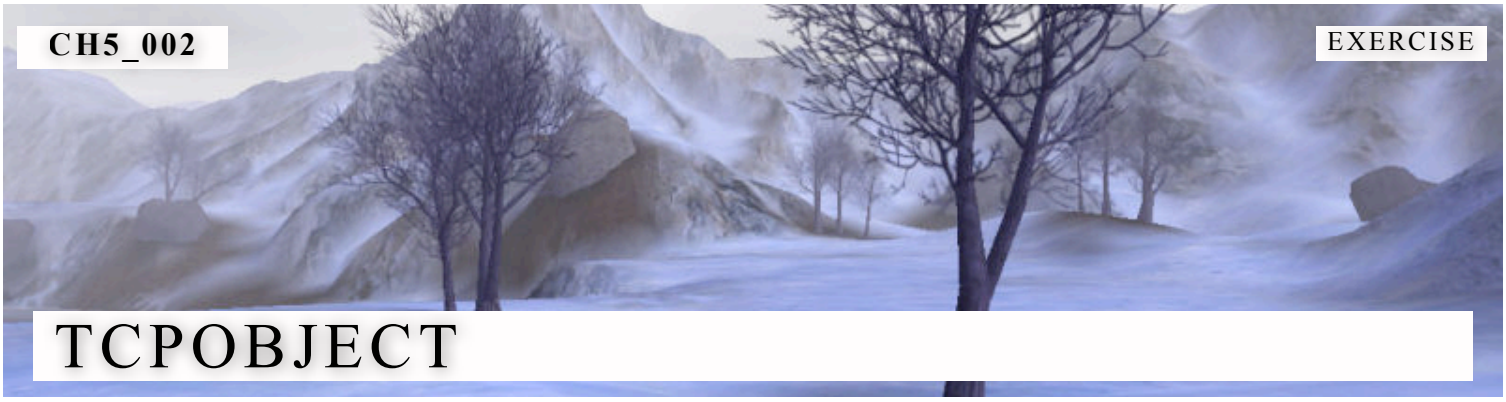

CH5_002

EXERCISE

TCPOBJECT

Exercise Files

Starter – "Kit/gpgt/server/scripts/gpgt/chapter5/exercise002.cs"
Answers – "Kit/gpgt/server/scripts/gpgt/chapter5/answers/exercise002_f.cs"

Exercise Mission

Chapter 5: "002_Communications: TCP Objects"

Synopsis

In this exercise, we will write a few different commands re-using and expanding upon the features we used in the chapter 3 exercises.

Prerequisites

1. *ch1_001.pdf "Using The Kit"*

Exercises

1. *Implement Callbacks (pg 2)*
2. *TCP0* - Connecting (pg 3)*
3. *TCP1* - Client to Proxy Messaging (pg 5)*
4. *TCP2* - Proxy to Client Messaging (pg 8)*
5. *TCP3* - Affect Of newline In Messaging (pg 10)*
6. *TCP4* - Disconnecting and Buffered Messages (pg 12)*
7. *TCP5* - Connecting (localhost Lookup) (pg 14)*

Copyright GarageGames.com 2000 - 2008 All Rights Reserved

TCPOBJECT

1 Implement Callbacks

Goal: Learn how to implement a simple version of all standard TCPObject callbacks.

Starter Code: You are provided with 6 callback bodies to start this exercise.

1. function TCPServer::onConnectRequest(%Obj , %addrBuf , %idBuf)
2. function TCPClient::onLine(%Obj , %line)
3. function TCPClient::onConnected(%Obj)
4. function TCPClient::onConnectFailed(%Obj)
5. function TCPClient::onDisconnect(%Obj)
6. function TCPProxyConnection::onDisconnect(%Obj)

Steps:

1. You will notice that the first callback, TCPServer::onConnectRequest(), already has an echo statement. Please modify the remainder to have similar statements. This kind of debug output will help us with our subsequent work in this exercise.
2. Please go back to chapter 5 and verify that all of the "suggested" callbacks are defined. If they are not, please implement bodies for the missing ones (in the appropriate namespace) and add echo statements to their bodies too.

