

1. Navigate to the Scripts folder in the Project window and open up **SpawnManager.cs**. On line 7, change the *enemyPrefab* variable to be an array and rename it to *enemyPrefabs*.

```
public GameObject[] enemyPrefabs;
```

2. In the SpawnEnemyWave method, we need to determine which prefab from the array we will spawn. We can do this using Random.Range. The updated method looks like this:

```
void SpawnEnemyWave(int enemiesToSpawn)
{
    for (int i = 0; i < enemiesToSpawn; i++)
    {
        int randomEnemy = Random.Range(0, enemyPrefab.Length);
        Instantiate(enemyPrefabs[randomEnemy], GenerateSpawnPosition(),
enemyPrefabs[randomEnemy].transform.rotation);
    }
}</pre>
```

3. Save the script and head back to Unity. Navigate to the Prefabs folder in the Project window. Left click on the **Enemy** prefab and then press **Ctrl + D** (PC) or **Cmd + D** (Mac) to duplicate the prefab. Rename it to "EnemyFast".



4. Select the *EnemyFast* prefab and in the Inspector adjust the **Speed** parameter of the **Enemy (Script) component**.

٦	🗖 🖬 🗹 Enemy (Script)		🔯 🕸 🏟
	Script	Enemy	0
	Speed	6	

 In the Project view, Navigate to Assets > Course Library > Materials. Select the Material called "PolygonPrototype_Texture_Grid_03" and press Ctrl + D (PC) or Cmd + D (Mac) to duplicate it. Rename it to "FastEnemyMaterial".



6. Select the new Material, and in the Inspector change the **Albedo Color** value to blue.

Color	Inspector		∂ •≡
	FastEnemyMaterial		🛐 🕂 🌣,
#	Shader Standard		•
	Rendering Mode	Opaque	\$
	Main Maps		
	─ ⊙ Albedo	_ J	
	⊙ Metallic	0	0
	Smoothness	o	0.5
	Source	Metallic Alpha	+
	⊙ Normal Map		
	⊙ Height Map		
	⊙ Occlusion		
	⊙ Detail Mask		
	Emission		
	Tiling	X 1 Y 1	
RGB 0-25:+	onset		
R 0	Secondary Maps		
G O	⊙Detail Albedo x2		
B 255	⊙ Normal Map		1
200	Tiling	X 1 Y 1	
A [255	Offset	X 0 Y 0	
Hexadecimal 0000FF	UV Set		;
▼ Swatches +=	Forward Rendering Opt	ions	
	Specular Highlights		

7. Go back to the *EnemyFast* prefab and assign the new Material to the **Mesh Renderer Material** parameter.

Select Material x	 Inspector ☑ EnemyFast 		🔒 +≡
Q)	Tag Enemy	‡ Layer Default	\$
Assets Self	▼ <mark>人 Transform</mark> Position Rotation	X 0 Y 0.15 X 0 Y 0	[2] ; ≉, Z 6 Z 0
PolygonMinis Fantasy 01	Scale	X 1.5 Y 1.5	Z 1.5
PolygonPrototype_Texture_01	🕨 🧾 🛛 Sphere (Mesh Filte	r)	🔯 🗟 🔅
PolygonPrototype_Texture_Grid_03 PolygonPrototype_Texture_Grid_07 PolygonPrototype_Texture_Grid_Gla	▼ 🛃 🗹 Mesh Renderer ▼ Materials Size	1	□ ↓ ☆.
SimpleIcons	Element 0	FastEnemyMaterial	C
FrameDebuggerRenderTargetDisplay	Light Probes	Blend Probes	÷
Default-Diffuse	Reflection Probes	Blend Probes	+
Default-Line	Anchor Override	None (Transform)	0
Default-Material	Cast Shadows	On	+
Default-Particle	Receive Shadows		
Default-ParticleSystem Default-Skybox	Motion Vectors	Per Object Motion	*
FastEnemyMaterial (Material) Assets/	Lightmap Static		

8. Select the *SpawnManager* in the Hierarchy. On the **Spawn Manager (Script)** component, change the size of the **Enemy Prefab** to **2** and assign the two enemy prefabs to the empty slots.

Inspector		≧ ≠≡			
🕤 🗹 Spawn Manager		🗌 🗌 Static 🔻			
Tag Untagged	+ Layer Default				
▼ 🙏 Transform		🔯 🕸 🔅			
Position	X -0.1609669 Y 0.7849145	Z -0.41			
Rotation	X 0 Y 0	Z 0			
Scale	X 1 Y 1	Ζ 1			
🔻 📾 🗹 Spawn Manager (Sc	ript)	🔯 다 🌣,			
Script	🗟 SpawnManager	0			
▼ Enemy Prefabs					
Size	2				
Element 0	🜍 Enemy	c			
Element 1	🜍 Enemy Fast	c			
Powerup Prefab	Powerup	0			
Enemy Count	0				
Wave Number	1				

9. Save the scene and test out the game. There is now an enemy that rolls towards you faster.



Medium - Homing Rockets

1. Let's first create a script for our Powerups. In the Project window, navigate to the scripts folder. Right-click and select **Create** > **C# Script**. Name this script *"PowerUp"*.

			Folder	
			C# Script	
			Shader	>
		\leq	Testing	>
	Contra		Playables	>
	Create Chewin Fundamer	,	Assembly Definition	
	Onen		TextMeshPro	>
	Delete		Scene	
	Rename		Prefab Variant	
	Copy Path	Alt+Ctrl+C	Audio Mixer	
		Altocario	Audio Mixer	
	Open Scene Additive		Material	
	Import New Asset		Lens Flare	
	Import Package	>	Render Texture	
	Export Package		Lightmap Parameters	
	Find References In Scene		Custom Render Texture	
	Select Dependencies		Sprite Atlas	
	Refresh	Ctrl+R	Sprites	>
Project 🛛 🗉 Console	Reimport		Tile	
Create *	Reimport All		Animator Controller	
Favorites	Fritzent Franze Darfalt		Animation	
🔻 🚞 Assets	Extract From Prefab		Animator Override Controller	
Course Library	Run API Updater		Avatar Mask	
Ground	Update UIElements Schema		Timeline	
Pickups	Open C# Project		Physic Material	
Powerup Indicators	Enemy PlayerController	RotateCame	Physics Material 2D	
Textures			GUI Skin	
Physics Materials			Custom Font	
Scenes			Legacy	>
Scripts 🗸 🚔 🗛	ssets/Scripts		Prush	

2. Double-click the script to open it. We're using an enum for the power up type to allow for more power ups to be added easily. Replace the contents of the script with this:

using UnityEngine;

public enum PowerUpType { None, Pushback, Rockets }

```
public class PowerUp : MonoBehaviour
{
    public PowerUpType powerUpType;
}
```

Save the script and head back to Unity.

3. In the Scripts folder, create a new script called *RocketBehaviour*. This script will hold the logic for the rocket.



4. The first thing we need to do in this script is set up the variables. Below the class definition, add the following:

```
private Transform target;
private float speed = 15.0f;
private bool homing;
private float rocketStrength = 15.0f;
private float aliveTimer = 5.0f;
```

5. Delete the **Start** method as we won't be using it. Below the **Update** method, create a new method called *Fire*. This will be called by our player when we spawn in the rockets, so needs to be public.

```
public void Fire(Transform newTarget)
{
    target = homingTarget;
    homing = true;
    Destroy(gameObject, aliveTimer);
}
```

This method takes in a Transform that we will set as the target. It will set the homing boolean to true and then set the GameObject to be destroyed after 5 seconds (as defined by aliveTimer).

