PROGRESS REPORT

MICROGAME #5: Platformer

Andrew Adame

ID: 007516100

LEGEND: COMPLETED – UNFINISHED – WIP – FIX – FIXED

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

UNITY PLAY: https://play.unity.com/mg/other/builds-fx-6

1. Create new project PlatformerProject
2. Create folders containing important assets (scripts, prefabs, animation, etc)
3. Create a basic Platformer Game
    a. Design Level
        i. Tilemap
            1. Floor
            2. Some Platforms
            3. Walls that confine Player and Enemies
        ii. Camera
            1. Follows Player
            2. Bounded to Level (Camera does not go past Walls
    b. Player
        i. Sprite
            1. Uses a robot sprite provided by professor
        ii. Behavior
            1. Components
                a. RigidBody2D
                b. CapsuleCollider2D
            2. Scripts
                a. PlyrCtrlr
                    i. Allows player control, movement, and abilities
            3. Abilities
                a. Jump
                    i. ERROR: Player double-jumps because of an unknown bug
                        1. CnJmp remains true after first jump
                    ii. Damages Enemy if Player lands on Enemy head

iii. Animation
1. Has an idle, jump-start, midair, land, and a walk animation

c. Enemies
i. Sprites
1. Uses a enemy sprites provided by professor
ii. Behavior
1. Enemies have three states that determine their current behavior
a. Move
i. Walks back and forth when Player not in ChseRng
b. Chase
i. Chases Player if in ChseRng
ii. Goes back to Move if Player escapes ChseRng
c. Attack
i. Attacks Player if in AtkRng
ii. Goes back to Chase if Player escapes AtkRng
iii. Utilizes AtkCldr to spawn a hitbox that determines if Player is hit by Attack
iii. Scripts
1. EnCtrlr
a. MelCtrlr
i. MelCldr
b. RngdCtrlr
i. bseblCtrlr
iv. Enemy Hierarchy
1. MelEn
a. Basic enemy who chases and hits Player with a melee weapon
2. RngdEn
a. Enemy who throws projectiles from a distance
3. RedEn
a. Tougher MelEn

d. Visuals
i. All sprites used were provided by the professor

e. Gameplay
i. Game Start
1. Player spawns in level, has all three Enemy types in level
ii. Objective
1. None: Level simply showcases a basic Platformer and its assets
2. Can pick up coins around the level
iii. Game Over

1. If Player health <= 0, game ends with an option to restart by pressing any key

f. UI

i. Health Bar

1. hlthImg
   a. Displays Health in green
   b. Goes down to reveal hlthGne underneath
2. hlthGne
   a. Displays Health lost

ii. Points

1. Displays Coins collected
2. Represented with a coin and number amount

iii. Game Over

1. Displays a red screen that states “Game Over! Press Any Key to Restart!”

g. **EXTRA**

i. Audio

ii. Will use project as a basis for a personal game project

SCRIPTS

PlyrCtrlr

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

public class PlyrCtrlr : MonoBehaviour
{

    public float spd;
    Rigidbody2D plyrRgdBdy;
    float inputX;

    public LayerMask wlLyr;
    public float ryLngth;
    [SerializeField]
    bool cnJmp;
    public float jmpHt;

    bool hurt;
    public float mxHlth;
    [SerializeField]
    float hlth;
    public float tmeBtwnHrt;
    float iframe;

    Animator anim;
```

```
    SpriteRenderer rend;

    [SerializeField]
    int coins;

    public Image hlthImg;
    public Text coinsTxt;

    public GameObject GmeOvrUI;
    bool gmeOvr;

    void Awake()
    {
        plyrRgdBdy = GetComponent<Rigidbody2D>();
        hlth = mxHlth;
        hurt = false;
        iframe = tmeBtwnHrt;
        anim = GetComponent<Animator>();
        rend = GetComponent<SpriteRenderer>();
        coins = 0;
        gmeOvr = false;
    }

    // Update is called once per frame
    void Update()
    {
        inputX = Input.GetAxisRaw("Horizontal");
        if(inputX != 0)
        {
            plyrRgdBdy.AddForce(Vector2.right * inputX * spd * Time.deltaTime);
        }

        rend.flipX = (inputX < 0);

        //Jump Condition
        RaycastHit2D hit = Physics2D.Raycast(transform.position, Vector2.down, ryLngth,
wlLyr);
        if(hit.collider != null)
        {
            cnJmp = true;
        }

        if(cnJmp && Input.GetKeyDown(KeyCode.Space))
        {
            plyrRgdBdy.AddForce(Vector2.up * jmpHt);
            cnJmp = false;
        }
        Debug.DrawRay(transform.position, Vector2.down * ryLngth);

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


        //Damage Test
        if(!hurt && Input.GetKeyDown(KeyCode.LeftControl))
```

```
        {
            dmg(1);
        }

        //UI
        hlthImg.fillAmount = Mathf.Lerp(hlthImg.fillAmount, hlth / mxHlth, Time.deltaTime
* 10f);
        coinsTxt.text = coins.ToString();

