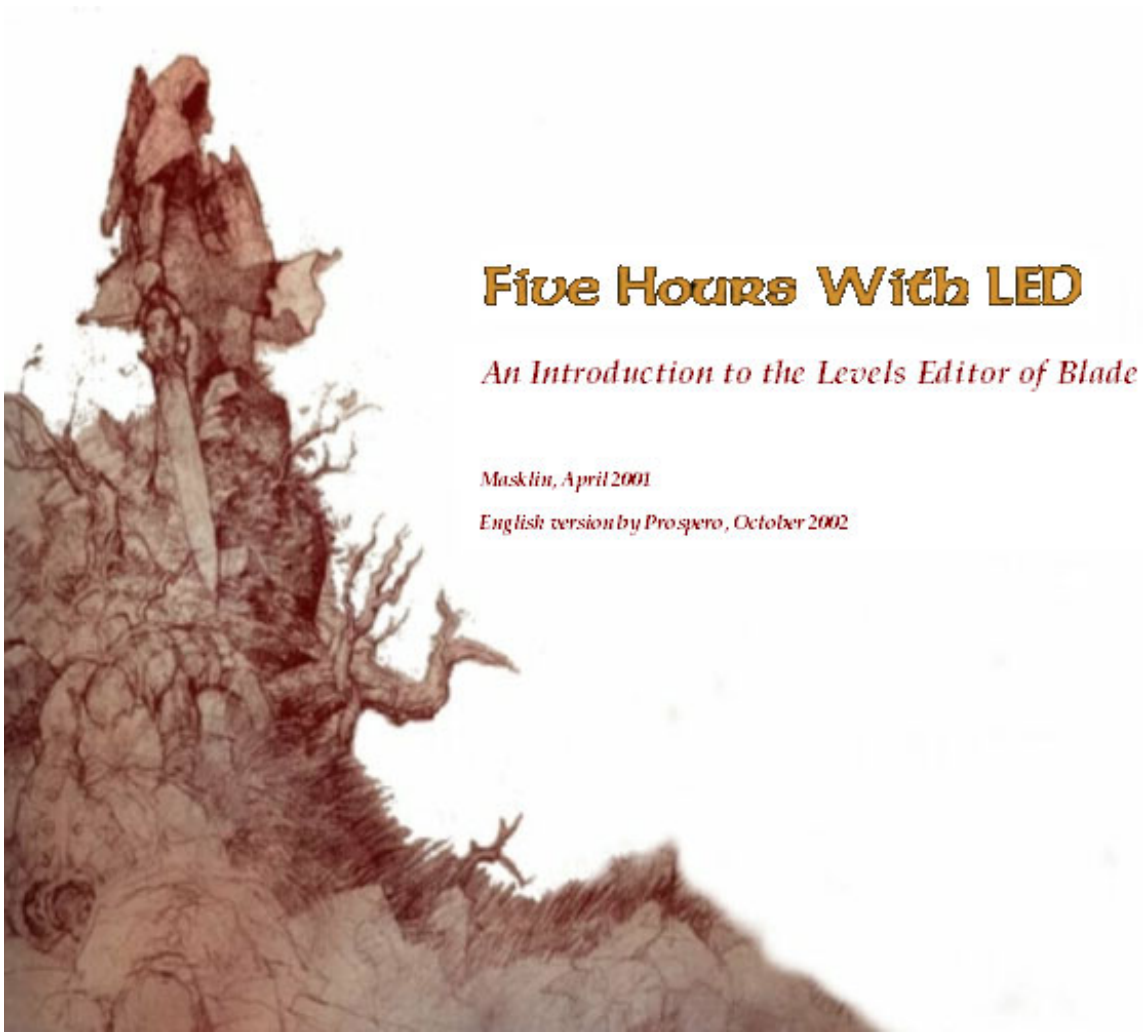


Five Hours With LED

An Introduction to the Levels Editor of Blade

Masklin, April 2001

English version by Prospero, October 2002

Index

Introduction.

LED is the Level Editor used by RAS to design all the game maps for *Severance: Blade of Darkness* (a.k.a: *Blade: The Edge of Darkness*). The editor has been released *'as is'* and without any type of support. Now RAS is closed and all the support comes from fans on the forums.

The first time that you are faced with LED it is usually quite frustrating. A Readme file with a few lines is the only initial help available and I assure you that to obtain the first results is a true test of willpower.

But after those first hours of desperation you begin to like the LED and the results begin. In fact to ensure that many of us stay on the right road I thought that it would be good to publish a small tutorial to help you during those first hours (I hope they are many less than five).

I have tried to be thorough with the content of this tutorial, but I am sure it contains many errors. When in the text I say "it is made this way", you should add "probably". Most of the statements are tested, but not everything is what seems. I would thank you to send me all the corrections, suggestions, criticisms, that you like.

The RAS Tools.

