Code a RTS with Unity

Gaëtan Blaise-Cazalet – Programming teacher

1. Open Unity, create a 3D project

Add a cube in it, set its scale to (100, 1, 100) and name it Terrain

2. Player and camera panning

Create an empty object, name it Player and make the main camera a child of Player object.

Create an InputManager script.

```
[SerializeField] float panSpeed;
float panDetect = 15f;

 void Update () {
  MoveCamera();
 }

public void MoveCamera() {
  float moveX = Camera.main.transform.position.x;
  float moveZ = Camera.main.transform.position.z;

  float mouseX = Input.mousePosition.x;
  float mouseY = Input.mousePosition.y;

  if( Input.GetKey(KeyCode.Z) || (mouseX > 0 && mouseX < panDetect) ) {
    moveX -= panSpeed * Time.deltaTime;
  }
  ...

  Vector3 newPosition = new Vector3(moveX, 0, moveZ);
  Camera.main.transform.position = newPosition;
}
```

Create the rest of the move conditions.

Then, add the script to the Player. Set the camera X angle to 30. Set the pan speed to 15.

3. Camera rotation

We’ll now manage the camera rotation, making it turn while pressing middle button.

First add some fields in our InputManager :

```
[SerializeField] float rotateSpeed;
[SerializeField] float rotateAmount;
[SerializeField] Quaternion rotation;
...
float minHeight = 10f;
float maxHeight = 50f;
```

Then add a RotateCamera() function and call it in the Update() method.

```
 void Update () {
  ...
  RotateCamera();
  }
```

```
void RotateCamera() {
  Vector3 origin = Camera.main.transform.rotation.eulerAngles;
  Vector3 destination = origin;

  if (Input.GetMouseButton(2)) {
    destination.x -= Input.GetAxis("Mouse Y") * rotateAmount;
    destination.y += Input.GetAxis("Mouse X") * rotateAmount;
  }
  if (destination != origin) {
    Camera.main.transform.eulerAngles = Vector3.MoveTowards(origin, destination, rotateSpeed * Time.deltaTime);
  }
}
```

4. Camera height

Add a moveY float in the MoveCamera() function

```
float moveY = Camera.main.transform.position.y;
```

Then update this variable with the mouse wheel :

```
...
moveY += Input.GetAxis("Mouse ScrollWheel") * panSpeed;
moveY = Mathf.Clamp(moveY, minHeight, maxHeight);

Vector3 newPosition = new Vector3(moveX, moveY, moveZ);
...
```

5. Reset camera with space

Reset the camera with the spacebar, using the memorized quaternion.

```
void RotateCamera() {
  ...

  if ( Input.GetKeyDown(KeyCode.Space) ) {
    Camera.main.transform.rotation = rotation;
  }
}
```

6. Stone ressource and NavMesh

Download the StoneResource.fbx file from the repository. Put it in your project and in the scene. Set its scale to (5, 5, 5). Make it a prefab.

Create a capsule and name it Civilian. Add a NavMeshAgent component on it.

Select the Terrain, go to Windows/AI/Navigation, click on Navigation Static on the Object tap, then go to the Bake tab and Bake. It will show a blue walkable area. Add a NavMesh Obstacle to the StoneResource component, and click the Carve tickbox. Now the Civilian no longer can though the stone resource.

Apply the changes on the StoneResource.

7. ResourceManager

Create a ResourceManager script. Add two members :

```
int stone;
int maxStone;
```

8. Selection by raycast

Create an ObjectInfo script. It will have a IsSelected and and an ObjectName public fields.

```
public bool IsSelected;
public string ObjectName;

  void Start () {
    IsSelected = false;
  }
```

Add two members in the InputManager :

```
GameObject selectedObject;
ObjectInfo selectedInfo;
```

Add a Selection() function and call it in the InputManager Update() when we click :

```
  void Update () {
  ...
  if(Input.GetMouseButton(0)) {
    Selection();
  }
}
```

The Selection method will use raycast.

```
void Selection() {
  Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
  RaycastHit hit;
  if(Physics.Raycast(ray, out hit, 100)) {
    if (hit.collider.tag == "Ground") {
      selectedObject = null;
    } else if (hit.collider.tag == "Selectable") {
      selectedObject = hit.collider.gameObject;
      selectedInfo = selectedObject.GetComponent<ObjectInfo>();
      selectedInfo.IsSelected = true;
    }
  }
}
```

Add the Ground and the Selectable tags. The Civilian and the StoneResource have the Selectable tag and the Terrain has the Ground tag.

Add some debug log to test the selection :

```
  void Selection() {
    …
    if(Physics.Raycast(ray, out hit, 100)) {
      if (hit.collider.tag == "Ground") {
        …
        Debug.Log("Deselected");
      } else if (hit.collider.tag == "Selectable") {
        …
        Debug.Log("Selected " + selectedInfo.ObjectName);
      }
    }
  }
```

9. Move the civilian

In ObjectInfo, add the UnityEngine.AI namespace, and a NavMeshAgent field. Load the component.

```
using UnityEngine.AI;

…

  NavMeshAgent agent;

  void Start () {
    IsSelected = false;
    agent = GetComponent<NavMeshAgent>();
  }
```

We will use the right mouse button to move the civilian.

```
  void Update() {
    if(Input.GetMouseButtonDown(1) && IsSelected) {
      RightClickCommand();
    }
  }

  void RightClickCommand () {
    Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
    RaycastHit hit;
    if (Physics.Raycast(ray, out hit, 100)) {
      // Move
      if(hit.collider.tag == "Ground") {
        agent.destination = hit.point;
        Debug.Log("Moving");
      }
    }
  }
```

10. Get the civilian to harverst

Add a Sphere collider on the StoneResource. Set it as a trigger. Change its size to make the collider bigger than the Resource.

