SEng 513 Final Project Report

Jaw Dropping Draws

Group 9: Stefan Jovanovic,
Anton Lysov,
Zia Rehman,
Jeffrey Tinkess,
Aleksandar Todorovic

Introduction

Jaw Dropping Draws is a project based on a popular game Pictionary. The basis of our game is simple; if you are the drawer you attempt to draw a picture of the word that has been given to you. If you aren't the drawer, try to guess the word based on the picture. Points are awarded for guessing the word correctly with a bonus for being the first, the drawing player gains points based on others guessing their drawing. We also offer storage of pictures that are particularly interesting or memorable, allowing any signed up user to recall pictures.

This project focuses entirely on providing entertainment, and we have focused our offered features on this goal. For example, one major feature we noticed was missing from current versions of online pictionary was the ability to store images. We believe that a large percentage of the entertainment value of pictionary comes from the sometimes random images that are created and the users reaction to them. For example, contrasting very high skill images against low effort or purposely obtuse images. One of our major goals for this project was to include the ability to gather and recall images such as these, extending the entertainment value generated by playing beyond any single games scope.

We were able to create a basic website capable of handling the basic functionalities of pictionary such as drawing, transmission and display of images, and word guessing. We also included the ability to chat between players and the above mentioned image storage to help increase the entertainment value of our app.

Project Description

Our website consists of a single page, with a notification box for sign in and several links which modify the page based on which feature the user wishes to use. When any user first reaches the website the signup/sign in dialog box automatically gains focus, however the drawing functionality can be reached without utilizing it by closing or clicking outside this dialog box. It can be reopened if accidentally closed or if the user changes their mind later by clicking the “” button.

The main page consists of four main parts: the navigation menu at the top which provides buttons each of which will modify the other three parts as required, a main window which almost always displays the drawing canvas, a chat window which doubles as the word guessing area, and a footer with game controls and feedback. The navigation menu currently only allows sign in/logout and changing between game mode and viewing mode. The My Account button is a placeholder for potential future functionality.

The main functionality is the game mode. In order to use it, any user would simply have to navigate to the website and either sign in or dismiss the sign in dialog window. Once this has been done, the application will automatically place the user into the drawing rotation and begin the game immediately. The user can type into the chat window at any time they are not drawing, and if the word being drawn is included in any chat message it will be detected and a message printed out declaring the results. The word itself will be censored so the remaining players can continue guessing.

When in game mode each user will either be the drawing player or a guessing player. The drawing canvas will only be interactive for the drawing player, and the drawing tools are only visible to the drawing player. In addition, chat is only interactive for guessing players: the drawing player cannot type in chat. This is a design decision to help prevent cheating, and encourage the drawing player to focus on drawing.

If the user signs in they will have access to a second screen, used to view images. On this screen will be either one or two buttons, used to move to the next or previous image saved on the database. The image will be displayed in the same area as the drawing canvas in game mode, but will be non-interactive. Only one image is displayed at a time.

Proposal vs Final Project

We made numerous changes to our project based on feedback from the instructor and limitations we discovered in the frameworks we used. The single largest change was related to game logic. In our original proposal we intended to create teams and combine images to create a composite image, however we decided to change to a single drawing player and have all other users play individually as guessers. This was for two reasons: it drastically simplified our testing and implementation, and our drawing framework was incapable of splitting images efficiently.

A second major change was our decision to remove facebook authorization. In the initial proposal we intended to use firebase authentication to allow both facebook and manual signups, however we decided that it was not worth the extra complexity. We still used firebase for authentication which means adding facebook login in the future would be very simple to execute.

We modified our drawing tool UI due to limitations within the fabric.js framework. We had initially intended to have a series of preselected colours and a constant brush size. However when we began implementing drawing functionality we found fabric.js does not support the colour selection method we had intended. The brush size slider we included was found while investigating implementation methods and included to improve the user experience.

Project Requirements

Single page interface

Our application uses a single HTML page served from node using express. This page is modified using jquery when the interface must change, so the user is never redirected. The authentication window is a separate popup dialogue box, however it never redirects so only a single URL is ever used.

Use of mostly HTML, CSS and JavaScript

We used almost exclusively HTML, CSS and javascript. We used two main libraries: firebase, which acts as a database for JSON objects and utilizes javascript for access and retrieval, and fabric.js which is a javascript library for drawing and rendering images. All code we wrote was HTML, CSS or javascript.

Tested on Chrome and Firefox

The program was tested on Chrome and Firefox in the on-campus labs, as well as chrome on individual laptops. We found no loss of functionality or issues with portability during our testing, and all libraries and language features used were fully cross platform supported.

Mobile Support

We use both flexbox and Bootstrap for page structure, both of which are supported on mobile devices. We avoided wherever possible computation-intensive client side code to maintain mobile friendly performance.

