COMP 4478 - Game Programming

Project 2: Platformer Game in UNITY

Concept Document

Group 5:

Kshitij Anilbhai Patel, Matthew Richard, Timothy Shaw,

Tusshar Singh, Vraj Tejaskumar Modi, Yingzhuo Sun

December 1st, 2023

Contents

Introduction

Welcome to Coin Quest, a vibrant and challenging 2D platformer that takes you on an exciting journey with a lively frog, Leafy! Your mission is to guide this spirited character through a series of obstacles to reach the elusive white flag and conquer each level. The game unfolds in a whimsical world filled with various obstacles that will test your agility and timing skills. Leafy, with its boundless energy, can jump around the map and off walls. As you navigate through the levels, you'll encounter spikes, bombs, and moving blocks adorned with dangerous spikes that pose a constant threat. Timing and precision are key as you leap through these hazards. Your ultimate goal is to guide Leafy safely through each level, avoiding all obstacles, while collecting coins along the way and reaching the checkered flag at the end. The levels become progressively more challenging, requiring quick thinking and precise jumps to overcome new and inventive obstacles. Unlocking each level brings you one step closer to becoming the ultimate Coin Quest champion. Coin Quest offers a visually engaging experience with vibrant colors, playful animations, and a lively soundtrack to complement the energetic gameplay. Are you ready to embark on this exhilarating adventure and guide Leafy to victory? Leap into action and conquer the Coin Quest!

The game we have made for Project 2 in COMP 4478 - Game Programming, is ‘Coin Quest’. Coin Quest is a two-dimensional coin collecting platformer made in Unity. Players control a frog, Leafy, who has the ability to run, jump, slide on walls, and jump off of walls. The player's goal is to collect coins and complete each level they face. To progress throughout the game, players must collect all the coins in each level. Players must do this all while navigating through various obstacles, avoiding enemies, and dodging traps. If a player manages to complete all levels, they have beaten Coin Quest.

Coin Quest was created with one purpose; to create a fun and challenging game for all age groups. Through interactive game elements and increasingly difficult levels, any player of the game will be stimulated by the obstacles at hand. Although the game gets progressively more difficult, the base of the game is a simple platformer. This allows even the youngest player to be able to understand the rules and be immersed in our game.

Following four weeks of lectures on Unity, our group has been able to compile this game. Coin Quest has provided each and every member of our group with an excellent opportunity to apply the theoretical concepts learned in class in the most creatively expressive way possible.

Assets

Sprites

In Coin Quest, we use a combination of custom created sprites and sprites from a public asset package.

Our custom sprites were made using ‘Piskel’, the online sprite editor, which was utilized to create the sprites. Figure 1 shows our coin sprite, which is the object that players are trying to collect throughout the game. The bomb, an obstacle players must avoid, is shown in Figure 2. Lastly, a custom pixel style button was made for the buttons on the menus (Figure 3).

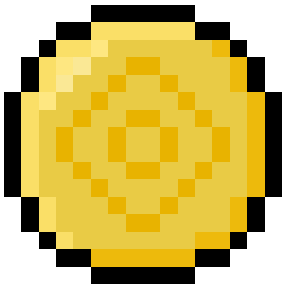

Figure 1. Coin Sprite.

Figure 2. Bomb Sprite.

Figure 3. Button Sprite.

Coin Quest heavily relies on an external asset package. The asset package we chose is called ‘Pixel Adventure 1’, which was found for free on the Unity Asset Store (https://assetstore.unity.com/packages/2d/characters/pixel-adventure-1-155360). Most importantly, this package includes the sprite for the player (Figure 4). As previously mentioned, our game includes a number of obstacles. A majority of these obstacles, aside from the custom bomb sprite, are found in the asset package (Figures 7, 8). Additional sprites used include the flag marking the end of each level, a trophy on the winner screen, a box that the player can push, and a moving platform (Figures 5, 6, 9, 10). In addition to the asset package including the sprites, most sprites also included animations, which we used throughout the game.

Figure 4. Player Sprite.

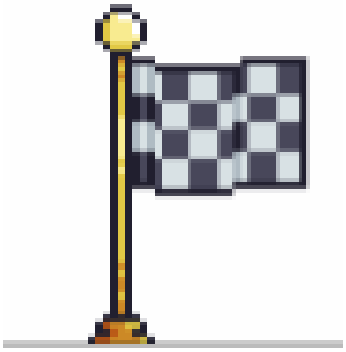

Figure 5. Flag Sprite.

Figure 6. Trophy Sprite.

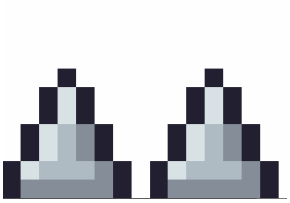

Figure 7. Spikes Sprite.

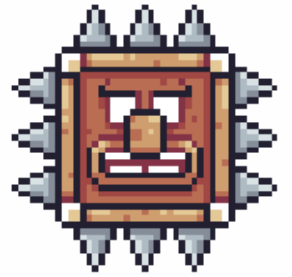

Figure 8. Spiked Head Sprite.

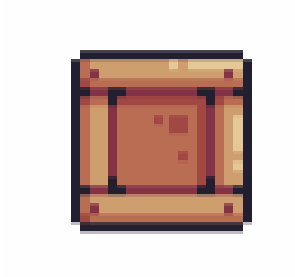

Figure 9. Box Sprite.

