**Twixt!**

By: Steven Speicher

2007

How The Program Works:

I used a main class called Twixt which does basically all of the operations. Another class called TwixtCell is a child class of JLabel. It is used to create the 24x24 array of board “cells” or pegs. The methods in this class are used to place a peg in Twixt’s computed location.

Incomplete In-Game:

This program proved very difficult and tedious. There were a few things that I did not get to work completely. Overlapping the graphics interface and a GUI environment was very complicated to grasp and implement. All of the processing for placing connectors is complete except for a minor combination of peg placements, which shouldn’t occur too often, so all I needed the graphics for was to actually create a visual for the player. It still operates fine, but can be difficult to understand to the untrained eye.

Another part that is not complete is declaring a winner. For a simple path, my implementation works, but with more complicated paths, not so much. I save the vertexes in a 576x576 array and if a vertex is connected to more than one other vertex, it only returns the lowest vertex and continues to check from there. Like if the peg 4 is connected to 26 and 30, but then 30 continues on to form a complete path, there will be no winner because my implementation only checks that 26 is not connected to another.

Another part that is not complete is saving and opening an incomplete game. You are able to start a new game, but an incomplete game does not work. I was attempted to have it save the array of pegs and connecters while saving, and then recall these and place them on the board upon opening a saved game. This didn’t all get finished or work.