Poly|Nav: Simple 2D Pathfinding Solution

PolyNav is a path finding system to use with Unity's 4.3+ 2D system. One of the best benefits of PolyNav2D is it's ease of use and setup. There are three prime components you should be aware of:

PolyNav2D is the main component responsible for navigation map generation. You can create one through "Tools/ParadoxNotion/PolyNav/Create PolyNav Map". There can be multiple in the scene if required.

PolyNavAgent is the component that you should attach on any game object that you would like to navigate around the world. This is not mandatory as you can create your own if you want to.

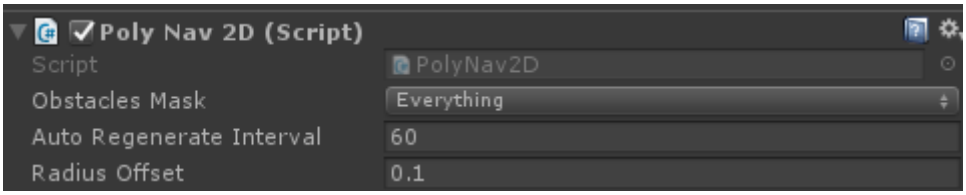
PolyNavObstacle is a component responsive for marking a gameobject that has a Collider2D as obstacle (or hole) for a PolyNav2D navigation map.

Step 1

The first thing you should want to do, is create your first PolyNav2D map through "Tools/ParadoxNotion/Create PolyNav Map". A PolygonCollider2D will automatically be attached as well on that new created gameobject. By using the default Unity controls for adjusting such a collider, you can define any shape for the navigation polygon.

(Ctrl + Click to remove a point. Shift + Click&Drag to move a point or to add a new if hovering an edge)

The options on the PolyNav2D component itself, are rather few.



Obstacles Mask defines which layers this map, will look for PolyNavObstacles to take into account when the map is generated. As such you can fine grain which NavObstacles are meant for which PolyNav2D maps, but you of course also use same NavObstacles on any amount of different maps.

Auto Regenerate Interval is the interval in frames which the map will check for NavObstacles changes (enable, disable or transform change) and if any occurs, the map will automatically be regenerated. You can set this option to 0 to basically disable this feature since it can be very constly on performance and rather manually regenerate the map using PolyNav2D.GenerateMap(); whenever is really required through code.

Radius Offset is the offset from the edges of the map, that agents will be kept at. Typically, you set this radius to be equal to the radius of the agents that will use this map.

To create an obstacle (or a hole) within the main navigation map polygon, you can do so through "Tools/ParadoxNotion/PolyNav/Create Obstacle". Since NavObstacles also come with a PolygonCollider2D, you can edit them the same way as you did the main navigation polygon.

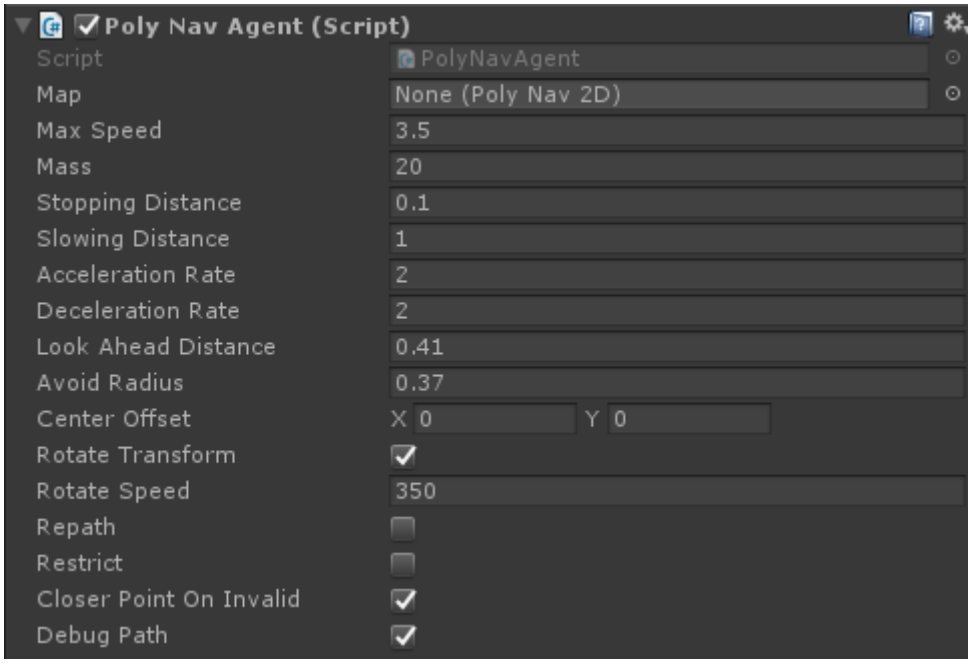
You don't have to be exact. You can have half an obstacle inside the walkable area, or multiple obstacles overlapping each other if required.

