Quest 9 - Steps

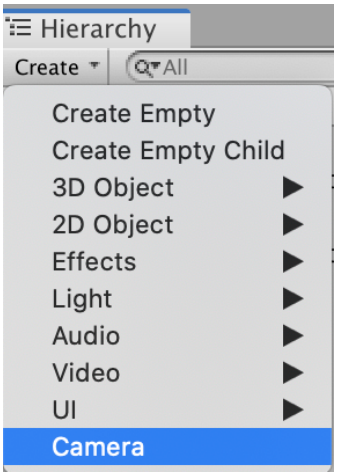
1) Setup/Cleanup

For this project, you'll need a copy of your landscape, but remove the RigidbodyFPSController. This project involves making new player characters. Any existing enemies in the landscape will either be deleted or updated based on revisions in future steps.

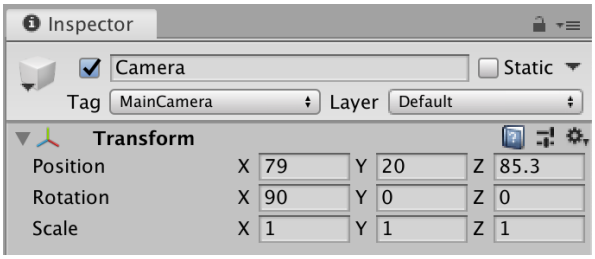
2) Camera

With the RigidbodyFPSController removed, the game needs both a new camera perspective and a new "Player." In RTS games, the player is sometimes portrayed as someone who oversees the playing field rather than an individual character in the game.

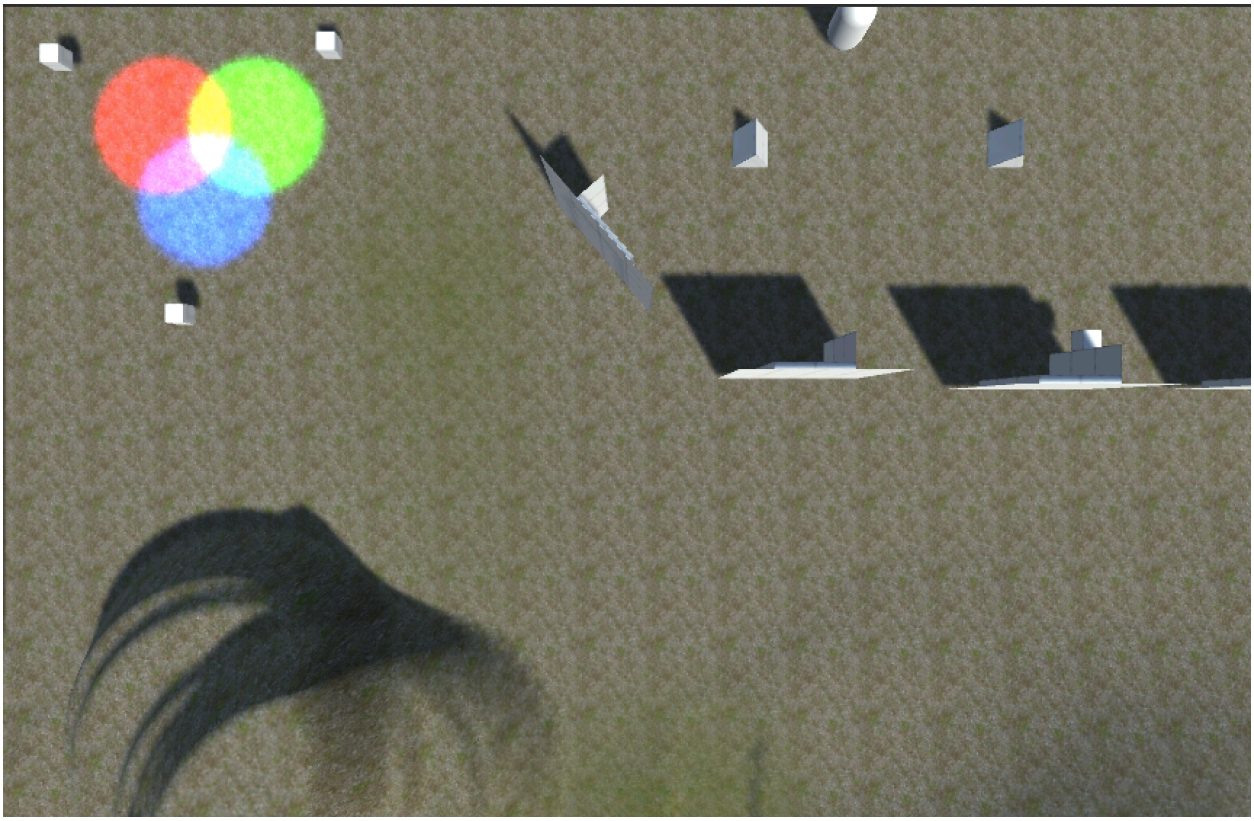
Create a new Camera object. Given prior cleanup, this should be the ONLY Camera and Audio Listener in the scene at this point.