Once we have decompressed the tools of RAS we will have four directories: 3Dmodeling, Compilers, Config and Led. To work with Led we will only need two of these directories: Led, which is the actual editor, and Compilers that contain two compilers, one for the maps and another for the textures.

A Blade level is composed of a basic group of files. Since this is a basic tutorial we will try to use all the resources that RAS has left in its directory structure. We will forget the possibility of creating new textures, objects, animations, etc.

To be able to use a level in the game we will need, at least, the following file types:

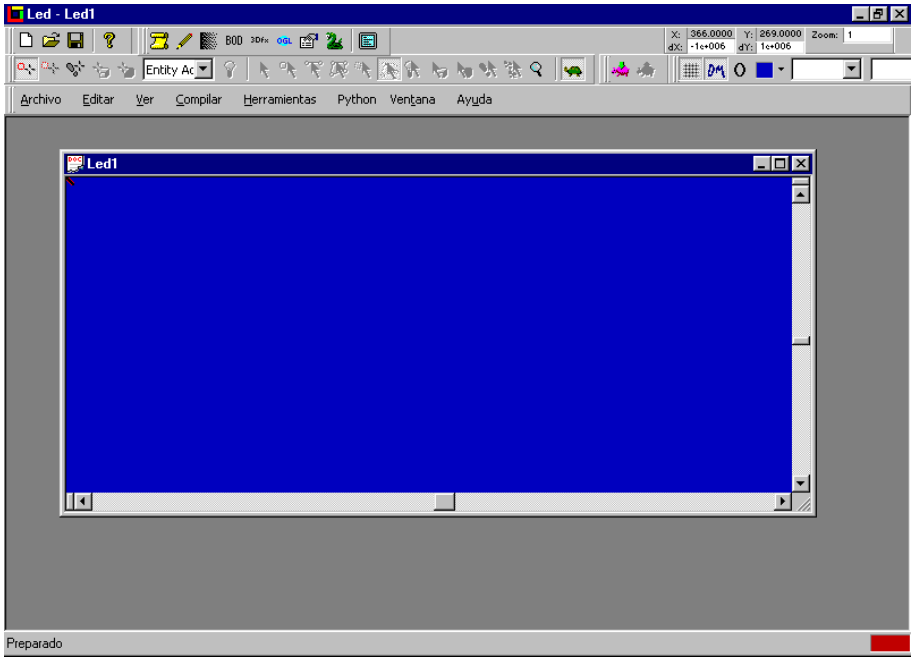
- A file with all the architectural details of the map: walls, doors, lights, etc. It has the extension bw.
- A file containing the textures used as background of the map. It is the surrounding landscape, sky, mountains, etc that we see in all the maps. It is usually a file with the name levelname_d.mmp (d stands for dome).
- One or more BOD files with the definitions of objects and characters of the game that inhabit our level. We will usually use the bods that come in BOD\3DObjs and BOD\3Dchars.
- One or more mmp files with the textures that we use in the map. We can have a file levelname.mmp with all the textures or only a few textures. We can also use the texture files of the original maps of Blade, such as mine.mmp or ragnar.mmp.
- A lvl file to load all these files.

If we also want to play with our level in Blade we will need additional files programmed in python that initialize the level, make the objects, characters, etc, but this will come later.

For our tutorial we will create a test sub-directory inside the Led folder. In this sub-directory we will create the whole structure of the files mentioned before while we progress in the design of our level.

First Steps With LED.

To start up the editor double-click on the Led_d.exe executable. The LED has the standard structure of the windows programs with several work areas, several toolbars and a menu:



The initial distribution can be different from the one shown above In the LED it is possible to move, hide and customize all the toolbars. This is the distribution that I use.

> If you want to return to the initial configuration of the LED you must delete the file Led_d.ini that you will find in the Windows folder.

I won't describe all the toolbars that appear, menu options, keys, etc. because I leave this work for the gentleman who wants to write a reference manual. But if you are not still convinced, there is an irrefutable argument: I don't know how to use all those buttons.

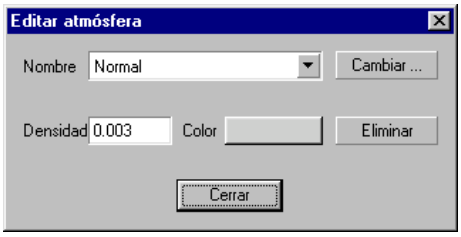
Atmospheres.

So lets be practical. To begin we pay attention to the Readme file, for although it's small it's comprehensive. The first step is to change the default atmosphere. An atmosphere gives the engine the details of the density and colour of the environment

(fog). As we will see later on, a map is made of one or more sectors and each sector can have a different atmosphere associated with it.

In the LED there is always a defined atmosphere that is applied by default to all the sectors. Unfortunately this atmosphere has a density of 1, what it means that in your map you won't be able to see anything. It is necessary to change this and to create a clearer atmosphere.

