Memorandum

To: Bruce Bolden
From: Adrian Beehner
Subject: Final Development Sprint Report for *Goofy Lights Editor* Project - Team 3
Date: 5/5/17

Introduction

The following document describes the results and work done by each team member of the Final Sprint for the *Goofy Lights Editor* Project from Team 3 (Java Team). The following appendices go into detail in regards to the final development Sprint on the project.

Overiew

In Appendix A I outlined the work done by all the teammates/displayed some sample code and output from our project during the final sprint. I provided details into what task each teammate was doing. The sample code (or sample output) was provided to provide insight on the work they performed on the final development sprint. I also briefly discussed the state of the GUI for the final sprint. Then in Appendix B I listed the revised timeline for the project, taking into regard the now completed final development sprint. Finally in Appendix C I discussed any complications/additional comments for the final sprint and discussed the team's accomplishments and closing thoughts on the project.

Summary

The final development sprint was another successful and final sprint to the project. Each team member contributed effectivey towards this sprint, to begin with, I created the Quicking Coloring Nodes functionality, allowing nodes to be color as the cursor moves by by holding Ctrl, while also fixing some bugs. Andrew Butler worked on creating the Fill Nodes functionality, allowing users to color large gaps in Grid Editor by holding down Shift. Robert Stewart created the Start and Duration Update functionality, allowing the user to enter a start time for frames as well as a duration time. Megan Phelan implemented a Checking for Changes Since Last Save functionality, which prompts the user to save before exiting the program if changes have been made to file. Animesh Pattanayak edited the Button Grid layout and also created a Fill Grid functionality, allowing a user to color the entire grid editor with a current selected color. Joesph Leister provided a cancel option when trying to close the program, with or without unsaved changes. Lastly Seth Forrest finalized the design specification and edited the Button Grid for adding the Play, Stop, and Pause buttons to it. All work further discussed in Appendix B. A revised timeline was added, reflecting progess made in final sprint. Then finally the final sprint made a great last hurdle, our of our core goals were accomplished and some bugs and additional features were implemented. This was an excellent team to work with and a successful project overall.

Appendicies

Appendicies A, B, and C are attached below, discussing *Sample Code/Work Done*, *Revised Timeline*, and *Additional Comments* respectively. I have also attached a table of contents.

Final Development Sprint Report for *Goofy Lights Editor* Project - Team 3

Adrian Beehner

Contents

A Sample Code/Work Done

The work performed by each team member is discussed below. The various work done included creating a quick cell coloring functionality and bug fixes, adding a fill Nodes functionality, editing the duration and start time of each frame, adding functionality to check for changes since last saving when exiting, creating a button grid and fill grid functionality, adding a cancel option, and finally editing buttons and finalizing Design Specification. These sections may not completely represent all the work a particular team member performed.

A.1 Quick Coloring Nodes Functionality - Adrian Beehner

I wrote functionality for quickly being able to color nodes in the Grid Editor. This feature works by the user selecting a color, and when holding down the Ctrl button, the mouse will color any node the cursor moves over, thus making it very quick to color nodes, versus having to click each node one by one. I also fixed various bugs such as an issue with the dimensions dialog box not accepting any dimension that was not a square dimension. Sample code of this is shown below in Figure 1.

```
1262                    btnGrid[r][c].addMouseMotionListener(new NodeMotionListener(r, c){
1263
1264                            @Override
1265                            public void mouseMoved(MouseEvent arg1) {
1266
1267                                    if(arg1.isControlDown())
1268                                    {
1269                                            color = colorChooser.getColor();
1270                                            btnGrid[this.getRow()][this.getCol()].setOpaque(true);
1271                                            btnGrid[this.getRow()][this.getCol()].setBackground(color);
1272                                            frames.get(currentFrame).setNodeColor(this.getRow(), this.getCol(), color);
1273                                            // redraw the grid
1274                                            initGrid();
1275                                            createNodeButtonEventHandlers();
1276                                    }
1277                            }
1278
1279                    });
```

