CS480 Tower iLLuminati Design Report

Jenny Huang

December 4, 2014

Contents

cover sheet - page 1

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

Abstract

Execute Summary 1/2 page

Our 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.

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.

2 Problem Definition - 1 page

2.1 Goals and deliverables

- Link .tan file with corresponding audio file
- Rewrite in C++
- Make light shows easier
  - Less clicks
  - Automated movement
  - Pre-generated shapes
  - Pre-generated animations
  - import from other files

2.2 Specifications & constraints

3 Project Plan - 1/2 page

3.1 Tasks and schedule

3.2 Team responsibilities

4 Concepts considered - 3 pages

4.1 Original ideas + those derived from other sources

4.2 Quantitative data or measurements

References: Literature/Catalog Review

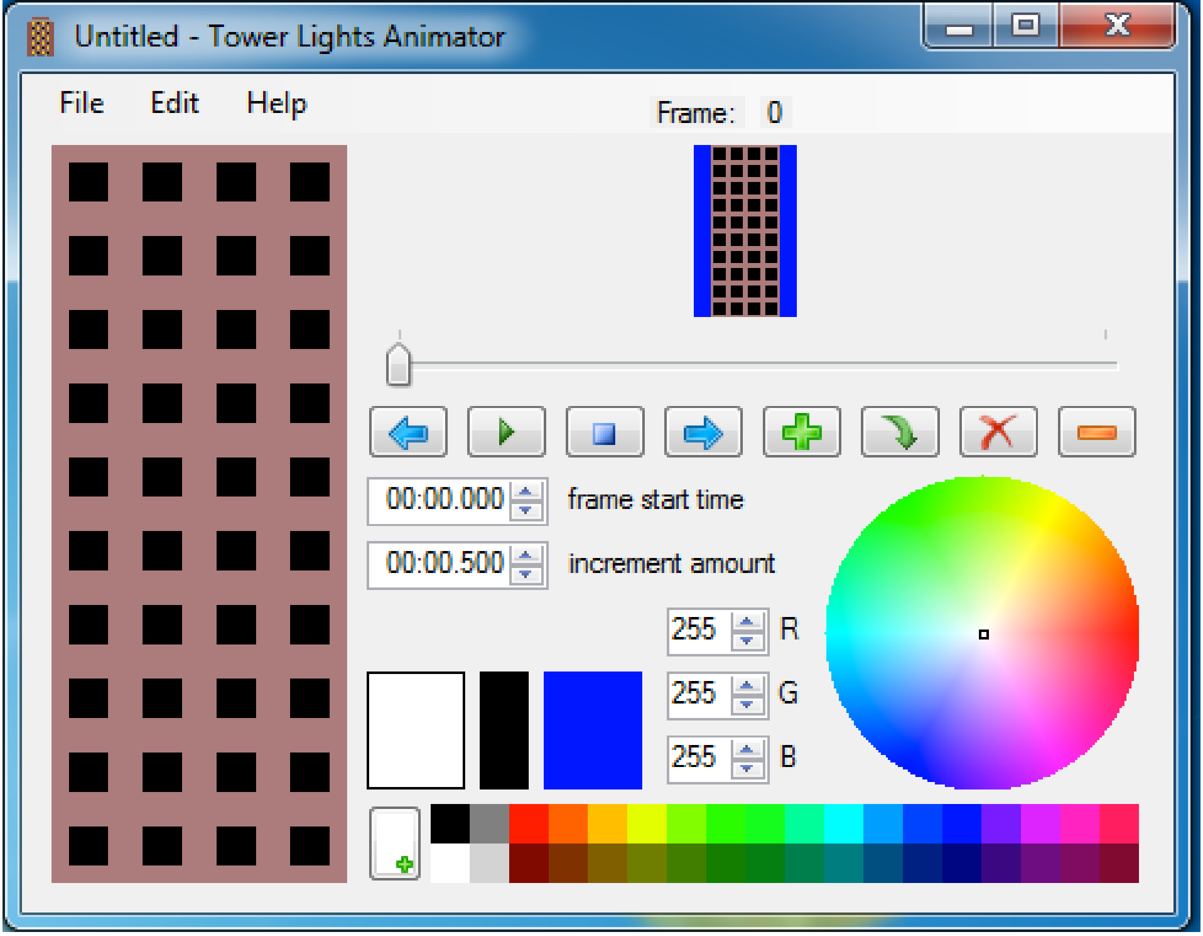


Figure 1: Base software Interface

5 concept Selection - 1 page

5.1 GUI Layout Selection

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

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)				