Add a RigidBody on the Civilian.

Create a NodeManager script.

```
    public enum ResourceType {
        Stone
    }
    public int Gatherers;

    public ResourceType resourceType;
    [SerializeField] float harvestTime;
    [SerializeField] float availableResource;


    // Use this for initialization
    void Start () {
        StartCoroutine(ResourceTick());
    }

    // Update is called once per frame
    void Update () {
        if (availableResource <= 0) {
            Destroy(gameObject);
        }
    }

    IEnumerator ResourceTick() {
        while(true) {
            yield return new WaitForSeconds(1);
            ResourceGather();
        }
    }

    public void ResourceGather() {
        if(Gatherers > 0) {
            availableResource -= Gatherers;
        }
    }
```

Let’s add the gathering logic in the ObjectInfo’s RightClickCommand :

```
void RightClickCommand() {
    …
    if (Physics.Raycast(ray, out hit, 100)) {
      // Move
      if(hit.collider.tag == "Ground") {
        …
      }
      // Harvest
      else if (hit.collider.tag == "Resource") {
        agent.destination = hit.collider.gameObject.transform.position;
        Debug.Log("Harvesting");
      }
    }
  }
```

Create the ressource tag and add it to the stone resource. Update the prefab.

11. Gather resources

Add fields in the ObjectInfo script.

```
    public NodeManager.ResourceType heldResourceType;

    [SerializeField] int maxHeldResource;

    int heldResource;
    GameObject[] resourceDrops;
```

Prepare also the held ressource increase with a new GatherTick method :

```
    public bool IsGathering;
```

```
    IEnumerator GatherTick() {
        while(true) {
            yield return new WaitForSeconds(1);
            if(IsGathering && heldResource < maxHeldResource) {
                heldResource++;
            }
        }
    }
```

Setup the coroutine the the Start() method.

```
    void Start()
    {
        IsSelected = false;
        agent = GetComponent<NavMeshAgent>();
        StartCoroutine(GatherTick());
    }
```

Gather resources when the trigger activates.

```
    public void OnTriggerEnter(Collider other)
    {
        GameObject hitObject = other.gameObject;
        if (hitObject.tag == "Resource")
        {
            hitObject.GetComponent<NodeManager>().Gatherers++;
            IsGathering = true;
            heldResourceType = hitObject.GetComponent<NodeManager>().NodeResourceType;
        }
    }
```

Stop gathering when exiting trigger :

```
    public void OnTriggerExit(Collider other)
    {
        GameObject hitObject = other.gameObject;
        if (hitObject.tag == "Resource")
        {
            IsGathering = false;
            hitObject.GetComponent<NodeManager>().Gatherers--;
        }
    }
```

In the Update function, prepare the Gathering stop :

```
        if (heldResource >= maxHeldResource)
        {
            /* Go to drop off building */
        }
```

12. Resource drop

Create a cub, call it ResourceDrop. Create a new “Drop” tag and set the ResourceDrop with it. Don’t forget to set the NavMesh again.

Add a ResourceManager script and put it on the Player object.

```
public class ResourceManager : MonoBehaviour
{
  public int Stone;
  [SerializeField] int maxStone = 100;

  // Use this for initialization
  void Start()
  {
    Stone = 0;
  }

  public void AddStone(int moreStone)
  {
    Stone += moreStone;
    Stone = Mathf.Min(Stone, maxStone);
  }
}
```

Don’t forget to set the player with the Player tag.

In ObjectInfo, add a GameObject array :

```
    GameObject[] resourceDrops;

```

When the Civilian has found enough resources, get all the resourceDrops :

```
    void Update()
    {
        if (heldResource >= maxHeldResource)
        {
            resourceDrops = GameObject.FindGameObjectsWithTag("Drops");
            agent.destination = GetClosestDropOff(resourceDrops).transform.position;
            resourceDrops = null;
        }
        …
    }
```

We have to create the GetClosestDropOff function.

```
    GameObject GetClosestDropOff(GameObject[] drops)
    {
        GameObject closest = null;
        float closestDistance = Mathf.Infinity;

        foreach (GameObject drop in drops)
        {
            Vector3 direction = drop.transform.position - transform.position;
            float distance = direction.sqrMagnitude;
            if (distance < closestDistance)
            {
                closestDistance = distance;
                closest = drop;
            }
        }
        return closest;
    }
```

Create a task enum, outside the ObjectInfo class.

```
public enum Task {
  Idle,
  Gathering,
  Delivering,
  Moving
}
```

Add a task member in ObjectInfo and Start ObjectInfo with the Idle task.

```
    Task task;
    void Start()
    {
        …
        task = Task.Idle;
    }
```

Update RightClickCommand :

```
    void RightClickCommand()
    {
        …
        if (Physics.Raycast(ray, out hit, 100))
        {
            // Move
            if (hit.collider.tag == "Ground")
            {
                …
            }
            // Harvest
            else if (hit.collider.tag == "Resource")
            {
                …
                task = Task.Gathering;
            }
        }
    }
```

Update OnTriggerEnter :

```
  public void OnTriggerEnter(Collider other)
  {
    GameObject hitObject = other.gameObject;
    if (hitObject.tag == "Resource" && task == Task.Gathering)
    {
      hitObject.GetComponent<NodeManager>().Gatherers++;
      IsGathering = true;
      heldResourceType = hitObject.GetComponent<NodeManager>().NodeResourceType;
    }
    else if (hitObject.tag == "Drop" && task == Task.Delivering)
    {
      GameObject.FindGameObjectWithTag("Player").GetComponent<ResourceManager>().AddStone(heldResource);
      heldResource = 0;
      task = Task.Gathering;
      // Get back to work !
    }
  }
```

Now, try to implement other functionalities.