Software Requirements Specification

for

Spritemón

Version 1.0 approved

Prepared by:
Egan Anderson
Jordan Davis
Makenzie Elliott
Jacob Osterloh
Jen Simons

University of Utah - cs3505

March 10, 2016

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this project is to develop a Sprite Editor called Spritemón. The entire scope of this project is covered in this SRS document. Version 1.0.

1.2 Document Conventions

Bold fonts have been used in this document when a specification refers to the Minimum System Requirement Specifications found in section 1.5.
Italic fonts have been used in this document when referring to a specific tool in the Spritemón program.

1.3 Intended Audience and Reading Suggestions

This document is intended for Professor Johnson, our TAs, and students of CS3505 who are interested in learning about our development process. It will also be used as a roadmap for our team while developing the product. It is organized on the top level by the most important categories of our planning process, including Introduction, Overall Description, External Interface Requirements, System Features, Other Nonfunctional Requirements, and other Requirements. Each of these categories are then divided into subcategories to further explore their specifics details. For those who want to learn about the main features that Spritemón supports, see sections 2.2, 3, and 4. For developers learning about the specifics of implementation, see sections 2.2, 2.3, 2.5, 3, 4, 5.

1.4 Product Scope

The purpose and objective of Spritemón is discussed in section 1.1. The goal is to produce a Sprite Editor that is user friendly and visually appealing. Course goals include being satisfied with our product and earning an A in CS 3505.

1.5 References

Minimum System Requirement Specifications:

A6: Planning Phase for a Sprite Editor by David Johnson on March 10, 2016: https://utah.instructure.com/courses/367600/assignments/2871038

2. Overall Description

2.1 Product Perspective

Spritemón is a stand alone product to improve upon standards that have been observed in existing Sprite Editors. (See Existing Marked Document on page 7.) UML is provided on page 9.

2.2 Product Functions

Requirements summary:

User will be able to:

- **draw custom images using a *brush tool***
  - undo and redo a drawing action
  - **change the size of the pixel created by the *brush tool***
  - use an *erase tool* to set the pixel back to transparent
  - use a *lighten tool* to lighten drawn pixels
  - use a *darken tool* to darken drawn pixels
  - use a *color picker tool* to choose a color from a pixel on the canvas
- **set/adjust number of frames of an animation**
  - duplicate frame
  - add frame
  - delete frame
  - **view the full animation and modify the frames per second in a preview window**
- **save & load projects (ASCII) via menu**
- **export sprites as an animated gif via menu**
- hover mouse over tool buttons to see button name and a “how to use” message
- use a *fill tool* that changes multiple pixels to a desired color

Our UML document should be referenced for a picture of the major classes that will handle these requirements and how they relate.

2.3 User Classes and Characteristics

As stated above, one goal is to have Spritemón be user friendly with the goal of having beginner's be successful with this program. Little to no technical expertise would be required to use the program.

2.4 User Documentation

Spritemón will offer important information about each tool as the user’s mouse hovers over it. This information will appear in a message terminal at the bottom of the screen. This will allow the user to learn how to use each tool without having to reference a separate manual or tutorial.

2.5 Assumptions and Dependencies

We are assuming that QPixMap can be used in conjunction with our model object to keep track of the various components of each drawing. If QPixMap is not compatible with our project the way we expect, we will have to spend more time developing the low level elements of the drawing functionality and will have less time to develop more advanced features.

3. External Interface Requirements

3.1 User Interfaces

The user will only interact with one main screen, divided into different windows. The interaction with each of these windows is standard and straight-forward. The user can change tools, manipulate frames, and control animation by clicking the respective buttons. The selected frame can be edited by clicking or dragging the mouse within the canvas window. The save and open functionalities will be housed in the menu bar. The user will be notified of any errors through a QMessageBox.

3.2 Hardware Interfaces

Computer and Monitor: Necessary in order to run and see the program.
Mouse: Essential to navigating the program and interacting with the interface.
Keyboard: Only necessary for entering the name of the file to be saved.

4. System Features

4.1 Frames

The *Frames* subwindow displays all of the frames that comprise the current project. The initial state of the project will have one frame. Subsequent frames can be added by pressing the “+” button. Frames are manipulated in the canvas subwindow. Use the canvas to change the drawing, duplicate the current frame, or delete the frame. This feature is of high priority.

4.2 Canvas

The *Canvas* subwindow shows a detail view of the frame that is currently selected. It allows the user to see in greater detail the drawing in that frame and allows them to manipulate it using the tools from the *Toolbar*. It contains *Undo* and *Redo* buttons which will reverse or redo the most recent action done by the user. It contains the *Clear* button which deletes all brush strokes in the current frame. It contains the *Duplicate* button which copies the current frame and uses it to create a new frame. It contains the *Delete* button which deletes the current frame.

