SOFTWARE DESIGN SPECIFICATION

for

Tower Animator

Version 1.0

Prepared by:
Dustin Fox, Lydia Engerbretson, Hannah Pearson, Brandon Brewster, Colton Hotchkiss, Andrew Avery, Alex Wezensky

University of Idaho - CS 383

November 11, 2016

Contents

Development Time Line

Week	Date	Goal
1	Tuesday, 11/1	Complete Design Specification
1	Thursday, 11/3	Design Class Diagrams and Functions that will be needed. Create Scrum Board and choose some tasks to work on.
2	Tuesday, 11/8	Initial scrum tasks completed. Discuss progress. Assign/-choose more tasks from the scrum board to complete prototype 1.
3	Wednesday, 11/16	Prototype 1 complete. Basic working animator; top priority functionality complete. Test, and see how to improve. More scrum board tasks to work on over break.
4	Wednesday, 11/30	Prototype 2 complete. Final prototype everything should be functional. Figure out what needs to be done to polish final product. Bug fixes, etc.
5	Wednesday, 12/7	Final Product complete.

1 Introduction

1.1 Purpose

This Product Design Specification describes the design, requirements, use cases, and development time line for a CS 383 final project: The Tower Animator.

1.2 Intended Audience and Reading Suggestions

This is intended for the reference of our development team, as well as our professor to understand our plan for this project.

1.3 Project Scope

This project is to create an intuitive animation program that is easily accessible to any student who would like to create an animation for the University of Idaho Tower Lights Show. The main goal for this program is to improve on the previous versions of the animation software and make it easier to use which would in turn result in more animations for the show.

2 Overall Description

2.1 Product Perspective

This program is supposed to replace the current Tower Animation software and make it more user friendly. It will create animation videos that are compatible with the current Tower Player software.

2.2 Product Functions

The following is a list of all of the things that the Tower Animator should be able to accomplish, organized from highest to lowest priority. These functions are elaborated in more detail in section 4.

- Save an animation video in .tan format that is currently in use.
- Create/draw objects that can be moved and animated with key frames
- Create and manage individual key frames
- Have a list of all objects in use with detailed information about their location and color that can be changed manually.
- Have an intuitive color selector for the colors of the animation objects
- Save work in progress video in a .animator format

2.3 User Classes and Characteristics

There are three user classes that we anticipate will use this product and we need to cater to them accordingly.

2.3.1 Novice Animator - Priority 1

Students, staff or majority of people who would like to create animations for the Tower Light Show. The main goal of this project is to make this program as accessible to as many people as possible, so we need to cater to users that do not have to have any extensive technical knowledge besides the basics of using a computer.

2.3.2 Expert Animator - Priority 2

In addition to making the program easy to use for someone who is new to animation software, our program should follow common animation techniques to cater to people who have experience animating using other software packages.

2.3.3 Programmer - Priority 3

Dr. Rinker mentioned that he wants this program to be written in C++ so that he can easily make additions to it if he needs or wants to. So with that being said, the project should not only be written in C++ but it should also be well documented so that it is easy for him to quickly understand the code base if he needs to make changes.

2.4 Operating Environment

The target operating environment is Windows Desktop. Our Program should support Windows 10, 8, 7 and possibly previous editions of windows. The possibility of supporting Mac or Linux would be a possibility with Qt but not within the scope of this project.

2.5 Design and Implementation Constraints

The main limitation of this project is time. We have a total of 5 weeks to complete this project from start to finish. Besides that the project must be written in C++ using the QT development platform and must output a .tan file that correctly plays in the Tower player program.