We choose *View->Atmospheres*. Select the atmosphere Def. If we want, we can change its name by clicking on it. Then we double-click and a window will pop up in which we can select the colour and the density: a value between 0 and 1. A density of 0.002 is good for a start. I leave the colour to your disgression, although you can use RGB: 210:210:210. (Neutral light grey)



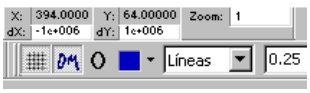
Coordinates.

LED works with a classic XYZ coordinate system. The XY coordinates are represented in the editing window. The Z coordinate sets parameters like height of the floor and roof in the property window. The unit in LED is equal approximately to a meter.

In the real game an identical coordinate system is used, but the unit is approximately the millimeter. Therefore, when we work with the python scripts, a measurement of 1000 units is equal approximately to 1 meter.

The Grid.

The LED includes a grid that helps in the layout of sectors. To see it, click on the button *Show/Hide grid*. There are two grid styles: lines (Líneas) and points (Puntos). You select the style from the combo box located to the right of activation button. Next there is another field in which we can indicate the size of the grid. A good size is 0.25 that it allows us to locate vertexes comfortably.



The grid is, at the start, only a guide. However, if we activate the button *Activate/Deactivate grid* then the LED will force us to locate all the vertexes at the intersections of the grid. The walls will be easy to locate and it will be easier to coincide two or more vertices. In special situations the grid can be disabled to allow us to draw freely, but in general, is better to enable it.

Zoom.

The LED allows us to work with zoom levels between 0 and 75. An appropriate setting to design a medium size room is 40. For wider spaces you can use a zoom of 20.

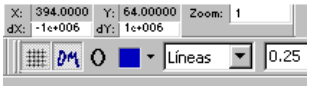
If you have a wheel mouse, you can modify the zoom level with it, although it is sometimes impractical as it can jump about somewhat. In general it can be more comfortable to write the level wanted in the frame that appears in the toolbar. After changing the value press Enter so that it provides effect. As we will see, this operation of pressing Enter will always accompany us in our work with LED.

> The – and + on the Keypad also operates the zoom …….Pro.

Positions.

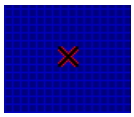
One of the more confusing aspects of the LED it is the character's initial positioning. If this is not set correctly, the character can appear in a solid area and movement is not possible. As consequence we only see a tremendously frustrating black screen.

In the toolbar we find fields that indicate the current position of the cursor. The positions change in multiples of the size of the grid if this is active.



The available space to carry out the map is very wide and, like we have been able to see in the game, the levels can be very extensive. Probably nobody who develops mods for Blade ever worries about the size of the map. However, it is convenient to begin to build around the origin point of the coordinates.

To get the editor window centred in a certain position you must r-click and select to *Centre At*. A dialog window appears with the 0,0 coordinates. Press OK and then, we click on the edit window so that it refreshes and you get a cross red mark on the grid.



This mark indicates the beginning character's position when the level is played in Blade and the initial camera position when we try to visualize the map in the 3D viewer.

When beginning the design of a level, the starting position is 0,0,0, but we will be able to change it in any moment using the option *Tool->Initial point*.

The orientation is not important in the first steps of the design of a level. The position can be more critical because the camera can be located in an invalid position. If the camera is in a solid area we won't be able to see anything.

To find a good initial position we move the mouse to the desired position and we notice the coordinates show in the toolbar. The Z coordinated indicates the height. A positive value is up from zero and a negative value down. An appropriate value of Z for the camera can be 1.75. If the map is centred around the origin of coordinates we can leave X and Y at zero. Although better to set something like 0.1 because sometimes strange things happen with a zero setting. So we fix the coordinates to 0.1, 0.1, 1.75. For the orientation choose 0, -100, 0.

> As we have commented before, the LED includes a cross marker that should indicate the initial position. Unfortunately it confuses the coordinate Y with the Z and also it confuses us. Please ignore this marker.

The Design Of The First Sector.

Concepts.

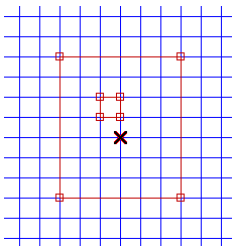
A room with four empty walls is the first step in the design of levels. It is probably the most difficult.

The first challenge is conceptual. In LED you don't build the walls but rather dig holes. If you have played Dungeon Keeper you know of what we speak.

Initially we have a solid space, an enormous cube, in which we will dig our map. To dig tunnels seems intuitive, but to dig an entire landscape can seem somewhat strange.

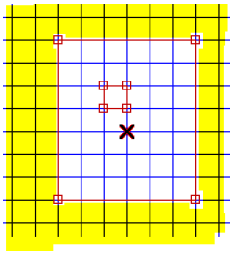
The holes are made by inserting vertexes, enclosing an area that we call a sector. A sector is a closed polygon that defines a surface with certain properties that we will see later on. We have to think of each sector as a new hole in the map.

