ECE 364 Project Phase 1
Flow

Due: November 19, 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 first 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.
- The following website may be very useful for learning TkInter: `http://effbot.org/tkinterbook/`

1 Introduction

```
MODULE NAME:            flow.py
INPUT:                  N/A
OUTPUT:                 N/A
# OF ARGUMENTS:         1
ARGUMENTS:              Configuration file name
RETURN CODE:            0 if successful, 1 otherwise
```

Your goal for this course project is to develop a Python version of the popular mobile-based game Flow. An image of a sample version of this game is provided below, and the link `http://moh97.us/flow/` provides an example flow game on the web. Please note that the figure below is a representation of what your Flow game might look like after completion of Phase I - this interface is still fairly raw, and is not what we anticipate finished Phase II projects will look like.

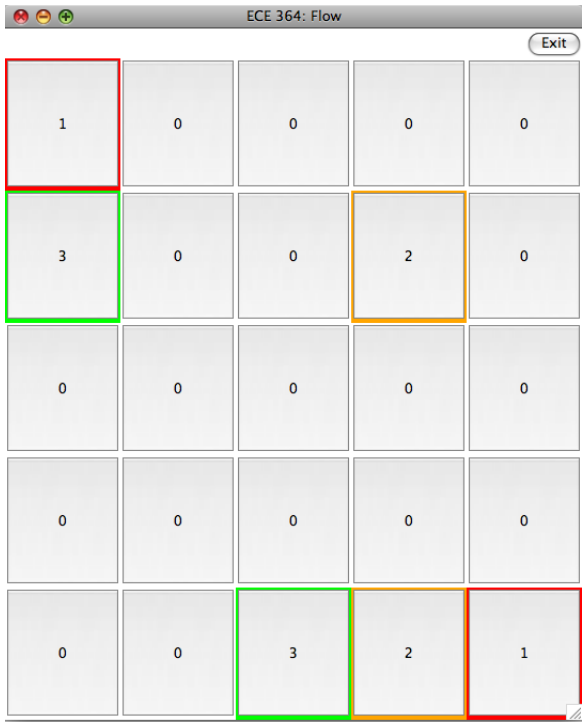

Figure 1: Flow Game UI (Tkinter)

Very basically, the object of the game is to connect all pairs of terminals (denoted in the figure by color) while 1) not crossing connections between terminal pairs and 2) filling the entire board with connections between terminals.

2 Toolbox Options

Use of a Python "toolkit" will be necessary to complete this project. Though there are many toolkits to choose from in the world of Python programming, we ask that you choose one of the following two: **Tkinter** and **Kivy**.

As per `https://wiki.python.org/moin/TkInter`, Tkinter "is Python's defect standard GUI package." There is a wealth of documentation for Tkinter online as it is an immensely popular and widely-used toolkit. The above address displays a page that contains a wealth of links to useful Tkinter materials. The course staff

also highly recommends the website `http://effbot.org/tkinterbook/` . Although Tkinter is an excellent toolkit, it is not easily exportable to a mobile platform. This is not exactly a trivial issue given that Flow has become very popular on mobile platforms, and a project that approximates a real application is what the course aims to provide.

The other toolkit that you may choose to use to implement your Flow game with is Kivy. Kivy is a Python toolkit that is used for developing multitouch application software on all of the major mobile platforms (iOS, Android, Windows). Kivy documentation and tutorials can be found at the Kivy homepage, `kivy.org`. Use of Kivy seems like a fairly straightforward solution to the issue of not being able to easily export Tkinter to mobile platforms, but it turns out that the current version of Python installed on the Purdue Linux grid does not fully support Kivy. As such, though Kivy is like a more natural toolkit for this project, we will not require that you use Kivy since it is possible that not everyone will have access to the requisite tools to use Kivy. A couple of points that you should consider when you are deciding which toolkit to use:

- The course staff is generally quite familiar with the Tkinter toolkit and can be of substantial help with most problems you encounter. We cannot claim the same level of expertise with Kivy so issues you run into may be more difficult to handle and overcome.
- There will be bonus opportunities afforded in Phase II of the project regardless of the toolkit used for implementation. However it is likely that, for a given extra feature, projects implemented with Kivy will be eligible for more extra credit than projects implemented with Tkinter.

3 Game Components, Rules, Board Display

The main component of the game is the game board, which is displayed graphically for the user to interact with using the keyboard and mouse. This game board is comprised of a set of nodes, which may or may not house a terminal when the game board is initially displayed (please see the Figure 1 for clarification).

