All Source Code of Game (Memory Defrag)

Here are the list of codes and scripts that been used to develops the game:

Camera Controller Script:

```
public class CameraController : MonoBehaviour {
    public PlayerMovement thePlayer;
    private Vector3 lastPlayerPosition;
    private float distanceToMove;

    // Use this for initialization
    void Start () {
        thePlayer = FindObjectOfType<PlayerMovement>();
        lastPlayerPosition = thePlayer.transform.position;
    }

    // Update is called once per frame
    void Update () {
        distanceToMove = thePlayer.transform.position.x - lastPlayerPosition.x;
        transform.position = new Vector3(transform.position.x + distanceToMove,
 transform.position.y, transform.position.z);
        lastPlayerPosition = thePlayer.transform.position;
```

Coin Generator Script:

```
    public class CoinGenerator : MonoBehaviour {
    public ObjectPooler coinPool;
    public float distanceBetweenCoins;
    public void SpawnCoins (Vector3 startPosition)
    {
```

```
        GameObject coin1 = coinPool.GetPooledObject();
        coin1.transform.position = startPosition;
        coin1.SetActive (true);
}
```

Wait for Player Script:

```
public class WaitforPlayer : MonoBehaviour {
    public Text instructionsText;

    // Use this for initialization
    void Start ()
            instructionsText.text = "PRESS SPACEBAR TO JUMP";
            Time.timeScale = 0;
        gameObject.SetActive (true);
    }

    // Update is called once per frame
    void Update ()
    {
        if (Input.GetKeyDown (KeyCode.Space)) {
            Time.timeScale = 1;
                Destroy (gameObject);
                Destroy (instructionsText);
        }
```

Score Manager Script:

```
public class ScoreManager : MonoBehaviour {
    public Text scoreText;
    public Text hiScoreText;
    public float scoreCount;
    public float hiScoreCount;
    public float pointsPerSecond;

    //reset score, increase score//
    public bool scoreIncreasing;
    public bool shouldDouble;

    // Use this for initialization
    void Start () {
        if(PlayerPrefs.HasKey("HighScore"))
        {
            hiScoreCount = PlayerPrefs.GetFloat ("HighScore");
        }
    }

    // Update is called once per frame
    void Update ()
    {
        if (scoreIncreasing)
        {
            scoreCount += pointsPerSecond * Time.deltaTime;
        }
        if (scoreCount > hiScoreCount)
```

```
        {
            hiScoreCount = scoreCount;
            PlayerPrefs.SetFloat("HighScore", hiScoreCount);
        }
        scoreText.text = "" + Mathf.Round (scoreCount);
        hiScoreText.text = "" + Mathf.Round (hiScoreCount);
        if (scoreCount < -0) {
            scoreCount = 0;
        }
    }
    public void AddScore(int pointsToAdd)
    {
        if (shouldDouble)
        {
            pointsToAdd = pointsToAdd * 2;
        }
        scoreCount += pointsToAdd;
    }
}
```

Status Script:

```
public class Status : MonoBehaviour {
    public void PlayBoy()
    {
        SceneManager.LoadScene("Boy");
    }
    public void PlayGirl()
    {
```

```
        SceneManager.LoadScene("Girl");
    }
    public void ReturnToMenu()
    {
        SceneManager.LoadScene ("Main Menu");
    }
    public void Instruction()
    {
        SceneManager.LoadScene ("Instructions");
    }
    public void Credits()
    {
        SceneManager.LoadScene ("Credits");
    }
    public void BackCredits()
    {
        SceneManager.LoadScene ("Status");
    }
```

