CS480 Tower iLLuminati Design Report

Jenny Huang

December 5, 2014

Contents

cover sheet - page 1

Project Title	Tower iLLuminati
Sponsor	Dr. Robert Rinker
Team name	Heyan Huang

Abstract

Execute Summary 1/2 page: ##################paddingOur current software is the v0.3 version writen in C#. And every "window" change needs a click. And the v0.2 was black and white, and since UI CS Curriculum is C++ based, we want to try our best to make a most basic software so that later on every ACM member with interest would be able to add any components that they are interested in and so that there won't be programming language barrier for them. lp It's a old project that repeats each year maybe just for us students to get practised. Our current version of project was pretty much a fully functioning one, and it was writen by one ACM student who graduated already, and it was writen in C# with his own interest.

1 Project Background

- Motivation for the work Our cliend Dr. Rinker is very environment-friendly on campus, and he lead the ACM team for the department, and he enjoys offering music-related fun activities for the compus like tower light show for homecoming events. For tower light show, the current tower animator c#-programming based software plays the .tan light file type separately from the .wav audio file, which is not convenient for him and his ACM parties to use.
- Identify the need Since University of Idaho Computer Science department is mainly c++-based programming, and the c# language does produce language barrier for a certain amount of ACM users, and the software is quite some distance from user-friendly and functionaly complete, there exist the need for potential refactor, reimplementaion, or update on this software.
- Describe the expected benefits Reimplement and update the software in c++ language will promote ACM user attendence on product software updates and also, we, as the senior design team would be able to get good practise and benefit a long way from various aspects like programming technical practise, softare engineering, project design and handling, as well as problem-solving skills.

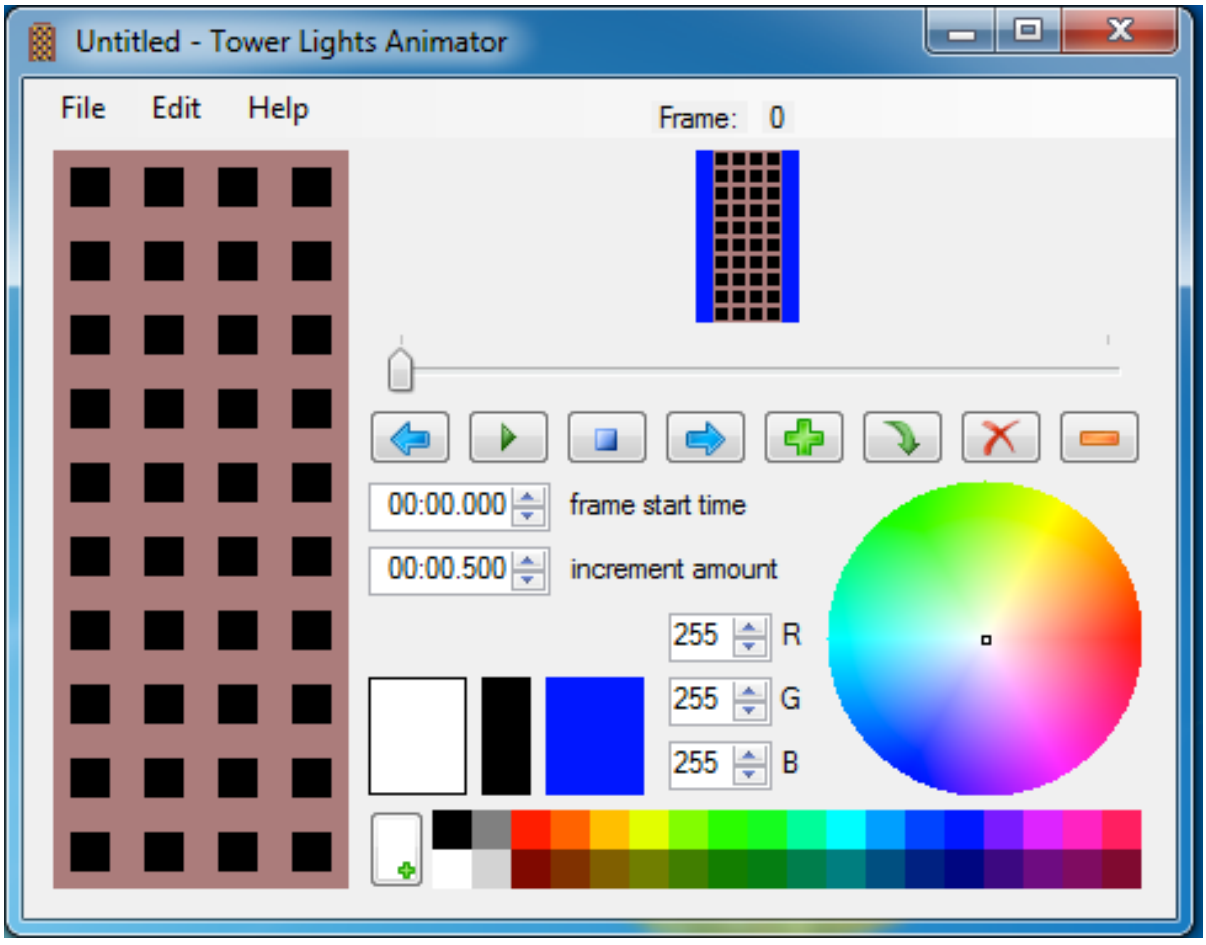

Figure 1: Base software Interface

2 Problem Definition

2.1 Goals

This design document describes the system architecture and design of a software for tower light animation synchronized with audio music for the University of Idaho, Senior Design course instructed by Professor Bolden. It is intended for the development team.

2.2 Scope

This application must satisfy these requirements as defined by the customer, Dr. Rinker.

- Must link the light animator .tan file with corresponding audio file
- Must redesign and reimplement the software in c++ and Qt Creator library for Graphical User Interface
- Must be a non-trival tower light animator software
- Should make the software user-friendly and easy to use
  - When design each frame, the user promote less clicks for picking color for each grid;
  - Should have commands for automated movements
  - Should design pre-generated shapes and patterns
  - Could have Pre-generated animations
  - The software could import from source files satisfies **certain** requirements
  - Since audio file will be provided already, the software promotes user-friendly feature, the design should include a parser to parse audio file for tower light frame intervals.
  - The software should be real-time, which depromotes slow response, and promote well-design for source program execution efficiency.

The scope of this project will not go beyond the defined requirements. was chosen for the card game. Tablet devices running Android 3.0 are the target device and environment. The only feature,not specified by the customer, to be included in the scope of work is the addition of a very simple menu screen with a single button with the label New Game. There will be no additional features added such as multiplayer or network capabilities.

