Dark Disciples II Editor – Tutorial

So you have played the Dark Disciples games and would like to build a campaign for them. From what I know (and believe me, I've tried a lot of things), the DD2 editor is probably the most efficient way to create a scenario somewhere between Ultima, Might & Magic and similar classics – it doesn't boast quite the flexibility of the Japanese RPGMaker, but once you understand the system, you will find that you still have more options than you'll likely ever use. Creativity makes the difference!

Creating a new mod

Let's make a clean slate for us to build on. All of the game data is stored in the "Mod" directory, so that's what we need to clear. You may wish to copy the original DD2 data for backup reasons – that's okay, as long as there is one and exactly one "Mod" directory in the DD2 folder itself (but it's totally fine to create a folder "Backup" and copy the original "Mod" directory into it).

"Mod" has three subdirectories:

- "LevelData" - This is where all the content of the different maps is stored. Delete them all.
- "GameData" - This is where all the content of the game objects is stored. Delete everything *EXCEPT* for the "item.dat" and "custom.bmp" files. All the other files will automatically be recreated when you start a new game with a mod where they are not present.
- "SaveGame" - The savegame folder, indeed. Delete everything. As is the case with the "GameData" directory, the files will be recreated. You also don't want to mix up savegames between different mods.

Now start DD2. Enter the editor via Ctrl+U from the main menu. You will face a blank level.

Creating Levels

The foundations of any game are its levels. We can divide the level creation process into two distinctly different steps: Drawing a map and populating it.

Drawing a map

Let's start off simple, maybe with a small castle. As you may recall, each map in DD2 consists of a grid of 50x50 fields. While it is possible to walk over the edge of a map onto another map, it feels much neater if you do so sparingly, usually most often in overland areas. For a castle, it thus makes sense to place the entrance gate at the bottom of the map. Navigate to a place that suits you with the arrow keys; you can always determine your position with the values at the edge of the map screen.

! A map in DD2 is made up of five *completely separate* layers:

- Bottom graphic layer – where you place graphical tiles
- Top graphic layer – where you place graphical tiles that will be shown on top of those in the bottom layer
- Access layer – where you will define whether a player can pass or look through certain fields
- Zone layer – we can ignore this at first
- Event layer – where you place everything from monsters to treasure chests to traps and so on

Now to place some tiles on the map, click on one of the eight gray squares in the upper right of the screen, then select the “T” icon or press “t”. A wide selection of map tiles will come up, roughly grouped by their function. The important concept here is, again, that the layers are completely independent, meaning that just because you place a graphic on the map that looks like a wall it doesn't restrict the player's movement in any way by itself. That's what the access layer is for.

To place the selected tile on the map, just click on the desired field. Note that the actually highlighted tile is placed; you can have up to eight tiles active for short selection for convenience.

Now, go on and draw a few rooms. Some practical applications for the distinction between “Bottom” and “Up” layers:

- Place some floor tiles (bottom) and some furniture on them (top)
- Place some walls (bottom) and some windows on them (top)
- Place some floor tiles (bottom) and some doorway arches on them (top), but no doors themselves – we'll get to those.

Let the tile selection inspire you – a throne room, a dining hall, a guard chamber – until at some point we have a small practice castle. Don't make it too large, though; apart from this being just an exercise, we'll need about a quarter of the map later on. To keep on overview of things, use the “Map” button below the main area.

Once we have a basic structure, we will define the player's actual movement possibilities. Select one of the eight tile slots and press “A”. A long list of access options will come up. For now, let us stick with the most basic ones:

X	= no movement, no sight. Example: Wall.
x	= no movement, but sight. Example: Furniture.
::	= movement and sight. Example: Free area.
U	= movement and sight, but player is displayed beneath “Top layer” tile. Example: Door archway.

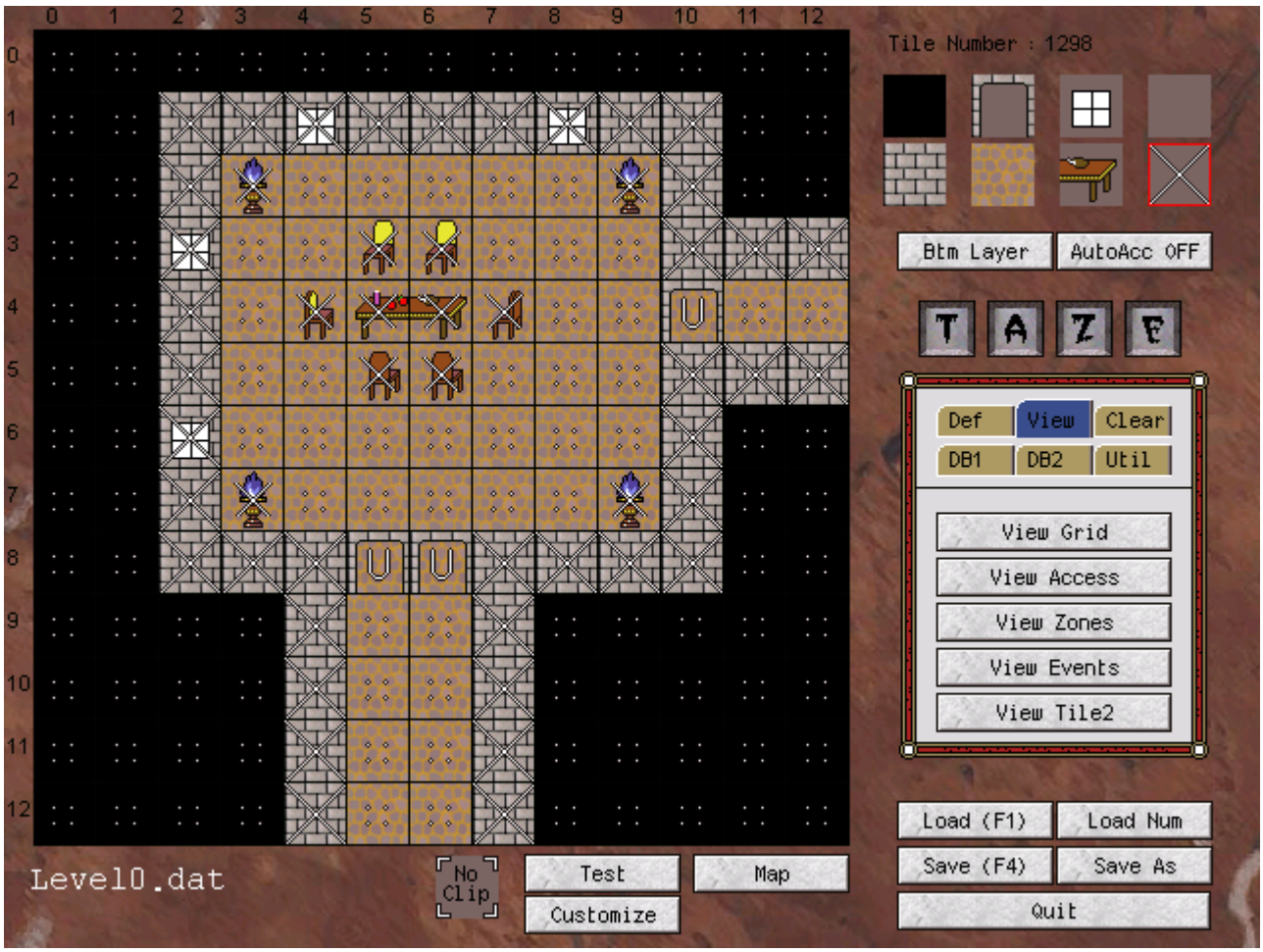
With these four, you should be able to define all the places in our castle for now. On the editing screen, it should look somewhat similar to picture (1) below.

