1 Introduction

In the past you may have played the game Capture The Flag on a grassy field; the field is divided between two teams, each with a flag, and as the game progresses, each team attempts to “capture” the opposing team’s flag by crossing to the other side of the field, stealing it, and returning to their base. The game is over once a team has scored some number of points. This project is based on the real-world game, although our rules for the game are somewhat different and are described in this specification.

Your task is to write a simulated version of this game that allows both the user and the computer to play against other players on the network. You are to implement a program that handles games that pit a local human player or machine player against remote player(s), which can be either human or machine.

2 Gameplay Overview

Upon startup, the server initializes the play area with two flags, each at a base, a set of cells owned by each of the two teams, and special cells described below. The play area can be thought of as some region of cells (or squares), each of which can contain at most one player at any time (except for jail cells). The play area need not be rectangular. Upon connection, the server assigns each player, which can be either a human player or machine player, to a team or honors a player’s request to join a particular team. Additional players may join the game at any time before its completion.

Since the goal of the game is to capture the opponent’s flag, players must travel from their cells, where they are considered safe, to the opponent’s cells, where they can be tagged. If a player is tagged while on the opponent’s side of the play area, they are sent to jail by the server, where they remain until they are rescued. In addition to regular cells, the play area may contain safety cells upon which any player can rest for a short duration and be considered safe. In addition, the play area may contain impassible wall cells through which no player may travel.

If a player reaches the opponent’s flag without being tagged, the flag may be harvested (by entering the cell containing the flag) and carried. While carrying the flag, a player’s movement rate is reduced. However, the flag may be passed to other players. The flag may travel faster while being passed (thrown) than while being carried by a player. If a player is tagged with the opponent’s flag, the flag is automatically returned to its base. If the flag is passed and no player receives the pass, the flag remains at the cell at which it landed. Once the opponent’s flag has been carried to the player’s flag, which must be at the base, the flag is captured and the capturing team scores. If the flag is not captured within a certain timeout, it is automatically returned to its base. After a flag is captured, the round is over. At the start of each round, all players must be ready before the round begins. Once a team scores a certain number of points, the server declares game over.

3 Rules

Here the above overview is made more concrete with details about program behavior, gameplay, human and machine players, output, and network communication.
3.1 Program Behavior

Your program’s main function must be in a class ctf. It should be executable with the following command line arguments, each of which begins with a dash and may be used in any order. You may add more command line arguments if you wish. You may find the class ucb.io.CommandArgs useful for this purpose. Optional arguments are listed in brackets:

```
java ctf server          Starts the game and connects to the specified server and allows the user to play the game. Uses the GUI and allows a human player to play the game.
java ctf [-t] -m server  Starts the game and connects to the specified server and starts the machine player to play the game. Uses the GUI to display the game in progress if the option -t is not specified.
-h                       Prints a help message of your choosing.
```

3.2 Gameplay

There are many different aspects of gameplay your ctf must handle according to the rules below.

3.2.1 Start of Game/Round

When the server is started, the play area is initialized and the game is considered paused until at least two players have joined the game and all players in the game have sent the ready message. Players may not move when the server is waiting to start the game or round. The same rules apply for the beginning of each round during gameplay. The server will indicate the beginning of a round by sending a round message.

3.2.2 Play Area

The play area is represented as a grid of cells. Each cell has a team affiliation. Each cell may contain a single player, except jail cells, which may contain any number of players. Furthermore, some cells have special attributes. A cell denoted as a safety cell allows safety for a specified duration, and may be entered by any player if unoccupied. Once the specified duration expires, a safety cell reverts to its team affiliation, and resets its timer once the player moves off of it. When tagged, a player is placed in one of the opponent’s jail cells, in which multiple tagged players may remain. Wall cells may be anywhere in the play area and may not be entered by any players. The border of the play area is automatically lined with wall cells.

3.2.3 Movement

Each player has several movement rates, all of which are specified in cells / tick. One tick is a short time unit decided by the server. The server specifies movement rates of the player in non-diagonal directions, in diagonal directions, and similar movement rates for when carrying the flag. A player must simply give a destination cell and the server will incrementally move the player towards the destination in the most direct path possible. If a player sends a new move request, the server discards the old destination cell and begins movement of the player towards the new cell.

Only one player may be in a non-jail cell at any time and may not be in wall cells. Furthermore, a player may enter a cell that contains the opponent’s flag (which causes it to be picked up), but may not enter a cell that contains the player’s flag. The player is stopped automatically if the player’s next move is into an occupied or disallowed cell.

