Devil Whiskey – Map Tutorial

Section 1 – Introduction, basics of map files

The maps in Devil Whiskey are all laid out as a grid of cells. The movement is based on a basic X/Y grid, with each grid cell having a possibility of a wall on any of its four sides. A cell can also include a "garnish", such as the cross in the Courtyard, a door, which is really just a wall texture you can walk through, or an empty space (no walls defined).

All the files are written in plain ASCII text, to be easily modifiable with a text editor – the developers primarily used Vim and XEmacs for the purpose of file editing. There are countless text editors available for most every platform, pick your favorite, or try a few if you don't have a favorite yet. Any line can be commented out by starting the line with a # symbol, this allows you to leave notes for yourself and other without messing up the game's interpretation of the file.

There are 3 basic files for creating a play map, each of which is required. They build upon each other, starting with a wall map file, which is used to build a cell file, which in turn is used to create the main map file. All map files are stored in <install-dir>\maps for an example, feel free to look at any cohesive group, such as cave_walls.cfg, cave_cells.cfg, and cave.cfg . You will notice there are a couple more cavexxxx.cfg files, more on those later.

Section 2 – Individual Map Files in detail

Walls

The walls file defines the basic wall textures you will use to create your maps, each different texture needs a line, and the format is as follows (< and > for clarity only):
<WallName> <texture pic file> <collision-integer> <special type> <height>

- wallname is simply the name you want to use to refer to this wall texture
- Texture pic file is the relative path and name of the texture file, created using TexMake (see another tutorial for how to use texmake).
- Collision Integer is set to either 0 or 1, following the C tradition of 0 representing false and non-zero representing true. If this value is set to 1, it means the player cannot pass through this texture.
- Special type: 0 = standard, 1= secret door, -1 floor tile (if none given, defaults to 0).
- Height – values: 1 – 4 if none given, defaults to 1.

So, a brief example - in cave_walls.cfg note this line:
Wall1 pics/walls/cave1.dat 1

This defines a wall texture, named Wall1, using the texture image pics/walls/cave1.dat (ie – it comes from <install-dir>/pics/walls/cave1.dat) and defines collision to be true, so you cannot walk through it. Values for Special type and height are not given, so they are their

default values of standard and 1 respectively. In other words, this is a normal wall, nothing special about it.

We will similarly define a door:
Door1 pics/walls/cave_door.dat 0

This creates a wall texture named Door1 using the image in pics/walls/cave_door.dat with no collision, standard type, standard height. So the player can walk through this wall texture, making it act as a doorway we can walk through.

You can use the same file for multiple definitions (makes it easy to add new textures without redoing your map file) and you can define a wall texture that is really only the floor (such as the beaten path from the adventurer's hall to the front gates of Rennibister.

A note about height – if the map does not define a ceiling (i.e., it's an outdoor map, see below), the height is as you may guess – how high off the ground the wall goes. If, however, the map *does* define a ceiling, the wall height will be interpreted as distance from the ceiling, not from the floor. This means if you have a portion of the map that is at height 4 (i.e., the ceiling is 4 units high), and you want to have a partial wall (that doesn't go all the way to the ground) that connects to a height 2 section of the map, you need a height 2 wall, if you want a partial wall that connects to a height 1 section, you need a height 3 wall, and so forth.

Cells

Using the walls you define in your walls.cfg file, you can then create a cell file, which defines each type of individual cell you will need in your main cfg file. Remember that each cell can have a wall at any of the cardinal directions, and/or a single garnish in the center of the cell. Throughout Devil Whiskey, our standard practice of representing the Cardinal directions (North, South, East, West) has been to start at North, and work our way counter-clockwise through the rest of the directions. So North comes first, then West, then South, then East. Here is an example of a line from the cave_cells.cfg file:
<Cell name> <N Wall> <W Wall> <S Wall> <E Wall> <type> <floor> <garnish>

- Cell Name – The name you will refer to the cell by in your main map cfg file
- N Wall – the wall tag defined in the maps' wall.cfg file that you will use on the north side of this type of cell
- W Wall – same as above, but on west side
- S Wall – same as above, but on south side
- E Wall – same as above, but on east side.
- type : 0, 1, or 2. 0 means normal cell, 1 means we're going to define a cell with a special floor texture, and 2 means the cell will contain a garnish
- floor – if type =1 or 2, give a floor texture, either texture name to use, or NULL if default
- garnish – if type = 1 or 2, give a garnish image to use, or NULL if no garnish.

