ネットワークプログラミングを用いたMMORPG制作

発展プログラミングレポート課題

5422045 根本勢也

概要

今回のプログラム制作ではネットワークプログラミングを使用し、TCP通信でサーバーとクライアント間のデータの送受信を行なった。サーバーのIPアドレスを”127.0.0.1”とし、ポート番号”5204”を用いて複数のプレイヤーがログインすることを想定したMMORPGゲームの制作について、本レポートで説明することとする。

1 制作内容

今回、複数のプレイヤーが同時にログインできるMMORPG([1] 剣と魔法のログレスを参考) を想定したゲーム制作プログラムを行なった。自身のプレイヤー目線の2Dゲームで、マップ内を駆け回ることができる。

1.1 サーバーとクライアント

本ネットワークプログラムでは、サーバーとクライアントがJSON形式でデータの送受信を行い、状態を同期させている。主に、プレイヤーの操作、NPCのインタラクション、エネミー戦闘の処理を行う。サーバーから最新のゲーム状態をクライアント側から受信し、クライアントでの操作情報をサーバーに送信して相互作用している。以下に送受信データの表を示す。

データ項目	送信元	受信先	データ形式	説明
プレイヤーの位置	クライアント	サーバー	JSON	プレイヤーの座標（cx, cy）を送信
ステージ情報	サーバー	クライアント	JSON	現在のステージ情報（stage）を送信
キャラクタータイプ	クライアント	サーバー	JSON	プレイヤーのキャラクタータイプ（characterType）
NPC情報	サーバー	クライアント	JSON	NPCの位置（x, y）とメッセージ（message）
エネミー情報	サーバー	クライアント	JSON	エネミーの位置（x, y）、HP（hp）
移動指令（方向）	クライアント	サーバー	JSON	プレイヤーの移動指令（direction）を送信
近接攻撃指令	クライアント	サーバー	JSON	プレイヤーの近接攻撃アクション（attack）を送信

1.2　ゲーム動作例

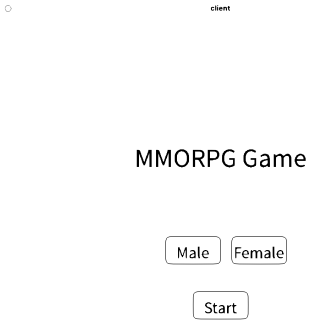

図 1　タイトル画面

タイトル画面では、、タイトル表示と性別選択、スタートボタンが実装されており、性別選択ボタンは選択すると重く文字が変わるようになっている。性別を選択してもらう理由として、操作するキャラクターの見た目が変わるためである。

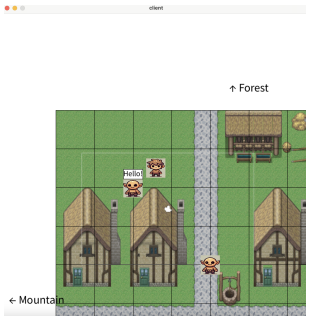

図 2　プレイ画面 1

画面の中心には自身の操作できる k っやらクターが表示されており、図 2 では一番最初のステージにいる状態である。マウスでクリックした場所、もしくわ WASD で移動ができ、最初のステージでは NPC のキャラクターに近づくことでセリフが頭の上に表示されるようになっている。また、ステージの左もしくは上に移動することで他の 2 つのステージ (山と森) に移動することができる。

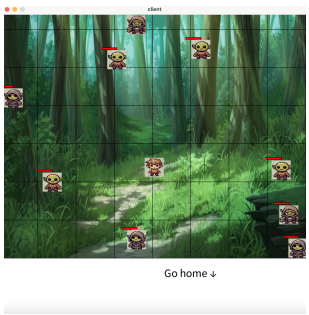

図 3　プレイ画面 2

図 3 は最初のステージから上に移動したステージである。ここでは、敵キャラクターが 10 体表示されており、頭の上に HP バーがある。プレイヤーは敵キャラクターと一定の近さになるとスペースキーで攻撃することができる。敵キャラクターの HP はそれぞれ違うため何度が攻撃をすることで倒すことができるようになっている。