Okay. Want to try out if everything works? Let's go:

Use “Save as” from the toolbar on the right and select a free map except map 0. Map 1 should do. Then determine the field where you would like the player to start on and note its coordinates (for instance, X = 40, Y = 25). Select the “DB1” tab on the toolbar, then “Objective”. This is also where you can later on define an introductory sequence, but for now the only important field is the starting location. Input the chosen coordinates – like Map 1, X = 40, Y = 25 – and save things.

When you quit back to the main menu and start a new game, you should find yourself at the chosen location. If not, the common error reasons at this point are the confusion of the X, Y and map values or a faulty “Mod” directory – maybe there are still the old files present, maybe one file too many was deleted (see the guide to the “Mod” directory above).

However, sooner or later, you will get things working and wander around in the practice castle. Now let's go populating it. That's where things get interesting.



Picture (1) - A part of a newly built castle with access definitions in place (X, x, U, ::)

Placing Events

"Populating" is not meant to be limited to people, I just liked the term. In fact, everything that your character can interact with in any way is called an Event. This includes traps, treasure, NPCs, monsters and so on.

In general, placing an event on the map works much like placing a tile. Select one of the eight tile slots and press "E". You will be shown a huge table of events from which to select.

! This is probably the biggest asset of DD2 – effectively, it has about 80 "QuickEvents", where even RPGMaker VX Ace has only 4! And nearly all of those have several features to manipulate, so you can place a vast variety of funny things in the player's way without a lot of effort. While it may happen sometimes that there is a very specific effect that you cannot duplicate exactly with what is available, there are more than enough options to stuff your game with interesting and challenging events, as is evidenced by the existing campaigns.

Basic event: Doors

For now, let's start small. First of all, we will want some doors in our castle. Select the doors tab and from there [Simple Door]. Place the door event on one of the archways I told you earlier on not to fill with a door tile, then right-click on it.

! Right-clicking on an event lets you modify its various settings. We'll get to the basics of flag settings later, but many events have various rather complex options of flag manipulations which would go vastly beyond this tutorial's limits. The best way to learn about the deeper intricacies is likely to play the DD2 and ToE campaigns and look how things were used there.

You will be able to choose from a variety of doors. They all behave the same way, though – they offer absolutely no resistance to the character (unless you modify the "blocking" flag, of course).

Now let's say there is a room that should not exactly be open to random strangers (i.e. rude and greedy adventurers). Select a [Comp. Door] event. You will again be able to select from the various door images, but the settings that influence how much of an obstacle this door will be are the others:

- Default: The status the door is in at the very beginning of the game.
  - Open: the door is open, but can be closed later.
  - Broken: the door is open and cannot be closed.
  - Closed: the door is closed but not locked and can easily be opened.
  - Locked: the door is locked. This is likely what you'll want your secure doors to be like at first.

Below that we have the parameters that gather the various options of opening the door. A complex door in DD2 can opened in one of four ways:

- Break with strength - "required strength" sets the amount of strength required. Keep in mind a starting character will not have more than 5 points of strength.
- Pick with Pick Locks skill – this stat sets the skill level the character will need to pick the lock. Keep in mind that to use this skill, the character will also need to have acquired a set of thieves' tools somehow.
- With the proper key – click on the key selection to choose the key that opens this lock or set it to "no key". ! You will not be able to add any actual keys to the game's item database, so don't use up half of the about 20 available keys in the first dungeon.
- With a "Knock" scroll – you can choose to make the door impervious to this by checking the appropriate box.

If the door is not supposed to be an actual door, but a portcullis or something similar, you can make the player be able to see through it by checking the appropriate box.

Then, at last, comes the most important parameter – the status flag.

Aside: Flags

! Flags are what holds DD2 scenarios together. Every change that is supposed to be tracked beyond the wandering through a single map or beyond a single game session is tracked with flags. This includes the status of doors (still locked or already smashed), whether a monster is still alive, whether an item has been taken, a trap has been disarmed and so on.

The game can manage up to 8,000 flags, which is plenty. The trick is that these flags are completely interchangeable. This means that you can make a door open when a monster dies just by making both of them refer to the same flag, and so on. There are plenty of options for funny tricks here – don't be afraid to experiment.

However, in most cases, you will want different flags to govern the different events. To make

managing this a little easier, the game has two functions which can be accessed through the "Util" tab on the right side.

Firstly, there's the "Search" option. You can search for just about anything there, but the option you'll likely use most is "Search for free flags", which does just that. You can also search for a certain flag, a certain item ("how often and where can a player find this key?") and so on.