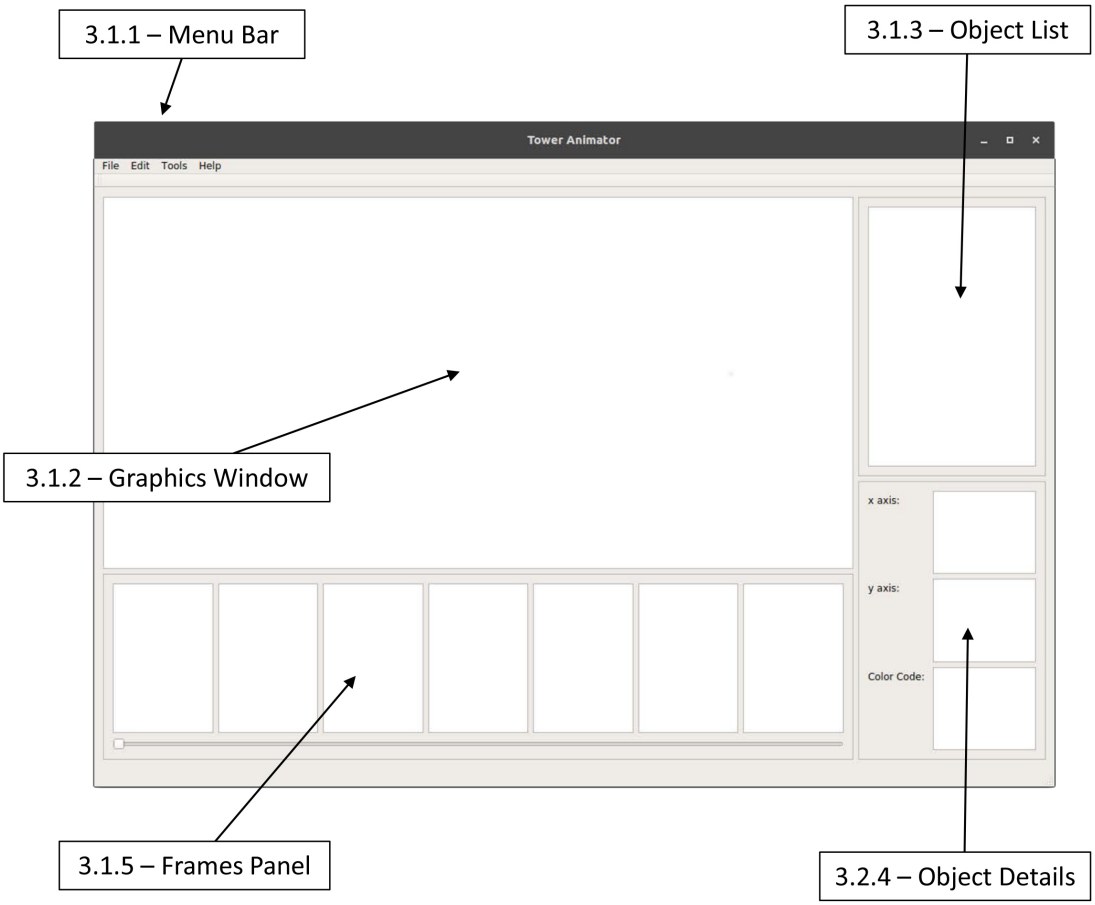
2.6 User Documentation

For this project we will include a User manual that outlines how to make a simple animation with screen shots to help someone get started with the animation process, as well as an installation guide that will help the user install the program and it's dependencies successfully.

3 External Interface Requirements

3.1 User Interface Design

The user interface will be designed to be as intuitive as possible. to include all of the features listed in section 2.2. Here is a prototype of the design labeled and described in the sections that follow.



3.1.1 Menu Bar

The tool bar across the top will be similar to other windows programs containing buttons such as File, Edit, Tools, and Help. Each of these buttons will have their own drop down list of options.

3.1.2 Graphics window

The main graphics window will be a giant grid with a box in the center that shows the tower itself. The graphics window will also contain all of the animation objects in use during the animation. They will be able to be moved around throughout the graphics window for animation.

3.1.3 Object List

The object list will contain a complete list of all of the objects currently in use on the screen from top to bottom in order of priority. Similar to how layers work in Photoshop.

3.1.4 Object Details

Once you select an object, all of that object's information will populate the Object Details panel so that you can change things like color or location.

3.1.5 Frames Panel

The frames panel along the bottom will contain all of the current frames as well as a scroll bar along the bottom that lets you scrub through your animation. The current active frame will populate the center box which will have a key frame button above it.

4 System Features

4.1 Tower Animator File Output

This feature is a .tan2 file output from the Tower Animator.