Powerup Manager:

```
public class PowerupManager : MonoBehaviour {
    private bool doublePoints;
    private bool safeMode;
    private bool powerupActive;
    private float powerupLengthCounter;
    private ScoreManager theScoreManager;
    private PlatformGenerator thePlatformGenerator;
    private GameManager theGameManager;
```

```
    private float normalPointsPerSecond;
    private float crateRate;
    private PlatformDestroyer[] crateList;

    // Use this for initialization
    void Start () {
        theScoreManager = FindObjectOfType<ScoreManager> ();
        thePlatformGenerator = FindObjectOfType<PlatformGenerator> ();
        theGameManager = FindObjectOfType<GameManager> ();
    }

    // Update is called once per frame
    void Update () {
        if (powerupActive)
        {
            powerupLengthCounter -= Time.deltaTime;

            if (theGameManager.powerupReset)
            {
                powerupLengthCounter = 0;
                theGameManager.powerupReset = false;
            }

            if (doublePoints)
            {
                theScoreManager.pointsPerSecond = normalPointsPerSecond * 1.2
f;
                theScoreManager.shouldDouble = true;
            }
```

```
        if (safeMode)
        {
            thePlatformGenerator.randomCrateSpawnRate = 0f;
        }
        if (powerupLengthCounter <= 0)
        {
            theScoreManager.pointsPerSecond = normalPointsPerSecond;
            theScoreManager.shouldDouble = false;
            thePlatformGenerator.randomCrateSpawnRate = crateRate;
            powerupActive = false;
        }
    }
}
public void ActivatePowerup(bool points, bool safe, float time)
{
    doublePoints = points;
    safeMode = safe;
    powerupLengthCounter = time;
    normalPointsPerSecond = theScoreManager.pointsPerSecond;
    crateRate = thePlatformGenerator.randomCrateSpawnRate;
    if (safeMode) {
        crateList = FindObjectsOfType<PlatformDestroyer> ();
        for (int i = 0; i < crateList.Length; i++) {
            if (crateList [i].gameObject.name.Contains ("Spike")) {
                crateList [i].gameObject.SetActive (false);
            }
        }
        powerupActive = true;
```

```
        }
    }
```

Scene Transition:

```
public class SceneTransition : MonoBehaviour {
    private Animator transitionAnim;
    private void Start()
    {
        transitionAnim = GetComponent<Animator> ();
    }
    public void LoadScene(string sceneName){
        StartCoroutine (Transition (sceneName));
    }
    IEnumerator Transition(string sceneName){
        transitionAnim.SetTrigger ("end");
        yield return new WaitForSeconds (1);
        SceneManager.LoadScene (sceneName);
    }
}
```

Powerups Script:

```
public class Powerups : MonoBehaviour {
    public bool doublePoints;
    public bool safeMode;
    public float powerupLength;
    private PowerupManager thePowerupManager;
    public Sprite[] powerupSprites;
    public GameObject coinParticle;
```

```
    // Use this for initialization
    void Start () {
        thePowerupManager = FindObjectOfType<PowerupManager> ();
    }
    void Awake(){
        int powerupSelector = Random.Range (0, 2);
        switch(powerupSelector)
        {
        case 0: doublePoints = true;
            break;
        case 1: safeMode = true;
            break;
        }
        GetComponent<SpriteRenderer> ().sprite = powerupSprites [powerupSel
ector];
    }
    void OnTriggerEnter2D(Collider2D other)
    {
        if (other.name == "Player")
        {
            thePowerupManager.ActivatePowerup (doublePoints, safeMode, powe
rupLength);
        }
        gameObject.SetActive (false);
        Instantiate (coinParticle, other.transform.position, Quaternion.identity);
    }
}
```

