The KsirK Skin Editor Handbook

Gaël Kleag de Chalendar

Contents

List of Tables

Abstract

KsirK is a computerized version of a well known strategy game. It is fully skinable. This manual explains how to create new skins from scratch, using a file browser, a SVG image editor and KsirK Skin Editor.

Chapter 1

Introduction

KsirK is a computerized version of a well known strategy game. It is fully skinable. This manual explains how to create new skins from scratch, using a file browser, a SVG image editor and KsirK Skin Editor.
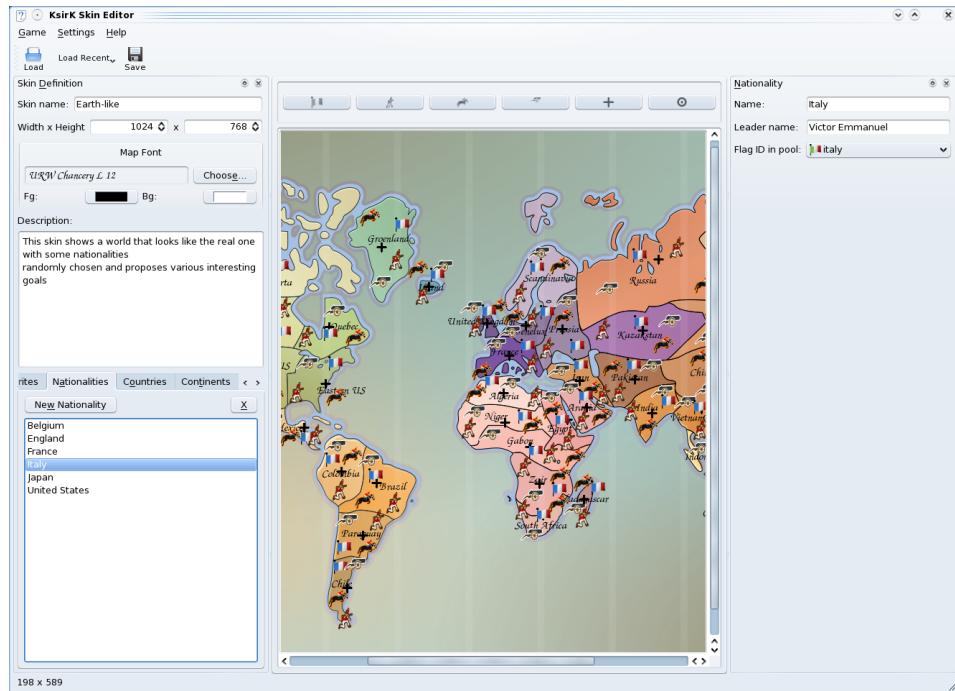
There is three main steps to create a new skin: creating the folders hierarchy; creating the graphics, all pooled inside a SVG file; and then placing sprites and naming things using KsirK Skin Editor.

Chapter 2

Creating a KsirK skin

KsirK is fully skinable. For who have graphic capabilities, creating a new skin is as simple as copying an existing one, editing or changing the graphics and updating the data in a `.deskt op` configuration file, which is done using the KsirK Skin Editor. A last step is to edit the file describing what to install.

All examples (text and images) are taken from the default skin of KsirK.

2.1 Structure of folders and files

- `Data/` : Data used by the program to handle the images of the skin.

  - `CMakeLists.txt` : Describes what to install.
  - `onu.desktop` : Main file for the description of the skin. See its description below.

- `Images/` : The images of the skin (map, sprites, etc.)

  - `CMakeLists.txt` : Describes what to install.
  - `pool.svg` : The map and all the sprites used in this skin.
  - `map-mask.png` : The image allowing the program to find which country the mouse is over during the game (see below).
  - `*.png` : The other images here are buttons images.

- `CMakeLists.txt` : Describes what to install.
- `Sounds/` : Sound files for the skin.

2.1.1 The Data folder

This folder contains only one file, `world.desktop`. This file is a description of the content of the skin. It is used to describe the organization of the world and how to display it. You don't have to fill it manualy as it will be handled by the KsirK Skin Editor. But, for the sake of completeness, we will now describe its structure.

An `onu` group contains a number of country entries, a number of nationality entries and lastly a number of continent entries. Then various groups contain the description of the different sprites, the description of the different countries, continents, nationalities and goals.

The following sections describe the content of each group.

2.1.1.1 The `onu` group

Entry name	Signification
`width`	The width of the skin's map.
`height`	The height of the skin's map.
`skinpath`	The path of the skin relative to the application data path (e.g.`skins/default`).
`nb-countries`	The number of countries (42 in the default skin). There should be the same number of country groups below.
`nb-nationalities`	The number of nationalities (6 in the default skin). There should be the same number of nationality groups below.
`nb-continents`	The number of continents (6 in the default skin). There should be the same number of continent groups below.
`pool`	The path and name of the pool file relative to the skin path (e.g.`Images/pool.svg`).
`map-mask`	The path and name of the map mask file relative to the skin path (e.g.`Images/map-mask.png`).
`format-version`	The version of the skin file format (2.0).
`name`	The display name of the skin.
`desc`	The long description of the skin.
`fighters-flag-y-diff`	The height difference between flags and cannons sprites.
`width-between-flag-and-fighter`	The number of pixels between the left most pixel of the flag and the right most pixel of the simple cannon (when not firing nor exploding).

