Adaptable RPS

General User Manual

1. Introduction
2. Game Admin Instructions
   1. Introduction - Admin
   2. Preparing for a Tournament
      1. Server GUI
         1. Finding/Assigning IP
         2. Assigning Port Number
         3. Running Server
      2. Game Controller GUI
         1. Connecting to Server
         2. Assigning Tournament Type
         3. Assigning Game Type
         4. Other Pre-Tournament Features
3. Player Instructions
   1. Introduction - Player
   2. Preparing for a Tournament
      1. Player GUI
         1. Assigning Player Type
         2. Connecting to Server
         3. Other Pre-Tournament Features

# Introduction

Welcome to the AdaptableRPS project! The goal of this project is to provide a fun and flexible environment for those new to programming to learn object oriented development (OOD). This project supports many ideas that most students are introduced to while learning OOD concepts such as inheritance, loose coupling, interfaces, etc…

Developers are allowed to create their own player AI, tournament design, and/or game type by simply implementing the respective interface/abstract class. The following pages will provide an overview of how to setup and play a tournament with multiple people on different computers over a local network.

Throughout this manual, figures will be referenced for clarity when instructions are provided. It is possible that the current graphical user interface (GUI) you are using will look different that what will be in the manual version you are reading now. Don’t panic! If the GUI version you are using was in a release branch then it has been tested and should operate as expected. The buttons, labels, and console window of the GUI is meant to be as descriptive as possible to the user so there shouldn’t be much effort in understanding a new feature. If you are absolutely confused, simply reference the commit messages of this directory for information on what was changed and why,

Please keep in mind that this project is growing and ever changing so the version you are using now may not be the most up to date version so please check back at the original GitHub site often to ensure you have the most up to date stable version.

Game Admin Instructions

# Introduction

The following information is focused on using the game administrator and game server. As a game admin, you do not have to implement your own Player class as all features available to the GUI have already been applied in the GameMasterClient. If you feel the need to modify GameMasterClient (and subsequently, GameMasterGUI) it is encouraged that you create your own as to avoid damaging the tested versions and forcing a reset on your Git repo. It is also easier to merge in a new file than to review each changed file in GitHub should your ideas prove useful to the next release.

The game admin is a client with privileges that other implemented Players do not posses, but it is still a client type. The running server is designed to run on its own without the admin performing any actions, but the admin is required for the tournament to proceed. The exact order to setting up and running a tournament isn’t as important as much as understanding what is required to do so. This understanding will be the key to operating a successful and fun tournament.

Operating the server itself takes very little understanding and doesn’t require any interaction with the GUI once it has started running. Modifying the server, however, can prove difficult and should only be done so if a severe error has occurred.

# Preparing for a Tournament

Before starting a tournament, the Player AI’s must be fully implemented and should be inspected (and possibly tested) by an experienced developer. It is assumed that the person(s) running the tournament are experienced.

This project already includes multiple games but new games can be implemented (and it is encouraged to do so) should those with the will and skill have the urge to do so. The GUI’s are designed to handle almost any and attempt to respond with a helpful message so explicit error raising is suggested in all forms of midfication excluding the GUI.

Any game implementation should be placed in the “AvailableGames” directory. The rules are similar for those implementing a new tournament type except that these implementations should be placed in the “AvailableTournaments” directory. The current version of this project can not recognize the implemented game/tournament unless it is in the respective directory. Please note that any custom games/tournaments should be thoroughly tested before using in a public setting to address bugs.

## Server GUI

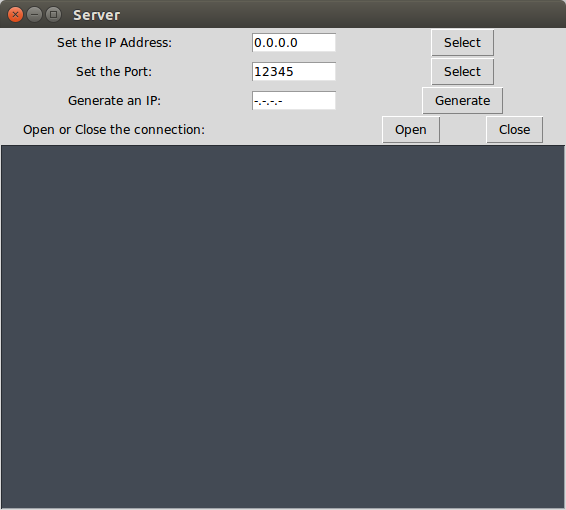


Figure 1: Server GUI

### Finding/Assigning IP

All clients (including the game admin) must connect to the server for any game features to operate. The user must assign their unique IP (address that is not "127.0.0.1"

