Player Attack|Coin Collection|Audio Management

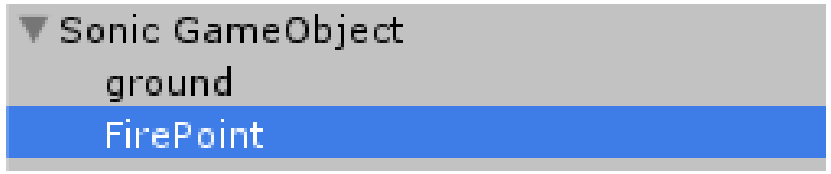
Exercise #27 – Player Shooting Mechanism

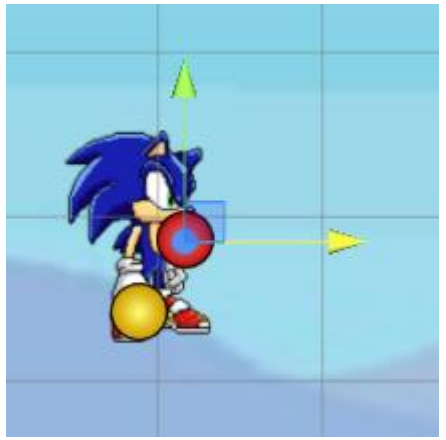
- Choose a suitable particle, weapon or bullet that you want your character to fling at the enemies to kill them
- Resize it so that it's not too big
- Add a rigidbody2D and change the gravity scale to *0*
- Add a collider2D and set IsTrigger to be true
- Add a script and name it BulletController. We'll come back to it later
- Turn the bullet into a **prefab**

Exercise #27 – Player Shooting Mechanism (Cont.)

- Create an Empty GameObject and name it FirePoint. This will be the point from which you fire bullets/particles/etc

- Change the object into a red sphere to be able to see it. Place it where you want on the player and make it a child of the player





Exercise #27 – Player Shooting Mechanism (Cont.)

- Go to the Player's controller script to give him/her the ability to fire, same as you game them the ability to walk, jump, etc!

```
public KeyCode Return;
public Transform firepoint;
public GameObject bullet;
```

- In the Update() function

```
if (Input.GetKeyDown(Return)) //When user presses the Enter button
{
    Shoot();
}
```

- Create a new function that will handle shooting

```
public void Shoot()
{
    Instantiate(bullet, firepoint.position, firepoint.rotation); //A spiky bullet gameobject is created, and its point of
    //birth is the exact position of firePoint (the red sphere we already placed on the player character in Unity)
}
```

Exercise #27 – Player Shooting Mechanism (Cont.)

- Go back to Unity and give the FirePoint and Bullet variables their correct game objects by dragging and dropping into the text boxes
  - (Also don't forget to give the Return KeyCode its correct button)

  - Make sure you drag the PREFAB of the spiky bullet! This will allow the player to shoot as many times as they want

Exercise #27 – Player Shooting Mechanism (Cont.)

- Now go back to the BulletController script and write the following

```
public class BulletController : MonoBehaviour {

    public float speed; //speed of the bullet
     //create a variable in which the player (and by extension his/her controller could be stored)

    // Use this for initialization
    void Start () {

        Controller player;

        player = FindObjectOfType<Controller>();//search for the object called Controller (the script that controls your player)

         if(player.transform.localScale.x < 0) //if player is moving to the left along x-axis
        {
            speed = -speed; //convert the speed to negative value
            transform.localScale = new Vector3(-(transform.localScale.x), transform.localScale.y, transform.localScale.z);
        }
    }
```

Exercise #27 – Player Shooting Mechanism (Cont.)