2　プログラムの動作の説明

2.1　サーバープログラム

2.1.1　クラス・関数一覧

クラス名・関数名	説明
Player	プレイヤーの状態や位置を管理するクラス
NPC	NPC の情報（位置、メッセージ）を管理するクラス
Enemy	敵キャラクターの情報（位置、HP、種類）を管理するクラス
Server	サーバーの機能（接続、クライアント管理）を提供するクラス
setup()	サーバーの初期化、NPC と敵の配置を行う。
draw()	ゲームの状態を描画し、クライアントに送信する。
clientEvent(Client client)	クライアントからのイベントを受け取り、プレイヤーのアクションに対応。
attackEnemy(Player player)	プレイヤーの近接攻撃処理、敵 HP の減少を行う。
startGameForPlayer(Player player)	プレイヤーの位置とステージの初期化を行う。
serverEvent(Server server, Client client)	新しいプレイヤーが接続したときに呼び出され、プレイヤーを追加。
disconnectEvent(Client client)	クライアントが切断したときに呼び出され、プレイヤーを削除。

2.1.2　Player クラス

このクラスは、ゲーム内のプレイヤーを管理するクラス。位置、目標位置、移動速度、ステージ、タイプ、背景画像の情報をもつ。

変数	型	説明
`id`	int	プレイヤーの id 子。
`cx`	float	プレイヤーの現在の X 座標。
`cy`	float	プレイヤーの現在の Y 座標。
`targetX`	float	プレイヤーの目標 X 座標。
`targetY`	float	プレイヤーの目標 Y 座標。
`speed`	float	プレイヤーの移動速度（デフォルトは 2）。
`currentStage`	String	プレイヤーが現在いるステージ（デフォルトは”default”）。

変数	型	説明
stageBackgrounds	HashMap¡String, String¿	ステージごとの背景画像を格納するマップ。
characterType	String	プレイヤーのキャラクタータイプ（デフォルトは”male”）。

メソッド名	引数	説明
move(float dx, float dy)	dx, dy (float)	プレイヤーを指定した方向（dx, dy）に移動させ、ステージ遷移を確認する。
moveToTarget()	なし	プレイヤーが目標位置（targetX, targetY）に向かって移動し、目標に到達したらその座標を更新する。
checkStageTransition()	なし	プレイヤーが画面の端に到達した場合、ステージを遷移させる。
toJSON()	なし	プレイヤーの情報を JSON 形式で返す。

2.1.3 NPC クラス

このクラスは、最初のステージにいる NPC を管理するクラス。位置とメッセージの情報をもつ。

変数	型	説明
id	int	NPC の一意な id。
x	float	NPC の X 座標。
y	float	NPC の Y 座標。

メソッド名	引数	説明
toJSON()	なし	NPC の情報を JSON 形式で返す。

2.1.4 Enemy クラス

このクラスは、ゲーム内の敵キャラクターを管理するクラス。位置と、タイプと HP の情報をもつ。

変数	型	説明
id	int	敵の id。
x	float	敵の X 座標。
y	float	敵の Y 座標。
type	String	敵のタイプ（画像ファイル名）。
hp	int	現在の敵の HP。
maxHp	int	敵の最大 HP。

メソッド名	引数	説明
toJSON()	なし	敵の情報を JSON 形式で返す。

2.1.5 Server の管理

ゲームの状態とクライアントとの通信を管理する。

変数	型	説明
server	Server	ゲームのサーバーインスタンス。
idOffset	int	プレイヤーに割り当てる ID のオフセット。
players	HashMap¡Client, Player¿	クライアントと対応するプレイヤーを管理するマップ。
npcs	ArrayList¡NPC¿	ゲーム内の NPC リスト。
stageEnemies	HashMap¡String, ArrayList¡Enemy¿¿	ステージごとの敵リストを管理するマップ。
enemyTypes	String[]	敵の画像ファイル名（タイプ）を格納する配列。

setup	対象	説明
server = new Server(this, 5204);	Server	サーバーインスタンスをポート 5204 で開始。
npcs.add(new NPC(1, 200, 200, "Hello!"));	NPC	4 人の NPC をゲームに追加し、それぞれの位置とメッセージを設定。
stageEnemies.put("mountain", new ArrayList<Enemy>());	stageEnemies	ステージ「mountain」と「forest」に空の敵リストを設定。
for (int i = 0; i < 10; i++) ...	敵	「mountain」と「forest」それぞれに 10 体の敵をランダムに配置。

draw	対象	説明
for (Client client : players.keySet()) ...	プレイヤー	すべてのプレイヤーに対して、現在の位置を更新し、必要な情報を送信。
player.moveToTarget();	Player	プレイヤーの位置を目標地点に向かって移動させる。
JSONArray playerArray = new JSONArray();	プレイヤー	プレイヤーが表示される範囲内のプレイヤーを JSON 形式で準備。
JSONArray enemyArray = new JSONArray();	敵	プレイヤーが現在いるステージに対応する敵を JSON 形式で準備。

draw	対象	説明
`JSONArray npcArray = new JSONArray();`	NPC	プレイヤーが「default」ステージにいる場合、NPC 情報を JSON 形式で準備。
`JSONObject message = new JSONObject();`	メッセージ	プレイヤー、NPC、敵の情報を含む JSON オブジェクトを作成。
`client.write(message.toString());`	クライアント	作成した JSON メッセージをクライアントに送信。