4.1.1 Description and Priority

When the user interacts with the Tower Animator's File dropdown, there are several options: New, Open, Open Audio File, Save, Save As, Export, Close, and Exit. This is a high priority feature since there needs to be a .tan2 file for the Tower of Lights Player to read.

4.1.2 Stimulus/Response Sequences

- When change is made and File to "New" is selected: a popup appears with "Do you want to save your current project" with options "Yes", "No", and "Cancel"
  - When "Cancel" is selected: Current, edited frames remain as they are
  - When "No" is selected: New, clear frames appear on application
  - When "Yes" is selected: Output needs to be in .tan2 format and interface for saving file to certain location
- When change is not made (frames are blank) and you select File to "New": nothing happens, blank frames remain on application.
- When change is made and File to "Open" or "Open Audio File" is selected: a popup appears with "Do you want to save your current project" with options "Yes", "No", and "Cancel"
  - When "Cancel" is selected: Popup goes away and edit(s) remain on frame(s)
  - When "No" is selected: Edits are cleared, interface for opening a .tan2 file
  - When "Yes" is selected: Output needs to be in .tan2 format, interface for opening a .tan2 file
- When change is not made and File to "Open" or "Open Audio File" is selected: interface for opening a file in .tan2 format appears
- When change is made and you go to File to "Save" or File to "Save as" in the application: output needs to be in .tan2 format and interface for saving file to certain location

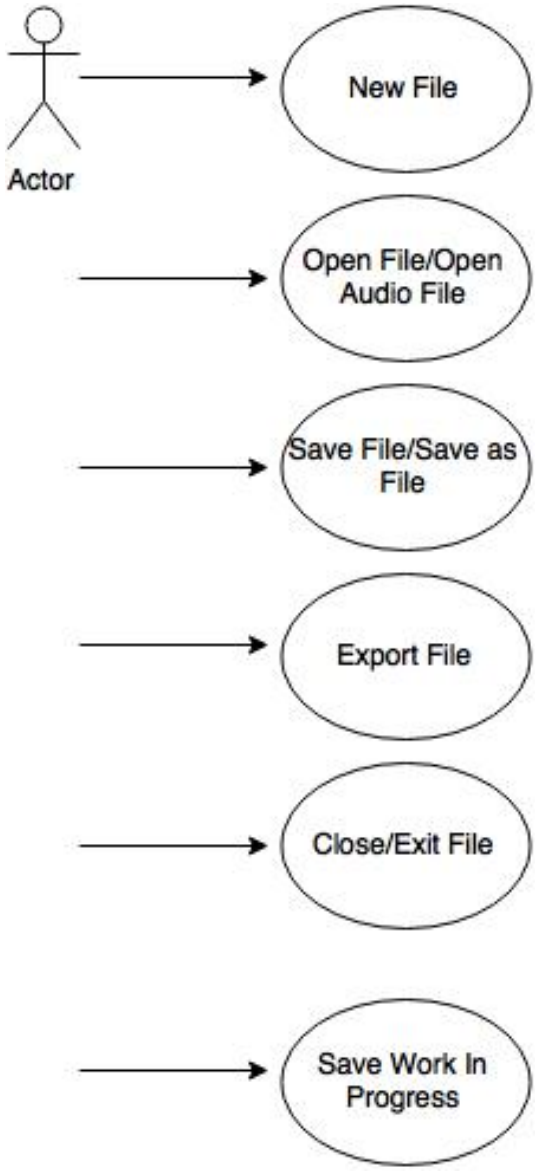
- When change is not made and you go to File to "Save" or File to "Save as" in the application: Re-save current file if file is already saved as .tan2 file or if the frames are blank, output needs to be in .tan2 format and interface for opening a .tan2 file.
- When change is made and you close out the application, go to File to "Close" or go to File to "Exit", a popup will appear with "Save", "Discard", and "Cancel" options.
  - When "Save" is selected: Output needs to be in .tan2 format and interface for saving file to certain location
  - When "Discard" is selected: application closes out
  - When "Cancel" is selected: application remains open
- When you select File to "Export": Output needs to be in .tan2 format and interface for exporting file to certain location
- .animator file for saving project: This is saving your work in progress, saving configurations and settings of a project, saving your objects that you created, so you can go back and edit the file again.
- When file has been open for 10 minutes, a popup appears: "You've been at it for a while, would you like to save?"
  - When "Yes" is selected: output needs to be in .tan2 format and interface for saving file to certain location
  - When "No" is selected: Application remains the same

