# A note on these directions

For each module project you will submit a document that identifies the location (on GitHub) of your project code; documentation relating to the work you have completed, or of incomplete work; and answer to some short reflective questions on your development process. I have provided more detailed directions in this document, as it is the first module project; future modules will have fewer directions.

# **Module 1 – Project Code**

Use the space below to enter the full URL for your project code, hosted on GitHub. Please ensure that the URL you enter is properly hyperlinked (use the Insert>Hyperlink menu item). If you copy and paste the URL directly from a browser it should automatically be hyperlinked.

|  |
| --- |
| [*https://github.com/nic-dgl213-f21/dgl213-module1-ItsJustJarvis*](https://github.com/nic-dgl213-f21/dgl213-module1-ItsJustJarvis) |

# **Module 1 – Documentation**

**Use the spaces below to summarize: 1. The hours spent on the \*coding\* portion of this module project; 2. Major changes you have made to the template code; 3. Parts of your code that do not work as expected, or that are incomplete. You may write using bullet points, rather than complete paragraphs, as long as each bullet point is written as a complete sentence.**

|  |  |
| --- | --- |
| **Approximate hours** | The coding portion of this assignment took me somewhere between 2 and 3 hours. I was able to implement the changes I wanted easily enough, but I took some time to improve the overall aesthetic and user interface of the game. I wanted it to be more appealing to the player, as I am a gamer myself and appreciate a good user interface for games. |
| **Major code changes** | I have completed my implementation (with some minor changes) according to Solution 2 for Problem 1, and Solution 1 for Problem 2 outlined in Milestone 1.  Adjustments made to **index.html** code:   * I have taken the liberty to create a better layout structure using new container elements to improve overall user experience. * The user interface is now sectioned based on use-case with the colour changing buttons, gameplay changing buttons (restart, and undo), as well as a new scoreboard area all in their own sections. * The page is much more responsive now, after implementing more flexbox and responsive units. * I have adjusted the CSS styling properties of the elements to improve the user interface’s appearance. * I have added a new area to hold the final results of a game, as well as the high score if the player wants to attempt a better score on the same grid using restart.   Adjustments made to **floodFill.js** functions:   * Function **startGame()** now initializes my scoring variables to a value of 0 at the start of each game. * Function **render()** now updates my scoreboard UI elements to keep a current value for points and number of clicks. * Function **updateGridAt()** now calls two new functions I have created to update the player score and check for win conditions (seen below). * Function **floodFill()** now calls a new function to track the number of cell changes after a color has changed (seen below). * Function **restart()** now adjusts the contents in the results section to show the most recent high score when a grid is restarted. * Function **gridClickHandler()** now increments the number of clicks taken within the grid. * Function **convertCartesiansToGrid()** now uses the canvas.clientWidth and canvas.clientHeight properties to improve click location tracking   New Additions to **floodFill.js**:   * New references have been added to UI elements including the undo button, and the scoreboard elements for number of clicks, points, and results/higscore. * New variables have been added to track values for **numberOfClicks, cellsChanged, playerScore,** and the **finalScore** * I have created a new const array **previousPointGains** to track the history of points gained for use in my undo function * A new function has been created named **updatePlayerPoints()** that calculates the players points based on 5 points per cell changed, and updates the current value in **playerPoints** before returning the **cellsChanged** variable to 0. Additionally, it pushes the most recent point gain to the **previousPointGains** array. * A new function has been created named **checkWinConditions()** that takes the grid as a parameter. It then finds the color value of the first grid cell and compares it to all consecutive cells in the grid. If a cell is found that does not match the function returns and exits. If all cells match, the players final score is calculated, and the results are output to the results section. If the player meets or surpasses the 200 point minimum, they have won. This function also calls on another new function to track the high scores for use in the case of restarting the same grid. * A new function has been created named **undoLastMove()** to implement the undo feature. It is called by the **undoClickHandler()** function, responding to **mousedown** events on the undo button. First it checks to ensure we do not undo past the initial grid state. It then pops off the last item in the **grids** array, adjusts the **playerPoints** value based on **previousPointGains**, decrements the **numberOfClicks**, and calls on render to render the last grid in the grids array. * A new event listener has been added to the undo button, as well as a new function **undoClickHandler()** that calls **undoLastMove()**. * A new function has been created named **trackCellChanges()** to track and increment the **cellChanges** variable. * A new function has been created named **trackHighScores()** that is used to compare high scores and keep a current value upon consecutive attempts of the same grid. |
| **Bugs or incomplete code** | I was able to remove any bugs present within the template code and squash any new ones that came up during my implementation. As it stands, it is complete according to the assignment requirements. I added some extras as well. If there are any bugs, I haven’t noticed I look forward to hearing from you. |

# Module 1 – Reflection

Use the spaces below to answer the questions posed. For each module project I will ask a series of module-specific questions that ask you to consider specific aspects of the module project implementation, and a series of general process questions that ask you to consider your working process throughout the module. The intent of these questions is to draw your attention to aspects of design and development that you may not have considered (the module-specific questions), and to provide you with an opportunity to consider how your work process and understanding has changed – and hopefully improved! – over the course of the semester.