4.3 Toolbar

The Toolbar subwindow will hold buttons to activate the indicated tool after mouse click:

- *Brush tool*: The pixel clicked will be changed to the color the user chose from the Color Chooser subwindow (the default color will be black if no color is chosen).
- *Erase tool:* The pixel clicked will be changed to transparent.
- *Fill tool:* The clicked pixel and pixels that are the same color as that pixel will be changed to the color specified by the user in the Color Chooser subwindow.
- *Lighten tool:* The clicked pixel will lighten slightly from its current color.
- *Darken tool:* The clicked pixel will darken slightly from its current color.
- *Color picker tool:* This will set the *brush tool* or *fill tool* to the color of the pixel that was clicked.

4.4 Color Chooser

The *Color Chooser* allows the user to select a color or change the color they are currently using. This can be executed by selecting the color of choice in the *Color Chooser* subwindow. The selected color will be displayed within the subwindow and will appear in any new brush strokes until the color is changed again.

4.5 Preview

The *Preview* sub-window will playback a scaled version of the the entire animation project to the user. This feature will sequentially show each frame in the specified order to create an animation. User will be able to change the *frames per second* speed using this sub-window by clicking on a slider and modifying its position. The user will also be able to view the actual size of the animation by selecting a button that will toggle animation size from actual to a smaller scaled value.

4.6 Menu Bar

The *Menu Bar* will have a *File* tab that allows the user to save and load a project file. The user can save a file by clicking on the *File* tab and selecting the *Save* option or load a project by selecting the *Load* option. Once selected a dialog box will appear to give the user the ability to name an unsaved project or load a project that has previously been saved. The *File* tab will also have an option to let the user export a sprite to an animated gif. The user can export the sprite by clicking on the *File* tab and selecting the *Export* option, a dialog box will appear that will allow the user to name the export and give it a location to save to. The *Menu Bar* will also have an *Edit* tab that will allow the user to *Undo/Redo* an action by selecting the *Edit* tab from the *Menu Bar* and selecting either the *Undo* or *Redo* option. The *Undo* and *Redo* option will reverse or redo the most recent action done by the user.

4.7 Message Terminal

The *Message Terminal* is a label at the bottom of the screen that will describe to the user the purpose of a tool. This feature is active whenever the cursor hovers over a tool. This feature is of low priority.

5. Other Nonfunctional Requirements

5.1 Safety Requirements

One of the safety concerns related to Spritemón involves saving the file. The first concern is that the user will lose past projects by saving new ones witht the same name. We need to alert the user if such a situation exists and make sure that they want to proceed with the save. The second concern is that the user will close the project without saving. We will alert the user that unsaved changes will be lost if they close the project and give them the option to save before proceeding with the close.

Another safety concern involves loading a file. If a user tries to load a file into Spritemón that is incompatible with the program we need to alert them of the situation.

6. Existing Market document

6.1 Piskel

The Piskel sprite creator has powerful functionality as well as an intuitive design and organization. One of the most prominent features on the page is the live preview. Every time a change is made to one of the frames, that change can immediately be observed in the live preview of the animation. This could help determine quickly whether a certain change improves the desired animation. This tool also allows the user to export their animations to a gif. This is an important feature since the sprites are often meant to be viewed in motion. Piskel offers many features that are too complex for beginners. Particularly the layering, shape selection, dithering tools could be confusing for somebody who is new to the concept of sprite generation. While these are powerful tools for those who know how to use them, they make the program more difficult for beginners to use.

Reviewed by: Egan Anderson
http://www.piskelapp.com/

6.2 Sploder

The Sploder graphics editor has many useful tools to assist the process of creating an animation. Some advantages of this program is that each frame allowed you to add multiple layers as well as change the entire frames hue, saturation, and brightness. Having multiple layers is useful because you can make changes to parts of your frame without modifying the other contained elements. Some other things I liked from his example was that it kept track of all the recently used colors, had various brush shapes. Despite this advantages, there were some important features that were missing that would cause me not to recommend this application. Biggest disadvantages were that there was no preview window for the animation and you couldn’t specify the size of the brush. It allows you to choose a canvas size at the beginning of your program, but the pen size is then scaled to the canvas size and immutable. Also, the user interface was very busy and hard to find which tool I wanted to use. If you hovered over a tool it would give a simple description what it was and a hotkey you could use to switch to the tool, but still wasn’t very user friendly.

