- Editor Manual

Contents:

1. Introduction

- 1.1. About

  1.2. Missing functionality

2. Starting map creation

- 2.1. Creating a new map

  2.2. Loading/saving

3. Editing the map

- 3.1. Placing terrain and objects

  3.2. Modifying objects

  - 3.2.1. Cities

    3.2.2. Ruins and temples

    3.3.3. Stacks

4. Some menu items

5. Managing players

6. Managing events

- 6.1. Introduction to the event system

  6.2. Editing the event system

  6.3. List of events

  6.4. List of reactions

  6.5. List of conditions

  6.6. Some examples

7. Editing the scenario manually

1. Introduction

1.1. About

This document describes the map editor. The editor has been added in version 0.3.5 and is at the time of this writing rather incomplete and crude. I hope it nevertheless makes some fun creating maps with it. If you encounter bugs or find some feature would be very important, just drop a mail. It may also happen that the document is not up to date, though it should be updated before releases.

1.2. Missing functionality

I have been coding the editor for three months now and have got a bit tired of it. So I hope you don't mind if some functions are missing. They will be implemented in the future versions. Until then, see section 7 for some hints how you can do things that the editor does not support.

Missing general things are:

- temple and ruin reports are missing
- keyboard support is missing
- the layout and usability is sometimes horrible
- the event system is complicated (this is rooted in the event system and propably not subject to change)

Missing functionalities are:

- it is not possible to set the actual hitpoins of an army, just the maximum (all armies start healthy into the game); the same goes for movement points
- items can not be placed on the map
- heroes cannot be assigned initial quests
- it is not possible to move objects, just to delete them and create a new one
- version support, i.e. if the savegame version changes, you need to patch the scenario file manually

2. Starting map creation

2.1 Creating a new map

In the file menu, select "New Map". In the first dialog, the terrain set (currently only default), the size and the fill style can be selected. The fill style selects which terrain the map should be initially filled with. A special fill style is "Random Map". It creates a random map with cities, temples, ruins etc. The terrain distribution and the number of buildings can be selected in a following dialog. However, the buildings have only default names.

As an alternative, you can load the file "random.map" in the FreeLords savegame directory (usually $HOME/.freelords). This file already has the players set up, names given to buildings and standard events set up. It is the last played random game.

2.2. Loading/Saving

The file menu also contains items for loading and saving. Loading a map pops up a dialog to select a map file. Note that these files must have an ending of ".map", else they are not displayed.

Selecting the item "Save" in the file menu saves the map under the last filename (i.e. the file the map was loaded from or the last saved file). If the filename does not exist, the functionality is the same as "Save As", which also pops up a dialog for selecting the file name and directory for saving. Note that a prefix of ".map" is automatically prepended unless you write it explicitely.

3. Editing the map

3.1. Placing terrain and objects

When you have loaded or created a new map, the first thing you propably want to do is modify the terrain and place some buildings/stacks. For this, you have several button at the right-hand side of the editor.

With the terrain icons, you select which terrain you want to place. Beside them, you have several other buttons for

- changing the terrain in a 1x1 area
- changing the terrain in a 3x3 area
- removing objects (the red cross); note that you are not queried if you really want to remove something, and be aware that stacks are always removed first!
- placing stacks, cities, ruins, temples

For most objects, some error handling is done, e.g. you won't be able to place cities in water, the terrain is automatically altered to grass. Note that this does not work for stacks, so ensure that stacks that are placed in water only consist of water-moving units!

3.2. Modifying objects

In general, you can always modify an object by right-clicking on it. When multiple objects are stacked (e.g. a stack on a ruin), you are asked what you want to edit.

3.2.1. Cities

Right-Clicking on a city opens the city dialog. Most of it should be fairly self-explaining, so I will just describe some concepts here.

The difference between basic and advanced productions is already detailed in the manual (basic = default armyset; the same for all players, advanced = player-specific armyset, removed when city is conquered).

What is worth a remark is the dropdown to select between "Untouched" and "Burnt down". When a player occupies a city, he may choose to raze it, which burns the city down (it is just a ruin then). Sometimes it may be interesting to have a city razed initially. To do this, select "Burnt down".

3.2.2. Ruins and temples

In short, what you can change here is mainly the name. In the case of ruins, you can also modify the occupant (which a hero has to fight when entering the ruin) by clicking on the button "Keeper". The following dialog is identical with the stack dialog.

3.2.3. Stacks

Right-clicking on a stack brings up this dialog. Again, I suppose the dialog is mainly self-explaining.

