

[Pizza Heist Code Walkthrough]

More details included in the code comments...

[Text Writer Files]

TextWriter.cs

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.UI;
5
6 public class TextWriter : MonoBehaviour
7 {
8
9     private static TextWriter instance;
10    private List<TextWriterSingle> textWriterSingleList;
11
12    private void Awake() {
13        Time.timeScale = 1;
14        instance = this;
15        textWriterSingleList = new List<TextWriterSingle>();
16    }
17
18    public static TextWriterSingle AddWriter_Static(Text uiText, string textToWrite, float timePerCharacter, bool invisibleCharacters, bool removeWriterBeforeAdd) {
19        if (removeWriterBeforeAdd) {
20            instance.RemoveWriter(uiText);
21        }
22        return instance.AddWriter(uiText, textToWrite, timePerCharacter, invisibleCharacters);
23    }
24
25    private TextWriterSingle AddWriter(Text uiText, string textToWrite, float timePerCharacter, bool invisibleCharacters) {
26        TextWriterSingle textWriterSingle = new TextWriterSingle(uiText, textToWrite, timePerCharacter, invisibleCharacters);
27        textWriterSingleList.Add(textWriterSingle);
28        return textWriterSingle;
29    }
30
31    public static void RemoveWriter_Static(Text uiText) {
32        instance.RemoveWriter(uiText);
33    }
34
35    private void RemoveWriter(Text uiText) {
36        for (int i = 0; i < textWriterSingleList.Count; i++) {
37            if (textWriterSingleList[i].GetUIText() == uiText) {
38                textWriterSingleList.RemoveAt(i);
39                i--;
40            }
41        }
42    }
```

Displays a string of text onto a UI text element, one character at a time, similar to a typewriter.

Used for all text elements throughout the game.

ChatRoom Files

Left to Right: ChatRoom.cs, ChatRoom2.cs, ChatRoom3.cs

```
TeamRocketPizza > code > Pizza Heist Game > Assets > Script > ChatRoom.cs
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.UI;
5 using UnityEngine.UI;
6
7 public class ChatRoom : MonoBehaviour
8 {
9     //References to UI components
10     public Button startChatButton;
11     public Image chatImage;
12     public Image bossIcon;
13     private Text messageText;
14     private Text clickToContinue;
15
16     private TextWriter.TextWriterSingle textWriterSingle;
17     private int currentMessageIndex = 0;
18     public float delay = 30f;
19     private string[] messageArray;
20
21     private void Awake() {
22
23     messageText = transform.Find("message").Find("bossMessage").GetComponent<Text>();
24     clickToContinue = transform.Find("message2").Find("clickToContinue").GetComponent<Text>();
25
26     //Array of messages to be displayed
27     messageArray = new string[] {
28         "Welcome new recruit.",
29         "Welcome new recruit.",
30         "Congratulations on successfully logging in.",
31         "I'll walk you through everything since it's your first day.",
32         "The folders on your screen will help you complete today's tasks.",
33         "We're going to start off simple.",
34         "We received an influx of emails so we need your help to sort it all",
35         "Click on the 'E-mail' folder when you're ready to begin."
36     };
37
38     startChatButton.onClick.AddListener(OnStartChatClicked); //Event listener to start chat w
39
40     //Initially hides UI components
41     messageText.gameObject.SetActive(false);
42     chatImage.gameObject.SetActive(false);
43     bossIcon.gameObject.SetActive(false);
44     clickToContinue.gameObject.SetActive(false);
45
46 }
```

