NUMBER PUNCH

<u>Table of Contents</u>

<u>Description</u>

The goal of our project was to develop a multiplayer reaction-based "limited number" matching game that could be played through the web. Our website will have a lobby system wherein players can create and join games, a system by which every user can keep track of his/her user stats, and a variety of cosmetic themes that he/she can change to suit their tastes.

A game of Number Punch starts with both players given an identical set of numbers; a standard game of Number Punch will provide both players with the set of integers ranging from 1 to 10. Players will also be given a constantly changing *target number*, a number that they have to match using one or the sum of any combination of their unused numbers from their set. The target number will change when one of two conditions is met: (1) when the target number has been met by the user or (2) when the user presses the target reset button. In the instance when a user meets the target number, the set of all numbers used in combination to fulfill the target number will be considered used -- their buttons in game will be grayed out and those numbers may not be used again to meet different target numbers. However, there will be some instances when a number cannot be met with any combination of the user's unused numbers. In game, there will exist a *reset button*, whose functionality is to provide a new target number withstanding penalties that can be set before the game starts (a standard game of Number Punch defaults to no penalties).

Process

We are using the software development process we each had previously used in CS 427, Extreme Programming (XP). This agile process advocates the short development cycles we marked with each iteration. In the style of XP our process included spike solutions, pair programming, unit testing, simple code, and a continuous planning feedback loop. Using the XP planning game, we came up with user stories and divided them into tasks and estimated units. Each iteration we updated, modified, or reorganized tasks and stories as the project grew. We made sure to write unit tests for each task to cover each new piece of code during the iteration it is written. Each iteration we also were expected to have some refactoring of previous iterations' code which fit very well in our XP process.

We used the course wiki as the main hub for our group's information, project description, and XP user stories and tasks. We also kept meeting agendas and minutes on our wiki. We used git for version control and Slack for communication.

Pair coding

We used the continuous code review process of pair programming throughout the project. Each iteration we rotated pairs and assigned each pair a task to complete. By pair programming we create better quality code and have more than one person responsible for every item while allowing everyone in the group to learn from each other.

Weekly group meetings

The full team met twice per iteration, each Saturday at 2pm. In the weekly group meetings we went over the status of each pairs assigned tasks and were able to provide help and clarity between pairs. On weekends buffering new iterations we assigned new pairs and reviewed our upcoming user stories and tasks, discussing and entering modifications on the wiki as necessary.

Requirements & Specifications

User Stories:

Lobby System - As a player of *Number Punch*, I can go to the main page of *Number Punch* to be able to see a lobby system which displays current game instances as that I can join and provides me with the option to create my own game.

Game Interface - As a player of *Number Punch*, I should be able to see an interface of buttons when I join or create a game which allow me to tell the game that I am ready to play and, if I created the game, to start the game as well. Furthermore, this interface should display a screen notifying me when someone wins the game.

Game Logic - As a player of *Number Punch*, I should be able to use the buttons in the interface to play a game of Number Punch. I should be able to click on a number to add it to my running total towards my target, and if that target is reached all selected numbers should be marked as matched and a new number be assigned. I should also be able to press a reset button in order to receive a new target number and deselect all of my currently selected numbers.

User Preferences - As a player of *Number Punch*, I should be able to open a menu of preferences which allow me to set a username, site style, and game background. These settings should persist.

Game Customization - As a player of *Number Punch*, I should be able to customize parameters of the game when I am creating a game. These parameters should include which type of game I want to play, how often I will be penalized for resetting my target number, and the number of numbers available for me to use during play.

Game Chat - As a player of *Number Punch*, I should be able to enter messages into a chat box during a game and submit them so that my opponent can see them.

Multiplication Mode - As a player of *Number Punch*, I should be able to enter a game mode wherein, instead of the traditional method of adding my selected numbers together to reach a target number, my selected numbers are multiplied together to reach my target.

Random Operator Mode - As a player of *Number Punch*, I should be able to select a game mode in which each time I am assigned a number I am also assigned a mode of reaching that number (addition or multiplication) at random.

User Statistics - As a player of *Number Punch*, I should be able to have a running total of the number of games which I have won/lost which will be displayed upon completing a game.

Use Cases:

Being a web based multiplayer game, the primary actor in all our use cases is always the user who is playing the game. All the functionalities implemented in the game happen at the user level as they require user interaction.

When a user enters the site, they will be in the scope of the Number Punch lobby and menu screen. From this point, a user will be able to accomplish these main goals :

1. Create a game
2. Join a game
3. Refresh game lobby
4. Configure main page preferences

Create Game

The user clicks the 'create game' button, which starts a new game instance and publicly displays the game's id on the lobby. The success of this case is always guaranteed.

