

The Report of Draw and Guess ver. Real-Time

Application Prog Using Java - CSCI 4963 - 01
Final Project

V 1.0.2

Yuetian Chen, Kevin Xia, Jeff Li
2022.08.21

Contents

0	TL;DR	2
1	Introduction	2
2	Implementation	3
2.1	Model	3
2.1.1	Game Model	3
2.1.2	User Model	3
2.2	Controller and Protocol	4
2.2.1	Config File	4
2.2.2	Connection between multiple clients and server	4
2.2.3	Main Game Logic	5
2.3	View and GUI	5
2.3.1	Design Methodology	5
2.3.2	Low performance overhead drawing board experience	6
2.3.3	GUI customization	7
2.4	External Utilities	8
2.4.1	Read Data, Data Representation, and Data Encryption	8
2.5	Documentation	9
3	Future work	9
4	Collaboration	10
5	Acknowledgement	10

0 TL;DR

Draw and Guess ver. Real-Time is an online word-guessing game. In the game, we form groups of 2-8 players and the system will designate one player as the painter according to the order of connection and randomly search for a word in the dictionary. The player needs to use the mouse to draw the specific word given by the system. Meanwhile, other players will try to guess the word. At the end of the round, another player will become the painter and the system will also give a new secret word as a reciprocal for the player to score.

Yuetian Chen was responsible for the project planning, GUI design and GUI implementation. Kevin Xia worked on the network, model and controller. Jeff Li worked on the dictionary, protocol, user manual and test manual. Everyone in our group was involved in the testing work. All of the content was implemented in Java.

1 Introduction

"Draw and Guess" is a very popular online guessing game nowadays. In a game, the system will provide players with multiple keywords to draw and create. The player then needs to use the mouse or touchpad to draw the specific word given by the system, and send it to the next player to guess after the drawing is done. At the same time, the previous player will also complete the drawing and hand it over to you for guessing. In the end, this error of information transmission often forms a humorous and funny misunderstanding, thus achieving the entertainment of the game.

However, in this respect, the existing rules of the game require the user to answer questions in multiplayer mode asynchronously (i.e., you have to finish the painting before you can answer the question), which slows down the pace of the game and makes it boring because of the repetitive work (one needs to paint multiple paintings to complete a round). So we've improved the game to make it a real-time sync.

Yuetian Chen creates some planning documentation to help us draft and organize the project's features before working on the project. This helped us think about how the app could be set up and also helped us to identify the classes we would need to create, the features that would be absolutely necessary vs ones that could be done if we had time. In this report we will focus on the implementation details of the various parts of the program and the establishment of the rules of the game.

2 Implementation

In our project, we make use of MVC model, where we separate the game model, UI and control into different part. In this way, changing implementation of each part won't affect each other since we use the same interface to send parameters to each other. In our main controller, we stored the model and UI object and wrote plenty of connection methods to allow UI to access the information it needed to show without knowing any information on what model structure is. Furthermore, model also can trigger an update on UI when the data change just by sending the data back to the controller. By using MVC, we can easily work on our own part and put them together at the end easily.

2.1 Model

In this section, we will talk about the model we use in our project. Through our implementation, the main goal is to separate server part and client part. We want the server to process and determine all the data and client will just receive and store the data. This will not only improve the performance on client side,

2.1.1 Game Model

For our game model, we separate it into client part and server part. The reason for that is client only need to know just enough of information they need to know to prevent too much data calculation that might desynchronized the data. In client model, it keeps track of all game information such as user list, game status and chat history. All data operation will be executed through data pack from the server. In server model, we keep track of the remain points to give to the next guesser and the guessing word dictionary because the clients do not need these data to run the game. Furthermore, all game mechanic handling such as word guessing and switching between game state are also handle in the server model. They will all be processed on server and then sent the result back to the client.

2.1.2 User Model

Within our game model, we contain a list of users in the room. On client side, we only store a list of client user model with their name, current score and their assigned ID. However, on server side, we store more information such as their player status and whether if they have guessed out the word.

2.2 Controller and Protocol

In this section, we will talk about the main controller and how it handles the running of the whole program. The controller is the core of the program as it connects all the other part together into a complete program.