```
TeamRocketPizza > code > Pizza Heist Game > Assets > Script > ChatRoom2.cs
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.UI;
5 using UnityEngine.UI;
6
7 public class ChatRoom2 : MonoBehaviour
8 {
9     private Text messageText;
10     private Text clickToContinue;
11     private TextWriter.TextWriterSingle textWriterSingle;
12     private string[] messageArray;
13     private int currentMessageIndex = 0;
14     public float delay = 10f;
15
16     private void Awake() {
17     messageText = transform.Find("message").Find("bossMessage").GetComponent<Text>();
18     clickToContinue = transform.Find("message2").Find("clickToContinue").GetComponent<Text>();
19     clickToContinue.gameObject.SetActive(false);
20
21     //Array of messages to be displayed
22     messageArray = new string[] {
23         "Oh shoot! Rookie, I have bad news.",
24         "I clicked on a phishing link and now the shop is compromised.",
25         "ksj7XBu weXNSpW9 28456",
26         "Quick! There's still a way to fix it!",
27         "We have an advanced anti-virus program built just for this.",
28         "Total $4687;K 39456",
29         "Once you get it working, it will be as if none of this happened in the first place!",
30         "It will detect irregular patterns and defend against anything malicious.",
31         "us*89sdHl 8*6ShdHl K3*8989fsd",
32         "It's a bit complicated, but I'll walk you through it.",
33         "Click on the 'Anti-Virus' folder. Hurry!"
34     };
35
36     //Functionality to continue the chat when the user clicks on the message box
37     transform.Find("message").GetComponent<Button UI>().ClickFunc = () => {
38         if (textWriterSingle != null && textWriterSingle.IsActive()) {
39             textWriterSingle.WriteAllAndDestroy();
40         } else {
41             ShowNextMessage();
42         }
43     };
44 }
```

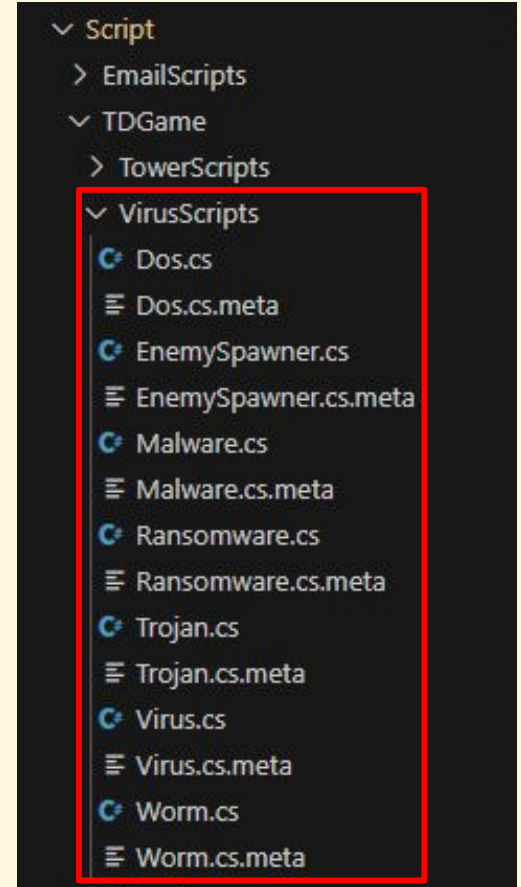
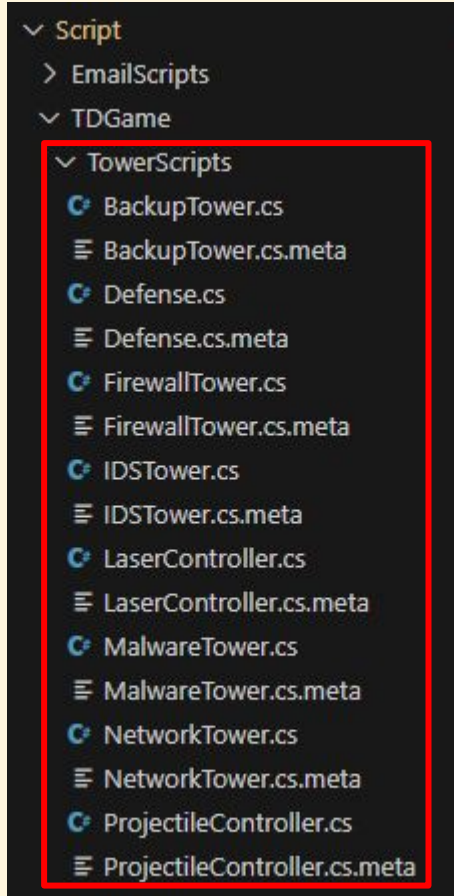
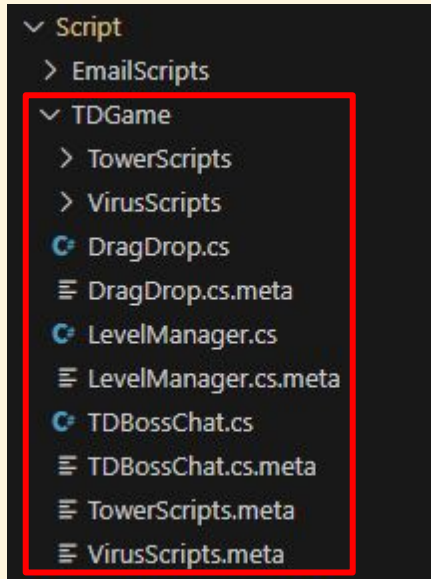
```
TeamRocketPizza > code > Pizza Heist Game > Assets > Script > ChatRoom3.cs
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.UI;
5 using UnityEngine.UI;
6
7 public class ChatRoom3 : MonoBehaviour
8 {
9     private Text messageText;
10     private Text clickToContinue;
11     private TextWriter.TextWriterSingle textWriterSingle;
12     private string[] messageArray;
13     private int currentMessageIndex = 0;
14     public float delay = 5f;
15
16     private void Awake() {
17     messageText = transform.Find("message").Find("bossMessage").GetComponent<Text>();
18     clickToContinue = transform.Find("message2").Find("clickToContinue").GetComponent<Text>();
19     clickToContinue.gameObject.SetActive(false);
20
21     //Array of messages to be displayed
22     messageArray = new string[] {
23         "Good work, new recruit.",
24         "Your survived your first day on the job.",
25         "And you even saved our shop!",
26         "Maybe I'll actually keep you around, rookie.",
27         "It's about time you clock out for the day.",
28         "I'll see you again tomorrow, bright and early.",
29         "Don't forget to log out!"
30     };
31
32     //Functionality to continue the chat when the user clicks on the message box
33     transform.Find("message").GetComponent<Button UI>().ClickFunc = () => {
34         if (textWriterSingle != null && textWriterSingle.IsActive()) {
35             textWriterSingle.WriteAllAndDestroy();
36         } else {
37             ShowNextMessage();
38         }
39     };
40
41     private void Start() {
42         ShowNextMessage();
43     }
44 }
```

Calls the TextWriter function to print an array of text to print the messages as seen on the different Desktop scenes

[Tower Defense Files]

Folders

All of the Tower Defense files are located in the “TDGame” folder. Tower and Virus files are in their respective folders.



Tower Files

Top: Defenses.cs, FirewallTower.cs

Bottom: MalwareTower.cs, NetworkTower.cs, IDSTower.cs

Creates the unique logic for the different towers.

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEditor;
4 using UnityEngine;
5 using UnityEngine.EventSystems;
6
7 // [DEFENSE TOWER PARENT CLASS]
8 // Purpose: hold function that all Tower Defense have
9 public class Defense : MonoBehaviour
10 {
11
12
13
14 [Header("References")]
15 public Transform rotPoint;
16 public LayerMask enemyMask;
17 public GameObject projectilePrefab;
18 public Transform firingPoint;
19
20
21 [Header("Attribute")]
22 public float firerate;
23 public float rotationSpeed;
24 public float targetRange;
25 public int cost;
26 public Transform target;
27 public float fireCooldown;
28 public bool isSold = false;
29 public Transform occupiedSlot;
30
31
32 private AudioSource audioOrig;
33
34 private void Start() { // get any component at the first frame
35     audioOrig = GetComponent();
36 }
37 }
```