4.1.3 Functional Requirements

The file output functional requirements for the Tower of Lights animator:

- Create a new file
- Open a file and audio file from a location (interface for browsing for a location)
- Name and save file to a location (interface for browsing for a location)
- Name and export a file to a location (interface for browsing for a location)
- Name and save .animator file (work and configurations in progress)
- Close and exit a file

4.1.4 Use Case Diagram



4.2 Color Selector

4.2.1 Description and Priority

The color selection feature allows the user to select colors from the RGB spectrum, by RGB value, Hex value, color history, or the saved color palette. The color selection feature is of high priority while the savable palette is a low priority.

4.2.2 Stimulus/Response Sequences

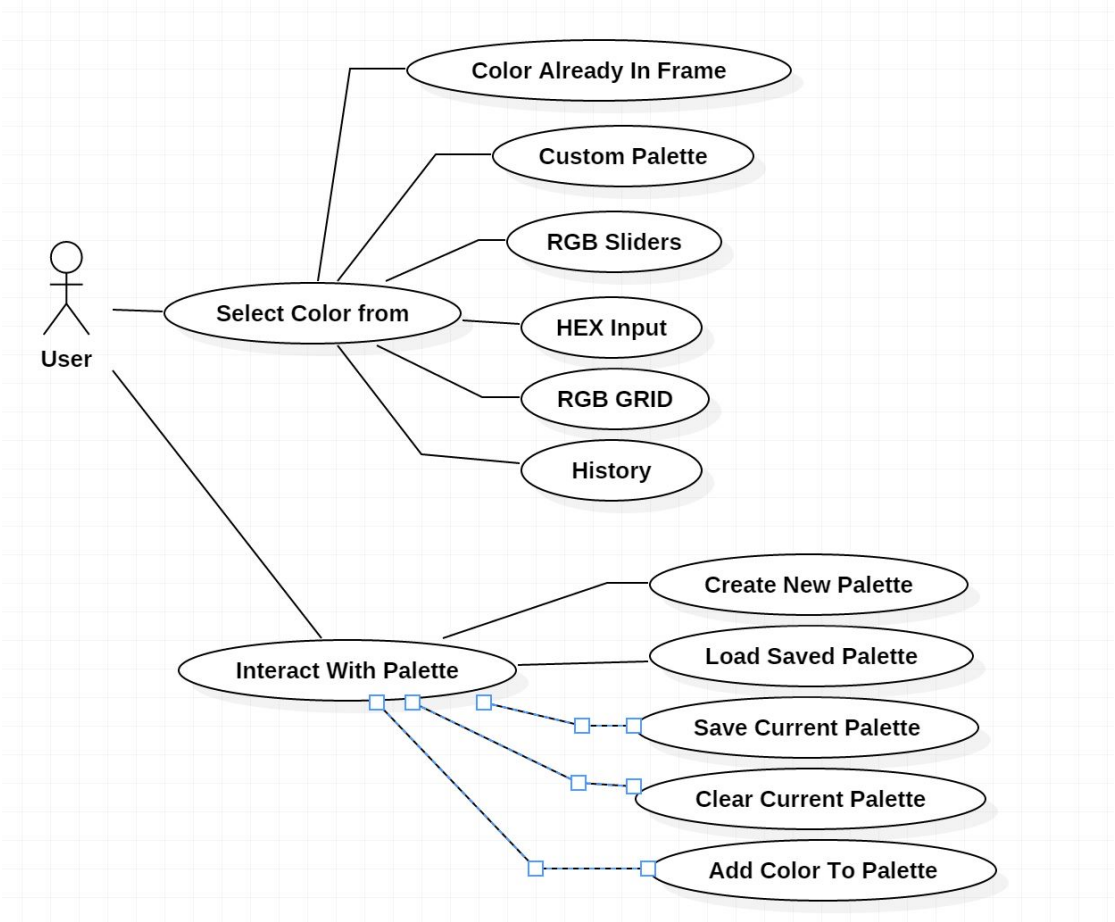
- The user is able to select a current color from:
  - The RGB color wheel/grid.
  - Using the RGB slider/inputing RGB values.
  - Inputing the Hex color value.
  - The history of colors area.
- Features of a savable color palette:
  - Users will be able to add colors to palette from the color currently selected or from the history of colors.
  - Users will be able to save their palettes for later use.
  - Users will be able to load palettes previously created.
  - Users will be able to clear their current palette.