Reviewed by: Jordan Davis
http://www.sploder.com/free-graphics-editor.php

6.3 Pixel.tools

The Pixel-Art Editor’s has an easy to use interface. Some of the tools are labeled by picture that gives a good description of what you can expect, while others are questionable. The amount of tools is not overwhelming but give enough to allow the user to be creative. Some of the advantages include the sliding bars at the bottom of the editor that allow you to manipulate the brush. There is also an array of numbers that when clicked will automatically set these fields for you so you don’t have to worry about getting back to the same setting, a good tool for consistency. Some disadvantage to this Editor is that it doesn’t give clear instructions on how to use the tools if the user had any questions. Another disadvantage is that some of the tools aren’t working but it’s most likely due to the fact that the user doesn’t know how to correctly use them.

Reviewed by: Makenzie Elliott
http://www.pixel.tools/

6.4 Pickle Editor

The Pickle Editor has a straightforward user interface. The tools are clearly labeled and visible which makes it very easy to add frames, remove frames, edit frames, and display frames. It also has many additional features such as canvas zooming, which allows the user to resize the panel to their convenience, and a mirror drawing mode, which allows the user to draw symmetrically across the X and Y axes. There are also hot keys that allow the user to easily switch between different tools without moving and clicking another tool. On the other hand, it isn’t very easy to use customized colors with this editor because there is no color wheel. A color wheel could be very helpful for the user to discover the colors that would work best for a specific sprite. Unfortunately, the use of this editor isn’t free. In fact, it costs $25, which doesn’t seem worth the extra functionality this editor may provide for a given project.

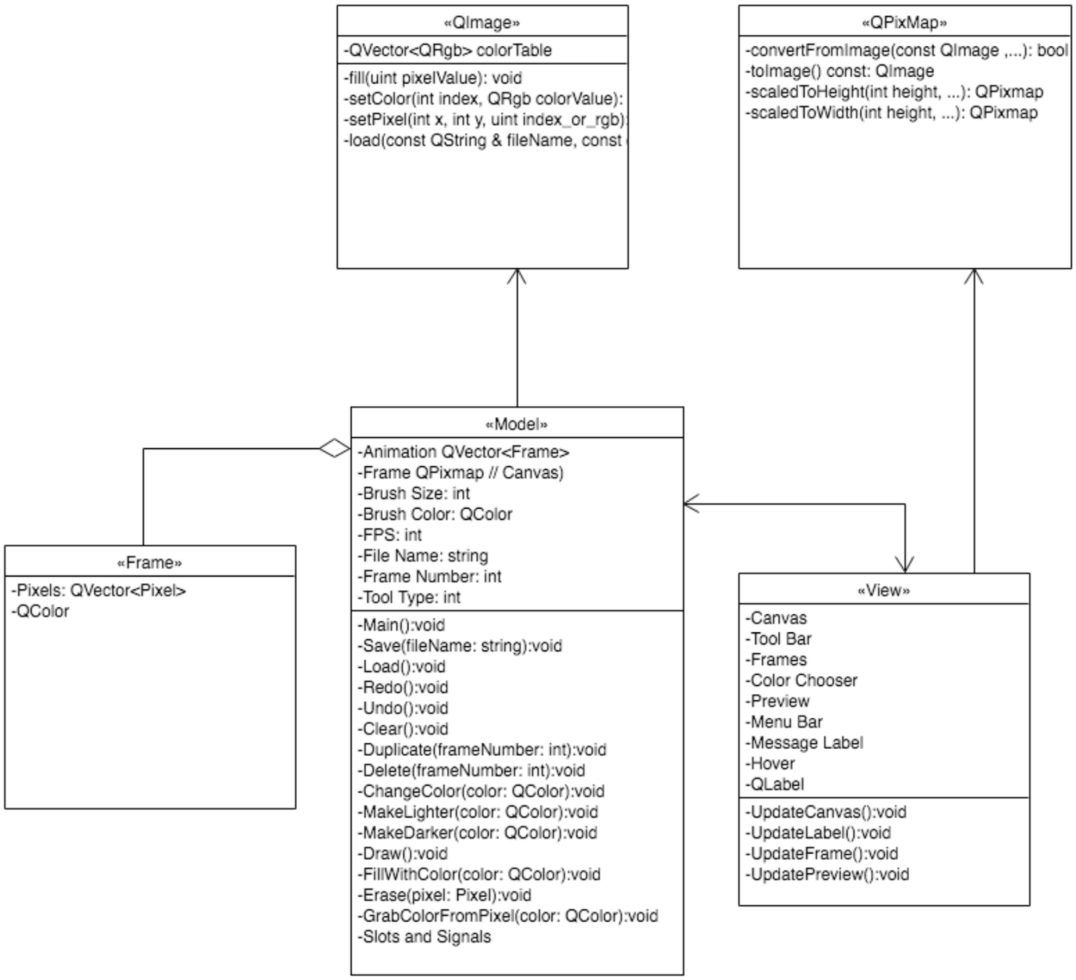
Reviewed by: Jake Osterloh
http://www.pickleeditor.com/