Player Movement:

```
public class PlayerMovement : MonoBehaviour {
    public float moveSpeed;
    private float moveSpeedStore;
    public float speedMultiplier;
    public float speedIncreaseMilestone;
    private float speedIncreaseMilestoneStore;
    public float speedMilestoneCount;
    private float speedMilestoneCountStore;
    public float jumpForce;
    public float jumpTime;
    private float jumpTimeCounter;
    private bool stoppedJumping;
    private bool canDoubleJump;
    private Rigidbody2D myRigidbody;
    public bool grounded;
    public LayerMask whatIsGround;
    public Transform groundCheck;
    public float groundCheckRadius;
    private Animator myAnimator;
    public GameManager theGameManager;
    public AudioSource jumpSound;
    public AudioSource deathSound;
    public float health;
    public float maxhealth;
    Image healthbar;
    public string mainMenuLevel;
    public float burnRate = 1f;
```

```
// Use this for initialization
void Start () {
    healthbar = GameObject.Find ("HealthBar").GetComponent<Image> ();
    myRigidbody = GetComponent<Rigidbody2D>();
    myAnimator = GetComponent<Animator>();
    jumpTimeCounter = jumpTime;
    speedMilestoneCount = speedIncreaseMilestone;
    moveSpeedStore = moveSpeed;
    speedMilestoneCountStore = speedMilestoneCount;
    speedIncreaseMilestoneStore = speedIncreaseMilestone;
    stoppedJumping = true;
}

// Update is called once per frame
void Update () {
    if (health > maxhealth) {
        health = maxhealth;
    }
    health -= burnRate * Time.deltaTime;
    if (burnRate <= 0) {
        healthbar.fillAmount = 0;
        theGameManager.RestartGame ();
    }
    if (health <= 0) {
        deathSound.Play ();
        theGameManager.RestartGame();
        healthbar.fillAmount = 0;
    }
    healthbar.fillAmount = health / maxhealth ;
```

```
        grounded = Physics2D.OverlapCircle (groundCheck.position, groundChe
ckRadius, whatIsGround);
        if (transform.position.x > speedMilestoneCount)
        {
            speedMilestoneCount += speedIncreaseMilestone;
            speedIncreaseMilestone = speedIncreaseMilestone * speedMultiplier;
            moveSpeed = moveSpeed * speedMultiplier;
        }
        myRigidbody.velocity = new Vector2(moveSpeed, myRigidbody.velocity.y
);
        if (Input.GetKeyDown(KeyCode.Space))
        {
            if (grounded)
            {
                myRigidbody.velocity = new Vector2(myRigidbody.velocity.x, jumpF
orce);
                stoppedJumping = false;
                jumpSound.Play ();
            }

            if (!grounded && canDoubleJump)
            {
                myRigidbody.velocity = new Vector2(myRigidbody.velocity.x, jumpF
orce);
                jumpTimeCounter = jumpTime;
                stoppedJumping = false;
                canDoubleJump = false;
                jumpSound.Play ();
            }
```

```
        }
        if ((Input.GetKey (KeyCode.Space) || Input.GetMouseButtonDown (0)) &&
 !stoppedJumping)
        {
            if (jumpTimeCounter > 0)
            {
                myRigidbody.velocity = new Vector2(myRigidbody.velocity.x, jumpF
orce);
                jumpTimeCounter -= Time.deltaTime;
            }
        }
        if(Input.GetKeyUp (KeyCode.Space) || Input.GetMouseButtonUp(0))
        {
            jumpTimeCounter = 0;
            stoppedJumping = true;
        }
        if (grounded)
        {
            jumpTimeCounter = jumpTime;
            canDoubleJump = true;
        }
        myAnimator.SetFloat("Speed", myRigidbody.velocity.x);
        myAnimator.SetBool("Grounded", grounded);
    }
    void OnCollisionEnter2D (Collision2D other)
    {
        if(other.gameObject.tag == "killbox")
        {
            theGameManager.RestartGame();
```

```
            moveSpeed = moveSpeedStore;
            speedMilestoneCount = speedMilestoneCountStore;
            speedIncreaseMilestone = speedIncreaseMilestoneStore;
            deathSound.Play ();
            healthbar.fillAmount = 0;
        }
    }
    public void GetDamage(float dmg)
    {
        health -= dmg;
    }
}
```

