Development log

-Maxim Gerashchenko

I was tasked with setting up the initial project and base RTS elements.

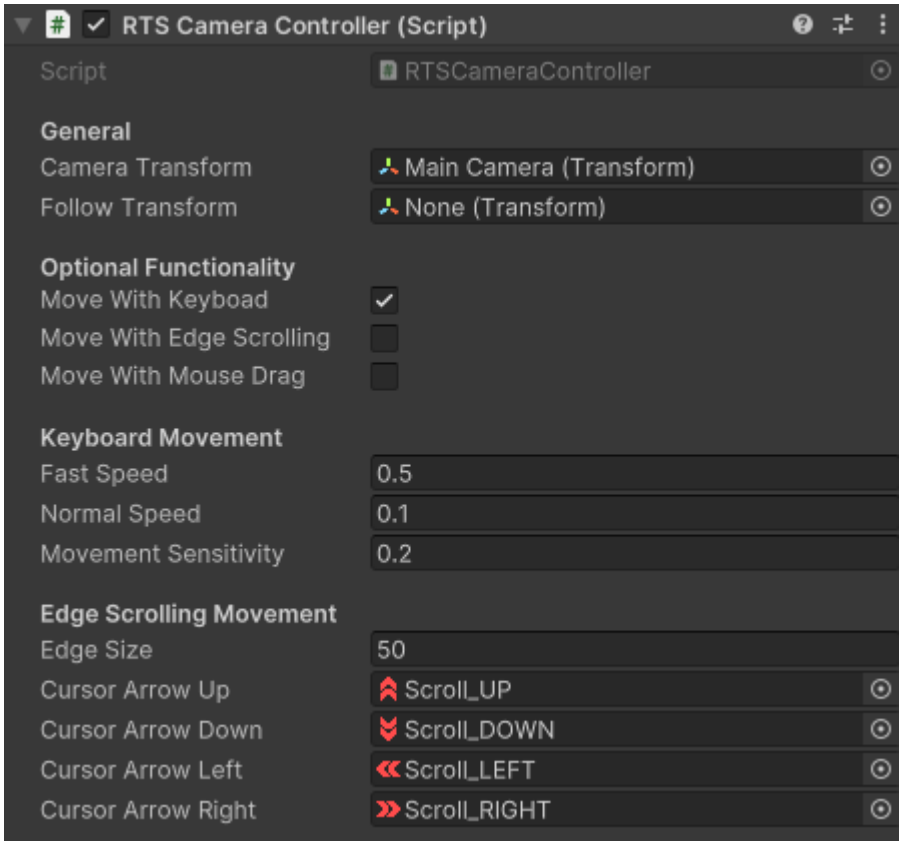
So far I have implemented the following:

- RTSCameraController.cs
- UnitSelectionManager.cs (keeps track of all units and selected units and handles the process of selecting them and enabling them to move)
- UnitMovement.cs (Handles unit movement)
- TargetSelectionManager (Assigns targets to units that are selected when something is right clicked ex. Recourse, enemy)
- Created Github repository with project base

Currently working on (To be implemented)

- Recourse Gathering System
- Building System

RTS Camera



I created the RTS Camera script which allows the camera to look over the scene in a rts style. As you can see in the inspector the camera has three ways of functioning,

Move with Keyboard which allows the player to move the camera with WASD or Arrow keys,

Edge scrolling which allows the player to move the camera by moving the cursor to the edge of the screen and changes to cursor to the arros scene in the inspector

And the final one which allows the player to move the camera by clicking and dragging on the ground to move the camera.

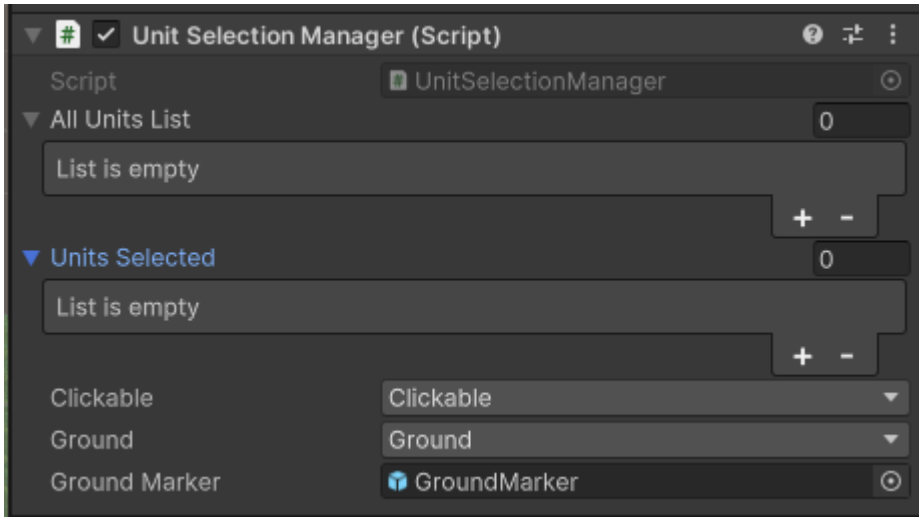
```
1 reference
void HandleCameraMovement()
{
    // Mouse Drag
    if (moveWithMouseDrag)
    {
        HandleMouseDragInput();
    }

    // Keyboard Control
    if (moveWithKeyboad)
    {
        if (Input.GetKey(KeyCode.LeftCommand))
        {
            movementSpeed = fastSpeed;
        }
        else
        {
            movementSpeed = normalSpeed;
        }

        if (Input.GetKey(KeyCode.W) || Input.GetKey(KeyCode.UpArrow))
        {
            newPosition += (transform.forward * movementSpeed);
        }
        if (Input.GetKey(KeyCode.S) || Input.GetKey(KeyCode.DownArrow))
        {
            newPosition += (transform.forward * -movementSpeed);
        }
        if (Input.GetKey(KeyCode.D) || Input.GetKey(KeyCode.RightArrow))
        {
            newPosition += (transform.right * movementSpeed);
        }
        if (Input.GetKey(KeyCode.A) || Input.GetKey(KeyCode.LeftArrow))
        {
            newPosition += (transform.right * -movementSpeed);
        }
    }
```

HandleCameraMovement which is called in update checks for players inputs and which style of movement is being used and moves the camera accordingly.