Figure 1: Sample Code of Quick Color Nodes Functionality

A.2 Fill Nodes Functionality - Andrew Butler

Andrew Butler wrote the functionality of filling nodes quickly, by having the user hold Shift and select and beginning and ending area on the grid editor to fill a square area with the selected color. This feature provided a quick way to color specific large areas of the grid editor without having to color the entire grid editor. Sample code of this functionality is shown in Figure 2 below.

```
1225                                    @Override
1226                                    public void mouseClicked(MouseEvent arg0) {
1227                                            if(arg0.isShiftDown()&&lastPressedRow>-1){
1228                                                    int maxRow=this.getRow()>lastPressedRow ? this.getRow() : lastPressedRow;
1229                                                    int minRow=this.getRow()<lastPressedRow ? this.getRow() : lastPressedRow;
1230                                                    int maxCol=this.getCol()>lastPressedCol ? this.getCol() : lastPressedCol;
1231                                                    int minCol=this.getCol()<lastPressedCol ? this.getCol() : lastPressedCol;
1232                                                    fillNodes(colorChooser.getColor(), minRow, minCol, maxRow, maxCol);
1233                                            }
1234                                            else{
1235                                                    color = colorChooser.getColor();
1236                                                    btnGrid[this.getRow()][this.getCol()].setOpaque(true);
1237                                                    btnGrid[this.getRow()][this.getCol()].setBackground(color);
1238                                                    frames.get(currentFrame).setNodeColor(this.getRow(), this.getCol(), color);
1239                                                    // redraw the grid
1240                                                    initGrid();
1241                                                    createNodeButtonEventHandlers();
1242                                            }
1243                                            lastPressedRow=this.getRow();
1244                                            lastPressedCol=this.getCol();
1245                                    }
```

Figure 2: Sample Code of Fill Nodes functionality

A.3 Start and Duration Update Functionality - Robert Stewart

Robert Stewart created the functionality of editing the duration and start times for each frame. The idea was to have the data values be updated when the fields for enter the information was updated. Error handling was also implemented, so a frame's start time could not be equal or less than the previous frame's start time, this is displayed by the text field displaying a red value. All frames start times were also updated upon updating a single start time for this functionality. Sample code of this functionality is shown in Figure 3 below.

```
405         private void UpdateStartTime(){
406                 boolean validInput = EnsureStringIsInt(startTimeField);
407
408                 if(validInput == true){
409                         int newStartTime = newStartTime = Integer.parseInt(startTimeField.getText());
410
411                         int previousStartTime = -1;
412                         int nextStartTime = 0;
413                         if(currentFrame != 0){
414                                 previousStartTime = frames.get(currentFrame-1).getStartingTime();
415                         }
416                         if(currentFrame != frames.size()-1){
417                                 nextStartTime = frames.get(currentFrame+1).getStartingTime();
418                         }
419
420                         if((newStartTime >= nextStartTime && currentFrame != frames.size()-1) || newStartTime <= previousStartTime ||
421                                         (currentFrame == 0 && newStartTime != 0)){
422                                 startTimeLabel.setForeground(Color.RED);
423                                 return;
424                         }
```

Figure 3: Sample Code Start and Duration Update Functionality

A.4 Checking for Changes Since Last Save - Megan Phelan

Megan Phelan implemented the functionality of checking for changes since the last save of the current tan file being worked on. The use was that when a user attempts to close the program, the program has actucally saved a copy of the configuration (frames) of the grid editor at the last save, and checks if the past configuration is the same as the current one of the gird editor. If so, the program closes, but if not, a dialog box pop ups, asking the user if they wish to save before exiting. Sample code of this functionality is shown below in Figure 4.

```
884         ////////////////////////////////////////////////////
885         // Method to check if frames have changed prior to
886         // last save before closing.
887         ////////////////////////////////////////////////////
888         private void grid_windowClosing(WindowEvent e) {
889                         if(frameChanged()) {
890                                 //ask user if they want to save before exit
891                                 int result = JOptionPane.showConfirmDialog(null, "Do you want to save before exit?", "Save?", JOptionPane.Y
892                                 if(result == JOptionPane.YES_OPTION) {
893                                         // check for file name, otherwise open save as dialog
894                                         if(currentFile != null) {
895                                                 try {
896                                                         File tanSaveFile = openFileChooser.getSelectedFile();
897                                                         if(!tanSaveFile.getName().endsWith(".tan")) {
898                                                                 String path = tanSaveFile.getAbsolutePath() + ".tan";
899                                                                 setTitle("GoofyGlasses Editor " + path);
900                                                                 File newSaveFile = new File(path);
901                                                                 TanFile.writeFile(newSaveFile, frames);
902                                                         }
903                                                         else {
904                                                                 TanFile.writeFile(tanSaveFile, frames);
905                                                                 currentFile = tanSaveFile.getAbsolutePath();
906                                                                 setTitle("GoofyGlasses Editor " + currentFile);
```

Figure 4: Sample Code of Checking for Changes Since Last Save Functionality

A.5 Button Grid and Fill Grid Button - Animesh Pattanayak

Animesh Pattanayak edited/modified the button grid to have an additional button. The additional button was called "Fill Grid", which would fill the entire grid editor with the currently selected color, the fill grid functionality itself was also created by Animesh Pattanayak. The button grid now appears as a 2x3 grid, with each button taking up a cell, and the direction button for shifting the nodes looking similar to the layout found to the arrows keys found a keyboard This provides a much better layout and design for the buttons overall. Sample Code of this fill grid functionality is shown below in Figure 5.