Join Game

The user clicks on one of the games displayed in the lobby or selects "Join a game" which selects a random opponent . This will connect the user to the second user who has created the game. If the process fails, the user is thrown back to the lobby menu.

Refresh game lobby

The user clicks the 'Refresh' button. This updates the games listed in the lobby its most recent state. The success of this case is always guaranteed.

Configure main page preferences

The user clicks on the ''user preferences' button which opens up a menu that allows the user to customize their main page. The user can check the radio buttons for the options they prefer and then hit the save button. This will apply the changes to the page. The success of this case is always guaranteed.

Once the player has created or joined a game, they will be in the scope of a match of Number punch. From this point, the user will be able to accomplish these main goals :

1. Ready up for a game
2. Start a game
3. Configure game settings
4. Forfeit a game
5. Select number buttons
6. Deselect number buttons
7. Reset number

Ready up for a game

The user clicks the ‘Ready’ Button. This will signal the game instance that the user has confirmed they are prepared to start the game. The success of this case is always guaranteed.

Start a game

The user clicks the ‘Start Game’ button. Only the user that created the game will have this option. The creator will not be able to use this button until both players have clicked the Ready Button.

Configure game settings

Only the user that created the game, will be shown a menu which will have options for different game types, penalties and a setter for the number of buttons the game will contain. The user can then select their preferences and hit the save button, which will complete the game creation process and allow other players to join game

Forfeit a game

The player clicks ‘Leave Game’ button. This sends the player back to the main menu. It will also trigger a win screen for the user still in the game. The success of this case is always guaranteed.

Select number buttons

While playing the game , the user can click on the deselected number buttons to select them. This will turn the number’s light from red to green and will make that particular number count towards achieving the target number. But, If a player selects a number button that makes the total go above the target number, the game will deselect all the selected numbers

Deselect number buttons

While playing the game , the user can click on the selected number buttons to deselect them. This will turn the number's light from green to red and will make that particular number not count towards achieving the target number.