For example, a basic sector is a square that forms a room. Up to this point everything it is very straightforward. If we want to add a column, it is natural to think of a putting in a new smaller sector, inside the first one:

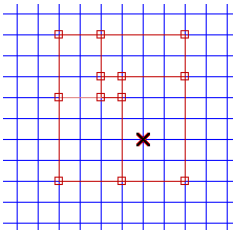


But the LED doesn't work this way. In fact, what we are doing is digging a hole in an area that is already hollow! The area that defines the external square is an empty cube. Putting another cube in this area will have no effect on the map.

We can see it clearer if we colour the solid area to distinguish the holes:



If you still don't grasp this concept, try it and see the result. Let us make the first room and then we will continue. It is not as complex as it may appear. If you are curious to know a way of creating a column, here is a solution using four sectors:



Get the idea?.... But now is not the time to be thinking of columns...

Building The Map.

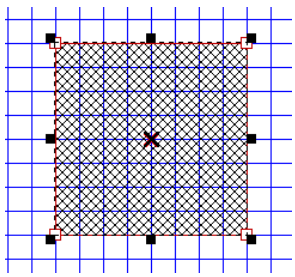
Now we have been introduced to the basic concepts we will build our first sector. Select a zoom of 20 and a grid of 1. We centre the map and click in the window. We will see appear a white point. This is one of the vertexes (corners) of our room.

A vertex has been created because by default the *Add vertex* tool  is active. We continue by adding another three vertexes until a polygon is formed more or less square. Don't try at the moment to be imaginative with the form of the sector. That can be hazardous.

The last vertex is superimposed on the first one so it closes the square and our first sector is made. We will see that the colour and the form of the vertexes change to indicate to us that the sector is created.

> If by mistake we have begun to put vertexes, or the sector is not staying like we wanted... you must finish creating the sector. Once created you will can to delete it, as we will see soon.

Once we have created the sector we use the *Select Sectors* tool  and we click on our sector:



With the sector selected we can delete it, or move it, or change its size as if it were a MS-Windows window.

You can use the *Select Vertex* tool  to change the form of the sector if it is not square. Click on one vertex to drag it to the position you want.

> Sometimes, It is not easy to displace a vertex. If the displacement is small the LED cannot find it. Is better to move it far and then to take it back to the required position.

You can select simultaneously more than one vertex by clicking on a point anywhere on the map and dragging the mouse. A line that indicates the multiple selections appears. This is very useful when we want to move walls or complete groups of walls conserving the geometry.

> The space key changes from the select vertex tool to the insert vertex tool and vice versa. Very useful when you are designing.

Now we select our sector and we click on the *Properties* button  or press the V key. The Property sheet window appears.

The property sheet has seven tabs. In the General tab we can set the height of the floor, of the roof and name the sector.

If we want to create sectors of different heights, stairs, maps with several floors...) we will be able to change the height of the floor giving a negative value to lower the height or positive to go up it. In our first sector we will leave it in 0.

The height represents the size of our cube. A height of 4 is reasonable for closed rooms. If our room becomes a palace one day we should think of setting higher roofs.

LED generates names automatically for all the sectors that we make. It is comfortable to maintain these names because they cannot repeat in the map and when we have near 3000 rooms.... however, now that we are beginning it can be useful to give it a name such as *Door* or *Window* so it is us more intuitive to work with (mainly when selecting them). Also, you can preserve the sector number in the name: *Door27*, *Window345*.

The Atmosphere tab allows us to select an atmosphere for our sector. As in our map we have only created one atmosphere, this is assigned in a predetermined way. Later on we could play with the assignment of different atmospheres to create more realistic maps. At the moment, to our first sector doesn't need it.

Now, we have our first map, and we should save it. Choose *File->Save* and give it the name tonta.mp. It is convenient that we create a sub-directory 'test', in the Led directory and that we save our work in it. Don't use spaces or long names in the directories or in the names of files. To compile we need to work in Dos and that would give us problems.

> The LED has an annoying habit of saving every time that we bring the pointer near the menu File. Now most of the time it is ok, but other times you can be quite irritating.

Can we already see our room? I'm sorry, but we need to talk about textures and some extra files. But the map is complete.

Textures.