When changing the ownership of a stack, make sure that it is consistent, i.e. don't place a player's stack in a neutral city. After all, the purpose of map creation is _not_ to stress the error routines of the game.

The properties of the stack's armies can be modified as well. The idea behind this is that sometimes finetuning of the strength etc. is needed e.g. to reflect a wounded or veteran army.

4. Some menu items

The other menu items not described elsewhere are:

- a find dialog; Enter the ID of the object you look for (the object may be a city, ruin, temple, stack or unit), press Return and the map should jump to the object's position.
- a stack and a city report; Click on one of the listed objects to jump to their positions

The other things (player and event management) are subject of their own chapters.

5. Managing players

Select the item "Players" from the "Edit" menu to open up the dialog for player management.

Clicking on "Prev" and "Next", you can scroll through all players. Using the two dropdowns, select the armyset of this player and the type. There are four types of players (as in the game): Human, Dummy AI (does nothing at all), Fast AI (assembles stacks and attacks) and Clever AI (tries to protect cities etc.).

The edit fields below should be fairly self-explaining, you can set the color, name and starting gold of the player.

There is always a neutral player which you can (and should) not remove. Furthermore, you can make players persistent (immortal, i.e. they don't die when loosing all cities). In the special case that you select a fast AI, there are two additional options you can choose. First, you can have the player join close stacks to increase their strength. Second, you can make the AI maniac. In this case, it will attack relatively blindly (nice for wandering monsters).

If you add a new player, you will be asked whether you want to append it at the end or before the actual player to give you some control about this.

6. Managing events

6.1. Introduction to the event system

Usually, when creating a scenario, you want some special effect, such as armies that ambush you or different scenario objectives, such as searching a special ruin or taking a certain stack to a certain position. In FreeLords, this can be done using the integrated event system. It is very powerful, but also quite complex and may demand some sort of programming abilities.

Before I go into the details, here some short explanation. The event system consists of three items: Events, Reactions and Conditions.

- Events are triggered by certain, well, events. As an example, the RuinSearch event is triggered whenever a player searches a specific ruin.
- Each event can have one or multiple reactions. When the event is triggered, all reactions are also triggered. Such reactions can be "Kill a specific player" or "Win the game".
- Each event and reaction can also have one or multiple conditions. The event or reaction is then only triggered if the condition is true. E.g. to ensure that a CityConquered event is only triggered if the city is conquered by a specific player, assign a condition to the event that checks if the given player is the active player.

Now some more details:

Events have a unique id (they need to be referenced occasionally), as mentionned previously a list of conditions and reactions, some (type-specific) data and a comment string. The comment string has no meaning for the game, but can explain the event, so other people understand what the event is for. Furthermore, an event starts active or deactivated at the beginning of the game. When the event is triggered (which only happens if the event is active), the events becomes deactivated. However, events are never deleted throughout the game, which makes it possible to (re)activate an event lateron.

Conditions simply have one or more data items (e.g. the id of the player that has to be active). When the event/reaction that has the condition is triggered, it first checks whether the condition returns true; if not, the event/reaction is not triggered. In case of an event, the event stays active if the condition is not fulfilled.

Reactions finally can have conditions associated with them and several data items. An important thing to note is that the reactions associated with the event are triggered in order of their appearance. This is crucial when you e.g. place a unit, show some messages and remove the unit again.

Many events, reactions and conditions require the id of an object as parameter. If this id does not exist, they will usually not be triggered. Only in the case of conditions, this can make trouble; some will always return true, others always false.

6.2. Editing the event system

There are three dialogs to edit events, conditions and reactions. They all have some special properties, so I would concentrate on the specialties here.

When you add an event or reaction, a list of selection pops up where you select the type of reaction or event. The type cannot be changed lateron! You have to remove the event/reaction and add another one.

The event dialog displays all events. You can select one, edit its properties etc. Using the buttons, you can add reactions or conditions and add and remove events. New events are always appended at the end of the list.

When you click on the "reaction" button, the reaction dialog pops up. Most of the dialog is the same as the event dialog, i.e. you have edit fields to change the reaction values, associate conditions with the reaction etc. However, as the order of reactions is important (first reactions are triggered first), adding a reaction is a bit different. When you want to add a reaction, select the reaction before which the new reaction is to be inserted and click on "Add". To append a reaction at the end of the list, click on the last item in the list and then on "Add".

The condition dialog has another specialty. When you add a condition, a standard condition is appended at the end of the list. The type of the condition can be changed at any time using a dropdown.