Output Goal:

We can't test these callbacks yet. The work we will do in this remainder of this exercise uses them.

Hints:

1. See the "Suggested TCPObject callback scoping" table for help on step 2 (above.)

Copyright GarageGames.com 2000 - 2008 All Rights Reserved

2 TCP0* - Connecting

Goal: Learn how to write the code necessary to create TCPObject client, server, and proxy instances. Write the the code to connect to them as well.

Starter Code: You are provided with three function bodies to start this exercise.

```
// 1
function TCPServer::onConnectRequest( %Obj , %addrBuf , %idBuf )
{
   echo("TCPServer::onConnectRequest( " @ %Obj @ " , " @ %addrBuf @ " ,
        " @ %idBuf @ " ) " );

   // ADD MORE CODE HERE

   TCPProxyGroup.add($lastTCPObjectProxy);
}


//2
function TCP0()
{
   cleanUp(); // PLEASE KEEP

//    %server = ??????
//    TCPServerGroup.add( %server );
//    listen?
//    %client = ??????
//    TCPClientGroup.add( %client );
//    connect?

}
```

Steps:

1. Using the supplied body for function two above, please fill in the code necessary to do the following:

   Create a server named "TCPServer".

   Make the new server listen on port 5000.

   Create a client named "TCPClient".

   Tell the client to connect to the server at address "127.0.0.1".

2. Fill in the body function 1 above (onConnectRequest()) and have it create a proxy named "TCPProxyConnection".

Copyright GarageGames.com 2000 - 2008 All Rights Reserved

TCPOBJECT

Output Goal:

When you run the completed exercise, you should see output something like this.

```
==>TCP0();
Binding server port to default IP
TCPClient::onConnected( 1867 )
TCPServer::onConnectRequest( 1866 , IP:127.0.0.1:3131 , 1296 )
```

Hints:

None.

Copyright GarageGames.com 2000 - 2008 All Rights Reserved

TCPOBJECT

3 TCP1* - Client to Proxy Messaging

Goal: Learn how to send a simple message from a client to the proxy it is connected to and examine the effects of:

- Sending too early.
- Early send followed by good send.
- Delayed send.

Starter Code: You are provided with three function bodies to start this exercise.

```
// 1
function TCP1()
{
   TCP0();
}

// 2
function TCP1a()
{
   TCP0();
}

// 3
function TCP1b()
{
   TCP0();
}
```

(Please notice that all three of the provided functions are re-using TCP0(), so you should do that part of the exercise first, and then put your new code after our call to this function.)

Steps:

1. In the first function, TCP1(), please write the code necessary to send this message.

   ```
   "Client to Proxy -> Message 0\n"
   ```

2. In the second function, TCP1a(), please write the code necessary to send these messages where you send the first line immediately and delay the second line by 1 second.

   ```
   "Client to Proxy -> Message 0\n"
   "Client to Proxy -> Message 1\n"
   ```

Copyright GarageGames.com 2000 - 2008 All Rights Reserved

TCPOBJECT

3. In the third function, TCP1b(), please write the code necessary to send all four lines of this message, with each send delayed by 1 second.

```
"Client to Proxy -> Message 0\n"
"Client to Proxy -> Message 1\n"
"Client to Proxy -> Message 2\n"
"Client to Proxy -> Message 3\n"
```

Output Goal:

When you call each of the above respective functions, you should (*) see these outputs:

```
==>tcp1();
Binding server port to default IP
TCPClient::onConnected( 2335 )
TCPServer::onConnectRequest( 2334 , IP:127.0.0.1:3165 , 1468 )

==>tcp1a();
Binding server port to default IP
TCPClient::onConnected( 2338 )
TCPServer::onConnectRequest( 2337 , IP:127.0.0.1:3166 , 1476 )
```