Rotate this Camera so that it faces downward with an altitude above the terrain of 20. The camera should also be tagged as MainCamera.

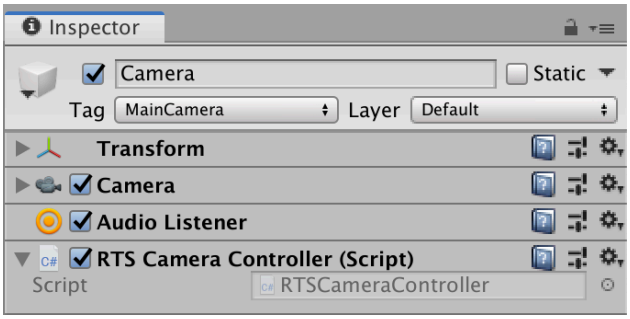


You should be able to look down on your landscape from above.



3) Camera Controls

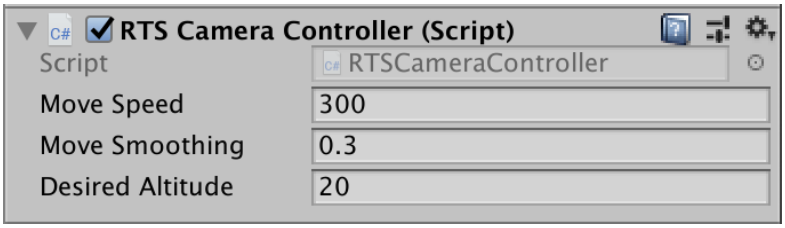
Create a new script called RTSCameraController.cs and attach it to the new Camera.



First, prepare the configuration values that define the camera's movement speed and smoothing delay.

```
public class RTSCameraController : MonoBehaviour
{
    // Configuration
    public float moveSpeed;
    public float moveSmoothing;
    public float desiredAltitude;
```

Assign values in the Unity inspector.



The Move function accepts a direction vector and scales it based on the configured speed. All of this is stored in the targetPosition variable which will be processed for movement in the Update function.

```
// State Tracking
Vector3 velocity;
Vector3 targetPosition;

// Methods
void Start() {
    targetPosition = transform.position;
}

void Move(Vector2 direction) {
    // Adjust input for configured speed
    direction = direction * moveSpeed;

    // Compute target position based on input
    targetPosition = new Vector3(transform.position.x + direction.x,
                                 desiredAltitude,
                                 transform.position.z + direction.y);
}
```