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEditor;
5
6
7 // [FIREWALL TOWER CHILD CLASS]
8 // Purpose: A child class that only holds function for the firewall tower
9
10 public class FirewallTower : Defense
11 {
12
13     private ParticleSystem partSys;
14     private new AudioSource audio;
15     private void Start() { // get game component at the start of the first frame
16         partSys = GetComponentInChildren<ParticleSystem>();
17         audio = GetComponent();
18     }
19
20     private void Update() { // New update method that changes the shooting and target Find
21         ClickEvent();
22         if (target == null) {
23             FindTarget();
24             return;
25         }
26
27         if (!CheckTargetInRange()) {
28             target = null;
29         }
30         else {
31             fireCooldown += Time.deltaTime;
32             if (fireCooldown >= 1f / firerate) {
33                 Shoot();
34                 fireCooldown = 0f;
35             }
36         }
37     }
38 }
```

```
1 using System;
2 using System.Collections;
3 using System.Collections.Generic;
4 using Unity.Mathematics;
5 using UnityEditor;
6 using UnityEngine;
7 using UnityEngine.Pool;
8 using UnityEngine.XR;
9
10 // [MALWARE TOWER CHILD CLASS]
11 // Purpose: A child class that only holds function for the malware tower
12 public class MalwareTower : Defense
13 {
14     private void Update() { // malware tower's update method
15         ClickEvent();
16         if (target == null) {
17             FindTarget(); // find target
18             return;
19         }
20
21         RotateTowardsTarget(); // rotate turret
22         if (!CheckTargetInRange()) { // check turret
23             target = null;
24         }
25         else {
26             fireCooldown += Time.deltaTime; // firecooldown timer
27             if (fireCooldown >= 1f / firerate) {
28                 Shoot(); // Shoot
29                 fireCooldown = 0f;
30             }
31         }
32     }
33 }
```

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEditor;
5
6 // [BACKUP TOWER CHILD CLASS]
7 // Purpose: A child class that only holds function for the backup tower
8 public class NetworkTower : Defense
9 {
10
11     private ParticleSystem partSys;
12     public int shockDamage;
13     private new AudioSource audio;
14     private void Start() {
15         partSys = GetComponentInChildren<ParticleSystem>();
16         audio = GetComponent();
17     }
18
19     private void Update() {
20         ClickEvent();
21         if (target == null) {
22             FindTarget();
23             return;
24         }
25
26         if (!CheckTargetInRange()) {
27             target = null;
28         }
29         else {
30             fireCooldown += Time.deltaTime;
31             if (fireCooldown >= 1f / firerate) {
32                 Shoot();
33                 fireCooldown = 0f;
34             }
35         }
36     }
37
38     public void Shockpulse() {
39         Collider2D[] colliders = Physics2D.OverlapCircleAll(transform.position, targetRange, enemyMask);
40         foreach (Collider2D c in colliders) {
41             Virus virus = c.GetComponent<Virus>();
42         }
43     }
44 }
```

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEditor;
5
6
7 // [IDS TOWER CHILD CLASS]
8 // Purpose: A child class that only holds function for the backup tower
9 public class IDSTower : Defense
10 {
11
12     private ParticleSystem partSys;
13     public int laserDamage;
14     private new AudioSource audio;
15     private void Start() { // Get component at the start of the first frame
16         audio = GetComponent();
17         partSys = GetComponentInChildren<ParticleSystem>();
18     }
19
20     private void Update() {
21         ClickEvent();
22         if (target == null) {
23             FindTarget();
24             return;
25         }
26
27         if (!CheckTargetInRange()) {
28             target = null;
29         }
30         else {
31             RotateTowardsTarget(); // Rotate target
32             fireCooldown += Time.deltaTime; // Cooldown timer
33             if (fireCooldown >= 1f / firerate) {
34                 Shoot(); // Shoot
35                 fireCooldown = 0f;
36             }
37         }
38     }
39 }
```

Tower Files

Left to Right: ProjectileController.cs, LaserController.cs

