The Project

Your team is going to develop a Qt-based sprite editor. A sprite is a small image used in games or other GUI applications that is rendered at different locations on the screen. Most sprites have animation cycles associated with them, so that a sprite is really an array of small images. The sequence of images might show the sprite walking, or exploding, or powering-up, etc.

A sprite editor usually includes a zoomed-in view of the image pixels and the ability to manipulate the pixels using simple tools (sometimes just clicking on pixels). A sprite editor has a sense of an array of images corresponding to the sequence of frames in an animation. A useful feature in an editor is a preview, which animates the sequence of images at adjustable rates.

An online example is Piskel: http://www.piskelapp.com

You got lots of practice with pixel-based digital images during the fern assignments. If you want a bit more, take a look at this document:

http://people.cs.clemson.edu/~dhouse/courses/405/notes/pixmaps-rgb.pdf

One thing to pay attention to is the idea of an alpha channel, which controls transparency for parts of the sprite image that should let the background through. You do not want a character to always be surrounded by a black or white background square, and transparency allows only the character part of an image to be drawn.

Qt Capabilities

Your team should spend some time seeing how Qt might support such an app. For example, look at drawing an image on the screen
https://doc.qt.io/qt-6.2/qtwidgets-painting-imagecomposition-example.html

and using mouse events https://doc.qt.io/qt-6/qtwidgets-widgets-scribble-example.html

Note that the scribble example is not really a great example of a sprite editor - it is a drawing program.

Stay away from Qt Quick examples, they are based on an internal scripting language approach. Similarly, some of you may be tempted by the Qt Graphics Scene. It is not really a good match to drawing a scaled up image.

Application Requirements

The application must be able to:

1. Set the size in pixels of the sprite. This can be bounded by a minimum, maximum, and by jumps in size if desired (historically, sprite rendering was most efficient for graphics cards when sized at powers of 2). Or it can be more continuous. Keep in mind that you are not making a paint program, so if you are drawing on 500x500 images, you are probably aimed at not enough sprite editor and too much paint program.
2. Adjust the number of frames for the sprite animation - this can be incremental, like "add a frame" and "delete a frame".
3. Allow the user to modify the pixels of a sprite. The most basic form is clicking on pixels. You should also allow the user to draw while moving or dragging the mouse.
4. Erase pixels.
5. Adjust the color being drawn, including transparency.
6. Provide a preview of the sprite animation cycle.
    1. The user should be able to adjust the frames per second of playback.
    2. There should be an option of seeing the playback at the actual size of the sprite (even when super tiny).
7. Save and load a project. All values in the project file should be represented by a json data structure. You should learn enough about json to decide how to do this (for example, see how to represent arrays in json). Even though enormously inefficient, the pixels should be represented by textual values of the red, green, blue, and alpha components. Your planning doc should have a minimal example of a sprite in this format (for example, a 1 frame 2x2 pixel image). Use a "sprite sheet project" .ssp extension when saving. You only need to worry about reading files produced by your system. Your team should research C++ or Qt libraries to assist in creating, reading, and writing json.

All other features are up to you. Additional features are needed for full points. See your research into other sprite editor products for ideas on features. Something like 2 or 3 additional features is about right.

Do not lose sight of the goal of this application - to make sprites. Some groups get carried away and implement some tiny Mario game engine in their editor, which is impressive, but does not count as a useful feature for a sprite editor. A sprite editor is also not Photoshop. There should be a focus on pixels and the relationship between frames of the animation.