2.2.1 Config File

When user first launch our game, the default config file will be generated into our root directory of the project. Even though there isn't much custom things user can change for their preference, having a config file allows us to store data that we can use in the future. First of all, users game name will be remembered after they first join or create the game. Therefore, they don't need to enter their name repeatedly for further different game. Furthermore, the address and port they have joined last time will be recorded. In this way, user can play with their friends' multiple times easily without need to re-enter the server information. If user is a host, the dictionary file will also be stored. The config file allows us to carry out the concept of user friendly, which we keep complex process of repeating entering data as a single time only. Therefore, the user can spend more time to play with their friends instead of checking the address every time they open the program.

2.2.2 Connection between multiple clients and server

In our game, as we want to give user a clear visualization of what is going on, we use multi-threading to handle the network connection part. The client is same as what we have done for online battleship game. The client will have a separate thread to try connecting to the server. If connection failed, user will be sent back to the setting panels and they can check if they input the information correctly. If connection success, they will enter the room and wait for game to get started. For server, since we need to handle more than one client, we need to have one thread for each client in order to communicate with all clients without blocking. As implementation, we use a thread to run the main server socket, where we listen the port and accept all the connections from the clients. Each time we receive a connection, we will store the socket into our server and create a new thread to read the message from the client. Therefore, we can react with any message we receive from the clients without blocking any part of our programs. Server can chat in the room while waiting for other players to join. One challenging part of our project is dealing with synchronizing data between different clients. To do that, we make use of protocol technique. Server will receive the data pack from the client and decode them through the protocol. After server determined that the data pack is valid, the command will be sent back to controller to do further calculation. As a result, most parts

of the logic processing of game state are actually happening on server and the client will receive data pack each time server determined that they need to get a notification on things change.

2.2.3 Main Game Logic

When the player first enters the room, they will be notified with a data pack from the server, which contains all information of user currently in the room. The player's chats will be first sent to the server and then distribute back to every client in order for all clients to receive the information. When the game began, the server will send a notification data pack and all clients will update their game state to running. During the game, a timer will be shown to tell user how much time they have left for drawing or guessing the word. When the player sends a message, it will first be processed by server to see if they already guess the right word. If they do, the message won't be broadcast to other clients since we don't want the user to spoiler the answers. If they don't, the message will then be compared with secret word to see if they get the right word. If the words match, a score data pack will be sent to all clients to let them know that someone guess out the word and earn the score. If words do not match, the server will just broadcast the raw message from the client. If all guesser guesses out the answer or the time run out, the game will end and the score will be added to drawer depend on how good they make others guess out the word. However, if all players guess right, it means that the drawing is too obviously and there's no challenging in guessing the words, which we will not give any points to the drawer. Therefore, the graph should be challenging enough while some people are still able to find out the meaning. Since our project is kind of party game, there's no win and lose and no ending of game. We still provide a scoreboard for players if they're interested in how much scores they earn.

2.3 View and GUI

In this section, we will focus on the view and GUI parts of this program, which includes the use of Euclidean distance-based compensation for drawing point medians, the custom interface's resolution-based auto-alignment feature, and a basic design outline presentation.

2.3.1 Design Methodology

First of all, to ensure the overall design is manageable and easy to implement, we chose to use pixel style as the basis of the design language and add game content on top of it, mainly because Pixel style has sharper edges and is more

suitable for Java swing's rectangular JPanel default borders The pixel style is consistent with the way we implemented the drawing board We planned to give the user a consistent and intuitive game experience through a sensible design language and a unified methodology, and in this project we identified

- Uniform color design
 - Background and body color palette (#358587)
 - Emphasis colors (#F4A896)
 - Character colors (#FDFDFD)
- Consistent font management
 - Title and important content (Gore Regular)
 - Service text (Windows command prompt)

We hope to add details to the design language based on this design solution in future development, and to bring a more intuitive design to the user without conflicting with the previous design.

