ECE 364 Project Phase 2
Flow

Due: December 2-6, 2013

By completing this project with a passing grade you will receive credit for CO2, CO3 and C04, C06 and C07

Instructions

- Work in your Lab13 directory.
- There are no files to copy for this lab.
- Remember to add and commit all files to SVN. Also remember to use your header file to start all scripts. **We will grade the version of the file that is in SVN!**
- Name and spell your scripts **exactly as instructed by this handout**. When you are required to generate output, make sure it matches the specifications exactly. Your scripts may be graded by a computer.
- This is the second of two phases for the course project for ECE 364. Each phase is worth 9% of your overall course grade.
- This is an **individual** project. All submissions will be checked for plagiarism using an online service.
- As before, the following websites may be very useful for learning TkInter and Kivy, respectively: `http://effbot.org/tkinterbook/` and `http://kivy.org`

1 Introduction

Now that you have completed a bare-bones Flow game GUI for Phase I of the final project, you will be asked to build upon your code in order to develop a version of Flow that is very much like the commercially available game (`http://moh97.us/flow/`). The remainder of this document will outline both required functionality and bonus functionality.

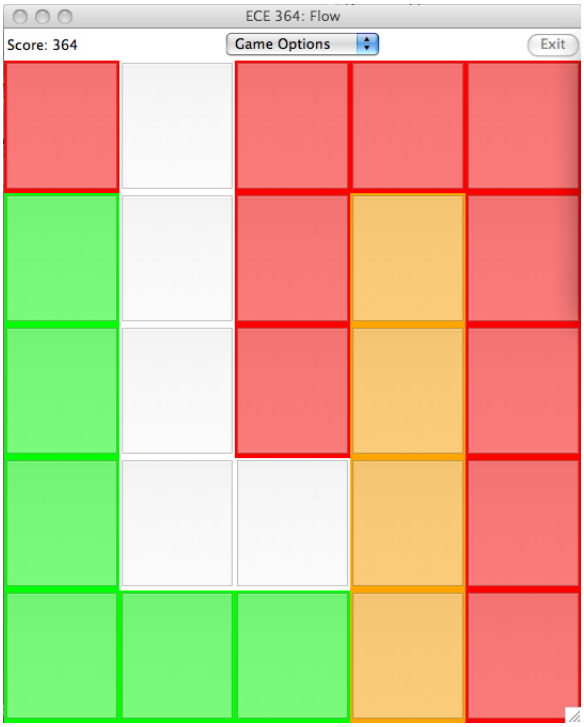

Figure 1: Sample Phase 2 Flow Game UI (Tkinter)

2 Required Functionality

- A player should be able to "backtrack" along the path that he/she is creating between two terminals without having to "cancel" the entire path and start from the beginning. That is, if a player has created a path that extends 3 nodes away from a terminal, that player should be able to click on a node between the "present" node and the terminal of origin in order to make this clicked-on node the new present node. The previous present node and all nodes that were between the old "farthest" node and the new present node should no longer belong to the flow after such a click.

- If a player is constructing a path between a pair of terminals, then clicks on a terminal that is not part of the pair, such a click should select this new terminal as the current terminal. The path corresponding to the first terminal pair should remain intact, and clicking on any node corresponding to this old path should select said node as the current node, resetting all nodes that extend past the present node to zero and allowing the user to continue building this path from the selected node. That is to say, the player should be able to leave a path partially finished while he/she works on another path, then resume working on the old path from where he/she left off.

- The player should be able to complete a game board by connecting all terminal pairs without crossing "flows" and filling every node on the game board in the process. Completion of a game board should

result in a pop-up success message that asks the player if he/she would like to 1) play a new board or 2) stop playing.

- Your Flow game should be able to play a number of different game boards during one playing session. That is, when a player has successfully completed a game board, the player should be given the options to stop playing or to continue playing on a new board. Your program should be able to handle any board that is M columns wide and N rows tall, i.e. any rectangular board. Additionally, the player should be able to select the option "New Game" while still playing a not completed board (see UI diagram for clarification). The course staff will be making 5-10 input boards available in the your Lab13 directory shortly, and you do not need to write any additional boards yourself (unless you work on the "Walls" bonus feature specified later in the document).

- You should implement save and load features so that a player may save a partially completed game and return to playing at a later time. Access to these features must be associated with the menu bar.

- You should implement a scoring system of some sort. You may construct this scoring system however you like within the following confines: the score must depend on 1)the number of paths completed and 2) the total number of clicks. That is, the highest possible score must correspond to full board completion in the smallest possible number of clicks.