6.4 C-64 Sprite Editor

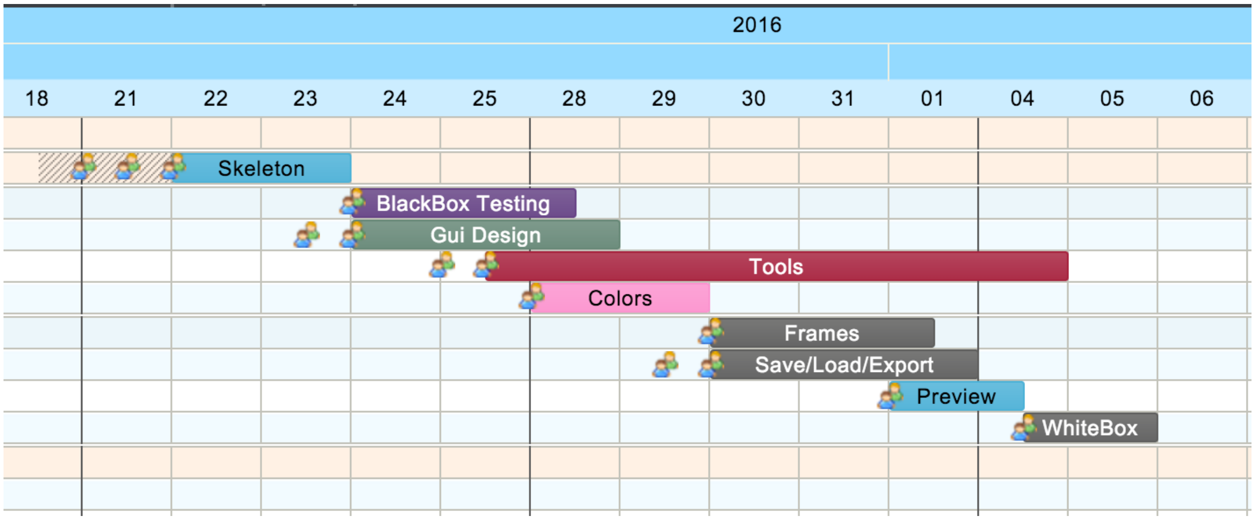
The C-64 Sprite Editor has a simple and clean webpage. When the mouse is hovered above a tool in the editor, clear instructions that explain how to use each tool and what it does are displayed at the bottom of the page. This is a simple and clear way to give instructions on how to use the tools in the editor. The webpage didn’t get much attention and it’s hard to say if that was by design, that the designer ran out of time for some reason or just general laziness in taking pride in appearance as well as functionality.

Reviewed by: Jen Simons
http://www.ccpssolutions.com/tools/c64se/

7. UML Diagram


«QImage»
-QVector<QRgb> colorTable
-fill(uint pixelValue): void
-setColor(int index, QRgb colorValue):
-setPixel(int x, int y, uint index_or_rgb):
-load(const QString & fileName, const
«QPixMap»
-convertFromImage(const QImage ,...): bool
-toImage() const: QImage
-scaledToHeight(int height, ...): QPixmap
-scaledToWidth(int height, ...): QPixmap
«Model»
-Animation QVector<Frame>
-Frame QPixmap // Canvas)
-Brush Size: int
-Brush Color: QColor
-FPS: int
-File Name: string
-Frame Number: int
-Tool Type: int
-Main():void
-Save(fileName: string):void
-Load():void
-Redo():void
-Undo():void
-Clear():void
-Duplicate(frameNumber: int):void
-Delete(frameNumber: int):void
-ChangeColor(color: QColor):void
-MakeLighter(color: QColor):void
-MakeDarker(color: QColor):void
-Draw():void
-FillWithColor(color: QColor):void
-Erase(pixel: Pixel):void
-GrabColorFromPixel(color: QColor):void
-Slots and Signals
«Frame»
-Pixels: QVector<Pixel>
-QColor
«View»
-Canvas
-Tool Bar
-Frames
-Color Chooser
-Preview
-Menu Bar
-Message Label
-Hover
-QLabel
-UpdateCanvas():void
-UpdateLabel():void
-UpdateFrame():void
-UpdatePreview():void

8. Gantt Chart


Skeleton
BlackBox Testing
Gui Design
Tools
Colors
Frames
Save/Load/Export
Preview
WhiteBox