

PROGRESS REPORT

MICROGAME #4: Tower Defense

Andrew Adame

ID: 007516100

LEGEND: COMPLETED – UNFINISHED – WIP – FIX - FIXED

GITHUB: <https://github.com/andrewadame/UnityProjectsCSE-4410/tree/master/TowerDefenseProject>

UNITY PLAY: <https://play.unity.com/mg/other/build-m8s>

1. Create new project Tower Defense Game
2. Create folders containing important assets (scripts, prefabs, animation, etc)
3. Create a basic Tower Defense Game
 - a. Design Level
 - i. Track
 1. Made a road for enemies to follow
 - ii. Waypoints
 1. Created a series of waypoints for enemies to follow
 - iii. End Goal
 1. An end goal for enemies to reach, de-spawn and damage Player health
 - b. Player
 - i. Tower Spawning
 1. Created several spawn for players to click on and spawn turrets
 2. Tower Spawn locations glow green if player can afford to spawn
 3. Tower Spawn locations glow red if player can not afford to spawn
 - ii. Tower Hierarchy
 1. Three types of Towers player can spawn
 - a. Cannon
 - b. Multi-Shot (Fires two shots at a time)
 - c. Mortar (Fires a shot that utilizes an AOE to damage enemies)
 - iii. Health
 1. Health bar that goes down every time enemy reaches a goal
 2. Utilizes provided red bar for visualization
 - c. Enemies
 - i. Movement

1. Enemies follow a set of waypoints towards the end goal at the end of the road
- ii. Enemy Hierarchy
 1. Four enemies with various different values
 - a. EnAPC
 - b. EnJeep
 - i. Faster, weaker, half amount of money earned when killed
 - c. EnTnk
 - i. Slower, tougher, twice as much money earned when killed
 - ii. Twice as much player health lost if reaches end goal
 - d. EnTruck
 - i. Somewhere between EnAPC and EnTnk
- d. Visuals
 - i. Sprites provided by instructor
 - ii. Animation
 1. Animation Script
 - a. Called "boomCtrlr"
 2. Muzzle flashes when Towers fire projectiles
 - a. VISUAL ERROR
 - i. Muzzle flash on Multi-Shot only plays on one barrel instead of both
 3. Enemies explode on death
- e. Audio
 - i. Background music plays on game start
 - ii. Towers produce sounds when firing
- f. Gameplay
 - i. Game Script
 1. Called "GmeCtrlr"
 - ii. Start
 1. Background music begins to play
 2. Random enemies begin to spawn and follow waypoint
 - iii. Health Tracking
 1. Default amount of health is 10
 - iv. Objective
 1. Survive as long as possible
 - v. Point System
 1. Player earns money upon destroying an enemy
 - a. EnAPC earns 10

- b. EnJeep earns 5
 - c. EnTnk earns 20
 - d. EnTruck earns 15
 - vi. Enemy Spawn System
 - 1. Enemy spawns at a steady rate
 - vii. Game Over
 - 1. Game ends upon health bar is empty
 - a. RESTART ERROR
 - i. When player restarts, game reloads scene but enemies no longer follow waypoints
- g. UI
 - i. Player Healthbar
 - 1. Represented as a red bar
 - 2. Disappears one by one with every damage taken
 - ii. Game Over
 - 1. Stops game with screen that displays the message "Game Over! Press Any Key to Restart!"
- h. EXTRA
 - i. More Audio
 - ii. Expanded soundtrack for background music
 - iii. More Levels
 - iv. Original sprites
 - v. Animation overhaul
 - vi. Upgrades
 - 1. Turret improvements?
 - vii. Larger Hierarchy of Enemies
 - viii. Turret Health?
 - 1. Enemies could attack back at Turrets?

SCRIPTS

GmeCtrlr

```

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

public class GmeCtrlr : MonoBehaviour
{
    public BseTwr twr;
    public BseTwr[] twrs;

    public Transform[] waypoints;

```

```

public float mxHlth;

[SerializeField]
float hlth;
public float LrpSpd;

public Image hlthImg;
[HideInInspector]
public float crntTwrcst;

public Text mnyTxt;
public float mny;

public float tmeBtwSpwnLw;
public float tmeBtwSpwnHi;

float cools;
public GameObject spwnPos;
public GameObject[] Enems;

public bool gmeOvr;
public GameObject gmeOvrUI;

private void Awake()
{
    hlth = mxHlth;
    hlthImg.fillAmount = hlth / mxHlth;
    //hlthImg.fillAmount = Mathf.Lerp(hlthImg.fillAmount, hlth / mxHlth, LrpSpd *
Time.deltaTime);
    UpdateTwrcst(0);
    gmeOvr = false;
}

public void UpdateTwrcst(int i)
{
    twrcst = twrcsts[i];
    crntTwrcst = twrcsts[i].cost;
}

void SpwnEnemy()
{
    Instantiate(Enems[Random.Range(0, Enems.Length)], spwnPos.transform.position,
Quaternion.Euler(0,0,-90));
    cools = Random.Range(tmeBtwSpwnLw, tmeBtwSpwnHi);
}

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

// Update is called once per frame
void Update()
{
    mnyTxt.text = "MONEY: " + mny.ToString();
}