NOTE: There is one special cell that MUST be defined in every cell file, named 'DEFAULT'. This is the cell definition that every cell in the map will have, unless overridden by a setting in the main config file (below).

Main Config file

Finally, the map needs this last file, to put it all together, and define a few global items needed. This is the most complex of the map files. The format follows:

<map name> <map width> <map height> <default floor> <default ceiling> <title bar image>
<walls file> <cells file> <events file> <monster file>
SPECLITE fr fg fb fd lr lg lb
audio/config 1
<audio file name> <audio track to use>
SPECCELL stairx stairy numOfTraps
trapX trapY trapType trapNum <repeat, one per line = numOfTraps>
RANDITEMS <numOfRandItems>
itemNameWithUnderscores relativeFreq <repeat, one per line = numOfRandItems>
mapX mapY cellName

Lets review each of these items in more detail:

First Line:

- Map name – the name to represent this map (unique from other maps)
- map width – the width in cells of the map
- map height – the height in cells of the map
- default floor – path to a default floor texture – will be used in all cells unless a floor texture is specified for a cell in the walls.cfg file
- default ceiling – the default ceiling texture for all cells
- title bar image – this is an image that contains text or whatever you'd like to go above the Upper Left window (navigation window)

Second Line:

- Walls file – the name of the walls.cfg file for this map (ie – maps/mywalls.cfg)
- Cells file – the name of the cells.cfg file for this map (ie – maps/mycells.cfg)
- events file – the name of the events.cfg file for this map – see below
- monster file – the name of the monster.cfg file for this map – see below

SPECLITE line:

The keyword SPECLITE followed by the lighting values, 7 real numbers ranging from 0.0 to 1.0 (inclusive) - with values for the following fog/colors:

- fr - fog color – red
- fg – fog color – green
- fb – fog color – blue
- fd – fog density (0.0 is no fog)
- lr - ambient light – red
- lg – ambient light – green
- lb – ambient light – blue

audio/config 1: This line states that the audio config files for this map are contained in the audio/config directory, and that there is only one audio config. It can be '0' if

no ambient audio is used, or more than 1 if you use multiple time-based audio config files (see 'rennibister.cfg' for an example).

audio file name/ start time – the name of the audio config file, and the time to start this config. If there is only one audio config, then the value will be '0' (for 'start at midnight'). If there is more than one, then this is the hour (24-hour day) that the particular config will become 'active'. There will be exactly the same number of config-file lines as specified in the config count (on the line above).

SPECCELL line:

- The Keyword SPECCELL
- stairX – the x location (cell wise) of the ENTRY stairs
- stairY – the y location (cell wise) of the ENTRY stairs
- numOfTraps – the total number of traps in this map
- special note – the stairX and stairY line are needed for the RELO spell

Trap Lines:

- trapX – the x location of the trap
- trapY – the y location of the trap
- trapType – the type of trap, 1 for normal (script based), 2 for spinners, 4 for anti-magic, and 8 for darkness traps
- trapNum – needed if trap is of type 1, calls the corresponding numbered trap script (trap000x) so setting this to 2 would call script: trap0002.py

RANDITEMS line:

- The keyword RANDITEMS
- numOfRandItems: an integer value, that specifies the total number of unique items you are about to list

Item lines:

- The name of the item, as given in bl2d_items.txt, with spaces replaced by underscores '_' so a Dwarven War Axe is listed as Dwarven_War_Axe. Spelling and capitalization matter
- relativeFreq - The relative frequency of the item drop. A larger number relative to the numbers given for other items listed means this item drops more often.

The actual map definition:

- mapX – the x coordinate of the cell
- mapY – the y coordinate of the cell
- cellName – the name of the cell to place at this x/y location

Tips on map design:

- Start your map on graph paper, which makes it easy to construct the text file and determine the necessary cells.
- Review some of the included maps to help get you started, there are lots of easy to read examples in the maps directory of your Devil Whiskey installation, check them out for examples – they're all plain text
- Give yourself some buffer room around the edges of the map (ie – blank cells), it helps to make defining outer walls easier, and gives you room to add blanks to the bottom of the cell