6. Next we will set up the code for moving and rotating the missile towards the target. Update the **Update** method to look like the following:

```
void Update()
{
    if(homing && target != null)
    {
        Vector3 moveDirection = (target.transform.position -
transform.position).normalized;
        transform.position += moveDirection * speed * Time.deltaTime;
        transform.LookAt(target);
    }
}
```

7. Now we will add in the **OnCollisionEnter** method. This will add a force to whatever is hit.

```
void OnCollisionEnter(Collision col)
{
    if (target != null)
    {
        if (col.gameObject.CompareTag(target.tag))
        {
            Rigidbody targetRigidbody = col.gameObject.GetComponent<Rigidbody>();
            Vector3 away = -col.contacts[0].normal;
            targetRigidbody.AddForce(away * rocketStrength, ForceMode.Impulse);
            Destroy(gameObject);
        }
    }
}
```

This method first checks if we have a target. If we do, we compare the tag of the colliding object with the tag of the target. If they match, we get the rigidbody of the target. We then use the normal of the collision contact to determine which direction to push the target in. Finally we apply the force to the target and destroy the missile.

Save the script and head back to Unity.

8. Open up the **PlayerController.cs** script. We will need some new variables. Below the current variable declarations, add the following:

```
public PowerUpType currentPowerUp = PowerUpType.None;
public GameObject rocketPrefab;
private GameObject tmpRocket;
private Coroutine powerupCountdown;
```

You will notice that we are using the **PowerUpType** enum here. This is used to help determine which logic to enable for the player when a power up is collected. We are also declaring two GameObject variables. The public one is used for the homing rocket prefab. The private one will be used for spawning in the homing rockets.

9. To add in our new logic, we need to update the **OnTriggerEnter** method. The new lines are highlighted

below.

```
private void OnTriggerEnter(Collider other)
{
    if (other.CompareTag("Powerup"))
    {
        hasPowerup = true;
        currentPowerUp = other.gameObject.GetComponent<PowerUp>().powerUpType;
        powerupIndicator.gameObject.SetActive(true);
        Destroy(other.gameObject);
        if(powerupCountdown != null)
        {
            StopCoroutine(powerupCountdown);
        }
        powerupCountdown = StartCoroutine(PowerupCountdownRoutine());
     }
}
```

10. We will need to have a way to return our currentPowerUp to none, we can do this within the **PowerupCountdownRoutine** Method.

```
IEnumerator PowerupCountdownRoutine()
{
    yield return new WaitForSeconds(7);
    hasPowerup = false;
    currentPowerUp = PowerUpType.None;
    powerupIndicator.gameObject.SetActive(false);
}
```

11. We are also going to need to change the logic in the **OnCollisionEnter** method. We need to replace the hasPowerup boolean check with a currentPowerUp check. We will also update the Debug.Log output to include the power up we are using. The updated OnCollisionEnter method will look like the following:

```
private void OnCollisionEnter(Collision collision)
{
    if (collision.gameObject.CompareTag("Enemy") && currentPowerUp ==
PowerUpType.Pushback)
    {
        Rigidbody enemyRigidbody = collision.gameObject.GetComponent<Rigidbody>();
        Vector3 awayFromPlayer = collision.gameObject.transform.position -
transform.position;
        enemyRigidbody.AddForce(awayFromPlayer * powerUpStrength,
ForceMode.Impulse);
        Debug.Log("Player collided with: " + collision.gameObject.name + " with
powerup set to " + currentPowerUp.ToString());
    }
}
```

12. After the **OnCollisionEnter** method, add the following:

```
void LaunchRockets()
{
    foreach(var enemy in FindObjectsOfType<Enemy>())
    {
        tmpRocket = Instantiate(rocketPrefab, transform.position + Vector3.up,
Quaternion.identity);
        tmpRocket.GetComponent<RocketBehaviour>().Fire(enemy.transform);
    }
}
```

Here we are using the same logic as our spawn manager to find all the enemies. We are then launching our missiles at each one. We launch the missiles from above the player, to stop the collision from pushing us back.

13. The last thing we need to do in the **PlayerController**, is add the logic to the **Update** method. We're going to check if the currentPowerUp is the rocket, and if the "**F**" key is pressed. If both are true, we will call the **LaunchRockets** method.

```
void Update()
{
    float forwardInput = Input.GetAxis("Vertical");
    playerRb.AddForce(focalPoint.transform.forward * forwardInput * speed);
    powerupIndicator.transform.position = transform.position + new Vector3(0,
-0.5f, 0);
    if (currentPowerUp == PowerUpType.Rockets && Input.GetKeyDown(KeyCode.F))
    {
      LaunchRockets();
    }
}
```

Save the script and return to Unity

14. Now that we have the logic set up for the missiles, we will need to adjust the SpawnManager to allow for two different types of power ups. Double click **SpawnManager.cs** to open it up. Find the variable named powerUpPrefab and change it to the following:

```
public GameObject[] powerupPrefabs;
```

15. Find the **Start** method and adjust it to look like the following:

```
void Start()
{
    int randomPowerup = Random.Range(0, powerupPrefabs.Length);
    Instantiate(powerupPrefabs[randomPowerup], GenerateSpawnPosition(),
    powerupPrefabs[randomPowerup].transform.rotation);
    SpawnEnemyWave(waveNumber);
}
```

What we are doing here, is getting a random number based on the length of the powerupPrefabs array, and then spawning the object at that position in the array.

We'll also need to add the same code to replace the existing powerup spawning in the Update method. The Update method should look like this:

```
void Update()
{
    enemyCount = FindObjectsOfType<Enemy>().Length;
    if (enemyCount == 0)
    {
        waveNumber++;
        SpawnEnemyWave(waveNumber);
        int randomPowerup = Random.Range(0, powerupPrefabs.Length);
        Instantiate(powerupPrefabs[randomPowerup], GenerateSpawnPosition(),
powerupPrefabs[randomPowerup].transform.rotation);
    }
}
```

Save the script and head back to Unity.

16. Navigate to the Prefabs folder and select the *Powerup* prefab. In the Inspector, click **Add Component** and search for the **PowerUp** script.

Inspector									1	
Powerup						_		Sta	itic	•
Tag Powerup		ŧ La	yer	Default						+
🔻 🙏 🛛 Transform	ı								킕	\$
Position	Х	0	Y	0		Z [4			
Rotation	Х	0	Y	0		z [0			
Scale	Х	2	Y	2		z [2			
▶ 🗒 SM_Icon_0	Gem_01 (Me	sh Filter)							킕	\$
🕨 🛃 🗹 Mesh Rend	lerer	lerer							킕	\$
⊳ 🧊 🗹 Вож Collid	er								킕	\$
PolygonP	rototype_Te	xture_01							킕	\$
🕨 🐨 Shader 💽	tandard									•
						_				
	Ad	ld Compone	ent							
	(Q power			0						
Search										
	📾 Power Up									
	New script	:		>						

After adding the **PowerUp** script, change the **Power Up Type field** to **Pushback**.