(1 second delay occurs here)

```
TCPProxyConnection::onLine( 2339 , Client to Proxy -> Message 1 )

==>tcp1b();
Binding server port to default IP
TCPServer::onConnectRequest( 2340 , IP:127.0.0.1:3167 , 1460 )
TCPClient::onConnected( 2341 )
```

(1 second delay occurs here)

```
TCPProxyConnection::onLine( 2342 , Client to Proxy -> Message 0 )
TCPProxyConnection::onLine( 2342 , Client to Proxy -> Message 1 )
TCPProxyConnection::onLine( 2342 , Client to Proxy -> Message 2 )
TCPProxyConnection::onLine( 2342 , Client to Proxy -> Message 3 )
```

(*) I say *should* because if your machine is slower than mine, or if events conspire to allow the connection to establish before the send in tcp1(), then you may see the message get sent. However, in general, you should wait to send messages on a TCPObject instance until that instance has said it is connected to the destination.

Copyright GarageGames.com 2000 - 2008 All Rights Reserved

TCPOBJECT

Hints:

1. You will need to use the schedule method for delayed sends:

```
// set object name to Billy in 1.5 seconds
//
%obj.schedule( 1500, setName, "Billy" );
```

2. The delay we are using is to get around the time it takes for a connection to be established. The better way to do this is to use the onConnected() callback to tell us when it is OK to send messages.

Copyright GarageGames.com 2000 - 2008 All Rights Reserved

4 TCP2* - Proxy to Client Messaging

Goal: Send messages back to a client from the proxy object it is attached to.

Starter Code: You are provided with three function bodies to start this exercise.

```
// 1
function TCP2()
{
   TCP0();
}

// 2
function TCP2a()
{
   TCP0();
}

// 3
function delayedProxySend()
{
   $lastTCPObjectProxy.send("Proxy to Client -> Message 0\n");
}
```

(Please notice that the first two provided functions are re-using TCP0(), so you should do that part of the exercise first, and then put your new code after our call to this function.)

Steps:

1. In the first function, TCP2(), please write the code necessary to send this message. (Send the message immediately.)

   ```
   "Proxy to Client -> Message 0\n"
   ```

2. In the second function, TCP2a(), please write the code necessary to send this message. (Send the message after one second.)

   ```
   "Proxy to Client -> Message 0\n"
   ```

Copyright GarageGames.com 2000 - 2008 All Rights Reserved

TCPOBJECT

Output Goal:

When you call each of the above respective functions, you will see these outputs:

```
==>tcp2();
Binding server port to default IP
gpgt/server/scripts/gpgt/chapter5/exercise002_f.cs (154): Unable to find object: '2342'
attempting to call function 'send'
TCPServer::onConnectRequest( 2343 , IP:127.0.0.1:3185 , 1548 )
TCPClient::onConnected( 2344 )


==>tcp2a();
Binding server port to default IP
TCPServer::onConnectRequest( 2346 , IP:127.0.0.1:3186 , 1456 )
TCPClient::onConnected( 2347 )
TCPClient::onLine( 2347 , Proxy to Client -> Message 0 )
```

Questions:

1. Do you know why we receive an error message for the call to TCP2()?

Hints:

None.

Copyright GarageGames.com 2000 - 2008 All Rights Reserved

5 TCP3* - Affect Of newline In Messaging

Goal: Learn about message buffering and how newlines affect this.

Starter Code: You are provided with two function bodies to start this exercise.

```
// 1
function TCP3()
{
   TCP0();
}

// 2
function TCP3a()
{
   TCP0();
}
```

(Please notice that each of the two provided functions are re-using TCP0(), so you should do that part of the exercise first, and then put your new code after our call to this function.)

