3XA3 Development Plan
Liquid Rescaling

Team 35 - Marshiel
Lab 03
Harsh Shah
Daniel Wolff
Marlee Roth

September 30, 2017

1 Team Meeting Plan

To make this project successful the team is going to have to meet outside of lab hours to have sufficient time to finish this project. The team will meet after the lab on Friday for 1-2 hours, depending on the situation, to further discuss current progress and what needs to accomplished for next week. If at any given point in time if the team needs more meetings the team will meet on Wednesday in the evening at 6pm. To make the process of developing this project more efficient the team shall finish majority of communication heavy work during lab times (i.e. lab 03 Wednesdays 8:30am - 10:20am and Fridays 2:30pm - 4:30pm).

2 Team Meeting Plan

The team is using a Facebook messenger, via a group chat made just for talking about the project. The uses the group chat to give quick progress reports, new findings and quick questions. More intricate things related to documents shall be communicated via Google Docs. Google Docs shall be used to write up rough copies of documents. Google Drive shall be used to communicate various files (including documents, code and other miscellaneous files). All finalised documents and code shall be uploaded to Git.

3 Team Member Roles

- Code Leader - Daniel Wolff
- Code Writer(s) - Daniel Wolff, Harsh Shah, Marlee Roth
- Code Testing Expert(s) - Harsh Shah, Daniel Wolff, Marlee Roth
- Code Testing Manager - Harsh Shah
- Documentation Expert(s) - Harsh Shah, Daniel Wolff, Marlee Roth
- Documentation Leader - Harsh Shah
- Project Development Plan Expert(s) - Marlee Roth, Daniel Wolff, Harsh Shah
- Team Communication Manager - Marlee Roth
- Proof of Concept Expert(s) - Marlee Roth, Daniel Wolff, Harsh Shah
- Demonstration Expert(s) - Marlee Roth, Daniel Wolff, Harsh Shah
- Demonstration Leader - Marlee Roth

4 Git-Flow Work Plan

Since this project is being developed by a small team, a simplified version of Vincent Driessen's *A Successful Git Branching Model*[Dri10] will be used. The repository will be centred around an origin that each developer can push and pull from. Each developer will be working on separate features, so the need to push and pull from one another is unnecessary.
The origin will consist of only two branches: *master* and *develop*. The master branch will only consist of change-sets with completed features, while the *develop* branch will only consist of change-sets dealing with the merging of features, as well as the fixing of any bugs. To develop a feature, one must branch from the *develop* branch, and then merge back after the feature is completed. Once a feature has been successfully merged to the *develop* branch (and all merge issues and bugs have been fixed), the most recent *develop* change-set will be merged into the master branch, where a tag will identify the new version of the application.
It is worth noting that all of our documentation for this project (including this document) will be contained within the origin of our repository. Due to these documents not technically counting as "features", they will be pushed and pulled directly from the *master* branch.
Milestones will be managed in the project documentation instead of within Git.

5 Proof of Concept Demonstration Plan

It is reasonable to create an application that performs a seam carving algorithm in the next two months. Although the seam carving is a complicated algorithm that requires difficult math and complicated logic, a C++ library exists on Wikidot [Bal] which will be used in the application. This library is available for free download online for all operating systems. The only potential issue that this library brings is that it is a C++ library - a language that the team is not familiar with. Fortunately, C++ is similar to other languages that the team members have used. This application will only be usable on Windows operating systems, as it has been decided to use Visual Studio (2017) IDE to develop a Windows Desktop Application. As for testing, since the seam carving is a liquid rescaling algorithm, multiple tests can be run to ensure image quality and its general ability to rescale images properly. If time becomes a factor in the development of the application, the scope of the project can always be lessened. As of now the plan is to attempt to successfully rescale images as well as to accomplish the removal and preservation of objects within images, but if need be it can just be an application that scales images. Overall, the seam carving algorithm is complicated and novel idea but with the use of the Wikidot library it is feasible to create an application using it with the next two months.

6 Technology

This project will be developed in C++ with CLI Support packages, using the Visual Studio (2017) IDE. This combination allows the user interface of the application to be developed with ease, leaving the majority of development power to be allocated to ensuring the programs functionality is flawless. Additionally, Visual Studio comes with a built in Git interface that avoids the use of tedious command line operations.
A testing framework will not be used for this project, as the core functionality of this application cannot be determined as correct or incorrect through any sort of comparison - it is mainly determined by the human eye ("is the resulting image of acceptable quality or not?"). In light of this, all testing will be done manually with a set of test images that will be designed to challenge the functionality of the application.
ShareLatex and the Sublime IDE will be used to generate the LaTeX documents for this project.

7 Coding Style

The C++ coding style that will be used will follow closely to the Google C++ Style Guide [Goo], however the assistance of Visual Studio's suggestions and critiques will be accepted while developing. In addition to these coding standards, a custom standard shall be defined for naming Windows Form Control variables (buttons, panels, etc); buttons shall be named "btnButtonName", panels shall

be named “pnlPanelName”, and so on. To define it generally, each variable referencing a control will have two or more lowercase letters prepending the control name that will describe the type of control being used (much like Hungarian notation, but more descriptive). For example, a button that submits a form can be named “btnSubmitForm”.

8 Project Schedule

The PDF of the schedule can be found in our git repository. The PDF can also be found at the following link:
https://drive.google.com/open?id=0BzEqx9gKJQxob216Vi04bWl0SWc

References

[Bal] Carlo Baldassi. Liquid rescale library.

[Dri10] Vincent Driessen. A successful git branching model, 2010.

[Goo] Google. Google c++ style guide.