```
1 using System.Collections;
2 using System.Collections.Generic;
3
4 using UnityEngine;
5
6 public class ProjectileController : MonoBehaviour
7 {
8
9
10     [Header("References")]
11     [SerializeField] private Rigidbody2D rb;
12     [SerializeField] private Transform rotPoint;
13
14     [Header("Attribute")]
15     [SerializeField] private float projSpeed = 1f;
16     [SerializeField] private int projDamage = 1;
17
18     private Transform target;
19     private Vector2 direction;
20
21     void Start(){
22         // If no target is set, move in the default forward direction
23         if (target == null){
24             Destroy(gameObject,0.2f); // Default forward direction of the projectile
25         }
26     }
27
28     void Update(){
29         if(target != null)
30         {
31             // Calculate direction towards the target
32             direction = (target.position - transform.position).normalized;
33             RotateTowardsTarget();
34         }
35
36         // Continue moving in the last known direction, even if the target is destroyed
37         rb.velocity = direction * projSpeed;
38     }
39
40     private void RotateTowardsTarget(){
41         if (target == null) return; // Avoid trying to rotate if there's no target
42
43         float angle = Mathf.Atan2( // Algorithm to track angle rotation towards target
44             target.position.y - transform.position.y,
45             target.position.x - transform.position.x)
46             * Mathf.Rad2Deg - 90f;
```

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5
6 // [LASER CONTROLLER CLASS]
7 // Purpose: A child class that only holds function for the backup tower
8 public class LaserController : MonoBehaviour
9 {
10     private IDSTower IDSTower;
11     public GameObject exploPref;
12     private Dictionary<GameObject, float> enemyCooldowns = new Dictionary<GameObject, float>();
13     public float hitCooldown = 0.5f; // Cooldown duration in seconds
14
15     private void OnParticleCollision(GameObject other)
16     {
17         if (other.CompareTag("Enemy"))
18         {
19             // Check if the enemy is in the cooldown period
20             if (enemyCooldowns.ContainsKey(other) && Time.time < enemyCooldowns[other])
21             {
22                 return; // Skip if still in cooldown
23             }
24
25             // Get the Virus component from the collided enemy
26             Virus virus = other.GetComponent<Virus>();
27             if (virus != null)
28             {
29                 // Apply damage to the enemy
30                 virus.TakeDamage(IDSTower.laserDamage);
31
32                 // Instantiate the explosion effect at the enemy's position
33                 GameObject ex = Instantiate(exploPref, virus.transform.position, Quaternion.identity);
34                 Destroy(ex, 1f); // Destroy the explosion effect after 1 second
35
36                 // Set the hit cooldown time for this enemy
37                 enemyCooldowns[other] = Time.time + hitCooldown;
38             }
39
40             Debug.Log("Enemy found and processed");
41         }
42     }
43 }
```

Handles the projectiles ejected from towers. Laser is only used for the IDS.

Virus Files

Virus.cs

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using System.Threading;
4 using UnityEngine;
5 using UnityEngine.EventSystems;
6
7 public abstract class Virus : MonoBehaviour
8 {
9
10     [Header("References")]
11     public Rigidbody2D rb;
12     public SpriteRenderer spriteRenderer;
13     // Start is called before the first frame update
14     [Header("Virus Stats")]
15     public int hitPoints;
16     public int coinsWorth;
17     public int damage;
18     public float speed;
19     public int waypointIndex = 0;
20     public Transform target;
21     // Update is called once per frame
22
23     private float origSpeed;
24     private Color originalColor;
25     private Coroutine slowEffectCoroutine;
26
27     public virtual void Start()
28     {
29         target = LevelManager.main.waypoints[waypointIndex];
30         //myTran = transform;
31         originalColor = spriteRenderer.color;
32         origSpeed = speed;
33     }
34
35     public IEnumerator WaitTimer(){
36         yield return new WaitForSeconds(0.2f);
37     }
38
39     protected virtual void Update()
40     {
41         if(Vector2.Distance(target.position, transform.position) < 0.1f){
42             waypointIndex++;
43         }
44     }
```

Parent class to all of the different viruses

Virus Files

Top: Trojan.cs, Worm.cs

Bottom: Ransomware.cs, Malware.cs, Dos.cs

Inherits from the parent class to create different unique viruses

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class Trojan : Virus
6 {
7
8     public Sprite happySprite; // The sprite for the happy state
9     public Sprite angrySprite; // The sprite for the berserk state
10    public int berserkSpeed = 3;
11    public int berserkHealth = 10; // Speed multiplier for berserk mode
12    private bool isBerserk = false; // Tracks whether the Trojan is in berserk mode
13
14
15    private Animator animator;
16
17    public override void Start()
18    {
19        base.Start();
20        spriteRenderer = GetComponent();
21        animator = GetComponent();
22
23        // Set initial happy state
24        spriteRenderer.sprite = happySprite;
25        spriteRenderer.color = Color.yellow; // Set the color to yellow for happy
26        animator.Play("HappyWalk");
27
28    public override void UseAbilities()
29    {
30        {
31            if (!isBerserk)
32            {
33                isBerserk = true;
34
35                // Transform to berserk state
36                spriteRenderer.sprite = angrySprite; // Change to the angry sprite
37                spriteRenderer.color = Color.red; // Change the color to red for angry
38                speed = berserkSpeed;
39                hitPoints = berserkHealth; // Increase speed
40                animator.Play("AngryWalk");
41                Debug.Log("Trojan is now berserk!");
42
43                // Optionally, add visual or sound effects here
44            }
45        }
46    }
47 }
```

