Creative Making

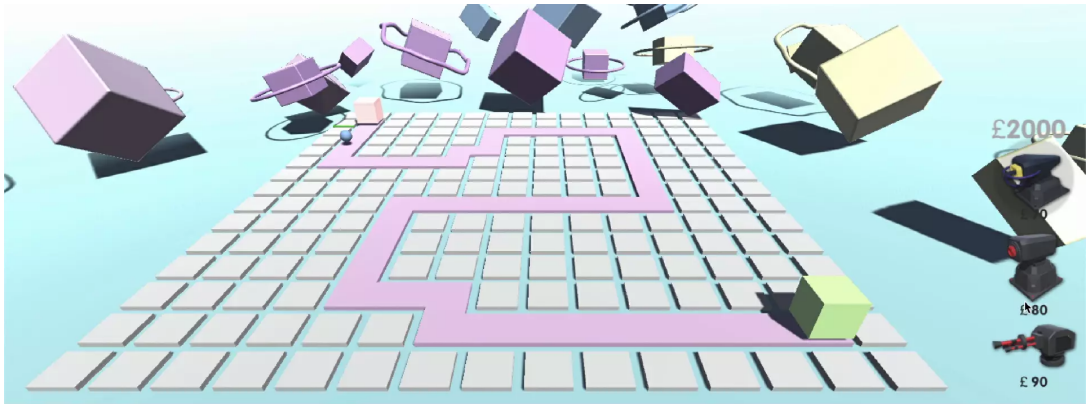
Advanced Visualisation and Computational Environments

Game Name: "CUBE UNIVERSE"

Pinsi Wang

Introduction

In this project, I decided to make a 3D tower defense game with Unity and use an Arduino board to control part of the game.



Inspiration and Game rules

In this 3D tower defense game, we are in a huge cube universe, which is full of harmony and peace. Suddenly, however, a group of mysterious spheres emerged from unknown dimensions, threatening our universe.

The goal of these spheres is to invade our universe and take away our freedom and peace. As the defenders of the cube universe, we decided to bravely resist the invaders. Players can choose three kinds of turrets on the right to attack the sphere, and the turrets can be upgraded. After upgrading, the attack strength and attack range of the turrets will be enhanced.

Project Download & video Link:

Zip: https://drive.google.com/file/d/1zB_jHBt6LnPRF0ieXj35pwQcQv09AgU0/view?usp=sharing
Folder(If the link above doesn't work, click this one): https://drive.google.com/drive/folders/1mBAwSp5SRKiS9tZPCfXxXgrtv5GHvpCq?usp=sharing
Video: https://youtu.be/1yM6wdxmteY

Coding Parts(Step by step):

1. In the Hierarchy layer, create a huge Plane as the ground.
2. Use cubes to set a 15*15 map.

3. Use cubes to set a start and end point.
4. Setting camera(All coding part of MainCamera Script):

```
public class ViewController : MonoBehaviour
{
    public float speed = 25;
    public float mouseSpeed = 600;
    // Update is called once per frame
    void Update()
    {
        float h = Input.GetAxis("Horizontal");
        float v = Input.GetAxis("Vertical");
        float mouse = Input.GetAxis("Mouse ScrollWheel");/
        transform.Translate(new Vector3(h*speed, -mouse*mouseSpeed, v*spee
d) * Time.deltaTime,Space.World);
    }
}
```

5. Setting enemy's path of action:

```
public class Waypoints : MonoBehaviour
{
    public static Transform[] positions;
    private void Awake()
    {

        positions = new Transform[transform.childCount];
        for (int i=0;i < positions.Length;i++)
        {
            positions[i] = transform.GetChild(i);
        }
    }
}
```

6. Setting Enemies and control enemies' moving:

```
public class Enemy : MonoBehaviour
{
    public float speed = 10;//Enemy's speed
    private Transform[] positions;
    private int index = 0;

    void Start()
    {
        positions = Waypoints.positions;//get enemy's path
    }
    void Update()
    {
        Move();
    }
    void Move()
```

```
    {
        if (index > positions.Length - 1) return;
        transform.Translate((positions[index].position - transform.positio
n).normalized * Time.deltaTime * speed);
        if( Vector3.Distance( positions[index].position,transform.positio
n) < 0.2f)
        {
            index++;
        }
    }
}