2.3 Deliverables

- A software which is well-designed and fully satisfies the client's requirements, and the software should also be user-friendly and executes smart and efficiently.
- A Software Requirement Specification document and a user-manual which is easy to read and understand, as well as fully functionaly to explain all the necessary manipulations.

2.4 Constraints

As a senior team, we design well-design and implementation for the software, but there are critical issues that we need take into consideration for, for example, 80% of the teammates are currently in the university famous <Compiler Design> class, and they can barelly spend many time on this project for the fall semester. As a result, when we design the project, we keep the client's requirements in mind, and try to design the software so that the software meets the minimum requirement to be a software while satisfying the clearly-specified scope, while at the same time that the software is very extendable to promote more features, efficiency and user-experience.

2.5 Reference Material

-
-

3 Project Plan - 1/2 page

3.1 Tasks and schedule

3.2 Team responsibilities

- One team member created a image GUI design for our Snapshot #1, which is included in Figure 2;

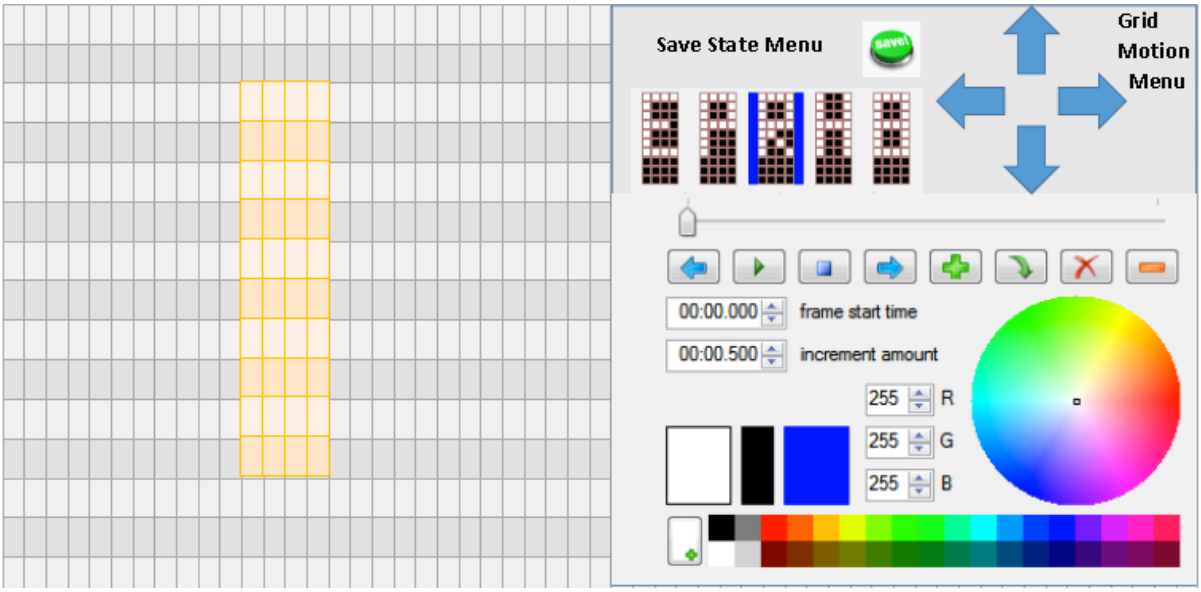

Figure 2: imaged-based GUI for Snapshot #1

- One team member created a state diagram, and configured Qt Creator environment for the team;
- One team member finished a wikipage;
- One team member gave a most basic C# code of base version software during one team member with Professor, and gave the presentation for design review;
- The team has one effective team meeting for menu bar design on Nov 16th;
- I was asked to type the team contract, and was the team member tried to design and implement a most basic clickable GUI as the project implementation starting point;
- I am right now asked to write the Design Report proposal individually for this course;

4 Concepts Considered - 3 pages

4.1 Original ideas + those derived from other sources

- Original ideas were mainly derived from our current base version of c#-programming based software;
- Implementation language c++ was selected because our university is applied the c++-based data structure and programming teaching style; And Qt Creator was selected and required by course instructor Professor Bolden;

Table 1: System Functions conpressed into Menubar

File	Edit	Play	Select	Mtool	Help
Open	Copy (C-c)	play from start	Row (SPC-mse)	Insert Shapes	Documentation
New	Cut (C-x)	play from current	Col (Sft-SPC-mse)	Define Pattern*	About
Save	Paste (C-v)	Pause	All (C-a)	Color Gradient*	
Save as	Insert After/Before	Stop	Invert Slt (C-i)		
Export	Clear	Move Forward	Slt Shift-C		
Close	Delete	Move Backward	Slt C-mouse		
Exit	Undo (C-z)				
	Redo (C-y)				

4.2 Quantitative Data or Measurements

- According to requirement that the software needs to be user-friendly, and must conduct less clicks when picking color and design color frames;
- According to requirement that the software needs to be smart, so that it can NOT be slow for source program execution.
- References: Literature/Catalog Review

5 Concept Selection - 1 page

5.1 How did you arrive at your final selection?

- Scientific naming conventions;
- Concepts comparson among team members;

5.2 Include morphological charts, decision matrices

5.3 GUI Layout Selection

We have been proposed two GUI designs, snapshotted when we discussed about them, which are listed as followed.

6 System Architecture - 2 pages

6.1 Architecture Design

The architectural design was developed by following the software requirements according to Software Requirement Specification (SRS). Here, the Tower Light Animator will be described in terms of basic modules which will give the reader a first understanding of the functionalities of the software.

The Tower Light Animator can be described in 5 different modules:

- GUI: It is in charge of giving access to our clients and any software user to interact with the program;
- Software Manager: This module is the managerment center of the several modules. It decides which other module is necessary to call to get source feed or to execute commands. Also, it is in charged of controlling the software module-to-module controll flow, supervising the commands rules and handling excepts and errors as well.
- Event Manager: This module is responsible for handling mouse click enents and keyboard events. It is responsible for receiving and handling these two device enents signals, sorting, filtering, identifying the source signals, and links, triggers the corresponding response to command center;

- File Manager: It is always responsible for writing the software generated product contents into output specific types of files. And it is also responsible for importing and parsing the source input file and stores the source information into efficiency-concentrated data structures, and parse data into visible graphics as well if necessary. This is the valued I added to the project, and will be fully conducted when the project is on track and in good shape.
- Command Center: This module is the center for specifying and defining the majority kinds of GUI-related activity body, like what should the software do if the use clicks a specific button with a mouse click on the left, the corresponding response the software would conduct will be defined in one command or more. And this is the center for all these commands.