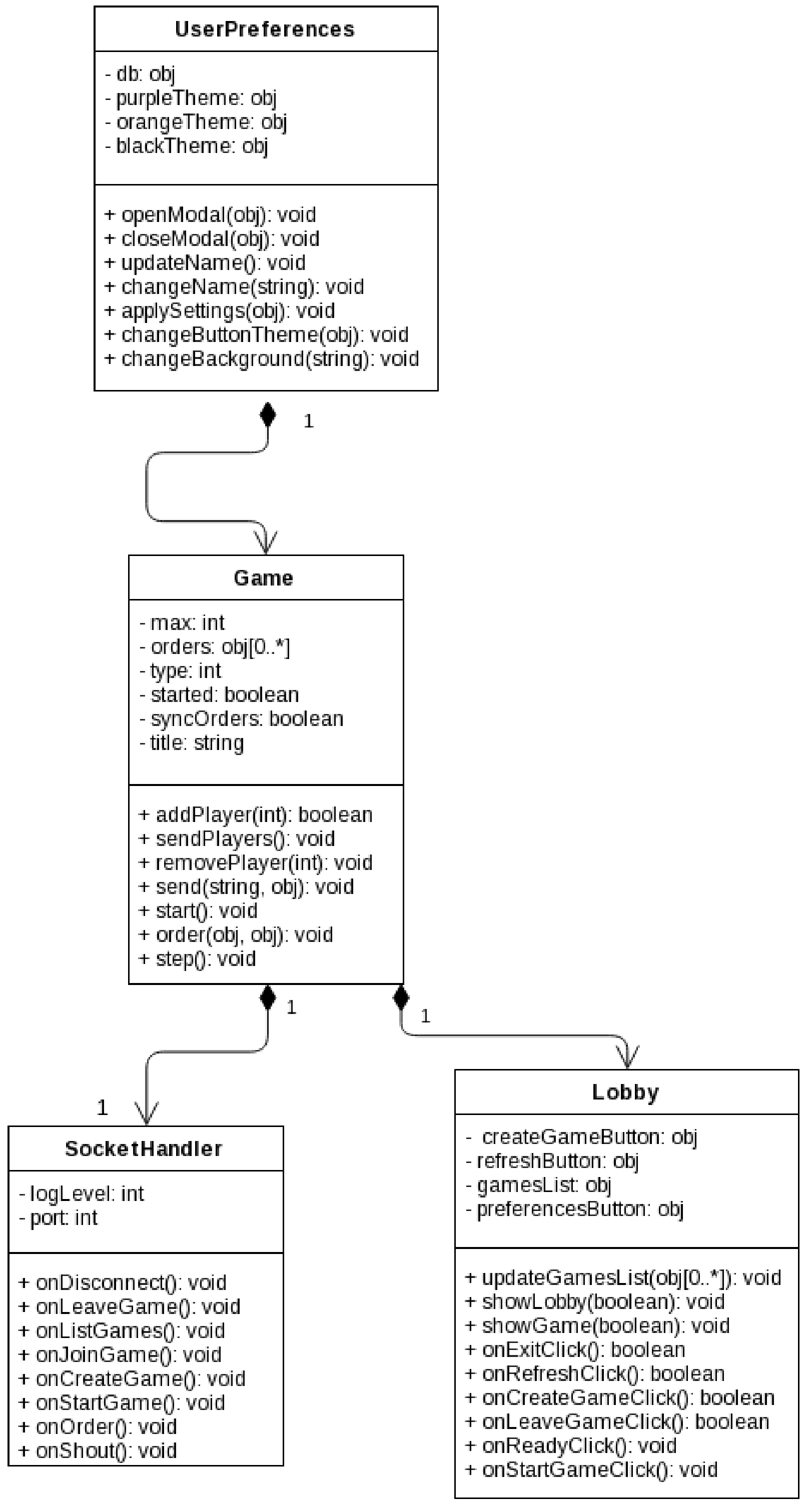
Reset Number

While playing the game, the user clicks the big circular button with the target number inside of it (Reset Button). This will generate a new target number for the opponent, deselect a selected number (if penalties apply) and show the changes on the opponent's screen. This can only be used when the game is in play, i.e, after a game has started.

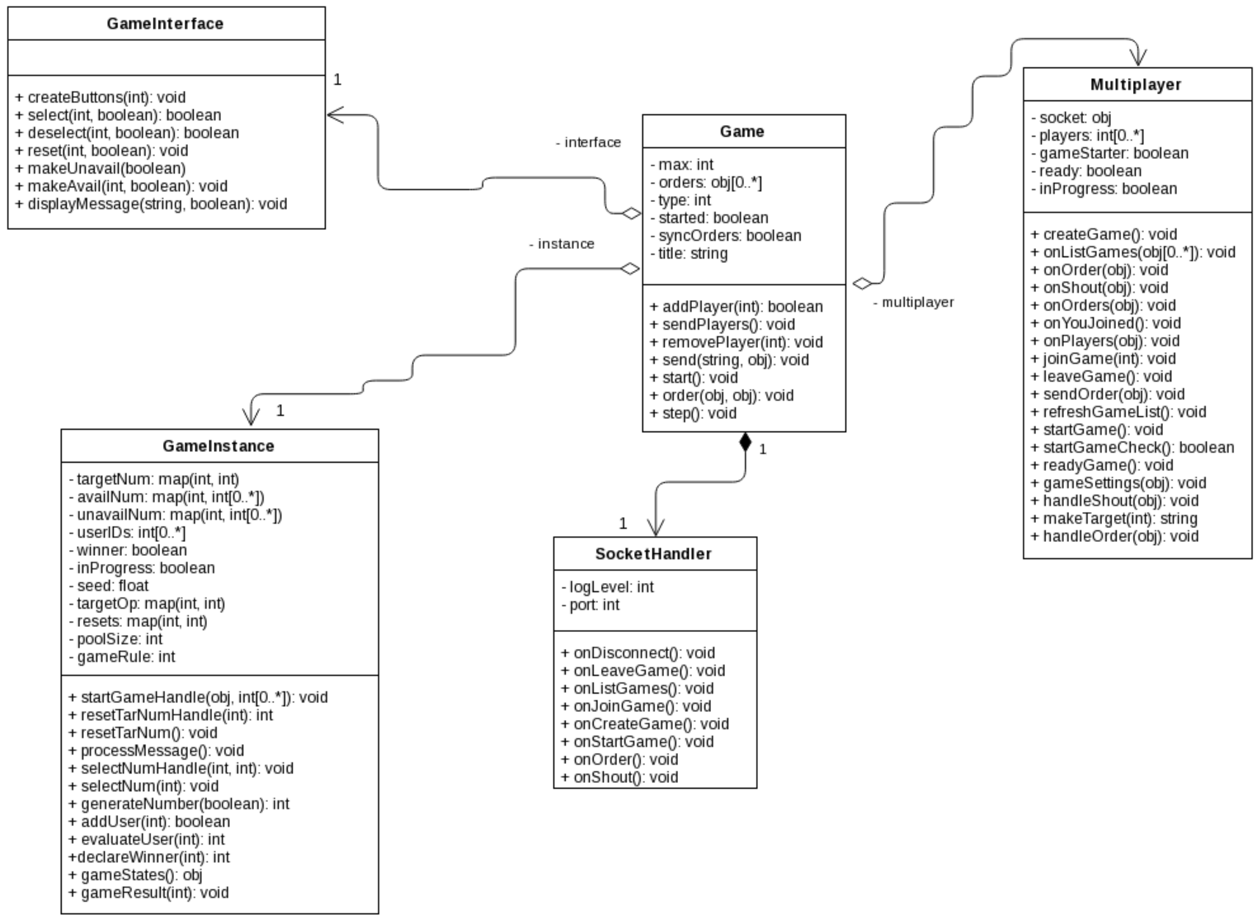
Architecture & Design

UML Diagrams:

In the lobby:



In a game:



Our system revolves around the node.js functionality of the server we implemented. Node gave us a lot of flexibility to do what we wanted with our project. For a lot of tasks our system tries to achieve, we use the server to bounce messages between users to describe many things. Some of these things include game settings, game state changes, and messages. These are all done through the “shout” and “order” systems. For these, “shout” is heard for all users when the game is not going, and “order” is only when the game is not going. Our desired user experience is that they load up the webpage, are able to choose their preferences, create or join a game, and jump right into a game. There is a stats backend that handles wins and losses stored in the users’ cookies.

Our systems main components are the server, the game interface, the game instance, and the socket handler.

Reflections

Matthew Pifko:
I had a great time with this project. The journey from proposal to final product was very fun and exciting for me. Building on top of the skills I earned from the previous course and learning new methods to succeed in group settings was very rewarding for me and I'm sure it was for everyone else in the group. While javascript was not my strongest language I feel like I learned a lot from the members in the group that had experience. I feel like I learned the most about QUnit testing and how to go about testing a big project this semester as last semester we only were in charge of a small project to test.

Patrick Sapin:
I am pleased with the way our project turned out and that it is a working game that people could play. I learned a lot about javascript and large projects with intricate javascript scopes. I also learned some new html/css and a lot about software development in general. I also got experience with Qunit and learned about automated testing for web applications and mocking networked applications. It was a neat project.

Garrett Nickel:
I had quite the learning experience for following team processes, creating diagrams, and developing asynchronous multiplayer web applications. I had no experience with socket.io, and minimal HTML experience beforehand. Now, I feel like I have the basics down for those things. As far as diagrams go, I've never created UML diagrams for large-scale projects that I was a part of like this. It was interesting to see the transition from diagram to task creation and implementation. Developing in a website in a team is something I have minimal experience with, as Jenkins plugin development rarely had conflicting HTML merges. Here, a lot of issues at merge-time arose and I learned it was important to communicate amongst other tasks on which aspects of certain files were being edited. Overall, I learned a lot of web programming and testing, specifically QUnit, as well as how to efficiently work in a team to create a web-based application.

Matthew Balsamo:
I thought this project was a wonderful learning experience. Before this semester I had virtually no experience doing web development. This project helped me to learn how to write good Javascript code and taught me good practices for web development in general. This project really sparked my interest in web development and I look forward to using socket.io to try out some personal projects in the future. I felt like the whole challenge of synchronizing client side representations with minimal information stored server side was actually very rewarding. Also, just the exposure to handling things asynchronously really helped

me to realize a whole side of development I hadn't ever really thought about before. It was also interesting to work on such a compact system in a group. Due to the fact that much of the development had to happen in the same files as things were evolving, we had our fair share of merge conflicts. Learning to stay on top of our tasks so that we had adequate time to resolve these was a valuable lesson. Overall, this project was just a great learning experience in terms of both coding and testing Javascript as well as working in a group to deliver a web-based project.

Austin Mei:
Overall, I had fun with the project. I learned quite a bit about writing "good" code and and picked up many new skills. Although I had worked on web applications before, I was mostly involved in doing work on the back end -- thus, I learned a lot about Javascript and HTML throughout the entirety of the project.

Joel Einbinder:
I wrote a lot of code, which is always a good thing. I hadn't even written test code for a game before, and it was a good experience learning about how to do that in an effective way. I was also able to clean up a lot of my frameworks which we used for the project.

Armaan Rai:
Working on this project was a great learning experience. I was able to better hone my overall web development skills while working on the different user stories. I had a lot of fun working on the more backend-y tasks and learning to use socket.io and Node.js. I had never done any game development before, so I learned a lot from the whole design process too. I also really liked working in the Agile dev process. Working in pairs, writing modular code and Qunit testing gave me a better insight into how big web projects are handled. Our team was really great. I was able to learn a lot while working with other the members.

Ran Crook:
I had fun working on this project. Our group was able to coordinate the work effectively. Over the course of the project, I was able to expand on my JavaScript knowledge. I acquired skills that I believe will be beneficial to me in future endeavors. Working in pairs helped me reflect on my own coding process. Dividing the work between pairs also pushed me to write better code (self-documenting, modular), since I knew other people would be relying on it. Overall, working in a team and having to coordinate our schedules was a good learning experience.