WAYPOINTS PATH SYSTEM v1.2

Basic manual

Overview

Universal and powerful waypoint system allows you to create waypoints and path easily and even in run-time. You can use it for any situations where path/waypoints following are needed:

- Moving platforms
- AI movement and patrolling NPCs behavior
- Obstacle avoidance
- Animated environment objects
- etc.

System supports:

- Dynamic path changing (i.e. even characters can be turned into waypoints)
- Paths intersecting and common/shared waypoints
- Different looping types
- Different facing /movement types
- Dynamic speed changing (when the waypoint is reached)
- Custom functions calling (when the waypoint is reached)
- Dynamic preventing of collision with mooving obstacles etc.

Moreover you can trigger delays and any other actions/functions right into waypoint!
This system works on all platforms supported by Unity3D.

How to use

To use Waypoints path system – you should just:

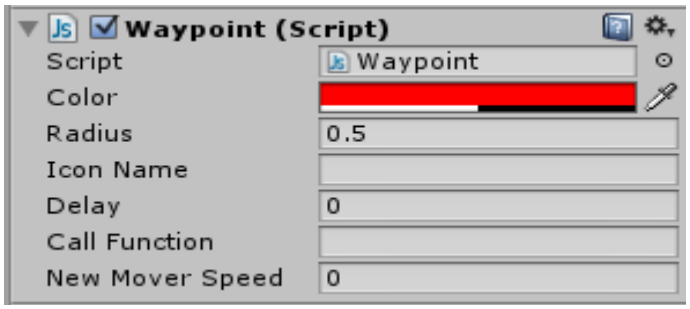
1. Assign *WaypointHolder* script to any gameObject
2. Create several waypoints (using prefab or just assign *Waypoint* script to any gameObject)
3. Assign waypoints to list in *WaypointHolder* or just make *WaypointHolder* parent to them (in this case path will be created automatically according to waypoints order in Hierarchy)
4. Attach *WaypointMover* script to any gameObject you want to move along path (and assign *WaypointHolder* to its related property).
5. If needed - tune movement parameters… and…. Enjoy!

Please check additional detail below.

Please don't hesitate to contact me in any reason by mail: ***AllebiGames@gmail.com***

Waypoint script description

Basic script that contains waypoint visualisation options and associated actions
Average structure looks like:



Debug visualization options

- **Color** - Waypoint gizmo color
- **Radius** - Waypoint gizmo size
- **IconName** - Waypoint gizmo icon filename

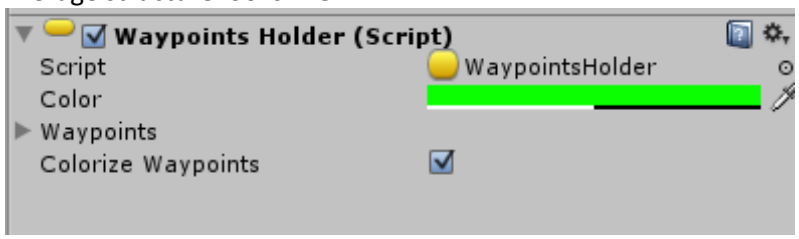
Associated actions

- **Delay** - Delay movement for next waypoint, when Mover object rich the waypoint
- **CallFunction** - Call function with this name (in any script attached to Mover object), when Mover object riches the waypoint
- **NewMoverSpeed** - If more than 0 will change speed(to specified here) of WaipointMover that reached this waypoint

WaypointsHolder script description

Object with this script hold waypoints as path and visualize it.
If list of waypoints is empty - Script will try to gather all child objects (with waypoint script attached) as waypoints on start. In this case path will be created automatically according to waypoints order in Hierarchy

Average structure looks like:



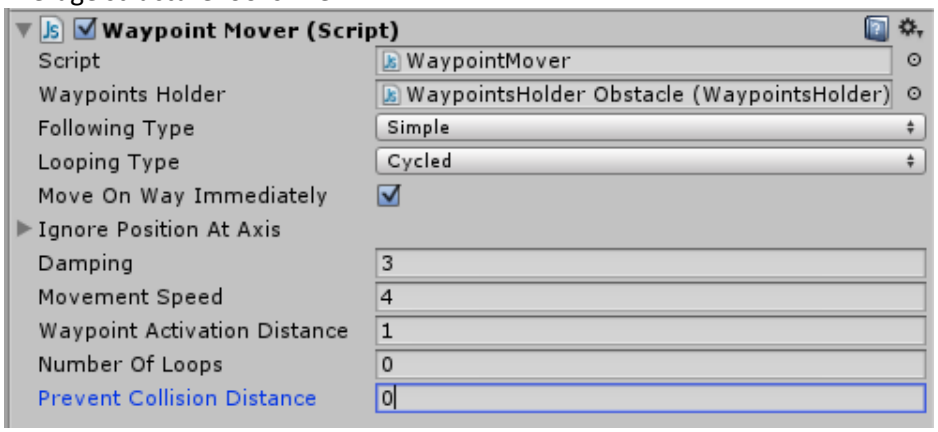
Parameters description:

- **Color** - Debug path lines color
- **Waypoints** - List of all waypoints assigned to this path
- **ColorizeWaypoints** - Repaint all waypoints in the color

WaypointMover script description

The most important script. It manages waypointed path from *waypointsHolder* and move object (to which it's attached) along the path.
Script also allows to setup different following and looping types for movement.

Average structure looks like:



Parameters description:

- **WaypointsHolder** - Move along the path holded in this WaypointsHolder

- **FollowingType** - Choose one of following type to use:
    - *Simple* - Just move object as it is (without any rotation or dumping)
    - *Facing* - Roughly face object on current waypoint
    - *SmoothFacing* - Face object on current waypoint and adapt path smoothly
    -
- **Looping Type** - Choose one of looping type to use:
    - *Once* - Only one cycle
    - *Cycled* - Infinite amounts of cycles
    - *PingPong* - Move object in another direction when it gets first/last point of path
    - *SeveralTimes* - Repeat loop specified number of loops

- **IgnorePositionAtAxis** - Ignore waypoint position along those axis

- **Damping** - Smooth facing/movement value

- **MovementSpeed** - Speed of object movement along the path

- **WaypointActivationDistance** - How far should object be to waypoint for its activation and choosing new

- **NumberOfLoops** - How much loops should be performed before stop. Use this parameter if loopingType = SeveralTimes

- **PreventCollisionDistance** - If more than 0 - Mover will suspend movement if there is any obstacle on this distance (in front of him)