SVG Player Project

Svetlana Marchenko
Saint Petersburg Academic University –
Nanotechnology Research and Education Centre RAS
Saint Petersburg, Russian Federation
svemarch@gmail.com

Abstract

NS-2 Network simulator is being used in a huge amount of scientific research. Nokia-Siemens Networks uses customized clone of this simulator. This clone is able to produce output about bursts as SVG picture series. The main goal of this project is to develop a tool for putting SVG images from series together and playing it as a video stream with forward/back navigation. We need to consider irregular time intervals between SVG images at the stage of playing the series of images as a video stream.

In this article architecture of SVG Player application and its current functionality are described.

Index Terms: SVG player, NS2.

I. Introduction

Network Simulator (Version 2), widely known as NS2, is simply an event-driven simulation tool that has proved useful in studying the dynamic nature of communication networks. Simulation of wired as well as wireless network functions and protocols (e.g., routing algorithms, TCP, UDP) can be done using NS2. In general, NS2 provides users with a way specifying such network protocols and simulating their corresponding behaviors [1]. The user describes a network topology, and then the main NS program simulates that topology with specified parameters.

Nokia-Siemens Networks uses customized clone of this simulator. This clone is able to produce output about bursts as SVG picture series. SVG images, resulted from simulation, need to be revised for making the network behavior analysis. Viewed images should follow in the order they were generated. At the same time it should be noted that time intervals between the moments of generating these SVG pictures during simulating could be different(irregular). Their duration is considered for the analysis.

Because of large quantity of generated SVG images, viewing and the analysis them manually seem to be difficult. Therefore, an application that could be used for loading a series of sequentially generated SVG images and play it as a video stream considering the irregular time intervals is needed. User should have the ability to stop the playback and go to the next picture with navigation tools in case the interval between successive images takes a long period of time. The capability to save the information about loaded SVG images as series (their names, time intervals and relative paths to images) in XML-based file is an another requirement for the application.

The main aim of this project is to develop application SVG Player – tool answered these specific requirements.

II. Architecture and implementation

The following development tools have been chosen: C++ Programming Language, Qt Framework and Qt Creator IDE.

Application development was started with the implementation of defined tasks but only for random SVG images. In that, linkage each image to concrete time interval was considered. As a simplification, on this implementation stage each image was related to time interval (with) the same value, but considering the fact that these intervals could be irregular. Also the format for special XML-based files with meta-information about the series was worked out.

On the next stage the decision about the way of associating generated images and actual simalation times was needed to define. To generate the names of output SVG images with specified format name@nnnnnn.svg (where nnnnnn – is a quantity of miliseconds from the beginning of simulating to time this image generated) seemed to be the most suitable solution of this problem. In this way time intervals after loading images with specific names format are computed by calculation the difference between values nnnnnn. As a preliminarily, all images in the series are sorted by their generating time values. In addition to this capability to work with series of images which names are not in the defined format, is remained in application.

For now, SVG Player has the following features:

- putting the series of SVG images;
- viewing them with forward/back navigation;
- playing the series as a video stream in direct and reverse orders;
- editing the series by adding/deleting pictures;
- saving/opening the series with XML-based files specified with extension .set;
- supporting the specific format of naming svg files.

Several screen shots showing the mentioned functionality are demonstrated in Fig. 1.

The project SVG Player was developed by using design pattern MVC (Model-View-Controller) [2]. Distribution responsibility between (among) the classed in the application according to chosen pattern is shown in Table 1.

SvgSeries and Shot classes are responsible for storing and managing information about the series: file names, xml-based file name (not necessary), time intervals values.

CentralWidget — shows SVG images depending on the time their loading required: in case a period of loading time exceeds the defined threshold value, on the widget «Loading...» sign is shown.

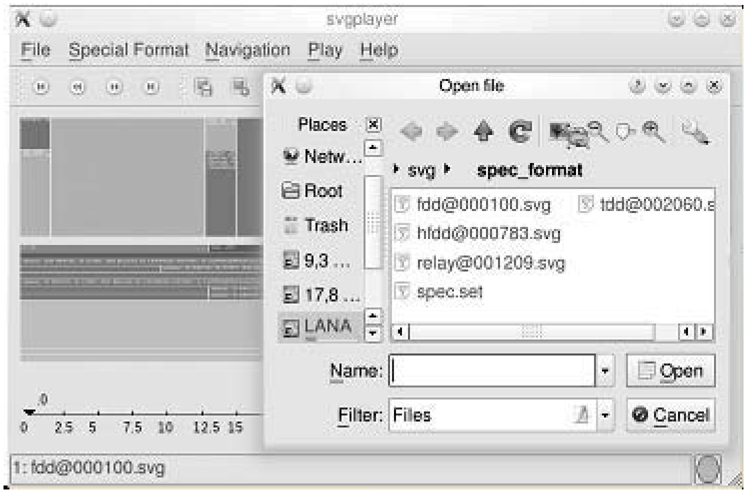
StatusBar и LoadIndicator are used for view the status of file loading, name of loaded image and this image number in the series.