- Next are the Update and OnTriggerEnter2D functions

```
    // Update is called once per frame
    void Update () {

        GetComponent<Rigidbody2D>().velocity = new Vector2(speed, GetComponent<Rigidbody2D>().velocity.y);//with every frame, the bullet
        //travels at the speed you determined with no change to the y-axis value because you want bullet to travel horizontally

    }

    void OnTriggerEnter2D(Collider2D other) //the bullet only affects any game object that is tagged enemy!
    {
        if(other.tag =="Enemy")
        {
            Destroy(other.gameObject);

                Destroy(this.gameObject);

        }

    }
}
```

Exercise #27 – Player Shooting Mechanism (Cont.)

- **TIP: If you want to create a particle effect so that your enemy explodes to bits the moment your bullet hits them, check out these 2 tutorials!**

- Particle Effects! - Unity 2D Platformer Tutorial - Part 5
  https://www.youtube.com/watch?v=vwUahWrY9Jg

- Shooting Projectiles & Camera Control - Unity 2D Platformer Tutorial - Part 8
  https://www.youtube.com/watch?v=8aVZuL9ocrk

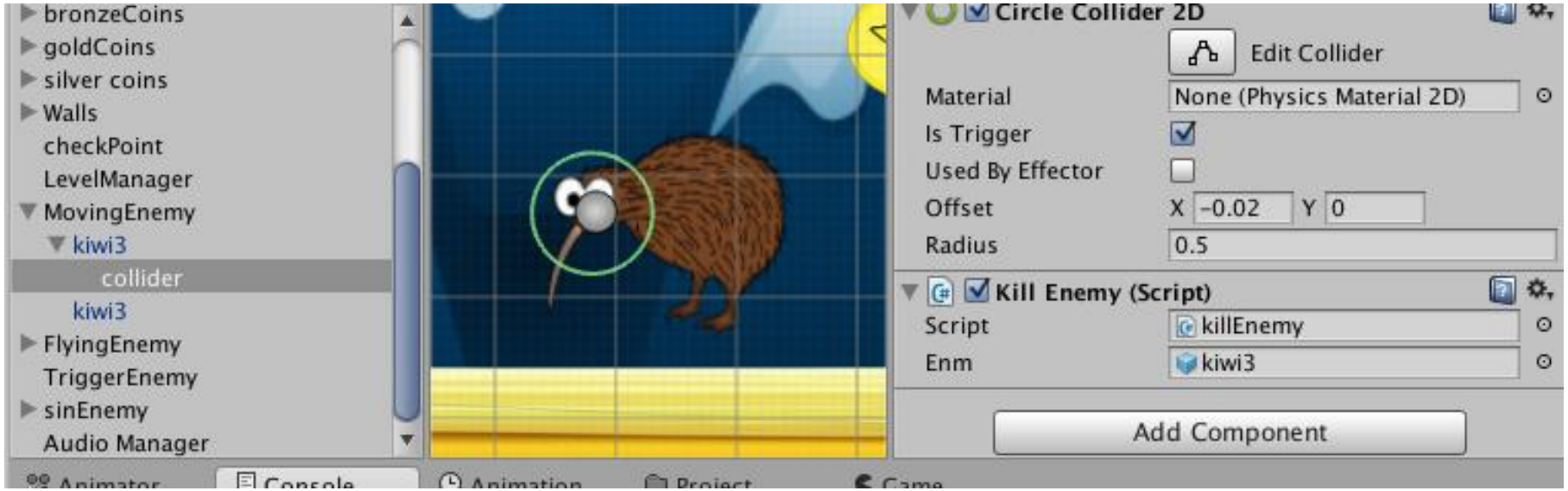
Exercise #28 – Stomp on an enemy's head

- You will need to have multiple 2Dcolliders on the enemy GameObject, and for the stomping mechanism, the collider must have *IsTrigger* checked

- In order to have two different colliders act in different ways, you must use two separate game objects. The way to get around this is to put the one collider on a game object, then add the second on a child of the game object

- To do this:
  - Create an empty game Object and make it a child of your enemy , add a circle collider to the empty object and also enable IsTrigger
  - Now the child object will need to have its own Monobehaviour script attached to recognize the new collider

Exercise #28 – Stomp on an enemy's head (Cont.)