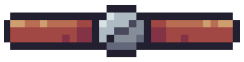

Figure 10. Platform Sprite.

Sound

Upon launching Coin Quest, users will start to hear an ambient tune. This is our background music, found on OpenGameArt (https://opengameart.org/content/green-hills). The music continues throughout each of the different levels and menus within our scene. To ensure that continuity is maintained, we ensured that the audio file does not restart after changing levels.

A number of sound effects supplement different queues in our game. Different sound effects are played when the player collects a coin, and when the player dies. Using jsfxr (https://sfxr.me/), we customized each sound to perfectly match the actions in our game.

Animations

In order to make our game lively and dynamic, certain game objects have set animations. All of the animations in our game are from the Pixel Adventure 1 asset package. Each object that has animations has an associated animation controller to determine when each animation is played. Figure 11 shows the most complex animator in the game, used by the player. This animator controls the many different states of the player, idle, running, jumping, falling, wall sliding, and dying. The coin plays a disappearing animation on collision, controlled in Figure 12. In the main menu and winner scene, a small graphic plays a continuous animation to add a simple, yet effective, visual on the scene; these animations have the simplest animators (Figures 13, 14).

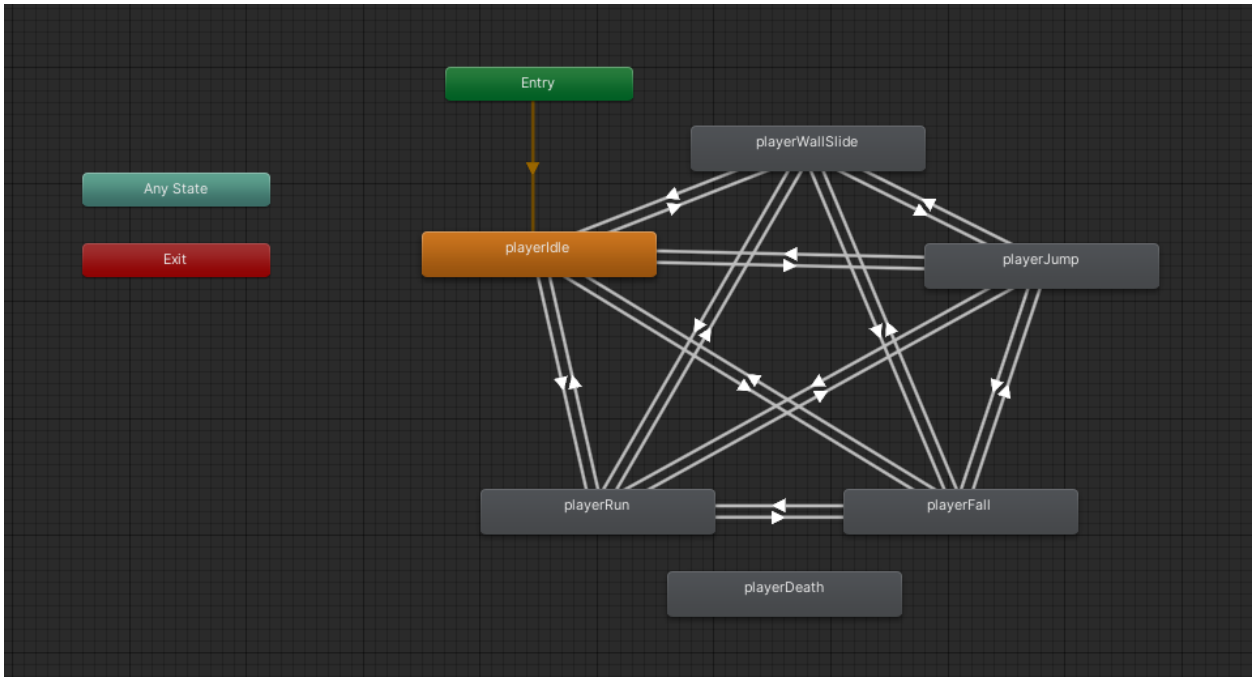

Figure 11. Player Animator.

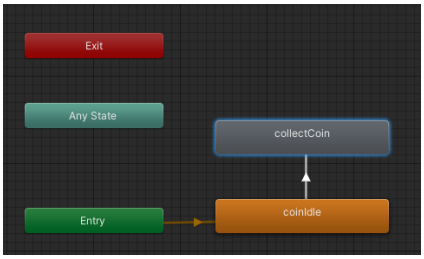

Figure 12. Coin Animator.

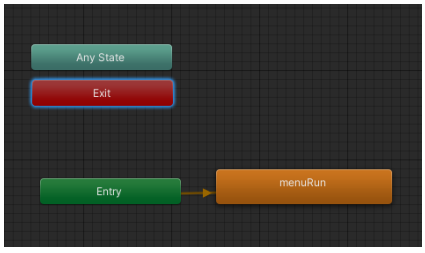

Figure 13. Menu Animator.

Figure 14. Trophy Animator.

Game Rules

Rules

Players must use A-D for horizontal movement of the character, and SpaceBar for jumping.
Players must collect all the coins in each level, without this, they cannot move forward.
Players must complete all five levels, collecting thirty coins, to beat the game.