```

7. Create an enemy incubator to manage the generation of enemies. First create 4 types of enemies, Enemy2, Enemy3, and Enemy4 prefabs with different color materials, and create a separate script Wave for generating the attributes required for each wave:

```
[System.Serializable]
{
    public GameObject enemyPrefab;
    public int count;
    public float rate;
}
```

8. Improve enemies. Make an empty object generator GameManager and put all management related scripts on it.

In GameManager Script:

```
public class EnemySpawner : MonoBehaviour
{
    public static int CountEnemyAlive = 0;//The current number of enemies
 alive, the default is 0
    public Wave[] waves;
    public Transform START;
    public float waveRate = 3;

    void Start()
    {
        StartCoroutine("SpawnEnemy");
    }

    public void Stop()
    {
        StopCoroutine("SpawnEnemy");
    }


 SpawnEnemy()
    {
        foreach(Wave wave in waves)
        {
```

```
            for(int i = 0; i < wave.count; i++)
            {
                GameObject.Instantiate(wave.enemyPrefab, START.position, Q
uaternion.identity);
                CountEnemyAlive++;
                if (i != wave.count - 1)
                {
                    yield return new WaitForSeconds(wave.rate);
                }
            }
            while (CountEnemyAlive > 0)
            {
                yield return 0;
            }
            yield return new WaitForSeconds(waveRate);
        }
        while(CountEnemyAlive > 0)
        {
            yield return 0;
        }
    }
}

```

9. Create prefabs of three kinds of turrets and import resources.
10. Create UI for turret selection in Canvas.
11. Create the data class of the turret, save the data related to the turret, and create the script TurretData:

```
[System.Serializable]
public class TurretData
{
    public GameObject turretPrefab;
    public int cost;
    public GameObject turretUpgradedPrefab;
    public int costUpgraded;
    public TurretType type;
}

public enum TurretType
{
    LaserTurret,
    MissileTurret,
    StandardTurret,
}
```

12. Listen to the event of turret selection, and create another BuildManager script in GameManager. (When testing, I set selectedTurredData to public, so that I can see whether there is data when I select the UI fort in the Inspector panel)
13. Detect which Cube the mouse clicks on:

```
public class BuildManager : MonoBehaviour
```

```
{
    public TurretData laserTurretData;
    public TurretData standardTurretData;
    private TurretData selectedTurredData;
    void Update()
    {
        if (Input.GetMouseButtonDown(0))
        {
            if (EventSystem.current.IsPointerOverGameObject() == false)
            {

                Ray ray = Camera.main.ScreenPointToRay(Input.mousePositio
n);
                RaycastHit hit;
                bool isCollider = Physics.Raycast(ray,out hit, 1000, Layer
Mask.GetMask("MapCube"));
                if (isCollider)
                {
                    //TODO creat turret
                }
            }
        }
    }

    public void OnLaserSelected(bool isOn)
    {
        if (isOn)
        {
            selectedTurredData = laserTurretData;
        }
    }
    public void OnMissileSelected(bool isOn)
    {
        if (isOn)
        {
            selectedTurredData = missileTurretData;
        }
    }
    public void OnStandardSelected(bool isOn)
    {
        if (isOn)
        {
            selectedTurredData = standardTurretData;
        }
    }

}
```

14. Check if it is enough to create a turret.

```
public class MapCube : MonoBehaviour
{
    [HideInInspector]
    public GameObject turretGo;
    public void BuildTurret(TurretData turretData)
    {
        //TODO
    }
}
```

15. money management

```
public class BuildManager : MonoBehaviour
{
    public TurretData laserTurretData;
    public TurretData missileTurretData;
    public TurretData standardTurretData;
    private TurretData selectedTurredData;
    private int money = 1000;
    public Text moneyText;
    public Animator moneyAnimator;

    void ChangeMoney(int change=0)
    {
        money += change;
        moneyText.text = "$ " + money;
    }

    void Update()
    {
        if (Input.GetMouseButtonDown(0))
        {
            if (EventSystem.current.IsPointerOverGameObject() == false)
            {
                Ray ray = Camera.main.ScreenPointToRay(Input.mousePositio
n);
                RaycastHit hit;
                bool isCollider = Physics.Raycast(ray,out hit, 1000, Layer
Mask.GetMask("MapCube"));
                if (isCollider)
                {
                    MapCube mapCube = hit.collider.GetComponent<MapCube>
();
                    if(mapCube.turretGo == null && selectedTurredData != n
ull)
                    {
                        if (money > selectedTurredData.cost)
                        {
                            money -= selectedTurretData.cost;
                            mapCube.BuildTurret(selectedTurredData);
```

```
                        }
                        else//notice money is not enough
                        {
                            moneyAnimator.SetTrigger("Flicker");
                        }
                    }
                    else
                    {
                        //TODO manage upgrade
                    }
                }
            }
        }
    }
    public void OnLaserSelected(bool isOn)
    {
        if (isOn)
        {
            selectedTurredData = laserTurretData;
        }
    }
    public void OnMissileSelected(bool isOn)
    {
        if (isOn)
        {
            selectedTurredData = missileTurretData;
        }
    }
    public void OnStandardSelected(bool isOn)
    {
        if (isOn)
        {
            selectedTurredData = standardTurretData;
        }
    }
}
```

