PathFinder for 3D GameStudio

Version 1.3

By MadJack
11 Mars 2002

MdJk_DLL is a freeware Pathfinder toolkit for 3D GameStudio.
You can download it at www.antares40.com and use it with usual licence terms for freeware.

Contents

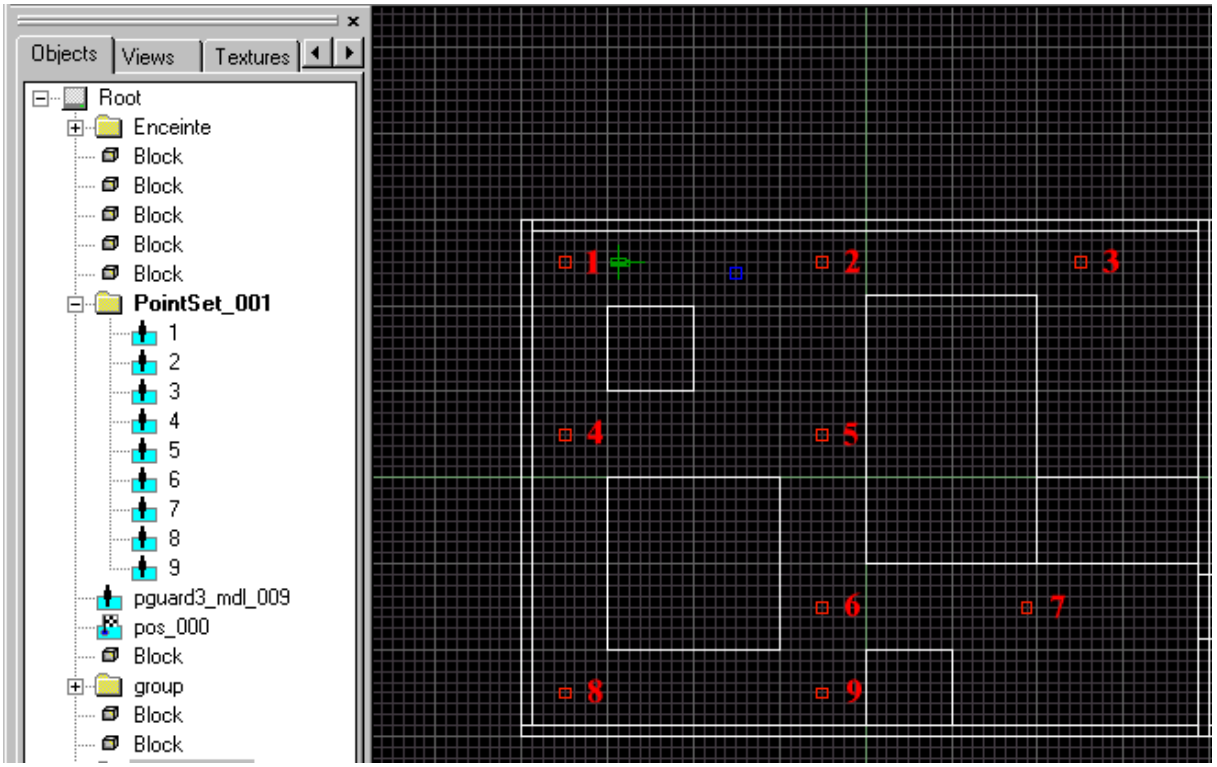
1 Method

The first idea was to define a list of nodes on the map and list all the possible paths. Given this list, we could get the shortest way between two points (and more). In order to save time I thought that we needed to save all pathes in a file. This solution is not so good, we can't change easily a node state (opened or closed).

Finally, in this first version, a new path is calculated on main function call. I don't really know if it takes too much time, I guess we'll have to be careful when setting nodes and links. Anyway, the best way to know is to test….

2 How to set points

It is much more friendly user to set points straith on the map with the map editor (WED), but points with properties (skills) don't exist in WED. So ~~I use a basic entity that is a small cube~~. I use a basic model that is a small cube.
I insert this cube at every path node on the map like this :

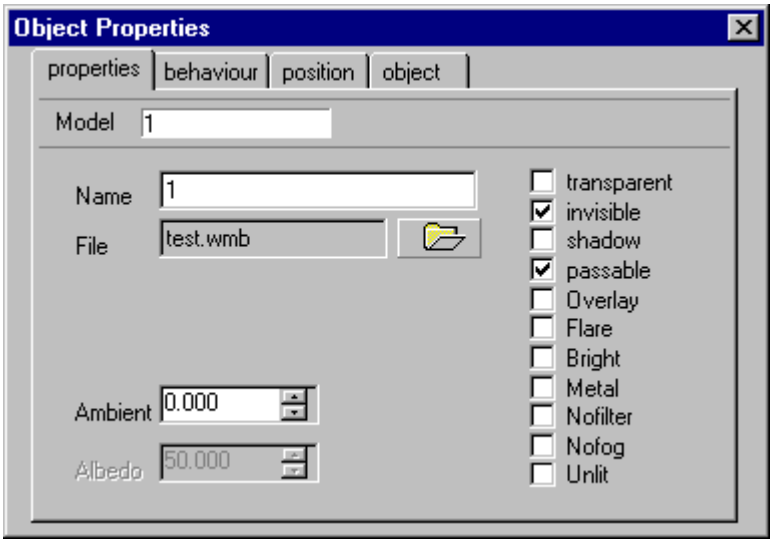


NB : Point's numbers are manually added in this picture.

