Multiplayer Game Development CSC 631 / 831

Spring 2016

Multiplayer Convergence Game

Design Document

2016-3-13

David Hoff – Team Lead, UI, Client

Harjit Randhawa – Content

Ivan Gonzalez – Server

Gilburt Bui – Integration, Concept

Jason Huang – UI, Client

Haichuan Duan – Database

1. Executive Summary:

This development effort is to create a multiplayer mobile version of the Convergence game to continue the efforts to use Gamification to forward the study of ecosystems. Previous students & teams have developed the single user desktop/laptop Convergence game with a number of ecosystems. The present game allows users try to match a "target" ecosystem parameters by adjusting the biomass parameter for each species in the ecosystem until his/her biomass graph matches the "target". The intent of this game is to help educate users on the dynamics of ecosystems and to use gamification principles to further the system of ecosystems.

The motivation for this development is that the present single user version seems to lack features that will draw the interest and longer-term participation of large numbers of players. As a result, this development effort was started to create a mobile, multiplayer game to attract more users and have them play for longer periods of time.

The goals for this development in approximate priority order are:

1. Add multiplayer capability
2. Enable mobile play
3. Create flexibility playing / scoring modes

These goals will be more fully developed in this document.

2. Use Cases:

A User will Enter the game from the Lobby. The first time they open up the game they will see helpful hints to help them understand the game. Their attention will be set on the center of the screen where they will see their graphs and the sliders that they will have to modify to play the game. After all players hit submit or the time expires the Information will be sent to the server where it shall be processed the new graphing data will be returned along with a flag indicating the winner of the round. At that point the user will see their results on the graph on their screen contrasted with the target graph next to it.

At this point the player may grow curious about how he did compared to the other players and they will be able to swipe to see the results of their opponents in graph form.

In the case that there are no other players willing to play the single player will get to make modifications on a single player mode similar to what exists currently. This will allow the player to grow their skill level allowing them to make better modifications in head to head games. The game will be skill based and the users will know it after playing a few games.

3. Game Object / Concept / Components:

Below are major elements of the game design with explanation.

Multiplayer Version:

The present game is single player only. The server is used in the sense that the user enters the lobby and selects the Convergence game. Also, there the ATN simulate that takes the user input parameters and calculates the new graph data. The goal of this project is to allow multiple players to play together. The game is patterned after poker with betting and multiple rounds. However, the winner of the round is determined differently. It is not based upon random card draws and card skills. Rather, it is based upon making the best choices in biomass adjustments.

Five players maximum per game:

The present thinking is to limit the total number of players in one game together to 5. That is, someone will have at most 4 opponents. Initially, it was proposed to display information about all of the opponents in corners of the players screen. Four corners leads to 4 opponents. However, it was decided that displaying everything on one screen was too busy. With mobile, the user can easily swipe left or right to see the opponents. (There could be one opponent or two opponents per screen, for example.) Therefore, the corners limitation does not exist. However, 4 opponents is probably a practical limit on the number of opponents a person can consider in his/her analysis.

Play is divided into rounds, one parameter change per round:

The game play is divided into rounds. There is a set time that players can evaluate their action for this found and press the “Accept” button to participate in the round. The player’s action is to adjust one of the biomass parameters in her ecosystem. In making the decision she can look at the

results of previous rounds in the form of stack charts. She can also compare the graph of the most recent round versus the target. If she chooses to participate she adjusts the chosen parameter accordingly and presses "Accept". She must do this by the time limit in order to participate in that round. Only one parameter can be changed per round. This is to provide some control to the game and also to increase the challenge. If a player moves one biomass parameter slider, and then moves a second slider, the first slider will move back to its original position. That is, whenever a slider is moved, all of the other sliders are reset to their original position if they had been moved.

Those that participate in the round will have their input run through the simulator and their screens will be updated. Players that do not press the "Accept" button in time will not participate in that round. They can participate in the next round.

The player with the best score, according to the scoring system chosen will receive the pot of money for that round. Her score will be increased by the pot amount. The remaining players will have their score reduced by their bet amount.