Setup

When a user first launches Coin Quest, our main menu appears before them. On this menu, there are two buttons. The first button, labeled ‘Play’, starts the game. The second button is labeled ‘Settings’, in which users can customize the game's settings to their preference before they start the game. After pressing play, users are entered into the world of Coin Quest.

If a player is unable to beat the game, and reaches the game over screen, they are prompted to press a button pointing to the main menu. If a player manages to beat the game, they are brought to the winner screen, including a similar button to bring them back to the main menu.

Victory Conditions

Players are attempting to reach the end of each level by navigating through the map. At the end of each level is a flag, marking the end of the level. Players must reach this flag to move on to the next level. However, each level in Coin Quest has a unique number of coins. If the player does not collect all the coins in the level, they are unable to go to the next level.

In Coin Quest, to achieve victory, the player must complete all five levels, collecting a total of thirty coins. Once done, players are met with a screen declaring them as a winner.

Sequence of Gameplay

When a user first launches Coin Quest, our main menu appears before them (Figure 15). On this menu, there are two buttons. The first button, labeled ‘Play’, starts the game. The second button

is labeled ‘Settings’, in which users can customize the game's settings to their preference before they start the game. After pressing play, users are entered into the world of Coin Quest.

Figure 15. Main Menu.

Players are attempting to reach the end of each level by navigating through the map. Figures 16, 17, 18, 19, and 20 display each level. At the end of each level is a flag, marking the end of the level. Players must reach this flag to move on to the next level. However, each level in Coin Quest has a unique number of coins. If the player does not collect all the coins in the level, they are unable to go to the next level. In addition, the levels increase in difficulty through the addition of various enemies and game objects. Level one includes spikes and a moving enemy with spikes that the player must dodge. Level two includes the same elements, however, the coins are placed in locations that require higher level mechanics to reach. In level three, we see the addition of a box object. Players can push this box to move it, and jump on it. Moving the box and jumping on it is required to reach coins placed high in the sky. Starting in the fourth level, the map is much bigger. Players need to spend more time navigating through the obstacles to reach the end. Lastly, level five introduces a moving platform. This platform allows players to move throughout the map, without facing any obstacles on the ground. If players manage to reach the end of this long level, they have beaten the game.

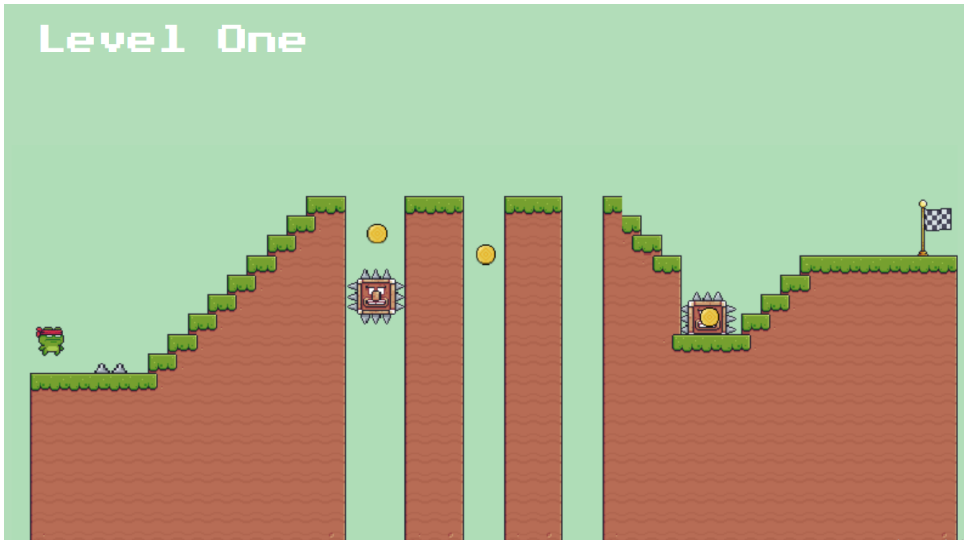

Figure 16. Level One.

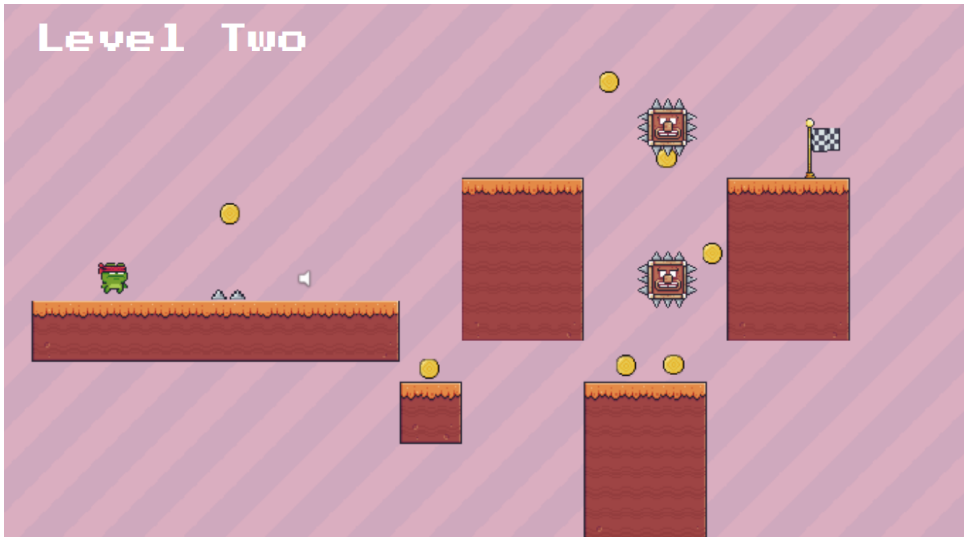

Figure 17. Level Two.