) to the server as well as an available port number. If the server portion is running on a Linux distribution (this is recommended and Ubuntu is preferred but not required) then the “Generate” button will automatically locate your unique ip and assign it as the host address for the server to use for connection. If the “Generate” button doesn’t work for you, you are still able to enter your ip address should you be able to locate it yourself. This address is entered into the “Set the IP Address” box and the “Select” button should be clicked after the IP has been entered to assign it as the value to use.

### Assigning Port Number

The default port number is “12345” (as shown in Figure 1) as this port is usually open for use. If this doesn’t work, it is suggested that all programs running that aren’t necessary at the time should be stopped and to attempt another serve. Once a stable port value is located, it should be written down and saved to avoid future debugging.

### Running Server

When the user is ready to start running the server, they should click the “Open” button. This will force the server to attempt to open a connected with the current input values. If the information entered is valid, a small message will be printed to the GUI console and the server will run. If there is a problem with this information, an error will print to the GUI console and the server will remain closed.

Once the server is running, it takes a forced key press to stop it. If you are running this in PyCharm (which is suggested) pressing the “stop” button will kill the server. At the time of writing this manual, the “Close” button on the server GUI doesn’t function properly.

Please note that forcing the server to close while clients are connected will result in an error client side though no permanent damage will be done.

### 

### 

## Game Controller GUI

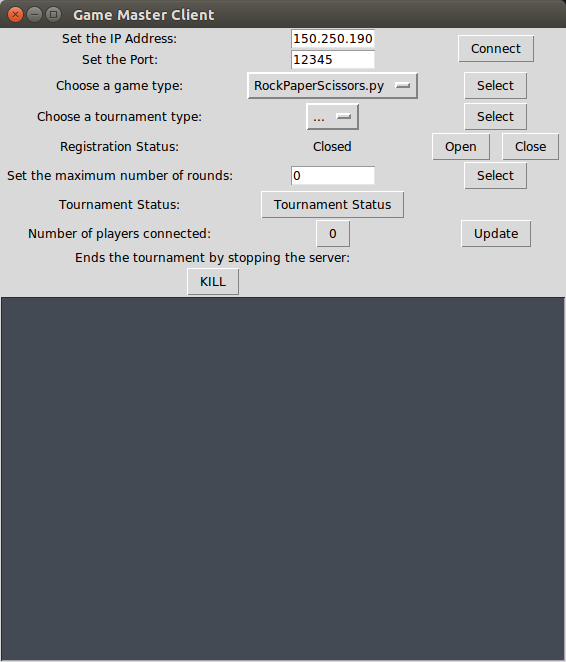


Figure 2: Game Master Client

### Connecting to Server

The user should request the proper IP address and port number from the server GUI. Once this information is received, they must be entered into the “Set the IP Address” and “Set the Port” boxes respectively. After the user is sure the information entered is correct, the “Connect” button should be pressed and a notification should display in the GUI’s console determining whether a valid connection was achieved. Until a valid connected is established, the features of the GUI will not function as they are commands submitted to a running server.

### Assigning Tournament Type

The drop down box next to the label titled “Choose a Tournament Type” should have all of the tournaments in the AvailableTournaments directory listed here. Selecting one of these tournaments should be performed before actually starting the game and shouldn’t be selected once it has started running. Once you select your desired tournament type, you must submit this request to the server by pressing the “Select” button next to the drop down box.

### Assigning Game Type

This functions exactly the same as assigning a tournament type. The drop down box for this is next to the “Choose a Game Type” label. Again, the respective “Select” button should be pressed to assign this game to the server.