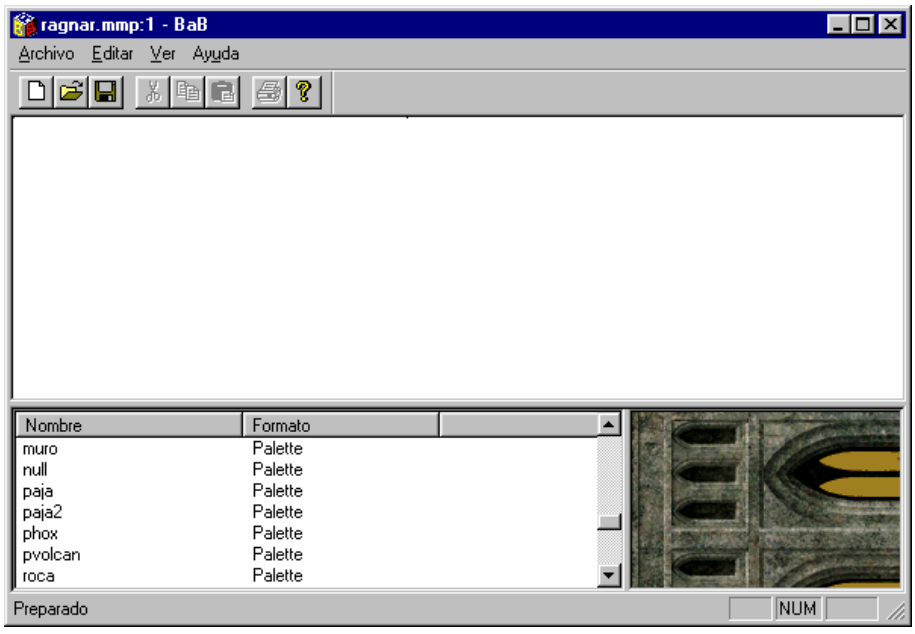
The walls, floors and roofs of the cube made earlier don't have any applied textures. A texture is a graphic that repeats over a surface to give it a more realistic aspect. In Blade, the textures are created by using 256x256 bmp files, but we won't explain those details at this stage because we will use the textures defined in the main game: there are a lot and they are of excellent quality. If you want to put a picture of the Simpson's on the wall you will have to contend with the creation of textures.

The textures are stored together with the colour palette used in a file with extension mmp. You can have as many mmps files as want, but is a common practice to use one unique file for each map. Inside an mmp file the textures have an internal name that we will use in LED to assign them to any surface.

The Texture Compiler.

In the *Compiler\Textures Compiler* folder you can find the texture compiler: Bab.exe. I have already said that we won't actually create textures, but the bab it is an excellent tool to visualize them.

We start up bab.exe and two windows appear. One associated to the bab and another to the console. This last one is not needed, as we won't be compiling yet.



Now open a *mmp* file from any folder into Blade Maps directory of your installation of Blade. Better still; make a copy in our test directory.

When opening an *mmp*, for example *ragnar.mmp*, you can see the names of the textures that it contains in the lower left panel. By selecting a texture name and double-clicking, this is shown in the lower right window. Now we can look for the textures that are suitable for the style of the map that we want to create.

We can load simultaneously several mmps in the Bab and to see all the textures at the same time although this can be somewhat confusing

For the first sector any texture is valid, for example bronce. We must remember the names because we will need them in the LED. The texture names are not always descriptive. The texture paja (straw) is shown in the image; the texture roca (rock) is a wooden surface. In short, the textures don't always have appropriate names, even in Spanish.

Once we have memorized the names of the textures that we like, can leave the compiler.

Texture Loading in LED.

Once we have chosen the textures, you must indicate to the LED that we want to use them. There are two basic methods:

- To drag the bmps files with the textures to the editor.
- To make a .dat file with the necessary information.

It seems clear that the first method is the easiest, but we will use the second. The fundamental reason is that we don't have the bmps of the textures. Up to now there is not any well-known form of extracting the bmps starting from the mmps.

> When this tutorial was published we didn't know another method. At the moment there is an easier way of working with textures. Please read the second tutorial: *Gates*.

The other method consists on creating a file test.dat in our directory with the following content:

```
1
NULL.BMP
NULL
muro.bmp
muro
paja.bmp
paja
bronce.bmp
bronce
roca.bmp
roca
.....
```

The first 1 is obligatory and the meaning is hidden inside the minds of RAS people at the moment. Next is the name of the bmp file with the texture and their internal name. It is not difficult to create these files but is somewhat tedious. Fortunately, SlayerDDD has done the work for us for all the mmps of all the levels of Blade. Go to the web page Led Editing and grab the dats.

As we will use the textures of ragnar.mmp we copy this file and the file ragnar.dat from SlayerDDD to our directory test. If you rename ragnar.dat to test.dat, we can eliminate textures that we won't use or even to add mmp textures corresponding to another level.

This second method has a problem: since we don't have the .bmps. The LED doesn't include samples of the textures, so the assignment of textures is made blindly. This may seem bad, but in practice the textures will become familiar. At the end you finish up working with three or four textures and it is not necessary to visualize them in the editor. Also, as we will see, this limitation doesn't affect the 3D viewer so the aesthetic aspect of the assignment of textures will be made in this viewer.

> Again... please read our second tutorial to details for an alternate method.