Inspector								Ê,	•≡
Powerup							Sta	tic	•
Tag Powerup	\$	Layer	Defau	lt					ŧ
▼🙏 Transform								킕	\$,
Position	X 0	Y	0		Ζ	4			
Rotation	X 0	Y	0		Ζ	0			
Scale	X 2	Y	2		Ζ	2			
▶ 🗒 SM_Icon_Gem_01 (Mesh Filter) 🛛 🔯 🛱									
🕨 🛃 🗹 Mesh Renderer								킕	\$,
🕨 🤪 🗹 Box Collider								킕	\$,
🔻 📾 🛛 Power Up (Script)								킕	\$,
Script	PowerUp								\odot
Power Up Type	None								÷
PolygonPrototype T	🗸 None					(2	7	\$,
Shader Standard	Pushb	ack					_	-	•
	Rocke	ts			_	_	_	_	
	Add Compo	nent							

17. Duplicate the *Powerup* prefab in the project view, and rename the duplicate to "*PowerupRockets*".



In the inspector for the *PowerupRockets*, change the **Power Up Type** to **Rockets**.

Inspector						_
PowerupRockets					s	Static 🔻
Tag Powerup	;	Layer	Defau	lt		\$
🔻 🙏 🛛 Transform					(🗿 다 🌣,
Position	X 0	Y	0	Z	4	
Rotation	X 0	Y	0	Z	0	
Scale	X 2	Y	2	Z	2	
▶ 🗒 SM_Icon_Gem_01 (Mesh Filter) 🔯 큐 🌣						
🕨 🛃 🗹 Mesh Renderer					(🗿 다 🔅
▶ 🤪 🗹 Box Collider					(🗿 다 🔅
🔻 🗰 Power Up (Script)					(🛯 🕸 🌣,
Script	Powerl	Jp				0
Power Up Type	Rockets					+
	Nor	e			6	1 7 0 .
Shader Standard	Pus	hback				•
	🗸 Roc	kets				
Add Component						

18. In the Project window, Navigate to **Assets** > **Course Library** > **_Source_Files** > **Materials**. Duplicate the Material called *PolygonPrototype_Texture_01* and rename it to *PowerUpRockets_Material*.



19. In the Inspector for the Material called *PowerUpRockets_Material*, adjust the Albedo Color. This will help the player distinguish the power ups from each other.

Color	Inspector		∂ •≡
	PowerupRockets_M	aterial	🛐 🕂 🌣,
1	Shader Standard		•
<i>v</i>	Rendering Mode	Opaque	\$
	Main Maps		
	o Albedo	J I	
	⊙ Metallic	0	0
	Smoothness		0.2
	Source	Metallic Alpha	+
	⊙ Normal Map		
	⊙ Height Map		
	○ Occlusion		
	⊙ Detail Mask		
	Emission		
	Tiling	X 1 Y 1	
RGB 0-25:+	Offset	X [0 Y [0	
R 0	Secondary Maps		
G 255	⊙ Detail Albedo x2		
8	⊙ Normal Map		1
	Tiling	X 1 Y 1	
A 255	Offset	X 0 Y 0	
Hexadecimal 00FF00	UV Set	000	+
▼ Swatches +=	Forward Rendering Opti	ons	
	Specular Highlights		

20. Navigate to **Assets** > **Prefabs**, and select the **PowerUpMissiles** prefab. In the Inspector, change the **Material** on the **Mesh Renderer** component to the PowerUpMissiles Material that was just created.

Inspector		а:
PowerupMis	siles	Static 🔻
Tag Powerup	 Layer Default 	•
Transform		0 : :
Position	X 0 Y 0	Z 3.02
Rotation	X 0 Y 0	Ζ 0
Scale	X 2 Y 2	Z 2
▶ SM_lcon_Gem	_01 (Mesh Filter)	Ø ‡ :
🔻 🖽 🗹 Mesh Rendere	er	0 ‡ :
Materials		
Size	1	
Element 0	PowerUpMissiles	\odot
▶ Lighting		
▼ Probes		
Light Probes	Blend Probes	•
Reflection Probes	Blend Probes	•
	 Inspector PowerupMis Tag Powerup Tag Powerup Transform Position Rotation Scale SM_lcon_Gem SM_lcon_Gem Mesh Rendered Materials Size Element 0 Lighting Probes Light Probes Reflection Probes 	 Inspector PowerupMissiles Tag Powerup Layer Default Transform Position X Y Transform Position X Y Rotation X Y Rotation X Y Y SM_lcon_Gem_01 (Mesh Filter) Size Mesh Renderer Materials Size Lighting Probes Light Probes Blend Probes Reflection Probes Blend Probes

21. Next, we are going to set up the missile prefab. In the Hierarchy, click **Create** > **3D Object** > **Sphere**. Rename the sphere to *Rocket*.



22. Instead of a Sphere Collider, we will use a Capsule Collider as we will be adjusting the scale of the GameObject. On the Rocket GameObject, Remove the **Sphere Collider** by clicking the **Gear icon** in the top right of the component and selecting **Remove Component**.

🔻 😂 🗹 Sphere Collider	🔲 🕫 🐟					
Is Trigger	Reset					
	Remove Component					
Material [Center >	Move Up					
Radius	Move Down					
Default Materia	Copy Component					
Shader Standar	Paste Component As New					
	Paste Component Values					
Ad	Find References In Scene					

23. Add a Capsule Collider by selecting Add Component > Physics > Capsule Collider

Inspector		<u> </u>
📄 🗹 Rocket		🗌 🗌 Static 🔻
Tag Untagged	‡ Layer Default	\$
🔻 🙏 🛛 Transform 🗌		🔄 🗐 🕂 🐥
Position	X 0 Y 0	Z 0
Rotation	X 0 Y 0	Z 0
Scale	X 1 Y 1	Z 1
🕨 🧮 🛛 Sphere (Mes	h Filter)	💽 🕂 🌣,
🕨 🛃 🗹 Mesh Render	er	💽 🕸 🌣
Default-Mate	erial	🛐 🌣.
Shader Stan	dard	•
	Add Component	
Q		
		-
	Physics	
<	1 11/5105	
Second Second	llider	
Sox Co	llider 🔺	
Second Se	llider e Collider ter Controller	

24. On the Capsule Collider, Change the **Direction** to **Z-axis**. We do this because our code is going to rotate the missile to face the enemy using the forward (Z) direction.

Inspector										1	•≡
🕜 🗹 Rocket									Sta	tic	•
Tag Untagged			÷ La	ye	r De	fault					ŧ
▼🙏 Transform										킕	\$
Position	Х	0		Υ	0		Z	0			
Rotation	Х	0		Υ	0		Z	0			
Scale	Х	1		Υ	1		Z	1			
🕨 🧮 🛛 Sphere (Mesh	Fi	lte	r)							킕	\$
🕨 🛃 🗹 Mesh Rendere	r									킕	\$
🔻 📒 🗹 Capsule Collid	er									킕	\$
		ሌ	Edit C	olli	der						
Is Trigger)									
Material	N	one	e (Physic	: M	ateria	al)	_	_			0
Center	Х	0		Υ	0		Z	0			_
Radius	0	.5									
Height	1										
Direction	Z	-A:	cis								ŧ
Default-Mater	-i		X-Axis								\$
Shader Stand	a		Y-Axis							_	+
		/	Z-Axis				_	-	_	_	_