Joining the Game from the Lobby:

Presently the Lobby gives the user the opportunity to play the Convergence game. This is the single player version. We will work with the Lobby team to add one or two additional buttons for the multiplayer game. The user will be able to see other multiplayer games in progress. The users will see the player id for those participating as well as the settings for that game session. The settings include the ecosystem being used, how the round winner is determined and the time available to enter your parameter selection for each round. A player can join an existing game session as long as the player count has not reached the maximum. The player's ecosystem will start at the initial point. Depending upon the scoring system chosen, it may not be practical to join a game session that is well advanced. If the winner of the round is the person with setting closest to the target, a new

player will find it very difficult to win a round in a game session that is well advanced.

The user will also have the option of starting her own room. Then, she can start playing as soon as someone else joins the room.

All players use the same ecosystem:

Multiple ecosystems have been prepared for the existing Convergence game. At least 5 are available. The ecosystems range from 5 to 17 species. The team would like to add more ecosystems. The ecosystem will be selected for the game and each player will be using the same ecosystem to keep the play more balanced.

Betting Amounts:

Allowing the players to specify a bet amount makes the game more interesting than if just a fixed bet amount is implemented. Allowing for the players to specify the bid amount does increase the complexity of the client-server interaction and the protocol development. This is because a system has to be developed to equalize the bets from all players, so that they agree upon one bet amount.

The initial development of the game may not include the adjustable bet feature as to insure that the core feature list is implemented and fully debugged this semester. However, in any case, all the “hooks” necessary to have adjustable bets will be implemented in the code from the beginning, make it as easy as possible to implement this feature.

Players continue in game until they run out of money or someone wins:

Everyone has a money balance. Each round that is played, the bet amount is subtracted. Each time a player wins, she receives the amount of the bot. If a player’s balance falls to $0 he has lost and is out of the game. Once someone exactly matches the “target” graph, then the game is over.

However, the winner is actually the person with the highest amount of money.

Determining who won the round:

There are different algorithms possible to determine who won the round. For each round and each player that participates in the round, the simulator calculates a delta between that person's graph and the "target" graph. This delta value will be used in determining who won the round. Three options regarding ways to use the delta values will be provided:

1. Closest to the target: This one is simple. The one with the smallest delta value wins the round.

2. Largest improvement from best so far: The best (lowest) delta value obtained so far is stored. Then the current delta is compared to the the best delta so far. The player with the largest improvement (reduction) in delta wins the round. This helps players that are behind stay in the game. It is possible that someone jumps out to a lead. It can be very difficult to catch this person. Other players keep improving, but lose every round because they cannot catch the leader.

3. Large improvement from best so far, measured as a percentage: This version is similar to the previous. However, the percentage improvement is used to determine the winner of the round. This makes it fairer for the leader. As one get close to the target, it is harder to make large absolute improvements. For example, if someone has a delta of 100. It may be a big accomplishment to get down to 60, which is only a 40 point improvement. However, it is a 40% improvement. However, someone with 1000 can much more easily get down to 950, a 50 point improvement, but only 5%. This third measurement scheme makes it more fair for the person who is converging upon the target.

Money from the Lobby:

There are two options for assigning the player money. One option is that each player receives a set amount, say $1000, when joining the game. Another (better) solution is that the player's money balance earned from the lobby will be used for the game. The Convergence team will work with the Lobby team on implementing this method.

User Instructions / Demo:

The members of the Convergence team found that understanding the principles behind the Convergence game and how to play was a little difficult. There seems to be a lack of document. User instructors or a demo mode will be added.

Setting up the Game Configuration:

A number of parameters are configured for each game session. These include which ecosystem will be used, how much time will be available to enter selection for the round, and which of the three methods is used to determine who won the round. It is proposed that the first person who starts a particular game session will specify all these choices and those will become the rules for that game session.

4. List of Functional Specifications:

- Players shall have account accessibility.
    - Account login and information shall be managed by the lobby.
- Players shall be provided in game currency.
    - Players shall be provided betting currency upon starting a game.
    - Convergence betting game currency does not reflect upon the currency used in the lobby but can be more so thought as points pertaining to that specific game room.
- Players shall use a Betting and Reward system indicating winners.
    - Rewards will be given to the winner upon winning the betting round.
    - Rewards will also be calculated and converted to lobby currency.
- Players shall be able to play a Multiplayer online version of the game.
    - The lobby will be used for matchmaking.
- The game shall be used for Ecological data generation.

