Software Requirements Specification

For

Sprite Editor Software

Version 1.0 approved

Prepared by Alisha Seo, Aimal Esa, Madelyn Sudac, Luis Alvarado Labarca, Ashley Lujan, James Donaldson

University of Utah, CS 3505 (Software Practice II)

November 2, 2023

1. Introduction

1.1 Purpose

Sprite Editor Software version 1.0. This requirements document will cover all of the requirements given to us by Professor Johnson in Assignment 7 - Sprite Editor

1.2 Document Conventions

All links are underlined. No other writing conventions were followed. Anything of significance or importance is emphasized in writing.

1.3 Intended Audience and Reading Suggestions

This document was created to be reviewed and graded by CS3505 course staff. It is also intended to be referenced by developers during the creation of the Sprite Editor Software.

1.4 Product Scope

The Sprite Editor Software will allow users to create pixel art. The goal is to create a simple and easy to use pixel art environment.

1.5 References

Specification Template

Market Research Document

SRS Reference/Notes

QT.io

2. Overall Description

2.1 Product Perspective

This is an original product intended for submission for a grade in CS 3505. Features take inspiration from other pixel editors found during market research (see Market Research Document).

2.2 Product Functions

- Canvas
- Frame Animator
- Palette

- Tools View
- Menu Bar
- Save and Load
- Export and Import

2.3 User Classes and Characteristics

Expected users to use this product are people interested in getting started in pixel art. Users will probably have a low level of experience in pixel art. This product will not satisfy those with a high level of experience.

2.4 Operating Environment

The program will work on both Windows and macOS. Additionally, the program is going to be made with recent versions of Qt, which is Qt 6.6.0 and its libraries. According to the website, qt.io, the operating systems which may run this are Windows 10 (1809 or later) and macOS (12/11/10.15) (64bit Intel, 64bit ARM; XCode 12) or later, and other versions of Linux, which will not be used.

2.5 Design and Implementation Constraints

The program will be constrained by the capabilities of Qt.

2.6 User Documentations

The program will come with a help menu that provides basic documentation for the editor

2.7 Assumptions and Dependencies

The Sprite Editor Software assumes that the working libraries and dependencies are compatible with Qt and Qt libraries as it is developed using Qt. The Sprite Editor Software also uses qmake as its default Makefile generator. If the dependencies and libraries are incompatible with Qt or are made with a different Makefile generator, there may be unanticipated outcomes or the program may become faulty. Therefore, any modifications or updates to the Sprite Editor Software should be done with Qt-compatible libraries or dependencies.

4. System Features

4.1 Choose and Draw Shapes

4.1.1 Description and Priority

This feature allows users to choose shapes from circles and rectangles from the toolbar and have them be drawn onto a location in the canvas. This is a low priority, as it is an extra feature. The user benefit is 9/9 as it will enhance the user's ability to draw. The penalty is 4/9 as it might be a complicated feature. The cost is 4/9, only because there will be work time and research time that has to go into this. The risk is a 2/9, since the feature will only add on top of existing functionality, but it may affect the program’s rendering if a problem like that occurs.

4.1.2 Stimulus/Response Sequences

The addable shapes will exist as options on the toolbar and users will have the ability to choose a shape and place it on the canvas. On choosing the shape, the next time a user clicks on the screen, it should draw the shape. Additionally, shapes should have the ability to have their measurements determined by the user.

In case the user drops the shape outside of the canvas. In that case, no shape should be drawn. Another case is that the user drops the shape in a place where some of it falls on the canvas and some outside. In that case, only the part of the shape that is within those bounds should be drawn.

4.1.3 Functional Requirements

REQ-1: Button clicked event for choosing a shape.

REQ-2: Mouse clicks on canvas to draw the shape.

REQ-3: Mouse press event or popup that allows the user to determine the shape’s size.

REQ-3: Getting Mouse x,y location for drawing the shape.

REQ-5: Ability to redraw canvas.

REQ-6: Ability to get canvas’s x, y

4.2 History (Undo/Redo)

4.2.1 Description and Priority

This feature allows the user to undo or redo a drawing sequence as they are drawing on the Canvas (see Canvas 4.6). Each undo or redo should be decided by a mouse cursor release. This should be considered as a Medium Priority. The benefit of having this feature is 8/9, because it allows the user to go back to a specific history

of their current frame without having to erase or modify their drawing. The penalty of not having this feature is 4/9, because it will be inconvenient for users when they decide they need to revert their art back to a specific point in its drawing history but are unable to do so without modifying it or if they forgot what it looked like before. The cost of having this feature is 6/9, because it would require the Sprite Editor Software to keep track of a multitude of modifications made to the drawing and find a way of restoring those modifications. The risk of having this feature is 7/9, because unless careful checks are in place, the user may be able to undo or redo the pixel art states without there being anything to undo or redo back to.