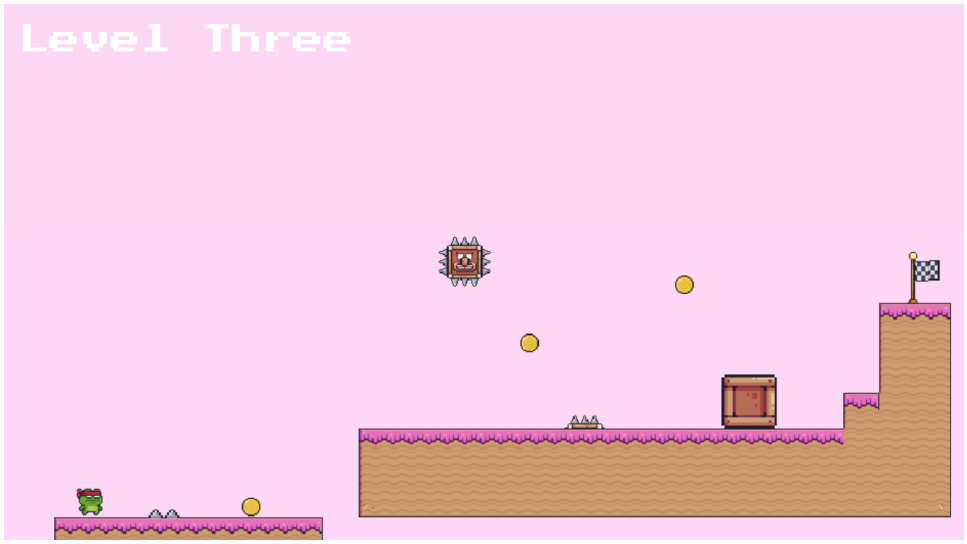

Figure 18. Level Three.

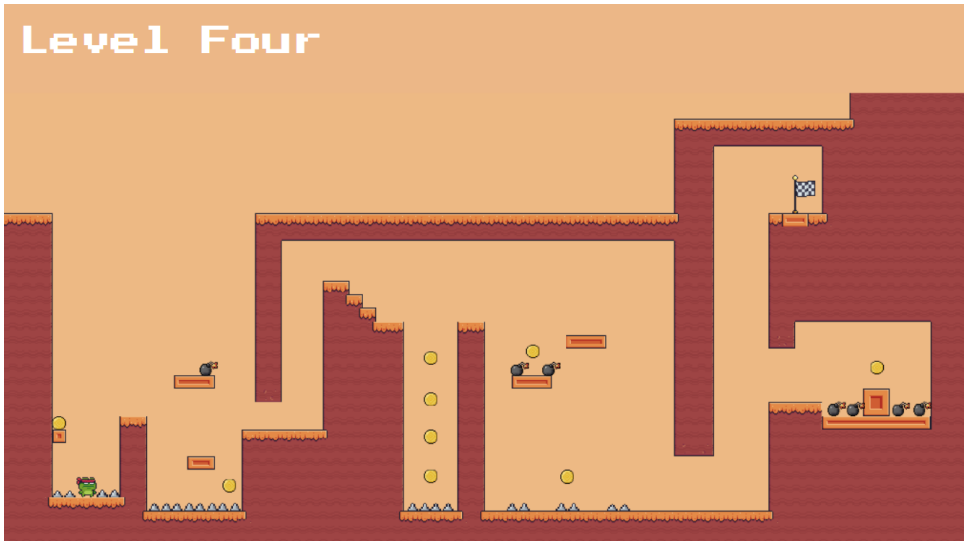

Figure 19. Level Four.

Figure 20. Level Five.

In Coin Quest, to achieve victory, the player must complete all five levels, collecting a total of thirty coins. Once done, players are met with a screen declaring them as a winner (Figure 21).

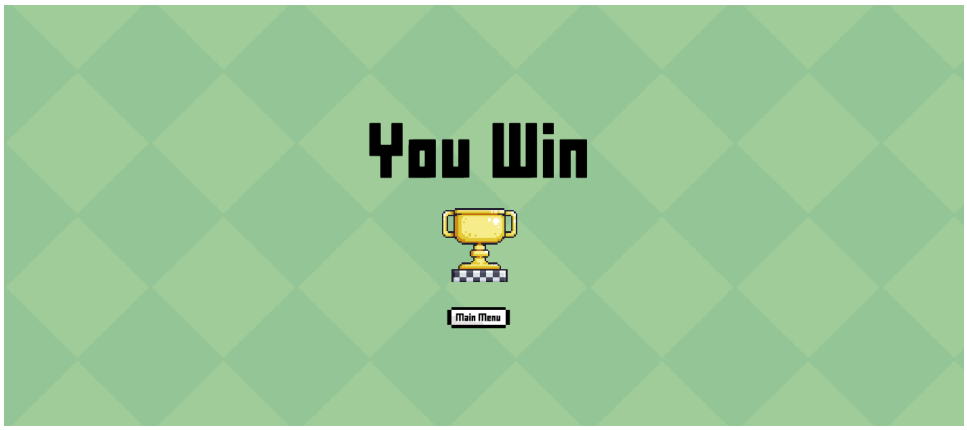

Figure 21. Winner Screen.