Platform Destroyer:

```
public class PlatformDestroyer : MonoBehaviour {
    public GameObject platformDestructionPoint;
    // Use this for initialization
    void Start () {
        platformDestructionPoint = GameObject.Find("PlatformDestructionPoint")
;
    }
    // Update is called once per frame
    void Update () {
        if(transform.position.x < platformDestructionPoint.transform.position.x)
        {
            //Destroy(gameObject);
            gameObject.SetActive(false);
        }
```

```
    }
}
```

Platform Generator:

```
public class PlatformGenerator : MonoBehaviour {
    public GameObject thePlatform;
    public Transform generationPoint;
    public float distanceBetween;
    private float platformWidth;
    public float distanceBetweenMin;
    public float distanceBetweenMax;
    private int platformSelector;
    private float[] platformWidths;
    public ObjectPooler[] theObjectPools;
    private float minHeight;
    public Transform maxHeightPoint;
    private float maxHeight;
    public float maxHeightChange;
    private float heightChange;
    private CoinGenerator theCoinGenerator;
    public float randomCoinSpawnRate;
    public float randomCrateSpawnRate;
    public ObjectPooler cratePool;
    public float powerupHeight;
    public ObjectPooler powerupPool;
    public float powerupSpawnRate;

    // Use this for initialization
```

```
    void Start () {
        platformWidths = new float[theObjectPools.Length];
        for (int i = 0; i < theObjectPools.Length; i++)
        {
            platformWidths [i] = theObjectPools[i].pooledObject.GetComponent<Bo
xCollider2D> ().size.x;
        }
        minHeight = transform.position.y;
        maxHeight = maxHeightPoint.position.y;
        theCoinGenerator = FindObjectOfType<CoinGenerator> ();
    }

    // Update is called once per frame
    void Update () {
        if(transform.position.x < generationPoint.position.x)
        {
            distanceBetween = Random.Range(distanceBetweenMin, distanceBet
ween Max);
            platformSelector = Random.Range (0, theObjectPools.Length);
            heightChange = transform.position.y + Random.Range (maxHeightCha
nge, -maxHeightChange);
            if (heightChange > maxHeight)
            {
                heightChange = maxHeight;
            }   else if (heightChange <minHeight)
            {
                heightChange = minHeight;
            }
            if (Random.Range (0f, 100f) < powerupSpawnRate)
```

```
        {
            GameObject newPowerup = powerupPool.GetPooledObject ();
            newPowerup.transform.position = transform.position + new Vector3
(distanceBetween / 2f, Random.Range(powerupHeight/2f, powerupHeight), 0f)
;
            newPowerup.SetActive (true);
        }
        transform.position = new Vector3(transform.position.x + (platformWidth
s[platformSelector] / 2) + distanceBetween, heightChange, transform.position.
z);
        //Instantiate(/*thePlatform*/ thePlatforms[platformSelector], transform.p
osition, transform.rotation);
        GameObject newPlatform = theObjectPools[platformSelector].GetPool
edObject();
        newPlatform.transform.position = transform.position;
        newPlatform.transform.rotation = transform.rotation;
        newPlatform.SetActive(true);
        if (Random.Range (0f, 100f) < randomCoinSpawnRate)
        {
            theCoinGenerator.SpawnCoins (new Vector3 (transform.position.x, t
ransform.position.y + 1f, transform.position.z));
        }
        if (Random.Range (0f, 100f) < randomCrateSpawnRate)
        {
            GameObject newCrate = cratePool.GetPooledObject ();
            float crateXPosition = Random.Range(-
platformWidths[platformSelector]  /2 + 1f, platformWidths[platformSelector] /2 -
1f);
            Vector3 cratePosition = new Vector3 (crateXPosition, 3.5f, 0f);
```

```
            newCrate.transform.position = transform.position + cratePosition;
            newCrate.transform.rotation = transform.rotation;
            newCrate.SetActive (true);
        }
        transform.position = new Vector3(transform.position.x + (platformWidth
s[platformSelector] / 2), transform.position.y, transform.position.z);
    }
  }
}
```