4.2.2 Stimulus/Response Sequences

With each mouse release after the user is painting (4.4), erasing (4.5), duplicating (4.3), or adding shapes (4.1), the software should capture the current pixel art state and save it into some form of storage within the software.

Each time the user presses the undo button, the software should restore the immediately preceding pixel art state that the current frame was in and replace the current art with the previous one. If there are no more pixel art states to restore, the software should disable the button to prevent the user from further undoing the art. This button should only be available after the user has added an object or paint to the Canvas, when the user first starts. Note that it should not be enabled if the user tries to erase a blank Canvas.

Each time the user presses the redo button, the software should restore the art state that was immediately after (“future” change) the current pixel art state. This should only happen if there is a “future” change the software can revert back to. Considering this, the software should take care to not enable the button until the undo button has been pressed at least once. The software should also disable the button if there is no “future” change that was made to the pixel art state that can be restored.

4.2.3 Functional Requirements

REQ-1: An undo button that restores a preceding pixel art state

REQ-2: A redo button that restores a pixel art state that was made after the current pixel art state

REQ-3: Software check that disables the buttons if there are no more states than can be restored

REQ-4: Software check that enables the buttons if there are pixel art states than can be restores

REQ-5: If the Sprite Editor Software encounters issues with redo or undo, the software should try to save the storage of pixel art states so the user does not encounter issues with redo or undo later on

4.3 Duplicate

4.3.1 Description and Priority

This feature allows users to copy the contents of a frame, and duplicate it onto a new frame. This is a low priority feature, as it is an extra feature but it will be a significant contribution to the overall functionality of the program. The user benefit is 7/9 as it will be very helpful to users that want to create sprites with smooth animations. The penalty would be 6/9 since having to repaint each new frame of the sprite would be an added struggle for the user. The cost is 3/9 as adding a duplicate screen is a more simple implementation, and likely will not require complex code. The risk is 3/9 since each execution of this feature should be similar, but errors in this feature would cause significant problems for the user.

4.3.2 Stimulus/Response Sequences

The ability to duplicate an existing frame will exist as an option for each individual frame. When the appropriate button is pressed, a new frame should be added immediately after the frame being duplicated, and the new frame should be a complete duplicate of the original frame's painted contents. Empty frames should be able to be copied and pasted using this feature.

4.3.3 Functional Requirements

REQ-1: A button for the copy and paste feature

REQ-2: A button clicked event for executing the copy and paste feature

REQ-3: Knowing where the frame is in the list of frames, in order to add the duplicate immediately afterwards

REQ-4: Being able to record the locations of each painted element and its color

REQ-5: Ability to use the locations and colors of the painted elements to redraw them on a new frame

4.4 Paint

4.4.1 Description and Priority

This feature allows the user to draw on the canvas (see 4.6 Canvas). The user has the ability to drag or click on the canvas to change a pixel into the user's desired color. The Paint option allows the user to decide the Paint (brush) size. This should be considered as a High Priority. The benefit of having a paint feature is 9/9, as it provides the main functionality as a Sprite Editor Software. The penalty is 9/9 as not having this feature will make the editor practically useless. The cost of having this feature implemented is 1/9, as it can serve as the base of multiple click/drag features

of the pixel editor as it should be one of the first features to be implemented. The risk is 3/9. Having this feature is slightly riskier as the developer will have to consider the bounded areas the user is allowed to paint in. In addition to bounds, the developer will also have to consider when the feature is in “paint mode.”

4.4.2 Stimulus/Response Sequences

The user will be able to select a Paint brush icon to enable “Paint mode”, which should be available in a toolbar. When this is selected, the Sprite Editor Software should disable the previously used feature and enable the Paint feature. The software should allow the user to select and use different colors and brush sizes.

The user will be able to apply color by clicking and/or dragging their cursor over the Canvas. When the user does this, the software should apply the selected color and brush/pixel size in accordance with the cursor position on the canvas.

The user should be able to draw on a blank canvas and draw over previously drawn objects on the canvas with different colors. When drawing over colors, the software should overwrite the previously drawn color with the new color and disregard it. At this time, the software should save the new color.

The program should prevent the user from being able to use the “Paint mode” outside of specified bounds of the canvas. The software should be able to recognize if the user is drawing outside the bounds of the canvas and “disable” the Paint feature to prevent the user from coloring.