TimeSlider shows time scale for the loaded series at each time moment the slider on the scale holds a corresponding position.

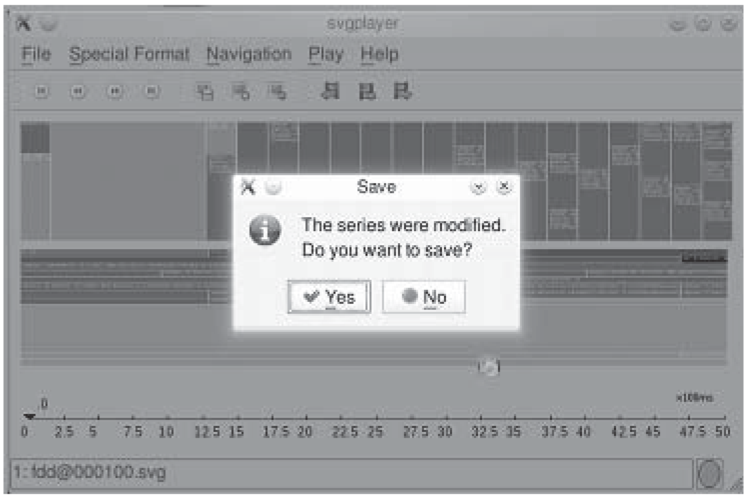
MainWindow involves (or include) the logic of application behavior: openning file dialog for putting SVG images and set files into the series, saving set files — and it runs playback the series and navigation between loaded images.

SvgLoader responses for the loading process of SVG pictures in an thread separated from the user graphical interface thread [3].

SvgSeriesHandler is for reading special XML-based files (with extension set) by using Simple API for XML and composition the SVG images series on basis of information from set file [4].



a) opening files



b) save the series

Fig. 1. SVG Player application screen shots

TABLE I
CLASSES` RESPONSIBILITY

Model	View	Controller
SvgSeries Shot	CentralWidget LoadIndicator StatusBar TimeSlider	MainWindow SvgLoader SvgSeriesHandler

III. PLAYING AS A VIDEO STREAM

During implementation the struggle with playback images` series as a video stream appeared: in case duration of loading exceeds the time interval after which the next picture should be shown. Decision to synchronize playback timer and a period of time loading was accepted as a solution of this problem. But another effort appears: the application should not seems for user to be buzzed because of the main window becomes to inactive status for a period of time loading. Solution to load a picture in another thread separated from the main graphical user interface thread was passed.

Primordially class QsvgWidget was used for viewing SVG images on the CentralWidget, it requires only picture file name and all details about loading the file and displaying process was buried inside(within) QsvgWidget implementation. The consequence of the solution to use another thread for loading is that we should run(manage) loading and rendering pictures by using class QSvgRenderer. Thereby the responsibility for loading and displaying processes was separated between different classes.

The sequence of actions required for loading and displaying pictures in Fig. 2 is illustrated.