25. Add the **Rocket Behaviour** script to the Rocket, by selecting **Add Component** and searching for it.

 Inspector 				<u> </u>	•=
🕜 🗹 Rocket				Static	-
Tag Untagged	÷ Laye	er Default			+
▼人 Transform				1	¢.,
Position	X 0 Y	0	Z 0		
Rotation	X 0 Y	0	Z 0		
Scale	X 1 Y	1	Z 1		
▶ 🧾 Sphere (Mest	Filter)			💽 井	۵.
🕨 🛃 🗹 Mesh Rendere	er			💽 井	\$,
🔻 📒 🗹 Capsule Collie	ler			💽 🖬	¢.,
	🔏 Edit Coll	ider			
Is Trigger					
Material	None (Physic N	1aterial)			0
Center	X 0 Y	0	Z 0		_
Radius	0.5				_
Height	1				
Direction	Z-Axis				÷
Default-Mate	rial				۰.
▶ U Shader Stand	lard				*
					_
	Add Componen	t			
Q Rocke	ť	8			
	Search				
🛤 Rocket I	Behaviour				
New scr	ipt)	>		

26. Adjust the scale of the Rocket to make it look more like a Rocket.



27. Drag the Rocket GameObject from the Hierarchy into the **Prefabs** folder. This will allow us to spawn in multiple Rockets at once.



After creating the prefab, right-click the Rocket in the Hierarchy and select **Delete**

'⊞ Hierarchy		≟ ∗≡
Create *	rAll	
🔻 🚭 Prototype	4*	*≡
🕡 Directior	al Light	
🥡 FX_Mist		>
🔹 🕨 🥡 SkyDom	e	>
🥡 Island		>
🗇 Player		
🔰 Powerup	Indicator	>
🕨 📝 Focal Po	nt	
Spawn M	anager	
Rocket	Comu	· · · · ·
	Сору	
	Paste	
	Rename	
	Duplicate	
	Dupicate	
	Delete	
	Open Prefab Asset	

28. In the Hierarchy, select the *Spawn Manager* GameObject. In the Inspector for the Spawn Manager, we now need to set up the Powerup Prefabs parameters. Change the size to **2** and assign the two Powerup prefabs to the fields.

Inspector	ê	-≡
🕤 🗹 Spawn Manag	er 🗌 Statio	•
Tag Untagged	+ Layer Default	+
▶ 🙏 Transform	🗐 a	\$,
🔻 📾 🗹 Spawn Manage	er (Script) 🛛 🗐 🗐	\$,
Script	SpawnManager	0
▼ Enemy Prefabs		
Size	2	
Element 0	🜍 Enemy	0
Element 1	🜍 EnemyFast	0
▼ Powerup Prefabs		
Size	2	
Element 0	🜍 Powerup	0
Element 1	🜍 PowerupRockets	0
Enemy Count	0	
Wave Number	1	
A	dd Component	

29. In the Hierarchy, select the Player GameObject. We will now set up the Rocket prefab. In the Inspector for the **Player Controller (Script)** component, assign the Rocket from the Prefabs folder to the **Rocket Prefab** Field.

 Inspector 		≟ •≡
Player		🗌 🗌 Static 🔻
Tag Untagged	‡ Layer Default	\$
▶ 🙏 Transform	I	🔯 🕂 🐥
🕨 🧾 Sphere (M	esh Filter)	💽 🕂 🌣
🕨 🛃 🗹 Mesh Rend	lerer	💽 🕂 🌣
🕨 😉 🗹 Sphere Co	llider	🔯 🕂 🌣
🕨 🙏 🔹 Rigidbody		💽 🕂 🌣
🔻 📾 🗹 Player Cor	troller (Script)	🔯 🕂 🐥
Script	PlayerController	0
Speed	5	
Has Powerup		
Powerup Indicato	r 😡 Powerup Indicator	0
Current Power Up	None	+
Rocket Prefab	Rocket	o

30. Save the scene and press play. Try to pick up the new powerup and then press F. Notice how rockets will spawn and start moving towards all the enemies.



Hard - Smashingly Good

1. Navigate to the Scripts folder and open up the script **Powerup.cs**. Update the PowerUpType enum to look like the following:

public enum PowerUpType { None, Pushback, Rockets, Smash }

Save the script and head back to Unity.

2. Open up the **PlayerController.cs** script. Before the **Start** method, add in the following new variables

```
public float hangTime;
public float smashSpeed;
public float explosionForce;
public float explosionRadius;
bool smashing = false;
float floorY;
```

3. Next, let's add in the Smash method. This method will be a coroutine so that we can wait while in the method. We will then launch the player in the air, and then smack them into the ground. When the hit the ground, they will do a shockwave like force to all nearby enemies.

```
IEnumerator Smash()
{
    var enemies = FindObjectsOfType<Enemy>();
    //Store the y position before taking off
    floorY = transform.position.y;
    //Calculate the amount of time we will go up
    float jumpTime = Time.time + hangTime;
    while(Time.time < jumpTime)</pre>
    {
        //move the player up while still keeping their x velocity.
        playerRb.velocity = new Vector2(playerRb.velocity.x, smashSpeed);
        yield return null;
    }
    //Now move the player down
    while(transform.position.y > floorY)
    {
        playerRb.velocity = new Vector2(playerRb.velocity.x, -smashSpeed * 2);
        yield return null;
    }
```

```
//Cycle through all enemies.
for (int i = 0; i < enemies.Length; i++)
{
    //Apply an explosion force that originates from our position.
    if(enemies[i] != null)
        enemies[i].GetComponent<Rigidbody>().AddExplosionForce(explosionForce,
transform.position, explosionRadius, 0.0f, ForceMode.Impulse);
    }
    //We are no longer smashing, so set the boolean to false
    smashing = false;
}
```

4. The final thing we need to do is add an if statement to the **Update** method. This if statement will check; is the current powerup the smash type, is the space bar pressed, and are we currently not smashing. If all are true, we'll set smashing to true and call out Smash method. The updated Update method looks like this:

```
void Update()
{
    float forwardInput = Input.GetAxis("Vertical");
    playerRb.AddForce(focalPoint.transform.forward * forwardInput * speed);
    powerupIndicator.transform.position = transform.position + new Vector3(0,
-0.5f, 0);
    if(currentPowerUp == PowerUpType.Rockets && Input.GetKeyDown(KeyCode.F))
    {
        LaunchRockets();
    }
    if(currentPowerUp == PowerUpType.Smash && Input.GetKeyDown(KeyCode.Space) &&
!smashing)
    {
        smashing = true;
        StartCoroutine(Smash());
    }
}
```

Save the script and head back to Unity.

5. We now need to create the new prefab for the smash powerup. Navigate to the Prefabs folder and drag the Powerup prefab into the Hierarchy.

E Hierarchy	ar= Came Scene	
Create * Q*All	Shaded 7 2D 상 400 도 7	
▼ € Prototype 4*		
Directional Light		
🗑 FX Mist		
▶ 🗑 SkyDome		
🗑 Island		
🗍 Player		
🗑 Powerup Indicator	,	
▶ 🗇 Focal Point		
🗍 Spawn Manager		
🜍 Powerup		
		<u> </u>
🗈 Project 📃 Console		
Create *		
Favorites	Assets > Prefabs	
-		
V 🔤 Assets		
🕨 🚞 Course Library		
Physics Materials		
Prefabs		
Scenes 🔤	Enemy EnemyFast Powerup PowerupRoc Rocket	
Scripts		
Packages		

6. In the Hierarchy, select the Powerup GameObject. In its Inspector, change the name to *PowerupSmash* and change the **Power Up Type** to **Smash**.