5. List of Non-functional Specifications:

- The system shall have a quick and robust matchmaking system.
- The Game shall be accessible on Android devices.
- Player account information shall be managed by the lobby.
- Client shall support fast response time.
- The server shall support at least 5 players per game.
- Game shall run on a MySQL database.

6. GUI Design:

Below are major principles used in the GUI Development.

Each Player will see all his/her information:

All of the information available to the player in the single-player version will be available in the multiplayer version. This include the stacked bar chart that shows progress from previous rounds. Also, as in the single-player version, the user will be able to see the predator / prey relationship for species that she selects.

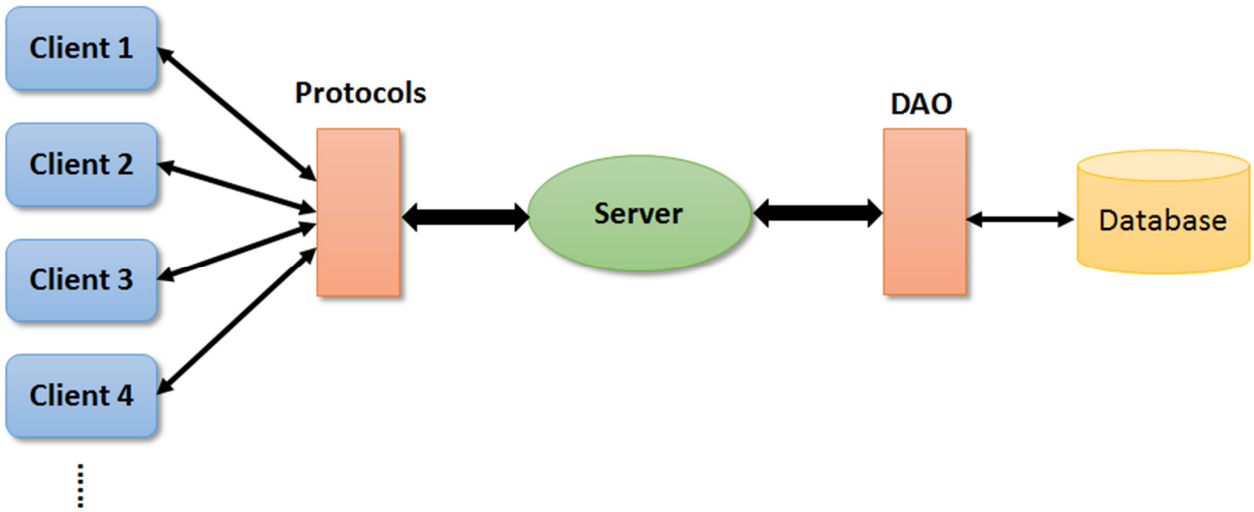
A summary of opponent information will be available to the Player:

The player will also be able to see some of the opponent information. This will help the user decide upon his/her action this round. It will also motivate to play better. Each opponent's progress chart will be viewable. The delta value from the target will be displayed as a number at the top of the chart. Important confidential information of the opponent will not be viewable, such as the settings of her parameter sliders.

Since this game will be optimized for mobile, mobile viewing strategies will take priority in the design of the GUI. The player will swipe to the left or right to view opponents. The player will swipe to the left to view opponents #1 and #2. the player will swipe to the right to view opponents #3 and #4, if all these opponents exists. The opponent player id will be displayed at the top. The graphic below shows how the swiping works. The player is in the middle (in the case of four opponents). In the case of a laptop or desktop, buttons will be on the left and right sides of the screen to emulate the swipe function.

7. Preliminary System Architecture:

Here is a brief diagram of our system architecture:



The major components of the system are discussed below:

Client

The client side are Convergence game apps running on Android devices and laptops. The client C# code will be written and tested in Unity 5 environment. The client side should contain these scripts:

- Main driver script: main loop that drives game progression and coordinates various aspects of functionality.
- Script for accepting player input in each round, including environment parameters and money amount bet.
- Script for communicating with the server by protocol, including request and response scripts. These will be constructed based on communication protocols described in the next section.
- Scripts to draw UI in each turn, including both the player's and his/her opponent's time graphs based on calculated ecosystem data returned from the server, as well as the betting amounts and time remaining.

