Colin Hite,

The following pages are copies of my scripts contained within my videogame “CaveGame”. All the assets were created solely by myself Colin Hite including the scripts and visuals. I would please ask that you not copy the scripts contained within. I am very happy with the result and hope you enjoy not only viewing these scripts but also playing the game itself.

A special thanks goes out to my professor, Marty Clayton as well as the unity team that

made this all possible.

Thanks!

```
using UnityEngine;
using System.Collections;

public class BatMove : MonoBehaviour {

    //public GameObject batObj;
    public float batSpeed;
    public Rigidbody2D bat;
    // Checks for player
    public Transform playerCheck;
    public float playerCheckRadius;
    public LayerMask whatIsPlayer;
    private bool playerDetect = false;
    // points value
    public int pointsToAdd;

    public int BatsToSubtract;

    //gives the bat speed
    void Start()
    {
        bat = GetComponent<Rigidbody2D>();
        bat.AddForce(-transform.right * batSpeed);
    }

    //sets player detect bounds
    void FixedUpdate()
    {
        playerDetect = Physics2D.OverlapCircle(playerCheck.position,
          playerCheckRadius, whatIsPlayer);
    }

    void Update()
    {
        //makes bat stay level
        transform.eulerAngles = new Vector3(0, 0, 0);

        //looks for player
        if (playerDetect == true)
        {

            Destroy(gameObject);
            ScoreManager.AddPoints(pointsToAdd);
            BatSpawn.AddBats(BatsToSubtract);
        }

        //Bat Flip
        if (GetComponent<Rigidbody2D>().velocity.x > 0)
            transform.localScale = new Vector3(-0.3f, 0.3f, 0.3f);

        else if (GetComponent<Rigidbody2D>().velocity.x < 0)
            transform.localScale = new Vector3(0.3f, 0.3f, 0.3f);
    }
}
```

```
using UnityEngine;
using System.Collections;

public class BatSpawn : MonoBehaviour {
    //bat limits
    public int maxBats = 4;
    public static int minBats = 0;

    //bat
    public Transform batSpawn;
    public GameObject bat;

    //spawn and duration
    public float duration = 0.5f;
    private float elapsed;
    public int[] batAngleSpawn = {-10, 25, 13, -34};

    void Update()
    {
        //stops bat count problems
        if (minBats < 0)
            minBats = 0;

        elapsed += Time.deltaTime;

        //grabs rotation array and chooses ramdom angle
        int spawnRotationIndex = Random.Range(0, batAngleSpawn.Length);

        //Makes new bats
        if (elapsed >= duration && maxBats > minBats)
        {
            Instantiate(bat, batSpawn.position, batSpawn.rotation);
            minBats += 1;
            elapsed = 0;
            transform.eulerAngles = new Vector3(0, 0, batAngleSpawn
              [spawnRotationIndex]);
            Debug.Log("Made Bat");
        }
    }

    //subtracts bats
    public static void AddBats(int BatsToSubtract)
    {
        minBats -= BatsToSubtract;
    }
}
```

```
using UnityEngine;
using System.Collections;

public class DodgePoints : MonoBehaviour
{
    //Point mod
    public int pointsToAdd;
    public static int hitLog = 1;

    // Pass function to restart multiplier
    public static void Multiplier(int multi = 1)
    {
        hitLog = multi;
    }

    //Adds points when Spike hits back wall and multiplies the score for each one
    void OnCollisionEnter2D (Collision2D col)
    {
        if (col.gameObject.tag == ("spike"))
        {
            Debug.Log("GotPoints");
            ScoreManager.AddPoints(pointsToAdd * hitLog);
            hitLog += 1;
        }
    }
}
```

```
using UnityEngine;
using System.Collections;

public class HeroMove : MonoBehaviour {
    //Player Movement
    public float moveSpeed;
    public float jumpHeight;

    public float moveVelocity;

    //Ground Check
    public Transform groundCheck;
    public float groundCheckRadius;
    public LayerMask whatIsGround;
    private bool grounded;

    //Double Jump
    private bool doubleJumped;

    //Sets grounded state
    void FixedUpdate()
    {
        grounded = Physics2D.OverlapCircle(groundCheck.position, groundCheckRadius,
          whatIsGround);
    }

    void Update()
    {

        //Stops player from sticking to the wall
        moveVelocity = 0f;

        //Move Right
        if (Input.GetKey(KeyCode.D))
        {
            moveVelocity = moveSpeed;
        }

        //Move Left
        if (Input.GetKey(KeyCode.A))
        {
            moveVelocity = -moveSpeed;
        }

        //Move Right
        if (Input.GetKey(KeyCode.RightArrow))
        {
            moveVelocity = moveSpeed;
        }

        //Move Left
        if (Input.GetKey(KeyCode.LeftArrow))
        {
            moveVelocity = -moveSpeed;
        }

        //Stops Player from sticking to the wall
        GetComponent<Rigidbody2D>().velocity = new Vector2(moveVelocity,
          GetComponent<Rigidbody2D> ().velocity.y);

        // Jump
        if (Input.GetKeyDown(KeyCode.Space) && grounded)
        {
            GetComponent<Rigidbody2D>().velocity = new Vector2
              (GetComponent<Rigidbody2D>().velocity.x, jumpHeight);
            //Debug.Log("I jumped");
        }

        //Player Flip
        if (GetComponent<Rigidbody2D>().velocity.x > 0)
            transform.localScale = new Vector3(1f, 1f, 1f);

        else if (GetComponent<Rigidbody2D>().velocity.x < 0)
            transform.localScale = new Vector3(-1f, 1f, 1f);

        //Double Jump
        if (grounded)
            doubleJumped = false;

        if (Input.GetKeyDown(KeyCode.Space) && !doubleJumped && !grounded)
        {
            GetComponent<Rigidbody2D>().velocity = new Vector2
              (GetComponent<Rigidbody2D>().velocity.x, jumpHeight);
            doubleJumped = true;
            //Debug.Log("I jumped twice");
        }


    }

}
```

```
using UnityEngine;
using System.Collections;

public class LoadSceneOnClick : MonoBehaviour {

    public GameObject loadingScreen;

    //loads new scene
    public void LoadScene(int level)
    {
        loadingScreen.SetActive(true);
        Application.LoadLevel(level);
    }
}
```

