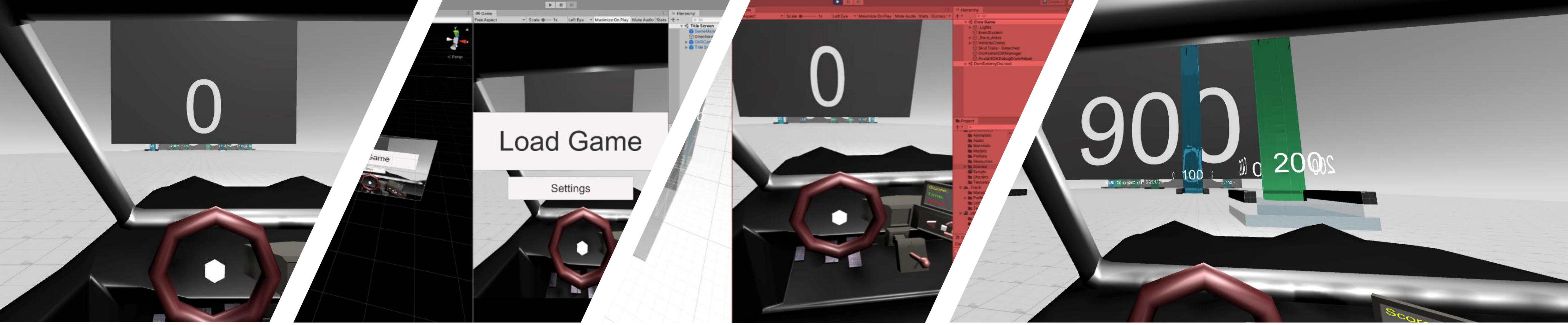
# Game Development in Unity Using Oculus Quest VR

