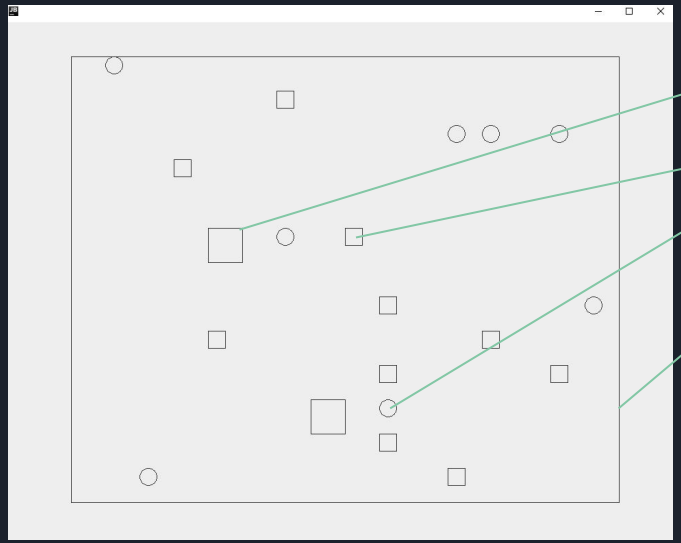
Java Socket Rogue Game

- Co-Op Multiplayer game concept
- Connection utilizing TCP with UTF8
- Functions and Classes
- Server State Diagram
- Client State Diagram

Anton Danker

Co-Op Multiplayer game concept



- Player
- “Pickup-able” items
- Blocking Border

This is a cooperative game in which the goal is for the two players to pick up their own respective items. When both have collected 10 items each the game is won. A player can only collects its own, e.g circle, items and not the other player’s, e.g squared items.

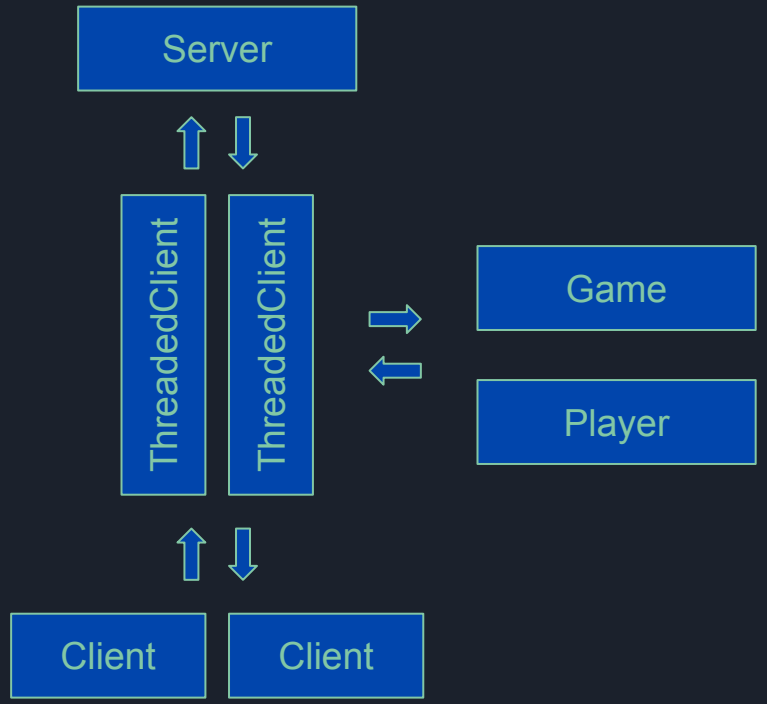
Anton Danker

Connection utilizing TCP with UTF8

- Server starts a ServerSocket on port: 4999
  It will then wait for a Client using TCP

- A ThreadedClient is created for each respective Client that is started. The ThreadedClient reads the UTF8 command coming from its Client.
- The appropriate response is then sent to the Server which forwards the command to all ThreadedClients.

- The ThreadedClients then make sure to update the Game and Player class according to changes being made.



Anton Danker

Functions and Classes

The Client will read the user's input and write it (writeUTF) into a UTF8 action to its outputStream

```
switch (e.getKeyChar()) {
    case 'w':
        try {outputStream.writeUTF( str: "move_up"); }
        catch (IOException ioe) { ioe.printStackTrace(); }
        break;
    case 's':
        try {outputStream.writeUTF( str: "move_down"); }
        catch (IOException ioe) { ioe.printStackTrace(); }
        break;
    case 'a':
        try {outputStream.writeUTF( str: "move_left"); }
        catch (IOException ioe) { ioe.printStackTrace(); }
        break;
    case 'd':
        try {outputStream.writeUTF( str: "move_right"); }
        catch (IOException ioe) { ioe.printStackTrace(); }
        break;
    }
```

