

Homework 2 Manual

October 28, 2015

Adrien Sambres
Debasish Guha Thakurta
Ansab Ijaz
Calvin Thanh

Game Setup Steps

- To run the game we need a functional MySQL DB instance running on the machine.
- The DB-Script.sql file will create the required schema. The game uses "root" user with no password to connect to the DB.
- Java 1.8 is required for running the server code.
- Import 'Server' project as an existing java project in Eclipse.
- After creating java project in Eclipse, goto folder named 'src'. Under 'src' folder, open package named 'core'. Under package 'core', run GameServer.java file as a java application. Then server will start listening for clients.
- To run a client, open a command prompt. Type "python <path-to-project-folder>\Client\uid-pwd.py" and hit enter.
- Next run each Client(uid-pwd.py) from the Client directory as you wish for each player.


GAME INTERFACE:

- ❖ Page One gives the user the options to Login or Register.
 - [REGISTRATION] On selecting Register the user will be asked for a username and a password.
 - If the username is free the user will be created in the MySql Database and user will be redirected to login page. Next steps are same if Login option is selected in Page 1.
 - If the username is already taken a Register error will be shown
 - [LOGIN] On selecting Login the user will be taken to the login screen where the user will have to enter a valid user and password.
 - If the login fails a Login error will be shown.
 - If the login succeeds:
 - If the user is a new user he/she will be redirected to the Character Select Page. On selecting character the user will be taken into the game world.
 - If the user is a returning user the Game will store the User selected character type and it's last known position and will directly load the information in the game world.
 - [MOVEMENT] Use following keys to move:
 - 'w' moves character forward
 - 's' moves character backwards
 - 'a' turns character right
 - 'd' turns character left
 - 'left-shift' speeds-up the character
 - [CHAT]
 - '0' (Zero-button) toggles chat window to show or hide and broadcast message to all players
 - '1' (One-button) toggles chat window to show or hide and broadcast message to individual player by preceding the message with the username followed by "/"
for example: panda/do you like bamboo?
 - [Exit] Use 'escape' key to quit the game. Pressing close('x') button on the right-top corner of the window will not save the game state.

- [NOTIFICATION]
 - Whenever a new player signs-in or signs-out, player-list is updated automatically. Pressing 'I' key toggles player-list on and off on the screen.
- [PLAYER'S INFORMATION]
 - On first login, player selects the character and is assigned a random location in the world.
 - On next logins, selected character is automatically positioned where the player last left the game.
- [SPHERE ROTATION]
 - As soon as any player's character gets close to static sphere, they rotate. This rotation can be seen by all signed-in players.

Database Schema

Users

 PK username (VARCHAR NOT NULL)

password (VARCHAR NOT NULL)

type (INT)

x (FLOAT)

y (FLOAT)

h (FLOAT)

Controls

ESC	Quit
W	Move Forward
S	Move Backward
A	Move Left
D	Move Right
Left-Shift	Speed-up
Left_arrow	Rotate Camera Left
Right_arrow	Rotate Camera Right
0	Toggle Chat Broadcast
1	Toggle Private Chat
	Begin with username followed by a "/"
L	List All Connected Users

*Chat message window must be toggled to view messages

CONSTANTS

100-level codes are reserved for client messages.
200-level codes are reserved for server messages.
300-level codes are reserved for unpaired messages.