- Blank cells need no definition, nor do they need lines in the map.cfg. If no lines are given, the cell will contain no walls, and have the default ceil and floor textures.
- Make sure your maps all have enclosures going around them, ie – no holes to let the player wander off the map.
- Don't overlap the same edge between two cells. If you have two cells, next to each other with the same y value (ie – one on left, one on right) – don't give both cells a wall on the common side. The more walls you have the slower the map runs/loads, and putting two walls together on the same edge causes graphical artifacts while playing.
- Use the DWMapEditor for verification – it's still in development for making new maps, but is an excellent tool to quickly review the maps you've made. While it does produce fairly reliable maps, it is somewhat untested, and there are a few known issues, such as it fails to write out a SPECCELL line, has no utility for adding traps or random items, and the SPECLITE cell it writes out is not standard (although functionally correct). The map editor was NOT used in the production of the maps for Devil Whiskey full 1.0, but was used for verification purposes.

Section 3 – the other map files

Mon

You'll notice that each map has a corresponding nameMon.cfg file for it, this is the monster file referenced in the main map config file – it contains information about which monsters will be in this map, their frequency of appearance, number of monsters available to the map, and the timeframe in which they will appear. The format is as follows:

<num of monsters> <frequency of random encounters>
<name> <grp base> <grp var> <initial range> <base occurrence> <extra occurrence> <entry hr> <leave hr>

First Line:

- numOfMonsters – the number of different monster definitions in this file
- frequency of random encounters – integer between 0 and 100, the larger the number, the more often the party gets randomly attacked

Definition Lines:

- name – the name of the monster from monsters.txt, with spaces replaced by underscores
- grp base – the base monster group size, ie – 2 here will mean the monster group always has at least 2 of this type of monster
- grp var – the variance of the monster group size – this defines the range of random numbers that can be added to the group, ie – defining 2 in grp base and 3 here means the monster group size could range from 2 to 5 monsters.
- initial range – the starting distance the monster begins combat at – ie 30 means the monster enters combat 30' away from the party
- base occurrence – the base chance of occurrence, it never changes, representing the fact that your party will never kill all of any monster type.
- extra occurrence – this is added to the base occurrence upon entering the map. When a monster group is killed, this number degrades, meaning that the monsters of that type will become less plentiful on this map.

- entry hr – the first hour at which these monsters become active on the map in 24 hour time ie – 0 is midnight
- leave hr – the last active hour these monsters are active on the map in 24 hour time – ie – 23 is 11:00 pm.

to be noted in regards to time – if the definition is 0 for entry hr and 23 for leave hr, the monster is available on this map at all times. This is useful in the cities and the wilds for making the more powerful enemies come out at night.

Event

The last map related file is arguably the most important after the map definition itself, the event file specifies what happens and where on this map. Entry/exit staircases and passageways, scripting plot encounters, chests, and anything else that might happen to the party happens through event scripts. Writing the event scripts is another tutorial on its' own, but the mapEvent.cfg file determines when a script is run. It has a simplistic format as such:

<triggerX> <triggerY> <event Processor>
<reboundX> <reboundY> <reboundFaceAngle> <scriptName>

These 2 lines must exist for each event that you wish to include in the map. The format specifies the following:

Line 1:

- triggerX – the x location that triggers this event
- triggerY – the y location that triggers this event
- eventProcessor – this is the number of the event to trigger, and will most commonly be 2000 – which is the python interpreter that handles scripted combats, stairs or other map changes, messages to player, plot advancement, etc. Other event numbers are listed in the next section

Line 2:

- reboundX – the x location the character is rebounded to once the script terminates with a false (0) result
- reboundY – the y location the character is rebounded to once the script exits with a false (0) result
- reboundFaceAngle – the angle the character is facing on rebound – 0 = north, 90 = west, 180 = south, 270 = east
- scriptName – the name of the python file to invoke without the .py extension, ie – if we wanted to call myPlotScript.py from this location, we would use myPlotScript for the script name.

Appendix: Event Numbers

1000 – Adventurer's hall
1001 – Elium's Energies
1002 – Temple of the Redeemer
1003 – Collegium of Magika
1005 – Order of the Lyre
1006 – Melkoran's Tavern
1007 – The Green Bard
1008 – Hughter's Owl
1009 – The Chiccasaw
1010 – The Marauder's Hole
1011 – Starfire Swords
1012 – Imperial Weapons
1013 – Arms of Valor
1014 – General Store
1017 – Landru's Craft Supply
1018 – Guard's Barracks
    ** NOTE: most barracks are actually scripted **
1022 – Casino
1023 – Arcanium
1050 – Generic Stranger's House
2000 – Python Interpreter for python scripts