6.3. List of events

Currently, there are 11 different events. Their types are:

KillAll	This event is triggered when all players have died except one. The neutral player does not count here, i.e. he may also still live.
PlayerDead	This event is triggered when a specific player dies. As parameter, it gets the player's id.
CityConq	This event is triggered if a city is conquered or razed and gets the city id as parameter. Note that a razed city cannot be conquered again and that razing is something that can happen. So make sure this event is not ignored.
ArmyKilled	When an army dies, this event gets to know it. It has the army id as parameter.
Round	This event is triggered at the beginning of a specific round with the round number as parameter. Use this event in connection with the Player condition to trigger some reactions at the beginning of a specific player's turn.
RuinSearch	Triggered when a ruin is searched. Gets the ruin id as parameter.
TempleSearch	The same for temples.
Dummy	This event is never triggered. It can become handy especially in connection with the RaiseEvent reaction and the Counter condition. An example is given in section 6.6
NextTurn	This event behaves a bit like the Round event, but is triggered at the next possible occasion. Together with the Player condition and the ActEvent reaction, this can be used by other events to signal a specific player that something has happened when other events have been triggered. See section 6.6. for an example.
StackKilled	This behaves like the ArmyKilled event, except that it awaits the killing of a stack. This event is redundand (it can be emulated using the ArmyKilled event, a !Dummy event, a !Counter condition and a RaiseEvent reaction), but may be interesting as a shortcut. When using this event, keep in mind that stacks can be joined (one of the two stacks is destroyed then), refilled and split. So if in doubt, use the ArmyKilled event instead. This event gets the stack id as parameter.
StackMove	This event has a position as parameter and is triggered when any stack moves to this position.

6.4. List of reactions

Currently, there are 15 reactions. These are:

Message	Probably one of the most standard reactions. It just displays some text in a message box. The text is also the parameter.
AddGold	This reaction adds gold to a player. It has two parameters, the player id and the amount of gold. Despite the name, by specifying negative gold amounts, gold can also be withdrawn.
AddUnit	This reaction adds a stack to a player. The player id is the one parameter, another is the position where the stack should appear. Finally, the stack can be selected using the stack dialog (a button "Stack" should appear when you select this event). If the position is blocked, the reaction tries to place the stack at one of the surrounding squares. If this doesn't work, too, then the reaction simply fails (this can be seen as a bug).

DelUnit	Kills a certain army. The army id has to be given as parameter.
Update	This reaction has no parameter. All it does is updating the game screen. This is useful e.g. when you add a unit and then display some message. Normally, the screen is not updated in between, so you would not see the unit. Using this reaction between the adding and the message, the unit will be displayed properly.
Center	This reaction centers the screen on a certain map position. It also updates the screen along the way.
CenterObject	This reaction does the same as Center, but centers on a certain object. The object can be a city, ruin, temple, stack or army. This is especially useful for moving objects (stacks, or better: armies).
CreateItem	Creates an item at a certain position. Parameters are the x/y value of the position where the item appears and the index of the item. Since the editor currently has no simple item viewer, I can only refer you to the file items.xml, where the item index can be found in between <d_index> and </d_index> or to the HeroItems page.
WinGame	Ends the game with a win. As a parameter, you can supply a number. The number is currently useless, but will propably be used in future versions for non-linear campaigns. For now, just ignore it.
LoseGame	The same as WinGame. The only difference between these two is the screen at the end of the game, which shows either a victory or a defeat. However, this screen currently doesn't even exist, but will do so soon.
RaiseEvent	Using this reaction, you can trigger another event. See section 6.6. for an example how to put this in use. The parameter is the id of the event to be triggered.
ActEvent	Enables/Disables an event. This reaction has two parameters, the event id and a value of 0(false) to disable or 1(true) to decide whether the event is enabled (1) or disabled(0). One use for this reaction are persistent events. Normally, an event is disabled after it has been triggered. However, by adding an ActEvent reaction to the chain that reenables the event, this event can be triggered infinite times.
RevivePlayer	Brings a dead player back to live. As argument it takes the id of the player. Note that the resurrected player has to get a city (see the TransferCity reaction) as well, else he may die very soon again.
KillPlayer	Kills a player. The player id is again the parameter. All stacks owned by this player are removed, all cities given to the neutral player.
TransferCity	Transfers the ownership of a city to a specific player. Obviously, the neccessary parameters are the id's of the player that gets the city and the id of the city itself.