A.6 Cancel Option - Joseph Leister

Joseph Leister provided a cancel option upon trying to exit the program without saving after changes and also providing a default option for canceling if the user tries to exit the program (with or without changes). This further enhanced the functionality of the "Checking

```
814                 fillGrid.addActionListener(new ActionListener(){
815                         @Override
816                         public void actionPerformed(ActionEvent e){
817                         //Fill entire grid with the selected color
818                                 for(int r = 0; r < gridRows; r++){
819                                         for(int c = 0; c < gridCols; c++){
820
821                                                 color = colorChooser.getColor();
822
823                                                 btnGrid[r][c].setOpaque(true);
824                                                 btnGrid[r][c].setBackground(color);
825                                                 frames.get(currentFrame).setNodeColor(r, c, color);
826                                         }
827                                 }
828                                 // redraw the grid
829                                 initGrid();
830                                 createNodeButtonEventHandlers();
831                         }
832                 });
```

Figure 5: Sample Code of Fill Grid Functionality

for changes since last save" functionality that Megan Phelan had. Sample code of the edit/added functionality is shown below in Figure 6.

```
137     -           setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    137 +           setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
138 138
139 139 +           // Quick-Fix for now, the "+" and "-" Frame Button display over the rest of graph if too small
140 140             setBounds(100, 100, 1000, 800);
    ❄   @@ -878,7 +878,7 @@ private boolean isInteger(String n) {
878 878         private void grid_windowClosing(WindowEvent e) {
879 879                 if(frameChanged()) {
880 880                         //ask user if they want to save before exit
881     -                   int result = JOptionPane.showConfirmDialog(null, "Do you want to save before exit?", "Save?", JOptic
    881 +                   int result = JOptionPane.showConfirmDialog(null, "Do you want to save before exit?", "Save?", JOptic
882 882                         if(result == JOptionPane.YES_OPTION) {
883 883                                 // check for file name, otherwise open save as dialog
884 884                                 if(currentFile != null) {
    ❄   @@ -904,8 +904,14 @@ private void grid_windowClosing(WindowEvent e) {
904 904                                         saveAsDialog(mntmSaveAs);
905 905                                 }
906 906                         }
    907 +                   else if(result == JOptionPane.CANCEL_OPTION) {
    908 +                           return;
    909 +                   }
    910 +                   else if(result == JOptionPane.NO_OPTION){
    911 +                           System.exit(0);
    912 +                   }
907 913                 }
908     -               System.exit(0);
```

Figure 6: Sample Code of Cancel Option

A.7 Finalizing Design Specification and Editing Button Grid - Seth Forrest

Seth Forrest worked on finalizing the Design Specification (as well as putting all the documentation together), and also edit the button grid originally worked on by Animesh Pattanayak. The "Play", "Stop", and "Pause" buttons are also added to the button grid, creating a 3x3 grid, instead of the original 2x3. Some irrelevant code was also removed by Seth Forrest, as the code was no longer relevant nor provided any functionality to the program. Sample code of the edit/added functionality is shown below in Figure 7.

```
219   -   buttonPanel.setLayout(new GridLayout(2,3));
220   -   buttonPanel.setPreferredSize(new Dimension(60,60));
219   +   buttonPanel.setLayout(new GridLayout(3,3));
220   +   buttonPanel.setPreferredSize(new Dimension(90,120));
221   +
222   +   buttonPanel.add(new JButton("Play"));
223   +   buttonPanel.add(new JButton("Stop"));
224   +   buttonPanel.add(new JButton("Pause"));
225   +
226   +   //createAddFrameEventHandler(frameActionPanel.getComponent(0));
227   +   createPlayEventHandler(buttonPanel.getComponent(0));
228   +   createStopEventHandler(buttonPanel.getComponent(1));
229   +   createPauseEventHandler(buttonPanel.getComponent(2));
230   +
313   -   contentPane.add(frameEditPanel, BorderLayout.NORTH);
314   -
315   -   // Add a "+" Button to add a frame (Added here as button should never move or change)
316   -   frameActionPanel = new JPanel();
317   -
318   -   frameEditPanel.add(frameActionPanel);
```

Figure 7: Sample Code of Edited Button Grid

A.8 Current GUI - Whole Team

The current GUI in regards to the final sprint is shown below in Figure 8, showing some of the added functionality and work done by the team. This sprint doesn't provide much of a visual change when compares to Spring 3 besides the newly made button grid, and some tweaks to the color chooser, as well as the duration and start time boxes for the frames. However, a lot of functionality was added and expanded on as mentioned above.