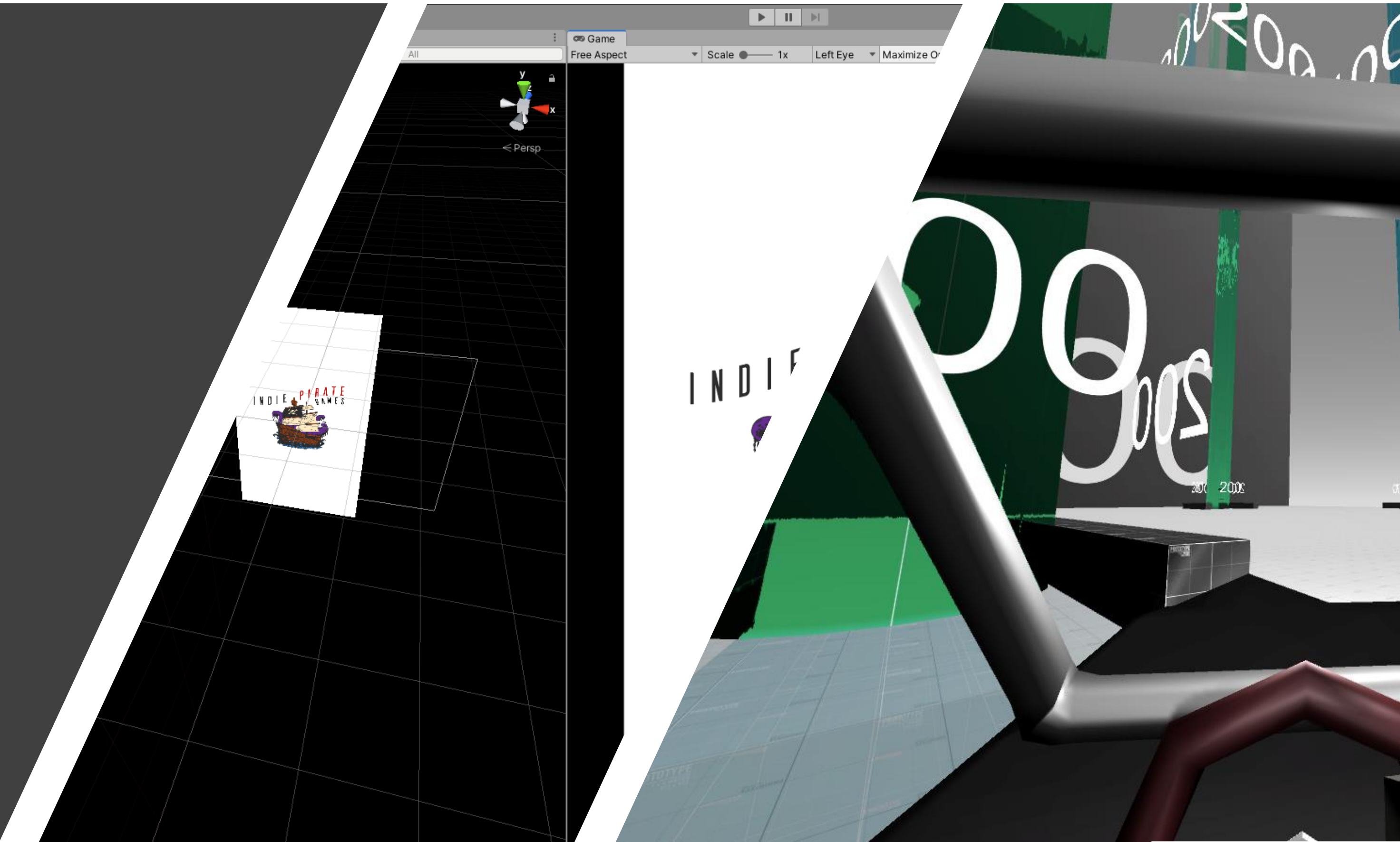
By Christopher Todd

Advisor: Dr. Daehee Kim

- What is Virtual Reality?
- What was I hoping to accomplish?
  - Development
  - Implement
  - Procedural game
- Related works?
  - iRacing for VR immersion
  - Borderlands for Procedural Generation



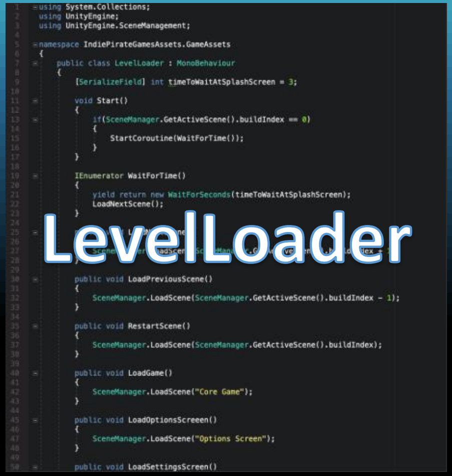
- What was I Able to Accomplish?
  - Setting Up a New Project
  - Understanding Synchronization
  - Learning What Was Different
  - Utilizing What I Discovered
  - Implementing What I Discovered



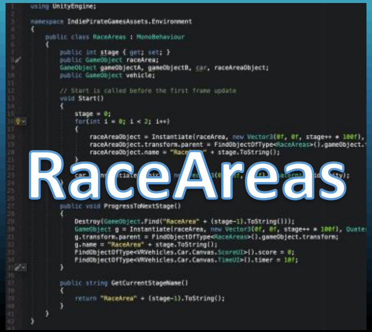
- Implementing My Game via Code
  - 830 lines of code in 29+ scripting classes
  - Code Will Be Available via Email
- Finishing the Game

## Scripts .cs

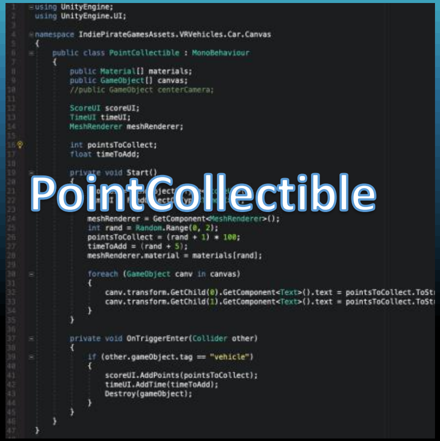
EndGameCollider
MoveWall
RaceAreas
StageCollider
Track
WallScore
AudioManager
CurrentBestStage
GameController
GameManager
LevelLoader
PlayerPrefsController
Sound
UserMessage
BonusPointDetection
GeneratePoints
PointCollectible
BreakLights
CarAudio
CarController
CarUserControl
ScoreUI
SkidTrail
SpeedUI
SteeringWheelColliders
SteeringWeelRotation
Suspension
TimeUI
WheelEffects