```

```

        if (!gmeOvr)
        {
            if (cools > 0)
            {
                cools -= Time.deltaTime;
            }
            else
            {
                SpwnEnmy();
            }
        }

        if(gmeOvr && Input.anyKeyDown)
        {
            SceneManager.LoadScene(SceneManager.GetActiveScene().name);
        }
    }

    public void TakeDamage(float dmg)
    {
        hlth -= dmg;
        if(hlth <= 0)
        {
            gmeOvr = true;
            gmeOvrUI.SetActive(true);
            Time.timeScale = 0f;
        }
        hlthImg.fillAmount = hlth / mxHlth;
    }

    public void GveMny(float amt)
    {
        mny += amt;
    }
}

```

TwrSpwnCtrlr

```

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

public class TwrSpwnCtrlr : MonoBehaviour
{
    GmeCtrlr cont;

    SpriteRenderer rend;

    private void Awake()
    {
        cont = FindObjectOfType<GmeCtrlr>();
        rend = GetComponent<SpriteRenderer>();
    }
    private void OnMouseDown()
    {
        if (cont.mny >= cont.crntTwrcst)

```

```

        {
            //Debug.Log("Spawn Tower");
            cont.GveMny(-cont.crntTwrCst);
            Instantiate(cont.twr, transform.position, transform.rotation);
            gameObject.SetActive(false);
        }
    }

    private void OnMouseOver()
    {
        if(cont.mny >= cont.crntTwrCst)
        {
            rend.color = Color.green;
        }
        else
        {
            rend.color = Color.red;
        }
    }

    private void OnMouseExit()
    {
        rend.color = Color.white;
    }

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

    // Update is called once per frame
    void Update()
    {
    }
}

```

BseTwr

```

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

public class BseTwr : MonoBehaviour
{
    public float StSpd;
    public float CnRot;
    public Transform trgt;
    public float atkRng;

    public Transform child;

    protected float cldwn;
    public GameObject blt;
}

```

```

public GameObject[] bltSpwnPos;

public float cost;

public GameObject flsh;

protected AudioSource srce;

private void Awake()
{
    srce = GetComponent<AudioSource>();
}

private void OnEnable()
{
    GetComponent<CircleCollider2D>().radius = atkRng;
    clwn = StSpd;
}

private void OnTriggerEnter2D(Collider2D collision)
{
    if (collision.gameObject.CompareTag("Enemy") && trgt == null)
    {
        trgt = collision.transform;
    }
}

private void OnTriggerStay2D(Collider2D collision)
{
    if (collision.gameObject.CompareTag("Enemy") && trgt == null)
    {
        trgt = collision.transform;
    }
}

private void OnTriggerExit2D(Collider2D collision)
{
    if (collision.gameObject.CompareTag("Enemy") && trgt == collision.transform)
    {
        trgt = null;
    }
}

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

// Update is called once per frame
void Update()
{
    if (trgt != null)
    {
        //Debug.Log(trgt.gameObject);
        Vector3 dir = trgt.transform.position - transform.position;
        float angle = Mathf.Atan2(dir.y, dir.x) * Mathf.Rad2Deg - 90;
    }
}

```

```

        transform.rotation = Quaternion.Lerp(transform.rotation,
Quaternion.AngleAxis(angle, Vector3.forward), Time.deltaTime * CnRot);
        if (cldwn > 0)
        {
            cldwn -= Time.deltaTime;
        }
        else
        {
            Shoot();
        }
    }
}

public virtual void Shoot()
{
}

private void LateUpdate()
{
    child.transform.rotation = Quaternion.identity;
}
}

```

BltCtrlr

```

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

public class BltCtrlr : MonoBehaviour
{
    public float spd;
    Rigidbody2D bltRgdBdy;
    public float dmg;

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

    private void OnEnable()
    {
        bltRgdBdy.AddForce(transform.up * spd);
        Invoke("Disable", 4f);
    }

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

    // Update is called once per frame
    void Update()
    {
    }
}