Pause Menu:

```
public class PauseMenu : MonoBehaviour {
  public string mainMenuLevel;
  public GameObject pauseMenu;
  public void PauseGame()
  {
      Time.timeScale = 0f;
    pauseMenu.SetActive (true);
  }
  public void ResumeGame()
  {
    Time.timeScale = 1f;
    pauseMenu.SetActive (false);
  }
  public void RestartGame()
  {
    Time.timeScale = 1f;
    pauseMenu.SetActive (false);
```

```
        FindObjectOfType<GameManager> ().Reset ();
    }
    public void QuitToStatus()
    {
        Time.timeScale = 1f;
        SceneManager.LoadScene ("Status");
    }
}
```

Parallax:

```
public class Parallax : MonoBehaviour {
    private float length, startpos;
    public GameObject cam;
    public float parallaxEffect;
    // Use this for initialization
    void Start () {
        startpos = transform.position.x;
        length = GetComponent<SpriteRenderer>().bounds.size.x;
    }

    // Update is called once per frame
    void Update () {
        float temp = (cam.transform.position.x * (1 - parallaxEffect));
        float dist = (cam.transform.position.x * parallaxEffect);
        transform.position = new Vector3(startpos + dist, transform.position.y, tra
nsform.position.z);
        if (temp > startpos + length) startpos += length;
        else if (temp < startpos - length) startpos -= length;
```

```
    }
}
```

Pickup Drinks:

```
public class PickupDrinks : MonoBehaviour {
    public int scoreToGive;
    public GameObject coinParticle;
    private ScoreManager theScoreManager;
    private AudioSource coinSound;

    // Use this for initialization
    void Start () {
        theScoreManager = FindObjectOfType<ScoreManager> ();
        coinSound = GameObject.Find ("CoinSound").GetComponent<AudioSou
rce> ();
    }
    // Update is called once per frame
    void Update () {
    }
    void OnTriggerEnter2D (Collider2D other)
    {
        if (other.gameObject.name == "Player")
        {
            theScoreManager.AddScore (scoreToGive);
            Instantiate (coinParticle, other.transform.position, Quaternion.identity);
            gameObject.SetActive (false);
            if (coinSound.isPlaying) {
                coinSound.Stop ();
```

```
            coinSound.Play ();
        } else
        {
            coinSound.Play ();
    }
```

Object Pooler:

```
public class ObjectPooler : MonoBehaviour
{
    public GameObject pooledObject;
    public int pooledAmount;
    List<GameObject> pooledObjects;

    // Use this for initialization
    void Start()
    {
        pooledObjects = new List<GameObject>();
        for (int i = 0; i < pooledAmount; i++)
        {
            GameObject obj = (GameObject)Instantiate(pooledObject);
            obj.SetActive (false);
            pooledObjects.Add (obj);
        }
    }
    public GameObject GetPooledObject()
    {
        for (int i = 0; i < pooledObjects.Count; i++)
        {
            if (!pooledObjects[i].activeInHierarchy)
```

```
            {
                return pooledObjects[i];
            }
        }
        GameObject obj = (GameObject)Instantiate(pooledObject);
        obj.SetActive(false);
        pooledObjects.Add(obj);
        return obj;
    }
}
        }
    }
```

Main Menu:

```
public class MainMenu : MonoBehaviour {
    public void PlayGame()
    {
        SceneManager.LoadScene("Status");
    }
    public void QuitGame()
    {
        Application.Quit ();
    }
    public void Reset()
    {
        PlayerPrefs.DeleteKey ("HighScore");
    }
}
```

HP Collide:

```
public class HPCollide : MonoBehaviour {
    void OnTriggerEnter2D (Collider2D col)
    {
        FindObjectOfType<PlayerMovement>().health += 10;
    }
}
```