4.4.3 Functional Requirements

REQ-1: A button that allows the user to select a “Paint mode”

REQ-2: Software restrictions that should prevent the user from coloring outside the Canvas bounds

REQ-3: A color palette system that allows the user to select a color

REQ-4: Software response to clicks and drags that adds color to the Canvas

REQ-5: A button or drop down that allows the user to select the brush size

REQ-6: Software response to overwriting previously drawn colors if a new color is selected and drawn over the same area

4.5 Eraser

4.5.1 Description and Priority

The eraser will allow the user to erase any marks they have made on the Canvas. It will do so by setting the color of that pixel to fully transparent. This is a high priority

item as it is essential to the sprite editor. The benefit of having this feature is a 9/9 because it allows the users to change or correct any of their drawings easily. The penalty of not having the eraser is a 9/9 because users would have to create a whole new drawing if they made a small mistake. It would also probably lead to losing many users. The cost of implementing this feature is a 2/9 as all it needs to implement is changing the color of the pixel to a fully transparent pixel of any color. The risk of implementing the eraser is a 2/9 because if it is implemented incorrectly, it can ruin the user's art.

4.5.2 Stimulus/Response Sequences

The user will click on the eraser icon which will then be highlighted. Then they are able to use the eraser in the same manner as the pen described in 4.4.2 except they cannot change the color of the eraser as it will only make things transparent.

4.5.3 Functional Requirements

REQ-1: A selector button for the eraser

REQ-2: A button clicked event to change the clicked pixel into a transparent pixel

4.6 Canvas

4.6.1 Description and Priority

The Canvas serves as a digital painting canvas, where it provides a set location on the screen where the user may alter appearance. The canvas is a surface where pixels may be filled in with colors and shapes. This is a top priority because it is one of the main functionality the program promises since it is a sprite editor. The benefit is a 9/9 since without the Canvas, there would be no program. The penalty is 0/9, because it would in fact be a huge penalty if we did not have this at all. Cost is small at 2/9 since it is not entirely difficult to draw on Qt and there are existing modules for it. Risk is a 0/9 since it is a simple feature and should be in the program.

4.6.2 Stimulus/Response Sequences

The Canvas should react to the mouse's movement and behavior over the Canvas itself. The appearance of the Canvas should be updated when certain buttons are pressed, like undo or redo. When the user is copying and pasting, the Canvas should be able to return the information of the pixels selected.

Additionally, the Canvas should be able to reorganize itself according to the pixel width/height the user specified.

4.6.3 Functional Requirements

REQ-1: Mouse events to process click and hover.

REQ-2: Display and draw.

4.7 Frames

4.7.1 Description and Priority

The frames widget showcases the current sprites that are to be used in an animation. Frames will be ordered in the sequence in which they will appear in the animation. Sprites can be added to the frame sequence in any desired location decided by the user. This is a top priority as it's fundamental to sprite animation. The benefit of the frames widget is 9/9 as without it we would be left with a paint program. The penalty of not implementing this would be 9/9 as we would not have the ability to animate sprites. The cost of implementation will be a 2/9 as its a fairly basic feature to develop. The risk of implementation is a 0/9 as it's a core aspect of the editor.

4.7.2 Stimulus/Response Sequences

Frames will respond to multiple user actions like that of clicks and drags. When the user clicks on the add/remove sprite button the frame sequence should be updated corresponding to the action. When a user clicks on a sprite in the sequence the Canvas should display the sprite and the remove button enabled for clicking. Further, if a user drags a clicked drag it should be moved into the position the user decides.

4.7.3 Functional Requirements

REQ-1: Display of sprites in a sequence

REQ-2: Add sprite button that when clicked adds a sprite to the sequence

REQ-3: Remove sprite button that when clicked removes a sprite from the sequence

REQ-4: Ability to select a sprite from the sequence and display it in the Canvas

REQ-5: Ability to select a sprite and remove it using remove sprite button

REQ-6: Ability to determine sprites order with user input

4.8 Saving and Importing

4.8.1 Description and Priority

This feature will allow the user to save the current sprite and import an existing sprite and continue to edit it. This is a high priority feature since it is an integral part of the

program as a whole. The user benefit is 9/9 since it will prevent work from being lost after exiting the program. The penalty is 8/9 because without the ability to save and import sprites, the editor isn't a useful tool. The risk is 4/9 because preserving the sprite's frames when saving and importing might pose a challenge when implementing.

4.8.2 Stimulus/Response Sequences

Clicking on save in the menu will allow the user to choose the name of the file before it's saved to the local computer. Clicking on import in the menu will open a pop up window showing the file explorer, and the file that is selected will be opened in the editor.