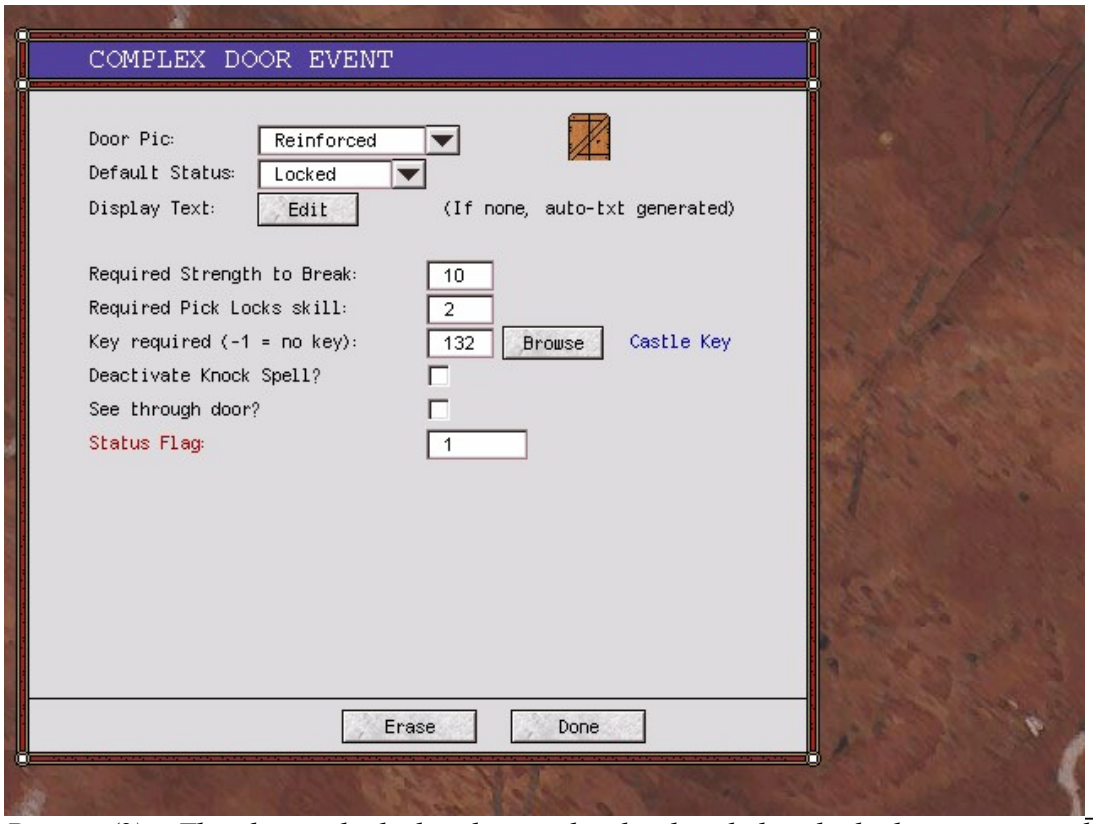
! Save your current level building progress before you use the search, as the search utility clears the memory and cycles through all the level data, meaning that unsaved progress will be lost.

Then there's the option to "Report Flag Conflicts", which collects all cases where flags are used twice by different events and creates a report in form of a .txt file in the DD2 directory. This tool should be used every once in a while.

All flags start with the value of -1. The 0 flag retains this value, as it is reset at every step the player makes; that's also why it is the default flag in any event settings. Usually, if you want nothing to happen at a certain parameter check, leaving things at flag 0 is the right idea.

End aside

So, back to our door. We haven't used any flags yet, but we definitely want the game to track whether it is open or locked. Let's have it being locked and use flag 1; set the other stats at your discretion.



Picture (2) – This door is locked and pretty hard to break, but the lock is easy to pick; it can also be opened with the Castle Key or a Knock spell. Flag number 1 tracks whether it's open, locked or broken.

A friendly character

An empty castle is boring. So let's place the castle's king next. From the Event list, select the "NPCs" tab, then => [Script] and place it in a central room (in the throne hall, if you made one).

Aside: Scripts

> ! DD2 "Script" events must not be confused with the scripts of many other game makers. Effectively, they are what "events" are in RPGMaker. But as there are already the 80 predefined events, you will rarely have the need to use the complex script creation procedure for anything outside of dialogues and rare happenings that are beyond the standard events' scape. On the downside, you also won't be able to manipulate much of the game's core mechanics. Still, there are a *lot* of in-game things you can do with a Script event – not even the author of this guide has exhausted all possibilities yet. The existing DD2 campaigns offer a lot of examples for creative script uses...

End Aside

Right-click on the [Script] event to edit its parameters. The first one is one you will probably use a lot later on, as it exists for nearly every event: "Event active".

Aside: "Event Active"

> ! This is one of the most important features of DD2 event manipulations. We will ignore it for our king, but with this parameter you can activate and deactivate complete events, meaning that you can make persons, traps, walls, triggers, just about everything appear and disappear by setting a single flag. As flags are completely interchangeable (see "Flags", above"), you can create extremely complex event intersections just with the right flag management.

End Aside

Aside: "0 > 0"

> ! When you check a newly placed event, you will usually see the condition check "If Flag 0 > 0". This is the standard setting for all flag checking parts of a newly placed event. It always returns "yes"; you can leave it alone unless you want to specifically manipulate the conditions.

End Aside

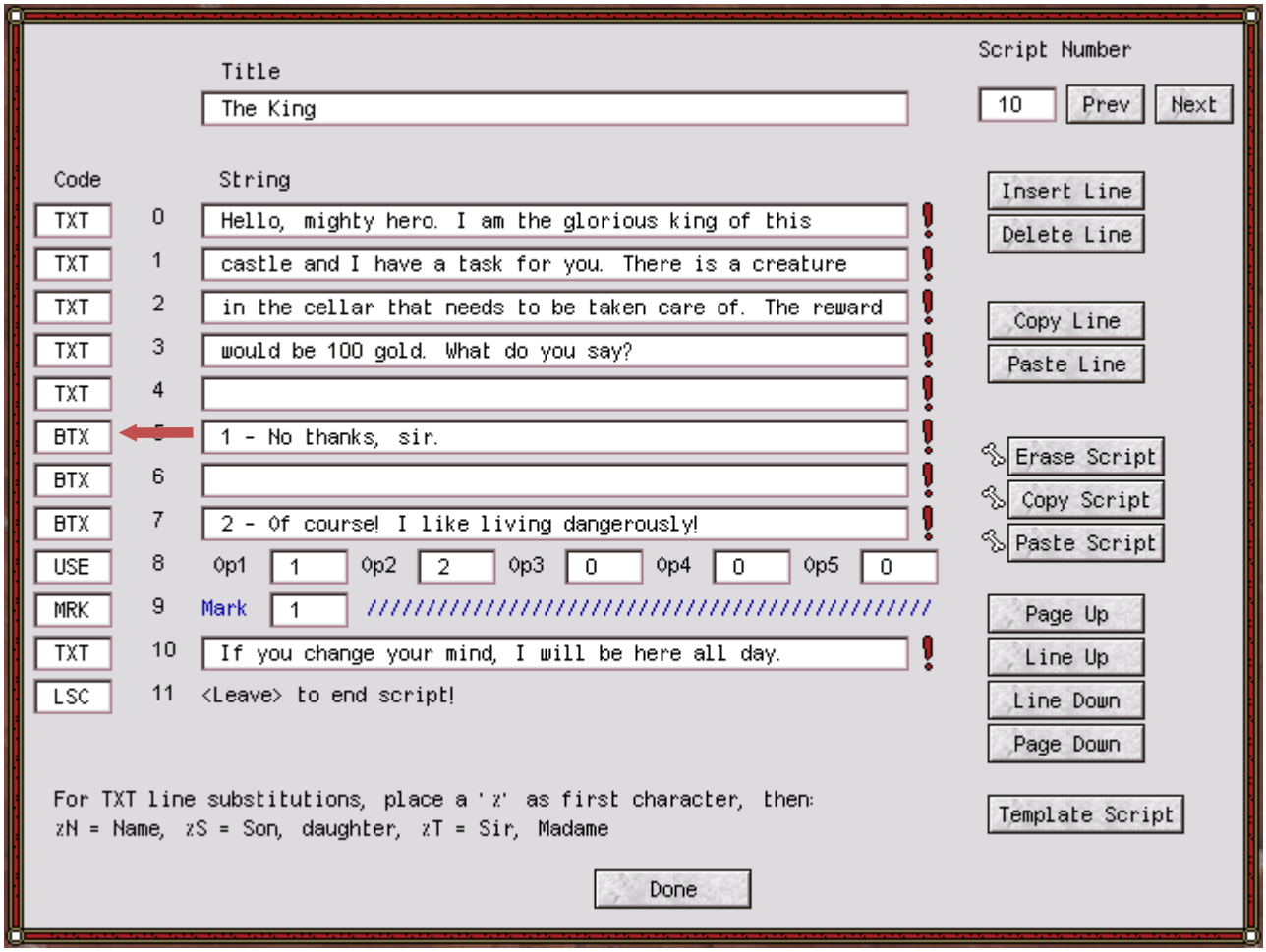
The script number, in our case, can be anything from 0 to 29 (meaning that the maximum number of scripts on one map is 30). "Browse" takes you directly to the script writing window. Before, though, you should also select a picture that represents the script event well. You can choose either an NPC picture or a tile picture, but unless you want our king to look like a wall or a plant, NPC pictures are probably the better option. Choose a suitable one, then select "Browse" to start script editing. Alternatively, the button "Define NPC scripts" also works.