Once the file test.dat, is available, load it in the LED by choosing the option *File->Import Textures*... and we open test.dat.

> In the open file dialog nothing will appear unless we type *. dat in the field File name.

Once imported, the textures can be seen by clicking on the *Show/Hide the palette of Textures* button  :

Umm...this is not altogether true. Because we don't have the bmps LED gives us a beautiful collection of skulls. In short, we close this window and we open our map: *File->Open*... tonta.mp.

> LED opens a window Led1 whenever it starts. Close this window before opening our map. The most comfortable way to open a map in LED is to double-click on the file tonta.mp. LED loads the map directly and sets the current directory as the default directory. Very useful when visualizing the map in the 3D view.

Textures Assignment.

We set a 20 unit zoom and we centre at 0,0, as usual. We choose a line grid of 0.25. We activate the selection tool and we refresh the screen by clicking on the edition window. Regrettably you must repeat this procedure whenever you start work on a map (may be the *File->Setup* option doesn't work) so I won't repeat it more.

We select the sector and we show the property sheet (key V). Then, we choose the Floor tab:

In this window a combo box appears with the list of all the available textures. We choose *lospie*, a texture that is appropriate for a floor. With the same texture you can get very different results varying the parameters of this tab: X-Y coordinated, angle and zoom. For the moment we limit ourselves to enlarging the texture up to 10, setting this value in Zoom. Later on we will have the opportunity to do more complicated assignments of textures.

And press Enter.

> Have you pressed the Enter key? I know it, it is not too standard in Windows, but you must press Enter for your values to take effect. LED requires that you press Enter to validate many properties. It is a very common failing not to do it and then to believe that our changes have not worked.
> (If you try to set an impossible reading, say with the floor higher than the ceiling then the settings will not take even after pressing Enter. For this reason it is a good habit to set the ceiling height first……Pro)

Now we select the tab *Ceiling* and we choose the texture *techo* (roof) that seems very appropriate. Zoom of 10 and Enter.

To be able to apply textures to a wall we must select it by using the *Select walls* tool  . After selecting a wall, open the property sheet (if it is not previously opened).

We choose the texture bronce (brass), the typical stone wall of the castles, with a Zoom of 10 and an angle of -90. We press Enter. We repeat the operation for the four sides of our box.

> It is possible to rotate a texture by pressing the T key. If the current angle is, for example -90, it passes at -180.
> On the other hand, the R key breaks a sector eliminating the selected side. It is useful when we want to modify the form of our sector substantially.

We can select more than one wall by holding down the Control key. Or all the walls that compose a sector by selecting the sector and r-clicking to choose the option *Select->Edges*. When we have selected several similar objects, the properties appear with an * if they are not the same in all the objects. If we change a property, the new

value is applied to all the selected objects, but the rest of attributes remain unchanged.

> And if I don't assign textures? LED use the default texture. Sometimes, the default texture is the NULL texture, a texture that should always exist but it should be used with respect. Because it doesn't mean that the surface is transparent, but rather that makes it so that the landscape is seen. What is logical in a window can be very confused inside a room. We will see it in the next tutorial. For the moment it is enough knowing that if we see the landscape where we don't expect it , we have probably forgotten to assign a texture.

Now, it is time to save and compile our map.

The Map Compiler.

The LED saves the maps in mp format, but Blade game can only read them in bw format. The map compiler carries out this conversion. It is in the directory *Compiler\Maps Compiler* and its name is bc.

It is a MS-DOS tool and so doesn't manage correctly the blank spaces in the names and it is intolerant of long names. Keep this in mind when naming our map.

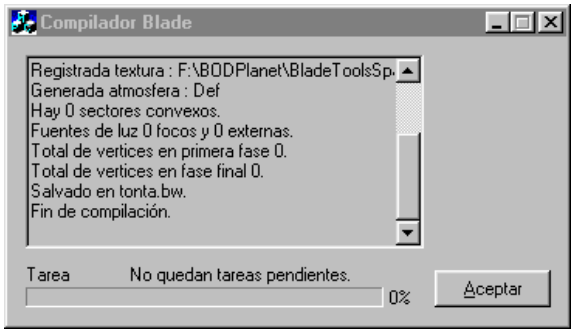
The problem is that the compiler is in the directory "*Maps Compiler*". Oh! It has a space. You can copy the bw to the compiler's directory, or copy the whole compiler to your working directory or to test with the short names. This seems the most reasonable thing.

To compile we open a DOS window and we cd into our test directory . There we will have the file bw. We execute the following sentence:

```
..\..\compil~1\mapsco~1\bc tonta.mp tonta.bw
```

> The conversion of long names to short in your Windows could be different. If it doesn't work you can use dir command to call Dos to the Compilers*Maps Compiler* directory. It is a good idea to create a file compile.bat because we will compile many times.

When the compiler is executed two windows will appear. One of messages with errors about película (movie) that we ignore and another with the compiler of Blade.