6.2 Decomposition Description

In this section, each module inside the Tower Light Animator software described previously is broken down into smaller specific tasks which will allow the reader to identify what is the expected behavior of each module.

- GUI
  - User interaction: this component is in charge of capturing the decisions made by the our clients and software user (buttons, sliders, etc.).
  - Software Feedback: this component shows all the alerts and current status of the in-progress tower light frame-making that are sent by the Software Manager module in order to inform the clients what is happening at every moment of the progress.
  - Entity Representation: this component contains the rest of the graphic elements that are part of the Tower Light Animator and the user can apply of (i.e. color sources, including predefined colors, picking from color wheel etc.).
  - It is in charge of giving access to our clients and any software user to interact with the program; The GUI is a "MUST" for the software.
- Software Manager
  - Control Flow: it controls the status of the software (i.e. player turn and time taken by the computer players)
  - This module is the managerment center of the several modules. It decides which other module is necessary to call to get source feed or to execute commands. Also, it is in charged of controlling the software module-to-module controll flow, supervising the commands rules and handling excepts and errors as well.
  - This module functions and promotes software efficiency.
- Event Manager
  - This module is responsible for handling mouse click enents and keyboard events. It is responsible for receiving and handling these two device enents signals, sorting, filtering, identifying the source signals, and links, triggers the corresponding response to command center;
  - This module functions for reacting to user-input events and promotes user-friendly features.
- Files Manager
  - It is always responsible for writing the software generated product contents into output specific types of files. And it is also responsible for importing and parsing the source input file and stores the source information into efficiency-concentrated data structures, and parse data into visible graphics as well if necessary. This is the valued I added to the project, and will be fully conducted when the project is on track and in good shape.
  - The module is a "MUST" for user-software interaction production – write into file, and also promotes user-friendly and software-smart features.
- Command Center
  - This module is the center for specifying and defining the majority kinds of GUI-related activity body, like what should the software do if the use clicks a specific button with a mouse click on the left, the corresponding response the software would conduct will be defined in one command or more. And this is the center for all these commands.
  - This module is the most basic execution element, and software can not function without this module.

Provide quantitative results from tests or analysis

7 Data Design

7.1 Data Description

The system architecture is implemented by using c++ and Qt Creator library specific classes. Accordingly, the system entities are stored, processed and organized in these classes via different data structures, including 2D array of QWidget pointers, array list, containers and maps.

7.2 Data Dictionary

8 Component Design

8.1 GUI

- User interaction
- QMenu, QAction
- QHBoxLayout, QVBoxLayout, QGridLayout
- QPushButton, MyPushButton
- QSpinBox, MyDoubleSpinBox
- QSlider
- QLabel
- ColorWheel
- Qt Quick for JS dynamic user-custom colors

8.2 Software Manager

- QThread
- QRunnable
- QtConcurrent
- QTimerEvent
- QChildEvent

8.3 Event Manager

- Q_{OBJECT} Signals & Slots
- EventHandler
- EventFilter
- QKeyEvent
- QKeySequence
- QMouseEvent
- QWidget::paintEvent

8.4 Files Manager

- QFile
- QTempraryFile
- QFileDevice
- QBuffer
- QProcess

8.5 Command Center

- These are mainly functions.

9 Human Interface Design

10 Future Work - 1/2 page

Recommendations for continued work

10.1 Features that didn't find their way into the current design

- Threads implementation
- Software IO manager and manipulations
- Audio file parse to generate consistance for timestamp increasement amount, a basic intuitive layout is listed as below presented in Figure 6, which has a timestamp double spinbox to show the detailed time and parsed wave drawing picture:

10.2 Estimate size and duration of the required effort

10.3 the work TODO

- for Qt Signals and Slots So far I got three ideas how to implement the connection between signals and slots;
  - Inheritance to rewrite signals and redefine slots;
  - Use eventFilter to find signal sender(), and conduct the connection by scanning the widgets;
  - Most basic method that I implemented already, by connecting one by one, and repeat; The very first method I tried a little bit, but failed to get my pushbutton painted. But I can spend some time to debug this one, and I could also try the second method as well. There must be some way to facillate the linking process.
- Implement functionality to GUI
- Implement multiple pixel movement
- Figure out modifications to .tan file layout
- Next Semester
  - Implement more advanced animations/movements
  - Add more GUI features

unreolved issues and plan for attacking these

11 Appendices

The supporting documents to long or detailed for main body includes the following several sections.

11.1 Calculations & drawings

11.2 Large tables & figures

Table 2: Commands and Fast keys

MainMenu	Commands	Fast Keyset
File	File	C-f
	New	
	Open	C-o
	Save	
	Save as	
	Export	
	Close	
	Exit	
Edit	Edit	C-e
	Cut	C-x
	Copy	C-c
	Paste	C-v
	Insert After/Before	
	Delete	
	Clear Frame	
	Undo	C-z
	Redo	C-y
Play	Play	C-p
	From Beginning	C-b
	From Current	C-n (now)
	Pause	
	Stop	
	Move Forward	
	Move Backward	
	Preview Mode	C-r (review)
Select	Select	C-s
	Row	SPC-mse
	Col	Sft-SPC-mse
	All	C-a
	Invert	C-i
	Deselect	C-d
Mtool	Mtool	C-m (movie)
	Insert Shapes	
	Define Pattern*	
	Color Gradient*	
Help	Help	C-h
	Documentation	
	About	

Visualization (sketches, drawings, diagrams)