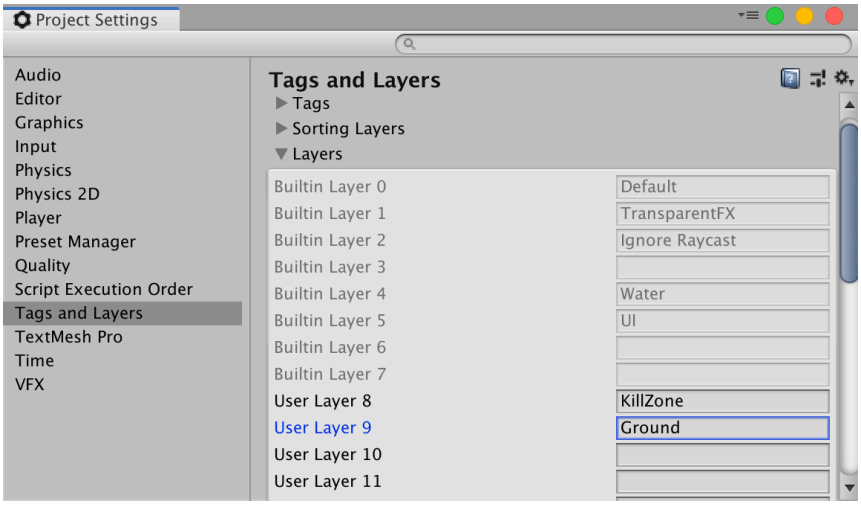
RTSCameraController will respond to directional input to smoothly move the camera. Meanwhile, the camera is smoothly moved to wherever the targetPosition is every frame, even if no keys are being pressed.

```
void Update() {
    // Combine Inputs
    Vector2 movement = Vector3.zero;
    if(Input.GetKey(KeyCode.W) || Input.GetKey(KeyCode.UpArrow)) {
        movement += Vector2.up;
    }
    if(Input.GetKey(KeyCode.S) || Input.GetKey(KeyCode.DownArrow)) {
        movement += Vector2.down;
    }
    if(Input.GetKey(KeyCode.A) || Input.GetKey(KeyCode.LeftArrow)) {
        movement += Vector2.left;
    }
    if(Input.GetKey(KeyCode.D) || Input.GetKey(KeyCode.RightArrow)) {
        movement += Vector2.right;
    }

    // Attempt Movement
    if(movement.sqrMagnitude > 0) {
        Move(movement * Time.deltaTime);
    }

    // Smoothly move the camera
    transform.position = Vector3.SmoothDamp(transform.position,
                                            targetPosition,
                                            ref velocity,
                                            moveSmoothing);
}
```

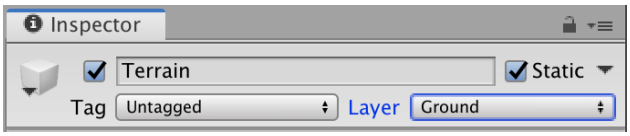
Playtest to ensure your camera smoothly moves in all the cardinal directions at a usable speed and height.

4) Further Camera Refinement

Ideally, the camera should not move past map terrain. In order to recognize valid “Ground” for the camera, create a “Ground” Layer in Edit->Project Settings->Tags and Layers. We can’t just check for a Terrain component because some levels may be built with ProBuilder or any other modeling software instead.



Take any landscape objects you have and set their Layer to Ground.

