Mathboard

- COMP3207 Group Coursework (cw2) -

Team K

dms1g14 Dan Schormans
jhb1g14 James Hueston Berry
jdl1g14 Josh Lyndsell
md8g14 Michael Dunn
pd7g14 Pip Dunne
sdjb1g14 Simeon Brooks

Source Code:
github.com/Simeon26/Mathboard

Link to running project on AppSpot
mathboardk.appspot.com

Description of prototype functionality

With the modern teaching environment becoming evermore de-localised, we implemented a virtual whiteboard for remote-teaching with an emphasis on mathematics.

1.1 Lobby System

The whiteboard, dubbed Mathboard, does not have any logins for simplicity. Instead, the host (the teacher), creates a 'room' which generates an associated 4-digit code. This can then be shared with potential viewers (the students) which allows them read-only access to the Mathboard.

1.2 Standard Whiteboard Features

As a whiteboard, the main feature is a pen. This can be used by the host to draw on the board in a variety of colours and thicknesses. The colours can be chosen either from a pre-defined pallet, a colours chooser (RGB or HSL) or a custom pallet to which colours can be saved.

The custom colour pallet can be saved and restored on any future Mathboard (provided they are on the same computer and browser, the storage is cookie-based).

All objects on the Mathboard can be edited as objects by the host - with resize, rotate, move and delete functions available.

1.3 Mathematical Features

The Mathboard is set apart from existing shared-whiteboards (of which there are many), by the integration of wolfram alpha. At the bottom of the hosts view is a text-query box into which almost anything can be typed from "Solve 2x+y^2=2 and x+y=0" to "Plot y=2x^3-9x^2+4x" and "What is the diameter of the earth?". The output of these queries is then inserted directly into the Mathboard. See figure 1 for the outputs of these particular queries.

Because of the sheer versatility of WolframAlpha (which is the engine behind Siri), this feature allows a plethora of mathematics to be computed directly into Mathboard.

$x = -1 - \sqrt{3} \approx -2.73205$ and $y = 1 + \sqrt{3} \approx 2.73205$

Figure 1.1: Output 1 for input Solve 2x+y^2=2 and x+y=0

$x = \sqrt{3} - 1 \approx 0.732051$ and $y = 1 - \sqrt{3} \approx -0.732051$

Figure 1.2: Output 2 for input Solve 2x+y^2=2 and x+y=0

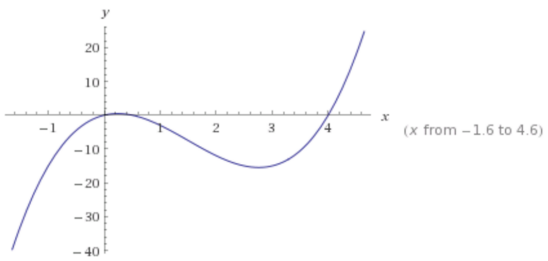

Figure 1.3: Plot y=2x^3-9x^2+4x

7917.5227 miles

Figure 1.4: What is the diameter of the earth?

List of tools and techniques used

We used many of the agile development processes' standard tools and structure. Specifically we made use of:

2.1 Git (GitHub)

One of the most important tools to the completion of this project was Git. We used a single, private repository (shared with all members of the group) initially only in order to track and maintain progress as a group.

This was then forked by different teams working on different parts of the project. These elements were brought together throughout using pull requests.

Extensive use was also made of the inbuilt GitHub editor for small changes - saving significant time over pulling, adding, committing and pushing on a local machine.

2.2 The Google App Engine (GAE) SDK

Although this partly goes without mentioning for a GAE based project, there were two elements worth comment. Firstly, the local development testing environment which allowed the running of a GAE application on the localhost. This greatly sped up the development process as the deploy process (to Google's appspot service) was fairly slow (about a minute).

For the most part the application was deployed only by those working on the back-end of the application. For ease we had two development GAE projects for different members to use, since the process of sharing a project was found to be generally unhelpful.

Within the platform, we used the Python27 execution environment for serving the graphics data and HTML pages, and the Google Datastore for persistence.

2.3 Pair Programming

Since the project team was fairly large (6 members), it was decided pair-programming should be used where possible. This was thought particularly important in-case the project schedule spilled into the Christmas vacation with the danger of team members being unreachable. As such, a broad division of work was decided on with:

- Michael and Simeon working on the python back-end
- James and Dan working on the lobby system
- Pip and Josh working on the front end JavaScript

This was only an outline and was followed less in the latter stages of development but provided a framework to start work and ensure that there were always people available to explain any part of the project.

2.4 Scrum Meetings

Particularly at the beginning of the project, when the broad-stroke ideas where being put in place, we held (approximately) weekly meetings to discuss progress and aims.

These were only some of the methods used - although they were the most important.

Relevant statistics and Libraries

3.1 Lines of code counts

	Source A (written)	Source B (adapted)	Source C (included)
Server side (Python)	182	4520	0
Client side (JavaScript, HTML, CSS)	1369	0	58314
Total	1551	4520	58314

3.2 Libraries used

A number of libraries were used as the basis for the project. They are listed below, split by those operating on the client-side of the application (JavaScript, HTML, CSS) and those on the GAE platform (Python).