In the ThreadedClient the action will be read (readUTF). For this example it's a move action so we check that the player is not moving out of bounds or into another player. checkItem checks if the Player stepped on an item extraFunc calls the Server function to send this information to all Clients

```
case "move_right":
    player = game.getPlayer(playerID);
    if (!(player.getxPos() + 50 >= 900) && moveBlock(player, move: "move_right")) {
        checkItem(player);
        player = game.getPlayer(playerID);
        player.move_right();
        extraFunc( cmdWord: "move_right,", player);
    }
break;
```

```
public void pickupItem(int inc
    game.removeItem(index);
    player.gainPoint();
    System.out.println(player.
    String command = "pickup_i
    Server.sendAll(command);
}
```

The pickupItem function will (among other things) modify the current player with gainPoint() .

This will give the player a point to its inventory

Anton Danker

Server messaging to all

The sendAll function performs the ThreadedClients send function to every user

Taken from Server

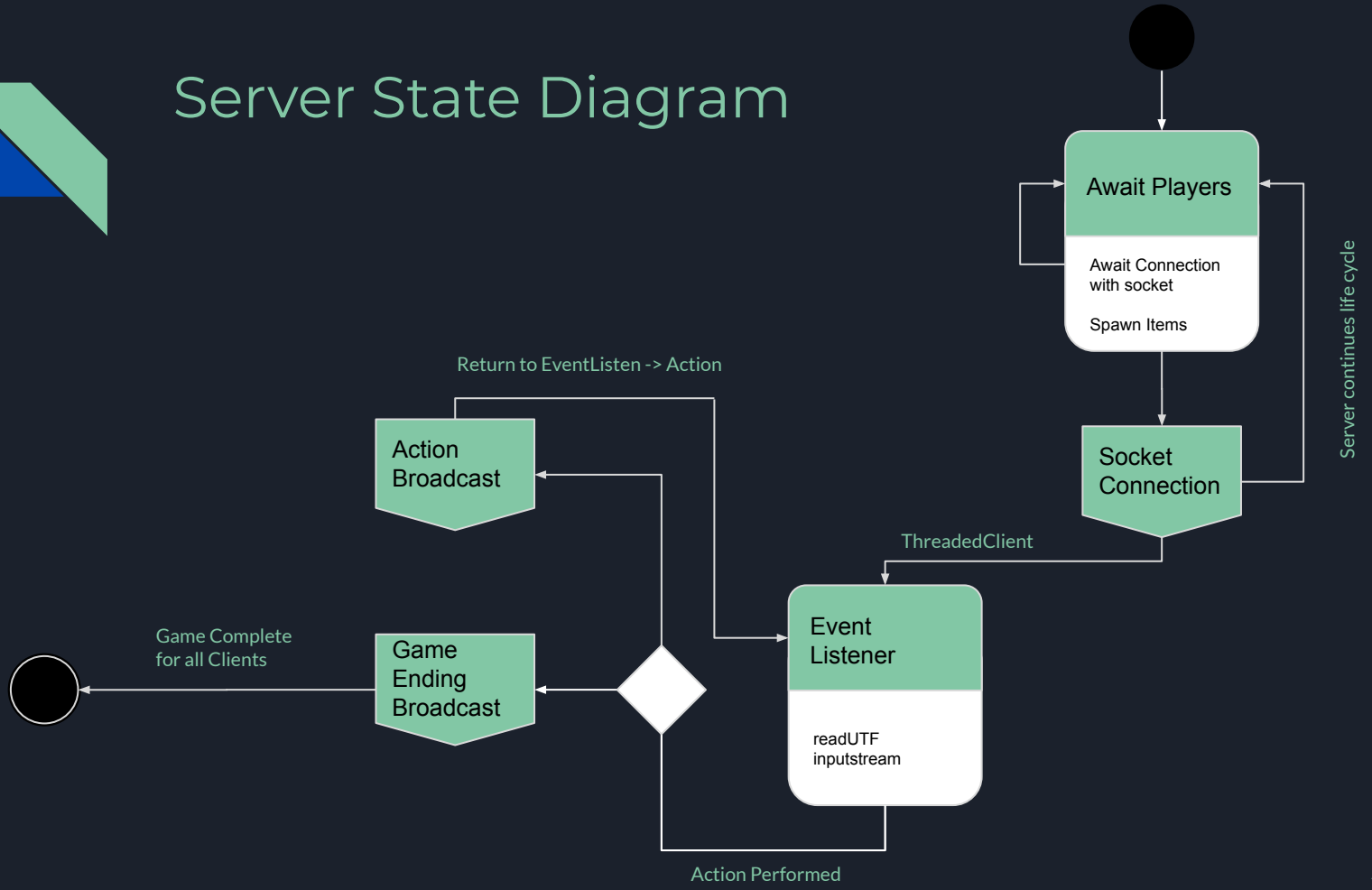
```
public static void sendAll(String event) throws IOException {
    if (allPlayers != null && !allPlayers.isEmpty()) {
        for (Map.Entry<String, ThreadedClient> t : allPlayers.entrySet()) {
            t.getValue().send(event);
        }
    } else {
        System.out.println("FAILED: There are no players to stream information to...");
    }
}
```