If the DLL get enough success, we could ask Doug to do something for us, for example :

- creating points as a new objet with automatic number
- the way to set links by drawing an arrow
- …

Note that those points are useful not only for path calculating but also for other AI purposes.

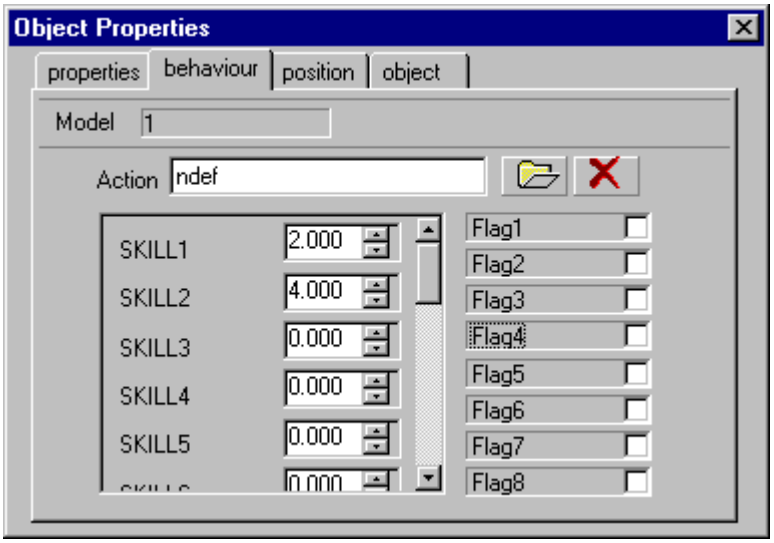


For each point, I open properties panel.

1° - Set the name of point to a number
This must be a unique ID.

2° - Set flags « invisible » and « passable »

Note : ~~If you use default move.wdl, script, it is better that you set points near from the ceiling, otherwise, your characters might start swimming when going thrue points~~…
Using model (waypoint.mdl) you can set point where you want.



In the bahviour panel, use editable skills 1 to 8, to set links with other points.

For example, if you look at the basic level shown in the first figure, you can see that from point number 1, we can go to point number 2 and point number 4.

So :

Set skill1 to 2
Set skill2 to 4

NEW

Flag 1 is set when the point is closed, here, setting flag 1 means that we can't pass thrue point 1.

Last, for reading facility, I group all these points (drag and drop is usefull here)

That'is finished with WED.

3 How to use MdJk_DLL.dll

Place MdJk_DLL.dll in your project folder.
In your script, you'll have to declare MdJk_DLL functions before to use it :

// MdJk_DLL.DLL: functions declaration

```
dllfunction mdjk_next_point(EntityPtr, Pt1, Pt2);     // EntityPtr is the pointer to the entity
                                                      // Pt1=Starting point
                                                      // Pt2 =Destination point
dllfunction mdjk_distance(EntityPtr);                 //
dllfunction mdjk_count(EntityPtr);                    //
dllfunction mdjk_point(EntityPtr, Index);             // Index = waypoint index
dllfunction mdjk_set_target(EntityPtr, &Vector, Index)// Vector = c-script vector to be set
                                                      // Index = waypoint index
dllfunction mdjk_find_point(EntityPtr, nbPoints)      //  nbPoints = number of points in map
                                                      // returns the nearest point entity sees
```

NEW

```
dllfunction mdjk_ent_see_point(EntityPtr, Pt2)          // Pt2 = point number (and not index)
                                                        // returns kYes or kNo.
dllfunction mdjk_set_target_num(EntityPtr, &Vector, Pt2) // Vector = c-script vector to be set
                                                        // Pt2 = destination point number (and not index)
```

After that, you can use functions between « open DLL… » and « close DLL » lines

Example :

```
// MdJk_DLL.dll: function call ----------------------------------------------------------------------------------------

// Variable used  in DLL test

        var     nextPoint ;
        var     pathLenth ;
        var     maxIndex ;
        var     aPoint ;
        var     aError ;
        var     aVector[3] ;
        var     MdJk_Handle ;
        …

    MdJk_Handle  = dll_open("MdJk_DLL.dll");        // this should be done only once after level load

        nextPoint = mdjk_next_point(me, 1, 6);      // this is the main function,
                                                    // I am under Point 1, I want to go to Point 6
                                                    // returns the first point to join
                                                    // store all waypoints  into an internal DLL array
                                                    // attached to the entity pointer « me »

        pathLenth = mdjk_distance(me);              // returns the minimal distance
                                                    // between Point 1 and Point 6

        maxIndex = mdjk_count(me);                  // returns index max of internal DLL array
                                                    // for current path
                                                    // Point index 0 is the starting point

        aPoint = mdjk_point(me, 2);                 // returns the point number strored by DLL
                                                    // at index 2 of the path array
                                                    // TAKE CARE : it means that( maxIndex >= 2)

        aError = mdjk_set_target(me, aVector, Index); // find point with  index
                                                    // set aVector[0] to point.x
                                                    // set aVector[1] to point.y
                                                    // set aVector[2] to point.z
                                                    // returns point state ( opened or closed)

NEW     aError = mdjk_ent_see_point(me, 5)          // find point 5
                                                    // returns kYes if me sees point 5 (kNo if not)

NEW     aError = mdjk_set_target_num (me, aVector, 5); // find point 5
                                                    // set aVector[0] to point.x
                                                    // set aVector[1] to point.y
                                                    // set aVector[2] to point.z
                                                    // returns point state (kOpened or kClosed)
```