2.1.6 attackEnemy 関数

この関数ではプレイヤーが敵キャラクターを攻撃した時の処理を表す。HP が 0 になった時リストから削除するようになっている。さらに、synchronized で複数のクライアントが同時に動作することを防いでいる。

処理	対象	説明
`synchronized void attackEnemy(Player player)`	メソッド	プレイヤーが敵を攻撃する処理。 synchronized キーワードを使って排他制御を行い、複数スレッドからの同時アクセスを防ぐ。
`if (stageEnemies.containsKey(player.currentStage))`	ステージ	プレイヤーの現在のステージに敵がいるかを確認。
`ArrayList<Enemy> enemies = stageEnemies.get(player.currentStage);`	敵	現在のステージに対応する敵のリストを取得。
`for (int i = enemies.size() - 1; i >= 0; i--)`	敵	敵リストの最後から順に敵をチェックする。 リストのサイズが変わる可能性があるため逆順で処理。
`Enemy enemy = enemies.get(i);`	敵	現在の敵を取得。
`if (dist(player.cx, player.cy, enemy.x, enemy.y) < 50)`	距離	プレイヤーと敵の距離が 50 ピクセル未満の場合、 攻撃が当たるとみなす。
`enemy.hp -= 5;`	敵	敵の HP を 5 減らす。
`if (enemy.hp <= 0)`	敵	敵の HP が 0 以下の場合、 敵が倒されたとみなし、リストから削除。
`enemies.remove(i);`	敵	HP が 0 以下になった敵を敵リストから削除。

2.1.7 クライアントとの通信

クライアントからのデータの送信を受け取る部分である。ゲームの状態と移動情報などを受け取る。

処理内容	説明
1. clientEvent(Client client) メソッド	
クライアントからのメッセージの受信	クライアントから送信されたメッセージを受け取って解析する。
ゲーム開始処理	‘start‘ アクションが含まれていれば、‘startGameForPlayer()‘ メソッドを呼び出し、プレイヤーの初期化を行う。
プレイヤーの移動処理	メッセージ内に ‘direction‘が含まれていれば、指定された方向にプレイヤーを移動させる。
ターゲット座標の更新	‘targetX‘と ‘targetY‘を使って、プレイヤーのターゲット位置を更新。
キャラクタータイプの更新	‘characterType‘を更新し、プレイヤーのキャラクタータイプを変更。
攻撃処理	‘attack‘が ‘melee‘であれば、‘attackEnemy()‘ メソッドを呼び出し、近接攻撃を行う。
2. startGameForPlayer(Player player) メソッド	
プレイヤーの位置初期化	プレイヤーをランダムな位置に配置し、‘cx‘, ‘cy‘, ‘targetX‘, ‘targetY‘を設定。
ステージ初期化	プレイヤーの ‘currentStage‘を ‘”default”‘に設定。
3. serverEvent(Server server, Client client) メソッド	
新規プレイヤーの追加	新しいクライアントが接続した際に呼び出され、プレイヤーを ‘players‘マップに追加。
4. disconnectEvent(Client client) メソッド	
プレイヤーの削除	クライアントが切断した際に、そのプレイヤーを ‘players‘マップから削除。

