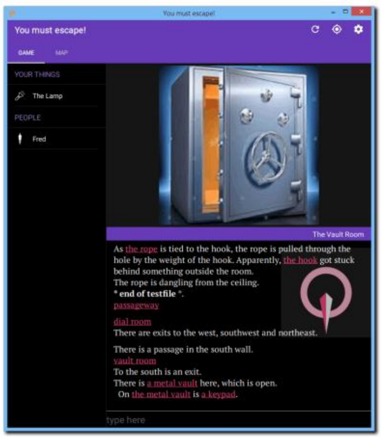
XVAN 2.6

-- IFI-XVAN --

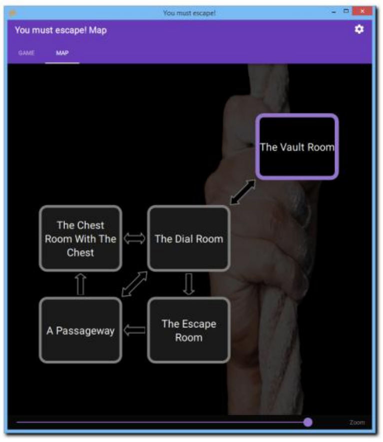
-- everything is a location, an object or a timer --



As the rope is tied to the hook, the rope is pulled through the hole by the weight of the hook. Apparently, the hook got stuck behind something outside the room.
The rope is dangling from the ceiling.
*** end of testfile ***.
passageway

dial room
There are exits to the west, southwest and northeast.

There is a passage in the south wall.
vault room
To the south is an exit.
There is a metal vault here, which is open.
On the metal vault is a keypad.



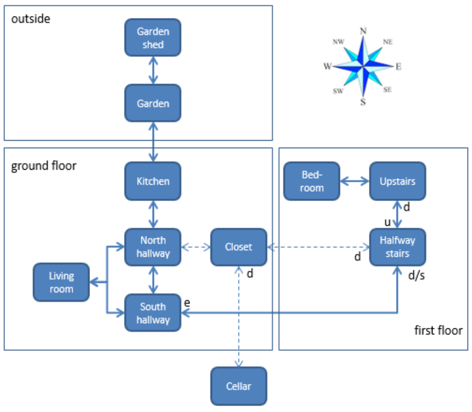


Table of Contents

Introduction

This document describes IFI-XVAN, the XVAN version with a Graphical User Interface (GUI). This document does not go into the “normal” operations of XVAN, these are all described in the following XVAN documentation:

- XVAN Installation and user guide;
- XVAN Introduction;
- XVAN Functions;
- XVAN Syntax;
- XVAN Tutorial;
- XVAN Library.

This document focuses on the extra functionality offered by IFI-XVAN.

About IFI-XVAN…

As described in the XVAN Introduction document, XVAN is a text adventure authoring system. It is a compiler, an interpreter and an authoring language.

IFI-XVAN comes with a special version of the interpreter that does not directly interact with the user. Instead, it connects to a Graphical User Interface (GUI) and the user interacts with the GUI. The GUI is called the front-end and the interpreter is called the back-end.