Exercise #28 – Stomp on an enemy's head (Cont.)

- Add a script called KillEnemy to the child game object. This script executes when the player's collider collides with the enemy's:

```
public GameObject enm;
```

- Create a variable of game object and name it enm (enemy)

```
void OnTriggerEnter2D(Collider2D collider)
{
    if(collider.tag == "Player")
    {
        Destroy(enm);
    }
}
```

- Create an OnTrigger function and write a statement where if the collider that collides with the child game object belongs to the player, the enemy game object is destroyed

Exercise #29 - Audio Manager

- Create a script that contains everything needed to control the sounds of your game, either the background music or the sound effects on different events. Call it AudioManager and attach it to a new empty gameObject called “Audio Manager”

- We need the Audio Manager to have 3 things:
  - Two Audio source to hold the sound effects and the Background music
  - Range of High and Low pitches
  - An instance of Audio Manager itself for other scripts to call it.

Exercise #29 - Audio Manager (Cont.)

- Variables needed:

```
public class AudioManager : MonoBehaviour
{
    public AudioSource efxSource;                   //Drag a reference to the audio source which will play the sound effects.
    public AudioSource musicSource;                 //Drag a reference to the audio source which will play the music.
    public static AudioManager instance = null;     //Allows other scripts to call functions from SoundManager.
    public float lowPitchRange = .95f;              //The lowest a sound effect will be randomly pitched.
    public float highPitchRange = 1.05f;            //The highest a sound effect will be randomly pitched.
```

Exercise #29 - Audio Manager (Cont.)

- In the Start() function, we name it Awake() so that it is called once the game is played. Here, we just create an instance of AudioManager and make sure it is never destroyed when reloading new scenes

```
void Awake ()
{
    //Check if there is already an instance of SoundManager
    if (instance == null)
        //if not, set it to this.
        instance = this;
    //If instance already exists:
    else if (instance != this)
        //Destroy this, this enforces our singleton pattern so there can only be one instance of SoundManager.
        Destroy (gameObject);

    //Set SoundManager to DontDestroyOnLoad so that it won't be destroyed when reloading our scene.
    DontDestroyOnLoad (gameObject);
}
```

Exercise #29 - Audio Manager (Cont.)

- Create a PlaySingle() function

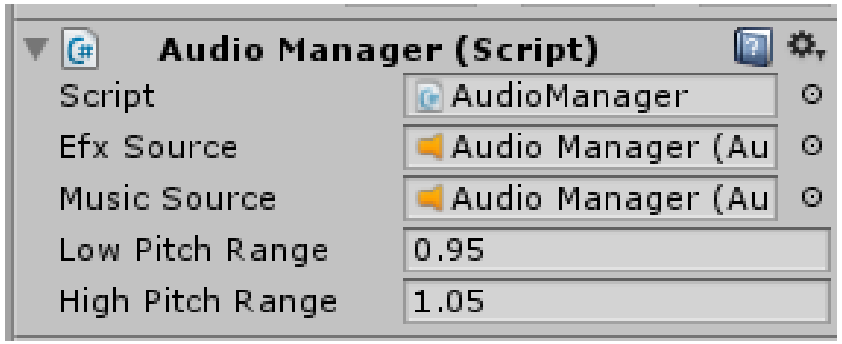
```
//Used to play single sound clips.
public void PlaySingle(AudioClip clip)
{
    //Set the clip of our efxSource audio source to the clip passed in as a parameter.
    efxSource.clip = clip;

    //Play the clip.
    efxSource.Play ();
}
```