Request Constants

CMSG_AUTH	101
CMSG_DISCONNECT	102
CMSG_REGISTER	103
CMSG_CREATE_CHARACTER	104
CMSG_CHAT_ALL	105
CMSG_CHAT_ONE	106
CMSG_MOVE	107
REQ_HEARTBEAT	301

Response Constants

SMSG_AUTH	201
SMSG_DISCONNECT	202
SMSG_REGISTER	203
SMSG_CREATE_CHARACTER	204
SMSG_CHAT_ALL	205
SMSG_CHAT_ONE	206
SMSG_MOVE	207

Client Side Protocol (Requests)

Type and usage	Format
RequestLogin Client requests to login with a username and password. The server validates this and responds with ResponseAuth.	Short Constants.CMSG_AUTH String Username String Password
RequestDisconnect Client wishes to log out from the game. No more requests are to be sent after this. The server will will update other users with ResponseDisconnect.	Short Constants.CMSG_DISCONNECT
RequestRegistration Client registers a new account with the server which includes a username and password. The server validates this and responds with ResponseRegistration.	Short Constants.CMSG_REGISTER String Username String Password
RequestCreateCharacter Client wishes to create a new character tied to their account which includes the character type and position.	Short Constants.CMSG_CREATE_CHARACTER String Username Short Type Float x Float y Float z
RequestChatAll The client chats to all other clients through the chat function. The client uses a String for the message being sent.	Short Constants.CMSG_CHAT String Username String message
RequestChatOne The client chats to one particular client through the chat function. The client uses a String for the message being sent.	Short Constants.CMSG_CHAT String Username String message String ReceiverUsername
RequestMove Client issues a change to their location. It is followed by creating a number of ResponseMove and Server will update other users with these ResponseMove	Short Constants.CMSG_MOVE Float x //location vector Float y Float z Float h //facing direction

<p>RequestKey</p> <p>Sends to the server the key pressed by the player.</p> <p>It functions this way: if the user press one move key, the value will be send by this request. When the user release the key, another request will be send to notify the stop</p> <p>Client issues a change to their location. It is followed by creating a number of ResponseMove and Server will update other users with these ResponseMove</p>	<p>Short Constants.CMSG_KEY</p> <p>String moveKey</p> <p>Float moveKeyValue</p>
<p>RequestHeartbeat</p> <p>Request sent every .02 seconds, to keep the connection between client and server alive, and et updates fom the server. The server will be able to check the client's queued response and send all of them out to the client socket.</p>	<p>Short Constants.REQ_HEARTBEAT</p>

Client Side Protocol (Requests)

Type and usage	Format
ResponseAuth Response to RequestLogin, could it be positive or negative authentication. Sets the character of the user to its last position and type	Short Constants.SMSG_AUTH String Username Short LoginResponse Short Type Float x Float y Float z
ResponseDisconnect Response that informs the other players that this Client had disconnected..	Short Constants.SMSG_DISCONNECT String Username (of the disconnected player)
ResponseRegistration Response to RequestRegistration.	Short Constants.SMSG_REGISTER String Username Short RegistrationResponse
ResponseCreateCharacter Response that informs other player in the world that a player has entered the game, and a character has been created. It also sends the data to create the character	Short Constants.SMSG_CHARACTER_CREATE String Uername Short Type Float x Float y Float z
ResponseChatAll Represent the reception of a message send by a player to all the players. It contains a message to show in the chat.	Short Constants.SMSG_CHAT_ALL String username String message
ResponseChatOne Represent the reception of a message send by a player to that particular player. It contains a message to show in the chat.	Short Constants.SMSG_CHAT_ONE String SenderUsername String ReceiverUsername String message
ResponseMove Response used to notify all the other player to a move someone. It contains the	Short Constants.SMSG_MOVE String username Float x //location vector

new position for that character	Float y Float z Float h //facing direction
ResponseKey This response tells the key pressed by one player. It is used to update the moving of his character.	Short Constants.SMSG_KEY String username String moveKey Float moveKeyVal

Collision detection:

Detection logic:

For every character, bounding radius is calculated on initialization. Whenever the player's character move, the distance between player's character and every other character is computed. This distance is then compared with the sum of player's bounding radius and other character's bounding radius. If distance is less than the sum of both radii then application detects a collision.

Collision resolution:

If collision is against a static object, character does not move forward. If collision is against a non-static object then character turns 180 degrees and starts moving in that direction.