If a player is unable to beat the game, and reaches the game over screen, they are prompted to press a button pointing to the main menu (Figure 22).

Figure 22. Game Over Screen.

Scripts

playerMovement.cs

This Unity script controls a player character's movement, allowing it to run, jump, and slide on walls. It uses animations based on the player's actions (idle, running, jumping, falling, and wall sliding). The script also handles player death animation triggered by an external event.Additionally, it incorporates a specific condition for wall sliding and jumping off walls while maintaining an organized structure for player animations.

damagingObject.cs

This Unity script handles interactions with damaging objects. Upon detecting a collision with a game object tagged as "Player," it resets the coin value, triggers a player death animation, plays a sound effect, and then initiates a one-second delay before loading the "GameOver" scene.

DeathTrigger.cs

This Unity script represents a death trigger zone. When the player collides with an object tagged as "Player" within this zone, it reloads the current scene using Application.LoadLevel.

SoundManager.cs

This Unity script manages game sounds, loading "coin" and "hitSound" audio clips on startup. It provides a static method PlaySound(string clip) to play specific sound clips like "coin" or "hit" through the attached AudioSource files.

backgroundMusic.cs

This Unity script implements a singleton pattern for background music, ensuring only one instance exists. It destroys any duplicate instances and uses DontDestroyOnLoad to persist the background music across scene changes, maintaining continuous playback.

EnemyMovement.cs

This Unity script enables an enemy to move back and forth between two points (pointA and pointB) at a defined speed. The script uses Vector3.MoveTowards for smooth movement, and when the enemy reaches a point, it switches direction. The current target is initially set to pointB in the Start method.

PushBox.cs

This Unity script enables a GameObject with the "PushBox" script to push other objects tagged as "Box" when colliding. It calculates the push direction based on the collision positions, normalizes it, and applies a specified push power to the rigidbody of the collided box, influencing its velocity.

MainMenu.cs

This Unity script defines a main menu with two functions: PlayGame loads the next scene in the build order, and QuitGame logs a message and quits the application when called.

NextLevel.cs

This Unity script defines a main menu with two functions: PlayGame loads the next scene in the build order, and QuitGame logs a message and quits the application when called.

menuButton

This Unity script defines a main menu with two functions: PlayGame loads the next scene in the build order, and QuitGame logs a message and quits the application when called.

Third Party Resource Links

‘FFF Forward’ Font:

https://www.1001fonts.com/fff-forward-font.html

‘Pixel Adventure 1’ Asset Package:

https://assetstore.unity.com/packages/2d/characters/pixel-adventure-1-155360

Background Music:

https://opengameart.org/content/green-hills

Audio Maker:

https://sfxr.me/

Sprite Maker:

https://www.piskelapp.com/

Appendix

backgroundMusic.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class backgroundMusic : MonoBehaviour
{
    private static backgroundMusic instance = null;
    public static backgroundMusic Instance
    {
        get { return instance; }
    }
    void Awake()
    {
        AudioListener.volume = PlayerPrefs.GetFloat("Volume", 0.5f);

        //singleton
        if (instance != null && instance != this)
        {
            Destroy(this.gameObject);
            return;
        }
        else
        {
            instance = this;
        }

        //key playing when loading new scenes
        DontDestroyOnLoad(this.gameObject);
    }
}
```

Coins.cs

```
using UnityEngine;
using System.Collections;

public class Coins : MonoBehaviour
{

    //create variables
    [SerializeField] private Animator anim;
    private bool isCollected = false;

    //coroutine
    IEnumerator coinAnimationWait()
    {
        //delay to play coin animation
        yield return new WaitForSeconds(0.3f);

        //remove object from scene
        Destroy(gameObject);

        //increase score
        CoinScript.coinValue += 1;
    }

    void OnTriggerEnter2D(Collider2D other)
    {

        //if the coin has not already been collected
```

```
        // this condition ensures a coin can't be picked up again during its animation or before removal
        if (isCollected || !other.CompareTag("Player"))
        {
            return;
        }

        isCollected = true;

        //trigger coin animation
        if (anim != null)
        {
            anim.SetBool("isCollected", true);
        }

        //play sound
        SoundManagerScript.PlaySound("coin");

        //start coroutine
        StartCoroutine(coinAnimationWait());

    }
}
```

coinScript.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class CoinScript : MonoBehaviour
{

    //initialize variables
    public static int coinValue = 0;
    Text coins;

    //set text to white
    void Start()
    {
        coins = GetComponent<Text>();
        coins.color = Color.white;
    }

    // Update is called once per frame
    void Update()
    {
        //display coin score
        coins.text = "Coins: " + coinValue;
    }

}
```

damagingObject.cs

```
using UnityEngine;
using System.Collections;
using UnityEngine.SceneManagement;

public class damagingObjects : MonoBehaviour
{


    //coroutine
```

```
    IEnumerator GameOverLevelWait()
    {

        //delay to allow the death animation to play
        yield return new WaitForSeconds(1f);

        //load game over scene
        SceneManager.LoadScene("GameOver");
    }

    private void OnTriggerEnter2D(Collider2D other)
    {

        //when player collides
        if (other.gameObject.CompareTag("Player"))
        {

            //reset score
            CoinScript.coinValue =0;

            //make reference to the playerMovement script
            playerMovement playerController = other.gameObject.GetComponent<playerMovement>();

            //if script exists
            if (playerController != null)
            {
                //trigger death function in movement script
                playerController.playerDying();
                SoundManagerScript.PlaySound("hit");

            }

            //start coroutine
            StartCoroutine(GameOverLevelWait());
        }
    }

}
```