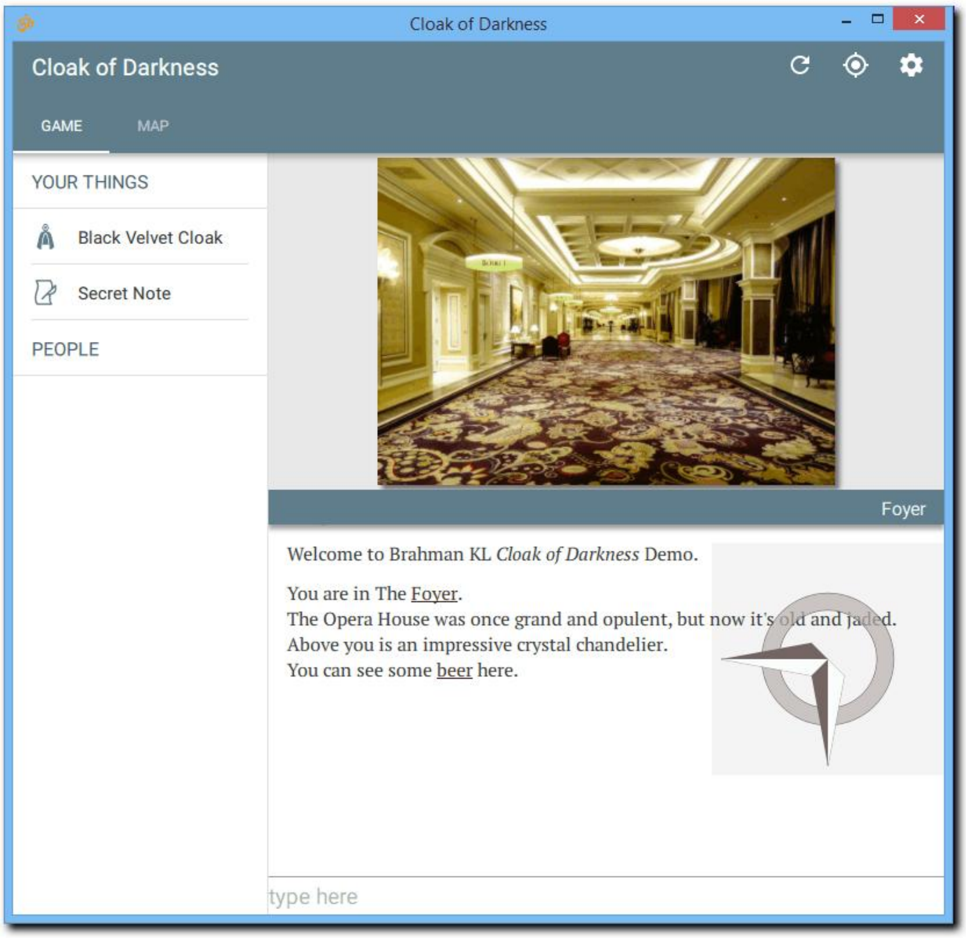
IFI-XVAN has the same compiler and authoring language[1] as “normal” XVAN.

Functional

When compared to the console and Glk versions, IFI-XVAN has additional graphical capabilities, e.g.

- Graphical display of the map;
- Display background pictures for locations;
- Display the player’s inventory as icons;
- Display the player’s current location;
- Clickable compass rose to move around;
- Clickable object names with predefined actions in the printed text.
- …

[1] With a few additions



Welcome to Brahman KL *Cloak of Darkness* Demo.

You are in The Foyer.
The Opera House was once grand and opulent, but now it's old and jaded.
Above you is an impressive crystal chandelier.
You can see some beer here.

Communication between interpreter back end and GUI is done through added XVAN functions. The author needs not to worry about this when they use the IFI Library. The IFI-library has normal XVAN code that handles the communication with the GUI. The IFI Library requires that the XVAN Library be used as well.

But the story author also has the option to communicate with the GUI directly by using the newly added XVAN functions, as described in the technical paragraph in this document.

Story info
The story info section in the story source file now allows the author to set some initial values for the GUI, like background color, enabling autolinks, enabling compass, etc

Following keywords are available (if they are not in the story file, default values are used):

Keyword	Purpose	Default	Remark
title	Story title	“”	
author	Story author	“”	
organization	Producing organization	“”	
covertext	Text on first screen	“”	
credits	credits	“”	

choice_mode	Story runs in choice mode	--	
hybrid_mode	Story runs in hybrid mode	--	
version	Story version	“1.0.0”	Format is “x.y.z”
Android market	Link to android app	“”	
Ios market	Link to ios app	“”	
backimage	Path to background image for map	“”	Path is relative to datadir
effect			
no_sidebar	Hide the sidebar in the GUI	--	If omitted, a sidebar will be displayed in the GUI.
no_textinput	No textinput possible, only (mouse) clicks	--	If omitted, the GUI will allow textinput.
no_compass	Hide the compass in the GUI.	--	If omitted, a compass will be displayed in the GUI.
primary_color	Background color		
autolink	If true, the front-end will try to map nouns to objects and make them clickable.	--	XVAN sends clickable nouns by itself, so the autolink feature needs not to be used.