2.3.2 Low performance overhead drawing board experience

In the implementation of the drawing board, we plan to use an 80x80 square matrix to represent the drawing area and use the `mouseDragged` event under `MouseMotionListener()` to track the mouse drawing path. By remapping based on the screen resolution, we can get the exact matrix item corresponding to the current cursor position and return it to each client's `PaintComponent()` for real-time content drawing.

However, in the process of implementation, we found that due to the limitation of Java swing itself, the tracking frequency of drawing events is highly dependent on the user's platform, the number of threads currently available and the operating system. This made it impossible to effectively track and draw the complete mouse path during the drawing process. This leads to broken touches during the drawing of lines, as shown in the Figure 1 in the left, which greatly affects the user's gaming experience.

In this regard, to solve this problem, we propose a compensation algorithm based on Euclidean distance. Specifically, for the current two points in 2 dimension space $(x_1, y_1), (x_2, y_2)$, we hope we can construct a linear function $y = kx + b$ and use it to traverse to get the coordinates of the point that should be filled between the two points. For the slope of the current linear function k , the two points are known and we can use the following formula to obtain it.

$$k = \frac{y_2 - y_1}{x_2 - x_1} \quad (1)$$

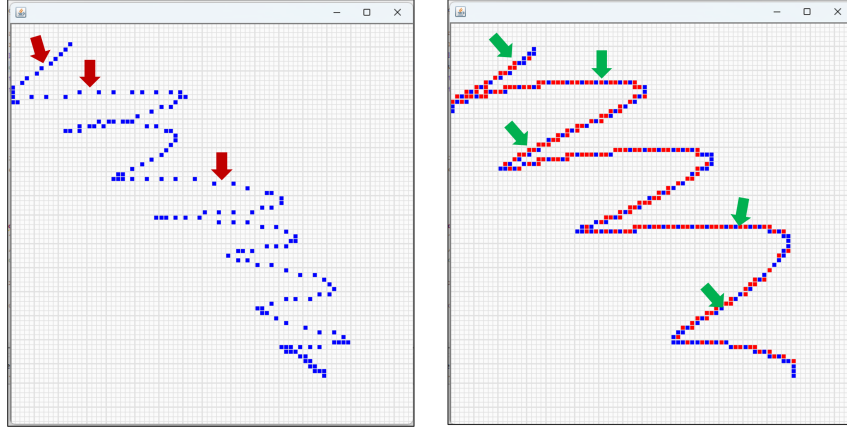


Figure 1: The comparison result of implementation, where the left graph shows that by using the default java listener implementation, the dot generation frequency cannot guarantee its consistency that causes the breakpoint, while on the right, by using the new implementation, we generate the compensation showing in red dots that fix this issue

Similarly, for the deviation, we can also obtain the following equation.


$$b = \frac{x_2 y_1 - x_1 y_2}{x_2 - x_1} \quad (2)$$

Experiments as presented in Figure 1 show that with this algorithm, we can effectively track the high-speed moving cursor dragging behavior and compensate for the lost data points to a large extent. This results in a smooth end-to-end user drawing experience. We hope to further complement this algorithm in the future. In this regard, by using a Bessel curve-based drawing logic, we can further simulate the user's mouse movement trend to optimize the drawing logic and create more coherent and beautiful line segments.

2.3.3 GUI customization

After implementing the design in the program, we found that using the basic functionality of Java Swing did not perfectly reproduce the design intent. Therefore, we made full customization of the content, which included

- Custom cursor

We have implemented global cursor customization within the window and extended the two states to improve the user experience. In the non-current user drawing round, we set the mouse  so that the user knows the current control state of the drawing board.

- Custom button rendering
We have set custom buttons on the welcome screen of the game as well as in the rooms. For the former, we made it more playable by stretching the button and using `setText` to customize the rendering state of the button when the mouse passes over it. For the latter, we removed the background of the button and implemented an `ImageIcon` exposure utility to allow the user to freely select different panel configurations.
- Custom transparent background JPanel
We changed the rendering logic of `JPanel` and removed the default background on top of that. With this implementation, we can unify different elements of the interface with more efficient background colors and patterns.
- Custom vertical display timer
We have implemented an adaptive vertical time display interface based on `Font Metrics`. By simulating the number of horizontal pixels of the current character on the corresponding screen, we can get an adaptive timestamp that will always remain 8 pixels down from the top left corner of the screen.
- Custom interactive background
We have changed the rendering logic of `JPanel` in the welcome screen, it will render the dots for decoration at the bottom in real time, and will move in equal proportion with the mouse movement.