While carrying the flag, the movement rate is slowed to the specified flag carrying movement rates, which remain until the flag is passed (and is no longer being carried by the current player), the player is tagged, or until the flag is captured.

The flag has its own movement rates when being passed – this rate is faster than the standard movement rate of a player. While in the air, nothing can interrupt the flag’s movement, not even wall cells. However,
the flag can only be in the air for a specified number of ticks, so pass requests that go beyond this limit are disallowed. Once it reaches its destination cell, its movement stops, even if the receiving player is no longer at that cell. Once its timeout decreases to zero ticks, it is immediately returned to its base.

3.2.4 Safety, Tagging, and Jail

All cells that are not affiliated with the opponent’s team are considered safe. When a player enters a safety cell (which is a special cell type as described above), a count-down begins (1 per tick) from some pre-determined value, and at the end of this count-down, the player is no longer safe. To reuse the safety cell, a player must move off of it and reenter it.

An opponent may only tag a player if that player is not safe and the opponent is adjacent to the player or in a cell that is adjacent to a cell adjacent to the player. Once tagged, a player is immediately sent to one of the opponent’s jail cells.

While in jail, the player may not move. Multiple players may be in one jail cell. If a player is carrying a flag when tagged, the flag is immediately returned to its base. A player may rescue all imprisoned teammates by entering any of the opponent’s jail cells from any non-jail cell. Upon being rescued, all imprisoned players may move freely. Any player that is not in jail may enter any jail cell as if it is a normal cell.

3.2.5 Passing The Flag

An important part of our capture the flag game is the ability to pass the flag. When carrying a flag, a player may choose to pass instead of continue carrying. Typically the flag travels in the air faster and can go over wall cells, and as a result, it is often possible to return the flag to the player’s base with passes when carrying would have taken too long.

To receive a pass, the receiving player must be in the path of the flag (while in flight) or in its destination cell – it is automatically received by the player. If it is the player’s flag, it is automatically returned to its base. If the flag arrives at its destination and no player is in that cell, it remains there until it is picked up or it times out. Also, to ensure fairness, during each pass, the flag can only be in the air for a certain number of ticks.

3.2.6 Capturing The Flag

If a player enters the cell with the opponent’s flag, the player automatically begins carrying it. If the opponent’s flag is returned to the player’s base, and the player’s flag is also at the player’s base, then the opponent’s flag is captured and the player’s team scores a point. This denotes the end of the round. If both flags are being carried at the same time, it is impossible for either flag to be captured until the flags’ timeouts are reached.

3.2.7 End Of Game

Players may leave the game at any time, but at the end of the game (once a certain score has been reached by a team), the server declares the game over.

4 Requirements

Thus far we have described what happens during gameplay, but not how the game itself is controlled and how entities in the game must behave – below is such a description.

4.1 Human Player Behavior

You will use a provided graphical user interface to view game progress, and for the human player, to accept actions. The human player must be able to control a single player on the screen using just the mouse.
The human player may click on any target cell to begin moving towards that cell. At any time the player may click on a new target cell to change the destination cell.

If an opponent is on an allied cell, and the opponent is in range, the player may tag the opponent by clicking on the opponent appropriately.

To pass the flag, the human player may click the pass button on a desired destination cell, which passes the flag that the player is carrying, and does nothing if the player is not carrying a flag.

4.2 Machine Player Behavior

You should implement a machine player capable of making legal moves at all time. Furthermore, it should be capable of the following:

- It must be able to find the opponent’s flag, pick it up, and capture it, provided that there is minimal opposition from the other team.

- It must be able to output whether it is possible, either by passing, carrying, or a combination of the two, to capture the flag before the flag’s timeout expires. It if it is possible, it should output the number of ticks required to return the flag along the shortest path. For this output, do not consider the potential moves of opponents. Note that you do not actually have to make your machine player follow this path – it just must detect and output its length in ticks. This output should be written to the file \textit{ctf.log}, in which the tick count or the impossibility of flag capture should be written as a line of text (each time this check is done).

- It must be able to tag opponents and free teammates from jail when appropriate.

5 Communication

A critical part of this program is its communication with the server and other players (through the server). The actual networking components have been written for you, as has the server – you should use the provided classes \textit{CtfMsg} and \textit{CtfSocket} to communicate with the server.