Technical

Note: in case you’re not interested in how ifi works “under the hood”, you may skip this section. It is not necessary knowledge for using IFI-XVAN functionality.

Front-end

The GUI used in IFI-XVAN is the Brahman GUI developed by Strandgames. This GUI has an open interface whose specs are here.

The GUI is not specifically for XVAN, any back-end can connect to it, provided it abides by the interface rules.

Back-end

The IFI-XVAN back-end is formed by an adapted version of the XVAN console interpreter. It no longer has a console or Glk window, but it receives its input from and sends its output to the GUI. The back-end is no longer an executable, but a shared library (.so for linux or .dll for windows).

Communication between back-end and front-end

The back-end and front-end exchange information via JSON text strings. JSON is an international standard, for more information check www.json.org. The GUI interface specs mentioned above define the syntax and semantics of the messages exchanged between front-end and back-end.

Examples:

When the user types “unlock the chest with the key” in the GUI, the GUI will send the following JSON message to the back-end: {“command” : “unlock the chest with the key”}. The “command” part tells the back-end that it should handle the upcoming string as user input.

After executing the user command, the back-end will send a reply JSON to the GUI. In our example this could be for example: {“text” : “The chest is now unlocked.”}. The “text” part tells the GUI that it must print this text in the text pane.

Besides the command and text, there are a lot of other messages that can be exchanged. E.g. there are messages for:

- Sending the player’s inventory to the GUI for display as icons;
- Sending map data to the GUI so it can draw the map;
- Sending the player’s current location to the GUI to highlight on the map;
- Sending possible exits for the current location so the GUI can draw a clickable compass;
- Etc, etc.

Communicating directly with the GUI from within an XVAN story
There are 3 XVAN-functions available to communicate directly to the GUI:

- Clearjson()
- Addjson()
- Sendjson()

The clearjson() and sendjson() functions have no parameters, the addjson() functions has the following parameters (table copied from XVAN functions document).

addjson()	Parameter 1	Parameter 2	Parameter 3	remarks
	string	location object	--	parameter 2 is optional

The IFI_XVAN interpreter maintains an internal json string that can be manipulated by these three functions:

Clearjson() clears the internal json string.
Addjson() adds the string parameter as text to the internal json string. In case a location or object is specified as the second parameter, it’s id number is added as text to the json string (e.g. id 4503 is added as 4 ASCII characters ‘4’, ‘5’, ‘0’ and ‘3’).
Sendjson() sends the internal json string to the GUI, but does not clear it afterwards.

Note: when using these functions, the author must have an understanding of the json format. Strings that do not comply with the json standard will be ignored by the GUI and it will not send an error message.

The ifi-actions in the IFI Library also use these functions. As an example, let’s have a look at the ifi_loc verb. Ifi_loc sends the player’s current location to the GUI, so it can be highlighted on the map:

```
$VERB ifi_loc
 DEFAULT
  clearjson()                          # just to make sure it's empty
  addjson("{\"location\":", l_location)  # \" prints a " in the string
  addjson("}")
  sendjson()
  clearjson()                          # as a courtesy to the next caller
ENDVERB
```

Basically, this verb builds the string {"location":location_id} and sends it to the GUI.

There are much more complicated ifi-actions. For example the verb ifi_items makes all objects held by the player work together to create 1 json string.

How do I "ifify" my XVAN story?

With the IFI Library, it's easy to enable your story for IFI-XVAN. With following steps you can take advantage of the additional GUI capabilities:

1. In descriptive text use [o_names] rather than plain nouns. E.g don't print:
   "There is an old chest here."
   but instead print:
   "There is [a] [o_chest] here."
   Why? The second line prints the description of the chest and adds markup to it, which makes it clickable on the screen. When clicked, the GUI sends the command "use <object>" to the back-end.