- Classes and Prototypes Modeling and/or Experimentation

11.3 Vendor data sheets

11.4 Computer Programs

- Coding

– Based on Design
– Testing

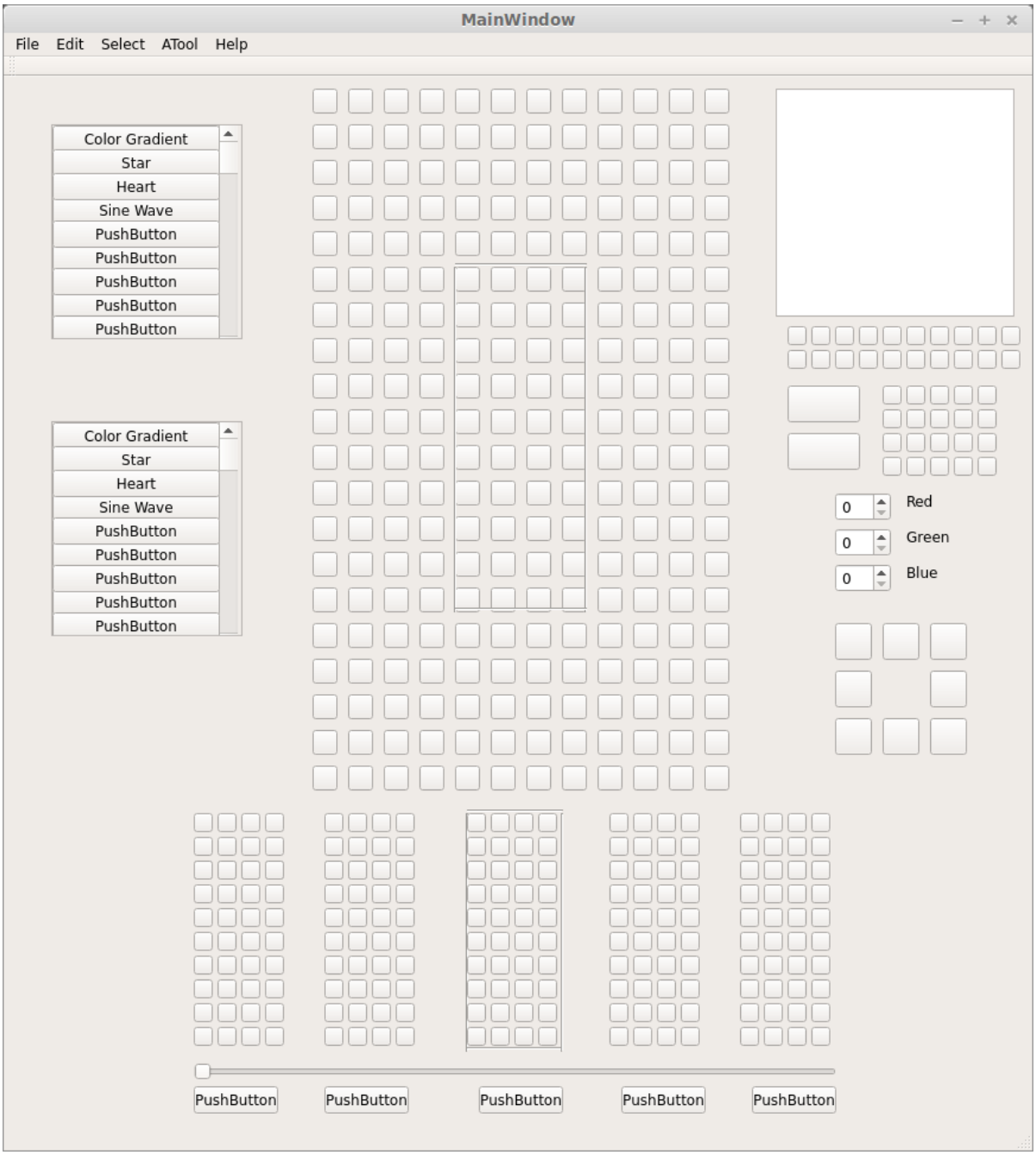