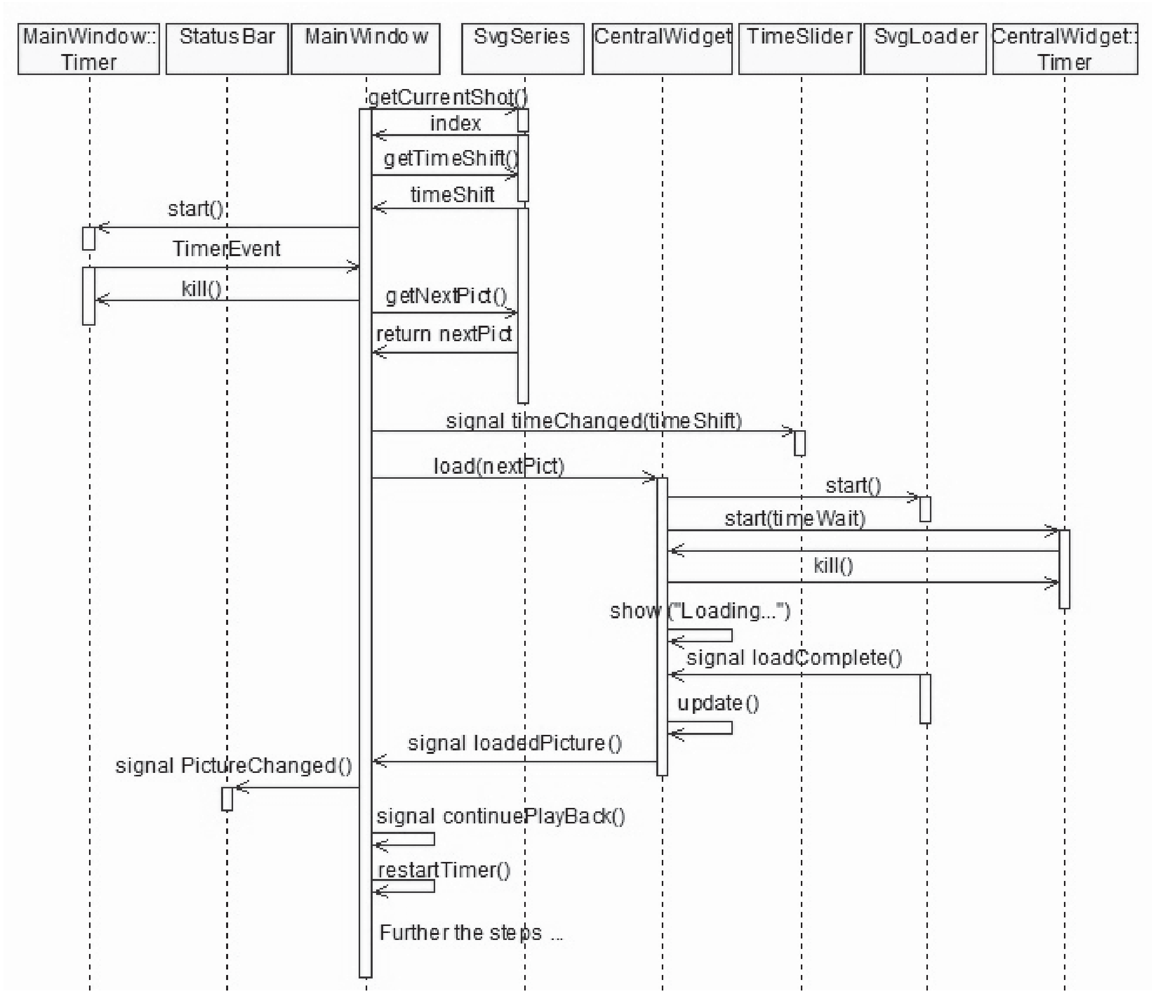


Fig. 2. Sequences diagram

First of all MainWindow object calls information about the image which is shown at this moment (picture index in the series and the time interval for displaying the next picture) and runs the timer with that time interval. When the timer event is received, Main Window object sends the signal timeChanged(timeShift) for updating TimeSlider objects state on the widget.

Now the next image should be displayed and MainWindow calls for CentralWidget load(nextPict) method. For this process the specific mechanism of loading pictures is used: Central Widget calls SvgLoader to start in the other thread and starts timer at once to track if loading process takes too much time. Then if the timer event from CentralWidget Timer occurred before the next image was loaded, CentralWidget will display on the widget for user that there is loading process. Only at the moment the image will be completely loaded and signal about this (loadComplete) will be received, the CentralWidget will be updated. By sending signals loadedPicture() and pictureChanged() MainWindow and StatusBar objects are informed that new image is shown. And signal continuePlayback() runs this loading mechanism again for the other picture with new time interval until last image in the series would be displayed.

VI. Conclusion

The SVGPlayer is mostly finished. Beside showing NS2-related images it can be used for playing any SVG series as cartoon. Installation packages and sources are available on https://sourceforge.net/projects/svgplayer/.

During using SVG Player application for pictures with different size, one aspect was noticed: it could be difficult to analyze some processes by using this player for picture series with big size of image files and too short time intervals between them. Therefore, loading process will take more time than size of time intervals. For this case could be alternative solution, for example, load all pictures or only part of them before they will be shown. But this solution also has an imperfection: series with large quantity of pictures could require too much memory. So, I think the right solution depends on data what SVG Player will be used for.

Acknowledgment

The author would like to Finnish-Russian University Cooperation in Telecommunications which supported this activities (see: http://fruct.org/node/2539).

References

[1] I. Teerawat, H. Ekram, “Introduction to Network Simulator NS2,” *Springer*, pp. 19–20, 2009.
[2] E. Gamma, R. Helm, R. Johnson, John M. Vlissides “Design patterns: Elements of Reusable Object-Oriented Software,” p. 374, 1994.
[3] J. Blanchette, M. Summerfield, “C++ GUI Programming with Qt 4,” *Prentice Hall*, p. 736, 2008.
[4] M. Shlee, “Qt4.5. Professional Programming in C++,” *BHV-Petersburg*, p. 896, 2009.
[5] “Qt Reference Documentation,” URL: http://doc.qt.nokia.com/4.7-snapshot/index.html