The general principle behind our network communication is as follows: A central server is run on a machine and ctf programs connect to it (using \textit{CtfSocket}). The server maintains a master state of the game, which means it knows everything about all players. For all messages it receives, the server checks legality (by the rules specified above). Each ctf passes messages to the server, which then updates the game state if the messages were legal and notifies all ctf s that need to know about the update. In general, messages that are passed as output from your game to the server are requests that may not be honored by the server; however, messages received by ctf from the server are mandates and must be followed.

To send and receive messages you must use \textit{CtfMsg} which represents an individual message. Use its methods to appropriately build a message to be sent to the server and use methods from \textit{CtfSocket} to send the message. Upon receipt of a message, you may use the accessor methods within \textit{CtfMsg} to retrieve the message’s contents.

\textit{CtfMsg} contains methods that enable you to send and receive the below messages and their parameters. Note that the below “messages” simply indicate the different types of messages that can be sent and received and their parameters – their contents are made easily accessible through methods in \textit{CtfMsg} (meaning you do not have to parse anything).

5.1 Output Messages (to the server)

\textit{ready}...indicates that your player is ready for play to begin. Use this message the beginning of a round or after joining a game that has not started yet.

\textit{move} X Y...requests to move your player to the cell [X, Y]. No guarantee is made that the player will
actually move to the specified cell.

tag N...requests to tag an opponent with name N.

pass X Y...requests to pass the flag to the cell [X, Y].

view X1 Y1 X2 Y2...requests that the current view-window be from [X1, Y1] to [X2, Y2] – the dimensions of the requested view-window must match those given by the server in its gamestate message and must include the player’s current cell.

message S...sends the text message S to the server to be broadcast to all other players. S may contain spaces, alphanumeric characters, and punctuation but may not contain other whitespace or characters.

messageteam S...sends the text message S as with message, but the message is only broadcast to teammates.

quit...notifies the server that you are quitting the game and disconnecting. This should be the last message you send to the server.

5.2 Input Messages (from the server)

gamestate VX VY MS MD MSC MDC MFS MFD R G T S O D...is the first message passed to ctf by the server and contains all information about the game. VX × VY are the dimensions of the viewable window of the play area. MS and MD are the sideways and diagonal movement rates of a player in cells / tick. MSC and MDC are the sideways and diagonal movement rates of a player while carrying a flag in cells / tick. MFS and MFD are the sideways and diagonal movement rates of a flag in cells / tick. R indicates whether the game has already started. G is the winning score required for the game to end. T is the number of ticks / second. S is the maximum duration of safety on a safety cell in ticks. O is the flag timeout in ticks. D is the maximum number of ticks the flag may be in the air during a pass.

cell X Y T C...notifies ctf of the state of the cell [X, Y], with T indicating whether the cell is an allied cell or an opponent’s cell and C indicating the cell type (as defined in CtfMsg). A CtfMsg may compose multiple cell messages into a single cell message for convenience – use its accessor methods to properly retrieve all the cell states that were sent.

playerjoin N T...notifies ctf that a player by the name N has joined on team T.

playerquit N...notifies ctf that a player by the name N has quit.

flag N X Y T S E...notifies ctf that a flag by the name N is at [X, Y] and is owned by team T. S indicates the state of the flag (being carried, in the air, or on the ground), and E is the name of the player carrying the flag if it is being carried.

player N X Y F...notifies ctf that a player by the name N is at [X, Y]. If the player is carrying a flag, F is the flag being carried.

tagged...notifies ctf that this player has been tagged and is in jail.

rescued...notifies ctf that this player has been rescued from jail.

score A O...announces the score – A for the allied team, O for the opposing team.
tick...notifies ctf that the current tick has expired.

message N S...delivers a message with text S from the player named N.

round...indicates that the current round has begun.

roundover...indicates that the current round is over.

gameover...indicates that the current game is over. This is the last message sent by the server.

6 Grading

Each part of this project will be graded separately and then as a single functioning unit. In addition to the usual requirements, your project will be graded on coding style, design (your DESIGN files), performance, and compatibility.

7 Acknowledgements

This was a large project to pull together, so from this summer and last, I’d like to thank Amir Kamil, Paul Hilfinger, Joseph Dale, Nadia Heninger, Melinda Ler, Joseph Davis, Tynia Yang, Adam Gousetis, Erik Klavon, and Terence Tong.