Figure 3: team manager proposed new design

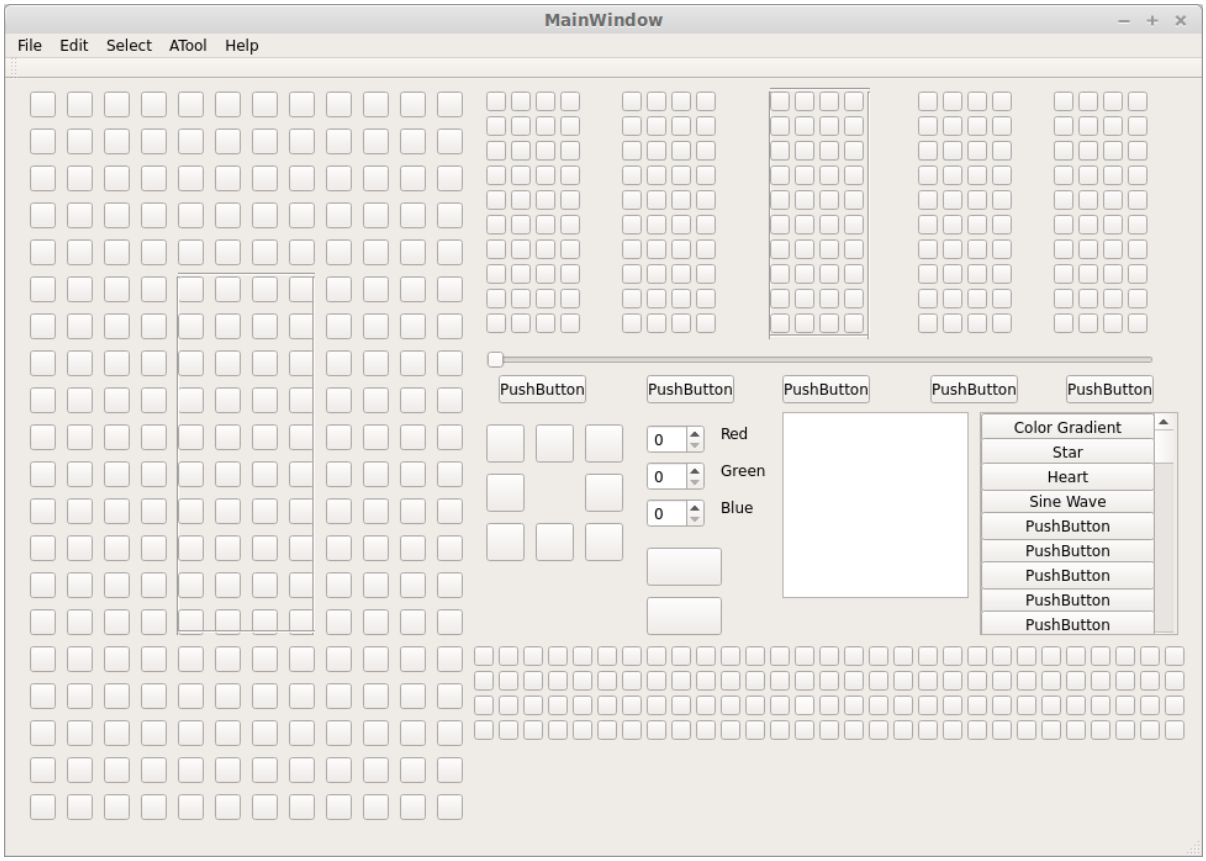

Figure 4: Original traditional design

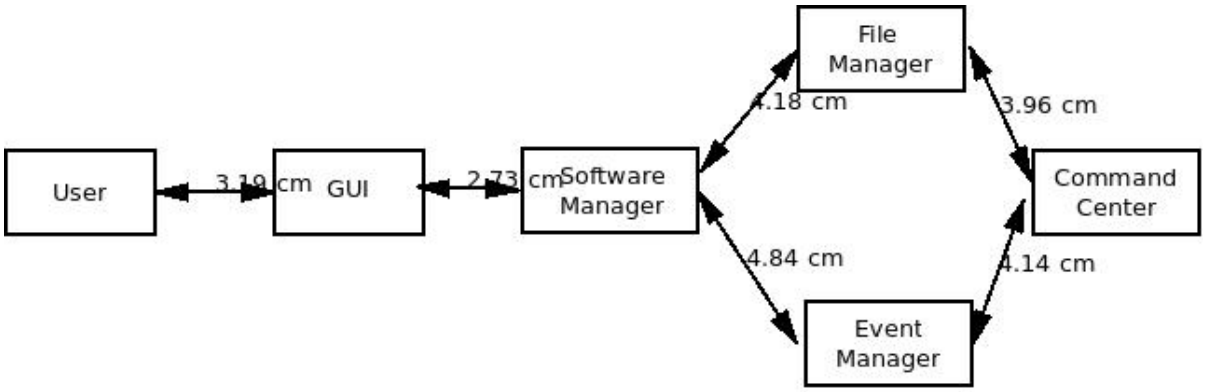

Figure 5: System Architecture High Level Model