## **Module-specific questions**

1. **A. Consider the final version of the Flood Fill game that you have submitted. Can the player conceivably win by playing only one move? Or perhaps with *no* moves? What are the conditions that would make this possible?**

|  |
| --- |
| With the final version I have completed in mind, there is no way for the player to conceivably win by playing only one move, or no moves. I would need to fundamentally change the entire scoring implementation, or the rendering of the grid, to allow this to be possible.  My scoring implementation requires the player to play the entire board until all cells are the same colour. The player gains 5 points for every cell that is changed on a click. Using the number of points gained throughout the game, as well as the number of clicks taken, a final score is produced. To win the player’s final score must be at least 200. I feel that this encourages the player to think carefully and make as big a move as possible on each click.  This scoring model relies on high scores and meeting a minimum threshold as the defining factors in whether a play session was successful or not. It leverages the restart function to allow the player to compete against themselves to get a better score on the same grid.  For this to be a game won with only one move the player would need to change the entire grid to one color with one click, which is not possible with this game’s implementation of 5 random colors. If it *were* somehow possible, they would achieve a score higher than they would be able to normally because (5\*81) / 1 = 405 (a score higher than I have been able to achieve thus far). But as I said the only way this would happen is if there were only 1 square a different colour than the rest of the grid. |

**B. Similarly, is it conceivable that the player may not be able to win the game at all? What are the conditions that would make this possible?**

|  |
| --- |
| With the way I have implemented the scoring system, the win condition may not always be met when you complete the grid. Near the end of my solution, I decided to add a minimum **finalScore** value that needed to be met to be considered as a win.  The game provides you with a minimum **finalScore** goal of **200 points** and tries to encourage you to complete the board in as few clicks as you can. This number is fairly easy to meet. What really determines whether you did well or not is how high your final score is, and even still the only way to compare this in its current implementation is to restart the same grid and attempt a better score. I think that this provides a competitive aspect, encouraging you to try and find better ways to play the same grid and beat your score.  So, as it stands, there definitely is a conceivable way the player may not win… but I do not think that would happen often. |

1. **Consider the final version of the Flood Fill game that you have submitted. In particular my solution to the score implementation task (as written in *green italicised text* in the milestone 1 document) contains some assumptions about the Flood Fill game – in particular, that all grids generated by the initialization algorithm are ‘equal’. This affects the player’s understanding of the score, since a high score may be the result of an ‘easy’ initial grid, rather than the result of player skill. Have you made any similar assumptions in your score implementation?**

|  |
| --- |
| I have made similar assumptions with my scoring implementation.  I decided to add the minimum **finalScore** of 200 later in my development. I decided that only relying on beating your own score wasn’t enough, and I needed to provide a way for the player to possibly lose to make things more interesting if they did not want to repeat the same grid. My implementation of this minimum requirement does operate under the same assumption that all grids are the same.  Setting a base minimum requirement for all grids is perhaps a little unfair, though I have not experienced a losing case thus far, unless on purpose. I do not think that this would occur very often unless the player is intentionally trying to lose.  I feel that the addition of the minimum score allows the player to feel a sense of success whether they decide to attempt a better score or not, which improves the overall fun of the game. |

## Process questions

1. **What did you learn in this module that will be most valuable to you as you progress through your education? What was most valuable to your future career?**

|  |
| --- |
| Working through a recursive equation like this has increased my overall understanding of how they function significantly. I struggled in the past to understand how recursion works, but at the start of this semester I had a moment of clarity, and it all clicked. I feel that this is going to be extremely valuable to my continued studies and eventually my career, as I know the importance of recursion in programming.  Additionally, I learned a lot about the canvas and how to mathematically operate within a grid. This is something I think will be greatly helpful to my interests in working in the video game industry. I know that a lot of web-based games are built on these concepts, and I look forward to learning more. |

1. **What was most challenging aspect of this module? What strategies did you use to overcome these challenges?**

|  |
| --- |
| It was challenging at first to parse through the floodfill equation and make sense of it. I feel that much of that is due to having a break from these concepts over the summer. I did not have as much time as I would have liked to practise this summer and it took a bit of refresher before I felt 100% comfortable again. Even still there are things I still need practise with.  Honestly, the most challenging aspects of this module are in sync with the most valuable aspects I listed above. I struggled a lot with recursion prior to this, and I had zero experience working with the canvas. I feel like that is what made this module valuable to me. I was able to explore, research and overcome problems that I had little experience working with before. And as I stated above, I gained a lot of clarity.  In order to overcome the challenging aspects, I applied myself and took the time to read through documentation. I researched concepts I was less familiar with, and I followed along closely with the lesson videos. I have performed quite well in my studies thus far and I attribute that to a strong work ethic and a determination to find solutions. I always work hard at finding the help I need to grasp concepts better, something I pride myself on. |