4.2.3 Functional Requirements

The Functional Requirements for color selector:

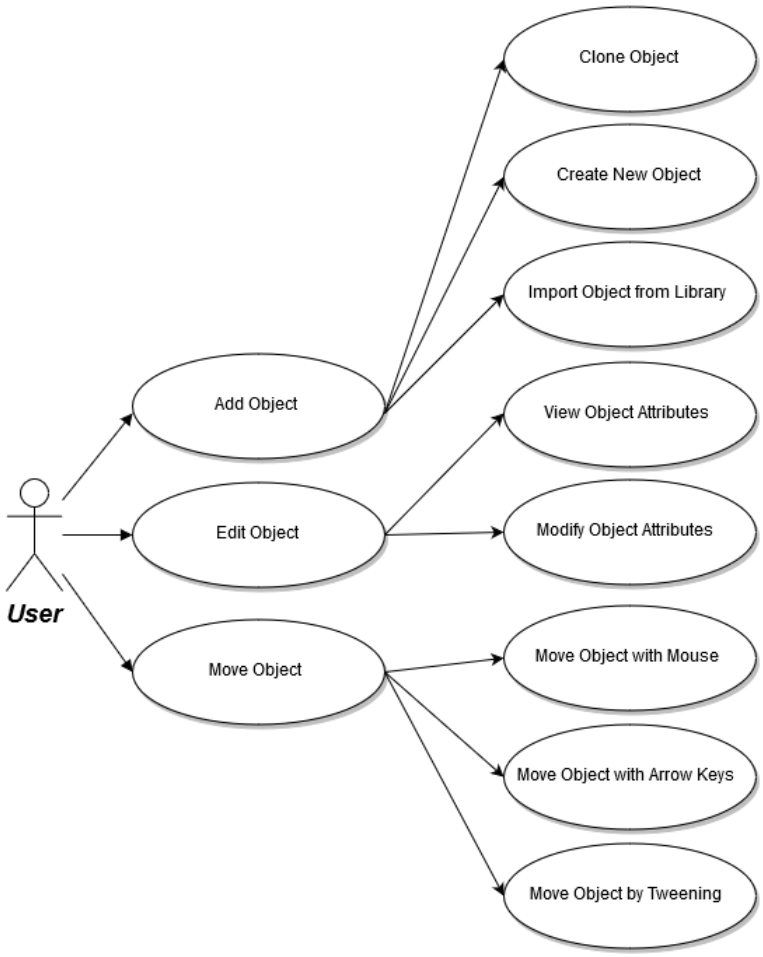
- Select a color from an RGB wheel/grid.
- Select a color by inputing RGB values.
- Select a color by its Hex value.
- Show a selectable history of colors used.
- Implement a savable color palette(if time permits).

4.2.4 Use Case Diagram



4.3 Object Interaction

Objects are a foundational building block in this animation program. This section details specifications for functionality that are concisely summarized in the diagram below. Additionally, these actions are ranked according to priority: high, moderate, and low. High priority items are essential to program functionality. Moderate priority items are requested by the customer and will improve user experience, but are not absolutely necessary for a functional program. Low priority items will be implemented after the others, if there is time.



UML Use Case Diagram for Object Interaction

4.3.1 Create New Object

Description and Priority

This action creates a new object. An object can be defined as a set of points that are grouped together in a set and share attributes. The action of creating a new object is very high priority because objects are the building blocks for frames, and frames are the building blocks of the animation that is produced.