2.4 External Utilities

In this section, we will discuss the format of the input file and data packet, and how we ensure the security of the data.

2.4.1 Read Data, Data Representation, and Data Encryption

We choose to use a .csv file as the dictionary input file. The structure of the file will be

Category, word, word...

By doing this, the user can easily customize their own dictionary for this game, which will help make the game interesting for different users. The data pack is formatted as

CATEGORY#ACTION#arg1#arg2...

For example, the message data pack will be

DATA#MESSAGE#message

Those data packs are sent as strings, so we need to encrypt the data to prevent cheating. At first, we chose to use Unicode as an encrypted format. But the Unicode is formatted as `\u1234`, and only the alphabetic character can be transferred into Unicode, which will give us a lot of trouble if the user is typing or `#`. We finally decided to use Base64 as the encrypted format. Base64 can convert all the characters and it has already been implemented by Java.

2.5 Documentation

- README

In README, For a user who wants to get use of the project, we listed all prerequisites for building and running the project with automation scripts.

- Test Manual

- Test scenarios & Test plan

As the code for the project was created, Yuetian Chen, Kevin Xia, and I tested and recorded our findings. We based our test scenarios on the use cases and user scenarios we wrote from the beginning of the assignment. I wrote the testing manual to include our testing strategy, testing scenario, and test cases.

- Test Case

The test cases for our project are mostly functionality tests. Functional testing follows the workflow of a normal user of the Draw and Guess application. The user will enter the required network information and dictionary file to create the room, at which point other players will join the room and start the game. Therefore, the test will focus on the interactions between these players and check that the functionality guaranteed by the application works well during these interactions. For example, when starting the game, new players will not be able to join the room. If the username input is empty or the host has not added a dictionary file, a warning will pop up. And the processing after the host or the player exits the room, etc.

3 Future work

Given the size of the project, we still hope to continue as an open source project and introduce new content on an ongoing basis after this course project ends. In

this regard, one feature that is still planned is the introduction of a scoreboard. Right now our scoring system is still implemented in the model layer, informing the user with the simplest and most concise information through Chat. In this respect, I think we can refine the GUI more and implement an interactive scoreboard to let the user know the current score and the number of users and the names of each person in real time.

In addition, we also want to make the game have more extensive support for high resolution. Due to the lack of testing tools, we are still lacking in multi-resolution support. This is reflected in the fact that on some Mac OS side, some panels do not display properly in certain situations. In this respect, getting the current screen resolution by dynamically is a potential solution.

Through this project, we hope to be more familiar with the full process of java end-to-end implementation and gain experience in developing complete projects.

4 Collaboration

Table 1 shows the division of labor among the various aspects of the project. All were given essentially the same amount of work. No external personnel and no related organizational bodies were involved in the development. We did not use unauthorized code from others as part of the implementation.

5 Acknowledgement

This project is part of the application prog using java course. We sincerely thank Prof. Konstantin Kuzmin and all the Mentor and TAs for their tireless efforts and help. Without you this project would not have been possible. We would also like to thank our roommates for their utmost patience and sanity while we discussed and implemented this project overnight. :-)

Task	Person in charge
Project Planning	Yuetian Chen, Kevin Xia
GUI design and implementation	Yuetian Chen
GUI implementation	Yuetian Chen
Server-side design and implementationc	Kevin Xia
Model	Kevin Xia
Controller	Kevin Xia
Dictionary parsing	Jeff Li
Protocol encryption & extensions	Jeff Li
User manual & test manual writing	Jeff Li, Kevin Xia, Yuetian Chen
Model Testing	Kevin Xia, Yuetian Chen, Jeff Li
GUI testing	Yuetian Chen, Kevin Xia, Jeff Li
L ^A T _E X	Yuetian Chen

Table 1: Collaboration Information. We are also responsible for the corresponding part of this report