DeathTrigger.cs

```
using UnityEngine;
using System.Collections;
using UnityEngine.SceneManagement;

public class DeathTrigger : MonoBehaviour
{


    void OnTriggerEnter2D(Collider2D other)
    {
        //when player collides with trigger, play the game over scene
        if (other.gameObject.CompareTag("Player"))
            SceneManager.LoadScene("GameOver");
    }
}
```

EnemyMovement.cs

```
using UnityEngine;

public class EnemyMovement : MonoBehaviour
{
```

```
    public Transform pointA;
    public Transform pointB;
    public float speed = 2.0f;
    private float threshold = 0.1f;

    private Transform currentTarget;

    void Start()
    {
        currentTarget = pointB;
    }

    void Update()
    {
        MoveBetweenPoints();
    }

    //move between set points
    private void MoveBetweenPoints()
    {
        transform.position = Vector3.MoveTowards(transform.position, currentTarget.position, speed * Time.deltaTime);

        if (Vector3.Distance(transform.position, currentTarget.position) < threshold)
        {
            currentTarget = currentTarget == pointA ? pointB : pointA;
        }
    }
}
```

MainMenu.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class MainMenu : MonoBehaviour
{

    public void PlayGame()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1 );
    }

    public void QuitGame()
    {
        Debug.Log("Quit!");
        Application.Quit();
    }
}
```

MainMenuSettings.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MainMenuSettings : MonoBehaviour
{
    [SerializeField]
    public GameObject mainMenu;
    [SerializeField]
    public GameObject settingsMenu;
    public void Settings()
```

```
    {
        mainMenu.SetActive(false);
        settingsMenu.SetActive(true);
    }
    public void BackToMenu()
    {
        settingsMenu.SetActive(false);
        mainMenu.SetActive(true);
    }

    // Start is called before the first frame update
    void Start()
    {

    }

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

    }
}
```

menuButton.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class menuButton : MonoBehaviour
{

    //function to load main menu scene
   public void loadMenu()
    {
        SceneManager.LoadScene("MainMenu");
    }
}
```

nextLevel.cs

```
using UnityEngine;
using System.Collections;
using UnityEngine.SceneManagement;
using System.Collections.Generic;

public class NextLevel : MonoBehaviour
{

    Scene scene;
    string sceneName;
    AsyncOperation asyncLoad;

    // Keeps track of how many coins you need for each level
    Dictionary<string, int> NeededCoinAmounts = new Dictionary<string, int>()
    {
        { "Tusshar_Scene", 3 },
        { "TimScene", 9 },
        { "Yingzhuo", 12 },
        { "MatthewScene", 21 },
        { "VrajScene", 29 }

    };
```

```
// Use this for initialization
void Start()
{
    scene = SceneManager.GetActiveScene();
    sceneName = scene.name;

    // Set flag transparent
    gameObject.GetComponent<SpriteRenderer>().color = new Color(1f, 1f, 1f, 0.5f);
}

// Update is called once per frame
void Update()
{
    if (CoinScript.coinValue >= NeededCoinAmounts[sceneName])
    {
        // Set flag fully visible
        gameObject.GetComponent<SpriteRenderer>().color = new Color(1f, 1f, 1f, 1f);
    }
}

//coroutine
IEnumerator levelChangeWait()
{

    //switch for changing levels
    switch (sceneName) {
        case "Tusshar_Scene":
            if (CoinScript.coinValue == NeededCoinAmounts[sceneName])
                asyncLoad = SceneManager.LoadSceneAsync("TimScene");
                break;
        case "TimScene":
            if (CoinScript.coinValue == NeededCoinAmounts[sceneName])
                asyncLoad = SceneManager.LoadSceneAsync("Yingzhuo");
                break;
        case "Yingzhuo":
            if (CoinScript.coinValue == NeededCoinAmounts[sceneName])
                asyncLoad = SceneManager.LoadSceneAsync("MatthewScene");
                break;
        case "MatthewScene":
            if (CoinScript.coinValue == NeededCoinAmounts[sceneName])
                asyncLoad = SceneManager.LoadSceneAsync("VrajScene");
                break;
        case "VrajScene":
            if (CoinScript.coinValue == NeededCoinAmounts[sceneName])
                asyncLoad = SceneManager.LoadSceneAsync("WinScene");
                break;
        default:
            asyncLoad = SceneManager.LoadSceneAsync("GameOver");
            break;
    }

    if (asyncLoad != null)
    {
        // Wait until the asynchronous scene fully loads
        while (!asyncLoad.isDone)
        {
            yield return null;
        }
    }


}


void OnTriggerEnter2D(Collider2D other)
{
```

```
        //if player collides, play next scene
        if (other.gameObject.CompareTag("Player"))
        {

            //start coroutine
            StartCoroutine(levelChangeWait());

        }

    }
}
```

Parallax.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Parallax : MonoBehaviour
{
    public Transform cam;
    public float relativeMovementSpeed = 0.3f;

    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        // one-line parallax effect
        transform.position = new Vector3(cam.position.x * relativeMovementSpeed - 10, cam.position.y *
relativeMovementSpeed + 5, transform.position.z);
    }
}
```