Event: A conversation

Conversations are usually what drives a scenario, giving the player's heroic quest a purpose. That's exactly what we want to do now. Click on "Create Script" to make the editing view come up.

On the left hand side, you will see small fields next to each line. Click on one to bring up the command list, which should be sufficient for conversations and advanced game world manipulations. Most of them are self-explanatory. Right now, we want to create a dialogue for the king, so select [TXT]. What do we want the king to say? Well, as we found above, it's about giving the character's journey a purpose, so let's start with the king giving the player a quest to get rid of a monster in the castle cellar. This means that at first, we will want some greeting lines, then the question whether the hero would take this task upon him. This consists of two parts – telling the

player what options exist, then a question command line. The options will be listed just like normal text. However, to set them apart from what the king is telling, I recommend using blue text, which is meant to be the standard. This would, then, look somewhat like this:



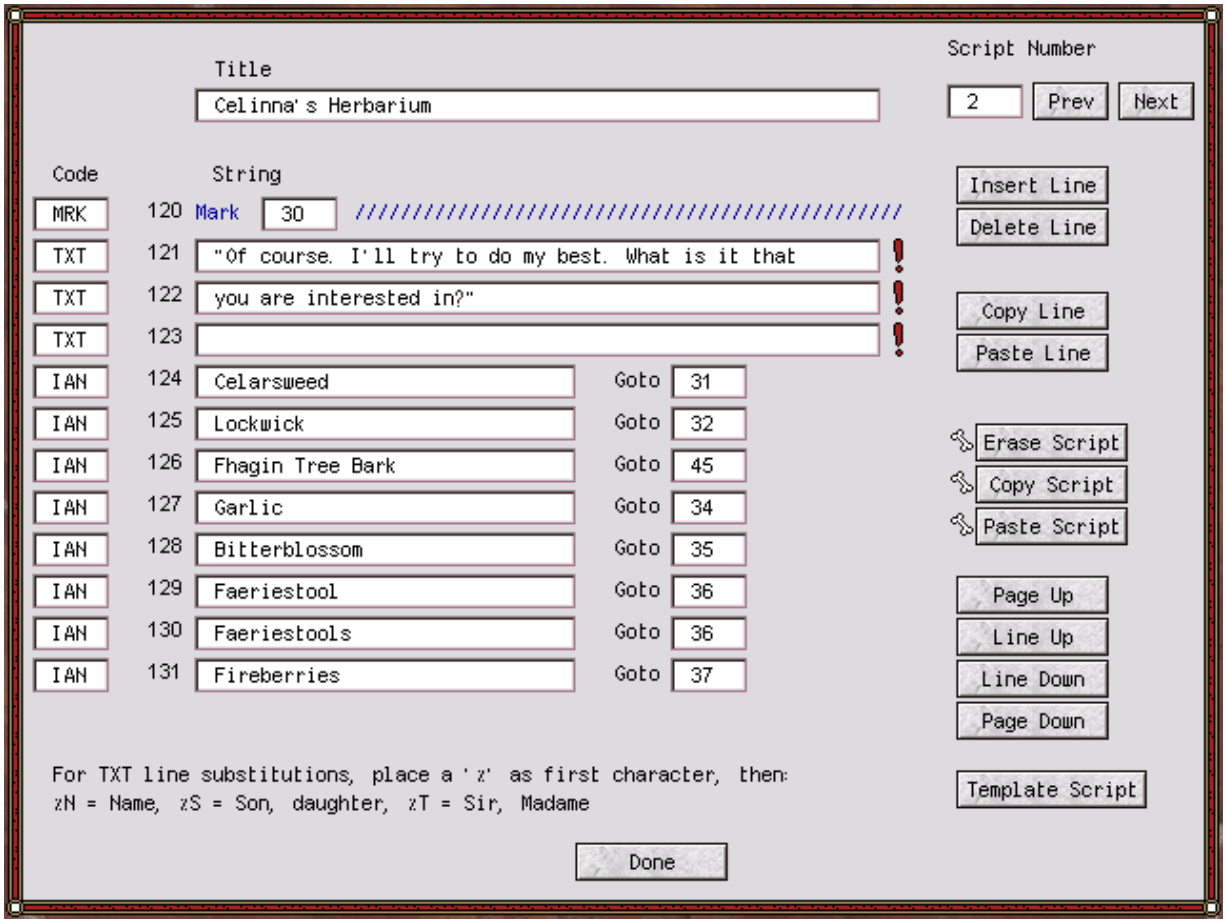
Picture (3) – [TXT] and [BTX] make lines appear in different colors. Below: The dialogue options.

As you can see, break lines in between make things that much more readable. About 20 lines of text fit on one dialogue screen in the game.

Now to allow the player to choose between those options, we need the [USE] command. This defines to which dialogue branch the conversation will be taken when the player uses options 1 to 5. This also means that in general, it's not possible to give the player more than 5 options to choose from, unless you let one of these options branch into more options (and tell the player that, say, option 5 gives "more" options). Note, however, that you are by no means required to use this choice system; the script command [IAN] = *If Answer* opens up an input line where the player can type any kind of text, so you can have conversations take place like in the earlier Ultima games.

