School of Electronics and Computer Science

Faculty of Physical and Applied Sciences

University of Southampton

Date: January 13, 2012

COMP3001 Group Scripting Coursework - Team B

Team Members:

Dexter Lowe (dl10g09@ecs.soton.ac.uk) James Skinner () Adam Thomas () Bahman Asadi () Peter West () Alex Parker (ajp3g08@ecs.soton.ac.uk) Tom Smith (taes1g09@ecs.soton.ac.uk)

http://3001Blokus.appspot.com

1 Description of Prototype Functionality

1.1 Blokus

Blokus is a 2 or 4 player strategic board game, where each player takes turns in placing coloured polyominoes (footnote: A polyomino is a 2D geometric shape formed of one or more equal squares edge-to-edge) on a 20x20 board with the aim of placing all 21 of their pieces. For a 4 player game, each player controls one colour, and for a 2 player game, each player controls two colours. Each players first turn must involve placing one of their pieces on one of the four corners of the board. Each subsequent turn must involve placing a piece with only corners touching their existing pieces (so there may be no flat edges of the same coloured shapes touching). No piece may overlap any other piece. Players may employ tactics to block another player from placing their pieces. When no further moves are possible by any player the game is over. A colour loses a point for each square in each unplaced piece. They gain 15 points for placing all their pieces and a further five if the final piece they placed was their monomino (single square).

Client-side validation is used to highlight the area the piece would cover to show whether a position is valid or invalid as appropriate. If the user wishes to rotate or flip the a piece they either hover over it in their piece-tray and use the control halo that appears or once picked up they can use left and right to rotate or h and v to flip horizontally and vertically respectively. Once the user has decided where to place it they click once more to place the piece on the board.

1.2 Authentication

Users are able to play games without logging in, however, those who register will be able to track their statistics about games they have won and lost. Three methods are available: registering with an account that will be native to the Blokus application, logging in with Facebook, or logging in with Google.

1.3 Lobby Screen

From the initial screen that this user is shown - the lobby screen - the user may play a 2 or 4 player game, quick game. Selecting one of these will place the user into a waiting room until there are enough players to start a game, where they will be taken to the game screen. Additionally, the user may select Private Game, where they will be given a unique url to share with their friends, so that they can play a game with only people that they know. While waiting for these players, they will be placed in the waiting room.

2 List of Tools and Techniques Used

2.1 Development Tools

- Various Text Editors with syntax highlighting for general development. No language specific IDEs were used to reduce platform incompatibility issues.
  - Sublime Text 2 (All)
  - Vim (Unix Based Environments)
- Google Chrome Developer Tools and Firebug were used for Javascript debugging.
- Mercurial (hosted on BitBucket) was used for collaborative source control.
  - Mercurial (Command Line) (All)
  - TortoiseHG (Windows Environments)
- Mozilla Firefox, Google Chrome, Chromium, IE9 and Safari were used for testing.
- mjson.tool was used to check the information in python objects.
- gvimdiff, Tortoise Diff, WinMerge, and KDiff3 were used to solve merge conflicts.
- cron was used to garbage collect completed or abandoned games and inactive temporary users from the database.

2.2 Project Management

- Issue tracking using BitBucket
- Google Docs were used for collaborative authoring and document sharing.
- Facebook was used for group contact, communication and organisation.

2.3 Techniques

We used a RAD (Rapid Application Development) approach to development having regular meetings and aiming to have a prototype with more functionality at each one. We also used Facebook and Google Docs to ensure that everyone knew the current state and design of the system.

Temporary data was created on the client side and a DEBUG page to emulate many server functions so that front-end development could continue even if the back-end was not working. In this way the various modules of the system were kept separate as far as possible. Also we used AJAX and JSON based techniques to simplify dynamic web pages and data transfer between the front and back end.

3 Relevant Statistics

3.1 External Sources

3.1.1 CSS

Resets all browser layout so all browsers are more likely to be the same.

A small script used to place labels inside input text fields.

3.1.2 Python

Lets Django work with non relational databases like on the GAE.

Creates REST interfaces for Django web-services.

Allows guest accounts and handles garbage collection for them.

Allows authenticating users via Facebook and Google, among others.

3.1.3 Javascript

jquery Allows easier manipulation of the DOM.

underscore Provides powerful tools to work with advanced data types.

backbone Provides simple routing and provides client side representation of models.

Raphaël Provides a simple and powerful interface for SVG.

3.2 Relevant Statistics