Login/Registration

Jeffrey Kiyomura

Login Summary

- Asks the user to input the username and password.
- client gets username and password and sends it for verification.
- server gets the information and checks whether it is value
- server sends results back to client
- client then does login based on the information sent from server.

Registration Summary

- It is very similar to Login.
- Differences:
  - Needs to check whether username is already created
  - makes sure password and verify passwords are the same.
- Once registration is complete it goes back to the login screen for login.

Login GUI


CONTESTED
Login
username:
password:
Cancel
Register
Submit

RequestLogin.py

```
class RequestLogin(ServerRequest):
    def send(self, message = None):
        try:
            print "username: ", list(message)[0], "; password: ", list(message)[1]
            pkg = PyDatagram()
            pkg.addUint16(Constants.CMSG_AUTH)
            pkg.addString(list(message)[0])
            pkg.addString(list(message)[1])

            self.cWriter.send(pkg, self.connection)

            #self.log('Sent [' + str(Constants.RAND_STRING) + '] Int Request')
        except:
            self.log('Bad [' + list(message)[0] + ", "+ list(message)[1] + '] Login Request')
            print_exc()
```

ResponseLogin.py

```
class ResponseLogin(ServerResponse):

    def execute(self, data):

        try:
            self.status = data.getString()
            print "ResponseLogin - ", self.status

            if(self.status == "Authorized"):
                print "Logged In!"
            elif self.status == "UnAuthorized":
                print "incorrect username/password"
                #self.l = login2()
                #self.l.throwIncorrectUsername()
                taskMgr.add(self.destroyIncorrectUsername, "destroyIncorrectUsername")
                self.throwIncorrectUsername()
            else:
                print "there was a problem with the server"
                taskMgr.add(self.destroyServerError, "destroyServerError")
                self.throwServerError()

            #self.log('Received [' + str(Constants.RAND_STRING) + '] String Response')

        except:
            self.log('Bad [' + str(self.status) + '] Login Response')
            print_exc()
    def throwIncorrectUsername(self):
        self.incorrectUsername = OnscreenText(text="Incorrect Username/Password",
                                    pos=(0, -.5), scale=0.05, fg=Constants.TEXT_ERROR_COLOR, mayChange=0)
    def throwServerError(self):
        self.serverError = OnscreenText(text="Unable to Connect to Server",
                                    pos=(0, -.5), scale=0.05, fg=Constants.TEXT_ERROR_COLOR, mayChange=0)
    def destroyIncorrectUsername(self, task):
        if task.time < 5.0:
            return task.cont
        else:
            self.incorrectUsername.destroy()
            return task.done
    def destroyServerError(self, task):
```

Registration GUI

RequestRegister.py

```
class RequestRegister(ServerRequest):
    def send(self, message = None):
        try:
            print "username: ", list(message)[0], "; password: ", list(message)[1]
            pkg = PyDatagram()
            pkg.addUint16(Constants.CMSG_REGISTER)
            pkg.addString(list(message)[0])
            pkg.addString(list(message)[1])

            self.cWriter.send(pkg, self.connection)

            #self.log('Sent [' + str(Constants.RAND_STRING) + '] Int Request')
        except:
            self.log('Bad [' + list(message)[0] + ", "+ list(message)[1] + '] Login Request')
            print_exc()
```