### Other Pre-Tournament Features

It is suggested that the game admin perform the pre-tournament features in the following order:

1. Set the maximum number of rounds for each match
2. Set the maximum number of players who can register to the tournament (coming soon).
3. Open registration
4. Request that players begin registering to the tournament now.
5. Verify the number of currently registered players in the tournament.
6. Close registration after all valid players have registered.
7. Players can now submit their moves (see Player instructions)

Player Instructions

# Introduction

The main feature of this project is how creative developers can be when creating their own player AI. As long as the functions are implemented in the custom player correctly, the dev can have as many other functions programmed into their player as they like to assist with logic. This AI will not have any outside assistance once it has started submitting its moves so all work must be performed in the implementation phase.

After the player is implemented, it should be added to the “AvailablePlayers” directory on the machine that the respective developer will be using for the tournament. If this can not be performed ahead of time, then the developer should push these changes to their own GitHub repository and pull it down on the machine. Alternatively, a flash drive could be used for transport.

# Preparing for a Tournament

The player implementation should be designed for the respective game to be played; ie: don’t make a “one size fits all” player. The developer should know the rules of the game and program to counter the opponent’s moves. Once all players have been implemented, it is suggested that all player AIs are inspected by a more experienced programmer to ensure that errors are caught before game time as a debugger isn’t built into the GUI. All players must be placed in the “AvailablePlayers” directory for selection in the GUI.

## Player GUI

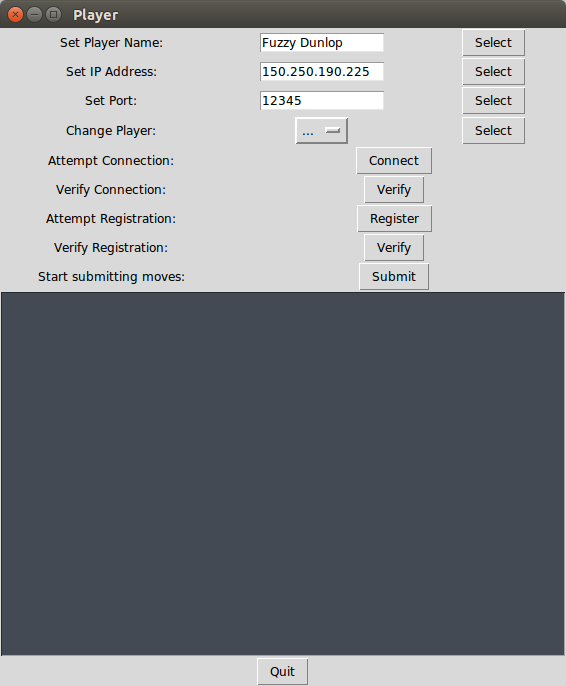


Figure 3: Player GUI

### Assigning Player Type

The player type should be chosen according to the game to be played in the current tournament. The drop down box next to the label “Change Player” will list all players that are in the “AvailablePlayers” directory and can be selected by clicking the desired player. Clicking the “Select” button next to this box will assign the player type. Note that this should only be performed before or in between tournaments, not during.

### Connecting to Server

In order for this client to connect to the server, it must have the server’s IP address and port number. Both should be obtained from the game admin and entered into the “Set IP Address” and “Set Port” boxes respectively. The select button next to each must be pressed to assign it. Once instructed by the game admin, the player should then press the connect button to attempt to connect to the server. They should receive a message to verify if the attempted connection was successful. Still, the “Verify” button next to the “Verify Connection” label should be pressed to make sure that the connection is consistent.

### Other Pre-Tournament Features

1. The player should set their name in the “Set Player Name” box and press the “Select” button to submit this name. Note that names with underscores are not supported at this time.
2. Upon instruction that the tournament has been opened, the player should press the “Register” button to register their player in the current tournament. After a successful message was received, the player should verify that they have been fully registered by pressing the “Verify” button next to the “Verify Registration” label.
3. After all the previous instructions have been followed and the game admin has stated that moves are being accepted, the player should then press the “Submit” button next to the “Start submitting moves” label to allow their AI to being playing against the other players.

# 

# 

# Contributions:

* Dr. Ganesh Baliga
  + GitHub: <https://github.com/geebzter>
  + Website: <http://elvis.rowan.edu/~baliga/>
* Paul Council:
  + GitHub: <https://github.com/PaulieC>
* Anand Patel:
  + GitHub: <https://github.com/patela29>