CS653A Project Report: Dots and Boxes

Krishna Karthik (14292)
Rohith Mukku (14402)

April 2018

1 Abstract

The project is based on a two player board game "Dots-and-Boxes". The game play area of Dots and Boxes consists of a N x N grid of "dots". A players' turn consists of connecting two horizontally or vertically adjacent dots with a line - diagonal lines aren't allowed and the dots must be next to each other. A point is scored each time a player completes a square. When a square is created, the turn stays with the player who made the square, otherwise the turns alternate. The game is over when all the dots are connected and $(N-1)^2$ squares have been made. As this is GUI based game we will be using Haskell packages like q Haskell, gloss for rendering. The game will be having human vs human and human vs computer with choice to select the level of difficulty.

2 Report

2.1 Structure

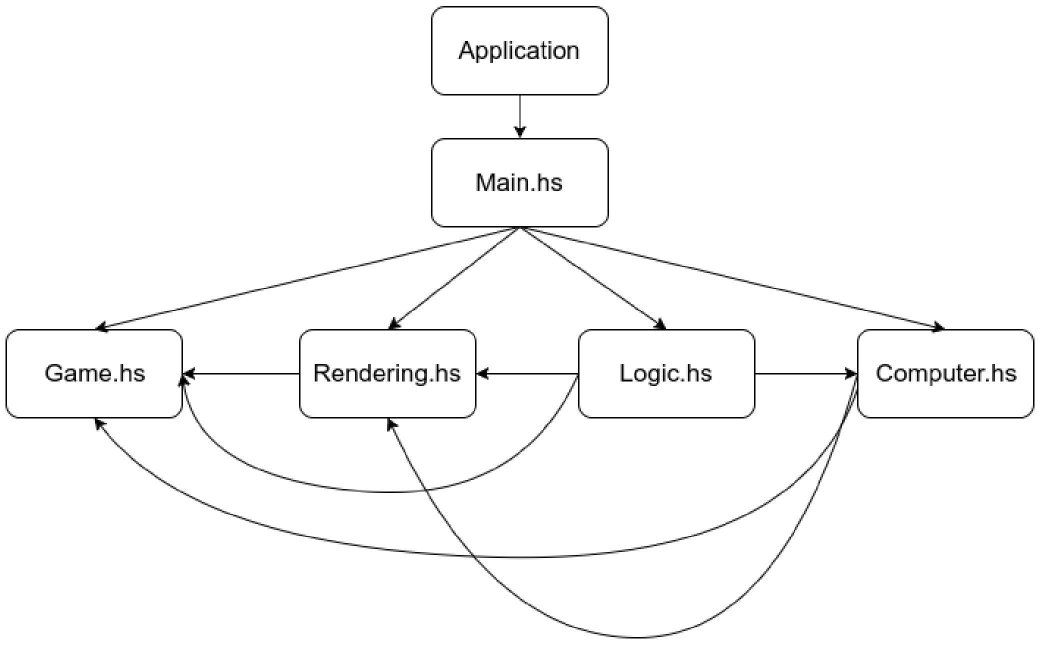

Figure 1: Structure

2.2 Approach

Project Structure

- **Main.hs:** It takes inputs from user i.e, number of dots and gameMode (vsHuman or vsComputer) and uses gloss's play function to initialize the program
- **Game.hs:** Here data constructors and types are defined. Also intial game is setup here
- **Rendering.hs:** It takes game as an input and returns Picture which will be used by gloss to render the scene. Here rendering of dots, lines, marker and their colors depending on the player are handled.
- **Logic.hs:** The main function in this is transformGame which takes an Event as an input. Event in our case is Key press of Up, down, right, left and space. Depending upon the event the marker is moved to the position or toggled. Using the state of marker we validate whether the line is possible or not. If it(drawLine) is possible we then draw the line and check whether the line caused a formation of new box. In Logic.hs we also check whether the gameMode is vsHuman or vsComputer and handle the cases for each of them.
- **Computer.hs:** It maintains possibleMoves which is a list of (Pos, Pos) where Pos is dot co-ordinates. easyMove function in it selects randomly from possibleMoves list

Implementations

- Uses marker to show the highlighted point
- Uses keys (up, down, right, left, space) for navigation
- Shows final colour as the winner
- Human vs Human
- Human vs Easy AI

2.3 Improvements, Deviations

- Hard AI, Medium AI
- Option board, Help board

3 References:

- Game rules <- https://classes.soe.ucsc.edu/cmps115/Spring02-01/GameRules.html
- https://hackage.haskell.org/package/gloss
- https://github.com/OlivierNicole/haskell-chess
- https://classes.soe.ucsc.edu/cmps115/Spring02-01/GameRules.html
- https://github.com/tsoding/profun