Server and Protocol

The server will have Java code to accept requests from and send responses to clients. These, along with the clients' communication scripts, will be based on established protocols that specify the data communication format. The protocols follow the request/response format. We have identified at least these request/response protocols that are necessary:

- GameStartRequest: clients request server to start game, get opponent's information and ecosystem model
- GameStartResponse: server responds to set up game & ecosystem
- HeartBeatResponse: server updates time left to bet & close bet window
- TurnSubmitRequest: clients submit betting amounts & parameters to Server
- TurnSubmitResponse: Server returns calculated score
- TurnEndResponse: Server returns turn end signal and winner's score

The server also needs to communicate with database through the DAO layer, which will be discussed in the next section.

Database and Data Access Objects (DAO)

The game will run on MySQL database back-end. We have identified at least these tables that need to be created:

- Ecosystem table: contains species information & parameters to calculate time graph
- Game table: contains each game's timestamp, ecosystem model used, winner's ID, score, winning bets and parameters
- Game_turn table: contains information about each turn in the game, including each user's submitted parameters and calculated results
- Game_player table: contains information about each game's players, including game ID, player ID, player's bets and winning status

The server will access the database for create, read, update and delete operations through the DAO layer.

8. Task List:

This section provides more details on functions and services provided by each block.

A. Server:

1. Respond to Lobby or client requests to see the number of game sessions presently being played. The number of game sessions will be returned.

2. Respond to Lobby or client requests for information about each game session by number. The server will respond to these requests by sending the list of IDs of players that are presently playing in the selected game session as well as the settings of the room such as round win method, round time and ecosystem in use. The response packet will include 5 player IDs with appropriate null values in the case where there are fewer than 5 players in a game session. The first player ID is the initiator of the game session and his/her player ID is the unique game session ID. A player can only play in one game session at a time with the same player ID.

For purposes of retrieving game session information, the game session numbers will range from 0 to n-1 where n is the number of active game sessions presently in progress. This is will be an internal number used between the clients and server only for the purpose of retrieving game session information to be presented to the user to select which game session to join or to decide to start a new game session.

The number assignment is dynamic in the sense that when game session 2 ends, game session 3 now is referred to as game session 2 for the purpose of this protocol. The initiator's player ID is the unique name for the game session.

3. Respond to client request to add a new player to an existing game session. The request will include the game session ID and the player ID. If the game session is full, the request will be denied. If it is not full, the player will be added to the game session. The player's money and all other

settings will be initialized. Notification will be sent to the other players. The notification includes the player ID.

4. Respond to client requests to set up a new game session based upon user request to start a new game session. The player ID becomes the ID of the game session. If a game session already exists with the same ID, the request will be refused. The request will also include the configuration settings for the game session such as round winning method, betting time and ecosystem. Once a second player has joined the session, the game will begin.

5. Send all clients in a game session notification that new betting round has begun. This notification includes the starting time of the betting time counter.

6. Send all clients in a game session notification of updated bet time. This is sent to each client every second. For example, if the bet window is 60 seconds. Then the new betting round notification is sent with the initial bet time of 60 seconds. Then, this update is sent at 59 seconds, 58 seconds, 57 seconds, all the way down to 1 second. The client uses this notification to update the time left on the screen. The last signal sent for that round is sent with 0 seconds. This signals that the round is closed and now the simulation will be run.

7. Receive signals from clients indicating that the “Accept” button has been pressed (before the time limit). The request will include which slider parameter value was changed and the new amount. The server will update its database with this information and note that this player will be participating in the betting round.

The client will not send this signal if it has already received the betting closed signal from the server. The client will enforce this to prevent a bad visual experience on the client side. That is, if the server did the checking on the betting window being closed, then the window could close before the client screen is updated. The player thinks he met the time window but did not.

8. Once the notification has been sent to all clients that the betting window time is 0 seconds (window is closed), the server will wait 1 second for any remaining bets to be received. Then for those clients that pressed “Accept” the simulation job will be submitted with their new parameters.

9. Once the simulator is done, the server will send the simulation results back to each client that participated in the round. The server will determine which player has won the round. Notification of round win/loss status will be included in the response. Also, the server will calculate each player’s new money balance and include that in the response. Those players with $0 balance will be removed from the game. Notification of removed players will be sent to every other player.

10. Notify client of player who has won the game. This could happen when only that player remains because all of the other players have left or been removed because their balances fell to $0. This could also happen because another player (or this player) exactly matched the “target” and this player had the most money. Note: When the “target” is matched, the player with the most money wins even if that is not the player who matched the “target”.