        //Animation
        anim.SetBool("Mvng", inputX != 0);
        anim.SetBool("CnJmp", cnJmp);
        anim.SetBool("Hrt", hurt);

        if(gmeOvr && Input.anyKeyDown)
        {
            SceneManager.LoadScene("SampleScene");
            Time.timeScale = 1f;
        }
    }

    public void dmg(float amt)
    {
        if (iframe < 0)
        {
            hlth -= amt;
            hurt = true;
            Invoke("ResetHurt", 0.2f);

            //Game Over
            if (hlth <= 0)
            {
                GameOver();

            }

            iframe = tmeBtwnHrt;
        }
    }

    private void GameOver()
    {
        gmeOvr = true;
        GmeOvrUI.SetActive(true);
        Time.timeScale = 0f;
    }

    void ResetHurt()
    {
        hurt = false;
    }

    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.gameObject.CompareTag("Coin"))
        {
            coins++;
```

```
            //Delete
            Destroy(collision.gameObject);

            /*Disable
            collision.gameObject.SetActive(false);
            */
        }
    }

    private void OnCollisionEnter2D(Collision2D collision)
    {
        //If player lands on top of enemy, do damage to enemy
        if(collision.gameObject.CompareTag("Enemy") && plyrRgdBdy.velocity.y < 0)
        {
            float bndsY =
collision.gameObject.GetComponent<SpriteRenderer>().bounds.size.y/2;

            //If player on enemy side, add force
            if(transform.position.y > collision.gameObject.transform.position.y + bndsY)
            {
                plyrRgdBdy.AddForceAtPosition(-plyrRgdBdy.velocity.normalized * jmpHt /
2f, plyrRgdBdy.position);
                collision.gameObject.GetComponent<EnCtrlr>().Damage(5f);
            }
        }
    }
}
```

EnCtrlr

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

public class EnCtrlr : MonoBehaviour
{
    public float mxHlth;
    protected float hlth;
    public Image hlthImg;

    public float spd;
    public float rnSpd;
    public float chsRng;
    public float atkRng;
    public enum enStes {move, chase, attack};
    public enStes crntSte = enStes.move;
    protected Rigidbody2D enRgdBdy;

    public LayerMask wlLyr;
    public float ryLgth;

    public int dir; //right = 1, left = -1

    protected SpriteRenderer rend;

    protected float dist;
```

```
protected PlyrCtrlr plyr;

public float tmeBtwnAtk;
protected float atkCldwn;

protected Animator anim;

private void OnEnable()
{
    hlth = mxHlth;
    //if Random.value is >= 0.5, then (?) right. Otherwise, left.
    dir = (Random.value >= 0.5f) ? 1 : -1;

    //if attack, wait X seconds till next attack
    atkCldwn = tmeBtwnAtk;
}

private void Awake()
{
    enRgdBdy = GetComponent<Rigidbody2D>();
    rend = GetComponent<SpriteRenderer>();
    plyr = FindObjectOfType<PlyrCtrlr>();

    anim = GetComponent<Animator>();
}

public virtual void Move(){}
public virtual void Chase(){}
public virtual void Attack(){}
public virtual void Damage(float amnt){}
public virtual void Die(){}

// Update is called once per frame
void Update()
{
    rend.flipX = (dir == -1);

    //States of Enemy
    switch(crntSte)
    {
        case enStes.move:
                Move();
        break;

        case enStes.chase:
                Chase();
        break;

        case enStes.attack:
                Attack();
        break;
    }

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

```
        hlthImg.fillAmount = Mathf.Lerp(hlthImg.fillAmount, hlth / mxHlth, Time.deltaTime
* 10f);
    }
}
```

CamCtrlr

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

public class CamCtrlr : 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);
    }
}
```

MelEnCtrlr

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

```
using UnityEngine;

public class MelEnCtrlr : EnCtrlr
{
    public override void Move()
    {

        dist = Vector2.Distance(transform.position, plyr.transform.position);

        RaycastHit2D hit = Physics2D.Raycast(transform.position, Vector2.right * dir,
ryLgth, wlLyr);
        RaycastHit2D hitDown = Physics2D.Raycast(transform.position, Vector2.right * dir
- Vector2.up, ryLgth, wlLyr);

        //If enemy hits wall, turn around
        if (hit.collider != null)
        {
            dir *= -1;
        }

        //If enemy hits ledge, turn around
        if (hitDown.collider == null)
        {
            //Debug.Log("Why tho");
            dir *= -1;
        }

        if (dist <= chsRng)
        {
            crntSte = enStes.chase;
        }

        Debug.DrawRay(transform.position, Vector2.right * dir * ryLgth * wlLyr);

        enRgdBdy.AddForce(Vector2.right * dir * spd * Time.deltaTime);
    }

    public override void Chase()
    {
        dist = Vector2.Distance(transform.position, plyr.transform.position);

        //If player is on left, go left. If right, go right
        if (transform.position.x > plyr.transform.position.x)
        {
            dir = -1;
        }
        else
        {
            dir = 1;
        }

        //if player outside chase range, become passive
        if(dist >= chsRng)
        {
            crntSte = enStes.move;
        }

        //if player in attack range, attack
```

```
        if (dist <= atkRng)
        {
            crntSte = enStes.attack;
        }

        //Enter run speed
        enRgdBdy.AddForce(Vector2.right * dir * rnSpd * Time.deltaTime);

    }

    public override void Attack()
    {
        if(atkCldwn <= 0)
        {
            //Debug.Log("Attack!");
            anim.SetBool("Attack", true);
            Invoke("ResetAttack", 0.1f);
            atkCldwn = tmeBtwnAtk;
        }
        else
        {
            crntSte = enStes.chase;
        }
    }

    private void ResetAttack()
    {
        anim.SetBool("Attack", false);
    }

    public override void Damage(float amnt)
    {
        hlth -= amnt;
        if (hlth <= 0) Die();
    }

    public override void Die()
    {
        //Delete
        Destroy(gameObject);

        /*Disable
        gameObject.SetActive(false);
        */
    }
}
```