5.2 How did you arrive at your final selection?

5.3 Include morphological charts, decision matrices

6 System Architecture - 2 pages

6.1 Describe the conceptual design ’ justify continued development

6.2 Describe the components and how they are integrated

6.2.1 Highlight novel features ’ your value added

6.2.2 How does each major component satisfy a requirement

6.3 Provide quantitative results from tests or analysis

6.4 Architecture Design

The software can be described

7 Human Interface Design

8 Future Work - 1/2 page

8.1 Recommendations for continued work

8.1.1 Features that didn’t find their way into the current design

8.1.2 Estimate size and duration of the required effort

- 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

9 Appendices

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

9.1 Calculations & drawings

9.2 Large tables & figures

Visualization (sketches, drawings, diagrams)

- Classes and Prototypes Modeling and/or Experimentation

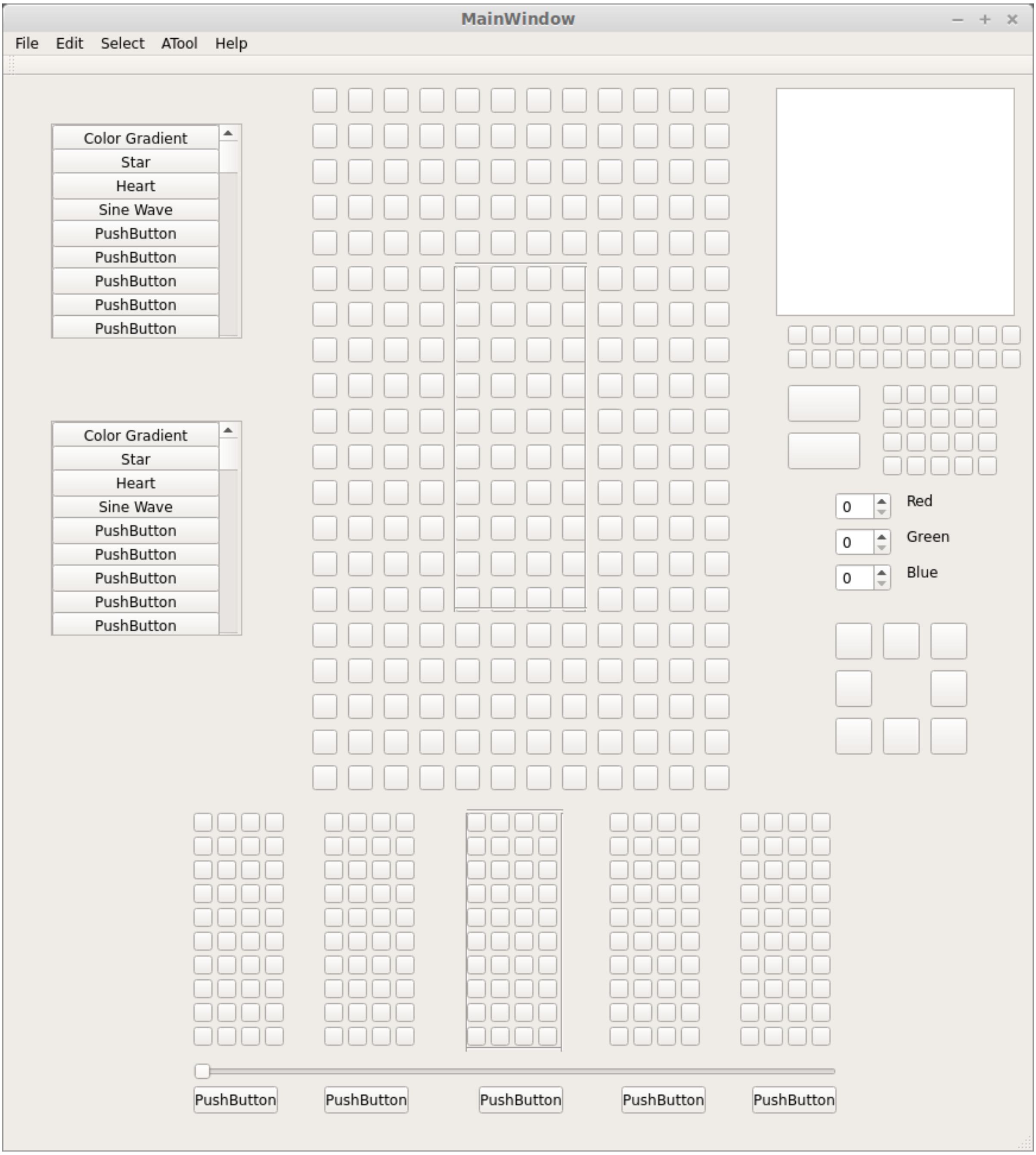