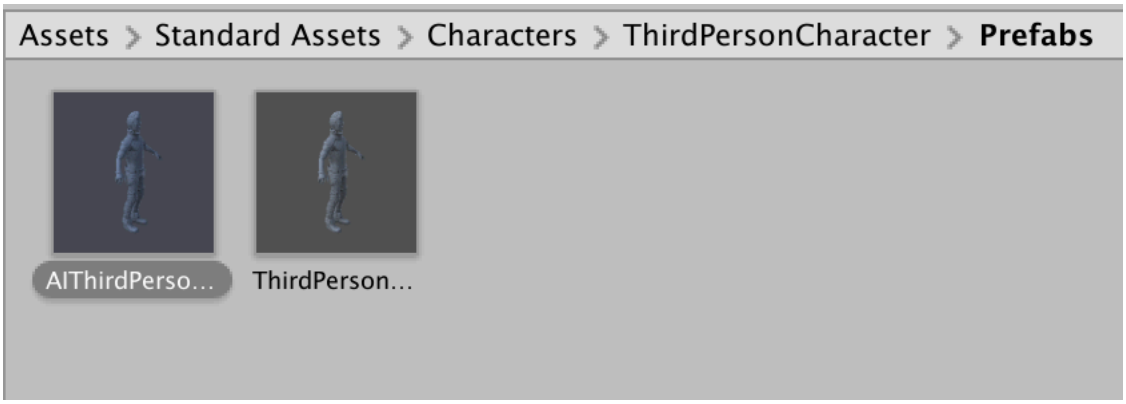


To prevent the Camera from leaving the terrain, we will raycast and check for ground below a candidate camera position before finalizing any updates to the target position. If no ground is found, that requested camera move will be rejected.

```
void Move(Vector2 direction) {
    // Adjust input for configured speed
    direction = direction * moveSpeed;

    // Compute target position based on input
    Vector3 checkPosition = new Vector3(transform.position.x + direction.x,
                                        desiredAltitude,
                                        transform.position.z + direction.y);

    // Check if new camera position would be above ground
    bool hitGround = Physics.Raycast(checkPosition,
                                     Vector3.down,
                                     Mathf.Infinity,
                                     LayerMask.GetMask("Ground"));
    if(hitGround) {
        targetPosition = checkPosition;
    }
}
```

5) Playable Units

The Unity Standard Assets package comes with a prefab that bundles an animated character model and Nav Mesh Agent.



Drag one onto your scene.

Create an RTSCharacterController.cs file and attach it to your AIThirdPersonController object. The SetDestination public function will be called to tell this character where to go.

```
using UnityEngine;
using UnityEngine.AI;

public class RTSCharacterController : MonoBehaviour
{
    public void SetDestination(Vector3 targetPosition) {
        GetComponent<NavMeshAgent>().destination = targetPosition;
    }
}
```

6) Unit Selection and Interactions

Create an RTSGameController.cs file and attach it to the Camera.

The RTSGameController manages interactions within the game. Left-clicking will select an object and set it as active by storing it in the currentSelection variable.

```
public class RTSGameController : MonoBehaviour
{
    // State Tracking
    public GameObject currentSelection;
```

Whenever the player performs a left-click, a raycast will check if a valid object can be marked as a current selection, otherwise, the current selection is cleared.

```
void Update() {
    // Selection Command
    if(Input.GetMouseButtonDown(0)) {
        // Get all objects under the mouse cursor
        Ray selectionRaycast = Camera.main.ScreenPointToRay(Input.mousePosition);
        RaycastHit[] hits = Physics.RaycastAll(selectionRaycast);

        // See if any are selectable
        bool foundSelection = false;
        foreach(RaycastHit hit in hits) {
            if(hit.collider.GetComponent<RTSCharacterController>()) {
                foundSelection = true;
                currentSelection = hit.collider.gameObject;
            }
        }

        // Deselect
        if(!foundSelection) {
            currentSelection = null;
        }
    }
}
```

If you playtest now and inspect the currentSelection variable, you will see that clicking the character selects it, while clicking anywhere else deselects the character.