Steps:

1. In the first function, TCP3(), please write the code necessary to send this message. (Delay send by one second.)

   ```
   "Client to Proxy -> Message 0 (no newline); "
   ```

2. In the first function, TCP3a(), please write the code necessary to send this message. (Delay send by one second.)

   ```
   "Client to Proxy -> Message 0 (no newline); "
   "Client to Proxy -> Message 1 (w/ newline); "
   ```

Copyright GarageGames.com 2000 - 2008 All Rights Reserved

TCPOBJECT

Output Goal:

When you call each of the above respective functions, you will see these outputs:

```
==>tcp3();
Binding server port to default IP
TCPClient::onConnected( 2350 )
TCPServer::onConnectRequest( 2349 , IP:127.0.0.1:3209 , 1640 )


==>tcp3a();
Binding server port to default IP
TCPServer::onConnectRequest( 2352 , IP:127.0.0.1:3210 , 1312 )
TCPClient::onConnected( 2353 )
TCPProxyConnection::onLine( 2354 , Client to Proxy -> Message 0 (no newline); Client to
Proxy -> Message 1 (w/ newline) )
```

Questions:

1. Can you explain why the first function doesn't seem to send a message, while the second one does, but the messages comes out on the same line?
2. Is the message from the first function lost?

Hints:

None.

Copyright GarageGames.com 2000 - 2008 All Rights Reserved

TCPOBJECT

6 TCP4* - Disconnecting and Buffered Messages

Goal: Learn about the effect that disconnecting a TCPObject has on buffered messages. Examine both local and remote disconnecting versus message buffering.

Starter Code: You are provided with 3 function bodies to start this exercise.

```
// 1
function TCP4()
{
   TCP0();
}


// 2
function TCP4a()
{
   TCP0();

   TCPClient.schedule( 1000 , send , "Client to Proxy -> Message 0 (no newline); ");

   schedule( 1500 , 0 , delayedProxyDisconnect );
}

// 3
function delayedProxyDisconnect()
{
//   ($lastTCPObjectProxy).???????
}
```

Steps:

1. In the first function, TCP4(), please write the code necessary to send this message and then to disconnect the client. (Delay both actions by one second.)

   ```
   "Client to Proxy -> Message 0 (no newline); "
   ```

2. In the second function, delayedProxyDisconnect(), please write the code necessary to disconnect the proxy from the client it is attached to.

Copyright GarageGames.com 2000 - 2008 All Rights Reserved

TCPOBJECT

Output Goal:

When you call each of the above respective functions, you will see these outputs:

```
==>tcp4();
Binding server port to default IP
TCPServer::onConnectRequest( 2768 , IP:127.0.0.1:3395 , 1880 )
TCPClient::onConnected( 2769 )
TCPProxyConnection::onLine( 2770 , Client to Proxy -> Message 0 (no newline);  )
TCPProxyConnection::onDisconnect( 2770 )


==>tcp4a();
Binding server port to default IP
TCPClient::onConnected( 2772 )
TCPServer::onConnectRequest( 2771 , IP:127.0.0.1:3410 , 1888 )
TCPClient::onDisconnect( 2772 )
```

Questions:

1. When we called TCP4(), the message got to its destination even though we did not use a newline. Why is this?
2. When we called TCP4a(), the client-to-proxy message was lost. Why is this?

Hints:

None.

Copyright GarageGames.com 2000 - 2008 All Rights Reserved

7 TCP5* - Connecting (localhost Lookup)

Goal: See an alternative addressing style in action.

Starter Code: You are provided with one function body to start this exercise.

```
// 1
function TCP5()
{
   cleanUp(); // PLEASE KEEP
}
```

Steps:

1. Please re-write the code for TCP0, but this time instead of using the address "127.0.0.1", use the hostname equivalent.

Output Goal:

When you call the above function, you will see this output:

```
==>tcp5();
Binding server port to default IP
TCPClient::onDNSResolved( 2775 )
TCPServer::onConnectRequest( 2774 , IP:127.0.0.1:3434 , 1992 )
TCPClient::onConnected( 2775 )
TCPProxyConnection::onLine( 2776 , Client to Proxy -> localhost works too! )
```

Hints:

None.

Copyright GarageGames.com 2000 - 2008 All Rights Reserved