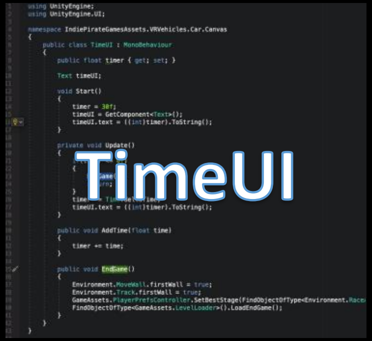
```csharp
using System.Collections;
using UnityEngine;
using UnityEngine.SceneManagement;

namespace IndiePirateGamesAssets.GameAssets
{
    public class LevelLoader : MonoBehaviour
    {
        [SerializeField] int timeToWaitAtSplashScreen = 3;

        void Start()
        {
            if(SceneManager.GetActiveScene().buildIndex == 0)
            {
                StartCoroutine(WaitForTime());
            }
        }

        IEnumerator WaitForTime()
        {
            yield return new WaitForSeconds(timeToWaitAtSplashScreen);
            LoadNextScene();
        }

        [illegible]
        {
            [illegible]
        }

        public void LoadPreviousScene()
        {
            SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex - 1);
        }

        public void RestartScene()
        {
            SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
        }

        public void LoadGame()
        {
            SceneManager.LoadScene("Core Game");
        }

        public void LoadOptionsScreeen()
        {
            SceneManager.LoadScene("Options Screen");
        }

        public void LoadSettingsScreen()
```



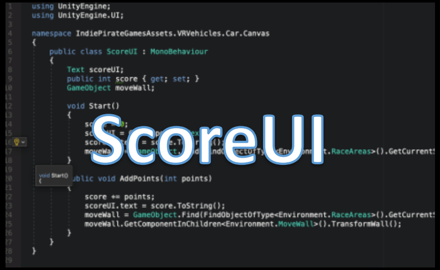
```csharp
using UnityEngine;

namespace IndiePirateGamesAssets.Environment
{
    public class RaceAreas : MonoBehaviour
    {
        public int stage { get; set; }
        public GameObject raceArea;
        GameObject gameObjectA, gameObjectB, car, raceAreaObject;
        public GameObject vehicle;

        // Start is called before the first frame update
        void Start()
        {
            stage = 0;
            for(int i = 0; i < 2; i++)
            {
                raceAreaObject = Instantiate(raceArea, new Vector3(0f, 0f, stage++ * 100f),
                raceAreaObject.transform.parent = FindObjectOfType<RaceAreas>().gameObject.
                raceAreaObject.name = "Race[illegible]" + stage.ToString();
            }

            car [illegible]
        }

        public void ProgressToNextStage()
        {
            Destroy(GameObject.Find("RaceArea" + (stage-1).ToString()));
            GameObject g = Instantiate(raceArea, new Vector3(0f, 0f, stage++ * 100f), Quate
            g.transform.parent = FindObjectOfType<RaceAreas>().gameObject.transform;
            g.name = "RaceArea" + stage.ToString();
            FindObjectOfType<VRVehicles.Car.Canvas.ScoreUI>().score = 0;
            FindObjectOfType<VRVehicles.Car.Canvas.TimeUI>().timer = 10f;
        }

        public string GetCurrentStageName()
        {
            return "RaceArea" + (stage-1).ToString();
        }
    }
}
```