```



```

{
}

private void Disable()
{
    Destroy(gameObject);
}

private void OnTriggerEnter2D(Collider2D collision)
{
    if (collision.gameObject.CompareTag("Enemy"))
    {
        collision.GetComponent<EnCtrlr>().TkeDmg(dmg);

        //destroy bullet on hit
        Destroy(gameObject);
    }
}
}

```

EnCtrlr

```

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

public class EnCtrlr : MonoBehaviour
{
    Rigidbody2D EnRgdBdy;
    public float spd;
    public float mxHlth;

    [SerializeField]
    float hlth;

    Transform trgt;
    public int crntWpyt;
    GmeCtrlr cont;
    public float rotSpd;

    float dist;

    bool canMove = true;

    public float dmg;

    public float drpMny;

    public GameObject boom;

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

```

```

}

private void OnEnable()
{
    hlth = mxHlth;

    crntWypt = 0;
    trgt = cont.waypoints[crntWypt];
}

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

// Update is called once per frame
void Update()
{
    Vector3 dir = trgt.transform.position - transform.position;
    float angle = Mathf.Atan2(dir.y, dir.x) * Mathf.Rad2Deg - 90;
    transform.rotation = Quaternion.Lerp(transform.rotation,
Quaternion.AngleAxis(angle, Vector3.forward), Time.deltaTime * rotSpd);

    if (canMove)
    {
        EnRgdBdy.AddForce(transform.up * spd * Time.deltaTime);
    }

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

    if (dist <= 0.05f)
    {
        if (crntWypt < cont.waypoints.Length - 1)    //still have waypoints, not last
waypoint
        {
            canMove = false;
            Invoke("CanMove", 0.5f);
            crntWypt++;
            trgt = cont.waypoints[crntWypt];
        }
        else    //last waypoint
        {
            cont.TakeDamage(dmg);
            Destroy(gameObject);
        }
    }
}

void CanMove()
{
    canMove = true;
}

public void TkeDmg(float dmge)
{

```

```

        hlth -= dmg;
        if(hlth <= 0)
        {
            cont.GveMny(drpMny);
            Instantiate(boom, transform.position, Quaternion.identity);
            Destroy(gameObject);
        }
        //Debug.Log("Enemy Take Damage");
    }
}

```

CanCtrlr

```

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

public class CanCtrlr : BseTwr
{
    public override void Shoot()
    {
        Instantiate(blt, bltSpwnPos[0].transform.position, transform.rotation);
        srce.Play();
        Instantiate(flsh, bltSpwnPos[0].transform.position, transform.rotation);
        cldwn = StSpd;
        base.Shoot();
    }
}

```

ShotCtrlr

```

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

public class ShotCtrlr : BseTwr
{
    public override void Shoot()
    {
        for (int i = 0; i < bltSpwnPos.Length; i++)
        {
            for (int j = 0; j < 5; j++)
            {
                Instantiate(blt, bltSpwnPos[i].transform.position, transform.rotation *
Quaternion.Euler(0, 0, (i*5)-15f));
            }
        }

        srce.Play();
        Instantiate(flsh, bltSpwnPos[0].transform.position, transform.rotation);
        cldwn = StSpd;
        base.Shoot();
    }
}

```

MoCtrlr

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

public class MoCtrlr : BseTwr
{
    public override void Shoot()
    {
        Instantiate(blt, bltSpwnPos[0].transform.position, transform.rotation);
        srce.Play();
        Instantiate(flsh, bltSpwnPos[0].transform.position, transform.rotation);
        cldwn = StSpd;
        base.Shoot();
    }
}
```

MoShellCtrlr

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

public class MoShellCtrlr : MonoBehaviour
{
    public float spd;
    Rigidbody2D bltRgdBdy;
    public float dmg;

    public float rad;
    public LayerMask enMsk;

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

    private void OnEnable()
    {
        bltRgdBdy.AddForce(transform.up * spd);
        Invoke("Disable", 4f);
    }

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

    // Update is called once per frame
    void Update()
    {
    }
}
```

```

private void Disable()
{
    Destroy(gameObject);
}

private void OnTriggerEnter2D(Collider2D collision)
{
    if (collision.gameObject.CompareTag("Enemy"))
    {
        Collider2D[] hit = Physics2D.OverlapCircleAll(transform.position, rad,
enMsk);
        foreach(Collider2D col in hit)
        {
            col.GetComponent<EnCtrlr>().TkeDmg(dmg);
        }
    }
}

private void OnDrawGizmos()
{
    Gizmos.DrawWireSphere(transform.position, rad);
}
}

```

boomCtrlr

```

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

public class boomCtrlr : MonoBehaviour
{
    public AnimationClip clip;

    private void OnEnable()
    {
        Invoke("Disable", clip.length);
    }

    private void Disable()
    {
        Destroy(gameObject);
    }
}

```