Stimulus/Response Sequences

An object is created when the user selects "Create New Object" from the Objects menu or right clicks the main grid and selects "Create New Object" from the menu by the mouse. This action could also potentially be mapped to a shortcut key.

The user then names the object, with the default name being "Object N" where N represents the number of objects existing when the new object is created. The user also

must choose a color, or colors, for the object and set the object's shape and starting coordinates by left-clicking squares on the main grid in the desired pattern or holding the left mouse and using it as a "paintbrush" to color squares. Squares do not have to be contiguous, so the user can create a pattern of disconnected dots that only comprise one object. Similarly, the user can set the object to be part of one or more groups, which can be used to move or modify all member objects simultaneously. If the user does not select any groups, the object is only part of the group "All objects". Additionally, the user can set the object to be semi-transparent by modifying the "Opacity" attribute. If the user does not modify this attribute, it is set at 100 (completely opaque) as a default.

Functional Requirements

The functional requirements of this action are

- To create a new object from a toolbar menu option
- To create a new object from a drop down (right click) menu option
- To name, place, color, and identify the new object

4.3.2 Import Object From Library

Description and Priority

This action is similar to creating a new object, only instead of manually setting the color and shape, the user can select from a library of existing objects. This action is low priority because the program will be entirely functional without it, but it is still a target action because it will make designing animations much easier.

Stimulus/Response Sequences

This action is triggered when a user right-clicks over any empty space in the main grid area. A window pops up, with a list of objects on the left and a preview pane on the right. The user can then drag-and-drop and object from the preview pane to the grid or select OK and the object will be placed on the grid (in the middle? where the user right clicked?) and can then be moved.

Functional Requirements

The functional requirement of this action is

- To insert a new object with predefined properties into the program

4.3.3 Add Object To Library

Allows user to create custom object libraries that can persist through various projects and project iterations. This is a very, very low priority action and is unlikely to be completed this semester.

4.3.4 Duplicate Object

Description and Priority

This action duplicates an existing object. It is a medium priority action because although its functionailty can be duplicated manually, it will save the user time and effort.

Stimulus/Response Sequences

This action can be triggered by right clicking on an existing object and selecting the option "Clone" from the menu that appears by the mouse pointer. The newly cloned object is placed next to the original and can then be moved. This action can also be triggered if the user right-click selects an object from the object list, with the same response.

Functional Requirements

The functional requirements of this action are

- To create an exact copy of an existing object
- To allow the user to move and edit the new object

4.3.5 Display Object Attributes

Description and Priority

This action displays pertinent object attributes. It is a high priority action, because a user must be able to display attributes in order to edit them.

This action displays attributes in object pane on the right side of the screen. Object attributes include the following:

- Name
- Location coordinates
- Color(s)
- Layer
- Opacity
- Group(s)

This is a high priority feature because manipulating objects is a foundational part of this animation program.

Stimulus/Response Sequences

This action is executed when user left-clicks object or left-click selects object from list of objects.

Functional Requirements

The functional requirements of this action are

- To list the properties of a selected object

4.3.6 Edit Object Attribute

Description and Priority

This action is exactly what it sounds like: editing object attributes. Since it deals with fundamental object manipulation, it is essential and thus a high priority task.

Stimulus/Response Sequences

A prerequisite action is displaying the object attributes. Once the attributes display pane is open, the individual values can be edited. As the values are edited, they are automatically saved (alternatively, the values can be saved when the user left-clicks "Save", which is probably easier to implement).

Functional Requirements

The functional requirements are to display object attributes, and to be able to edit them.

4.3.7 Move Object With Mouse

Description and Priority

This action changes the coordinates of an object or objects (if a group of objects is being moved). It is a very high priority action.

Stimulus/Response Sequences

The user moves an object by left-clicking it with the mouse and then dragging it to the new location. An unnamed group of objects can be selected by shift-left-click-dragging the mouse to select an area or ctrl-left-click over each object that is to be moved.

Functional Requirements

To move the object, we need to implement the functionality of the shift and control keys. The objects location values will be changed when the object is moved.