PauseMenu.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.PlayerLoop;
using UnityEngine.SceneManagement;

public class PauseMenu : MonoBehaviour
{

    [SerializeField]
    public GameObject pauseMenu;
    [SerializeField]
    public GameObject settingsMenu;

    public bool is_paused = false;
    public void Pause()
    {
        // freeze all movement
        Time.timeScale = 0;

        // show pause screen
        pauseMenu.SetActive(true);
        is_paused = true;
```

```
    }

    public void UnPause()
    {
        // unfreeze movement
        Time.timeScale = 1.0f;

        // hide all menus
        pauseMenu.SetActive(false);
        settingsMenu.SetActive(false);
        is_paused = false;
    }

    public void LoadMenu()
    {
        SceneManager.LoadScene("MainMenu");
    }

    public void ExitGame()
    {
        Application.Quit();
    }

    // Shows the settings screen
    public void Settings()
    {
        pauseMenu.SetActive(false);
        settingsMenu.SetActive(true);
    }

    // goes from settings to pause menu
    public void BackToPause()
    {
        settingsMenu.SetActive(false);
        pauseMenu.SetActive(true);
    }

    public void Start()
    {
        DontDestroyOnLoad(this.gameObject);
        AudioListener.volume = PlayerPrefs.GetFloat("Volume", 0.5f);
    }

    public void Update()
    {
        // pause/unpause when escape is pressed
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            if (is_paused)
            {
                UnPause();
            }
            else
            {
                Pause();
            }
        }

        string sceneName = SceneManager.GetActiveScene().name;
        if (sceneName == "MainMenu" || sceneName == "GameOver")
        {
            // unpause and destroy menu if in main menu or game over screen
            UnPause();
            Destroy(gameObject);
        }
    }
}
```

playButton.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class playButton : MonoBehaviour
{

    //From main menu, load first scene
    public void loadGame()
    {
        SceneManager.LoadScene("Tusshar_Scene");
    }
}
```

playerMovement.cs

```
using System.Collections;
using System.Collections.Generic;
using System.Diagnostics;
using UnityEngine;

public class playerMovement : MonoBehaviour
{

    //create variables
    private Rigidbody2D rb;
    private Animator anim;
    private SpriteRenderer sr;
    private CapsuleCollider2D capsule;
    private PauseMenu pauseMenu;
    private float dirX;
    private float speed = 7f;
    private float jumpSpeed = 30f;
    private bool isDying = false;
    animationState state;
    [SerializeField] private LayerMask jumpableGround;
    private enum animationState { idle, running, jumping, falling, wallSliding }
    private float wallSlidingSpeed = 2f;
    [SerializeField] private Transform wallcheck;
    [SerializeField] LayerMask wallLayer;

    //start method
    void Start()
    {
        rb = GetComponent<Rigidbody2D>();
        anim = GetComponent<Animator>();
        sr = GetComponent<SpriteRenderer>();
        capsule = GetComponent<CapsuleCollider2D>();
        pauseMenu = GameObject.Find("PauseMenuCanvas").GetComponent<PauseMenu>();
    }

    //update method
    void Update()
    {
        if (!isDying && !pauseMenu.is_paused)
        {
            UpdateMovement();
            UpdateAnimations();
        }

    }
```

```
    //update movement method for player direciton and jumping
    private void UpdateMovement()
    {

        //change player velocity based on direction
        dirX = Input.GetAxisRaw("Horizontal");
        rb.velocity = new Vector2(dirX * speed, rb.velocity.y);

        //if the player is on the ground and space is pressed, jump
        if (Input.GetKeyDown("space") && IsGrounded())
        {
            //play sound
            SoundManagerScript.PlaySound("jump");
            rb.velocity = new Vector2(rb.velocity.x, jumpSpeed);
        }

    }

    //method to play the different animations
    private void UpdateAnimations()
    {

        //if the character is moving to the right, play the running animation
        if (dirX > 0f)
        {
            state = animationState.running;
            sr.flipX = false;
            //flip the wallcheck object to the direction the player is facing
            float wallCheckX = Mathf.Abs(wallcheck.localPosition.x);
            wallcheck.localPosition = new Vector3(sr.flipX ? -wallCheckX : wallCheckX, wallcheck.localPosition.y,
wallcheck.localPosition.z);
        }
        //if the character is moving to the left, flip the player then play the running animation
        else if (dirX < 0f)
        {
            state = animationState.running;
            sr.flipX = true;
            //flip the wallcheck object to the direction the player is facing
            float wallCheckX = Mathf.Abs(wallcheck.localPosition.x);
            wallcheck.localPosition = new Vector3(sr.flipX ? -wallCheckX : wallCheckX, wallcheck.localPosition.y,
wallcheck.localPosition.z);
        }
        //when the player is not moving, play the idle animation
        else
        {
            state = animationState.idle;
        }

        //play the jumping animation when the player jumps
        if (rb.velocity.y > 0.05f)
        {
            state = animationState.jumping;

        }
        //when the player is falling, play the fall animation
        else if (rb.velocity.y < -0.05f)
        {
            state = animationState.falling;
        }

        //if the player is on the wall and facing the wall, play the wall slide animation
        if (IsWalled() && !IsGrounded() && Mathf.Sign(rb.velocity.x) == Mathf.Sign(dirX))
        {

            //slide player down the wall
            rb.velocity = new Vector2(rb.velocity.x, Mathf.Clamp(rb.velocity.y, -wallSlidingSpeed, float.MaxValue));
```

```
            state = animationState.wallSliding;

            //if player is wall sliding and pressed space, jump from the wall
            if (Input.GetKeyDown("space"))
            {
                rb.velocity = new Vector2(-Mathf.Sign(dirX) * speed, jumpSpeed);

                state = animationState.jumping;
                //play sound
                SoundManagerScript.PlaySound("jump");
            }
        }
        else
        {

        }

        //play the animation based on the states set in the if statements above
        anim.SetInteger("state", (int)state);
    }

    //check if the player is on the ground
    private bool IsGrounded()
    {
        return Physics2D.BoxCast(capsule.bounds.center, capsule.bounds.size, 0f, Vector2.down, .1f, jumpableGround);
    }

    //check if the player in on the wall
    private bool IsWalled()
    {
        return Physics2D.OverlapCircle(wallcheck.position, 0.2f, wallLayer);
    }

    //trigger death animation
    public void playerDying()
    {
        isDying = true;
        anim.Play("playerDeath");
    }


}
```

