AggregatGames' Pathfinding

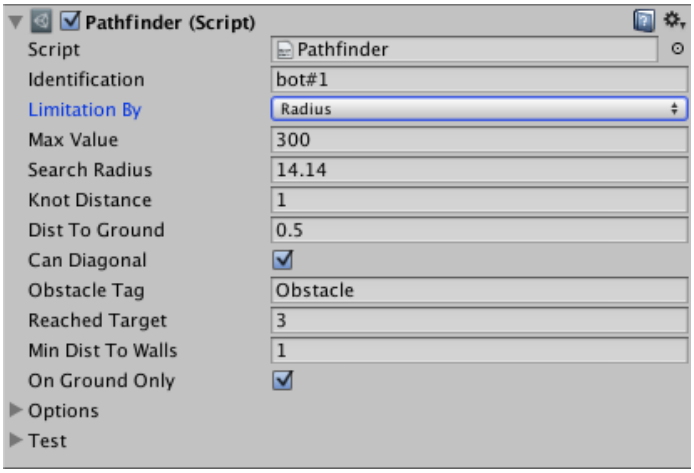
0.: General information

Our Pathfinding asset is based on the A* Pathfinding Algorithm and shows your AI the shortest possible route. This Pathfinding system works without any type of Navigation Mesh what makes it possible to have intelligent, pathfinding AI in an open world scene finding it's path. The asset also gives you the ability to determine when the AI reached it's target, how far the knots the AI will be walking on will be apart and how much the AI should calculate.

1.: Project setup

The setup of the Pathfinder couldn't be easier. Select your AI in the Hierarchy, click “Add Component” in the inspector, type in “Pathfinder” and add the Pathfinder Script to your AI.

You should now have a menu that looks like this [img1] in your Inspector. Every variable has a tooltip showing up by hovering the variable name and explaining the variable. The foldout “Test” is for debugging and project planning purposes, we will go into that later.



img1

2.: Scripting

To use the Pathfinder include the Pathfinding Asset (*using AggregatGames.AI.Pathfinding;*) create a new variable of the type *Pathfinder,* another one of the type *PathKnot[]* and an integer (int) for the knot index you are currently on. The Pathfinder variable should be the Pathfinder Script you just added to your AI, the *PathKnot[]* should be unassigned for now and the integer should have a value of -1.

> *I'm using the integer to tell me, which index of the knots the AI still has to go to and which one it reached but also if the AI should walk, stay or if an error in the pathfinding occurred. It is not necessary in this case to use an int, it can be solved individually.*

Create a new method (this step is not necessary but I would recommend it) which you can give the parameter *Transform target*. Create the following if statement:

```
if (knotIndex == -1 || knots == null || pathfinder.target != target.position) {
```

In this case *knots* is my *PathKnot[]* and *pathfinder* is my *Pathfinder*. If the statement is true call the method *findPath* from your Pathfinder and give it the parameters:

(Vector3 startPosition, Vector3 targetPosition, method foundPath)

startPosition should be *transform.position* to get your AI's position.

targetPosition is *target.position*

and for *foundPath* just create a new public method with the parameter *(Pathinfo info)*.

Now ask the pathfinder if there is a valid path by calling *info.foundPath* in your foundPath method and if it is true set your variable from type *PathKnot[]* to *pathfinder.getPath()* and *knotIndex* to *0*. If there isn't a path that could be found set *knotIndex* to *-3*. Go back to your *if (knotIndex == -1 || knots == null || pathfinder.target != target.position)* and add an else if like this:

```
} else if (knotIndex != -3 && knotIndex != -2) {
```

©AggregatGames – FreakySheldon 2014

to it. Now do what ever you want to do with the knots. You can get a knot's position by calling *knot.position*. Have a look at the example script to get your AI follow the path.
With our asset we give you the opportunity to dynamically trigger an obstacle by scripting or create entries only some Pathfinders can find a path through. To make a mesh to be detected as obstacle you can give it your obstacle tag or add the obstacle script to it but there is more to it. To create a trigger able obstacle create a new class and extend *AggregatGames.AI.Pathfinding.Obstacle*. So that it should look like this:

public class [your classname] : Obstacle {

now override the obstacle's *isObstacle* method:

public override bool isObstacle(Pathfinder pathfinder) {

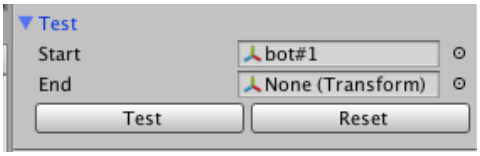
and write in there if the pathfinder given as parameter should be allowed to pass or not, maybe your PathfinderID “Hobbit” should pass but the PathfinderID “Balrog” should not, that would look like that:

```
using UnityEngine;
using System.Collections;
using AggregatGames.AI.Pathfinding;

public class specificObstacle : Obstacle {
    public string forPathfinder;

    public override bool isObstacle(Pathfinder pathfinder) {
        return (pathfinder.identification == forPathfinder);
    }
}
```

Now add the specificObstacle to a wall and set forPathfinder to “Balrog” or whatever the ID of the Pathfinder is that should not be able to pass.