0 Inspec	tor		â •	≡
🍟 🗹	PowerupSma	ish	🗌 Static '	•
Tag	Powerup	🕈 Layer Defa	ult (•
Prefab	Open	Select Ov	verrides •	•
► 🙏 🛛 Tr	ransform		🔯 🕂 🛛	\$,
▶ 🔠 SN	M_Icon_Gem_	_01 (Mesh Filter)	💽 🖓 🖓	\$,
🕨 🛃 🗹 M	esh Renderer		💽 🖬 🖞	\$,
🕨 🧊 🗹 B o	ox Collider		💽 🖬 🖞	\$,
🔻 🗰 🛛 P c	ower Up (Scri	pt)	💽 🖓	\$,
Script		PowerUp		0
Power	Ир Туре	Smash		ŧ

7. In the Inspector, select the Gear icon on the Material of the GameObject and click **Select Material**. This will show you where the material is located in the project.

Inspect	tor				â •	=
🍟 🗹 [PowerupSm	ash			🗌 🗌 Static	•
Tag	Powerup		🔹 Layer 🛛	Default		;
Prefab	Open		Select	Overrid	es .	
► 🙏 — Tr	ansform				💽 🖬	\$,
▶ 🛄 SM	1_Icon_Gem	_01 (M	(esh Filter))	🔯 🗐	¢,
🕨 🔣 🗹 Me	esh Rendere	r			💽 🗐	\$,
🕨 📦 🗹 Bo	x Collider				💽 🗐	¢,
V 🗰 Po	ower Up (Scr	ipt)			[]] 二	\$,
Script		Pow	ierUp			0
Powerl	Јр Туре	Smash				Ľ
P	olygonProtot	ype_T	exture_01		💽 🖬	¢,
, 🔍 s	hader Standa	ard	Select Shad	der		
			Edit Shader	r		
	A	dd	C 1 . 14 .			
			Select Mate	erial		
			Create Mat	terial Prese	et .	
			Copy Mate	erial Prope	rties	
			Paste Mate	erial Prope	rties	
			Copy Atlas	;		
			Paste Atlas	;		
			Reset			
		_				

8. Duplicate the Material by either, selecting the Material and pressing **Ctrl + D** or **Cmd + D**, or going to **Edit** > **Duplicate**. Name the new material *PowerUpSmash_Material*.

Project Console								
Create *								
Favorites	Assets > Course Lib	orary > _Source	_Files > Materials					
Assets Course Library Course Library Source_files FBX Materials Ground Materials Pickups Powerup Indicators Sky Textures Physics Materials Physics Materials Scenes Scenes Scripts Scenes	FX_Dust	FX_Mist	PolygonMinis_Fan	PolygonPrototype	PowerupRockets	PowerUpSmash	SimpleIcons	SimpleSky
- ackages	Assets/Course Lib	orary/ Source F	iles/Materials/Powert	JpSmash Material.	mat			

9. Select the PowerupSmash GameObject in the Hierarchy. On the Mesh Renderer component, expand the Materials section and click the selector next to Element 0. Type in *PowerUpSmash* and select the Material we just created.

Select Material ×	 Inspector 	2	-≡
	PowerupS	mash 🗌 Statio	-
Q PowerupSmash 🛞	Tag Powerup	‡] Layer (Default	+
Assets Scene O	Prefab Open	Select Overrides	•
None	▶↓ Transform	Image: A state of the state	¢.,
B Poweropsmasn_Material	▶ 🗒 SM_Icon_Ge	m_01 (Mesh Filter) 🛛 🔯 🗐	\$,
	🔻 🛃 🗹 Mesh Render	rer 🔯 🗐 🗐	\$,
	Materials		
	Size	1	
	Element 0	PowerUpSmash_Material	o
	Light Probes	Blend Probes	+
	Reflection Probes	Blend Probes	+
	Anchor Override	None (Transform)	\odot
	Cast Shadows	On	+
	Receive Shadows		
	Motion Vectors	Per Object Motion	+
	Lightmap Static		
	① To enable gener Renderer, pleas	ation of lightmaps for this Mesh e enable the 'Lightmap Static' propert	у.
	Dynamic Occluded		
	▶ 🥡 🗹 Вох Collider	💽 🗐	\$.
	🔻 🗰 Power Up (S	cript) 💿 🗐	\$.
	Script	PowerUp	\odot
	Power Up Type	Smash	÷
Rowerlin Space Material (Mate	PowerUpSm	ash Material 🛛 🗐 🖬	۵.
Foweropsmasn_material (Mate	Shader Star	ndard	•
	Ad	ld Component	

10. Expand the Material at the bottom of the Inspector and change the albedo value. We put ours as red.

Color	PowerUpSmas	sh_Material	🔯 🕂 🐥
1	Rendering Mode	Opaque	+)
	Main Maps		
	S Albedo	— 1	
	⊙ Metallic	0	0
	Smoothness		0.2
	Source	Metallic Alpha	\$
	⊙ Normal Map		
	⊙ Height Map		
	© Occlusion		
	⊙ Detail Mask		
	Emission		
	Tiling	X 1	Y 1
	Offset	X 0	Y 0
RGB 0-255	Secondary Maps		
R 255	⊙ Detail Albedo x2		
G	⊙ Normal Map		1
	Tiling	X 1	Y 1
в	Offset	X 0	Y 0
A 255	UV Set	UV0	+
Hexadecimal FF0000	Forward Rendering	Options	
▼ Swatches →	≡ Specular Highlights		
	Reflections		

11. Navigate to the Prefabs folder, and drag the PowerupSmash GameObject from the Hierarchy into the Project view. When the Create Prefab pop-up appears, click **Original Prefab**. After you have turned it into a prefab, delete the PowerupSmash GameObject from the Hierarchy.



12. In the Hierarchy, select the SpawnManager GameObject. In the Inspector, on the Spawn Manager component, change the Powerup Prefabs size to 3. Click the selector button and select the PowerupSmash Prefab.

Inspector		≧ •≡
Spawn Manager		📃 🗌 Static 🔻
Tag Untagged	‡ Layer Default	\$
Transform		🔯 및 및 🔅
	(Corint)	
Script	SpawpManager	
- Script	Spawimanager	
Chemy Pretabs		
Size	2	
Element 0	Second Se	
Element 1	👽 Enemy Fast	0
V Powerup Prefabs		
Size	3	
Element 0	Powerup	0
Element 1	PowerupRockets	
Element 2	PowerupSmash	0
Energy Count	0	
Ways Number		
wave Number	1	
	Add Component	
	Add Component	
	 Inspector Spawn Manager Tag Untagged Transform Spawn Manager (Script Enemy Prefabs Size Element 0 Element 1 Powerup Prefabs Size Element 1 Element 1 Element 1 Element 2 Enemy Count Wave Number 	 Inspector Spawn Manager Tag Untagged ‡ Layer Default Transform Script Spawn Manager (Script) Script SpawnManager Enemy Prefabs Size Element 0 EnemyFast Powerup Prefabs Size Element 1 Powerup Prefabs Size Element 0 Powerup Rockets Element 1 PowerupSmash Enemy Count Marke Number Add Component

13. Save the scene and press play. Pick-up a red powerup and press space to use the smash attack



14.

15.