2.2 クライアントプログラム

2.2.1 クラス・関数一覧

クラス名・関数名	説明
setup()	ゲームの初期設定を行う関数。サーバーの接続、キャラクター画像の読み込み、ステージ背景の設定などを行う。
draw()	ゲームの状態を描画する関数。タイトル画面やゲームワールドを描画し、サーバーとデータを同期。
drawTitleScreen()	タイトル画面を描画する関数。ゲームのタイトルやスタートボタン、性別選択ボタンを表示。
drawGameWorld()	ゲームワールドを描画する関数。プレイヤー、NPC、敵キャラクター、ステージなどを描画。
drawNPCs()	NPC を描画する関数。プレイヤーとの距離が近いと、メッセージを表示する。
drawEnemies()	敵キャラクターを描画する関数。HP バーとともに敵の種類に応じた画像を表示。
drawField()	ゲームステージのグリッドを描画する関数。ステージによって異なる指示を表示する。
drawPlayers()	プレイヤーキャラクターを描画する関数。キャラクターの種類に応じた画像（男性・女性）を表示。
keyPressed()	キー入力を処理する関数。移動や攻撃の指示をサーバーに送信。
mousePressed()	マウスクリックを処理する関数。タイトル画面でボタンがクリックされた場合や、ゲーム内で移動先を指定。
sendMessageToServer()	サーバーにメッセージを送信する関数。クライアントとサーバー間でデータを送信する。
sendCharacterSelection()	プレイヤーのキャラクター選択（男性・女性）をサーバーに送信する関数。
clientEvent()	クライアントからのデータ受信を処理する関数。サーバーから送られてきたゲーム状態を更新。
Button	ボタンの表示とクリック判定を行うクラス。ラベル、位置、サイズ、色などを設定。

2.2.2 Buttun クラス

このクラスは、ゲームのタイトル画面で表示されるボタンの管理を行っている。描画と、マウスがボタンの上にあるかの判断をする関数を持つ。

クラス名・関数名	説明
display()	ボタンを画面に描画するメソッド。背景色（白)、枠線、ラベルのテキストを描画する。
isMouseOver()	マウスがボタン上にあるかを判定するメソッド。ボタンの範囲内にマウスカーソルがある場合は ‘true‘ を返す。
setTextColor(color newColor)	ボタン上のテキストの色を変更するメソッド。新しい色を引数として受け取る。

2.2.3 setup() と draw()

ゲームの初期化と描画処理を行っている。

動作	説明
client = new Client(this, ”127.0.0.1”, 5204)	サーバーにポート 5204 で接続するクライアントを作成。
maleImage, femaleImage, npcImage	プレイヤーの性別（男性、女性）や NPC の画像をそれぞれ読み込む。
stageBackgrounds.put()	ゲーム内の各ステージ（default, mountain, forest）の背景画像をハッシュマップに格納する。
enemyImages.put()	各敵キャラクターの画像（gob1.png, gob2.png, gob3.png）をハッシュマップに格納する。
startButton, maleButton, femaleButton	「Start」「Male」「Female」のボタンをそれぞれ作成。位置とサイズを指定。
if (onTitleScreen)	タイトル画面の描画処理。
if(status .getJSONArray(”players”) == null) return;	プレイヤー情報が取得できない場合は処理を終了する。
synchronized (status)	ステータスの同期を行い、ゲームの状態を描画する。
stageBackgrounds .containsKey()	ステージに応じた背景画像を描画。
PImage bg = stageBackgrounds.get()	現在のステージに対応する背景画像を取得。
pushMatrix()	座標系の変換を適用する。
translate(-cx + width / 2, -cy + height / 2)	プレイヤーの位置に基づいて移動。
image(bg, 500, 500, gameWidth, gameHeight)	背景画像を指定の位置とサイズで描画。
popMatrix()	座標系の変換を元に戻す。

動作	説明
drawGameWorld()	ゲームの世界（フィールドやキャラクター）の描画を行う関数。

2.2.4 drawTitleScreen 関数

タイトル画面の描画を行う。

動作	説明
startButton.display()	「Start」ボタンを表示。
maleButton.display()	「Male」ボタンを表示。
femaleButton.display()	「Female」ボタンを表示。

2.2.5 drawGameWorld 関数

背景画像、プレイヤー、NPC、敵キャラクターの描画をする関数。

動作	説明
pushMatrix()	描画の行列を保存し、変換を適用する準備。
translate(-cx + width / 2, -cy + height / 2)	画面の中心にプレイヤーを合わせるために、描画の位置を調整。
drawField()	ゲームのフィールドを描画。グリッド線を表示。
drawPlayers()	プレイヤーキャラクターを描画。
if (status.hasKey(”stage”) status.getString(”stage”) .equals(”default”))	ステージが「default」の場合、NPC を描画。
drawNPCs()	NPC（ノンプレイヤーキャラクター）を描画。
if (status.hasKey(”stage”) status.getString(”stage”) .equals(”mountain”))	ステージが「mountain」の場合、敵を描画。
drawEnemies()	敵キャラクターを描画。
if (status.hasKey(”stage”) status.getString(”stage”) .equals(”forest”))	ステージが「forest」の場合、敵を描画。
popMatrix()	描画の行列を元に戻す。