MelCldr

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

public class MelCldr : MonoBehaviour
{
    public SpriteRenderer prntRnd;
    public float atk;
```

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

    }

    // Update is called once per frame
    void Update()
    {
        //if MelEn, flipX is false. If false, x = -1
        transform.localScale = new Vector3(prntRnd.flipX? -1 : 1, 1, 1);
    }

    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.gameObject.CompareTag("Player"))
        {
            collision.gameObject.GetComponent<PlyrCtrlr>().dmg(atk);
        }
    }

    private void OnTriggerStay2D(Collider2D collision)
    {
        if(collision.gameObject.CompareTag("Player"))
        {
            collision.gameObject.GetComponent<PlyrCtrlr>().dmg(atk);
        }
    }
}
```

RngdEnCtrlr

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

public class RngdEnCtrlr : EnCtrlr
{
    public GameObject bsBll;
    public override void Move()
    {

        dist = Vector2.Distance(transform.position, plyr.transform.position);

        RaycastHit2D hit = Physics2D.Raycast(transform.position, Vector2.right * dir,
ryLgth, wlLyr);
        RaycastHit2D hitDown = Physics2D.Raycast(transform.position, Vector2.right * dir
- Vector2.up, ryLgth, wlLyr);

        //If enemy hits wall, turn around
        if (hit.collider != null)
        {
            dir *= -1;
        }

        //If enemy hits ledge, turn around
```

```
        if (hitDown.collider == null)
        {
            //Debug.Log("Why tho");
            dir *= -1;
        }

        if (dist <= chsRng)
        {
            crntSte = enStes.chase;
        }

        Debug.DrawRay(transform.position, Vector2.right * dir * ryLgth * wlLyr);

        enRgdBdy.AddForce(Vector2.right * dir * spd * Time.deltaTime);
    }

    public override void Chase()
    {
        dist = Vector2.Distance(transform.position, plyr.transform.position);

        //If player is on left, go left. If right, go right
        if (transform.position.x > plyr.transform.position.x)
        {
            dir = -1;
        }
        else
        {
            dir = 1;
        }

        //if player outside chase range, become passive
        if (dist >= chsRng)
        {
            crntSte = enStes.move;
        }

        //if player in attack range, attack
        if (dist <= atkRng)
        {
            crntSte = enStes.attack;
        }

        //Enter run speed
        enRgdBdy.AddForce(Vector2.right * dir * rnSpd * Time.deltaTime);

    }

    public override void Attack()
    {
        if (atkCldwn <= 0)
        {
            //Debug.Log("Attack!");
            anim.SetBool("Attack", true);
            Vector3 dir = plyr.transform.position - transform.position;
            float angle = Mathf.Atan2(dir.y, dir.x) * Mathf.Rad2Deg - 90;
            Instantiate(bsBll, transform.position, Quaternion.AngleAxis(angle,
Vector3.forward));
```

```
            Invoke("ResetAttack", 0.1f);
            atkCldwn = tmeBtwnAtk;
        }
        else
        {
            crntSte = enStes.chase;
        }
    }

    private void ResetAttack()
    {
        anim.SetBool("Attack", false);
    }

    public override void Damage(float amnt)
    {
        hlth -= amnt;
        if (hlth <= 0) Die();
    }

    public override void Die()
    {
        //Delete
        Destroy(gameObject);

        /*Disable
        gameObject.SetActive(false);
        */
    }
}
```

bsbllCtrlr

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

public class bsbllCtrlr : MonoBehaviour
{
    public float spd;
    Rigidbody2D bsbllRgdBdy;
    public float dmg;

    private void Awake()
    {
        bsbllRgdBdy = GetComponent<Rigidbody2D>();
    }

    private void OnEnable()
    {
        bsbllRgdBdy.AddForce(transform.up * spd);
    }

    private void Disable()
    {
        gameObject.SetActive(false);
    }
```

```
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.gameObject.CompareTag("Player"))
        {
            collision.GetComponent<PlyrCtrlr>().dmg(dmg);

            //destroy baseball on hit
            Invoke("Disable", 0.001f);
        }

        if(collision.gameObject.CompareTag("Wall"))
        {
            Invoke("Disable", 0.001f);
        }
    }
    private void OnDisable()
    {
        CancelInvoke();
    }
}
```