BtlShyp API Documentation

Nicholas Goldberg
Steven Julien
Jonathan Mirabile
Christopher Nash

API (Nov 21 @ 9:30PM)

Create a document detailing what you project will do from a user perspective, and a separate page detailing the interfaces for the project. These include how the network messages to and from the server will be handled, and how much of the API for these messages will be coded by the group vs. how much will be expected from the individual developer.

The more detail you have in the interface spec, the easier and faster it will be to get your server code and network interface written.

User Perspective

BtylShyp is a networked version of the Milton Bradley board game “BattleShip”. Multiple players can join a network which is hosted by a server. After being connected to the main game room, all players can communicate via a global chat room. All communication is visible to all players. Players can then request to join a game of “BtlShyp”. Once the server affirmatively acknowledges the player’s “Join Game” request, that player is entered into a game. If another player is currently waiting for an opponent, the player will join the awaiting opponent. Otherwise the player will wait for an opponent. Once two players have been joined together, that game is full, however the server can host multiple games at the same time, so the aforementioned process can be repeated.
Game begins with the user being prompted to place their ships on a 5x5 grid board. Each player has four ships: a patrol boat, a submarine, a destroyer, and a battleship. Once both players have all ships placed on their grids with no boats overlapping or extending beyond the bounds of the board, the gameplay can start. The server decides who plays first and prompts for the user’s attack. The player chooses a coordinate on the grid as their point of attack and fires on that position. If the attack coordinates match that of one of the opponent’s boats, then a “Hit” is registered, else a “Miss”. Both players are made aware of the the status of the shot. Players alternate turns. If a “Hit” is the last available coordinate of a boat, the boat is sunk. All players in the BtlShyp game room are notified of the attacker’s username and the sinking of the ship. Play continues until all of one player’s boats are sunk. The remaining player is declared the winner and the game room is notified. If a player leaves the game early, the remaining player is declared the winner. Upon completion of the game, the players are ejected from the game and once again have access to join a new game.

Application Programming Interface

Please see our documentation here.
https://github.com/WeberCS4230/BtlShyp/blob/master/docs/design/README.md

Examples of stringified Messages

```
AttackAttemptMessage --
{"xCoord":2,"yCoord":2,"type":"ATTACK_ATTEMPT","username":"Testy McTester"}
ChatMessage -- {"text":"Test chat message","type":"CHAT","username":"Testy
McTester"}
GameStartMessage -- {"opponentUsername":"Testy
McTester","type":"GAME_START","username":"Testy McTester"}
GameWonAttemptMessage -- {"type":"GAME_WON_ATTEMPT","username":"Testy McTester"}
GameWonResponseMessage --
{"gameResult":"WIN","type":"GAME_WON_RESPONSE","username":"Testy McTester"}
JoinAttemptMessage -- {"type":"JOIN_ATTEMPT","username":"Testy McTester"}
JoinResponseMessage --
{"confirmJoin":"ACCEPT","type":"JOIN_RESPONSE","username":"Server"}
ShipsPlacedMessage -- {"type":"SHIPS_PLACED","username":"Testy McTester"}
AttackResponseMessage --
{"hitOrMiss":"HIT","shipSunk":"BATTLESHIP","type":"ATTACK_RESPONSE","username":"T
esty McTester"}
TurnStartMessage -- {"turn":"START","type":"TURN","username":"Testy McTester"}
```

Message Types

```java
package main.btlshyp.message;

public enum MessageType {
  CHAT,
  JOIN_ATTEMPT,
  JOIN_RESPONSE,
  GAME_START,
  SHIPS_PLACED,
  ATTACK_ATTEMPT,
  ATTACK_RESPONSE,
  TURN,
  GAME_WON_ATTEMPT,
  GAME_WON_RESPONSE,
}