Unit Selection Manager



Unit Selection Manger stores all units and selected units in lists. If a unit is selected the UnitMovement script is enabled which then allows the units to move to where the player clicks. If the player clicks left clicks on the ground with units selected they are all deselected.

This script allows for the selection of units in various ways:

Multi Select(Holding down shift and left clicking on units allows you to select mulitple units)

SelectByClicking(just left clicking allows you to select a singular unit)

DragSelect(holding down left click and dragging creates ui selection box that will selected all units inside the box (also handled in a seperate script)

```
1 reference
private void SelectByClicking(GameObject unit)
{
    DeselectAll();
    unitsSelected.Add(unit);
    SelectUnit(unit, true);
}
```

```
5 references
private void SelectUnit(GameObject unit, bool isSelected)
{
    // Find the "GroundMarker" child of the unit
    Transform marker = unit.transform.Find("GroundMarker");

    if (marker != null)
    {
        // Show or hide the marker based on selection status
        marker.gameObject.SetActive(isSelected);

        if (isSelected)
        {
            // Position the marker directly under the unit
            marker.position = unit.transform.position - new Vector3(0, 0.0f, 0);  // Adjust Y to be under the unit

            marker.rotation = Quaternion.Euler(180f, 0f, 0f);
        }
    }
    else
    {
        Debug.LogWarning("GroundMarker not found on " + unit.name);
    }

    // Enable or disable unit movement
    EnableUnitMovement(unit, isSelected);
}
```

```
1 reference
private void MultiSelect(GameObject unit)
{
    if (!unitsSelected.Contains(unit))
    {
        unitsSelected.Add(unit);
        SelectUnit(unit, true);
    }
    else
    {
        SelectUnit(unit, false);
        unitsSelected.Remove(unit);
    }
}
```

```
0 references
private void Update()
{
    if (Input.GetMouseButtonDown(0))
    {
        RaycastHit hit;
        Ray ray = cam.ScreenPointToRay(Input.mousePosition);
        if (Physics.Raycast(ray, out hit, Mathf.Infinity, clickable))
        {
            if (Input.GetKey(KeyCode.LeftShift))
            {
                MultiSelect(hit.collider.gameObject);
            }
            else
            {
                SelectByClicking(hit.collider.gameObject);
            }
        }
        else
        {
            if (Input.GetKey(KeyCode.LeftShift) == false)
            {
                DeselectAll();
            }
        }
    }

    if (Input.GetMouseButtonDown(1) && unitsSelected.Count > 0)
    {
        RaycastHit hit;
        Ray ray = cam.ScreenPointToRay(Input.mousePosition);
        if (Physics.Raycast(ray, out hit, Mathf.Infinity, ground))
        {
            if (hit.collider.CompareTag("Ground"))
            {
               groundMarker.transform.position = hit.point;
               groundMarker.SetActive(false);
               groundMarker.SetActive(true);
            }
            else
            {
                return;
            }

        }
    }
```

UnitMovement

Unit Movement Handles movement using two main functions

```
// Move unit towards its current target
2 references
private void FollowTarget(GameObject currentTarget)
{
    if (currentTarget == null) return;

    NavMeshAgent unitAgent = GetComponent<NavMeshAgent>();
    Animator unitAnimator = GetComponentInChildren<Animator>();

    unitAgent.SetDestination(currentTarget.transform.position); // Move towards the target
    unitAnimator.SetBool("isMoving", true);
}


    // Move all selected units to a target position
    1 reference
    private void MoveUnitsToPosition(Vector3 targetPosition)
    {
        foreach (var unit in UnitSelectionManager.Instance.unitsSelected)
        {
            NavMeshAgent unitAgent = unit.GetComponent<NavMeshAgent>();
            Animator unitAnimator = unit.GetComponentInChildren<Animator>();

            unitAgent.SetDestination(targetPosition);  // Move the unit
            unitAnimator.SetBool("isMoving", true); // Set the moving animation
        }
    }
```

If following an enemy Follow Target is used, if Following where the player has clicked Move Units to Position is used.

Target Selection Manager

This script passes a target to UnitMovement when a target is selected,

```
1 reference
private void AssignTarget(GameObject newTarget)
{
    // Instantiate the target marker at the target's position
    // Instantiate the target marker at the target's position, rotated correctly
    GameObject marker = Instantiate(targetMarkerPrefab, newTarget.transform.position, Quaternion.Euler(0f, 0f, -180f)); // Adjust rotation

    marker.transform.position = new Vector3(marker.transform.position.x, 0.02f, marker.transform.position.z);
    // Destroy the marker after 2 seconds
    Destroy(marker, 2f);

    // Assign the target to all selected units
    foreach (var unit in UnitSelectionManager.Instance.unitsSelected)
    {
        unit.GetComponent<UnitMovement>().currentTarget = newTarget;
    }
}
```

Reflection

Working on this project has helped me improve my Teamwork and Communication. It has also greatly improved my coding as i have got to work on buidling more complex systems and am enjoying the RTS elements that i have implemented and feel they meet the standards we are trying to achieve.