6.5. List of conditions

Currently, there are 5 conditions, namely:

Player	Checks if a specific player is active. Useful e.g. in connection with the Round event, which is then triggered if a specific player's round starts. The parameter is the player id.
Counter	This condition is quite unique. It has a counter as parameter, and each time the event or reaction that this condition belongs to is triggered, the counter is decreased. When it reaches zero, this condition is fulfilled. See the example for an application of this condition.
Living	This condition checks if a specific player lives. The parameter here is the player's id.
Dead	Checks if a certain player is dead. The parameter here again is the player id.
Army	This condition is supplied with an army id and checks if this army is in the currently active stack (the active stack is the stack that does whatever is being done now, i.e. the one that

currently searches the ruin etc.). Useful e.g. with the StackMove event to determine that a specific army has moved to a specific location.

6.6. Some examples

a) Standard setup

- As a first example, I will describe the standard setup for a random game. I assume here some human and some AI players.

  What we want is: If a player dies, every other (human) player should be informed about the player's death. If a human player has killed every opposition, the game should be won, else the game is lost.

  The realization works somewhat like this. For each player X's death, we create a (deactivated) NextTurn event for every human player (apart from X if he is human) with a Message reaction displaying "Player X has died". Now we create a PlayerDead event with X's id and reactions that enable each of the NextTurn events. This way, when X dies, every other human player gets a message at the beginning of his next turn telling him that X is dead.

  Finally, we create a KillAll event. This event has, for each player X apart from the neutral player again, a Message reaction and a WinGame (if X is human) or LoseGame(if X is an AI player) reaction. Both these reactions have a Living condition with X's id associated with them.

  Note: If you don't create any events for your scenario, the game will never end! So some minimal events for winning/loosing the game are neccessary!

b) using the dummy event and the counter condition

- Here I want to describe a situation where the counter condition becomes handy. Imagine three ruins on the map. We want to have the player win the game when he has searched all three ruins. How do we do that?

  The answer is simple. Create a Dummy event with a WinGame reaction and a Counter condition. The conditions counter is set to three. Now, for each ruin, create a RuinSearch event (with the condition that the designated player is active) with a RaiseEvent reaction. The reaction is given the id of the Dummy event.

  What happens? Whenever our player searches a ruin, one of the three RuinSearch events is triggered. Their reactions then trigger the Dummy event. Since a ruin can be searched only once, the counter is decreased for every ruin that has been searched. When all three ruins have been searched, the counter finally reaches zero and the game is won.

  Note that in this example the case of another player searching the ruins is not handled. In this case, the game would never end. To capture this, you need to employ some more events.

7. Editing the scenario manually

If you want to do some excentric things which are currently not implemented or so, there are two possibilities.

The first one is to send a mail (preferably to our developer list). Since there will be only a few people who create maps, we will propably be so glad to have someone doing this (with the prospect of getting more maps

for FreeLords), that we will implement the issues relatively fast. Then you only need to wait for a new release or cvs version and are ready.

However, sometimes you do not want to wait, or you e.g. need to upgrade the scenario file to a new version. Then the only possibility is to edit the scenario file manually. Don't be discouraged, it is easier than it sounds.

The mapfile uses a specific XML format. Just open it with a text editor and you will propably be able to identify large parts of it. You should know three things when editing it, though (apart from doing a backup copy !!!):

1. Each object in the game (the stack, cities etc.) stores all its data between two tags. So a player stores all his data between <player> and </player>. Within these tags, all the subobjects are also stored. A stack that consists of armies will then store the data like this:

    ♦ <stack>

        ◊ ..stack data... <army>

            · ..data of first army...
        </army> <army>

            · ..data of second army...
        </army>
        ◊ ..
    </stack>

    The actual data is stored between tags that start with "d_". E.g. let us look at some player tag:

    ♦ <player>

        ◊ <d_id>4</d_id> <d_name>Evil Bad Genius</d_name>
        ◊ ..
    </player>

    That's most you need to know to edit the things.
2. The meaning of all the tags is documented in the file Savefile in the documentation directory.
3. When the version has changed, the version number is at the very first opening tag in the mapfile. By adjusting it to the new value, you can make the parser accept your map. However, be sure to read and apply the changes listed in the file Savefile for this new version number (or these numbers if you have skipped some), else the whole map may break apart. If you want to contribute the map to our game and are experiencing such a problem, we would also give some help with these problems.

That's it. If you find it a bit unsatisfying that the manual ends here, remember: Open Source lives from contribution, and the minimal contribution is some feedback *HintHintHint* 

Editor Manual (last edited 2006-05-02 14:45:25 by dejvid)