```

```java
package main.btlshyp.message;

import lombok.ToString;

/**
 * Base class for the Message hierarchy. Messages are sent and received to an from a connected server
 * things such as Chat Messages, Game Play Messages, etc.
 */
@ToString
public abstract class Message  {

 public MessageType type;

 public abstract MessageType getType();

 @Getter @Setter
 private String username;


 protected Message(String username) {
    this.username = username;
 }
}
```

```java
package main.btlshyp.message;

import lombok.Getter;

public class AttackAttemptMessage extends Message  {

    public AttackAttemptMessage(String username, int x, int y) {
        super(username);
        this.type = getType();
        this.xCoord = x;
        this.yCoord = y;
    }

    @Getter @Setter
    private int xCoord;

    @Getter @Setter
    private int yCoord;

    @Override
    public MessageType getType() {
        return MessageType.ATTACK_ATTEMPT;
    }


}
```

```java
package main.btlshyp.message;

import lombok.Getter;

public class AttackResponseMessage extends Message  {

   public static enum HitOrMiss {HIT, MISS};
   public static enum ShipSunk {NONE, PATROL_BOAT, SUBMARINE, DESTROYER, BATTLESHIP};
   @Getter @Setter
   private HitOrMiss hitOrMiss;
   @Getter @Setter
   private ShipSunk shipSunk;

   public AttackResponseMessage(String username, HitOrMiss hitOrMiss, ShipSunk shipSunk) {
      super(username);
      this.type = getType();
      this.hitOrMiss = hitOrMiss;
      this.shipSunk = shipSunk;
   }

   @Override
   public MessageType getType() {
      return MessageType.ATTACK_RESPONSE;
   }

}
```

```java
public class ChatMessage extends Message{

   @Getter
   private String text;

   public ChatMessage(String text) {
      super("Server");

      setText(text);
   }
   public ChatMessage(String text, String username) {
      super(username);
      this.type = getType();
      setText(text);
   }
   public void setText(String text) {
      if (text == null) {
         throw new IllegalArgumentException("Chat text cannot be empty");
      }
      this.text = text;
   }
   @Override
   public MessageType getType() {
      return MessageType.CHAT;
   }

}
```

```java
package main.btlshyp.message;

import lombok.Getter;

public class GameStartMessage extends Message  {

    public GameStartMessage(String username) {
        super(username);
        this.opponentUsername = username;
        this.type = getType();
    }

    @Getter
    private String opponentUsername;

    @Override
    public MessageType getType() {
        return MessageType.GAME_START;
    }


}
```

```java
package main.btlshyp.message;

public class GameWonAttemptMessage extends Message  {

    public GameWonAttemptMessage(String username) {
        super(username);
        this.type = getType();
    }

    @Override
    public MessageType getType() {
        return MessageType.GAME_WON_ATTEMPT;
    }

}
```

```java
package main.btlshyp.message;

import lombok.Getter;

public class GameWonResponseMessage extends Message  {

  public GameWonResponseMessage(String username, GameResult gameResult) {
    super(username);
    this.type = getType();
    this.gameResult = gameResult;
  }

  public static enum GameResult {WIN, LOSE};

  @Getter @Setter
  private GameResult gameResult;

  @Override
  public MessageType getType() {
    return MessageType.GAME_WON_RESPONSE;
  }

}
```

```java
package main.btlshyp.message;

public class JoinAttemptMessage extends Message{

  public JoinAttemptMessage(String username) {
    super(username);
    this.type = getType();
  }

  public MessageType getType() {
    return MessageType.JOIN_ATTEMPT;
  }
}
```

```java
package main.btlshyp.message;

import lombok.Getter;

public class JoinResponseMessage extends Message {

   public static enum ConfirmJoin {ACCEPT, REJECT};

   @Getter
   private ConfirmJoin confirmJoin;

   public JoinResponseMessage(ConfirmJoin confirmJoin) {
      super("Server");
      this.type = getType();
      this.confirmJoin = confirmJoin;
   }

   @Override
   public MessageType getType() {
      return MessageType.JOIN_RESPONSE;
   }

}
```

```java
package main.btlshyp.message;

public class ShipsPlacedMessage extends Message  {

   public ShipsPlacedMessage(String username) {
      super(username);
      this.type = getType();
   }

   @Override
   public MessageType getType() {
      return MessageType.SHIPS_PLACED;
   }

}
```

```java
package main.btlshyp.message;

import lombok.Getter;

public class TurnStartMessage extends Message  {

    public TurnStartMessage(String username, Turn turn) {
        super(username);
        this.type = getType();
        this.turn = turn;
    }

    public static enum Turn {START,END};
    @Getter @Setter
    private Turn turn;

    @Override
    public MessageType getType() {
        return MessageType.TURN;
    }

}

```