9.3 Vendor data sheets

9.4 Computer Programs

- Coding
  - Based on Design
  - Testing

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	

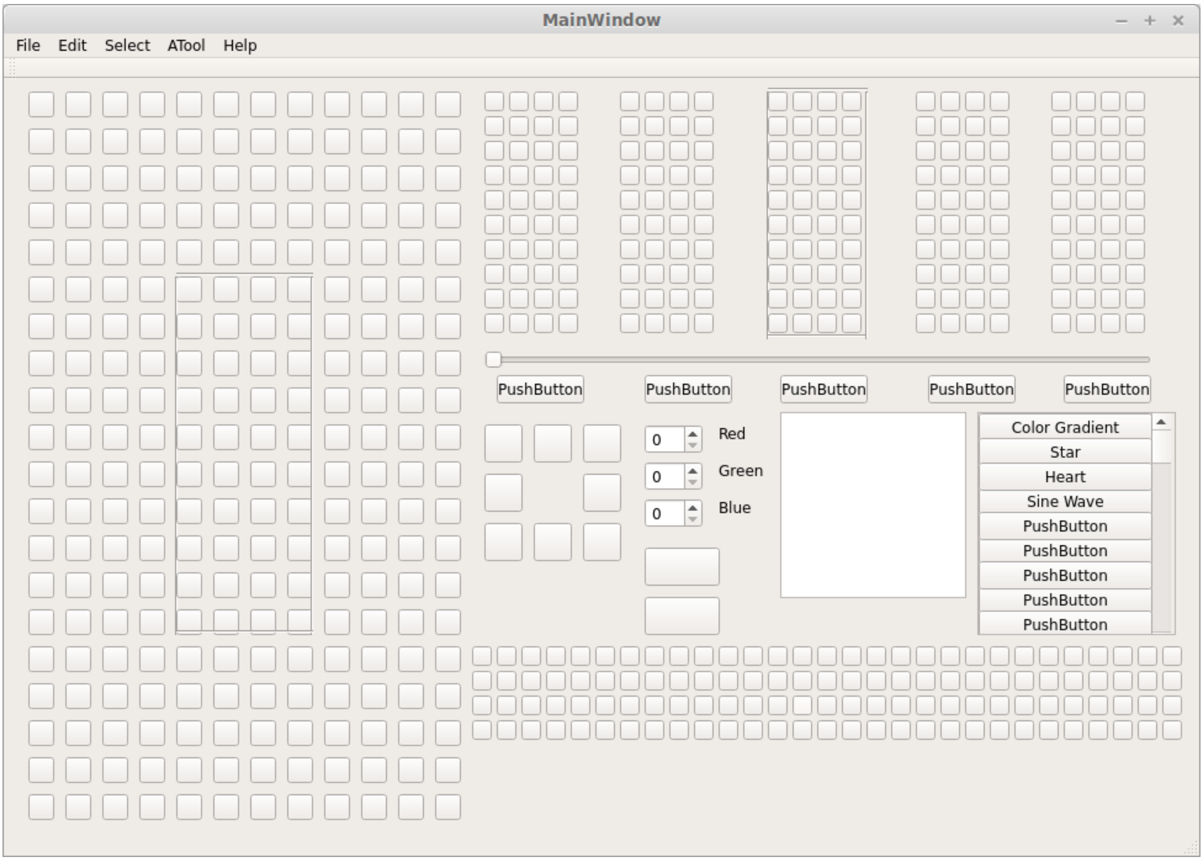


0 Red

0 Green

0 Blue

Figure 2: team manager proposed new design

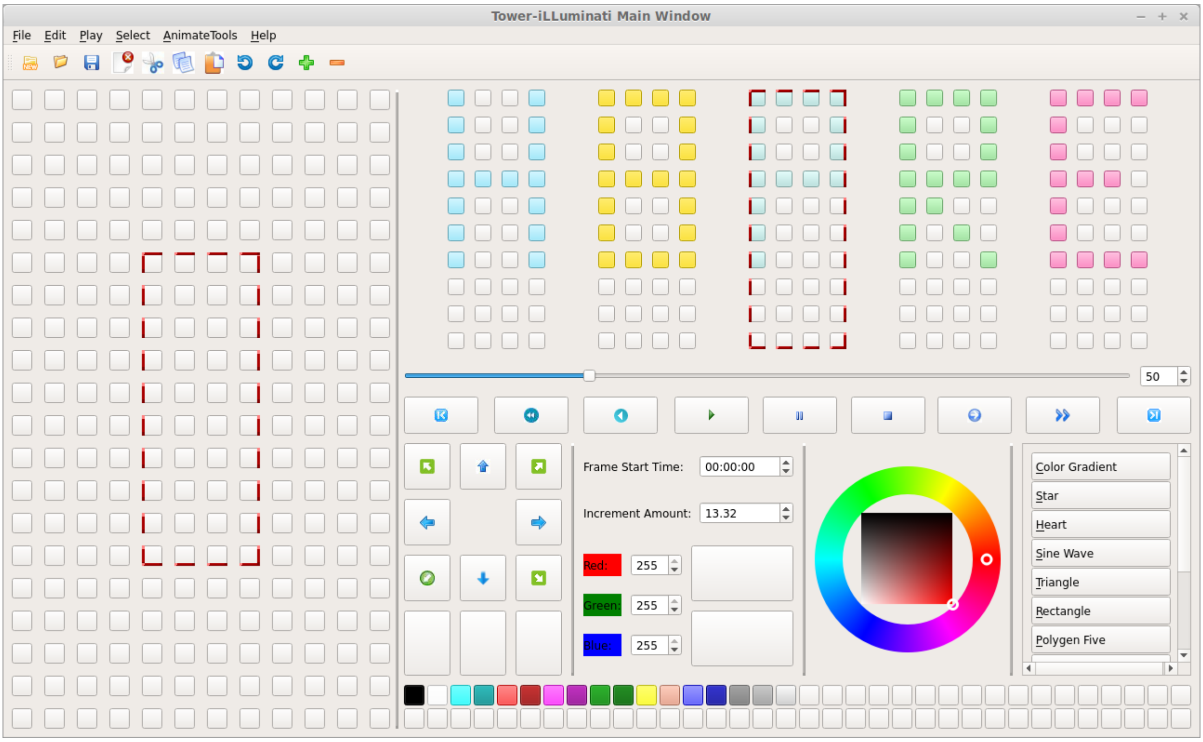


0 Red

0 Green

0 Blue

Figure 3: Original traditional design



50

Frame Start Time: 00:00:00

Increment Amount: 13.32

Red: 255

Green: 255

Blue: 255

Figure 4: Main Windows

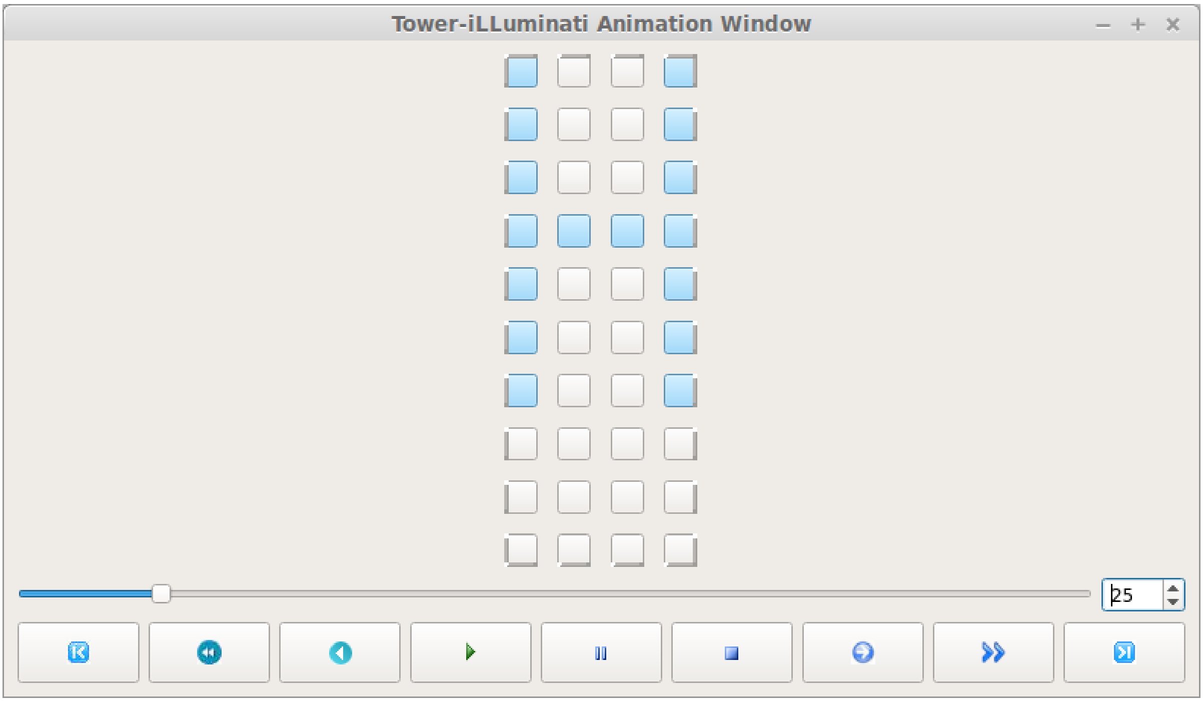