PushBox.cs

```
using UnityEngine;

public class PushBox : MonoBehaviour
{
    public float pushPower = 2.0f;

    void OnCollisionStay2D(Collision2D collision)
    {

        //if collision object is the box
        if (collision.gameObject.CompareTag("Box"))
        {
            Rigidbody2D boxRigidbody = collision.gameObject.GetComponent<Rigidbody2D>();

            if (boxRigidbody != null)
            {
                //when player colliders with box, move box in direction of player movement
                Vector2 pushDirection = new Vector2(collision.transform.position.x - transform.position.x,
0).normalized;
                boxRigidbody.velocity = pushDirection * pushPower;
```

```
            }
        }
    }
}
```

scoreReset.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class scoreReset : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {
        //when game restarts, reset score
        CoinScript.coinValue = 0;
    }

}
```

ScoreScript.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class ScoreScript : MonoBehaviour
{

    //create variables
    public static int scoreValue = 0;
    Text score;

    // Start is called before the first frame update
    void Start()
    {
        score = GetComponent<Text>();
    }

    // Update is called once per frame
    void Update()
    {
        //set score text
        score.text = "Score: " + scoreValue;
    }
}
```

SettingsMenu.cs

```
using System.Collections;
using System.Collections.Generic;
using TMPro;
using UnityEngine;
using UnityEngine.Audio;
using UnityEngine.UI;

public class SettingsMenu : MonoBehaviour
{
    [SerializeField] private Slider slider;
```

```
    [SerializeField] private TextMeshProUGUI fullscreenText;
    [SerializeField] private GameObject dropdown;

    public void OnChangeSlider()
    {
        float value = slider.value;

        // Change volume based on slider
        AudioListener.volume = value;
        PlayerPrefs.SetFloat("Volume", value);
        PlayerPrefs.Save();
    }

    public void SetFullscreen()
    {
        Screen.fullScreen = !Screen.fullScreen;
        fullscreenText.text = (Screen.fullScreen ? "Enable" : "Disable") + " Fullscreen";
    }

    public void SetResolution()
    {
        // 0 is 1080p, 1 is 720p
        int resIndex = dropdown.GetComponent<TMP_Dropdown>().value;

        // Set resolution
        switch (resIndex)
        {
            case 0:
                Screen.SetResolution(1920, 1080, Screen.fullScreen);
                break;
            case 1:
                Screen.SetResolution(1280, 720, Screen.fullScreen);
                break;
            default:
                break;
        }

        // Remember player settings
        PlayerPrefs.SetInt("Resolution", resIndex);
        PlayerPrefs.Save();
    }

    void Start()
    {
        // Set remembered volume
        float startVal = PlayerPrefs.GetFloat("Volume", 0.5f);
        slider.value = startVal;
        AudioListener.volume = startVal;

        fullscreenText.text = (!Screen.fullScreen ? "Enable" : "Disable") + " Fullscreen";

        // Set remembered resolution - default to 1080p
        dropdown.GetComponent<TMP_Dropdown>().value = PlayerPrefs.GetInt("Resolution", 0);
    }

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

    }
}
```

SoundManagerScript.cs

```
using System.Collections;
using System.Collections.Generic;
```

```
using UnityEngine;

public class SoundManagerScript : MonoBehaviour
{
    //create variables
    public static AudioClip coinSound, hitSound, jumpSound;
    static AudioSource audioSource;

    // Start is called before the first frame update
    void Start()
    {

        //load sound clips
        coinSound = Resources.Load<AudioClip>("coin");
        hitSound = Resources.Load<AudioClip>("hitSound");
        jumpSound = Resources.Load<AudioClip>("jump");

        //get source
        audioSource = GetComponent<AudioSource>();
    }

    //play sound on trigger
    public static void PlaySound(string clip)
    {
        //switch for different sounds
        switch (clip)
        {

            case "coin":
                audioSource.PlayOneShot(coinSound);
                break;

            case "hit":
                audioSource.PlayOneShot(hitSound);
                break;

            case "jump":
                audioSource.PlayOneShot(jumpSound);
                break;

        }
    }
}
```