Expert - Boss Battle

After a certain number of waves, program a mini "boss battle," where the boss has some completely new abilities. For example, maybe the boss can fire projectiles at you, maybe it is extremely agile, or maybe it occasionally generates little minions that come after you.

1. Navigate to the Scripts folder and open up the **Enemy** script. Below the current variable declarations, add the following:

```
public bool isBoss = false;
public float spawnInterval;
private float nextSpawn;
public int miniEnemySpawnCount;
private SpawnManager spawnManager;
```

2. Inside the **Start** method, let's check if the enemy is a boss. If it is, we will set up the spawnManager variable. The update Start method should look like this:

```
void Start()
{
    enemyRb = GetComponent<Rigidbody>();
    player = GameObject.Find("Player");
    if (isBoss)
    {
        spawnManager = FindObjectOfType<SpawnManager>();
    }
}
```

3. Next let's update the **Update** method to call a method we'll make shortly on the SpawnManager. The updated method should look like this:

```
void Update()
{
    Vector3 lookDirection = (player.transform.position -
transform.position).normalized;
    enemyRb.AddForce(lookDirection * speed);
    if(isBoss)
    {
        if(isBoss)
        {
            if(Time.time > nextSpawn)
            {
                nextSpawn = Time.time + spawnInterval;
        }
    }
}
```

```
spawnManager.SpawnMiniEnemy(miniEnemySpawnCount);
    }
if(transform.position.y < -10)
{
    Destroy(gameObject);
}</pre>
```

4. Save the script and open up the **SpawnManager** script. Below the current variable decelerations, add the following:

```
public GameObject bossPrefab;
public GameObject[] miniEnemyPrefabs;
public int bossRound;
```

}

5. Below the **GenerateSpawnPosition** method, we will need to create two new methods. The first will handle spawning the boss. The second will handle spawning the mini enemies. Lets start with the boss method. Add the following:

```
void SpawnBossWave(int currentRound)
{
    int miniEnemysToSpawn;
    //We dont want to divide by 0!
    if (bossRound != 0)
    {
        miniEnemysToSpawn = currentRound / bossRound;
    }
    else
    {
        miniEnemysToSpawn = 1;
    }
    var boss = Instantiate(bossPrefab, GenerateSpawnPosition(),
bossPrefab.transform.rotation);
    boss.GetComponent<Enemy>().miniEnemySpawnCount = miniEnemysToSpawn;
}
```

Let's break this down. First we are dividing the current round by the specified boss round and then storing it into a variable for future use. Then we spawn in the boss and set the amount of mini enemies they will spawn to the variable we just created.

6. Next let's create the mini enemy spawning method. Below the **SpawnBossWave** method, add the following:

```
public void SpawnMiniEnemy(int amount)
{
```

```
for(int i = 0; i < amount; i++)
{
    int randomMini = Random.Range(0, miniEnemyPrefabs.Length);
    Instantiate(miniEnemyPrefabs[randomMini], GenerateSpawnPosition(),
miniEnemyPrefabs[randomMini].transform.rotation);
    }
}</pre>
```

7. The last thing we need to do in the **SpawnManager** script is to adjust the Update method. The updated **Update** method should look like this:

```
void Update()
{
    enemyCount = FindObjectsOfType<Enemy>().Length;
    if(enemyCount == 0)
    {
        waveNumber++;
        //Spawn a boss every x number of waves
        if (waveNumber % bossRound == 0)
        {
            SpawnBossWave(waveNumber);
        }
        else
        {
            SpawnEnemyWave(waveNumber);
        }
        //Updated to select a random powerup prefab for the Medium Challenge
        int randomPowerup = Random.Range(0, powerupPrefabs.Length);
        Instantiate(powerupPrefabs[randomPowerup], GenerateSpawnPosition(),
powerupPrefabs[randomPowerup].transform.rotation);
    }
}
```

The code we added, uses a modulus operator to check if we are currently on a boss round. For example, if bossRound is set to 3 and the waveNumber was set to 6 the modulus operator would return 0 as the can be divided without a remainder.

Save the script and return to Unity.

8. Back in the editor, we will need to create the new prefabs for the boss and mini enemies. Let's start with the boss. Right-click in the Hierarchy and select **3D** > **Sphere**. Rename the Sphere to Boss and adjust the scale to 2.5 on all axes.

O Inspector				∂ •≡
Boss				🗌 🗌 Static 🔻
Tag Untagged	\$	Layer Default		\$
▼人 Transform				💿 🕂 🌣
Position	X 0	Y O	Z	0
Rotation	X 0	Y 0	Z	0
Scale	X 2.5	Y 2.5	Z	2.5