To mark those parts of the script where you would like the conversation continue, use the [MRK] command. Do not confuse marks with script numbers! Mark numbers can go from 1 to 99 (mark 0 equals undefined). While the program doesn't care in which order the marks are set, meaning you can have mark 17 following mark 63, you should probably stick to the standard order for purposes of your own overview.

! Some of the most common errors when editing a DD2 scenario are script errors; undefined [USE] options, missing [MRK] commands, twice-used [MRK] numbers and so on. Do yourself a favor and try to keep it organized. I speak from experience…



Picture (4) – Using the [IAN] command to direct a conversation

Now what happens if the player actually chooses option 2? We want the king to give him the quest, we want the quest to appear in the quest log and we want to give the player the key to the cellar. This we accomplish with the following commands:

[TXT] – of course.
[GIT] – gives the player an item: Click on the parameter space to bring up the view of the item database; choose the "keys" tab and select key number 132, the Castle Key, for convenience.
[SQU] – Set Quest

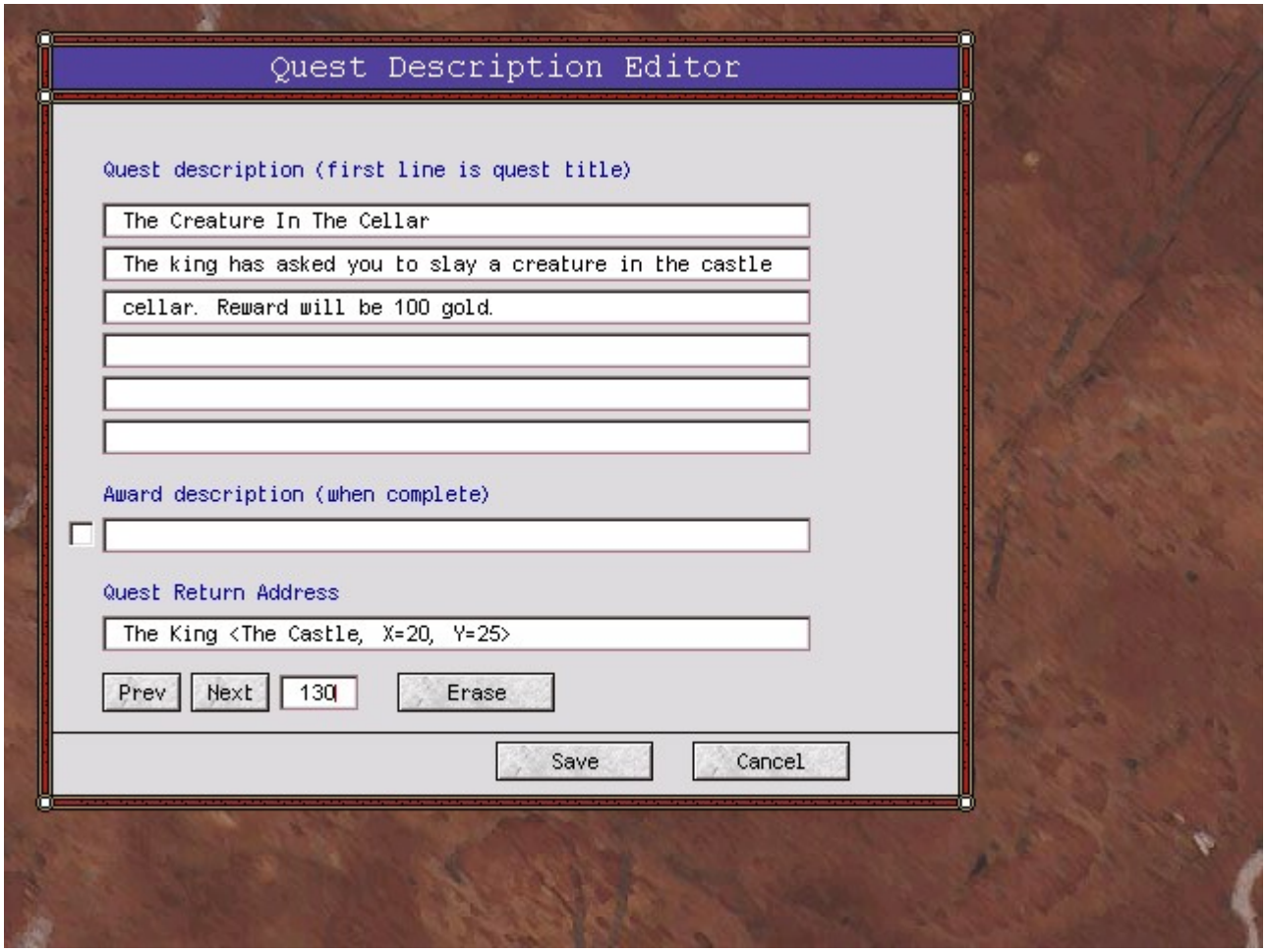
Aside: Quests

Quests are the primary way of giving the player a direction to follow. Conveniently, DD2 comes with a fully functional quest log which can take up to 200 quests. The quest database can be accessed via the tabs "DB1", "Quests" on the right side of the main editing screen. In general, a quest should have a title, a short description of what to do (and possibly where, unless finding that out is part of the quest) and a direction of who to turn to for a reward. The option of giving the player a message for the "award" section of the quest log should be restricted to the most important quests, as the award page can only track 20 quests.

! Quests can have one of four statuses to be tracked with the [IQU] script command:

- Inactive – player has not come in contact with this specific quest
- Active – the player is right now on this quest
- Cancelled – the player once had this quest, but for whatever reason has dropped off
- Resolved – the player has successfully finished the quest