Table 2.1: The onu group entries

2.1.1.2 The sprites description groups

Each type of sprite (flag, infantry, cavalry, cannon, firing cannon and exploding cannon) is defined by a group. Only some sprites have a width entry. This entry is used for relative positioning during animations: the cannons, firing or exploding should not 'move' around the country's flag during fight.

Entry name	Signification
`width`	The reference width of the flag sprite frames.
`height`	The reference height of the flag sprite frames.
`frames`	The number of frames of the sprite.
`versions`	The number of versions of the flag sprites.

Table 2.2: Sprite description group entries

2.1.1.3 The countries description groups

Each country listed in the `onu` group, countries entry has its own group whose label is the country name. The table below lists the entries of these groups.

Entry name	Signification
`id`	The integer unique identification number of the country, must start at zero (0).
`name`	The displayed name of the country.
`<sprite>-point`	Each type of sprite (flag, infantry, etc.) in the country is displayed at a certain position defined by this entry made of two integers separated by a comma.
`neighbours`	The list of the ids of the neighbors of the current country.

Table 2.3: Country entries

2.1.1.4 The nationalities description groups

Each nationality listed in the `onu` group, nationalities entry has its own group whose label is the nationality name. The table below lists the entries of these groups.

Entry name	Signification
`name`	The name of the nationality (e.g. `Japan`).
`leader`	The name that will be proposed to the player choosing this nationality.
`flag`	The name of the country's flag sprite element in the `onu` SVG file.

Table 2.4: Nationality entries

2.1.1.5 The continents description groups

Each continent listed in the `onu` group, continents entry has its own group whose label is the continent name. The table below lists the entries of these groups.

Entry name	Signification
`name`	The name of the continent (e.g. `Africa`).
`id`	The integer unique identification number of the continent.
`bonus`	The number of armies obtained at the end of the turn by the player owning all the countries of the continent.
`continent-countries`	The list of the ids of the countries inside the current continent.

Table 2.5: Continent entries

2.1.1.6 The goals description groups

Each goal listed in the `onu` group, goals entry has its own group whose label is the goal name. The table below lists the entries of these groups.

Entry name	Signification
type	The type of goal. Can be continents, countries or player.
nbArmiesByCountry	Set the minimal number of armies the player will have to put on each country.
nbCountries	Set the number of countries the player must possess.
nbCountriesFallback	For a player goal type, if the target player is killed by another one, this sets the number of countries the player will have to conquer instead.
desc	A full text description of the goal with appropriate placeholders (needs more doc here).
continents	The list of continents ids the player will have to conquer.

Table 2.6: Goal entries

2.1.2 The Images folder

This folder contains the buttons specific to KsirK and the pool SVG file that contains the map and all sprites.

2.1.2.1 The **pool.svg** file

This SVG file contains the map and all the sprites of the game. Each element is named such that the game can select and render separately each of them.

In the default skin, the sprites are flags, infantry, cavalry and cannon. Infantry represents one army, five for cavalry and ten for cannons. Cannons are also used to display fighting armies. That's why there is three kinds of cannon sprites: normal static or moving cannon, firing one and exploding one.

Each sprite image is the concatenation of the various views of the object and for each view, its frames. Views are organized vertically and frames horizontally. Where to cut the image in individual frames is found by dividing the height of the image by the number of views and the width by the number of frames. These data are found in the `onu.desktop` file.

An example of a sprite image: `cannon.png`

For armies sprites, there is three different views, from top to bottom:

- looking right

- looking left and

- facer

For flags, there is only one view. All sprites backgrounds should be set to transparent.

An example of a flag sprite image: `italy.png`

Element name	Sample image	Signification
map		Map of the skins's world. Countries can have arbitrary shapes, but should be grouped in eye-recognizable continents, by location only or by color.
italy		The flag of the Italy nation. There is one such entry for each nation.
infantry		Icon representing one army.
cavalry		Icon representing five armies.
cannon		Icon representing ten armies.
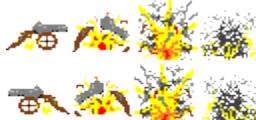
`firing`		Icon representing armies during a fight.
`exploding`		Icon representing armies when losing a fight.
`alaska`		The shape of the Alaska country. This is used to highligth the country. There is one such shape for each country. Its color is not used. As such, the color is free.
`reddices`		The red dices used to show battles results.
`bluedices`		The blue dices used to show battles results.
`mark1`		This mark is added to fighting sprites to show the number of armies they represent, here 1. There is also a `mark2` and a `mark3` elements.