11. Notify client of the player who has matched the “target” perfectly. This message could also be sent to the same client as the one who received the won the game message above.

12. Receive request from client for number of opponents and respond with opponent count.

13. Receive request from client for information about a given opponent number. Respond with opponent information which is listed in the client command section.

B. Client/UI:

A new client will have to be created. It will be the multiplayer client. First a desktop/laptop version will be created and proven. Then an Android version will be created. The existing single player client will remain for those wanting to play the single player version. The starting point for the multiplayer client will be the existing single player client. The additions described here will be added to the new multiplayer client.

The multiplayer client is initiated from the Lobby. It is assumed that in the Lobby the user will decide if she wants to join an existing game or start a new one. This information will be passed to the multiplayer client and referenced by the player ID.

1. If no game session was selected in the Lobby or if the selected game became filled or closed, the client will present to the player the existing game sessions available to join and also the option to start a new game session. To obtain this information, the client must first submit the request to the server for the number of existing game sessions. After receiving that number, the client must request one-by-one the information for each game session. The information returned includes the IDs of the players and the game session configuration information. The first player ID is the unique game session ID. Please see server explanation for more details.

2. If the player decides to join an existing game, submit request to server from player joining an existing game. This request can be passed to the client from the Lobby or selected by the player while in the client. The client submits the game session ID and the player ID in the request submission. Receive response from Server. If Server denies the request (i.e. game is now full from another person joining), report to player and present other existing game sessions with option to join or start a new game session.

3. Submit request to server from player to start a new game session. The client will have to ask the player for the configuration settings for the game

session such as round winning method, betting time and ecosystem. This information will be passed onto the server to configure the new game session. The player ID will be used as the unique game session ID. The game session will begin once a second player has joined.

4. Receive notification from server that a new player has joined the game. The notification includes the player ID. Display to player for certain period of time.

5. Receive signal from server of notification that a new betting round has begun. This includes the starting time of the betting time counter. The betting counter value must be updated. This value is displayed to the user. Also the fact that a new betting round has begun will be indicated by the client.

6. Receive signal from server indicating the update in time on the betting window. The signal will include the number of seconds left. Update the display with this time. The last update for the round has 0 seconds left. The client will update the screen showing that simulation is being done. Once the update with 0 seconds is received this means that the betting window is closed and the client will turn off the “Accept” button and not submit any bet request for this round.

7. When the player presses the “Accept” button the client will submit the parameter slider information and the bet amount (when implemented) to the server. Only 1 slide parameter is changed. The parameter number and the new amount will be sent. This request will only be sent by the client if the bet window is still open. The client will not send a bet signal to the server if the betting window is closed. When betting amounts are implemented, the client will verify that the bet amount does not exceed the money balance before submitting the bet.

8. Receive the signal from the server that the simulation is complete. The server will send all of the information that was sent in the single player version of the game. The server will also send the win/loss status and the client will signal the player of this status. The server will also send the new money balance. If the balance is $0 then the server will automatically

remove the player from the game. The client will notify the player that he/she has been removed from the game. That game session will end. The player will have the option of joining another game or returning to the Lobby. The issue of money balance from the Lobby must be addressed.

9. Receive notice that this player has won the game. This could happen because all other players either left the game or were removed because their balances fell to $0. This could also happen because another player (or this player) exactly matched the “target” and this player had the most money. Note: When the “target” is matched, the player with the most money wins even if that is not the player with who matched the “target”. Notify the player that she has won the game.

10. Receive notice that this player has matched the “target” perfectly. Notify the player. This message could also be displayed with the won the game message above.

11. Add buttons on the left and right side of the screen to allow the player to see the opponent's vital information. For Android this will be changed to swiping to left and right. Presently, the only information shown is:

a. Money balance

b. If “Accepted” pressed and bet amount (when implemented)

c. Color-coded stack chart of delta values in previous rounds

When the user selects this, the client will make a request to the server for the number of opponents present. This will determine what the left & right buttons do. Opponents #1 and #2 will be seen in the left direction. Opponents #3 and #4 will be seen in the right direction. Once the client receives the number of opponents then the client will request information about the opponent according to whether the user pushed the left button or the right button. Once received, the screen will be updated with this information. The user returns to the player by pressing the button to go back in the other direction.

