

AI Navigation with Waypoints in Unity

Objective

This lab introduces AI navigation using waypoints in Unity.

Lab Setup

1. Assets Needed:

- A ground plane.
- Tank model.
- Palm tree prefab (used as waypoints).

Step 1: Preparing the Scene



Step 2: Creating the Waypoint Navigation Script

Create a New Script:

- In the "Scripts" folder, create a new C# script named FollowWaypoint.

Explanation of the Code:

- **Waypoints Array:** Stores all the waypoints the tank will navigate through.

- **Distance Check:** Determines if the tank is close enough to switch to the next waypoint.
- **Smooth Rotation:** Uses Quaternion.Slerp for gradual turning towards the next waypoint.
`transform.rotation = Quaternion.Slerp(transform.rotation, lookRotation, Time.deltaTime * rotSpeed);`
- **Forward Movement:** Moves the tank along its local Z-axis.
`transform.Translate(0, 0, speed * Time.deltaTime);`

Assigning Waypoints in Unity

Attach the Script:

- Select the tank objects in the Hierarchy.
- Drag the FollowWaypoint script onto it.
- Attach the waypoints to the script

Test the System:

- Press Play. The tank should navigate through the waypoints in sequence, looping indefinitely.

Optional Enhancing Movement

1. Smoother Rotation:

- Replace the existing rotation logic with the following:

`Vector3 direction = waypoints[currentWaypoint].transform.position - transform.position;`

`Quaternion lookRotation = Quaternion.LookRotation(direction);`

`transform.rotation = Quaternion.Slerp(transform.rotation, lookRotation, Time.deltaTime * rotSpeed);`

- Adjust rotSpeed in the Inspector for gradual or sharp turns.

Adding Physics

1. Add Colliders and Rigidbodies:

- Add more tanks with different speed & rotations
- Select all tank objects.
- Add a **Box Collider** and **Rigidbody** component to each tank.
- In the Rigidbody settings, freeze rotation on the X and Z axes.

2. Enable Physical Interaction:

- Tanks will now collide with each other, adding realism to their navigation.



The “Stuck” Waypoint Problem

If a car or tank **misses** a specific waypoint after being knocked off course, it may never pass within the required distance to update its next waypoint. In racing scenarios, AI might:

- **Fail** to progress further around the circuit.
- End up circling around a single tree, waiting to get bumped back on course.

This highlights a **drawback** of enforcing precise waypoint checks when physics can disrupt movement.

Extra : The “Progress Tracker” Approach

A common solution is creating a hidden object (a “progress tracker”) that strictly follows the ideal race line or waypoint sequence. Your actual tanks then steer toward this moving tracker instead of a single fixed checkpoint.

- Allows smooth continuous movement even if the tank is physically bumped off track.
- Doesn't force the tank to "correct" itself back to a single spot on the map it missed.

When **physics** and **waypoints** mix, AI agents can get bumped off-course and end up stuck circling a single waypoint. To solve this, you can introduce a **tracker object**—a hidden "ghost" that follows the precise waypoint route. Your AI then simply **follows** that tracker, rather than an exact tree or checkpoint on the map.