16. Display the prompt effect of no money.
17. create effects:

```
public class MapCube : MonoBehaviour
{
    [HideInInspector]
    public GameObject turretGo;
    public GameObject buildEffect;

    public void BuildTurret(TurretData turretData)
    {
        turretGo = GameObject.Instantiate(turretData.turretPrefab, transfo
rm.position, Quaternion.identity);
```

```
        GameObject effect = GameObject.Instantiate(buildEffect, transform.
position, Quaternion.identity);
        Destroy(effect, 1);
    }
}
```

18. Added trigger to detect enemies near turrets. (In order to save performance, I select Project Settings —Physics in Edit, and I can set the trigger between the lower layers. Turret only collides with Enemy.)
19. Control the turret to fire bullets, and add attack interval time.
20. control bullet flight.

```
public class Bullet : MonoBehaviour
{
    public int damage = 50;
    public float speed = 40;
    private Transform target;
    public GameObject explosionEffectPrefab;

    public void SetTarget(Transform _target)
    {
        this.target = _target;
    }
    private void Update()
    {
        if(target == null)
        {
            Die();
            return;
        }
        transform.LookAt(target.position);
        transform.Translate(Vector3.forward * speed * Time.deltaTime);
    }
    private void OnTriggerEnter(Collider other)
    {
        if(other.tag == "Enemy")
        {
            other.GetComponent<Enemy>().TakeDamage(damage);
            Die();
        }
    }
    void Die()
    {
        GameObject effect = GameObject.Instantiate(explosionEffectPrefab, tr
        Destroy(effect, 1);
        Destroy(this.gameObject);
    }
}
```

21. Control the collision between the bullet and the enemy, so that the bullet explodes when it hits the enemy. (All code part of Enemy Script:)

```
public class Enemy : MonoBehaviour
{
    public float speed = 10; // Enemy movement speed
    public float hp = 150; // Enemy health points
    private float totalHp; // Total health points of the enemy
    private Transform[] positions; // Waypoint positions for enemy movemen
t
    private int index = 0; // Current waypoint position index
    public GameObject explosionEffect; // Explosion effect prefab
    private Slider hpSlider; // Slider for displaying enemy health

    void Start()
    {
        positions = Waypoints.positions; // Get the waypoint positions fro
m the Waypoints class
        totalHp = hp; // Set the total health points to the initial health
points
        hpSlider = GetComponentInChildren<Slider>(); // Get the slider com
ponent for health display
    }

    void Update()
    {
        Move(); // Move the enemy
    }

    void Move()
    {
        // Move towards the current waypoint position
        transform.Translate((positions[index].position - transform.positio
n).normalized * Time.deltaTime * speed);

        // Check if the enemy has reached the current waypoint
        if (Vector3.Distance(positions[index].position, transform.positio
n) < 0.2f)
        {
            index++; // Move to the next waypoint position
        }

        // Check if the enemy has reached the end of the waypoints
        if (index > positions.Length - 1)
        {
            ReachDestination(); // Enemy has reached the destination, trig
ger game over
        }
    }
```

```
40
41      // Called when the enemy reaches the destination
42      void ReachDestination()
43      {
44          GameManager.Instance.Faild();
45          GameObject.Destroy(this.gameObject); // Destroy the enemy game obj
ect
46      }
47
48      private void OnDestroy()
49      {
50          EnemySpawner.CountEnemyAlive--; // Decrement the count of alive en
emies in the EnemySpawner
51      }
52
53      // Called to apply damage to the enemy
54      public void TakeDamage(float damage)
55      {
56          if (hp <= 0) return; // If the enemy is already dead, return
57
58          hp -= damage; // Decrease the enemy's health by the damage amount
59          hpSlider.value = (float)hp / totalHp; // Update the health slider
 value
60
61          if (hp <= 0)
62          {
63              Die(); // Enemy has no health remaining, trigger death
64          }
65      }
66
67      // Called when the enemy dies
68      void Die()
69      {
70          // Instantiate the explosion effect at the enemy's position and ro
tation
71          GameObject effect = GameObject.Instantiate(explosionEffect, transf
orm.position, transform.rotation);
72
73          // Destroy the explosion effect after 1.5 seconds
74          Destroy(effect, 1.5f);
75
76          // Destroy the enemy game object
77          Destroy(this.gameObject);
78      }
79  }
```