   The XVAN Library (which is required when using the IFI Library) has a default 'use' verb that:
   - examines the object if it was not examined before, or
   - tries to take the object if it is already examined and it is takeable.
   But of course, you can define an objects own response to the use command. E.g for a lamp, there may be a local trigger t_use connected to "use lamp" that will turn on or off the lamp, depending on its state.

   Secondly, you may drag an item from the sidebar and drop it on an item in the text, e.g. drop the key item on the chest hyperlink. This will send the command use key with chest to the back-end.
   XVAN Library default actions for use X with Y (depending on context) are:
   - ask Y about X (if both X and Y are alive)
   - give Y to X (if X is alive)
   - unlock Y with X (if X is a key and Y is locked)
   - put X in Y (if Y is a container)
   - put X on Y in all other cases.

2. Make sure to use the following directions in the story and make sure they are known by these names (synonyms with other names are allowed):

N, NE, E, SE ,S , SW, W, NW, Up, Down, In, Out.

3. Use attribute r_ifi_picture for background pictures for locations. The background picture will be displayed when the player is in the location. Create a (local) description with a text string that holds the path name to the picture and assign that description to r_ifi_picture. In case there are more than 1 possible pictures, use more descriptions and change the value of r_ifi_picture at the right time. The path name will be relative to the ifi parameter configdir which has default value: <configdir default value>.

   Example:

```
….
DESCRIPTIONS
  d_background   “resources\\background.png”
  …..
ATTRIBUTES
  r_ifi_picture = d_background
….
```

4. Use attribute r_ifi_gx and r_ifi_gy with locations to denote the relative positions on the map. Note: gx goes from left to right, gy goes top down (and *not* bottom up). gx and gy are integers.

5. Use attribute r_ifi_icons for icon pictures for objects. The icons will be displayed in the sidebar as the player’s inventory and as the people the player has already met. Create a (local) description with a text string that holds the path name to the icon and assign that description to r_ifi_icon. In case there are more than 1 possible icons (e.g. with lamp on, icon with light beam), use more descriptions and change the value of r_ifi_iconm at the right time. The path name will be relative to the ifi parameter configdir which has default value: <configdir default value>.

6. For each location that must show on the map add the following line to the TRIGGERS-section:
   "ifi_map" -> t_ifi_place

7. For each character that must show in the GUI’s people section, add the following line in the TRIGGERS section:
   "ifi_people" -> t_ifi_people

8. “ifi_map” and “ifi_people” have scope all_locs. In case you have locations or objects with a t_default trigger, this trigger will fire every turn for these ifi actions (unless they are added in step 6 or 7 of course). So, make sure to check for “ifi_map” and ”ifi_people” in your default trigger if it must not fire.

   Example:

```
if equal(%action, ifi_map) or equal(%action, ifi_people) then nomatch() endif
```

Compatibilty

With IFI added, there are now 3 versions of the XVAN interpreter: console, Glk and IFI. Although Glk and IFI have some additional functions, it is not necessary to make special versions of your story for Glk and IFI.

- the console interpreter will ignore the Glk and IFI functions in a compiled story file;
- the Glk interpreter will ignore the IFI functions;
- and the IFI interpreter will ignore the Glk functions.

Likewise, a story with only console functions can still be played in the Glk or IFI interpreter. It will just not take advantage of the extra possibilities (and the map will be messy, because all locations will have coordinates 0,0).

Where to get IFI Xvan

IFI XVAN is best downloaded from the Brahman repo that is maintained by Strandgames. It containes the GUI and the XVAN backend, as well as instructions how to build the application.

Using the IFI Library in your story

The IFI Library is a single file called something like “IFI Library x-y.lib”.

The file must be placed in the same folder as your story file(s). In your story file, insert the .lib file with the line:

- $insert ".\\IFI Library x-y.lib" or
- $insert “.//IFI Librray x-y.lib”

And please note that the IFI Library also requires the use of the XVAN Library.