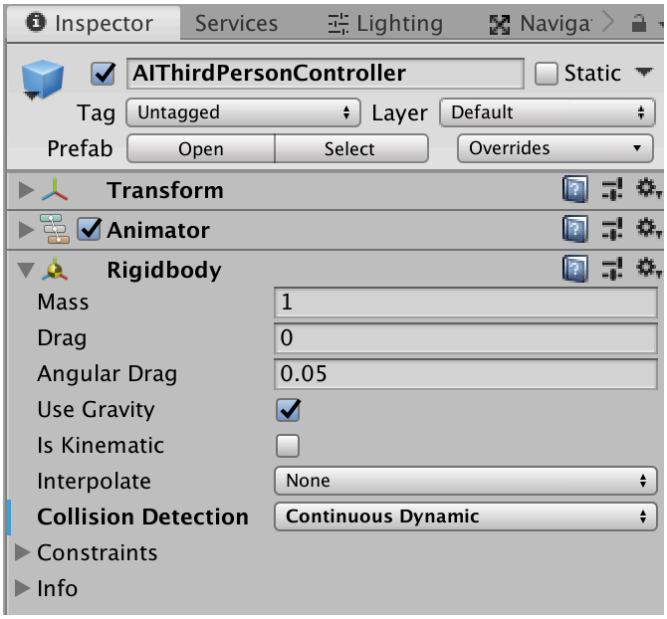
On right-click, we will set that click location as the destination of the character (if one is selected).

```
// Interact Command
if(Input.GetMouseButtonDown(1) && currentSelection) {
    RTSCharacterController rtsCharacterController = currentSelection.GetComponent<RTSCharacterController>();
    if(rtsCharacterController) {
        // Get all objects under the mouse cursor
        Ray selectionRaycast = Camera.main.ScreenPointToRay(Input.mousePosition);
        RaycastHit[] hits = Physics.RaycastAll(selectionRaycast);

        // Check for possible interactions
        foreach(RaycastHit hit in hits) {
            if(hit.collider.gameObject.layer == LayerMask.NameToLayer("Ground")) {
                // Move player to destination
                rtsCharacterController.SetDestination(hit.point);
            }
        }
    }
}
```

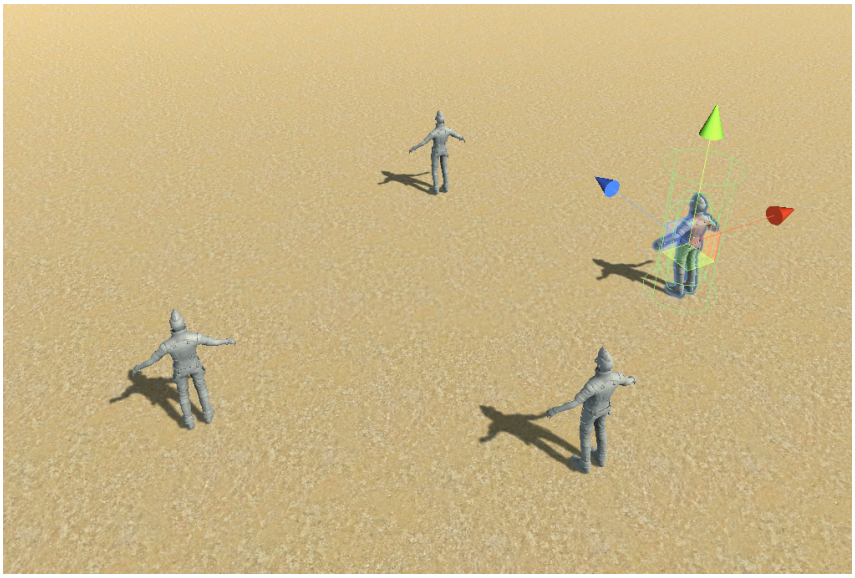
If you playtest now, the character should move to locations pointed out on right-click once the character has been marked as active with left-click.

7) Collision Check Fixes and Multiple Units

Occasionally units won't select even if we are clearly clicking on them. Because raycasting uses the physics engine, the rigidbody and collision settings will affect our ability to click objects. Setting the AIThirdPersonController's rigidbody's "Collision Detection" to "Continuous Dynamic" will help with the reliability of our selection clicks.



Duplicate the AIThirdPersonController in the scene until there are at least 3 characters to choose from. Playtest to ensure they can each be independently selected and commanded to move.



8) To be continued…