3.2.1 Client-side libraries

- Bootstrap - 3.3.7 - `getbootstrap.com` - To ease styling of the app
- Fabric.js - 1.6.6 - `fabricjs.com` - A wrapper for the HTML5 canvas with serialize functionality
- jQuery - 1.12.4 - `jquery.com` - For manipulation of the DOM

3.2.2 Server-side libraries

The majority of these were GAE libraries required for using the platform's features. These were:

- webapp2 - For handling requests as a WSGI application
- Setuptools - Used for managing imported python packages
- codecs - For reading text files
- os - Getting the path of the active project
- sys - Basic python module for running on GAE
- cgi - Alternative request handler
- urllib(2) - For fetching data via secondary requests
- random - For generating the random room codes
- db - For reading and writing data to the Google DataStore

The server-side also made use of:

- Wolframalpha - `wolframalpha.com` - For querying the WolframAlpha Engine
- Jaraco.itertools - `github.com/jaraco/jaraco.itertools` - Util package for iteration problems
- More-itertools - `pypi.python.org/pypi/more-itertools` - Further iteration based utilities
- Inflect - `pypi.python.org/pypi/inflect` - Used for correct usage of plurals in written parts
- Six - `pypi.python.org/pypi/six` - Used too soften the differences between different python versions
- Xmltodict - `github.com/martinblech/xmltodict` - XML parser and encoder

Brief overview of design and implementation, including key design decisions

4.1 Development of the idea

Initial design specifications for the prototype were based on the ideas of group collaboration in learning tools. Specifically that it is often extremely difficult to use traditional communication tools such as VoIP or instant messaging services to discuss maths, since manually typing or saying formulae is difficult, and it is almost impossible to describe even simple plots.

The initial plans came in the form of a "paint" style application that was viewable live. This would allow for problems like communicating formulae and plots to be simple. We decided that implementing this could be done most simply by using a drawing library in JavaScript (fabric.js), making drawings visible to everyone in a certain room by creating new whiteboard emulations. These rooms are created and organised by the Python back-end.

The idea to add functionality that made displaying mathematics came from the team's personal use of collaborative learning tools – particularly Khan Academy, whose online lessons rely heavily on pre-recorded "whiteboard-style" content. Often in these videos, the speed at which mathematical concepts can be taught was often handicapped by the speed at which relevant plots and formulae could be drawn out and in many cases this would be exacerbated by the use of computer drawing tools.

We decided that finding a way for users to quickly and simply create clear, readable plots was important. We decided to use services provided by the wolfram alpha API, as the wolfram alpha website as very good at both interpretation of mathematical input and clean-looking output of mathematical plots and equations. Another benefit of using wolfram alpha as part of the website is that it can act as a built-in calculator in the website, giving extra functionality to the hosts of a room, without making the software any more difficult to use.

The notion of adding a chat-box to the side of the whiteboard was a way of increasing the flexibility of use of the software, not only does this allow a host to teach a lesson to people non-locally without using extra software. This feature is also particularly useful if the application is to be used as part of a MOOC (a large, free, web-based class), as not only can teachers use the chat, but students can too, meaning that students can reinforce their learning as a group in the chat - one of the most meaningful features of a MOOC.

4.2 Design Pattern

The application is based around the Fabric.js library. This wraps an HTML5 canvas with useful functionality allowing for the programmatic addition of graphics, manipulation of tools on it and most importantly a serialize function that converted any graphics on the canvas to JSON.

This is supported by a Python back-end which stores this JSON alongside a room identifier in the Google DataStore. This allows for the viewers to poll the back-end (supplying the room identifier) and be updated with the JSON encoded graphics.

Critical evaluation of the prototype submitted

5.1 Strengths

The greatest strength of our 'Mathboard' application is its feature richness while still maintaining total ease of use, "viewers" can begin learning in just seconds with only a room number, this is very important for a learning environment that is often under time constraints. "Teachers" need no prior knowledge to start using the 'Mathboard' right away. Other strengths of this application is the chat functionality, this allows both interactive learning, communicating with the teacher, and collaborative learning, by sharing ideas with the other "viewers" of the session. One of the greatest strengths of this application is the immediate updates from the "host" to the "viewers" with only a one second refresh rate for an unlimited number of users.

5.2 Weaknesses

Upon testing of the application, we have noticed a weaknesses that may require further development, there is no easy method implemented within the 'Mathboard' to notify the "viewers" of which room number they should be entering, this would therefore have to be transferred via alternate communication streams.

5.3 Time Management

Agile development was a great technique to ensure we were never overwhelmed by the quantity of work left to produce with this project. The use of frequent scrums and pair programming set a good structure for completing tasks although they were not stuck to as definite rules.

5.4 Further Development

At the top of the next-features-to-add list would have been to have some form of public room as well as private. These would be listed on the front page with a brief description for public class rooms. Other features that we would have liked to have implemented:

- Hotkeys for the difference tools when drawing on the Mathboard (e.g. 'r' brings up red pen).
- Further editing of existing objects (e.g. change the colour of a line that has already been drawn).
- Bigger whiteboard, and crucially one that scrolls. As it is a small screened device has a limited space to work within.
- Multi-user editing. It would have been desirable to have Mathboards that are universally editable, and also an option in a teacher-student Mathboard to give a student temporary access to 'write on the board'.
- Maths typesetting in the chat window so that students could communicate complicated mathematical expressions in the same way that the teacher can through the board.
- This item would likely have never been possibly with the limitations of GAE (when used for free), but complimentary audio and or video streams alongside the board, or even layered, would have made the tool even more versatile (if less lightweight).