4.3.8 Move Object With Arrow Keys

Description and Priority

This action allows the user to move objects by pressing the arrow keys. It is a moderate priority task because the user can also move objects using the mouse, so the program will be functional without it.

Stimulus/Response Sequences

Prerequisite to this action is that an object or group of objects must be selected, either from the objects list or from the main grid area. Once an object is selected and its attributes are being displayed in the right-hand attribute display pane, the user can move the object by pressing the arrow keys or, alternatively, the WASD keys. Optionally, these keys could be mapped by the user to any keys that are not already being used. For each time the key is pressed, the object will move one step in the corresponding direction. If the user holds the key, the object will move at a set (possibly increasing) rate in that direction. A lower priority part of this task would be to add the feature of having two arrow keys pressed at once, which moves the object in a combination of those directions.

Functional Requirements

The functional requirements of this action are

- To add the functionality of the arrow keys
- To update the location values of the object as it moves

4.3.9 Tweening

NOTE, This is more of a key-framing task than an object manipulation task!!!!

Description and Priority

Automatically generate intermediate frames between a specified start frame and a specified end frame to represent the movement of object. This is a moderate priority task that is not vital to program functionality but has been specifically requested by our customer Dr. Rinker.

Stimulus/Response Sequences

Functional Requirements

To be enumerated later.

4.4 Timeline and Frame Management

4.4.1 Description and Priority

The timeline is a linear representation of all frames in the animation, with the ability to select and manipulate frames in a number of different ways. This is a required functionality and visual representation for any basic animation and has a high priority.

4.4.2 Stimulus/Response Sequences

Selecting a frame will highlight it in the timeline. Deleting a frame will prompt the users with a confirmation dialogue to prevent accidentally deleting frames.

4.4.3 Functional Requirements

REQ-1: The timeline allows users to select a frame to edit. Once selected, the frame will appear in the editor and all related information for the frame will be displayed such as timing info, object info, and color presets.

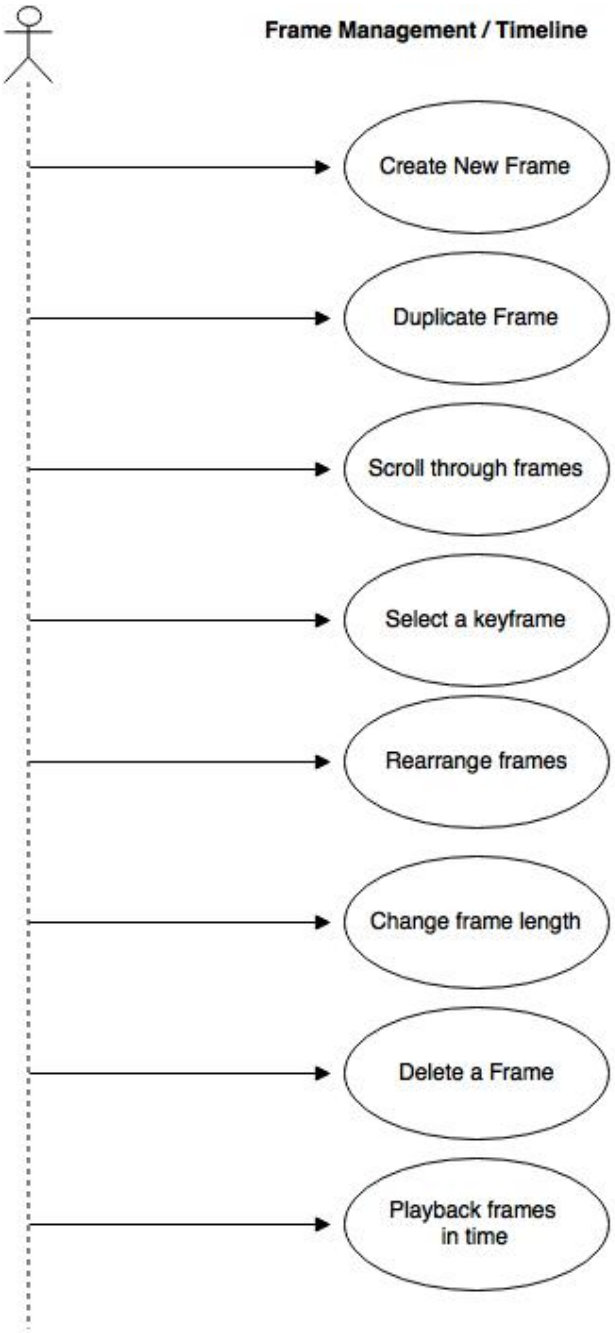
REQ-2: Users can duplicate a frame, placing the duplicated frame immediately next in line and adjusting its timing to match, and shifting all later frames to accommodate. All other frame info will also be duplicated.