3.: “Test”-Foldout

The “Test”-Foldout can be used to plan your project or debug by testing situations in the editor. When you open the foldout you'll see two variables: Start and End and two buttons: Test and Reset ([img2]). You are not able to change the Start-, but the End variable. When you press Test, the Pathfinder will find the shortest distance between the Start and the End and will show it's result (only if you turned DrawGizmos on).



img2

Reset will only clear the path you found before and the information if the Pathfinder is calculating or not. The boolean *isCalculating* is important for the Pathfinder in game to prevent that it finds more than one graph at a time per instance.

4.: “Options”-Foldout

The “Options”-Foldout contains some options for debugging purposes. You can turn debugging on and off, change the level off debugging and determine which Gizmo should be drawn.

5.: Documentation

AggregatGames.AI.Pathfinding

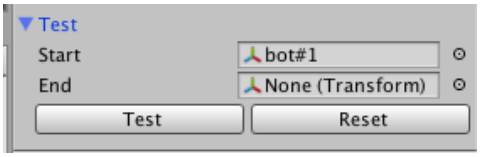
- *(class)* Pathfinder
  - *(string)* signature
  - *(string)* contact
  - *(Vector3)* target
  - *(Vector3)* start
  - *(string)* identification
  - (*limitationType*) limitationBy;
  - *(int)* maxValue
  - (*int*) knotsPerFrame

©AggregatGames – FreakySheldon 2014

    - (*float*) searchRadius
    - *(int)* knotDistance
    - (*float*) distToGround
    - *(bool)* canDiagonal
    - (*bool*) smoothPath
    - *(string)* obstacleTag
    - *(float)* reachedTarget
    - *(float)* minDistToWalls
    - *(bool)* isCalculating
    - *(bool)* onGroundOnly
    - *(Options)* options
    - *(PathKnot)* knotThatFoundTarget
    - *(void)* foundPath
    - *(PathKnot[])* getPath()
    - *(void)* findPath(Vector3 start, Vector3 target, method returnMethod)
    - (*bool*) knotValueLimited()
    - (*bool*) rangeLimited()
- *(enum)* limitationType
    - MaxValue
    - Radius
    - Both
    - None
- (*class*) Options
    - (*bool*) debug
    - (*int*) debugLevel
    - (*bool*) drawGizmos
    - (*GizmoOptions*) gizmoOptions
- (*class*) GizmoOptions
    - (*bool*) drawSearchRadius
    - (*bool*) drawHitbox
    - (*bool*) drawPath
- *(class)* PathKnot
    - *(static float)* extraForDiagonal
    - *(Vector3)* position
    - *(float)* value
    - *(int)* pathknotsBefore
    - *(List<PathKnot>)* before
    - *(boolean)* isChecked
    - *(boolean)* foundTarget
    - (*PathKnot*) shortCut
    - (*Constructor*) PathKnot(List<PathKnot> before, Vector3 position, int distToStart, int distToTarget, bool diagonal)
- *(class)* Path
    - *(PathKnot[])* getPath()
    - (*List<PathKnot>*) getPathList()
    - (*DynamicObstacle[]*) getDynamicObstaclesBlocking(PathKnot knot1, PathKnot knot2)
    - (*bool*) blockedByDynamicObstacle(PathKnot knot1, PathKnot knot2)
    - (*bool*) blockedByDynamicObstacle()

©AggregatGames – FreakySheldon 2014

- *(class)* PathInfo
  - *(Path)* path
  - *(boolean)* foundPath
  - (*string*) comment
- *(class)* Obstacle
  - *(virtual boolean)* isObstacle(Pathfinder pathfinder)



img2

6.: Some bugs

During the time of creation we found a few bugs in our pathfinder, luckily they don't have bad effects and are solvable. Okay... actually we found only one bug important enough to be listed here.
The bug is that it happens that your pathfinder thinks it is calculating and blocks other calculations, when it is not calculating. This can happen when you run it but any of you'r parameters or variables are invalid and the Pathfinder can't finish calculating when you press the Test button. This is fixed by pressing reset.

7.: Contact information

If you still have some questions have a look at our tutorial: [Product Release] Innovative Pathfinding
Or contact our support or the creator
If you like our products please donate at our contribution page to make us able to create more good stuff.

©AggregatGames – FreakySheldon 2014