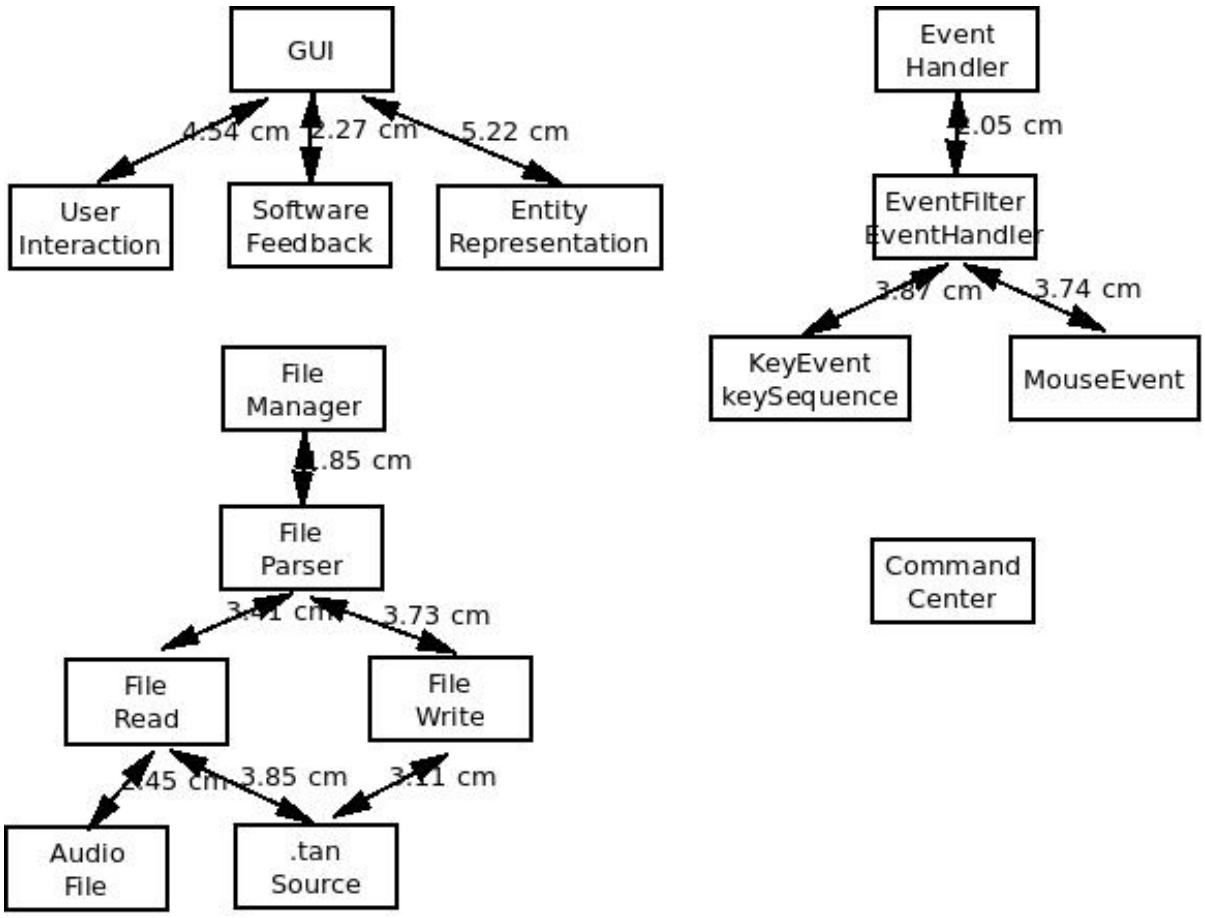

Figure 6: System Architecture Block Diagram for Tower Light Animator

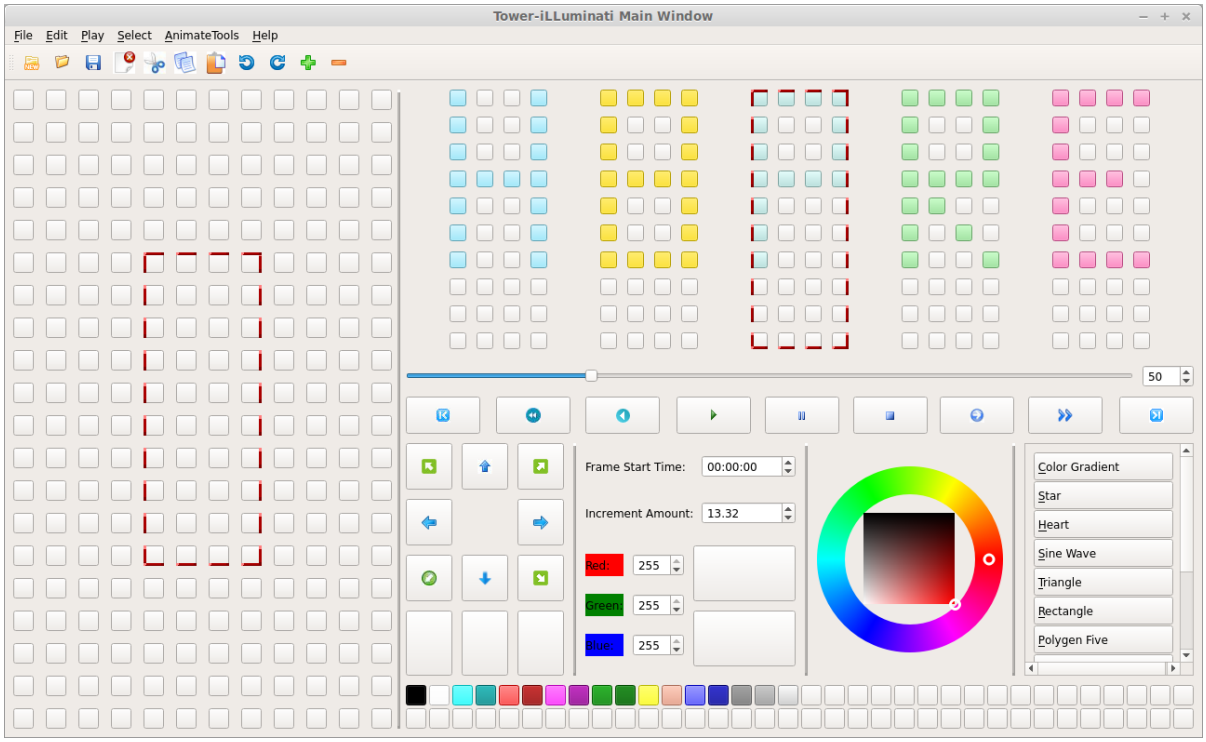

Figure 7: Main Windows

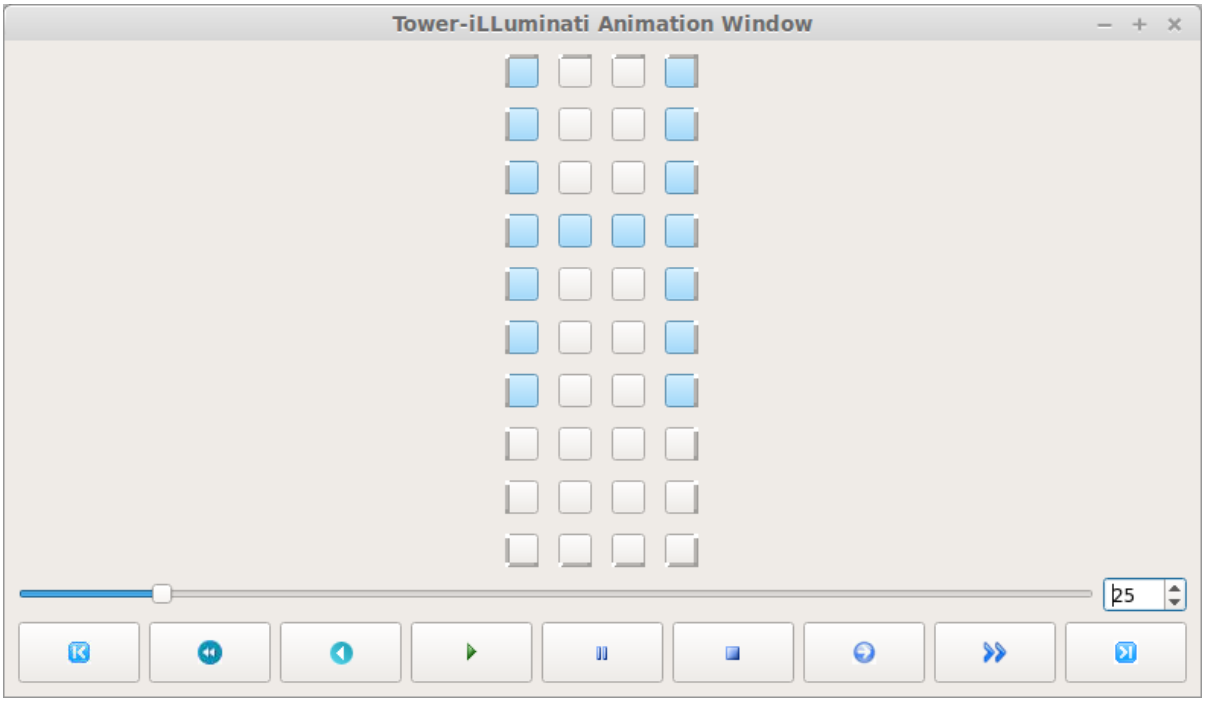