The PolyNavObstacle component can of course also be added manually on a gameobject that you want to make obstacle. This gameobject can pre-exist in the scene or be part of a runtime instantiated prefab if required.

The PolyNavObstacle has an option to I**nvert Polygon**. If the component is added on a gameobject that already has an automated created PolygonCollider because of a Sprite component attached, it will be set to true. If you think something goes wrong with the navigation map, it will most probably be because the exiting polygon collider of a PolyNavObstacle is not inverted. So please try to set this to true first.

Step 2

Select the game object that you want to be able to navigate within the map and attach the PolyNavAgent component to it.



Map is the PolyNav2D map that this agent is bound to move on. If left as None, then the first map found in the scene will be used. It is also quite possible to alter this property in runtime to make the agent change bound map!
Max Speed is of course the maximum speed that the agent can move.
Mass is the mass of the agent. The bigger the mass the more inertia it will have.
Stopping Distance is the distance from the goal that the agent will and consider reached.
Slowing Distance is the distance from the goal that the agent will begin slowing down.
~~**Acceleration Rate** is the rate at which the agent accelerates towards Max Speed once start moving.~~
~~**Deceleration Rate** is the rate at which the agent will slow down when reaching the goal.~~
Look Ahead Distance is the distance that the agent look ahead for obstacles or other agents if other agents have an Avoidance Radius. You can set this value to 0, to disable obstacle and agent look ahead.
Avoid Radius is the agent avoidance radius with other agents. Leave it at 0 if you don't need avoidance.
Center Offset is a virtual offset from the transform.position that the pathfinding will work with. This can be useful if you want to basically re-pivot the agent center without changing the sprite real center.
Rotate Transform if checked will also rotate the agent.
Rotate Speed is the speed that the agent will rotate if Rotate Transform is checked.
Repath if enabled the agent will revaluate and repath its current path for occasions where the map has changed. Disable this for performance.
Restrict if enabled will force the agent position within valid areas. Disable this for performance.
Closer Point On Invalid will make the agent go to the closer possible position if requested goal is invalid.
Debug Path if checked will draw the path.

There are 2 demo scripts provided to move the agent about for a quick start. **ClickToMove** and **MoveBetween**

ClickToMove will simply move the agent at the clicked position while *MoveBetween* will move the agent between some way points at random selection.

Simply attach either on the agent gameobject if you want a quick start!

There are also some other helpful scripts included under the DEMO/Scripts folder, worth to check out!

Step 3

Regarding scripting, there are 2 important function on the PolyNavAgent:

bool SetDestination(Vector2 goal, Action<bool> callback = null)
Sets a new goal for the agent. You can optionally provide a callback function which will be called on either arrival with a *True* argument or if the destination is or becomes invalid with a *False* argument. The function itself return whether or not the destination is valid in the first place.

An alternative way instead of providing a callback function will be to subscribe to the events that the Poly-NavAgent component raise:

OnDestinationReached
OnDestinationInvalid
OnNavigationStarted
OnNavigationPointReached<Vector2> (this is when a path corner point is reached)

Please check the demo scripts ClickToMove and MoveBetween for example usage.

Stop()
Will simply stop the agent from moving.

There are also some important properties that you can use:

pathPending:bool	is any path pending right now for the agent?
hasPath:bool	does the agent has any active path?
nextPoint:Vector2	the next point in the path that the agent is moving towards.
remainingDistance:float	the remaining distance along the path for the goal.
movingDirection:Vector2	the direction the agent is currently moving at.

Of course there are a lot for functions and properties on the PolyNavAgent worth to check out!

Extra

Notice that you can also request a path directly without using the PolyNavAgent component at all, but rather directly from a PolyNav2D map instance and for any purpose. You can do so like this:
PolyNav2D.FindPath(Vector2 start, Vector2 end, Action<Vector2[]> callback);

The callback will provide the resulting path, or null if a path is impossible or not found.

Well that's it! Remember that you can enable "Gizmos" in play mode on the top right of the "Game" tab, so that you are able to view the polygon map, obstacles and agent paths for debugging.

Thanks and I hope you enjoy the simplicity of Poly|Nav

Gavalakis Vaggelis - support@paradoxnotion.com

Unity Forum Link