PROGRESS REPORT

MICROGAME #6: Racer

Andrew Adame

ID: 007516100

LEGEND: COMPLETED – UNFINISHED – WIP – FIX – FIXED

GITHUB: https://github.com/andrewadame/UnityProjectsCSE-4410/tree/master/RacingProject

UNITY PLAY: https://play.unity.com/mg/other/webgl-builds-186380

1. Create new project Racing Project
2. Create folders containing important assets (scripts, prefabs, animation, etc)
3. Create a basic Racing Game
    a. Design Level
        i. Tilemap
            1. Provided by professor
        ii. Camera
            1. CmCtrlr
                a. Two cameras that follow two players in the game
    b. Players
        i. Sprite
            1. RdCar
            2. PrpCar
        ii. Behavior
            1. Components
                a. BoxCollider2D
                b. Rigidbody2D
                    i. Added preferred physics that give cars some weight
        iii. Obstacles
            1. Added puddles that spinout players upon contact
        iv. Scripts
            1. GmeCtrlr
            2. CrCtrlr
    c. Visuals
        i. All sprites used were provided by the professor
    d. Gameplay
        i. Game Start

1. Starts with countdown, then race begins

ii. Objective

1. Complete a set amount of laps

iii. Game Over

1. Displays race winner, allows players to restart

e. UI

i. Countdown for Race

f. **EXTRA**

i. Audio

ii. Visual Lap Counter

SCRIPTS

GmeCtrlr

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

public class GmeCtrlr : MonoBehaviour
{
    public int laps;
    public Text winTxt;
    bool endGme = false;

    public Text cntdwn;
    public float tmeToStrt = 3f;
    public bool strtd = false;

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

    }

    // Update is called once per frame
    void Update()
    {
        if(tmeToStrt > 0)
        {
            tmeToStrt -= Time.deltaTime;
            cntdwn.text = Mathf.RoundToInt(tmeToStrt).ToString();
        }
        else
        {
            strtd = true;

            cntdwn.gameObject.SetActive(false);
        }


        if (endGme && Input.anyKeyDown)
```

```
        {
            SceneManager.LoadScene("SampleScene");
        }
    }

    public void EndGame(int num)
    {
        endGme = true;
        winTxt.gameObject.SetActive(true);
        winTxt.text = "Player " + num + " wins! Press any key to restart!";
    }
}
```

CarCtrlr

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

public class CarCtrlr : MonoBehaviour
{
    Rigidbody2D crRgdBdy;
    public float spd;
    public float bckSpd;
    public float rtSpd;
    Vector2 input;

    public string inputXNme;
    public string inputYNme;

    public GameObject cam;
    public int num;

    [SerializeField]
    int crntLp = 0;
    GmeCtrlr cont;

    float cools;
    public float slckTmr;
    public bool slicked = false;
    public float slickRot;
    Vector2 slickDir;

    public float rgDrg;
    public float slkDrg;
    float crntDrg;
    public float drgLrp;
    public bool htChckPnts = false;

    //Lap Txt
    public Text lpTxt;

    private void OnEnable()
    {
        //Create Cam and follow
```

```
        GameObject c = Instantiate(cam, transform.position, Quaternion.identity);

        c.GetComponent<CmCtrlr>().trgt = transform;
        if (num == 1)
        {
            c.GetComponent<Camera>().rect = new Rect(new Vector2(0f, 0f), new
Vector2(0.5f, 1f));
        }
        else
        {
            c.GetComponent<Camera>().rect = new Rect(new Vector2(0.5f, 0f), new
Vector2(0.5f, 1f));
        }

        crntLp = 0;
        crntDrg = rgDrg;
    }

    private void Awake()
    {
        crRgdBdy = GetComponent<Rigidbody2D>();
        cont = FindObjectOfType<GmeCtrlr>();
    }

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

    }

    // Update is called once per frame
    void Update()
    {
        //UI
        lpTxt.text = "Lap: " + crntLp.ToString() + "/" + cont.laps;

        //Controls
        if (cont.strtd && !slicked)
        {
            input = new Vector2(Input.GetAxis(inputXNme), Input.GetAxis(inputYNme));
            if (input.x != 0)
            {
                transform.Rotate(0, 0, -rtSpd * Time.deltaTime * input.x);
            }

            if (input.y > 0)
            {
                crRgdBdy.AddForce(transform.up * input.y * spd * Time.deltaTime);
            }

            if (input.y < 0)
            {
                crRgdBdy.AddForce(transform.up * input.y * bckSpd * Time.deltaTime);
            }
        }
        if(slicked)
        {
            crRgdBdy.AddForce(slickDir * bckSpd * Time.deltaTime);
```

```
            transform.Rotate(0, 0, slickRot * Time.deltaTime);

            if(cools <= 0)
            {
                slicked = false;
            }
        }

        if(cools > 0)
        {
            cools -= Time.deltaTime;
        }

        crntDrg = slicked ? slkDrg : rgDrg;
        crRgdBdy.drag = Mathf.Lerp(crRgdBdy.drag, crntDrg, drgLrp * Time.deltaTime) ;
    }

    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.gameObject.CompareTag("Goal") && htChckPnts)
        {
            crntLp++;

            if (crntLp >= cont.laps)
            {
                cont.EndGame(num);
            }

            htChckPnts = false;
        }

        if (collision.gameObject.CompareTag("Obstacle"))
        {
            slickDir = transform.up;
            cools = slckTmr;
            slicked = true;
        }

        if(collision.gameObject.CompareTag("Checkpoint"))
        {
            htChckPnts = true;
        }

    }
}
```

CmCtrlr

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

public class CmCtrlr : MonoBehaviour
{
    public Transform trgt;
    public float lrpSpd;
```

```
    Vector3 tempPos;
    //[SerializeField]
    //float minX, minY, maxX, maxY;

    // Update is called once per frame
    void FixedUpdate()
    {
        if (trgt == null) return;
        tempPos = trgt.position;
        tempPos.z = -10;

        /*
        //MIN
        if (trgt.position.x < minX)
        {
            tempPos.x = minX;
        }
        if (trgt.position.y < minY)
        {
            tempPos.y = minY;
        }

        //MAX
        if (trgt.position.x > maxX)
        {
            tempPos.x = maxX;
        }
        if (trgt.position.y > maxY)
        {
            tempPos.y = maxY;
        }
        */
        transform.position = Vector3.Lerp(transform.position, tempPos, lrpSpd *
Time.deltaTime);
    }
}
```