Figure 8: Pop-up Windows

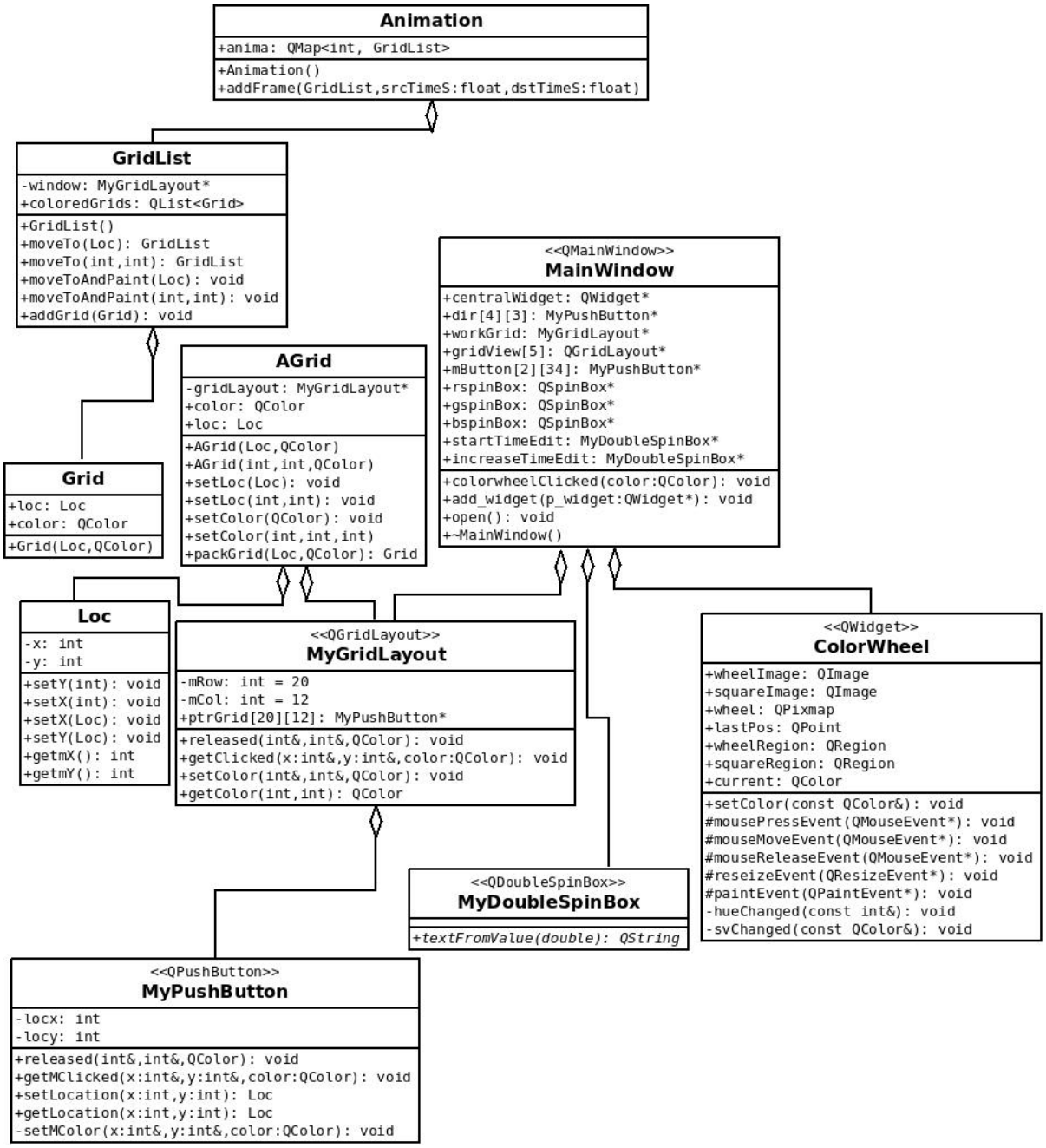

Figure 9: partially implemented detailed class diagram

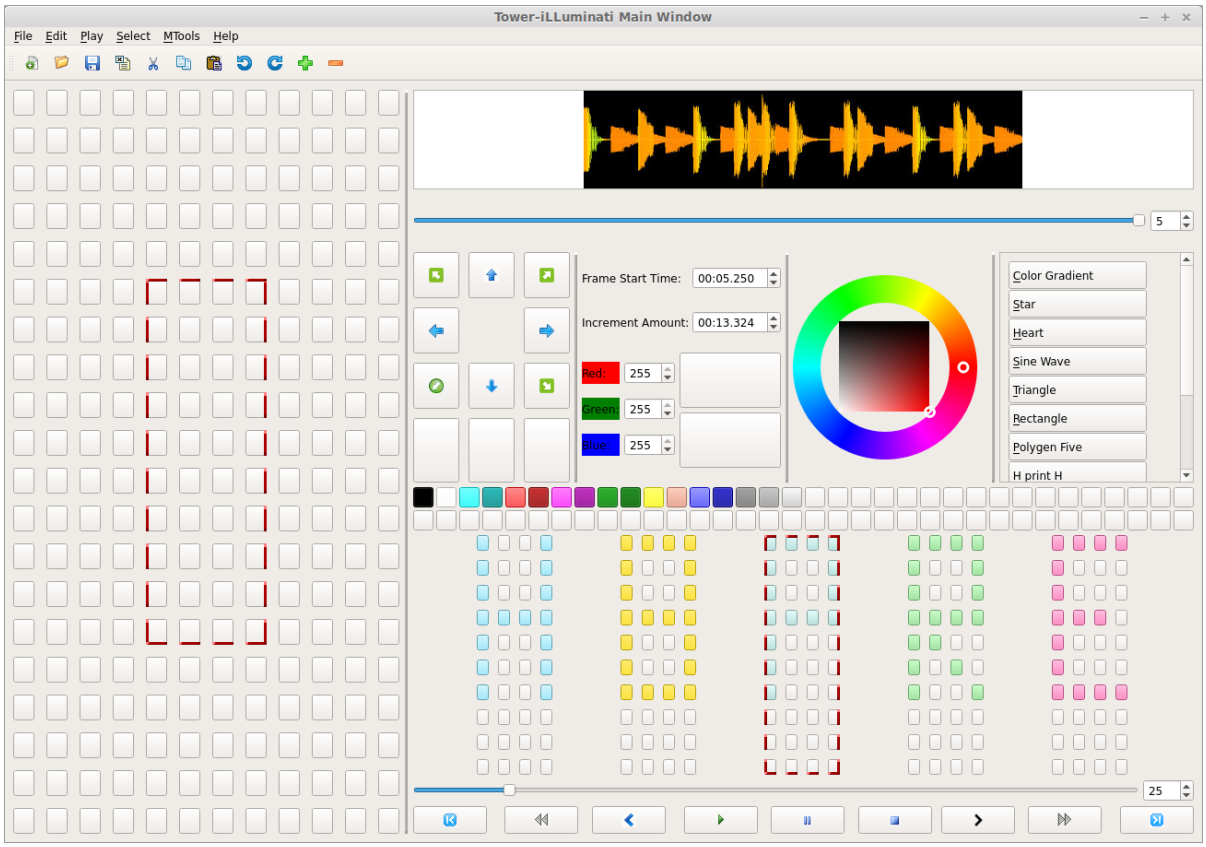