We only have to press the *Accept* (OK) button. If everything has gone well we will have a file test.bw in our working directory.

I suppose that you will want to see it. To see it in the game you only can use BODLoader 0.5a or superior that I heard is very good. But this is a LED tutorial and

our first sector is too silly to play with yet... so, we will continue with the LED and its 3D viewer.

> In the English version of the map compiler, the python15.dll is missing! However there is one in the textures compiler, which can be copied……..Pro.

The 3D view.

LED includes a specialized version of the engine of Blade so it is possible to visualize the map, as it will be seen in the game without leaving the editor. Two versions of this tool exist: a 3dfx and another OpenGL. We will use the OpenGL.

The Level File (lvl).

As in the main game, the LED 3D viewer needs to know certain information about our map before being able to render it. This information should be given in a text file called *levelname.lvl*, in our map, *tonta.lvl*.

To create the file *tonta.lvl* we open any text editor and we include the following text:

```
Bitmaps -> ragnar.mmp

World -> tonta.bw
WorldDome -> casa_d.mmp
```

In that file we see the path to the textures file (*ragnar.mmp*), the name of the compiled map (*tonta.bw*) and the name of the dome (*casa_d.mmp*). This last file contains the texture used to show the landscape. In our example we use the dome of Blade Characters' selection level (*BOD\Maps\casa*). We should copy this file to our working directory.

Finally, Our Map.

It is the moment to press the OpenGL button 

. You can see a message about beta state of the 3D viewer that will come as no surprise, and we open our level file tonta.lvl.

Ohhh...! It is simply amazing! A beautiful *first sector*. Ok, the textures are somewhat repetitive, a little uninspiring, and where are the doors? And isn't it said that Blade has fantastic illumination?

> If you see just a black screen, don't worry. Return to the topic of Atmospheres and the starting position. If you don't understand it, continue reading, for later on we talk again about positions. And you can always grab the map from here.

OK.... this map is not ready for publication, but we have already made something. Now we will try to take advantage of the 3D view.

In the first place it is necessary to learn how to move. To move left/right and forward/back you can use the O-P and Q-A keys or the traditional arrows. You can change the inclination of the camera with the W-S keys.

In the 3D view, we are not moving a player, you only move the camera. You will be able to cross walls if they are not too thick. If we cross a too thick solid area it seems that the engine is blocked. We will sometimes be able to go behind the wall, but other times we will have to jump to a new position. You can r-click and choose the option *Set Start* to select a new position. We will be able to give any coordinate / orientation.

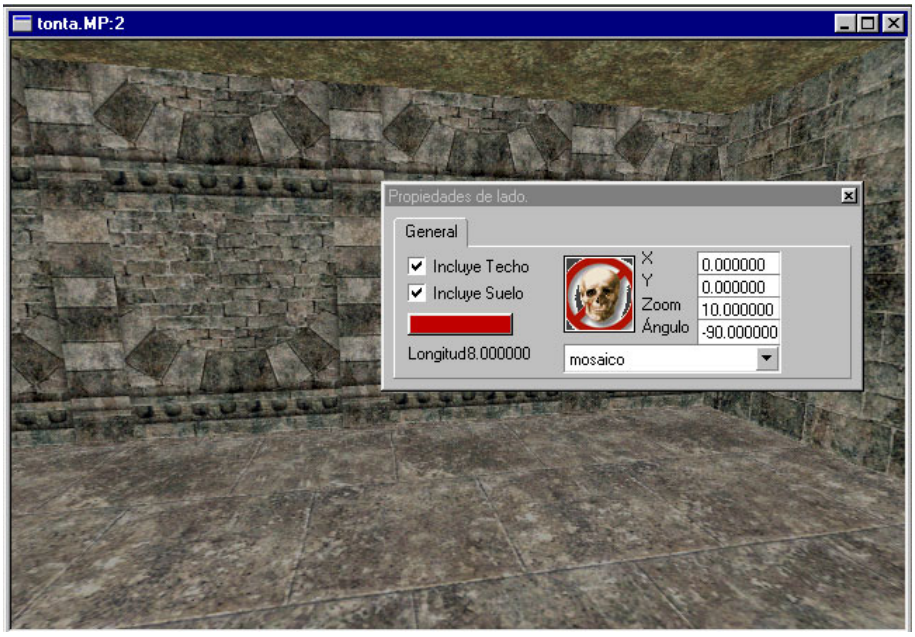
> You must try not to do big jumps between coordinates because you can end up disoriented. If you appear in a solid area you will be blocked and the screen can turn black or not refresh. Use Set Start to change to another position again. These changes aren't saved in your map. If the starting position that you used has been a failure, look for a good position with this option and don't forget to change it in the original map.

Texture Assignment.