Help Screen Manager:

```
public class HelpScreenManager : MonoBehaviour {

    // Use this for initialization
    void Start () {
    }

    // Update is called once per frame
    void Update () {
    }
    public void BackButton()
    {
        SceneManager.LoadScene ("Status");
    }
}
```

Game Manager:

```
public class GameManager : MonoBehaviour {
    public Transform platformGenerator;
    private Vector3 platformStartPoint;
```

```
public PlayerMovement thePlayer;
private Vector3 playerStartPoint;
private PlatformDestroyer[] platformList;
private ScoreManager theScoreManager;
public DeathMenu theDeathScreen;
public bool powerupReset;

// Use this for initialization
void Start ()
{
    platformStartPoint = platformGenerator.position;
    playerStartPoint = thePlayer.transform.position;
    theScoreManager = FindObjectOfType<ScoreManager> ();
}

// Update is called once per frame
void Update ()
{
}
public void RestartGame()
{
    theScoreManager.scoreIncreasing = false;
    thePlayer.gameObject.SetActive (false);
    theDeathScreen.gameObject.SetActive (true);
    //StartCoroutine ("RestartGameCo");
}
public void Reset()
{
    theDeathScreen.gameObject.SetActive (false);
```

```
        platformList = FindObjectsOfType<PlatformDestroyer>();
        for(int i=0; i<platformList.Length; i++)
        {
            platformList[i].gameObject.SetActive(false);
        }
        thePlayer.transform.position = playerStartPoint;
        platformGenerator.position = platformStartPoint;
        thePlayer.gameObject.SetActive (true);
        theScoreManager.scoreCount = 0;
        theScoreManager.scoreIncreasing = true;
        powerupReset = true;
    }
    /*public IEnumerator RestartGameCo()
    {
        theScoreManager.scoreIncreasing = false;
        thePlayer.gameObject.SetActive (false);
        yield return new WaitForSeconds (0.9f);
        platformList = FindObjectsOfType<PlatformDestroyer>();
        for(int i=0; i<platformList.Length; i++)
            {
                platformList[i].gameObject.SetActive(false);
            }
        thePlayer.transform.position = playerStartPoint;
        platformGenerator.position = platformStartPoint;
        thePlayer.gameObject.SetActive (true);
        theScoreManager.scoreCount = 0;
        theScoreManager.scoreIncreasing = true;
    }*/
}
```

Enemies Script:

```
public class Enemies : MonoBehaviour {
    public GameObject coinParticle;
    private ScoreManager theScore;
    void Start()
    {
        theScore = FindObjectOfType<ScoreManager> ();
    }
    void OnTriggerEnter2D(Collider2D other)
    {
        if (other.tag == "Player")
        {
            Instantiate (coinParticle, other.transform.position, Quaternion.identity);
            PlayerMovement playermove = other.GetComponent<PlayerMovemen
t> ();
            playermove.GetDamage (15f);
            theScore.scoreCount -= 50f;
            gameObject.SetActive (false);
        }
    }
}
```

Don’t Destroy Script:

```
public class DontDestroy : MonoBehaviour {
    void Awake()
    {
        GameObject[] objs = GameObject.FindGameObjectsWithTag ("score");
        if (objs.Length > 1) {
```

```
            DontDestroyOnLoad (this.gameObject);
        }
    }
```

Death Menu Script:

```
public class DeathMenu : MonoBehaviour {
    public string mainMenuLevel;
    public void RestartGame()
    {
        FindObjectOfType<GameManager> ().Reset ();
        SceneManager.LoadScene(4);
    }
    public void Boy()
    {
        FindObjectOfType<GameManager> ().Reset ();
        SceneManager.LoadScene (3);
    }
    public void QuitToMain()
    {
        SceneManager.LoadScene(mainMenuLevel);
    }
}
```