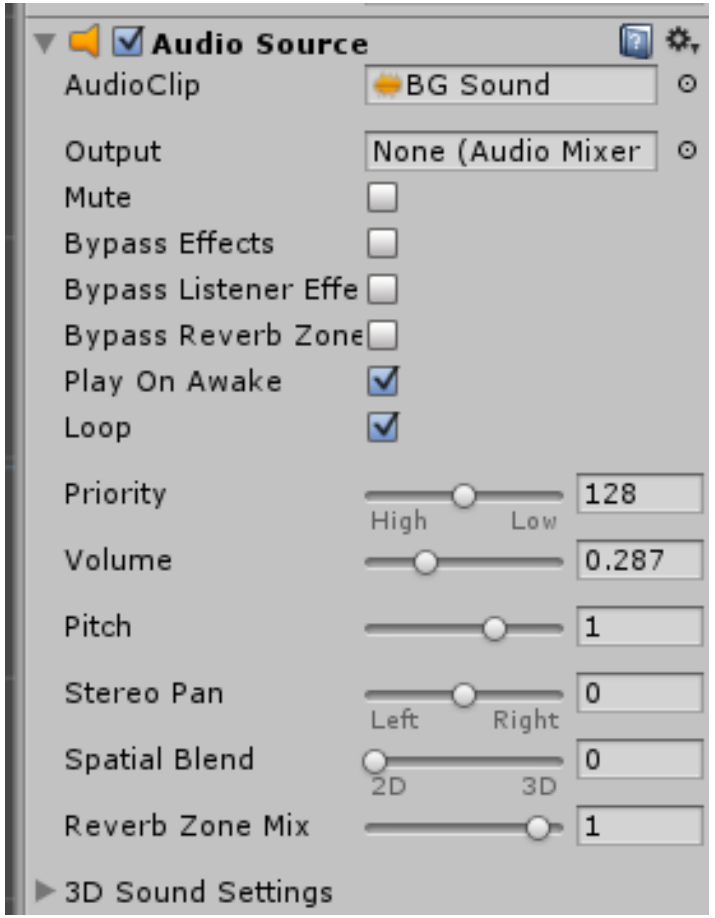
- Create a RandomizeSfx () function

```
//RandomizeSfx chooses randomly between various audio clips and slightly changes their pitch.
public void RandomizeSfx (params AudioClip[] clips)
{
    //Generate a random number between 0 and the length of our array of clips passed in.
    int randomIndex = Random.Range(0, clips.Length);

    //Choose a random pitch to play back our clip at between our high and low pitch ranges.
    float randomPitch = Random.Range(lowPitchRange, highPitchRange);

    //Set the pitch of the audio source to the randomly chosen pitch.
    efxSource.pitch = randomPitch;

    //Set the clip to the clip at our randomly chosen index.
    efxSource.clip = clips[randomIndex];

    //Play the clip.
    efxSource.Play();
}
```

Exercise #29 - Audio Manager (Cont.)

- Now, to initialize the two AudioSource variables in the Audio Manager script, we need to create two Audio Source components in the Audio Manager game object and give them their suitable audio clips to be run afterwards. For the first Audio source , we need to check the Play On Awake and Loop check boxes since that Audio Source holds our Background music that will be looping the whole time. While the second Audio Source, we leave it as it is, just uncheck the Play On Awake check box.
- We, then, drag and drop those two Audio Sources into their corresponding variables in the Audio Manager script.





Exercise #30 – Player Sound Effects

- If you need to add sound effect when your player jumps:
  - Go to the “Controller” script and add two AudioClip variables, to hold two different sounds that will be randomly played.

```
public AudioClip jump1;
public AudioClip jump2;
```

  - Then, inside the Jump() function, where the Jumping animation take place, call the RandomizeSfx() function and pass to it your two AudioClip variables.

```
void Jump()
{
    GetComponent<Rigidbody2D>().velocity = new Vector2(GetComponent<Rigidbody2D>().velocity.x, jumpHeight); //player character jumps
    //vertically along the y-axis without disrupting horizontal walk
     AudioManager.instance.RandomizeSfx(jump1, jump2);
}
```

  - Following the same technique, you can add sound effects to every action / event taking place in your game.