REQ-3: Not all frames will be displayed on the timeline, but a scroll wheel will allow users to access all frames.

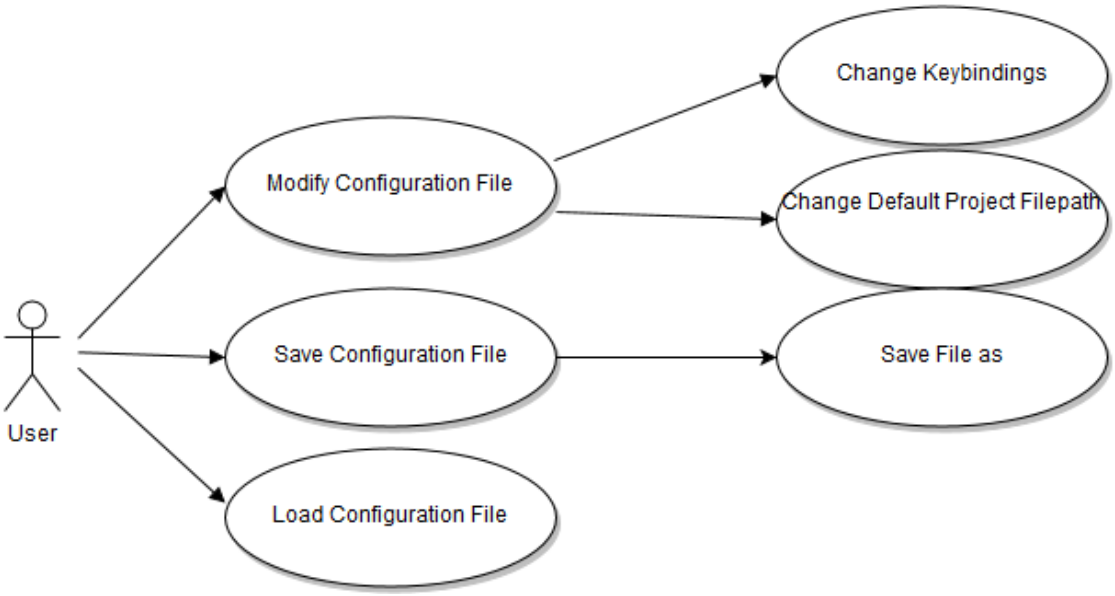
REQ-4: Clicking and dragging a frame will allow users to rearrange them. This will keep the duration of the frame, but will change its start and stop time to place it in the desired location. All other frames will also be shifted to accommodate the change.

REQ-5: Deleteing a frame will permanently remove a frame from the timeline and will shift all later frames to fill the gap.

REQ-6: There will be a play button on the timeline that will allow users to playback all frames with the selected music.


Frame Management / Timeline
Create New Frame
Duplicate Frame
Scroll through frames
Select a keyframe
Rearrange frames
Change frame length
Delete a Frame
Playback frames
in time

4.5 Program Configuration



UML Use Case Diagram for Configuring the Program

4.5.1 Description and Priority

The ability to customize various features provided by the application is common in other animation software as a means of increasing productivity. This is a proposed feature with a low priority.

4.5.2 Stimulus/Response Sequences

Pressing a settings button will open up a sub menu with the choices to modify the configuration file, save the current settings as a separate file, or load and use another configuration file.

4.5.3 Functional Requirements

The requirements of this action are to be able to read and write user defined settings from a file. The exact data to be stored in this file is currently unspecified.