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

public class NewHealth : MonoBehaviour {

    // Health and respawn
    private SpikeSpawn spikeSpawn;
    public float maxHealth = 100;
    public float currentHealth = 0;
    public Respawn respawn;
    public GameObject healthBar;

    //--Damage Flash
    public float flashSpeed = 5f;
    public Color damage = new Color(1f, 0f, 0f, 0.1f);
    public Image damageImage;
    bool damaged = false;

    // Death screen
    public GameObject killScreen;
    // point multiplier
    public int hitLog;

    //sets health to max and starts key functions related to health
    void Start ()
    {
        spikeSpawn = FindObjectOfType<SpikeSpawn>();
        respawn = FindObjectOfType<Respawn> ();
        currentHealth = maxHealth;
        InvokeRepeating("decreseHealth", 1f, 1f);
    }

    //updates image to a red state and back to a clear state
    void Update()
    {
        if (damaged == true)
        {
            damageImage.color = damage;
        }
        else
        {
            damageImage.color = Color.Lerp(damageImage.color, Color.clear, flashSpeed 
              * Time.deltaTime);
        }
        damaged = false;
        //Debug.Log("Flashed Damage Image");
    }

   //Makes the player lose health and upon 0 health lose the game
    void OnCollisionEnter2D (Collision2D col)
    {
        if(col.gameObject.tag == ("spike") || col.gameObject.tag == ("bat"))
        {
            damaged = true;
            currentHealth -= 10;

            //Handles point multiplier loss
            DodgePoints.Multiplier(hitLog = 1);
            Debug.Log("Lost Multiplier");
            Debug.Log("Got Hit took damage");
        }

        if (currentHealth == 0)
        {
            gameObject.SetActive(false);
            spikeSpawn.enabled = false;
            killScreen.SetActive(true);
            respawn.RespawnPlayer();
        }
    }

    // modifies the health bar position
    void decreseHealth()
    {
        float myHealth = currentHealth / maxHealth;
        HealthBar(myHealth);
    }

    void HealthBar(float myHealth)
    {
        healthBar.transform.localScale = new Vector3(myHealth,
          healthBar.transform.localScale.y, healthBar.transform.localScale.z);
    }

}
```

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

public class ScoreManager : MonoBehaviour {

    public static int score;

    Text text;

    //sets score to 0 and grabs the text object
    void Start ()
    {
        text = GetComponent<Text>();

        score = 0;
    }

    //sets the score string to the UI and prevents negative points
    void Update ()
    {
        if (score < 0)
            score = 0;

        text.text = "Score: " + score;
    }

    //point controller
    public static void AddPoints (int pointsToAdd)
    {
        score += pointsToAdd;
    }
    //point reset
    public static void Reset ()
    {
        score = 0;
    }
}
```

```
using UnityEngine;
using System.Collections;

public class SpikeMoveDes : MonoBehaviour {

    //spike variables
    public float spikeLife = 1.0f;
    public int spikeSpeed;
    public Rigidbody2D spike;


    void Start ()
    {
        spike = GetComponent<Rigidbody2D>();
    }

    //sets spike speed and direction and destroys it over time
    void Update ()
    {
        spike.velocity = new Vector2(-spikeSpeed, spike.velocity.y);
        Destroy(gameObject, spikeLife);
    }

    //Sets bounds for what makes the spike disappear
    void OnCollisionEnter2D (Collision2D col)
    {
        if (col.gameObject.tag == ("wall"))
        {
            Destroy(gameObject);
        }
        else if (col.gameObject.tag == "Player")
        {
            Destroy(gameObject);
        }
        else
            Debug.Log("Is Dead");

    }
}
```

```
using UnityEngine;
using System.Collections;

public class SpikeSpawn : MonoBehaviour {

    //spike
    public Transform deathSpikeSpawn;
    public GameObject spike;

    //Spike life span
    public float duration = 0.5f;
    private float elapsed;


    void Update ()
    {
        elapsed += Time.deltaTime;

        //makes new spike
        if (elapsed >= duration)
        {
            Instantiate(spike, deathSpikeSpawn.position, deathSpikeSpawn.rotation);
            elapsed = 0;
            Debug.Log("Made Spike");
        }
    }
}
```

```
using UnityEngine;
using System.Collections;

public class TileParallax : MonoBehaviour {

    //Background
    public float scrollSpeed;
    public float tileSizeZ;

    //Starting point
    private Vector3 startPosition;

    //sets the starting point
    void Start ()
    {
        startPosition = transform.position;
    }

    //Moves the background image tiles
    void FixedUpdate ()
    {
        float newPosition = Mathf.Repeat(Time.time * scrollSpeed, tileSizeZ);
        transform.position = startPosition + Vector3.right * newPosition;
    }
}
```