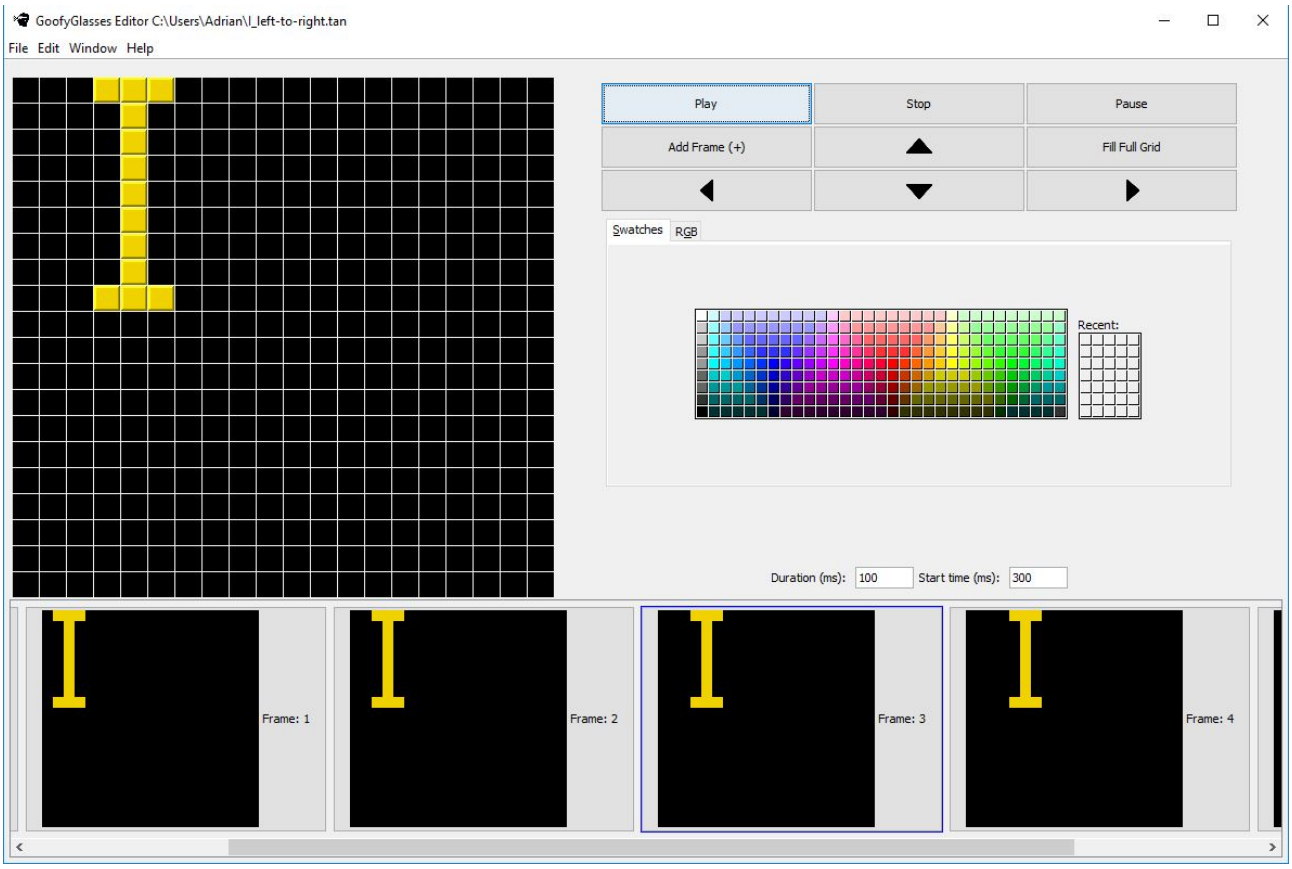

Figure 8: Sample Output of GUI

B Revised Timeline

The following schedule/timeline below outlines the goals met at these dates. This schedule is reflected off of the work performed during the final sprint, regarding all the work done as mentioned in Appendix A. Refer to Figure 9 below for the updated timeline.