Table 2.7: The elements of the pool

2.1.2.2 The `map-mask.png` file

This is a png file representing also the map of the skins's world, but with coded colors that allows one to identify uniquely each country. Countries must have at least the same shapes as the corresponding countries on the `map.png` file so that users can click anywhere in a viewable country and select the good one. Countries, like little islands as Indonesia in the default map can be greater than the equivalent in the `map.png` file to facilitate the selection of the country.

The blue component (in RGB model) of a country's color identifies the country: index 0 is country 0 in the `onu.xml` file, index 1 is country 1, etc. White (0xFFFFFF in RGB) identifies the absence of

country (sees or, why not, no man's lands).

2.1.2.3 The buttons images

The filenames are hard-coded in the code, so they should not be modified. The table below shows all the buttons that should be provided.

| Image	Filename	Signification
	`newNetGame.png`	Tries to join a network game.
	`recycling.png`	After all players have put their armies, it is possible to modify the distributions (named recycling). This button asks to do such a redistribution.
	`recyclingFinished.png`	Chooses not to redistribute. All clients windows must have this button clicked to really end the distribution.
	`nextPlayer.png`	Ends the current player's turn and switch to the next one or to the next turn if the current player was the last one.
	`attackOne.png`	Chooses to start an attack with one army.
	`attackTwo.png`	Chooses to start an attack with two armies.
	`attackThree.png`	Chooses to start an attack with three armies.
	`defendOne.png`	Chooses to defend an attacked country with one army.
	`defendTwo.png`	Chooses to defend an attacked country with two armies.
	`moveArmies.png`	Chooses to start the moving of armies between two countries (last action in the turn).

Table 2.8: The buttons of the GUI

2.1.3 The Sounds folder

This folder contains three sound files:

- `roll.wav` played when armies are moving

- `cannon.wav` played when firing and

- `crash.wav` played when a cannon explodes

2.2 Using the KsirK Skin Editor

After creating the graphics in the right folder and SVG file, you just have to create countries, players, etc. and associate them with the right SVG element in the pool. This can be done with the KsirK Skin Editor. Note that each part of this application is auto-described by tooltips and contextual help usually reachable through the **Ctrl**+**F1** shortcut.
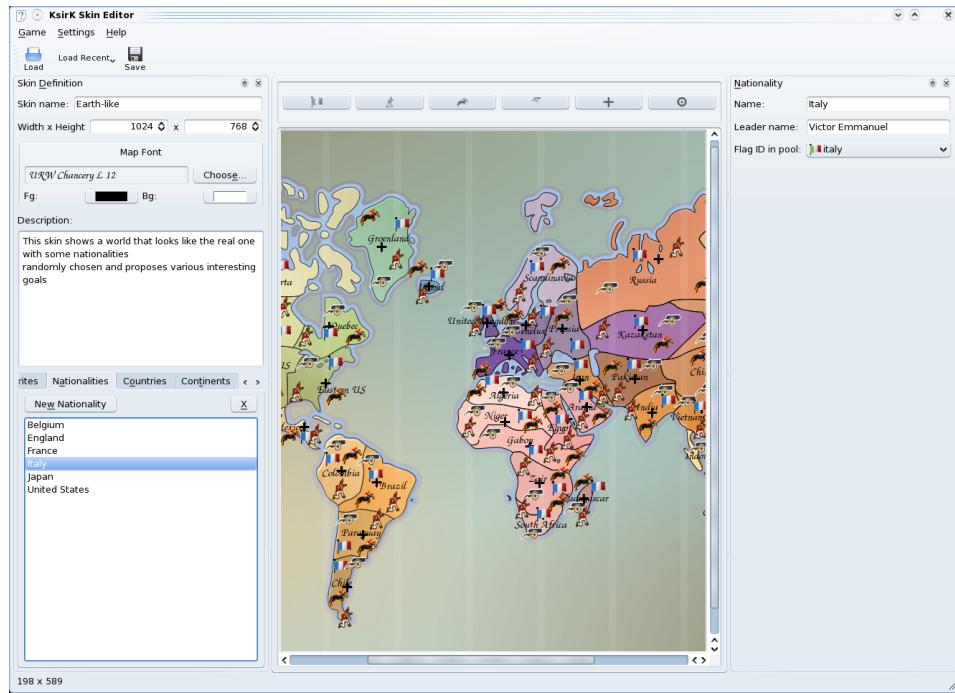
Chapter 3

Command Reference

3.1 The main KsirK Skin Editor window

3.1.1 The buttons

Load...
Causes the program to load an existing skin.

Save
Saves the currently edited skin.

Chapter 4

Developer's Guide to KsirK Skin Editor

Please refer to https://api.kde.org/ for the documentation of the API.

Chapter 5

Questions and Answers

This document may have been updated since your installation. You can find the latest version at https://docs.kde.org/ .

Chapter 6

Credits and License

KsirK Skin Editor

Program copyright 2009 Gaël de Chalendar kleag@free.fr

Documentation copyright 2009 Gaël de Chalendar kleag@free.fr

This documentation is licensed under the terms of the GNU Free Documentation License.
This program is licensed under the terms of the GNU General Public License.