9. Next, add two components to the Boss GameObject. The first is a Rigidbody and the second is the Enemy script.

Imported Boss Tag Untagged + Layer Default + Transform Position X Position X O Y <th>6 Inspector</th> <th></th> <th></th> <th></th> <th></th> <th><u> </u></th> <th>_</th>	6 Inspector					<u> </u>	_
Tag Untagged ‡ Layer Default ‡ Transform Imaged ‡ Layer Default ‡ Position × 0 Y 0 Z 0 Rotation × 0 Y 0 Z 0 Scale × 2.5 Y 2.5 Z 2.5 Sphere (Mesh Filter) Image Image Image Image Image Image Image Image Image Image Is Sphere Collider Image Image Image Image Is State Image Image Image Image Is Image Image Image Image Image Is Image Image Image Image Image Is Image <t< td=""><td>Boss</td><td></td><td>_</td><td></td><td>Stat</td><td>ic i</td><td>-</td></t<>	Boss		_		Stat	ic i	-
Transform Position X 0 Y 0 Y 0 Y 0 Y 0 Y 0 Y 0 Y 0 Y 0 Y 0 Y 0 Y 0 Y 0 Y 0 Y	Tag Untagged	t) Lawar (Default			Juan		
A Transform Position X 0 Y 0	Tag Oncaggeo	Layer Delaute	_	6			-
Position X 0 Y 0 Z 0 Rotation X 0 Y 0 Z 0 Scale X 2.5 Y 2.5 Z 2.5 Sphere (Mesh Filter) I 1 * Image: I	Transform	× •	- 0		2	1	φ,
Rotation X 0 Y 0 2 0 Scale X 2.5 Y 2.5 Z 2.5 Sphere (Mesh Filter) I I I Image: Imag	Position	X 0 Y 0	2	0			-
Scale X 2.5 Y 2.5 Z 2.5 ■ Sphere (Mesh Filter) ■ ↓ ◆ ● Mesh Renderer ■ ↓ ◆ ● Sphere Collider ■ ↓ ◆ ● Sphere Collider ■ ↓ ◆ Material None (Physic Material) 0 Y 0 Center X 0 X 0 Y 0 Radius 0.5 A Rigidbody ● Rigidbody ● Angular Drag 0.05 Use Gravity Is Kinematic Interpolate None Constraints ● Enemy (Script) Script ● Enemy (Script) Speed 3 Is Boss Spawn Interval 0	Rotation	X 0 Y 0	2	0			-
Sphere (Mesh Filter) □ ↓ ↓ Image: Mesh Renderer □ ↓ ↓ Image: Sphere Collider □ ↓ ↓ Image: Sphere Collider □ ↓ ↓ Is Trigger Image: Sphere Collider Is Trigger Image: Sphere Collider Material None (Physic Material) Center × 0 ↓ 0 ⊥ 2 0 Radius 0.5 A Rigidbody □ ↓ ↓ Mass 1 Drag 0 Angular Drag 0.05 Use Gravity Image: Sphere Collison Discrete Interpolate None Constraints Image: Sphere Collison Collison Discrete Script Enemy Speed 3 Is Boss Image: Sphere Collison	Scale	X 2.5 Y 2.5	2	2.5	_		
Mesh Renderer Sphere Collider Is Trigger Material None (Physic Material) Center X 0 Y 0 Z 0 Radius 0.5 Rigidbody Image: Sphere Collider X 0 Y 0 Z 0 Radius 0.5 A Rigidbody Image: Sphere Collider X 0 Y 0 Z 0 Radius 0.5 A Rigidbody Image: Sphere Collider Image:	Sphere (Mesh Filt)	er)			2	1	\$,
Sphere Collider Sphere Collider Is Trigger Material None (Physic Material) Center X 0 Y 0 Z 0 Radius 0.5 Rigidbody Rigidbody Mass 1 Drag 0 Angular Drag 0.05 Use Gravity Is Kinematic Interpolate None Collision Detection Discrete Constraints Part * Constraints Part * Speed 3 Is Boss Spawn Interval 0	🕨 🛃 🗹 Mesh Renderer				2	1	\$,
Is Trigger Material None (Physic Material) © Center X 0 Y 0 Z 0 Radius 0.5 A Rigidbody Is Rigidbody Mass 1 Drag 0 Angular Drag 0.05 Use Gravity Is Kinematic Interpolate None Collision Detection Discrete Is Enemy Speed 3 Is Boss Spawn Interval 0 Maini Enemy Spawn Count	🔻 😂 🗹 Sphere Collider				2	1	\$,
Is Trigger Material None (Physic Material) Center X 0 Y 0 Z 0 Radius 0.5 Rigidbody Mass 1 Drag 0 Angular Drag 0.05 Use Gravity Is Kinematic Interpolate None Collision Detection Discrete Constraints Enemy (Script) Speed 3 Is Boss Spawn Interval 0 Mini Enemy Spawn Count 0		🔥 Edit Collider					
Material None (Physic Material) Center X 0 Y 0 Z 0 Radius 0.5 Angular Drag 0 Angular Drag 0.05 Use Gravity ✓ Is Kinematic I Interpolate None Collision Detection Discrete Ø € Enemy (Script) Ø 2 0 Speed 3 Is Boss I Spawn Interval 0 Material 0	Is Trigger						
Center X 0 Y 0 Z 0 Radius 0.5 Rigidbody 2 0 Mass 1 Drag 0 Angular Drag 0.05 Use Gravity 2 Is Kinematic 1 Interpolate None + Collision Detection Discrete + Constraints 2 Constraints 2 Speed 3 Is Boss 1 Spawn Interval 0 Mini Enemy Spawn Count 0	Material	None (Physic Material)	-				0
Radius 0.5 Arigidbody Image: I	Center	X 0 Y 0	zſ	0		-	٦
Aigidbody Mass Drag Drag O Angular Drag 0.05 Use Gravity Is Kinematic Interpolate None Collision Detection Discrete Constraints Image: Speed Speed Spawn Interval Mini Enemy Spawn Count	Radius	0.5					۲
Mass 1 Drag 0 Angular Drag 0.05 Use Gravity Image: Constraint of the second	🛚 🙏 Rigidbody			(2	1	\$
Drag 0 Angular Drag 0.05 Use Gravity Image: Constraint of the second of the se	Mass	1	_		_	-	
Angular Drag 0.05 Use Gravity Is Kinematic Interpolate None + Collision Detection Discrete + Constraints Constraints Constraints Script Enemy OC Speed 3 Is Boss Spawn Interval 0 Mini Enemy Spawn Count 0	Drag	0					
Use Gravity Is Kinematic Interpolate None Collision Detection Discrete Constraints Constraints Enemy (Script) Script Speed 3 Is Boss Spawn Interval 0 Mini Enemy Spawn Count 0	Angular Drag	0.05					
Is Kinematic Interpolate None Collision Detection Discrete Constraints Constr	Use Gravity						
Interpolate None + Collision Detection Discrete + Constraints Image: Constraints + Image: Image: Constraints Image: Constraints Image: Constraints Image: Image: Constreta Image: Constraints Im	Is Kinematic						
Collision Detection Discrete ≠ Constraints Image: Constraints Image: Constraints Image: Image:	Interpolate	None					÷
Constraints	Collision Detection	Discrete					÷
Image: Second	Constraints						
Script Enemy O Speed 3 Is Boss S Spawn Interval 0 Mini Enemy Spawn Count 0	🛙 🎟 🗹 Enemy (Script)				2	; ! :	\$
Speed 3 Is Boss Is Spawn Interval 0 Mini Enemy Spawn Count 0	Script	Enemy .					0
Is Boss Spawn Interval 0 Mini Enemy Spawn Count 0	Speed	3					
Spawn Interval 0 Mini Enemy Spawn Count 0	Is Boss						
Mini Enemy Spawn Count 0	Spawn Interval	0					
	Mini Enemy Spawn Count	0					

10. On the **Sphere Collider** click the selector icon to the right of the Material field. Look for the Bouncy Material and select it. This will ensure the boss has the same behaviour as the enemies when colliding with GameObjects.

Select PhysicMaterial x	 Inspector 			≧ *≡
Q	✓ Boss Tag Untagged	+ Laye	r Default	🔄 🗌 Static 🔻
Assets Scene None	V L Transform Position	xo	0	Z 0
	Rotation Scale	X 0 X 2.5	(0 (2.5	Z 0 Z 2.5
	Sphere (Mesh Fil	lter)		[2] 二 章
	▼ 🤩 🗹 Mesh Renderer ▼ 🤐 🗹 Sphere Collider			·····································
		🔏 Edit Colli	der	
	Is Trigger Material	Bouncy		G
	Center Radius	X 0	0	Z 0

11. On the **Enemy** component, we need to set up some values. Change the **Speed** value to **2**, check the **Is Boss** checkbox, and Change Spawn Interval to **3**.

Inspector							2	•≡
🕜 🗹 Boss] 🗆 :	Stat	ic	•
Tag Untagged		🕴 Layer	Default					\$
▼人 Transform				_			Į.	\$,
Position	X 0	Y	0	Ζ	0			
Rotation	X 0	Y	0	Ζ	0			
Scale	X 2.5	Y	2.5	z	2.5			
🕨 🧾 Sphere (Mesh Filt	er)						į.	\$,
🕨 🛃 🗹 Mesh Renderer							į.	\$,
🕨 😉 🗹 Sphere Collider							ł.	\$,
🕨 🙏 🛛 Rigidbody							ł	\$,
🔻 📾 🗹 Enemy (Script)					I	2	1	\$,
Script	🕞 Enen	ру						\odot
Speed	2							
Is Boss								
Spawn Interval	3							
Mini Enemy Spawn Count	0							

12. On the **Mesh Renderer** component, we will set a material. Navigate to the folder *Course Library* > *Materials*. We selected *PolygonPrototype_Texture_Grid_07* and adjusted the albedo value to give a turquoise color. Drag this onto the Boss GameObject.