End Aside



Picture (5) – An entry in the quest database

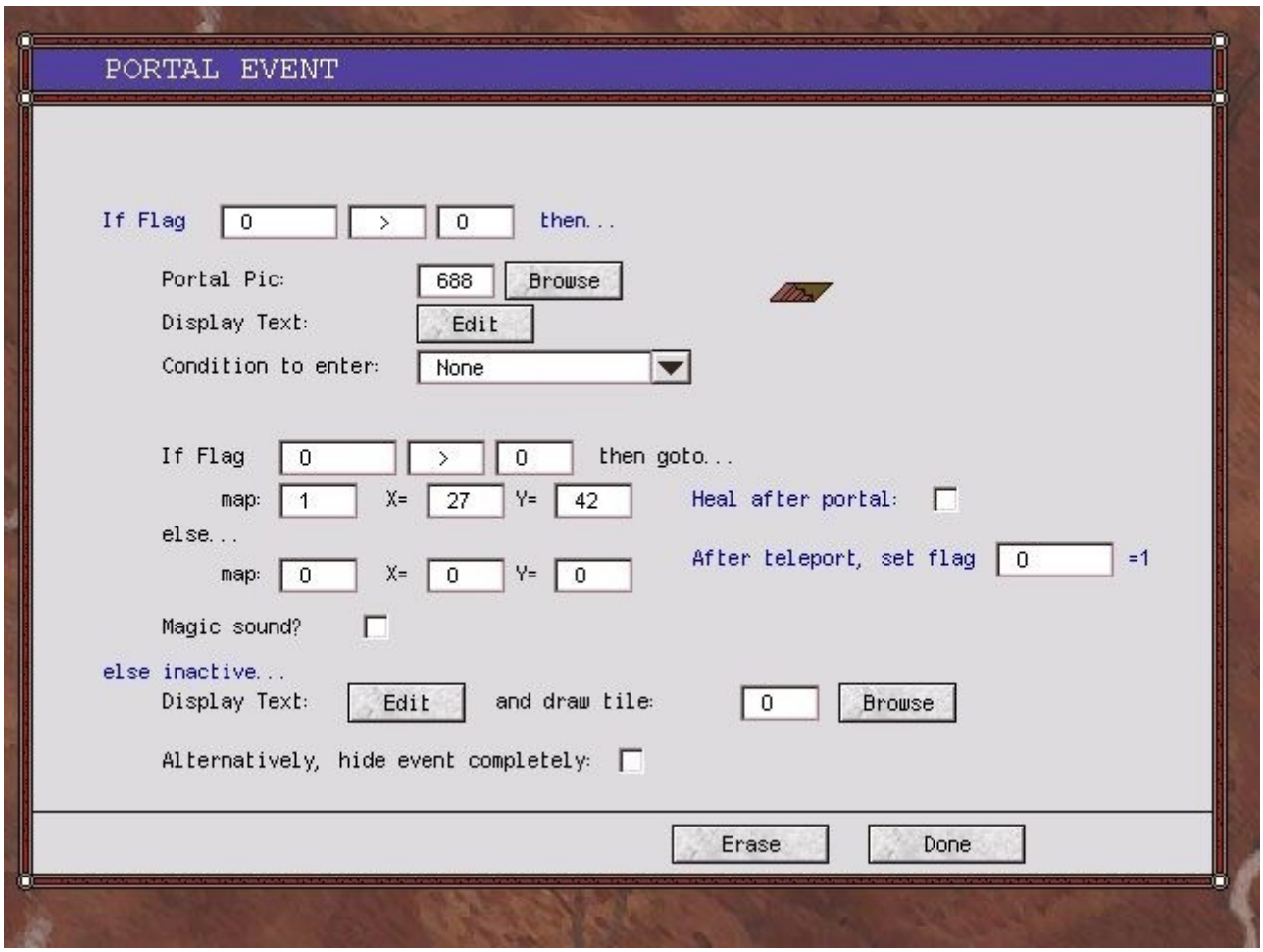
Produce an entry in the quest database like the one in picture (5), although quest number 1 in this case might be more appropriate, then give the king's dialogue – at the point where the player has accepted the quest, so for the example in picture (4) it would be at mark 2 – the [SQU] command to set quest 1 as active.

(The dialogue is not complete yet – we will revisit the king after we are done with the monster.)

Now for the monster. Remember that you should leave some space of the map unused for now? That's where we draw our monster cellar. Proceed as with the rest of the castle, but don't connect the two parts; it's supposed to be a cellar, after all, so it should be reached by stairs.

This we accomplish by using the [PORTAL] event, found in the events catalogue under the "Access" tab. We will need to place two portals, the stairs down and the stairs back up.

The [PORTAL] event can look a little intimidating at first. However, we don't need the majority of options right now, and once you go creating something more advanced, you'll be glad they are there.



Picture (6) – A simple portal event

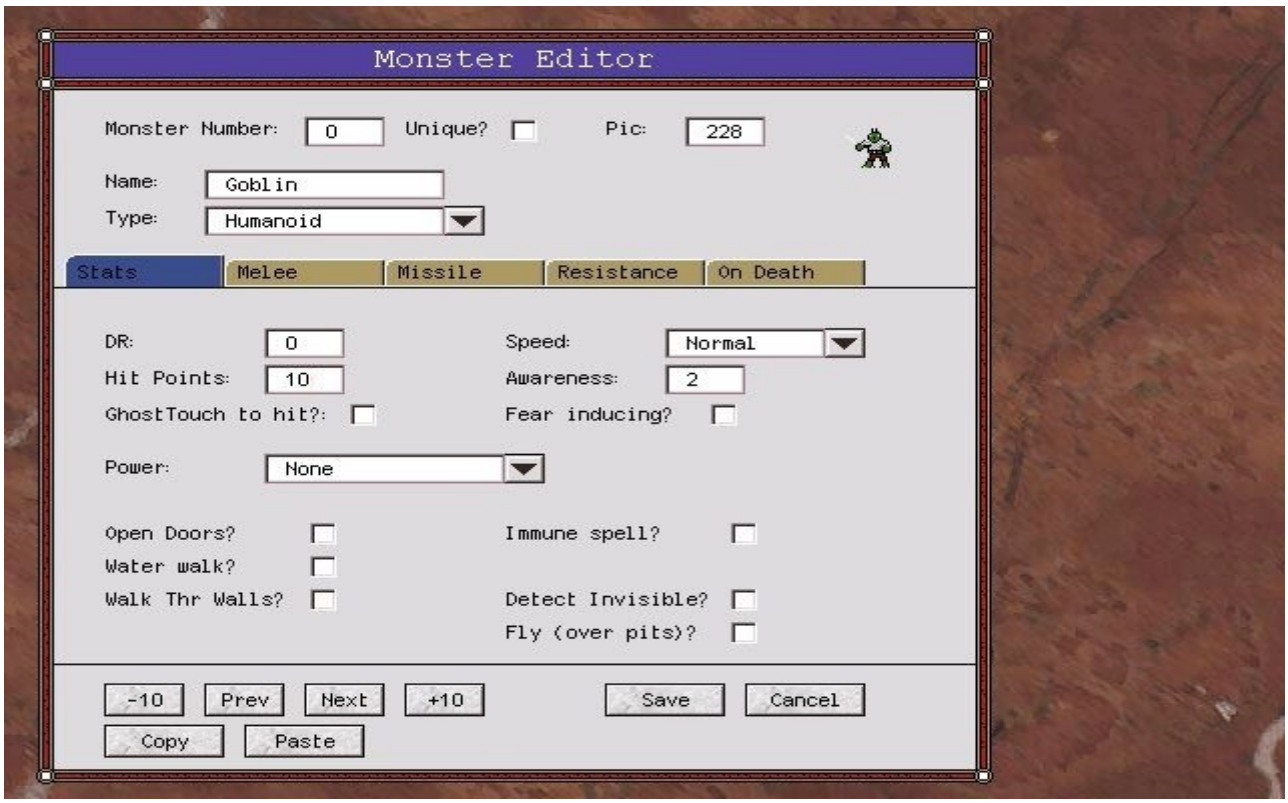
Take a look at the example picture. As you see, there are only very few things which we need to apply special settings to:

- The most important one is the destination. For a standard portal, this is the first line that says “map:”. Input the coordinates of your cellar entry here.
- Next, you will want the portal to actually show up on your map, so click “Browse” at the “Portal Pic” line and choose something suitable. The standard stairs can be found in the “Bits 1” category.
- Finally, there's the “Display Text” line. Click the “Edit” button and put in something appropriate. This can be something simple like “Go down the stairs?” or something more elaborate: “An old, seldom used staircase leads down into the castle cellar here. Small spiders have woven their webs at nearly every corner. Are you sure you want to descend the stairs?”

After you have placed the stairs down, don't forget to place the stairs back up! They should, of course, be at or right next to the place where the downward stairs lead to, otherwise things get a little confusing...

Now for the monster. As this will be the first monster in our scenario, we should probably refrain from placing a black dragon here. A goblin or a beetle a new hero should be able to defeat. Hopefully.

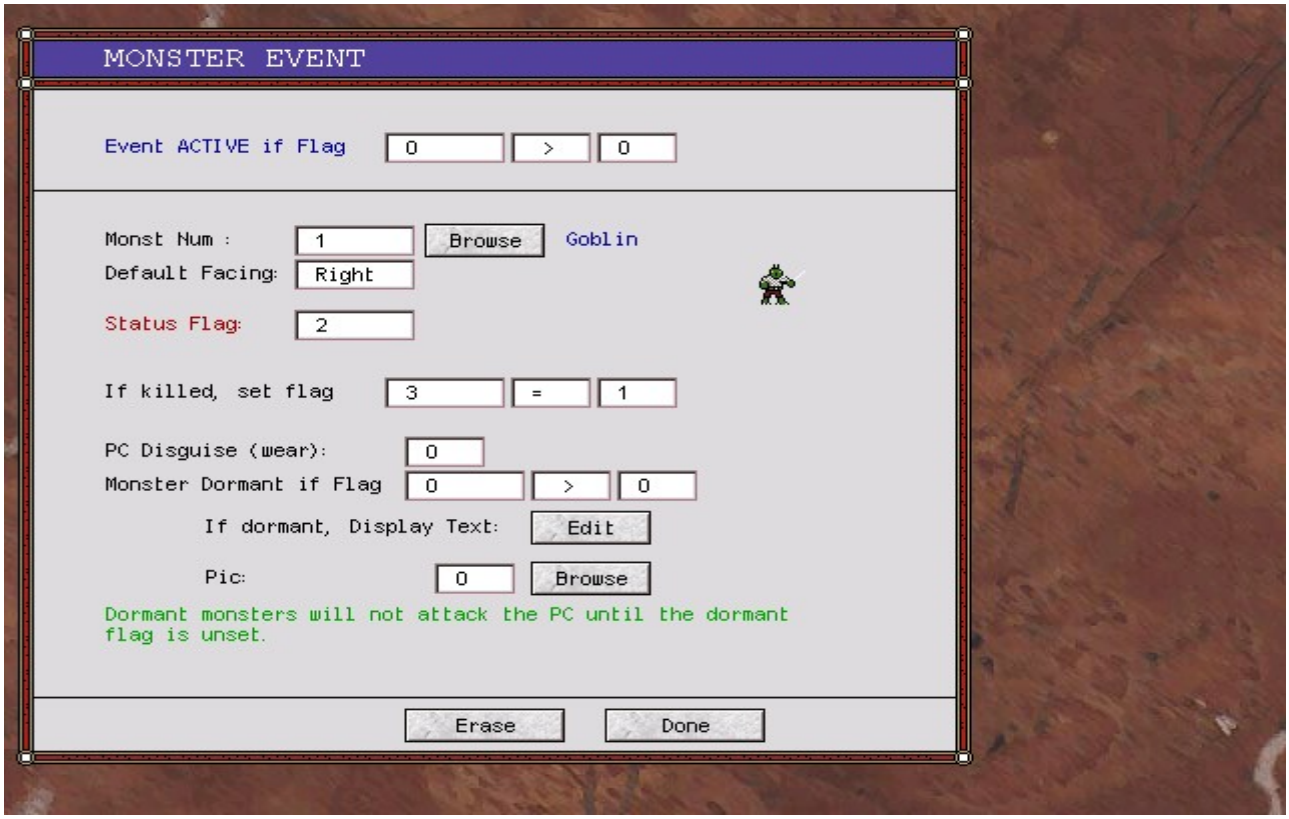
First of all, we need to create a monster so we can actually place something. Select the “DB1” tab on the right, then “Monsters”. You will be faced with a screen like the following (without the data):



Picture (7) – A suitable monster for a beginning hero

If you have played the Dark Disciples games, you no doubt have faced countless monsters and should be roughly familiar with the array of abilities and features the monsters might have. For our standard Goblin, just let him be at 10 HP, no need to change the other data.

To place the green guy in the cellar, use the NPC => [MONSTER] event. Right-click to see this:



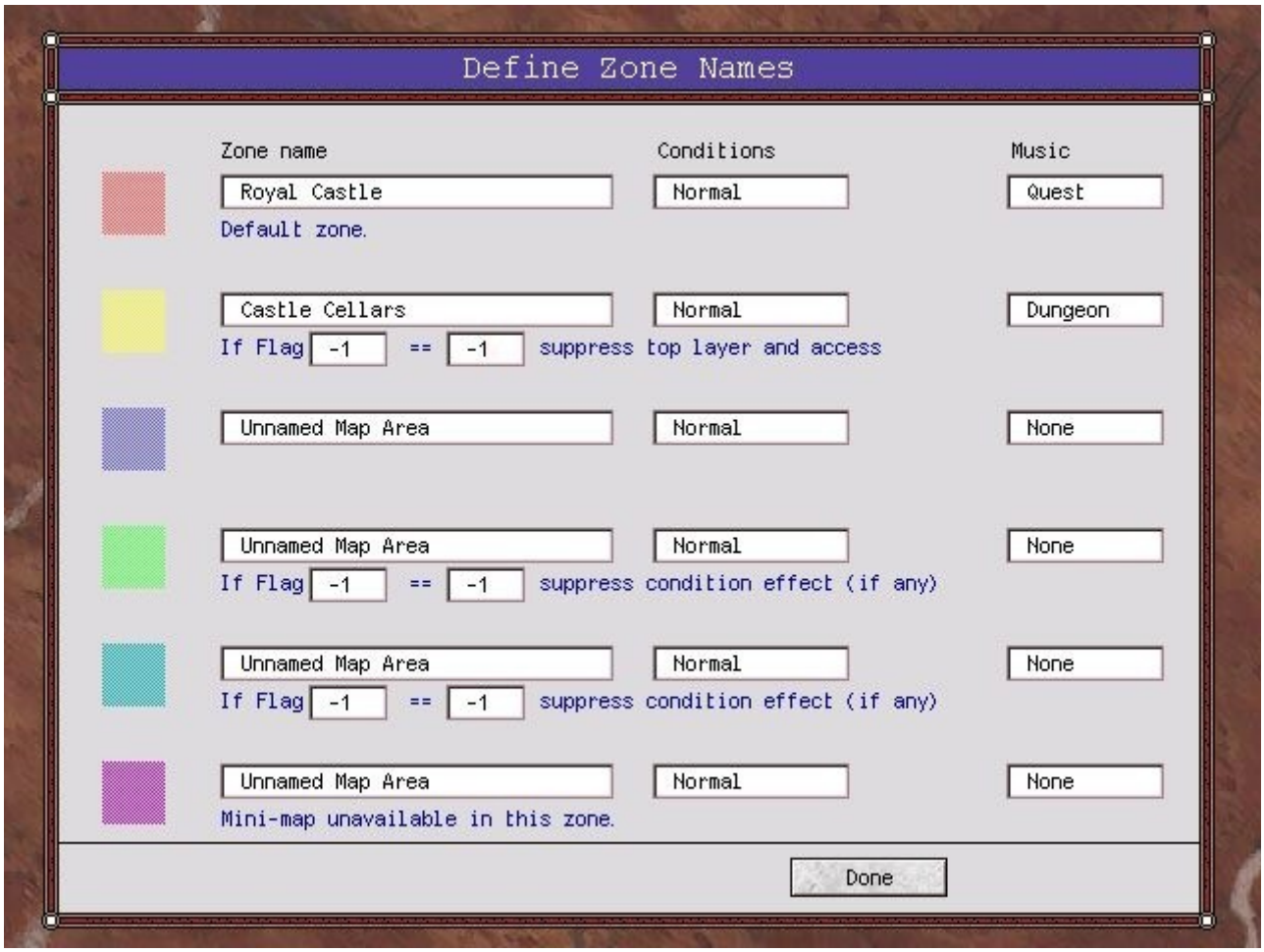
Picture (8) – A [MONSTER] event

Contrary to the [Portal] event, you need to pay a little more attention to the settings here. For the status flag and its importance, refer to the *Aside: Flags* explanation above. Be sure to use a unique flag here. Even more interesting is the “If killed set flag” setting. Use another unused flag here and make the engine set it to 1 once the brave hero has killed the Goblin.

For now, our cellar is complete. If you wish to, add some additional furniture from the range of available tiles – maybe some crates and barrels, maybe some bones and blood (who knows if this area might not have been a torture chamber for an earlier king?). Don't forget to set the access settings accordingly!

One thing is left. If we let the map be as it is, the player will know the cellar to be just another section of the castle map; in particular, they will both show up on the automap at the same time. We don't want this. So we will use the Zone settings, which also have some other uses as well.

To define the different zones of a map, select “Define Zones” from the “Def” tab. This brings up the following screen:



Picture (9) – The zone definition menu

As you can see, several important characteristics can be defined here. It is really recommended to give names to your zones as well as select some accompanying music. The “Conditions” settings and especially the “If Flag” options are much advanced and not relevant here. However, there's one important feature, namely the violet zone's ability to turn off the automap! So, unless you want to irritate the player on purpose, don't use the violet zone on your maps.

After having defined the zones, we just need to tell the game which are of the map belongs to which zone. This is done much like placing tiles; select one of the eight placement icons in the upper right and choose “Z” to be presented with the six zones. If you have defined the zones like in picture 9, select the yellow zone and fill the cellar area with it.

Finally, we want the king to give the player a reward for slaying the vicious Goblin! Bring up the script for the king. When we last left it, it presented the player with the option of accepting the quest. However, even if the player accepted that quest, the king would ask him to slay the monster again and again whenever the player would approach him. So we need the king to check whether the player is already on the quest and if so, whether he has managed to fulfill it.

This check would have to be performed at the very beginning of the dialogue, before the first greeting. Hmmm, there's no space left. Well, just use “Insert Line” (at the right side of the screen) and insert a few lines; you can do this a few more times if it turns out you will need more space.

Select the “IQU” (If Quest) command from the script command list. If you have followed the tutorial default settings, the quest has the number 1. Set the parameters so that the king checks for quest 1 to be “Active”; if so, the player should be sent to an unused mark below the rest of the dialogue.

If the quest is active, there are two options – the player either has slain the Goblin, or he's still idling around. How do we check that? Well, remember the flag settings of the Goblin's [Monster] event? We wanted it to set an unused flag (say, 3) to 1. So we can have the king check if flag 3 is set to one – using the “IFL” (If Flag) command – and if it is indeed so, send the player to yet another unused mark (say, 4). ! IF NOT, however, we still must include some text and commands so the player isn't left in the middle of nowhere. So, below the IFL command, insert some text lines akin to “You haven't killed the monster in the cellar yet. I will not give you a reward until you have done so.”, then use “LSC” (Leave Script”) to kick the player out.

At the mark to which we go if the Goblin is killed, we want the following commands:

- “SQU” (Set Quest) 1 = Complete
- “GGP” (Give Gold Pieces) = 100
- “GXP” (Give eXperience Points) = 20
- some text in which the king congratulates the hero
- and finally, “LSC” again.

As you might guess, this will give the player the appropriate reward. The SQU command is necessary because we don't want the player to receive the same reward a dozen times for solving a quest once: Return to the beginning of the script. Insert another line with the command “IQU”, checking whether quest 1 is complete. If so, send the player to another mark (say, mark 5) where he receives a warm welcome from the king, but nothing else, and then is kicked out of the script right away (“LSC”).

After all of this is done, start the game and try out what we have done so far. You should be able to get to the cellar and back; see the castle and the cellars on the automap, but not both of them ar once; get the quest, kill the Goblin and get the reward – but only once.

This should get you started. But there are many, many more options hidden in the editor. Your best bet is probably to play the campaigns available and find out how things were done there. If there are questions about this guide, contact me at *classicgamer@gmx.net.* Good luck and happy crafting!