Exercise #31 – Enemy Hit Sound Effects

- You will need to add sound effect when an enemy hits your player, to do so:
  - Go to the “Enemy Controller” script and add two AudioClip variables, to hold two different sounds that will be randomly played.

```
public AudioClip hit1;
public AudioClip hit2;
```

  - Then, inside the OnTriggerEnter2D() function, where the damage takes effect when an enemy hits a player, call the RandomizeSfx() function and pass to it your two AudioClip variables.

```
void OnTriggerEnter2D(Collider2D other)
{
    if (other.tag == "Player")
    {
        AudioManager.instance.RandomizeSfx(hit1, hit2);
        FindObjectOfType<PlayerStats>().TakeDamage(damage);
    }
}
```

Exercise #32 – Collect Coins/Items

- First, create coins:
- Choose suitable coin sprites and make them varied – gold, silver and bronze coins. Each will have a different value when the player picks them up
- Add a Circle Collider 2D to each coin
- Check the “Is Trigger” property
- Create a new script and name it CoinPickup

```
void OnTriggerEnter2D (Collider2D other)
{
    if (other.name == "Sonic GameObject")

        FindObjectOfType<Controller>().totalcoin += coin_value;
        Destroy(this.gameObject);

}
```

Exercise #32 – Collect Coins/Items (Cont.)

- The script executes when the player's collider collides with the coin's collider due to the checked IsTrigger property

- The Destroy function makes the coin object disappear off the screen, giving the illusion that the player has picked it up

- The public variable also allows to set the value of the coin . Set the values for the coins as follows:
  - coinBronze: Set the "Coin Value" property to 1.
  - coinSilver: Set the "Coin Value" property to 5.
  - coinGold: Set the "Coin Value" property to 10.

Exercise #32 – Collect Coins/Items (Cont.)

- You have the player collecting coins, but the total number of coins are not stored yet
- Create a new variable and a function to store that value in PlayerStats. Something like this:

```
public class PlayerStats : MonoBehaviour
{
  public int health = 6;
  public int coinsCollected = 0;

  public void CollectCoin(int coinValue)
  {
    this.coinsCollected = this.coinsCollected + coinValue;
  }
}
```

Exercise #33 – Game Over Sound

- You will need to add sound effect when your game is over, to do so:
  - Go to the “PlayerStats” script and add one AudioClip variable, to hold one single sound that will be on Game Over.

```
public AudioClip GameOverSound;
```

Then, inside the TakeDamage() function, where the we check on the health and lives of our player, call the PlaySingle() function and pass to it your AudioClip variable.

Also, we need to stop the BackGround music playing since the game is over.

```
if (this.lives > 0f && this.health == 0f)
{
    FindObjectOfType<LevelManager>().RespawnPlayer();
    this.health = 6;
    this.lives--;
}
else if (this.lives == 0 && this.health == 0)
{
    Debug.Log("Gameover");      // ADD Gameover splash screen
    AudioManager.instance.PlaySingle(GameOverSound);
    AudioManager.instance.musicSource.Stop();
    Destroy(this.gameObject);
}
```

Useful References

- **Sound Effects - Unity 2D Platformer Tutorial - Part 10.** URL retrieved from: https://www.youtube.com/watch?v=0QWXZDcJEIw
- **Unity 2D Game Development 18 : Audio Clips.** URL retrieved from: https://www.youtube.com/watch?v=FezUGIk-4C4
- **Creating 2D Games in Unity 4.5 #32 - Sounds.** URL retrieved from: https://www.youtube.com/watch?v=pBxG57bjDZs
- **Audio and Sound Manager.** URL retrieved from: https://unity3d.com/learn/tutorials/projects/2d-roguelike/audio
- **Pickups & Points - Unity 2D Platformer Tutorial - Part 6.** URL Retrieved from: https://www.youtube.com/watch?v=J4dkxdbe8FI
- **How To Add A Health Bar & Health Pickups - Unity 2D Platformer Tutorial - Part 26.** URL Retrieved from: https://www.youtube.com/watch?v=-a5JIZih0TM