Taken from ThreadedClient

```
public void send(String event) throws IOException {
    outputStream.writeUTF(event);
}
```

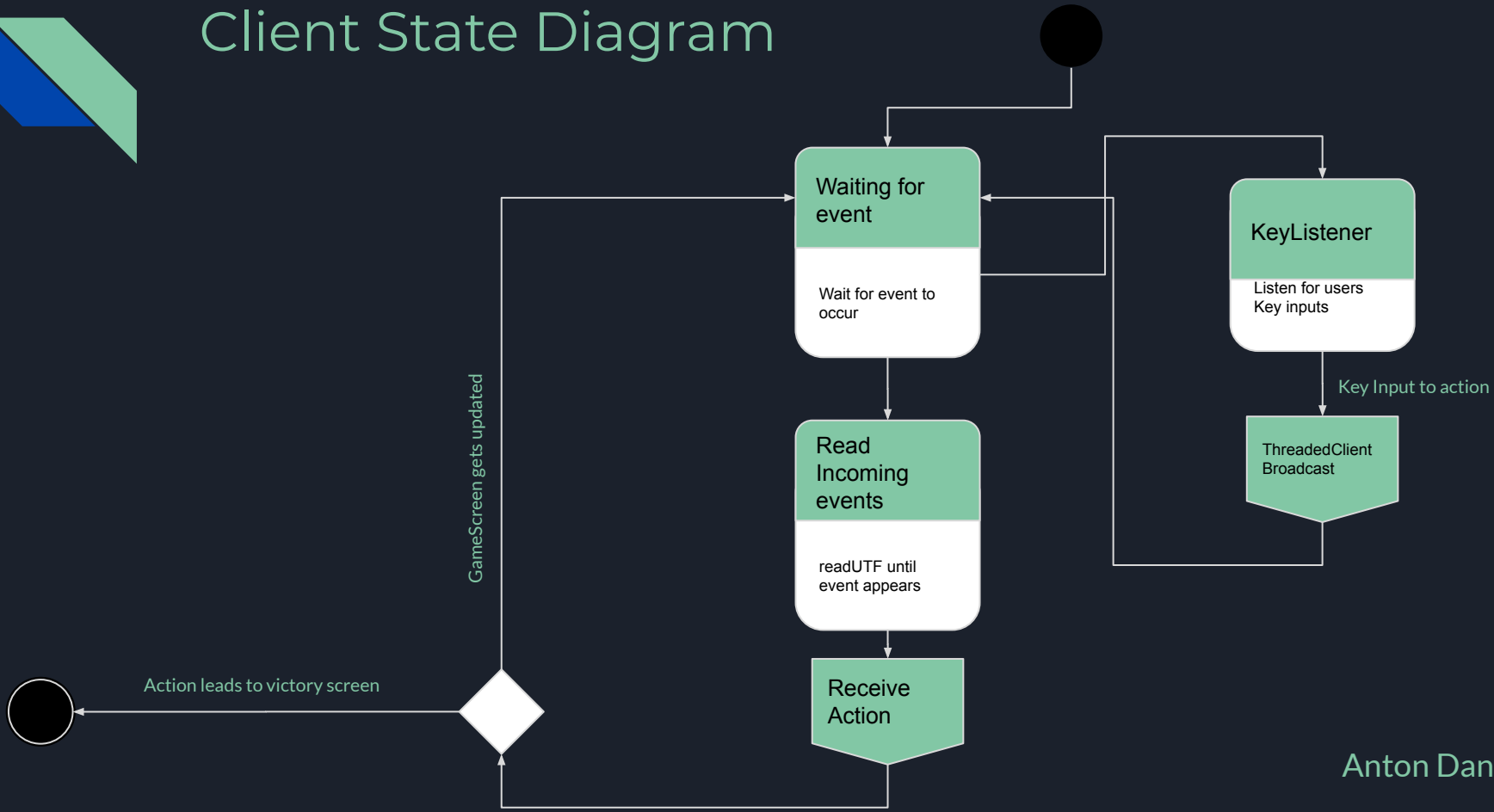
Anton Danker

Server State Diagram



Anton Danker

Client State Diagram



Anton Danker