Figure 5: Pop-up Windows

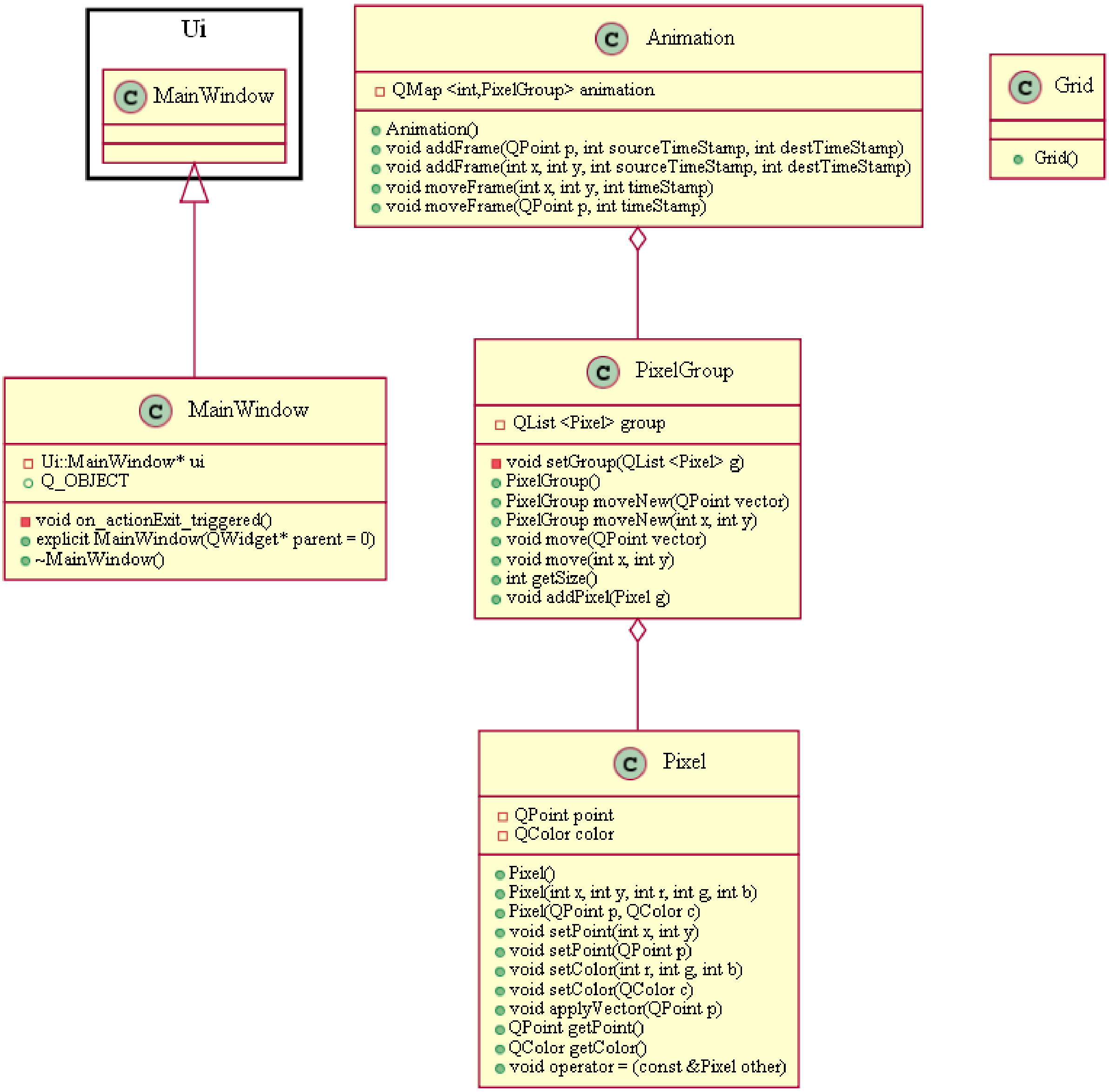

Ui
MainWindow
Animation
QMap <int,PixelGroup> animation
Animation()
void addFrame(QPoint p, int sourceTimeStamp, int destTimeStamp)
void addFrame(int x, int y, int sourceTimeStamp, int destTimeStamp)
void moveFrame(int x, int y, int timeStamp)
void moveFrame(QPoint p, int timeStamp)
Grid
Grid()
MainWindow
Ui::MainWindow* ui
Q_OBJECT
void on_actionExit_triggered()
explicit MainWindow(QWidget* parent = 0)
~MainWindow()
PixelGroup
QList <Pixel> group
void setGroup(QList <Pixel> g)
PixelGroup()
PixelGroup moveNew(QPoint vector)
PixelGroup moveNew(int x, int y)
void move(QPoint vector)
void move(int x, int y)
int getSize()
void addPixel(Pixel g)
Pixel
QPoint point
QColor color
Pixel()
Pixel(int x, int y, int r, int g, int b)
Pixel(QPoint p, QColor c)
void setPoint(int x, int y)
void setPoint(QPoint p)
void setColor(int r, int g, int b)
void setColor(QColor c)
void applyVector(QPoint p)
QPoint getPoint()
QColor getColor()
void operator = (const &Pixel other)