dll_close(MdJk_Handle); *// Note, if you use the predefined dll_handle,*
// you can't open two DLL at the same time.
// This should be done only when level change

//----------

CAUTION :

You must call *mdjk_next_point* function at first, this is the main function that calculates the full path from a point to another. Most functions are only available after the main function call. As soon as you close the DLL, all the memory used by DLL is free (unless there is a bug…).

So : you can use *mdjk_distance* , *mdjk_count* and *mdjk_point* only **between**

MdJk_Handle = dll_open("MdJk_DLL.dll");

…

MdJk_Handle = dll_close(MdJk_Handle);

and **after**

nextPoint =mdjk_next_point(aEnt, Pt1, Pt2);

If you want, you can keep track of a full path by reading all points into a WDL array by reading each point from index *0* to *maxIndex*. In this case, calcul of the path will be done only one time.
Use *mdjk_set_target* function to set destination vector. Testing return value you'll know if the destination point is closed or not.

4 Errors value

Here is a list of errors or states that this DLL might returns :

```
kErrorNoWay = 0 ;
kErrorFunctionWDL = -1 ;        // C-Script function not found
kErrorStartPoint = -2 ;         // Starting point not found
kErrorTooComplex = -3;          // Path too complex
kErrorDestPoint = -4;           // Destination point not found
kErrorIndexNeg = -5;            // Index must be positive
kErrorIndexMax = -6 ;           // Index too big
kPointOpened = -7;              // Point opened
kPointClosed = -8 ;             // Point closed
kErrorYouAreHere =-9;           // Starting point = Destination point
kErrorLocked = -12;             // Entity is locked in a room
kYes = -13 ;                    // Yes
kNo = -14 ;                     // No
kOutOfLimit = -15 ;             // Point number is less than 1 or more than 999
kOutOfSkill = -16 ;             // Skill is less than 1 or more than 48
```

NEW

~~A good result is always positive (except for the point opened or closed), when zero or negative, we should do something.~~

5 Remarks

~~5.1 Wait()~~ *Corrected in Version 1.2*

~~So far, DLL has only one Array to store current path, that means that in if we write an action like this :~~

~~*action do_something*~~
~~*{*~~
~~*…*~~

~~*dll_open("MdJk_DLL.dll");*~~

~~*nextPoint = mdjk_next_point(Pt1,Pt2);*~~

~~*…*~~
~~*wait(1) ;*~~
~~*}*~~
~~the path from Pt1 to Pt2 we got stay available if there is only **one** entity that use the DLL . If there is another entities that use DLL, the *wait(1)* instruction could give hand to this other entity that might need to calculate its own path by calling *mdjk_next_point(Ptx,Pty)*, doing so, waypoint array change, and coming back to first entity path from Pt1 to Pt2 is nomore available.~~

~~So :~~
~~If there are several entities using DLL, call *mdjk_next_point(Pt1,Pt2)* inside the frame cycle or store all the path in an c-script array attached to your specific entity. Always remember that path is lost after a *wait()* or *waitt()* instruction~~.

5.2 WED

As we use normal ~~entities~~ models (V 1.3) in WED, there is no tool to set links between points, we have to edit properties and set skills values, in this case, it's very easy to forget something. If your character arrives under Pt2 and if you forgot to set Pt2 properties in WED, that means that from Pt2 you can't go anywhere. Pt2 is a dead point…

5.3 Limitations

Never put more than 999 points in your map it is a DLL limit. By the way, the complexity of your level is the first limit…

5.4 Some ideas

Using flag1 for a given entity used as a point (for example *Pt_n*), you can easily close or open a door. You just need to set a point above this door (*Pt_n*) and when you close or open door, set *Pt_n.flag1* to *on* or *off*. (remember *flag1=on* means point is closed). When you call *mdjk_next_point(aEnt, Pt1, Pt2,* this flag is read by DLL.

mdjk_set_target(aEnt, aVector, Index) is a function similar to *ent_waypoint*, but if you look at its return value, you can see that here, your character can discover that this next point is suddenly closed ! So do something…

5.5 Not yet tested

Two new functions are in the DLL not yet tested…

mdjk_get_skill(numPoint, numSkill) ; *// get skill numSkill for point numPoint*

mdjk_get_random_next(numPoint) ; *// get a random point reachable from point numPoint*

Regards,

MadJack.