Tag Untagged	t Laver Default	
	, Layer (Denaid]
Position	×0 ×0 70	-1: ×c
Position		
Scale	X 2.5 Y 2.5 Z 2.5	
Sobere (Mesh	Filter)	고 🌣
V 📴 🗹 Mesh Rendere	r (1.0
Materials		
Size	1	
Element 0	PolygonPrototype Texture Grid 07	0
Light Probes	Blend Probes	+
Reflection Probes	Blend Probes	+
Anchor Override	None (Transform)	0
Cast Shadows	On	+
Receive Shadows		
Receive Shadows Motion Vectors	Per Object Motion	+
Receive Shadows Motion Vectors	Per Object Motion	+
Receive Shadows Motion Vectors Lightmap Static	Per Object Motion	\$
Receive Shadows Motion Vectors Lightmap Static ① To enable generati the 'Lightmap Stat	Per Object Motion ion of lightmaps for this Mesh Renderer, please er ic' property.	¢ nable
Receive Shadows Motion Vectors Lightmap Static ① To enable generati the 'Lightmap State Dynamic Occluded	Per Object Motion ion of lightmaps for this Mesh Renderer, please er ic' property.	¢ nable
Receive Shadows Motion Vectors Lightmap Static ① To enable generati the 'Lightmap Stat Dynamic Occluded	Per Object Motion Per Object Motion ion of lightmaps for this Mesh Renderer, please er ic' property. r	÷ nable
Receive Shadows Motion Vectors Lightmap Static ① To enable generati the 'Lightmap State Dynamic Occluded Oynamic Occluded Sphere Collide	Per Object Motion ion of lightmaps for this Mesh Renderer, please er ic' property. r	; nable
Receive Shadows Motion Vectors Lightmap Static ① To enable generati the 'Lightmap State Dynamic Occluded ② Sphere Collide A Rigidbody ③ @ Enemy (Script	Per Object Motion ion of lightmaps for this Mesh Renderer, please er ic' property. r	; nable] 7
Receive Shadows Motion Vectors Lightmap Static To enable generati the 'Lightmap Stat Dynamic Occluded Sphere Collide Rigidbody Gript	Per Object Motion Per Object Motion ion of lightmaps for this Mesh Renderer, please er ic' property. r Enemy Enemy	; nable] ;; ☆] ;; ☆] ;; ☆
Receive Shadows Motion Vectors Lightmap Static To enable generati the 'Lightmap Stat Dynamic Occluded Sphere Collide Rigidbody Gript Speed	Per Object Motion Per Object Motion ion of lightmaps for this Mesh Renderer, please er ic' property. r Enemy Z	+ nable
Receive Shadows Motion Vectors Lightmap Static To enable generati the 'Lightmap State Dynamic Occluded Sphere Collide Rigidbody C Press Script Speed Is Boss	Per Object Motion ion of lightmaps for this Mesh Renderer, please er ic' property. r Enemy 2 V V V C C C C C C C C C C	; nable
Receive Shadows Motion Vectors Lightmap Static To enable generati the 'Lightmap Stat Dynamic Occluded Sphere Collide Script Speed Is Boss Spawn Interval	Per Object Motion ion of lightmaps for this Mesh Renderer, please er ic' property. r Enemy C S S S S S S S S S S S S	; nable

13. Drag the Boss GameObject from the Hierarchy into the Prefabs folder to turn it into a prefab. After turning it into a prefab, delete it from the Hierarchy.

🗈 Project 🔋 Console							
Create *							
Favorites	Assets > Prefabs						
 Assets Course Library Source_Files Ground Materials Pickups Powerup Indicators Sky Textures Physics Materials Prefabs Scenes Scripts Packages 	Boss	Enemy	EnemyFast	Powerup	PowerupRockets	PowerupSmash	Rocket

14. Drag the Enemy and EnemyFast prefabs into the Hierarchy. We will use these as a base for our mini enemies. Select them both in the Hierarchy and adjust the scale to 1. Add the prefix of *Mini* to both GameObjects names.

Inspector	≜ •≡	Inspector	⊇ •≡
😭 🖌 MiniEnemy	🗌 Static 🔻	😭 🖌 MiniEnemyFast	🗌 Static 🔻
Tag Enemy + Layer	Default ‡	Tag Enemy 🕴 Layer	Default +
Prefab Open Select (Overrides 🔹	Prefab Open Select	Overrides 🔹
▼人 Transform	🔯 🕸 🌣	▼ 🙏 Transform	💽 🗟 🌣
Position X 0 Y 0.15	Z 6	Position X 0 Y 0.15	Z 6
Rotation X 0 Y 0	Z 0	Rotation X 0 Y 0	Z 0
Scale X 1 Y 1	Z 1	Scale X 1 Y 1	Z 1

15. Drag the GameObjects from the Hierarchy to the Prefabs folder. When prompted with a window, select Original Prefab. After creating the prefabs, delete the GameObjects from the Hierarchy.

Create Prefab	×
\triangleleft	Would you like to create a new original Prefab or a variant of this Prefab?
	Original Prefab Variant Cancel

16. The final thing we need to do is to set up the **Spawn Manager** variables. First drag the Boss prefab from the Project view into the **Boss Prefab** field.



17. Next, drag the two MiniEnemy's into the Mini Enemy Prefabs field.

		∂ •≡	Inspector		<u> -</u>
			😭 🖌 Spawn Manage	r 🗌 Stat	tic 🔻
Assets > Pre	fabs		Tag Untagged	tayer Default	\$
۲			▼ 人 Transform Position Rotation	X -0.1609665 Y 0.7849145 Z -0.41 X 0 Y 0 Z 0	;¦ ¢
			Scale	X 1 Y 1 Z 1	
Boss	Enemy	EnemyFast	🔻 🖬 🗹 Spawn Manager	(Script)	<u>,</u> 0
			Script	🖻 SpawnManager	0
			▼ Enemy Prefabs		
			Size	2	
			Element 0	😂 Enemy	0
MiniEnemy	MiniEnemy	Powerup	Element 1	😂 Enemy Fast	0
			▼ Powerup Prefabs		
			Size	3	
			Element 0	🜍 Powerup	0
			Element 1	PowerupRockets	0
PowerupRo	PowerupSm	Rocket	Element 2	📦 Powerup Smash	0
			Enemy Count	0	
			Wave Number	1	
			Boss Prefab	📦 Boss	0
			🔻 Mini Enemy Prefabs		
			Size	0	
			Boss Round	0	
				Add Component	

18. The last thing we need to do is change the **Boss Round**. This will be used to determine which round will be the boss one. If you put 5, after 4 normal rounds a boss round would happen. To make it a bit quicker to show the boss, we put our **Boss Round** value to **2**.

🔻 📾 🗹 Spawn Manager	(Script) 🛛 🔯 🗐 🗐	\$,
Script	🖬 SpawnManager	\odot
▼ Enemy Prefabs		
Size	2	
Element 0	🜍 Enemy	o
Element 1	🜍 EnemyFast	0
▼ Powerup Prefabs		
Size	3	
Element 0	🜍 Powerup	o
Element 1	PowerupRockets	Θ
Element 2	📦 Powerup Smash	Θ
Enemy Count	0	
Wave Number	1	
Boss Prefab	🜍 Boss	Θ
🔻 Mini Enemy Prefabs		
Size	2	
Element 0	🞯 MiniEnemy	Ο
Element 1	🜍 MiniEnemyFast	ο
Boss Round	2	

19. Save the scene and press play. Every 2 rounds a boss should spawn. The boss will keep spawning mini enemies until you knock them off the platform.