You will use an input file populated with integers to specify dimensions and makeup of your game board. The game board need not be square, but must have the form of an M by N matrix. Every entry of the game board (node) that does not correspond to a terminal pair should appear in the input file as a zero (0), while each terminal pair should be assigned a unique integer to represent it. For example, let's say that a one (1) in the input file corresponds to a blue box on the game board. Then there must be two and only two ones in the input file and two and only two blue nodes on the game board. A sample input file format is given below:

```
1,0,0,0,0
3,0,0,2,0
0,0,0,0,0
0,0,0,0,0
0,0,3,2,1
```

You will likely want to use the special widgets *Frame*, *Canvas* and *Button* to set up this game board (though you might not need all three). Please refer to `http://effbot.org/tkinterbook/frame.htm`, `http://effbot.org/tkinterbook/canvas.htm`, `http://effbot.org/tkinterbook/frame.htm` to get started on displaying a game board.

As denoted in Figure 1 and described in previous paragraphs, terminal pairs are denoted by color, i.e. two nodes that are the same (non-white) color when the game board is instantiated constitute a terminal pair. For the first phase of the project, we only ask that you are able to connect one terminal pair with nodes of the same color as the pair. That is, you should be able to select one of the two terminals in a pair, then select an empty node that is cardinally (not diagonally) adjacent to the selected terminal, changing the selected node's color to the terminal's color. You should be able to repeat this process until the two terminals are connected by nodes of the terminal pair's color. **Note that although you are only required to connect one pair of terminals in this phase, you will be required to connect all terminal pairs in Phase II so it may be advantageous to get a head start on this.** Some additional requirements/notes:

- Only boxes that are cardinally adjacent to the selected box can be chosen next (can't move diagonally).
- You cannot "backtrack", i.e. choose a node that is already the color of the present terminal pair to be the next selected box. This issue will be revisited in Phase II, where a "backtrack" will change the previous node from the terminal color pair back to white.
- You cannot choose a box that is the color of a different terminal pair, i.e. a green box cannot become a red box.
- You should be able to "cancel" a path by clicking on the terminal of origin. That is, if you have clicked on a terminal and have subsequently created a three-node path stemming from the terminal, a click on the terminal should change the color of the nodes on the path back to white (the terminal's color stays the same). If you are attempting to cancel a path that has reached its destination terminal, clicking on either terminal should cancel the path between the terminals.

In addition to the gameplay specifications listed above, there is a set of display/gameflow features that must be implemented in this phase:

- The top of the game interface should display the game's title (as in Figure 1).
- Exit with a success message pop-up box when all terminal pairs have been connected and all nodes have been colored. If you are only able to connect one pair of terminals by the Phase I due date, please exit with this success message pop-up box when this pair is connected (this will be enough for full marks even though it is not nearly as demanding). Be aware that fulfillment of the completeness condition will need to
- Add a menu bar to the top of the game interface. This menu bar only needs to contain an *Exit* button for Phase I. Again, please see Figure 1 for clarification. Be aware that this menu bar will be required to house more elements for Phase II, i.e. it is a good idea to create a class associated with the menu bar that can be developed more extensively later.

4 Implementation Details and Other Notes

This project is fairly open-ended in character and will be especially open-ended in Phase II. You will have a great deal of choice in regards to what features you add to your Flow game and how these features are implemented. The one major implementation requirement associate with Phase I is this: you must write and use at least two classes in a meaningful way. One of these classes must be a "main game" class, and it is likely a good idea to have at least two other classes: one associated with the nodes of the game board and one associated with the menu bar. **Failure to develop and use at least two classes in a meaningful way will disqualify you from receiving more than 50% credit for Phase I**. We do not want you to define your Flow game with only one class. Such a class would likely be made up of a large number of loosely-associated functions, and would be very difficult to build upon moving forward. We think it will become clear to you that using distinct classes to define different objects associated with the Flow game is an organized, efficient, and fairly easy way to write a relatively complex program.

Regarding input files, recall that you must use the contents of an input file to define the initial layout of your game board. Also note that Phase II will likely require you to have multiple game boards available for the player, so it is a good idea to think about how you might structure your code to accommodate this requirement.

Again, there is a wealth of documentation on both the Tkinter and Kivy toolkits available online, and we highly recommend that you take advantage of these resources. Do not hesitate to come to office hours to discuss the project with the course staff, and note that TAs will be available to answer questions during the hours that were previously allocated for lab time.