22. Add explosion effects, modify the collision detection method, and create a Particle System named ExplosionEffect effects.

23. When repairing the destruction of the enemy, the bullet will also be destroyed. In the Update method of the Bullet script, it is judged whether to destroy itself.
24. Fixed removing turret references to enemies when the enemy is destroyed.
25. Control the enemy's destruction explosion.
26. Complete the destruction explosion effect of other enemies.
27. Control the turret to point at the enemy to attack.
28. Turns red when the map cube is selected.(All code part of MapCube Script:)

```
public class MapCube : MonoBehaviour
{
    [HideInInspector]
    public GameObject turretGo; // Reference to the instantiated turret game object
    [HideInInspector]
    public TurretData turretData; // Data for the current turret
    [HideInInspector]
    public bool isUpgraded = false; // Flag indicating if the turret is upgraded
    public GameObject buildEffect; // Build effect prefab
    private Renderer renderer; // Renderer component of the map cube

    private void Start()
    {
        renderer = GetComponent<Renderer>(); // Get the Renderer component
    }

    public void BuildTurret(TurretData turretData)
    {
        this.turretData = turretData; // Set the turret data
        isUpgraded = false; // Reset the upgrade flag
        turretGo = GameObject.Instantiate(turretData.turretPrefab, transform.position, Quaternion.identity); // Instantiate the turret prefab
        GameObject effect = GameObject.Instantiate(buildEffect, transform.position, Quaternion.identity); // Instantiate the build effect
        Destroy(effect, 1); // Destroy the build effect after 1 second
    }

    public void UpgradeTurret()
    {
        if (isUpgraded == true) return; // If the turret is already upgraded, return
        Destroy(turretGo); // Destroy the current turret
        turretGo = GameObject.Instantiate(turretData.turretUpgradedPrefab, transform.position, Quaternion.identity); // Instantiate the upgraded turret prefab
        isUpgraded = true; // Set the upgrade flag
        GameObject effect = GameObject.Instantiate(buildEffect, transform.position, Quaternion.identity); // Instantiate the build effect
        Destroy(effect, 1); // Destroy the build effect after 1 second
```

```
    }

    public void DestroyTurret()
    {
        Destroy(turretGo); // Destroy the current turret
        isUpgraded = false; // Reset the upgrade flag
        turretGo = null; // Reset the turret reference
        turretData = null; // Reset the turret data
        GameObject effect = GameObject.Instantiate(buildEffect, transform.position, Quaternion.identity); // Instantiate the build effect
        Destroy(effect, 1); // Destroy the build effect after 1 second
    }

    private void OnMouseEnter()
    {
        if (turretGo == null && EventSystem.current.IsPointerOverGameObject() == false) // Check if the map cube is empty and the mouse is not over any UI element
        {
            renderer.material.color = Color.green; // Change the color of the map cube to green
        }
    }

    private void OnMouseExit()
    {
        renderer.material.color = Color.white; // Change the color of the map cube back to white
    }
}

