University of Southampton

COMP3001: Scripting Languages

Group Coursework Assignment
Team D

Authors:

Hendrik J. Appel (hja1g11)
Tristan Bunyan (tb15g11)
Felix Chapman (fc4g11)
Sebastian Claici (sc2g11)
Antonios Georgiadis (ag9g10)
Alexander Hutson (ah16g11)
Mohammad Ali Khan (mak1g11)

January 8, 2014

1 Description of prototype functionality

We have developed an educational game that helps users improve their typing speed and accuracy in an enjoyable way. It was inspired by the breakout arcade game PONG from Atari and various online ‘type racers’. Much like the original PONG, our game involves a moving ball and two paddles and the goal is to score more points than the opponent (AI or human) or to simply survive as much as possible against an unbeatable enemy. Our innovation lies with the fact that instead of free control of the paddle using simple keys, our version lets the user move the paddle only after they have completed a typing challenge. This is done by presenting users with three possible paddle positions and three corresponding words or phrases; only after they have typed the correct word or phrase does the paddle move.

In order to engage users we offer a number of different game modes. For users who prefer single player games we have a campaign mode and a challenge mode. In the campaign mode, a user plays against a realistic AI opponent until either reach a target score; if the user wins, they then progress to the next level which will have a harder AI opponent and more difficult words. In challenge mode, users play against an unbeatable opponent and the goal is to survive as long as possible with a constantly accelerating ball. For each user we store their highest campaign level as well as their challenge mode survival times. To make our game even more competitive, we provide a hi-score table functionality where users can view the top ranking players of the aforementioned modes. In addition, the game includes a player vs player (PvP) mode where users can compete directly against each other over the internet. The PvP mode uses an ELO ranking system and an associated matchmaking algorithm which attempts to match players of similar skill based on their ranking. Moreover, we developed a lobbies system that allows players to choose their opponents in order to give friends the ability to play specifically against one another.

Since our game stores numerous scores (e.g. campaign progress, challenge survival time, PvP rankings), we decided that users would have to register and login to play. To make this as simple as possible, the game only requires a username and a password. Alternatively, users also have the option to login using their Facebook.

2 List of tools and techniques used

2.1 Tools

In order to effectively function as a team of this size we took early action and agreed on which tools, techniques and practices we would use. Naturally, different team members have different preferences, but we ensured our submitted code would always follow specific quality and structural standards. Below we present a grouped list of the tools we used for the development of our prototype.

- *Code development*: vim, Sublime Text 2.
- *Code management and versioning*: Git, meld merge tool.
- *Testing and development*: Google AppEngine launcher, JSFiddle, JSLint.
- *Communication*: Facebook messages, Skype.
- *Documentation*: Google Drive, $\LaTeX$.

2.2 Techniques

Whilst we as a team appreciate that agile methodologies for software development such as the SCRUM method were very powerful, this task would be ongoing alongside many other work commitments of each of the developers. This meant that daily sprints and meetings would not be possible. We nonetheless tried to take a number of the key aspects from such methodologies when developing this system.

1. Weekly semi-formal meetings: Each member presents the work they have completed since the last meeting. At the end of a meeting, each member must have a list of tasks to work on for the next meeting. These tasks are documented in a shared forum. This agreement of work to be done prevents overlapping or duplication of work. Publicly documented tasks mean everyone is accountable for their assigned tasks.

2. Pair Programming: When working on some of the more fundamental aspects of the system, we felt that making use of pair programming would be beneficial, as it reduces the likelihood of code defects and bugs being introduced. This technique was applied mostly when developing the core features of the system - the areas where defects would be the most costly.

3 Relevant statistics

As we can see in Fig. 1, the number of commits made increases steadily over time. This behaviour is to be expected as early on in the project a lot of the work was in planning - designing our prototype and user interface - whereas later on in the project we were focused on code production. There are a few notable dips in productivity: the first, around week 7, being caused by deadlines for two separate modules occurring in the same week. The later dip in week 10 was due to the Christmas vacation where team members had family commitments.

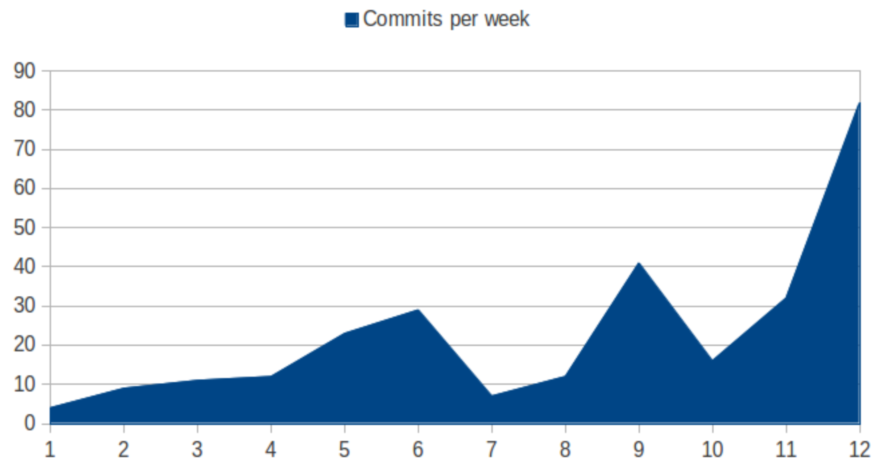

Fig. 1 Total project commits per week.