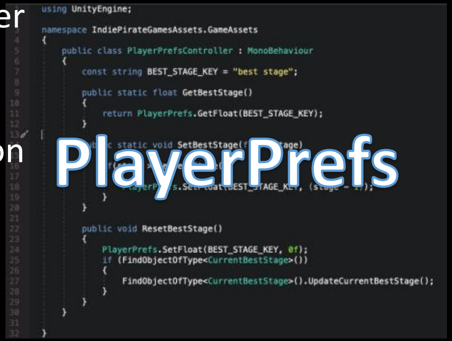
```csharp
using UnityEngine;
using UnityEngine.UI;

namespace IndiePirateGamesAssets.VRVehicles.Car.Canvas
{
    public class PointCollectible : MonoBehaviour
    {
        public Material[] materials;
        public GameObject[] canvas;
        //public GameObject centerCamera;

        ScoreUI scoreUI;
        TimeUI timeUI;
        MeshRenderer meshRenderer;

        int pointsToCollect;
        float timeToAdd;

        private void Start()
        {
            [illegible]
            [illegible]
            meshRenderer = GetComponent<MeshRenderer>();
            int rand = Random.Range(0, 2);
            pointsToCollect = (rand + 1) * 100;
            timeToAdd = (rand + 5);
            meshRenderer.material = materials[rand];

            foreach (GameObject canv in canvas)
            {
                canv.transform.GetChild(0).GetComponent<Text>().text = pointsToCollect.ToSt
                canv.transform.GetChild(1).GetComponent<Text>().text = pointsToCollect.ToSt
            }
        }

        private void OnTriggerEnter(Collider other)
        {
            if (other.gameObject.tag == "vehicle")
            {
                scoreUI.AddPoints(pointsToCollect);
                timeUI.AddTime(timeToAdd);
                Destroy(gameObject);
            }
        }
    }
}
```



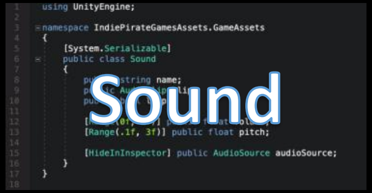
```csharp
using UnityEngine;
using UnityEngine.UI;

namespace IndiePirateGamesAssets.VRVehicles.Car.Canvas
{
    public class TimeUI : MonoBehaviour
    {
        public float timer { get; set; }

        Text timeUI;

        void Start()
        {
            timer = 30f;
            timeUI = GetComponent<Text>();
            timeUI.text = ((int)timer).ToString();
        }

        private void Update()
        {
            [illegible]
            {
                [illegible]
                [illegible]
            }
            timer [illegible]
            timeUI.text = ((int)timer).ToString();
        }

        public void AddTime(float time)
        {
            timer += time;
        }

        public void EndGame()
        {
            Environment.MoveWall.firstWall = true;
            Environment.Track.firstWall = true;
            GameAssets.PlayerPrefsController.SetBestStage(FindObjectOfType<Environment.Race
            FindObjectOfType<GameAssets.LevelLoader>().LoadEndGame();
        }
    }
}
```



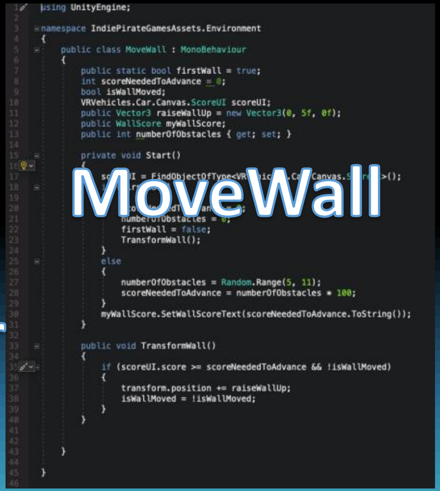
```csharp
using UnityEngine;
using UnityEngine.UI;

namespace IndiePirateGamesAssets.VRVehicles.Car.Canvas
{
    public class ScoreUI : MonoBehaviour
    {
        Text scoreUI;
        public int score { get; set; }
        GameObject moveWall;

        void Start()
        {
            sc[illegible]0;
            sc[illegible]
            scor[illegible] = score.To[illegible]();
            moveWa[illegible]t.RaceAreas>().GetCurrent
        }

        public void AddPoints(int points)
        {
            score += points;
            scoreUI.text = score.ToString();
            moveWall = GameObject.Find(FindObjectOfType<Environment.RaceAreas>().GetCurrent
            moveWall.GetComponentInChildren<Environment.MoveWall>().TransformWall();
        }
    }
}
```