```

29. Fixed the trigger effect of the turret.
30. Added health bar display for enemies.
31. Create UI buttons for turret upgrades.
32. Add animation to the button and optimize the button display.
33. Controls the creation of methods to show and hide the escalation panel.(All coding part of BuildManager Script:)

```
public class BuildManager : MonoBehaviour
{
    public TurretData laserTurretData; // Data for laser turret
    public TurretData missileTurretData; // Data for missile turret
    public TurretData standardTurretData; // Data for standard turret
    private TurretData selectedTurredData; // Currently selected turret data
    public Text moneyText; // Text object for displaying money amount
    public Animator moneyAnimator; // Animator for money animation
    private int money = 1000; // Amount of money
    public GameObject upgradeCanvas; // Canvas for turret upgrade UI
```

```
    public Button buttonUpgrade; // Button for upgrading turret
    private MapCube selectedMapCube; // Selected map cube (turret positio
n)
    private Animator upgradeCanvasAnimator; // Animator for upgrade canvas
animation

    void ChangeMoney(int change = 0)
    {
        money += change;
        moneyText.text = "$ " + money;
    }

    private void Start()
    {
        upgradeCanvasAnimator = upgradeCanvas.GetComponent<Animator>();
    }

    void Update()
    {
        if (Input.GetMouseButtonDown(0))
        {
            if (EventSystem.current.IsPointerOverGameObject() == false) //
Check if the mouse is not over any UI element
            {
                Ray ray = Camera.main.ScreenPointToRay(Input.mousePositio
n);
                RaycastHit hit;
                bool isCollider = Physics.Raycast(ray, out hit, 1000, Laye
rMask.GetMask("MapCube")); // Check if the raycast hit a map cube
                if (isCollider)
                {
                    MapCube mapCube = hit.collider.GetComponent<MapCube>
(); // Get the MapCube component of the hit cube
                    if (mapCube.turretGo == null && selectedTurredData !=
 null) // If the map cube is empty and there is a selected turret data
                    {
                        if (money > selectedTurredData.cost) // Check if t
here is enough money to build the selected turret
                        {
                            ChangeMoney(-selectedTurredData.cost); // Subt
ract the cost of the turret from the money
                            mapCube.BuildTurret(selectedTurredData); // Bu
ild the selected turret on the map cube
                        }
                        else // If there is not enough money to build the
 turret
                        {
                            moneyAnimator.SetTrigger("Flicker"); // Trigge
r the money animation to indicate insufficient funds
```

```
                        }
                    }
                    else if (mapCube.turretGo != null) // If the map cube
 already has a turret
                    {
                        if (mapCube.turretGo == selectedMapCube && upgrade
Canvas.activeInHierarchy) // If the selected map cube is the same as the p
revious one and the upgrade canvas is active
                        {
                            StartCoroutine("HideUpgradeUI"); // Hide the u
pgrade UI
                        }
                        else // If the selected map cube is different or t
he upgrade canvas is not active
                        {
                            ShowUpgradeUI(mapCube.transform.position, mapC
ube.isUpgraded); // Show the upgrade UI at the position of the map cube
                        }
                        selectedMapCube = mapCube; // Set the selected map
cube to the current cube
                    }
                }
            }
        }
    }

    public void OnLaserSelected(bool isOn)
    {
        if (isOn)
        {
            selectedTurredData = laserTurretData; // Set the selected turr
et data to the laser turret data
        }
    }

    public void OnMissileSelected(bool isOn)
    {
        if (isOn)
        {
            selectedTurredData = missileTurretData; // Set the selected tu
rret data to the missile turret data
        }
    }

    public void OnStandardSelected(bool isOn)
    {
        if (isOn)

```

34. Controlling the Showing and Hiding of the Upgrade Panel.
35. Finishing the animation of upgrade panel.
36. Finishing the click events of the upgrade button and demolition button. Add UpgradeTurret upgrade method in MapCube.
37. Control upgrade and dismantling treatment of turrets.
38. Modify construction special effects (add special effects when upgrading and demolishing)
39. Upgrade prefabs that create standard turrets.
40. Developed and completed the prefab of the atomic bomb turret.
41. Added laser turrets.
42. Use LineRenderer to display laser light.
43. Added laser damage and damage effects.
44. Added an upgraded version of the laser tower prefab.
45. Added gold cost when leveling up. In the OnUpgradeButtonDown method in the BuildManager script, reduce the money. If it is not enough, use the flashing money effect directly.
46. Design the UI interface at the end of the game.
47. Controls the display of the failure interface.

Because EnemySpawner and GameManager are on the same object GameManager, the generation of enemies is controlled in GameManager.