We budgeted a total of 420 hours to work on this project - 60 hours per team member. Tab. 1 shows approximations for the number of hours spent in total on different areas of the project. In tab. 2, the final line of code count is shown for each language used.

Tab. 1 Breakdown of work allocation.

Area	**Hours Spent**
Technical Meetings	40
Code Development - Javascript	100
Code Development - Python	80
HTML + CSS	60
Testing Prototype	20
Project Report	40
Prototype Demonstration	60
Total	400

Tab. 2 Breakdown of lines of code.

Language	**Blank**	**Comment**	**Code**
Javascript	205	98	1129
Python	264	69	857
HTML	38	6	436
CSS	46	7	258
Total	562	188	2711

4 Brief overview of design and implementation

Since the application is provided via Google App Engine, we use the webapp2 Framework for URI routing, request and exception handling.

With regards to the game, the paddles can move to three different positions: up, centre and down. Three words can be typed, correlating with the three paddle positions. If the word is typed correctly for the position where the ball will go, the paddle will move to return the ball. The skill of the AI increases with the campaign level: the higher the campaign level the lower the probability the AI will misjudge and move to the opposite direction from the expected ball position. Whenever the game requires additional words, it sends a JSON request to the server by using Ajax. To obtain the words, the gaming client queries the server with a specified difficulty level and the server then dispatches this response to the `Sampler` class, which randomly chooses from the words of that difficulty level. Currently the difficulty level corresponds to word length.

For storage purposes we use the Google datastore. Our application stores both single user and Player vs Player (PvP) session information. The datastore is accessed via GQL queries which follows a similar syntax to SQL. We wanted our application to offer social network integration; hence we created two types of users: those who registered using the application itself and those that logged-in through Facebook. The only difference between the two types of users is that for the latter we store the Facebook ID login information instead of the password; everything else is handled by the social network service.

For each user, we store their username, the campaign level they have reached, their high score in Challenge mode and their PvP rank. With regards to the latter, a user starts with a rank of 1500 which is then adjusted according to our ELO rating system whenever PvP matches are played. To secure user accounts and thus user sessions, we hash and salt the passwords upon registration and store them in the datastore. This is implemented via the Python `hashlib` library. When a login attempt is made, the entered password is hashed and checked against the stored hash. Once logged in, we wished that the user's session is kept updated with all the relevant scoring information. We achieve this via cookies from the webapp2 Framework that store the users name and PvP rating.

For PvP mode, players can join or leave game sessions hosted by other players. These sessions are placed in the datastore and are sought through either matchmaking (based on PvP rating) or a lobby system. The matchmaking involves the user sending a message to the server and waiting for a response of any available sessions found in the datastore. If a game is found, one player joins the other's session and players can begin communication. Otherwise, the user creates their own session which is then stored in the datastore available for others to join. Player communication happens by sending JSON-formatted POST messages to the server. The server uses the Google Channels API to forward messages to the other player. The deterministic nature of the game means only a minimal number of messages need to be sent between the players. Thus, messages are only sent each round and when the ball or paddle trajectory changes.

5 Critical evaluation of the prototype submitted

In this section we provide a critical evaluation of the prototype. We present elements that were executed well as well as those that could be improved or that were omitted altogether.

To begin with, the actual game used by the application is robust. Arrows clearly indicate the starting direction of the ball when the round begins and there is a brief countdown to let players prepare. The world length and the AI for the opposition in the campaign scale naturally, relatively changing difficulty levels with campaign progress. All these features result in an intuitive and pleasant game. It is easy for a user to grasp and fulfils the educational purpose of helping users type faster.

The interface for the application is very clean and simple. A visitor who is not logged in will be automatically directed to either login or register. The main page of the application launches the user straight into playing the game. Various game modes are available as well as the option to view high scores with a single click. This simplicity is important since complex interfaces can bore and confuse new users.

With regards to the Player vs. Player (PvP) aspect of the application, there is some flexibility allowing the user to do both a matchup against a player of similar rank as well as find a game in a lobby system. The game runs very similarly to when playing against an AI opponent, but the additional element of being able to play against friends adds a competitive edge that will serve to reinforce the appeal of the game.

One area in which we feel the application could be improved is social network integration. Although the application supports users logging in via their Facebook accounts, there could be greater lengths taken here. For instance, the functionality to share campaign progress or high scores on social media would have added a new layer of competitiveness. Furthermore, there could be more social media platforms available for the convenience of all visiting users.

Another point worth mentioning is that, despite our product being a prototype, we had hoped to implement a custom mode that would allow users to freely choose word length and the ability of the opposing AI. This feature would have let users learn at their own pace and could serve as a practice ground for words of a specific length in order to get past a certain level in the campaign. However this did not come to fruition due to other time constraints for our group members.

On the whole, the group feels that the prototype is well made, engaging and educational. Therefore, even considering the minor drawbacks mentioned above, the application is a successful embodiment of what we had planned to make.