In the case that the user tries to open a file that is incompatible with the sprite editor, the file should not be opened, a message should be displayed explaining why the file cannot be opened in the sprite editor, and the file explorer should stay open to allow the user to select a different file to import into the editor. In the case that the user tries to save a sprite file with no name, the file should not be saved, a message should be displayed saying that a filename is required in order to save the sprite file, and the user should be given another chance to give a file name.

4.8.3 Functional Requirements

REQ-1: Menu buttons for both the save and import options

REQ-2: Pop-up window for the save feature, prompting the user to provide a file name

REQ-3: Pop-up window for the import feature, prompting the user to select the desired file from the file explorer

REQ-4: Ability to check the compatibility of a selected file with the sprite editor

REQ-5: Ability to preserve the sprite's individual frames when saving and importing

4.9 Preview Animation

4.9.1 Description and Priority

The feature will allow the user to preview animations created in the sprite animator. Each frame will be animated at a specific speed (FPS) defined by the user inside the sprite animator. The user benefit is 5/9 as it will be very useful for those creating animations, but not useful for those just using the Pixel Editor Software to create individual sprites. The penalty is 8/9 as if the user is unable to preview the animation they've made, then it will be very inconvenient for them to make the animation look how they would like it to. The risk is 3/9 as if we are able to create the sprite animator and save/export, it will be very easy to implement a preview panel. Cost is

4/9 because it requires some time to send each frame to the UI and requires some complexity that will determine how fast (FPS) the animation will run.

4.9.2 Stimulus/Response Sequences

Upon adding a frame to the animation editor, the frame should be shown in the preview window. When another frame is added to the animation,, the preview pane should automatically play the animation, even if the animation was paused previously.

The preview window should have a button that allows the user to pause or play the animation. The speed of the animation should be able to be adjusted in the animation view on the bottom of the screen, using a FPS slider. The minimum value of the playback speed should be 1 FPS and the maximum should be 30 FPS. A higher FPS value should result in a faster playback speed.

4.9.3 Functional Requirements

REQ-1: The preview panel cycles though frames in the animation view.
REQ-2: The preview can be paused and played with a button.
REQ-3: The speed of the animation in the preview pane can be adjusted using an FPS slider in the animation view.

4.10 Color Palette

4.10.1 Description and Priority

The color palette allows users to change their pen and brush color by selecting a color from the Color palette, which displays a certain number of colors the user can pick. This is a medium priority. Technically, the program can be written without this implementation and only allow a user to draw a black color. However, it is an assignment requirement, so this needs to be implemented. The benefit is a 9/9 since users will have more creative liberty. The cost is 4/9 as there needs to be research done on the team’s side. Penalty is 0/9 since having this feature will benefit user experience and it doesn’t seem to be a challenging feature. Risk is 0/9 since it is an independent aspect of the project.

4.10.2 Stimulus/Response Sequences

The color palette should respond to a user clicking on a color. The proper response is the program changing the color it is drawing with to the current color selected. There is a case where a user tries to draw without having clicked on a color. In that case, the default color should be black.

4.10.3 Functional Requirement

REQ-1: Menu type library that makes it easy to know what color has been chosen
REQ-2: Ability to display RGB on screen
REQ-3: On-select event for color options

4.11 Clear

4.11.1 Description and Priority

The clear function allows the user to completely erase all modifications the user has made on the current working frame. This feature should be considered as a low priority as there are other options for erasing and undoing modifications (see Eraser 4.5 and History 4.2). The benefit of having this clear functionality is 6/9, because it makes it easier for users to erase ideas and artwork without the use of the eraser tool. The cost is 2/9, because the complexity of having to delete everything from a frame should be easy to implement. The penalty of not having this feature is 3/9, because, as aforementioned, there are other options of having to clear a frame such as deleting a frame or using an eraser. The risk of implementing this feature is 5/9, because it would be an inconvenience if it’s not implemented correctly, e.g. some other features or artwork ends up being cleared when the user tries to use this feature.

4.11.2 Stimulus/Response Sequences

When the user presses the clear button, the system should respond by clearing all the modifications the user has made. This can be done in multiple ways and is up to the developer to decide how it should be implemented. However, this feature should not be able to clear all modifications made to all frames and just the working frame, nor should the modification be able to delete a frame (that is a separate feature under Frames 4.7).

4.11.3 Functional Requirement

REQ-1: A button that allows the user to clear their current working frame
REQ-2: If there were no modifications made to the frame, the feature should do nothing.
REQ-3: Ability to draw certain RGB colors on screen.