ResponseRegister.py

```
class ResponseRegister(ServerResponse):

    def execute(self, data):
        try:
            self.status = data.getString()
            print "ResponseRegister - ", self.status

            if(self.status == "Success"):

                print "Created Successfully!"
            elif self.status == "Same Username":
                print "Already have that username"
                #self.l = login2()
                #self.l.throwIncorrectUsername()
                taskMgr.add(self.destroyIncorrectUsername, "destroyIncorrectUsername")
                self.throwIncorrectUsername()
            else:
                print "there was a problem with the server"
                taskMgr.add(self.destroyServerError, "destroyServerError")
                self.throwServerError()

            #self.log('Received [' + str(Constants.RAND_STRING) + '] String Response')

        except:
            self.log('Bad [' + str(self.status) + '] Register Response')
            print_exc()
    def throwIncorrectUsername(self):
        self.incorrectUsername = OnscreenText(text="Already have that username",
                                pos=(0, -.5), scale=0.05, fg=Constants.TEXT_ERROR_COLOR, mayChange=0)
    def throwServerError(self):
        self.serverError = OnscreenText(text="Unable to Connect to Server",
                                pos=(0, -.5), scale=0.05, fg=Constants.TEXT_ERROR_COLOR, mayChange=0)
    def destroyIncorrectUsername(self, task):
        if task.time < 5.0:
            return task.cont
        else:
            self.incorrectUsername.destroy()
            return task.done
    def destroyServerError(self, task):
```

Serverside

- The server will parse the username/password strings and check with the database if it is stored.
- It will then send the results back to the client.

RequestLogin.java

```java
public class RequestLogin extends GameRequest {

    // Data
    private String username;
    private String password;
    // Responses
    private ResponseLogin responseLogin;

    public RequestLogin() {
        responses.add(responseLogin = new ResponseLogin());
    }

    @Override
    public void parse() throws IOException {
        this.username = DataReader.readString(dataInput);
        //System.out.println("Requesting --------");
        //System.out.println("username: "+username);
        this.password = DataReader.readString(dataInput);
        //System.out.println("password: "+password);
    }

    @Override
    public void doBusiness() throws Exception {
        responseLogin.setUsername(username);
        responseLogin.setPassword(password);

    }
}
```

ResponseLogin.java

```java
public class ResponseLogin extends GameResponse {

    private String username;
    private String password;

    public ResponseLogin() {
        responseCode = Constants.CMSG_AUTH;
    }

    @Override
    public byte[] constructResponseInBytes() {
        GamePacket packet = new GamePacket(responseCode);
        //packet.addString(username);
        //packet.addString(password);
        if((username.equalsIgnoreCase("jeff") && password.equals("Kiyo")) ||
                (username.equalsIgnoreCase("csula") && password.equals("Student"))){
            packet.addString("Authorized");
            packet.addString(username);
        }else{
            packet.addString("UnAuthorized");
            packet.addString(username);
        }

        return packet.getBytes();
    }
    public String getUsername(){
        return username;
    }
    public String getPassword(){
        return password;
    }
    public void setUsername(String message){
        this.username = message;
    }
    public void setPassword(String message){
        this.password = message;
    }
}
```

RequestRegister.java

```java
public class RequestRegister extends GameRequest {

    // Data
    private String username;
    private String password;
    // Responses
    private ResponseRegister responseRegister;

    public RequestRegister() {
        responses.add(responseRegister = new ResponseRegister());
    }

    @Override
    public void parse() throws IOException {
        this.username = DataReader.readString(dataInput);
        //System.out.println("Requesting --------");
        //System.out.println("username: "+username);
        this.password = DataReader.readString(dataInput);
        //System.out.println("password: "+password);
    }

    @Override
    public void doBusiness() throws Exception {
        responseRegister.setUsername(username);
        responseRegister.setPassword(password);

    }
}
```

ResponseRegister.java

```java
public class ResponseRegister extends GameResponse {

    private String username;
    private String password;

    public ResponseRegister() {
        responseCode = Constants.CMSG_REGISTER;
    }

    @Override
    public byte[] constructResponseInBytes() {
        GamePacket packet = new GamePacket(responseCode);
        //packet.addString(username);
        //packet.addString(password);
        if(username.equalsIgnoreCase("jeff") || username.equalsIgnoreCase("csula")){
            packet.addString("Same Username");
        }else{
            packet.addString("Success");
        }

        return packet.getBytes();
    }
    public String getUsername(){
        return username;
    }
    public String getPassword(){
        return password;
    }
    public void setUsername(String message){
        this.username = message;
    }
    public void setPassword(String message){
        this.password = message;
    }
}
```