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class Worm : Virus
6 {
7     public GameObject virusPrefab; // The prefab to use for duplication
8     public int numberOfDuplicates = 2; // Number of duplicates to spawn
9     private static int activeDuplicates = 0; // Tracks how many duplicates are still alive
10    private bool isOriginal = true; // Indicates whether this is the original worm
11
12    private ParticleSystem explode;
13
14    public override void Start()
15    {
16        base.Start();
17        explode = GetComponentInChildren<ParticleSystem>();
18
19    public override void UseAbilities()
20    {
21        Debug.Log("UseAbilities");
22
23        // Spawn the specified number of duplicates
24        for (int i = 0; i < numberOfDuplicates; i++)
25        {
26            Vector3 spawnPosition = transform.position + new Vector3(Random.Range(-0.5f, 0.5f), Random.Range(-0.5f, 0.5f), 0f);
27            GameObject duplicate = Instantiate(virusPrefab, spawnPosition, Quaternion.identity);
28
29            // Mark the duplicate as a non-original copy
30            Worm duplicateWorm = duplicate.GetComponent<Worm>();
31            if (duplicateWorm != null)
32            {
33                duplicateWorm.isOriginal = false;
34                duplicateWorm.SetPosition(wormPosition, target);
35                activeDuplicates++;
36            }
37        }
38
39    }
40
41    // Method to set the duplicate's waypoint index and target
42    public void SetPathInfo(int currentWaypointIndex, Transform currentTarget)
43    {
44        waypointIndex = currentWaypointIndex;
45        target = currentTarget;
46
47    public override void OnDeath()
48    {
49        if (!isOriginal)
50        {
51            // Destroy the duplicate
52            Destroy(gameObject);
53        }
54    }
55 }
```

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class Ransomware : Virus
6 {
7
8     public int taxAmount;
9     // Start is called before the first frame update
10    public override void UseAbilities()
11    {
12        if (waypointIndex == LevelManager.main.waypoints.Length)
13        {
14            if (LevelManager.main.coins >= 0)
15            {
16                LevelManager.main.coins -= taxAmount;
17                if (LevelManager.main.coins <= 0)
18                {
19                    LevelManager.main.coins = 0;
20                }
21            }
22        }
23
24        new void Update()
25        {
26            base.Update();
27            UseAbilities();
28        }
29    }
30 }
```

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class Malware : Virus
6 {
7
8     // Start is called before the first frame update
9     public override void UseAbilities()
10    {
11        {
12            return;
13        }
14
15        new void Update()
16        {
17            base.Update();
18        }
19    }
20 }
```

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class Dos : Virus
6 {
7     public float stunRange;
8     public float stunDuration;
9     public float stunCooldown;
10    private float nextStunTime;
11    private ParticleSystem partSys;
12
13    // Dictionary to track currently stunned towers
14    private Dictionary<Defense, Coroutine> stunnedTowers = new Dictionary<Defense, Coroutine>();
15
16    public override void Start()
17    {
18        base.Start();
19        partSys = GetComponentInChildren<ParticleSystem>();
20        nextStunTime = Time.time;
21
22    public override void UseAbilities()
23    {
24        if (Time.time > nextStunTime)
25        {
26            Collider2D[] colliders = Physics2D.OverlapCircleAll(transform.position, stunRange);
27            foreach (Collider2D collider in colliders)
28            {
29                Defense tower = collider.GetComponent<Defense>();
30                if (tower != null)
31                {
32                    // Only start a stun if the tower is not already stunned
33                    if (!stunnedTowers.ContainsKey(tower))
34                    {
35                        Coroutine stunCoroutine = StartCoroutine(StunTower(tower, stunDuration));
36                        stunnedTowers.Add(tower, stunCoroutine);
37                        partSys.Play();
38                    }
39                }
40            }
41
42            nextStunTime = Time.time + stunCooldown;
43        }
44    }
45 }
```

Virus Files

EnemySpawner.cs

```
1  using System;
2  using System.Collections;
3  using System.Collections.Generic;
4  using UnityEngine;
5  using UnityEngine.Events;
6  using UnityEngine.UI;
7
8  public class EnemySpawner : MonoBehaviour
9  {
10     // Start is called before the first frame update
11     [Header("References")]
12     [SerializeField] private GameObject[] enemyPrefabs;
13
14     [Header("Attributes")]
15     [SerializeField] private int baseEnemyCount = 8;
16     [SerializeField] private float spawnInterval = 0.75f;
17     [SerializeField] private float enemiesPerSecondCap = 15f;
18     [SerializeField] private float timeBetweenWaves = 2f;
19     [SerializeField] private float difficultyScalingFactor = 0.35f;
20
21     [Header("Events")]
22     public static UnityEvent enemyDestroy = new UnityEvent();
23     [SerializeField] private Button startWaveButton;
24     [SerializeField] private Color originalColor; // Original color
25     [SerializeField] private Color dullColor; // Dull color
26
27     private Image buttonImage;
28     private int currentEnemyWave = 1;
29     private float timeSinceLastSpawn;
30     private int enemiesAlive;
31     private float eps; //enemies per second
32     private int enemiesLeftToSpawn;
33     private bool isSpawning = false;
34     private Text wave;
35
36
37
38
39
40     void Start()
41     {
42
43         buttonImage = startWaveButton.GetComponent<Image>();
44         wave = GameObject.Find("WaveText").GetComponent<Text>();
45         if (startWaveButton != null)
46         {
47             startWaveButton.onClick.AddListener(OnStartWaveButtonClicked);
48         }
49     }
50 }
```