2.2.6 drawNPCs 関数

最初のステージにのみ、NPC を表示する関数。メッセージの処理も行っている。

動作	説明
if(!status.hasKey(”stage”) !status.getString(”stage”) .equals(”default”))	ステージが「default」でない場合、NPC を描画しない。
JSONArray npcs = status.getJSONArray(”npcs”)	ゲーム状態から NPC の配列を取得。
JSONObject npc = npcs.getJSONObject(i)	各 NPC の情報（位置、メッセージなど）を取得。
if (dist(playerX, playerY, npcX, npcY) ¡ 100)	プレイヤーと NPC の距離が 100 以下の場合、NPC のメッセージを表示。

2.2.7 drawEnemies 関数

特定のステージにのみ敵キャラクターを描画する関数。HP バーの処理も行っている。

動作	説明
if(!status.hasKey(”enemies”)) return;	敵キャラクターが存在しない場合、何もしない。
JSONArrayenemies =status.getJSONArray (”enemies”)	ゲーム状態から敵キャラクターの配列を取得。
JSONObjectenemy =enemies.getJSONObject(i)	各敵キャラクターの情報（位置、種類、HP など）を取得。
if (enemyImage != null)	敵キャラクターの画像が存在する場合、その画像を描画。
else	敵キャラクターの画像が存在しない場合、代わりに赤い円を描画。

2.2.8 drawField 関数

ゲームのステージを描画する関数。ステージは格子状の線で構成され、ゲーム内の各エリアを示すテキストも表示している。

動作	説明
if (status.hasKey(”stage”) status.getString(”stage”) .equals(”default”))	ステージが「default」の場合、エリア情報を描画。
elseif(status.hasKey(”stage”) status.getString(”stage”) .equals(”mountain”))	ステージが「mountain」の場合、帰還エリア情報を描画。
elseif(status.hasKey(”stage”) status.getString(”stage”) .equals(”forest”))	ステージが「forest」の場合、帰還エリア情報を描画。

2.2.9 drawPlayers 関数

プレイヤーを描画するための関数。

動作	説明
JSONArray players = status.getJSONArray(”players”)	プレイヤー情報を取得するため、‘players‘ 配列を取得。
for (int i = 0; i ¡ players.size(); i++)	プレイヤーの数だけループ処理を実行。
JSONObject player = players.getJSONObject(i)	プレイヤーごとの情報を ‘JSONObject‘ として取得。
if(characterType.equals(”male”))	キャラクターが男性の場合の処理。

2.2.10 マウスとキーボード操作

プレイヤーの操作を管理する関数。

関数	説明
keyPressed()	w のクリックで上へ、a のクリックで左へ、s のクリックで下に、d のクリックで右に移動する。space キーで攻撃

関数	説明
mousePressed()	タイトル画面では、male を押すと文字の色を変えキャラクター (画像) を male.png に female で female.png にする。start ボタンでゲーム画面に移動。ゲーム画面では、クリックした場所にキャラクターを移動させる。

2.2.11 サーバーとの通信

クライアント側での操作をサーバーに送信する処理。

関数	説明
sendMessageToServer()	メッセージを文字列に変換し、サーバーに送信する。
sendCharacterSelection()	キャラクターのタイプをサーバーに送信する。
clientEvent()	サーバーからデータを受信する。synchronized で他のクライアントと同時にアクセスするのを防いでいる。

3 工夫した点

- 複数の画像ファイルを用いて、キャラクターの見た目の変更や、ステージの背景画像、ステージごとにいる NPC や敵キャラクターの工夫を行なった。
- 移動性に特化させるために、マウスクリックで移動できる機能を実装
- Synchronized ブロックを用いて、複数のクライアントで同時に同じデータ更新を起きないようにした。攻撃を行って同時に HP を減らすときや、違いの行動パターンを重複しないようにした。
- MMO 感を出すために、NPC と敵キャラの実装をし、メッセージがでたり、敵を攻撃できるようにプレイ性を増やした。
- ステージの切り替えなどを行って複数のプレイヤーが接続してもリアルタムでゲームを楽しめるよう工夫を行なった。

4 参考資料・文献

1. Marvelous Inc. Aiming Inc, 剣と魔法のログレス [ゲームアプリ]
   公式サイト:https://aismiley.co.jp/ai_news/what-is-multimodal-ai-model/
2. 参考資料 PDF: [ProcessingCheetSheet]
3. 発展プログラミング授業資料:[GitBook]
4. ORACLE. *Java HashMap Documentation*.
   取得元: https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html