- Numbers corresponding to terminals / nodes should not be visible in the game UI.

Completion and rigorous demonstration of these functions will be sufficient for full credit in Phase 2 of the project. **Keep in mind that all functionality required for Phase 1 will still be required to work when you demonstrate Phase 2.**

3 Bonus Features

As not all of the bonus features listed below are equally difficult to implement, we have instituted a tier system for judging bonus features that you may add to your Flow game. "Easy" tier bonus features are worth up to an additional 5% of your total Phase 2 grade, "Medium" tier features are worth up to 10%, and "Hard" tier features are worth up to 15%, though the course staff reserves the sole discretion for determining how much credit your extra features are worth. Important notes:

- Bonus features are eligible for partial credit, i.e making significant progress toward completion of a bonus feature will likely merit some bonus points. However, note that this means that total, thorough completion of a bonus feature is required to receive all of the possible points for the feature.

- **You may implement additional features that are not listed here.** You are eligible to receive bonus for such features, though in order to be eligible **you must get approval from the course staff for your feature.**

- You are encouraged to work on multiple bonus features, but for the sake of your time be aware that the **maximum** grade that a student can achieve for Phase 2 of the project is **125%.**

3.1 Easy Tier

- **Update your Flow GUI**, i.e. add a "skin" or other display-related feature. The standard UI, particularly in Tkinter, looks fairly bland and could use some more visual appeal.

- **Give the player a hint**, i.e. show the correct solution to one full "flow". This function should be bound to a button that can be found in the menu bar.

3.2 Medium Tier

- **Timed Play / Scoring**, i.e. make time a component of gameplay. Perhaps time can be incorporated into your scoring system or games can only be completed successfully if the player finishes in a specified amount of time. **Note:** A "game timeout" feature on its own can only earn up to 6%. A full 10% bonus requires incorporation of time into the scoring system or some other facet of gameplay.

- **Replay Mode / Play Solution**, i.e. allow the player to watch a replay of his/or her solution to the game board OR allow the player to view a node-by-node solution to the game board if the player has not completed the game. Note that this bonus feature does not require the game to "auto-play" the board. Boards can be pre-solved, and "viewing a solution" can be akin to "replaying" this pre-solved board.

- **Mobile Readiness (Kivy Only)**, i.e. demonstrate that your game is ready to be played on a mobile device as of the time of your demonstration. We will likely require you to demonstrate gameplay on an emulator. Please inform the course staff if you intend to demonstrate this feature.

3.3 Hard Tier

- **Walls:** Normally, any free node that is cardinally adjacent to the current node should be able to be selected as the next node. However, if there is a "wall" between nodes, no path can pass from one node to the other. As such, this feature provides an added difficulty for the player since his path options are further limited. Walls should be specified in the input file (possibly by a vertical bar in between numbers), which means that you will have to construct an input file to demonstrate this feature. Creating an input file should be very easy compared to the implementation of the actual "wall" feature. **Note:** It is very likely that you will need to import the PIL (Python Image Library) module to implement this feature in Tkinter.

- **Directional Paths:** It is clear that for a 3 x 3 board with only one pair of terminals there exists more than one solution that fills the entire board. But how can we distinguish these different solutions if there is no way to specify the path that the player has taken? For this bonus, find a way to visually specify the direction/order of each path for a general input board in a manner similar to the commercial version of Flow. **Note:** It is very likely that you will need to import the PIL (Python Image Library) module to implement this feature in Tkinter.

- **Autoplay:** Construct a mode (accessible in the menu bar) that solves the current board. **This feature cannot just replay a pre-solved board!** The solution should appear on the screen one node at a time. Keep in mind that this is likely very difficult, and may be eligible for more than the standard "Hard" bonus depending on how fast / complete / bug-free your implementation is.

4 Demonstration

Just like Phase 1, the grading of your Phase 2 Flow game will be demonstration-based. **You may come in to demonstrate your game during any office hour period (including Wednesday lab times) during the week after Thanksgiving break, December 2 through December 6.** You may also request to meet with the course staff to demonstrate at a different time, though we do not guarantee any availability outside of these office hours. With this in mind, if you would like to demonstrate at a time outside of office hours you should make arrangements as soon as possible. Be aware that no demonstrations may take place after 5:00PM on Friday, December 6.

Again, please do not hesitate to contact the course staff with questions about this document or about the final project in general. We are more than willing to help you with your game, and in fact are eager help you extend your implementation and see how robust your game can become. Good luck and have fun!