GRAPHREC

SPECIFICATION

Version 1.01

Prepared by Petr Koupý <petr.koupy@gmail.com>

Last Updated on November 21, 2009

Table of Contents

Revision History

Name	Date	Reason for Changes	Version
Petr Koupý	Jun 9, 2009	Initial Draft	1.0
Petr Koupý	Nov 21, 2009	Public Release Edit	1.01

1 Introduction

1.1 About Specification

This specification is a starting point for the creation of GraphRec 1.0. Topics covered by this specification consist of functional requirements. Final product will reflect every requirement described in this document. Some design decisions however might be altered during development, mainly because of constraints caused by third party components. Specification will be kept up to date and all major changes will be recorded in the revision history. Open issues are marked throughout the text. Wording in this specification should not be taken strictly but rather abstract. For example, when there is discussion about providing some function with reference to another object, it should not be comprehended as reference in sense of C++. It is just any reference type provided by one particular language.

1.2 Project Overview

"Project GraphRec" is the codename for new graph recording software. Main goal is to develop program that will capture image sequences and video of moving entities on generic graph consisting of nodes that are connected by edges. Each node can be occupied by at most one entity at the same time. Entity can move from one node to another through existing edge. Arbitrary number of entities can move simultaneously. Since time line consists of discrete time steps, all parallel moves will start and finish together despite various distances between nodes. Here is a brief description of how program is supposed to work:

- User provides input file containing graph representation, initial positions of entities and complete list of entity movements.
- Program fetches input and calculates position of nodes to produce reasonable graph layout.
- User can interactively adjust and save the appearance of graph (e.g. layout, coloring).
- In a similar way as with video recorder, user can start, stop, seek and record animation. Animation can be captured either as an image sequence or as a video file.

1.3 Project Purpose

The purpose of the program is to act as a visual frontend for existing text based applications that deal with path-planning problems. Program might be used as a teaching or presentation tool, which helps people to visualize how presented path planning algorithm works. Image and video outputs might accompany scientific articles or posters either on a paper or on the web pages.

GraphRec 1.0 will be created with respect to requirements of *multirobot* – the path planning software currently developed on Computer Science department of Charles University in Prague by RNDr. Pavel Surynek, Ph.D. (http://ktiml.mff.cuni.cz/~surynek/). Since this puts a limit on universality of the application, there will be made an effort to produce easily adjustable code and flexible architecture, so that some possible future version could be more general and compatible with more path planning applications.

1.4 Project Scope

Program should not be considered as graph lay-outing software. Although program will implement some layout algorithms to provide decent looking graphs, it is not a main goal. Program also should not be mistaken with applications that actually solve path-planning problems. Core functionality will be focused on animation, interactive user interface and production of image and video output. In a model situation where there are three types of software – one for graph creation, one for path planning and one for visualization – the program is supposed to be the third one. Program will be strictly GUI application not providing any kind of command line interface.

2 Overall Description

2.1 Features

- Input
  - User can load multiple files at once.
  - User can multi-select what to load in case that file contains more than one graph.
  - Validation and possibility to review error log.
- GUI
  - Multiple independent tabs. Each tab contains context of one graph.
  - More than one tab can be viewed at once.
  - Zooming, scrolling and rotating.
  - User can select, move or color multiple graph nodes at once.
- Layout
  - Algorithms for automatic graph lay outing.
  - Ability to adjust graph layout manually by selecting and dragging nodes.
  - Ability to color every part of graph (e.g. vertices, edges, entities, background, labels).
- Animation
  - Ability to start, step, stop and seek animation.
  - Variable speed of animation.
  - Possibility to highlight moving entities.
- Capturing
  - Manual image snapshots.
  - Automatic capturing of image sequences (with adjustable interval).
  - Video capturing.
- Output
  - Both raster and vector image formats.
  - Possibility to save high quality images (high resolution and DPI).
  - Compressed video.
  - Possibility to save graph layout, coloring and current view state into text file.
  - All opened tabs can be saved into single file at once.

2.2 Design Goals

Main goal is flexibility. User interface should be highly flexible, simple and intuitive. User should not feel to be limited in any way. This implies tabbed interface with possibility to view arbitrary number of tabs at a time. All windows must be resizable. Window controls must be able to adapt to various sizes of their parents or to various color schemes and DPI settings of host operating system. Frequently used controls should be easily accessible (e.g. through a toolbar). Dialogs should be non-modal if possible. Modal message boxes should be used with caution. Navigation should be possible with both keyboard and mouse.

2.3 Operating Environment

Program will run on multiple platforms. It is targeted to all three major operating systems – MS Windows, Linux and Mac OS. Program should not need admin privileges to run (with exception of installation). All runtime libraries and dependencies must be distributed along with program. Program will be redistributed as an archive or an installation package.

2.4 Implementation Constraints

Source code should be flexible and maintainable. It must be easy to change or to add features, modes of operation, file formats etc. Special attention should be taken to adaptability of input parser. This probably implies the use of regular expressions. Code itself should be self-explaining, well formatted and easily readable, possibly without distracting comments. Function and variable names should be self-describing. Some subset of Hungarian notation should be considered.

Implementation will be based on Qt framework (http://www.qtsoftware.com/) developed by Qt Software (Nokia). Since Qt is implemented in C++, the language to be used is C++ along with STL. Because Qt framework offers very good data structures and some advanced abstractions, usage of STL should be reduced to minimum (e.g. numeric operations). Qt framework covers some

requirements of this specification – it is multiplatform, window controls are based on layouts, it contains rich graphic subsystems, and finally there is support for both raster and vector image formats, concurrency, events and regular expressions. Video capturing will be based on some open source video library (e.g. FFmpeg, Xvid). Program should be developed under Qt Creator, which is multiplatform IDE specially designed for development of Qt applications. Another possibility is to develop under MS Visual Studio. The usage of version control system is recommended (probably SVN).

2.5 Assumptions and Dependencies

Program will be dependent on third party libraries – namely Qt runtime and FFmpeg video library. Although Qt libraries will probably not be a cause of any trouble, there is some concern about pumping Qt's visual data into FFmpeg codecs. It might appear to be either slow or impossible, so that some workaround will be enforced. Even if it worked well it might not be fast enough for real-time video compression (assuming encoders that are designed for real-time compression). There is also some uncertainty about what the quality of video output will be like. Those potential problems might alter application architecture in some way. During development there will be definitely carried out some benchmarks and tests in order to evaluate how video capturing will exactly work. There are some possibilities:

- Quality will be good and it will be fast, so that the real-time compression would be feasible.
- Quality will be unsatisfactory or the real-time compression will be impossible. In this case, the solution would be to capture video uncompressed and then compress it after the animation is over.
- Combination of above cases. It might be also possible to include more video libraries – namely Xvid.

3 Additional Requirements

3.1 Internationalization

GraphRec 1.0 is supposed to be English only application. However, since Qt provides very good support for localization, every string for GUI should be wrapped in translation function. This approach could be good starting point for possible localization in future.

3.2 Licenses

Both Qt framework and FFmpeg library are licensed under GNU GPL or GNU LGPL. Since these licenses are viral, GraphRec must be licensed under these or less restrictive licenses. GraphRec will be probably licensed either under GPL only or under dual license similarly to Qt or FFmpeg.

3.3 Documentation

Documentation for GraphRec will consist of manual for programmer, manual for user and installation manual (if needed). These documents should be written in a way that it would be possible to publish them as PDF files or HTML files.