I suppose that you will already have got bored with playing with the first sector. We will make some changes to see how they are reflected in the 3D view. If you don't have the windows maximized then you can change from the Editor window to the 3D view at any time. It is important that you don't close the 3D window, because the

associated button only works once. If you close it, you will have to exit from the LED and open it again so that the 3D view is functional.

This being clear, we move to the editor window and we select the north wall. Enable the property sheet (V) and change the texture to *mosaico* (mosaic). I suppose that you won't have forgotten to press Enter so go to the 3D window and. surprise! The north wall has a new texture now. As we might expect *mosaico* (mosaic) isn't a mosaic, rather some arches. But the interesting thing is that we can maintain the property sheet enabled and change textures (and their properties) and to see them directly in the 3D window. Maybe, it was not so bad not having the bmps...



Have you already noticed that the wall blinks? That is because it is the selected surface. By r-clicking on the 3D window you can enable the option to *Select Surfaces*. Now we can click on any surface and to change the characteristics of the texture in the property window. For example, you select the wall with the supposed mosaic and change the Angle to 90 degrees. Press Enter and click on the 3D window so that it refreshes. The arches have now been rotated.

But there is another method to change textures in the 3D window. Close the property window so that it doesn't intrude. Make the 3D window active and press the M key. You will see the name of the texture and all their properties on screen. Now we can control all the aspects of the assignment of textures with the keyboard (numeric block enabled):

- Keypad + - to increase / decrease the zoom level
- Keypad Start (7) - PrevPag. (9) To rotate the texture.
- Keypad cursors to move the texture.
- Keypad / - * assign previous / next texture
- Extended Start – End to assign first / last texture
- Extended PrevPag – NextPag to fast setback / advances of textures.

Hi, have you returned? I know it is amusing. There are a lot of textures... and we are only using *ragnar.mmp*. But this is very promising?

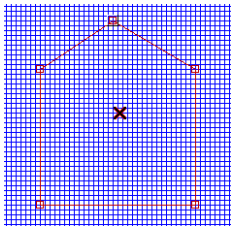
This method is comfortable to select a good texture and to choose the best zoom combination, angle, In the practice it is quicker to assign the textures and properties from the property sheet, mainly when textures are assigned to multiple sides.

> If the design is confused and we don't find an easy way to distinguish the surfaces (I hope it is not happening with this first sector) the X key draws the polygons that compose all the sectors. The C key is also very useful, for, besides the frame rate, it shows the tris number, something that no designer of maps can forget if he wants to sleep peacefully.

Recompilation.

The synchronization between the 3D view and the editor is brilliant. It is a pity that it only applies to the textures. If we make any change in the structure of the map such as to add a vertex, or to modify a sector, then you must to recompile the map.

Let us return to the editor window and let us select the north side. We position the mouse approximately in the centre of the vector and r-click. We choose to *Insert vertex.* A new vertex appears. We select it and we drag it a little up. The sector will be something similar to this:



> The sectors cannot have any form. They should be convex. To LED a sector is convex if you can draw a line from any of the vertexes to all the other ones without leaving the sector. In another way: imagine a light on each vertex, if from each vertex you can see all the lights then the sector is convex. If you have doubts, you can select the sector, r-click and choose *Divide in convex.* If the sector is not convex, LED divides it in two or more convex sectors.

Now we save and recompile the map. I suppose that you will have your Dos windows at hand. You already know from using the bc. Or you can double click on your compile.bat. Once generated the new file with the compiled map (bw) it is necessary to tell to the 3D window that the map has been modified. We r-click and choose the option *Actualize bw*. LED requests us a file and we choose tonta.bw. There is our as modified map...

> If you save the map but not recompile it, when you actualize the bw in the 3D window it loses all the textures. Does everything appear as a recently bought house (somebody that knows this that is?). Please compile again and actualize the bw to return to our great castle again.

We have created a new side in the sector, the texture of this wall has not still been assigned. Fortunately, the LED has a texture that applies by default. You already know how to change it.

Farewell.

Now, if you have arrived here you probably already have your first sector. Maybe you want to create a second room and a door to it. Or a door to the street , to see the outside. Stairs or windows. In short, design a true map.

I don't know if this basic tutor is useful. If you find it interesting we could continue with another chapter of the saga. We will see if we have time.

Credits.

I get a lot of material of the RAS forums and Bladex and of all the guys that live there.

Especially I am in debt with BirdMadBoy, for his patience when introducing me in the secrets of the LED; SlayerDDD for his advice and excellent Tutorial; NiceMice by his web page; Phaydde for help to the whole community with Bladex; Josh, for so many technical conversations; with Moria people that make you at home; with TacTo, that we abandon him; with the regulars of the Editing forums for their willing help (Pulsar, Duke, Tom Trific,...) and of course with RAS for this brilliant game and especially with Genesis that visit the forum.

Prospero has helped me a lot to complete a dream: to see the tutorial translated to English.

If I have forgotten anyone please send me a mail.