Figure 10: GUI with audio file parsed for timestamps

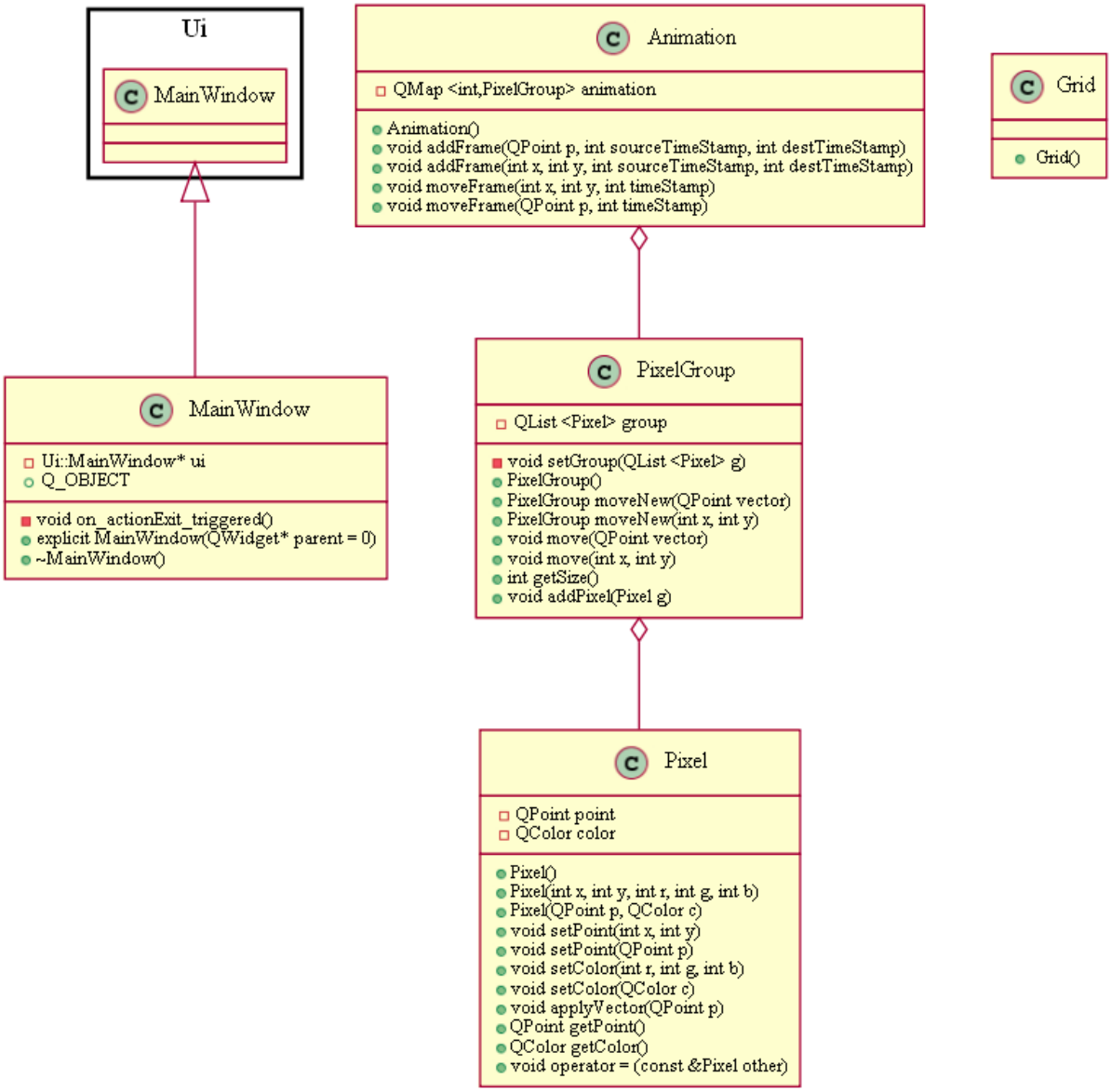

Figure 11: System Architecture Class Diagram

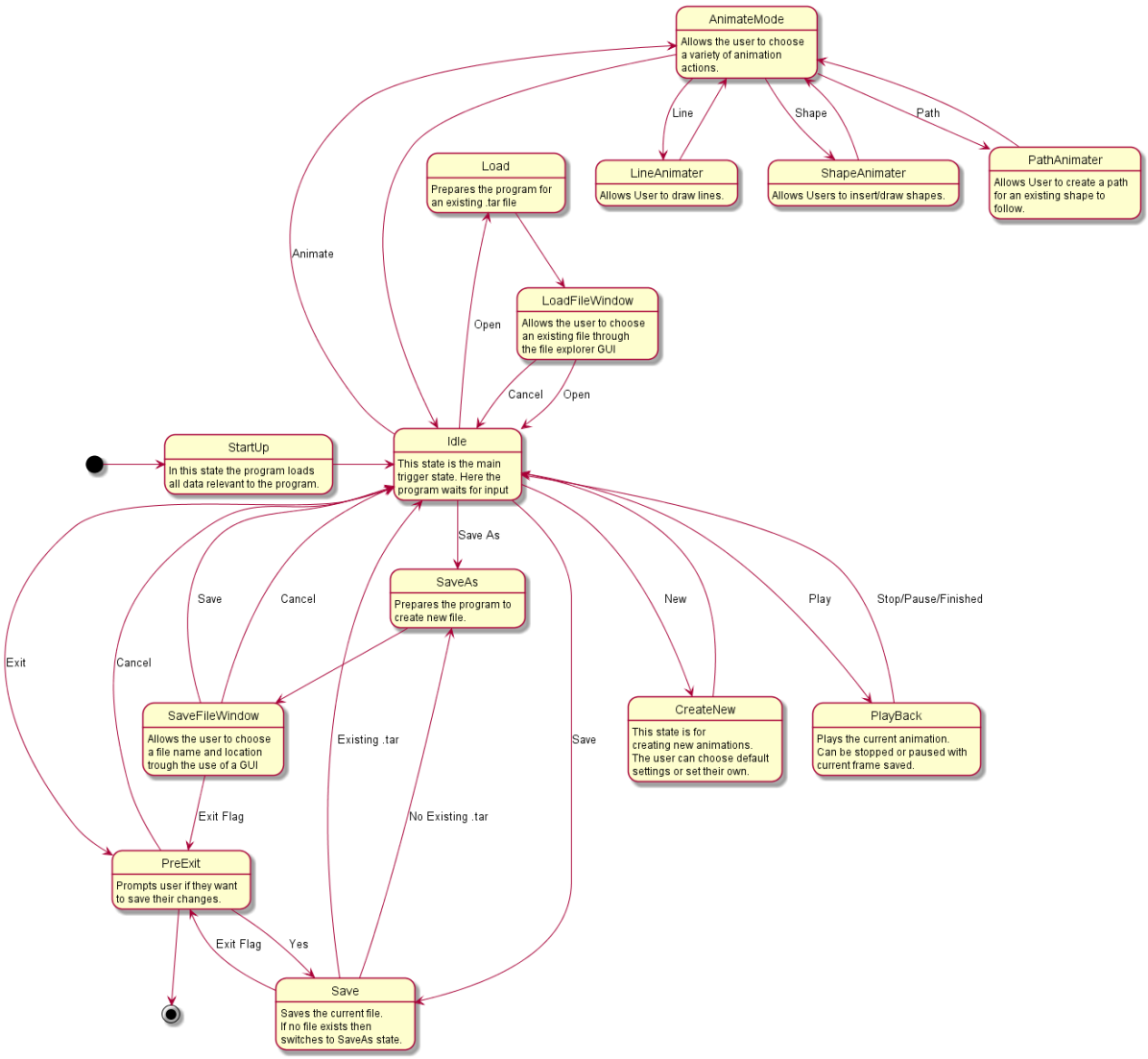

Figure 12: Software State Diagram