Node.js on server

Our server side code is entirely written in node.js, and all communication to and from firebase occurs via node.js.

Support for multiple/simultaneous users

Our game currently supports any realistic number of simultaneous users, using asynchronous communication and node.js's ability to handle multiple connections without freezing. Users are placed into a queue and rotated through as the game progresses, with users able to drop in/drop out quickly and without impacting the user experience of other players.

User authentication

We use Firebase Authentication to calculate unique user tokens, used when logging into the database. Currently our authentication is limited to basic username/password, but expansion to facebook authentication is possible with a simple modification.

Persistence

If a user logs in their images are persistent across sessions, due to the unique nature of firebases authentication tokens. Currently user game data such as score is not persistent, however it is possible to add this at a later date by utilizing the provided tokens in more areas.

Interaction between users

Interaction takes place in two main areas: players are able to chat with each other, allowing group communication and interaction between guessing players, and the drawing page itself, which is live updated and broadcast from the drawing player to all guessing players.

Technologies Used

Fabric.js

The main library used for our project is an open source javascript drawing library called Fabric.js. Fabric is intended for interactive image rendering including animation, drawing and serialization/deserialization of images. We use fabric by tracing the users mouse and creating new fabric graphics objects, then rendering them on every mouseUp event and broadcasting the entire canvas to the server for distribution. Fabric also handles rendering the broadcast image on the receiving end. Finally, Fabric serializes the canvas on demand (when the “save image” button is pressed) and sends to the server as a JSON string, which is the format used by firebase for database operations. Retrieved database images can be rendered in the same manner as rendering broadcast images.

We used fabric due to its simple implementation, and the fact that its output matches and is easily used by both firebase and socket.io, limiting the amount of conversion we had to include.

Firebase

One of our main features (image storage and retrieval) required a database capable of storing JSON objects. Some of our group had previously used Firebase, which provided simplified database operations and reduced the complexity of our task as we did not have to set up an SQL server and DBMS. Firebase also included authentication capabilities, which we used to ensure only registered users could utilize database resources.

Socket.io

We utilized Socket.io for client-server communications due to its simple use and the fact that everyone was familiar with its use from assignment 3. All communication between the client and server is handled with socket.io channels, with only two communications in the entire project using other protocols (namely, the authentication call from the client directly to the firebase server, and the push/retrieval of images from the server to firebase).

Node.js

Node.js was used for our server-side code for several reasons, including our familiarity with the software and its multithreaded, non-blocking properties which were required to minimize lag when transmitting the drawn image back and forth to a large number of clients.

Program Execution

In order to setup and run our program, first clone the git repository from https://github.com/STEjovanovicFAN/Drawing-Project-SENG-513 . Once cloned, open the project folder and install express using the command “npm install express” from the terminal command line. The server can then be executed using the terminal command “node server.js”. You should see the message “server is now listening on port 3000” in the terminal which confirms correct initialization of the server. Any number of browser windows in chrome or firefox can now be navigated to “localhost:3000” to open a client-side view of the program.

Future Work

If given a further 4 months of work on this project, we would begin with a thorough refactoring of the code base. There are several issues that could be corrected to improve the quality of the code itself, such as combining parallel socket.io messages and improving game start and transition functions. This would likely be 1 to 3 weeks of work for our team (note all time estimates are based on a standard semester workload).

Our second task would be implementing any features that were removed or reduced due to timing issues. For example, including a “my account” page attached to the current dummy button that allows users to change account details such as password and migrating the change username functionality from chat to this tab. Facebook authorization would also fall into this category. This should take only 2 weeks as most features are simple and we included tie in points for them.

We could then work on adding larger features that would be nice to have and improve the user experience, but are too complex or time consuming to include in our submitted work. One of these features would be game groups, where instead of having a global pool of users with a single drawer we split users into multiple groups, each with a single drawer. This would ensure more opportunity to draw and improved performance for the average user. Another feature is the ability to create private groups, where the creating user has the ability to control game logic constraints such as time to draw, canvas size, points awarded and number of players.

Once the features we are sure would work are implemented, we would like to attempt our initial plan of having teams drawing and image splitting as an alternative game mode. While fabric.js does not support splitting images this way it may be possible to find a way to make it work, if we had 4-6 weeks to experiment with server-side image manipulation.

Finally we would like to attempt integration of a third party API such as google advertisements that would provide some form of revenue stream. While it is unlikely this app would provide a significant revenue stream, having the opportunity to work on this task would serve as an ideal learning opportunity and potentially provide enough funding to pay for its own online hosting and expanded firebase operations.

Conclusions & Miscellaneous

This project was heavily modified from its original proposal, however we found that it still provides a fun and entertaining way to spend a short break. It also provided an excellent learning opportunity for every group member, and each of us now knows far more about javascript and web application development that we did prior to our work here.