C. Database:

1. Database shall have a table to store all parameters to set up the convergence ecosystems that will be played. Each ecosystem shall be represented by one row in the database. The columns correspond to parameters in the allometric tropic networks model.

2. Database shall have a table to store information about all betting games played, including all the game's id, user's ids, game start time, game end time, winner's id, winner's parameters, scores and money won.

3. Database shall have a table to store each submitted turn in a game, including each game player's submitted parameters, scores and betting amounts.

4. Each table in the database shall have a DAO class as the access interface. The DAO class will perform all database operations such as create, read, update and delete. Server classes directly call DAO classes to fulfill database requests.

5. Database integrity shall be preserved at all times, including when client drops out, session ends prematurely and SQL errors are encountered. To achieve this, database operations shall be properly surrounded with try...catch… blocks and only commits when an operation completes successfully.

D. Integration:

1. Currently Convergence is just a single player game with a Database storing resulting data of the games. We would like to integrate a server that would handle the multiplayer aspect of the game taking care of bets, time limits and parameter changes.

E. Concept:

1. The main concept of the game is to have 2 to 4 players compete with each other to obtain the most credits by the end of the game which is when someone matches the target graph.

2. Within each round each player would take turns putting bets into a pot and changing one parameter of the graph within a time limit. The bets would have a set minimum and maximum that and long with the time limit is set by the player who created the game.

3. After all player submitted their bets and parameter changes, the changes would be compared to the target graph the player with closets matching graph wins the pot.

4. The game would end when a player completely matches the graph or all but one player goes bankrupt.

F. Content:

1. The Game will contain Dialog and Fragments to aid the user experience, this will allow us to give the user updates on their progress, their current balance of credits and money, and perhaps helpful hints or messages from other players without over crowding the screen.

2. The Graphs will be the main focus of the player screen along with the sliders, when the user swipes the screen they shall be able to see the progress of other players towards the target graph.

3 A sample screen has been provided above, this is just a prototype and has already changed but is being used to give an idea of the layout and UI of the game.

G. Other Teams (i.e. Lobby):

1. Add one or two buttons for players to see existing multiplayer games in progress. The player can choose to join an existing game whose player count has not reached the maximum or choose to start a game. The player's choice will be provided to the client that is initiated by the Lobby.

2. The Lobby will also create a notification system to allow players to ask if others would like to join. This may be a simple add on to their existing chat feature wherein a bot is called when a single player initiates convergence.

9. Development Strategy:

It is important to choose a development strategy that both maximizes the opportunities for success as well as the number of new features that can be added. To this end the major tasks with large risks will be identified and implemented in a staged fashion to maximize the opportunity for success. The following major tasks with large risks have been identified. They will be implemented in the priority order listed below:

1. Multiplayer Capability:

   Multiplayer capability is the big part of the project and involves substantial work and risk. The multiplayer capability also complicates the client GUI code. This capability will be proven first.

2. Port to Android:

   Porting to Android is a key part of this semester's class. Therefore, it is a high priority task.

There are additional capabilities that are desired for this development. All "hooks" necessary for these items will be put into the original code. However, the implementation of these items will be done after the above higher priority items are proven. These items are:

1. Adjustable Bet Amounts:

   Allowing the user to select the bet amount will make the game more interesting. It does complicate the game because additional protocols are required to notify and update the bets of the other players is required. Also, an adjustment of the round time will probably be required. There will be a maximum bet amount limit.

2. Exit Game Button:

Provide the player a button to exit the game.

3. Using Money assigned by Lobby:

Initially a set amount of money (i.e. $1000) could be assigned to the player upon joining or starting a game. A better solution, if time permits, is to receive the money from the Lobby and return the money balance to the Lobby upon exiting the game.

10. Schedule:

- 3/13 Complete Design Document
- 3/20 Game Architecture Complete, New Coding Work Defined
- 4/3 All New Code Written
- 4/10 New GUI Display Running (potentially not fully debugged)
- 4/17 First Multiplayer play Demonstrated, (potentially not fully debugged)
- 4/24 Full Functionality Demonstrated, All Known Bugs Fixed, Game screen on Android
- 5/1 Android full functional, Regression Testing
- 5/8 Class Demonstration