Figure 6: System Architecture Class Diagram

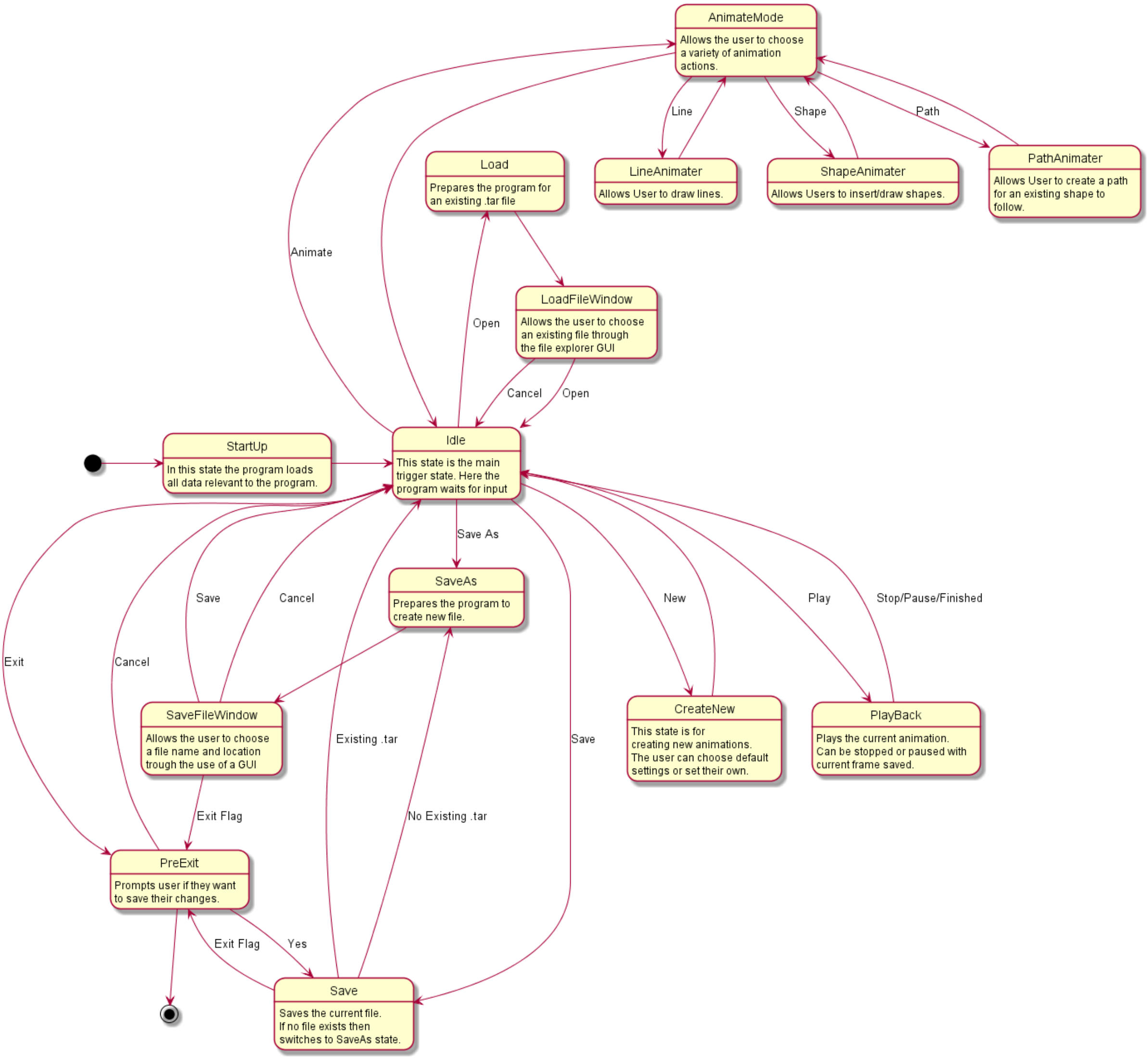

AnimateMode
Allows the user to choose a variety of animation actions.
Line
Shape
Path
Load
Prepares the program for an existing .tar file
LineAnimater
Allows User to draw lines.
ShapeAnimater
Allows Users to insert/draw shapes.
PathAnimater
Allows User to create a path for an existing shape to follow.
Animate
Open
LoadFileWindow
Allows the user to choose an existing file through the file explorer GUI
Cancel
Open
StartUp
In this state the program loads all data relevant to the program.
Idle
This state is the main trigger state. Here the program waits for input
Save As
SaveAs
Prepares the program to create new file.
Save
Cancel
New
Play
Stop/Pause/Finished
Exit
Cancel
SaveFileWindow
Allows the user to choose a file name and location trough the use of a GUI
Existing .tar
Save
CreateNew
This state is for creating new animations. The user can choose default settings or set their own.
PlayBack
Plays the current animation. Can be stopped or paused with current frame saved.
Exit Flag
No Existing .tar
PreExit
Prompts user if they want to save their changes.
Exit Flag
Yes
Save
Saves the current file. If no file exists then switches to SaveAs state.

Figure 7: Software State Diagram