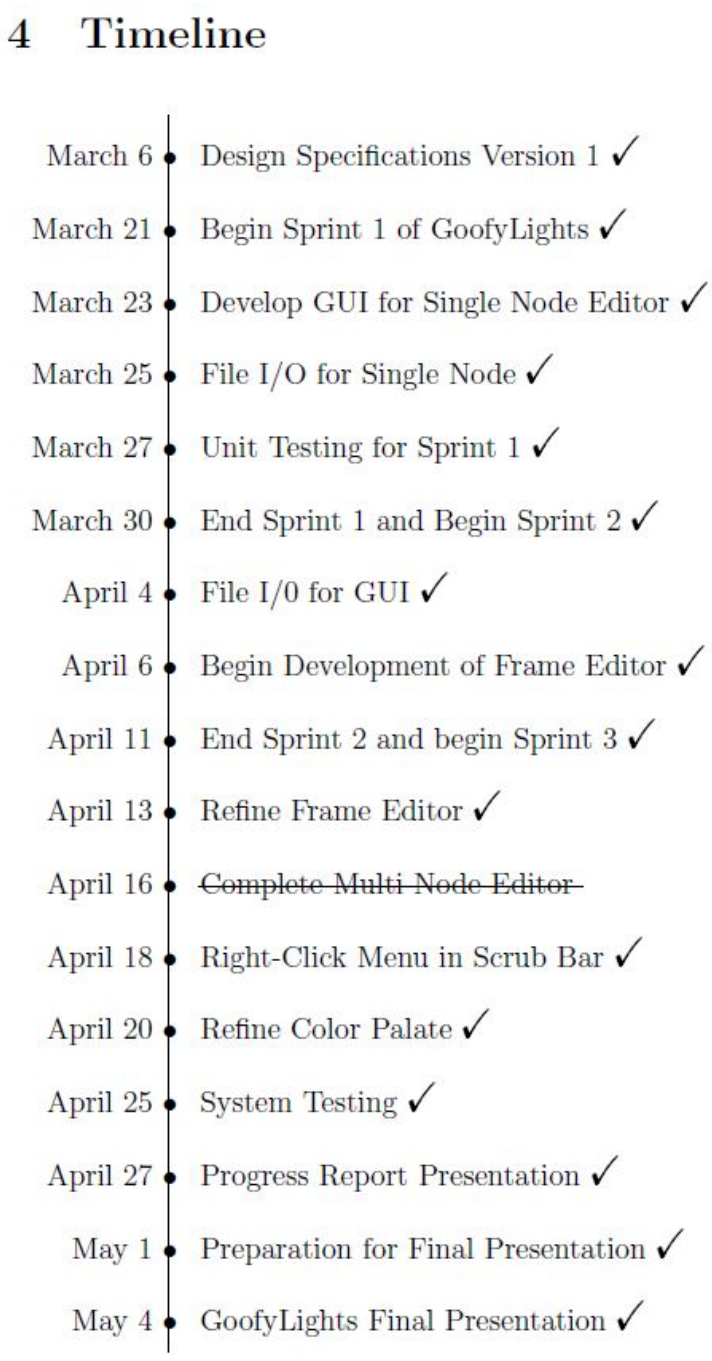

Figure 9: Revised Timeline - Final Sprint

C Additional Comments

The final sprint was really successful, everyone put in all the necessary time and work for this final and provided a great end product. Almost anything of missing functionality or issue was fixed in this final sprint, some features were more finely tuned and others were added given the additional amount of time. The amount of work load was definitely lighter for this last sprint as most of the work for the program had been completed by Sprint 3, so being ahead of schedule definitely provided a very stress-free environment for the final sprint. This in turn allowed for some testing and bug fixing as well. Our final meeting was helpful in establishing any last minute items to attend to as well. Our group made the smart decision to not try to overload with attempting to add a ton of additional features for the final sprint, as we realized how quickly things could become difficult and time consuming if we added all the additional features we wanted. Slack provided great communication once again and everything went smoothly as ever in the project. I believe our team accomplished our task effectively and efficiently while working together in a timely and professional manner. The software we developed was well made and has some great features that makes it unique. This was an excellent team to be a part of, as everyone did just an amazing job and usually blew me away with how well they were at communicating and getting tasks done. I was very fortunate to have such a team and this was a great learning experience as well, as the team dynamic was far different than anything else so far in the CS curriculum (at least in my own experience). My teammates are all great individuals and hard workers, and it'd be a pleasure to work with any of them again in the future.