Handles the spawning rate for viruses to appear

Game Files

Left to Right: DragDrop.cs, LevelManager.cs

```
1 using System.Collections.Generic;
2 using UnityEngine;
3 using UnityEngine.EventSystems;
4 using UnityEngine.Tilemaps;
5 using UnityEngine.UI;
6
7
8 public class DragDrop : MonoBehaviour, IBeginDragHandler, IDragHandler, IEndDragHandler, IPointerDownHandler
9 {
10     [SerializeField] private GameObject prefabInstance;
11     [SerializeField] private GameObject prefabCollider;
12     private GameObject currentInstance;
13     private Canvas canvas; //grab the component from canvas
14     private Camera mainCamera;
15     private Defense tower;
16     private List<Transform> slotTransforms = new List<Transform>();
17     private Text defDesc;
18     private Text defName;
19
20
21 void Start()
22 {
23     canvas = GetComponentInParent<Canvas>();
24     GameObject[] slots = GameObject.FindGameObjectsWithTag("Slots");
25     foreach (GameObject slot in slots)
26     {
27         slotTransforms.Add(slot.transform);
28     }
29     mainCamera = Camera.main;
30     defDesc = GameObject.Find("DefenseDescription").GetComponent<Text>();
31     defName = GameObject.Find("DefenseName").GetComponent<Text>();
32 }
33
34 public void OnBeginDrag(PointerEventData eventData)
35 {
36     Debug.Log("OnBeginDrag");
37
38     if (prefabInstance != null)
39     {
40         // Instantiate the prefab in the game world
41         currentInstance = Instantiate(prefabInstance);
42         tower = currentInstance.GetComponent<Defense>();
43
44         tower.enabled = false;
45
46         // Set the initial position of the object
47         Vector3 worldPosition;
48         RectTransformUtility.ScreenPointToWorldPointInRectangle(
```

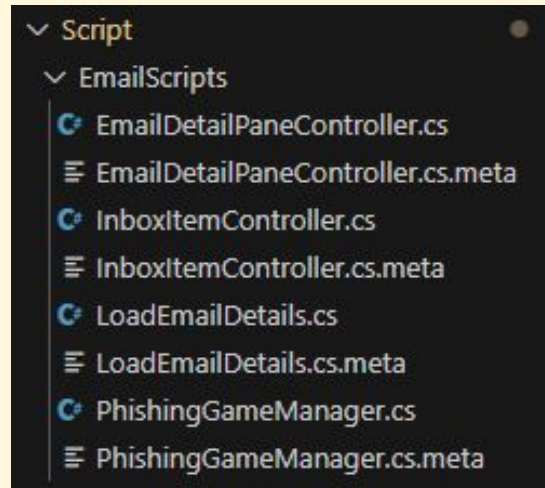
```
1 using System.Collections;
2 using System.Collections.Generic;
3 using System.Xml.Serialization;
4 using JetBrains.Annotations;
5 using UnityEngine;
6 using UnityEngine.SceneManagement;
7 using UnityEngine.UI;
8
9
10 using Image = UnityEngine.UI.Image;
11
12 public class LevelManager : MonoBehaviour
13 {
14     // Start is called before the first frame update
15     public static LevelManager main;
16     public Transform startPoint;
17     public Transform[] waypoints;
18     public LevelLoader transitionRef;
19     [SerializeField] private Button fastForwardButton;
20     [SerializeField] private Color originalColor; // Original color
21     [SerializeField] private Color dullColor; // Dull color
22     private Text healthText;
23     private Text coinText;
24     private Text GameState;
25     private CanvasGroup _cgroup;
26     private CanvasGroup wavegroup;
27     private CanvasGroup Chatcgroup;
28     private CanvasGroup imgGroup;
29     public int coins;
30     public int health;
31     private bool gameOver;
32     private GameStateEnum currentState;
33     private EnemySpawner enemySpawner;
34     private bool isFastForwarding = false;
35
36     [Header("Virus images")]
37     [SerializeField] private Image images;
38     [SerializeField] private Sprite malware;
39     [SerializeField] private Sprite ransomware;
40     [SerializeField] private Sprite dos;
41     [SerializeField] private Sprite worm;
42     [SerializeField] private Sprite trojan;
43 }
```

Handles the game functions by letting users drag and drop towers to placeholders and managing the different levels

[Phishing Email Files]

Folders

All of the Phishing Email files are located in the “EmailScripts” folder



PhishingGameManager.cs

Main controller for the game that controls the game flow and interactions between different game objects.