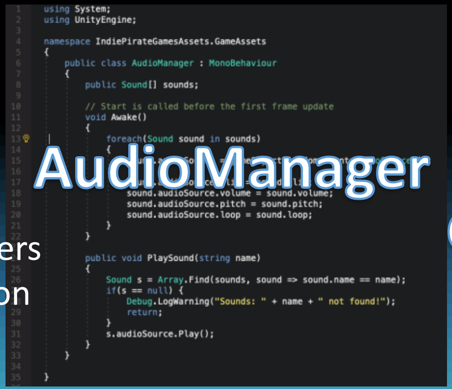
```csharp
using UnityEngine;

namespace IndiePirateGamesAssets.GameAssets
{
    public class PlayerPrefsController : MonoBehaviour
    {
        const string BEST_STAGE_KEY = "best stage";

        public static float GetBestStage()
        {
            return PlayerPrefs.GetFloat(BEST_STAGE_KEY);
        }

        [illegible] static void SetBestStage([illegible]tage)
        {
            [illegible]
            {
                [illegible]s.Se[illegible](BEST_STAGE_KEY, (stage - 1));
            }
        }

        public void ResetBestStage()
        {
            PlayerPrefs.SetFloat(BEST_STAGE_KEY, 0f);
            if (FindObjectOfType<CurrentBestStage>())
            {
                FindObjectOfType<CurrentBestStage>().UpdateCurrentBestStage();
            }
        }
    }

}
```



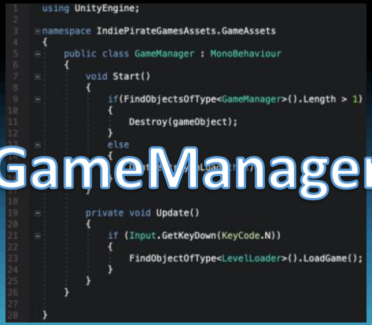
```csharp
using UnityEngine;

namespace IndiePirateGamesAssets.GameAssets
{
    [System.Serializable]
    public class Sound
    {
        pu[illegible] string name;
        [illegible]
        [illegible]

        [illegible]
        [Range(.1f, 3f)] public float pitch;

        [HideInInspector] public AudioSource audioSource;
    }
}
```