(All code part of GameManager Script:)

```
public class GameManager : MonoBehaviour
{
    public GameObject end UI; // Ending UI page
    public Text endMessage; // Text component for displaying the end messa
ge
    public static GameManager Instance; // Singleton instance of the GameM
anager
    private EnemySpawner enemySpawner; // Reference to the EnemySpawner co
mponent

    private void Awake()
    {
        Instance = this; // Set the instance to this GameManager
        enemySpawner = GetComponent<EnemySpawner>(); // Get the EnemySpawn
er component
    }

    public void Win()
    {
        endUI.SetActive(true); // Activate the end UI
        endMessage.text = "Victory"; // Set the end message text to "Victo
ry"
    }

    public void Faild()
    {
        enemySpawner.Stop(); // Stop creating enemies
        endUI.SetActive(true); // Activate the end UI
```

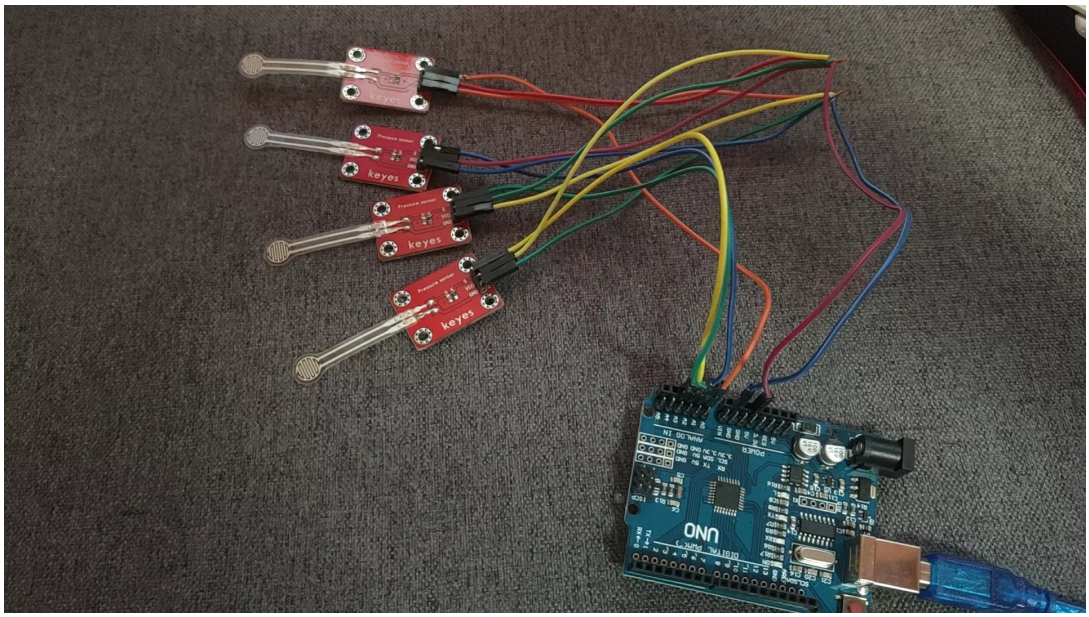
```
        endMessage.text = "Failed"; // Set the end message text to "Faile
d"
    }

    public void OnButtonRetry()
    {
        endUI.SetActive(false); // Deactivate the end UI
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
 // Reload the current scene
    }

    public void OnButtonMenu()
    {
        SceneManager.LoadScene("MainMenu"); // Load the "MainMenu" scene
    }
}

```

48. Added win page and replay, menu button clicks.
49. Develop menu scene. Adjust camera position.
50. Controls click event handling for the start and exit buttons.

(All coding part of GameMenu:)

```
public class GameMenu : MonoBehaviour
{
    public void OnStartGame()
    {
        SceneManager.LoadScene("Tower Defence");
    }
    public void OnExitGame()
    {
#if UNITY_EDITOR
        UnityEditor.EditorApplication.isPlaying = false;
#else
        Application.Quit();
#endif
    }
}
```

51. Connected with Arduino board




```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(A0, INPUT);
  pinMode(A1, INPUT);
  pinMode(A2, INPUT);
  pinMode(A3, INPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  Serial.print(analogRead(A0));
  Serial.print(",");
  Serial.print(analogRead(A1));
  Serial.print(",");
  Serial.print(analogRead(A2));
  Serial.print(",");
  Serial.println(analogRead(A3));
}
```

Conclusion

In this project, I learned a lot about Unity and learned how to combine Arduino and Unity. For example, learn to use the Unity engine for game development, including basic operations such as scene creation, object management, and collision detection, AI and path planning,etc. In the future, I hope that I can use an Arduino board with a pressure sensor or other sensors to control the construction and upgrade of the turret.