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine.SceneManagement;
4 using UnityEngine;
5 using UnityEngine.UI;
6
7 /*
8  * Title: PhishingGameManager
9  * Description: This is the main controller for the Phishing Game. It controls the game flow and the interactions between the different game objects.
10  * Author: Brian Ramos Cazares
11  *
12  * Note:
13  * - This script is not ideal and needs to be refactored. It is currently doing too much and needs to be broken down into smaller scripts. It all works nonetheless.
14  * -
15  */
16
17
18 public class PhishingGameManager : MonoBehaviour
19 {
20
21     //PREFAB SETUP -----
22     [Header("Prefab Setup")] //This will display a header in the inspector
23     public LevelLoader ReplayTransition; // Level Loader for the game
24     public TextAsset emailsJson; // JSON file containing the emails
25     public GameObject emailObjectPrefab; // Prefab for the email inbox object
26     public GameObject emailDetailsPrefab; // Prefab for the email details object
27     public GameObject emailListPane; // Parent object for the email list, aka the container that will hold the email inbox objects
28     public GameObject emailDetailsPane; // Parent object for the email details, aka the container that will hold the email details objects
29     public GameObject gameEndPrefab; // Prefab for the game end object, aka the game over screen,
30     public GameObject bossChatPrefab; // Prefab for the boss chat object, this will be used to display the boss's messages when the player guesses incorrectly
31     //public GameObject bossChatPane; // Parent object for the boss chat, aka the container that will hold the boss chat objects
32     public GameObject HeartPrefab; // Prefab for a life heart object
33     public GameObject HeartContainer; // Parent object for the life hearts, aka the container that will hold the life heart objects
34     public GameObject AnswerCorrectPrefab; // Prefab for the correct answer object
35
36     //private TextWriter.TextWriterSingle textWriterSingle;
37     //TEXTWRITER SETUP -----
38     [Header("Text Writer Setup")] //This will display a header in the inspector
39     public Text dialogueText; // UI Text that will display the message
40     public Button continueButton; // Continue Button to proceed to the next message
41     public string[] messages; // Array of messages to cycle through
42     private int currentMessageIndex = 0; // Tracks the current message index
43     private bool isTyping = false; // Tracks if the typewriter is currently animating
44     private TextWriter.TextWriterSingle textWriterSingle;
45
46     //START BUTTON FUNCTIONALITY -----
47     [Header("Start Button Functionality")] //This will display a header in the inspector
48     public Button menuButton; // computer menu button in taskbar
```

LoadEmailDetails.cs

Finds and loads email details to the EmailDetailPane

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.UI;
5
6
7 /*
8  * Title: LoadEmailDetails
9  * Description: This script finds and loads the email details to the EmailDetailPane GameObjects in the EmailDetailPane scene.
10 * Author: Brian Ramos Cazares
11 *
12 * Note:
13 *   - This Scripts is not used in the current version of the game. LinkInEmail is not used in the current version of the game.
14 *   - Created to test the loading of email details to the EmailDetailPane GameObjects.
15 */
16
17 public class LoadEmailDetails : MonoBehaviour
18 {
19     // Start is called before the first frame update
20
21     //Variables to store email details
22     public bool isPhishingEmail;
23     public string senderName;
24     public string senderEmail;
25     public string subject;
26
27     public string emailContent;
28     public string linkInEmail;
29
30
31     //Corresponding variables to store the email details
32     public GameObject emailDetailsPrefab;
33
34
35
36
37     void Start()
38     {
39         LoadEmailDetailstoPrefab();
40     }
41 }
```


InboxItemController.cs

Holds and controls the InboxItem game objects in the Inbox scene

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  /*
6   * Title: InboxItemController
7   * Description: Holds/Controls the InboxItem GameObjects in the Inbox scene.
8   * Author: Brian Ramos Cazares
9   */
10
11
12
13
14 public class InboxItemController : MonoBehaviour
15 {
16
17     public Email email; // The email that this InboxItem represents
18
19     public void SetEmail(Email email) // Set the email that this InboxItem represents
20     {
21         this.email = email; // Set the email
22     }
23     public Email GetEmail() // Get the email that this InboxItem represents
24     {
25         return email; // Return the email
26     }
27
28 }
```

EmailDetailPaneController.cs


Holds and controls the EmailDetailPane game objects in the EmailDetailPane scene

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using Unity.VisualScripting;
4  using UnityEngine;
5  using UnityEngine.UI;
6
7  /*
8   * Title: EmailDetailPaneController
9   * Description: Holds/Controls the EmailDetailPane GameObjects in the EmailDetailPane scene.
10  * Author: Brian Ramos Cazares
11  *
12  * Note:
13  *   - This Scripts interacts with the Phishing/NotPhishing buttons in the EmailDetailPane scene.
14  */
15
16  public class EmailDetailPaneController : MonoBehaviour
17  {
18      private Email email; // The email that this InboxItem represents
19      public Button notPhishingButton; // The Not Phishing button
20      public Button phishingButton; // The Phishing button
21
22      public Email currentEmail; // The current email object
23      public GameObject inboxEntry; // The related inbox entry GameObject
24
25      // Delegate and event to notify the PhishingGameManager
26      public delegate void EmailAction(Email email, GameObject inboxEntry, bool isCorrectGuess);
27
28
29      public void SetUp(Email email, GameObject associatedInboxEntry) // Set the email that this InboxItem will represent
30      {
31          currentEmail = email; // Set the email
32          inboxEntry = associatedInboxEntry; // Set the associated inbox entry
33
34          notPhishingButton = GameObject.FindWithTag("NotPhishing").GetComponent<Button>(); // Find the Not Phishing button
35          phishingButton = GameObject.FindWithTag("IsPhishing").GetComponent<Button>(); // Find the Phishing button
36
37
38      }
```

[Other Files]

NavController.cs

Loads the next scene

TeamRocketPizza > code > Pizza Heist Game > Assets > Script >  NavController.cs

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.SceneManagement;
5
6  public class NavController : MonoBehaviour
7  {
8      // Load the next scene
9      public void GoToScene(string sceneName){
10         SceneManager.LoadScene(sceneName);
11     }
12 }
13
```

button_UI.cs

Handles the custom behavior set for the UI buttons, including triggering sound effects