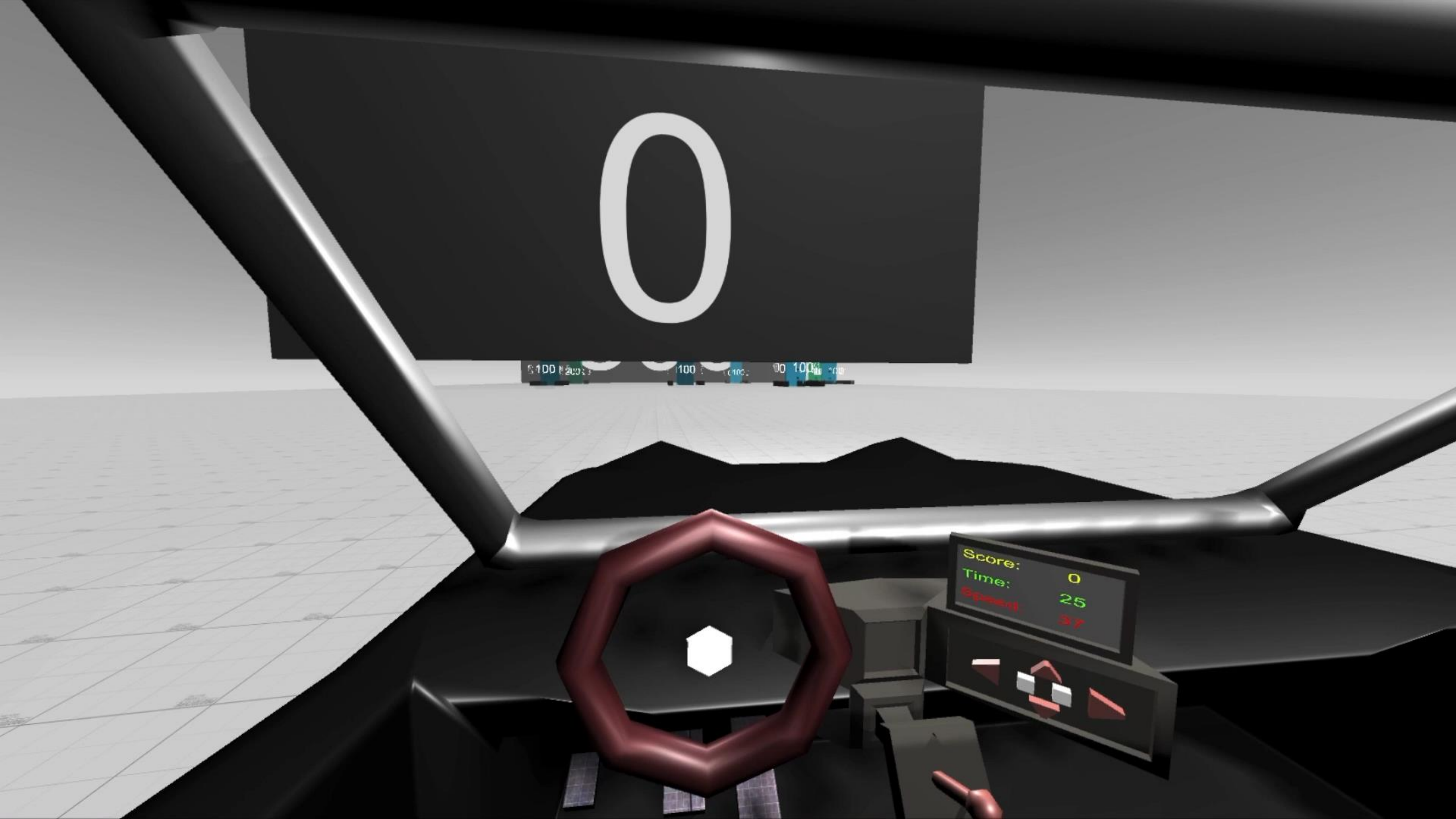
```csharp
using UnityEngine;

namespace IndiePirateGamesAssets.Environment
{
    public class MoveWall : MonoBehaviour
    {
        public static bool firstWall = true;
        int scoreNeededToAdvance = 0;
        bool isWallMoved;
        VRVehicles.Car.Canvas.ScoreUI scoreUI;
        public Vector3 raiseWallUp = new Vector3(0, 5f, 0f);
        public WallScore myWallScore;
        public int numberOfObstacles { get; set; }

        private void Start()
        {
            sc[illegible]I = FindObjectOfType<VR[illegible]Canvas.[illegible]>();
            [illegible]
            {
                [illegible] = 0;
                numberOfObstacles = 0;
                firstWall = false;
                TransformWall();
            }
            else
            {
                numberOfObstacles = Random.Range(5, 11);
                scoreNeededToAdvance = numberOfObstacles * 100;
            }
            myWallScore.SetWallScoreText(scoreNeededToAdvance.ToString());
        }

        public void TransformWall()
        {
            if (scoreUI.score >= scoreNeededToAdvance && !isWallMoved)
            {
                transform.position += raiseWallUp;
                isWallMoved = !isWallMoved;
            }
        }


    }

}
```



```csharp
using System;
using UnityEngine;

namespace IndiePirateGamesAssets.GameAssets
{
    public class AudioManager : MonoBehaviour
    {
        public Sound[] sounds;

        // Start is called before the first frame update
        void Awake()
        {
            foreach(Sound sound in sounds)
            {
                [illegible]
                [illegible]
                [illegible]
                sound.audioSource.volume = sound.volume;
                sound.audioSource.pitch = sound.pitch;
                sound.audioSource.loop = sound.loop;
            }
        }

        public void PlaySound(string name)
        {
            Sound s = Array.Find(sounds, sound => sound.name == name);
            if(s == null) {
                Debug.LogWarning("Sounds: " + name + " not found!");
                return;
            }
            s.audioSource.Play();
        }
    }

}
```



```csharp
using UnityEngine;

namespace IndiePirateGamesAssets.GameAssets
{
    public class GameManager : MonoBehaviour
    {
        void Start()
        {
            if(FindObjectsOfType<GameManager>().Length > 1)
            {
                Destroy(gameObject);
            }
            else
            {
                [illegible]
            }
        }

        private void Update()
        {
            if (Input.GetKeyDown(KeyCode.N))
            {
                FindObjectOfType<LevelLoader>().LoadGame();
            }
        }
    }

}
```


0
100
100
100
Score: 0
Time: 25
Speed: 37

## Questions & Answers

Christopher Todd
(ctodd1@csustan.edu)

Advisor: Dr. Daehee Kim
(dkim10@csustan.edu)