```
1 using System;
2 using UnityEngine;
3 using UnityEngine.UI;
4 using UnityEngine.EventSystems;
5
6
7 /*
8  * Button in the UI
9  * */
10 public class Button_UI : MonoBehaviour, IPointerEnterHandler, IPointerExitHandler, IPointerClickHandler, IPointerDownHandler, IPointerUpHandler {
11
12     public Action ClickFunc = null;
13     public Action MouseRightClickFunc = null;
14     public Action MouseMiddleClickFunc = null;
15     public Action MouseDownOnceFunc = null;
16     public Action MouseUpFunc = null;
17     public Action MouseOverOnceTooltipFunc = null;
18     public Action MouseOutOnceTooltipFunc = null;
19     public Action MouseOverOnceFunc = null;
20     public Action MouseOutOnceFunc = null;
21     public Action MouseOverPerSecFunc = null; //Triggers every sec if mouseOver
22     public Action MouseUpdate = null;
23     public Action<PointerEventData> OnPointerClickFunc;
24
25     public enum HoverBehaviour {
26         Custom,
27         Change_Color,
28         Change_Image,
29         Change_SetActive,
30     }
31     public HoverBehaviour hoverBehaviourType = HoverBehaviour.Custom;
32     private Action hoverBehaviourFunc_Enter, hoverBehaviourFunc_Exit;
33
34     public Color hoverBehaviour_Color_Enter, hoverBehaviour_Color_Exit;
35     public Image hoverBehaviour_Image;
36     public Sprite hoverBehaviour_Sprite_Exit, hoverBehaviour_Sprite_Enter;
37     public bool hoverBehaviour_Move = false;
38     public Vector2 hoverBehaviour_Move_Amount = Vector2.zero;
39     private Vector2 posExit, posEnter;
40     public bool triggerMouseOutFuncOnClick = false;
41     private bool mouseOver;
42     private float mouseOverPerSecFuncTimer;
43
44     private Action internalOnPointerEnterFunc, internalOnPointerExitFunc, internalOnPointerClickFunc;
45
46     #if SOUND_MANAGER
47         public Sound_Manager.Sound mouseOverSound, mouseClickSound;
```

InputHandlerCopy.cs

Verifies input to pass specified constraints before proceeding to the next scene

```
1  using System.Collections;
2  using UnityEngine;
3  using UnityEngine.UI;
4  using System.Text.RegularExpressions;
5  using System;
6
7  public class InputHandlerCopy : MonoBehaviour
8  {
9      [SerializeField] InputField inputField;
10     [SerializeField] InputField userName;
11     [SerializeField] Text resultText;
12     [SerializeField] Button submitButton;
13     public LevelLoader transitionLoad;
14     Boolean login;
15     public Image accessBar;
16     public Text accessGranted;
17
18     private void Awake() {
19         accessBar.gameObject.SetActive(false);
20         accessGranted.gameObject.SetActive(false);
21     }
22
23     private void Start()
24     {
25
26         submitButton.onClick.AddListener(ValidateInput);
27     }
28
29     public void ValidateInput() {
30         string input = inputField.text;
31         login = false;
32
33         // Check if the password length is between 6 and 12 characters
34         if (input.Length < 12 || input.Length > 24)
35         {
36             resultText.text = "Password must be between 12 and 24 characters.";
37             return;
38         }
39
40         // Check if password contains at least one uppercase letter
41         if (!Regex.IsMatch(input, "[A-Z]"))
42         {
43             resultText.text = "Password must contain at least one uppercase letter.";
44             return;
45         }
46     }
```

CutSceneHandler.cs

Handles the animations and sound effects used for the cutscene sequences

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class CutsceneHandler : MonoBehaviour
6 {
7
8     public Animator[] anim;
9     private AudioSource woosh;
10    // Start is called before the first frame update
11    void Start()
12    {
13        woosh = GetComponent();
14        StartCoroutine(waitAnimation());
15    }
16
17    // Update is called once per frame
18    // void Update()
19    // {
20
21
22    //     StartCoroutine(waitAnimation());
23    // }
24
25    private IEnumerator waitAnimation(){
26        for(int i = 0; i < anim.Length; i++){
27
28            anim[i].SetTrigger("Slide");
29            woosh.Play();
30            if(i == anim.Length-1){
31                yield return new WaitForSeconds(0.1f);
32                anim[i].SetTrigger("Floating");
33            }
34
35            yield return new WaitForSeconds(2f);
36
37        }
38    }
39 }
```

AnimationHandler.cs

Handles the animation for specified components

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class AnimationController : MonoBehaviour
6  {
7      [SerializeField] Animator anim;
8      // Start is called before the first frame update
9      void Start()
10     {
11         anim = GetComponent<Animator>();
12     }
13
14     // Update is called once per frame
15     void Update()
16     {
17
18     }
19 }
20
```


GameEndController.cs

Handles the game ending

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.SceneManagement;
5 using UnityEngine.UI;
6
7 public class GameEndController : MonoBehaviour
8 {
9     // Start is called before the first frame update
10
11     public Button ReplayButton;
12     public Button ReturnButton;
13     public LevelLoader transitionRef;
14
15     public string returnSceneName;
16
17
18
19     void Start()
20     {
21
22         AddListenersToButtons();
23     }
24
25     // Update is called once per frame
26     void Update()
27     {
28     }
29
30
31     void AddListenersToButtons()
32     {
33         // Set up the button listeners
34         ReplayButton.onClick.AddListener(OnReplayButtonClicked);
35         ReturnButton.onClick.AddListener(OnReturnButtonClicked);
36         Debug.Log("Listeners added");
37     }
38
39
40
41
42
43     public void OnReplayButtonClicked()
44     {
45         // load the game scene
46         Debug.Log("Replay button clicked");
47         Time.timeScale = 1;
48         transitionRef.LoadNextLevel(SceneManager.GetActiveScene().name);
```