

1702330 - Networked Applications 2

by Daniel Newman

FILE	1623_DANIEL_NEWMAN_1702330_-_NETWORKED_APPLICATIONS_2_252504_542784617.DOCX (1.32M)		
TIME SUBMITTED	30-APR-2020 02:57PM (UTC+0100)	WORD COUNT	1467
SUBMISSION ID	125682106	CHARACTER COUNT	7188

Student Name: Daniel Newman

Student Number: 1702330

Module Code: ACCA6015

Module Title: Networked Application

Lecturer: Dr Tim Bashford

Table of Contents

Table of Figures.....	2
Networked Applications	3
Introduction	3
Features and Design of Application	3
Features	3
Design.....	3
Demo Testing	6
Best Practice.....	8
How does the application adhere to best practice?	8
Evaluation	8
Conclusion.....	9
References	9

Table of Figures

Figure 1 - Application Design	4
Figure 2 - Source.cpp	4
Figure 3 - Source.cpp 2.....	4
Figure 4 - QuizGameNetwork.cpp.....	5
Figure 5 - QuizGame.cpp.....	6
Figure 6 - Error Messages present in code	7
Figure 7 - Demo - Part 1	7
Figure 8 - Demo - Part 2	7
Figure 9 - Demo - Part 3	8

Networked Applications

Introduction

In this report the author will discuss the features and design of their developed application. The author will also discuss best practice in relation to IPv4 addresses and Networked applications in general.

Features and Design of Application

Features

The author has created a Quiz game application, and so there are appropriate features in the application. Player 1 and Player 2 first enter their names and their name details are saved to refer to the players in the game. Player 1 will then be asked 10 questions. Their points will be saved if they get the answers correct. Player 2 will begin their turn and answer the same questions as the other player. Both player's points will be revealed after Player 2 has finished answering their questions as well as the game printing out the answers to both players and deciding on a winner.

As for network features, the players should be able to connect together and disconnect when they want. As well as this, the quiz should operate over a network to send information back and forth between the users and notify them of their turns beginning.

Design

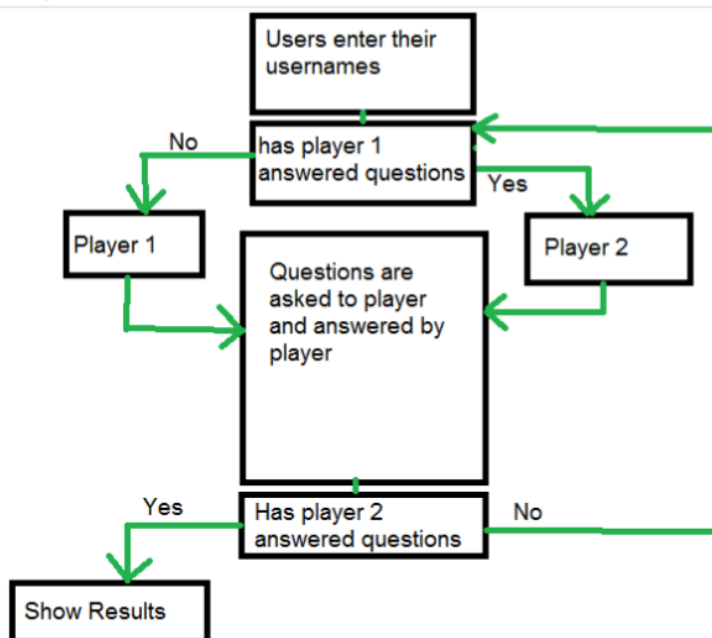


Figure 1 - Application Design

Above is a design for how the basic principles of the game should operate. There are two points in the system where the game checks if the users have answered questions to continue the loop for the second player.

```

14  using namespace qmessages;
15  using namespace std;
16
17  void ClientHandler(ConnectionInstance* ci)
18  {
19      Message* m;
20      ConnectedPlayer* player;
21
22      while (true)
23      {
24          string msgStr = ci->ReceiveString();
25          m = Message::Unserialise(msgStr);
26
27          if (m->GetType() == MessageType::connect)
28          {
29              player = new ConnectedPlayer(((ConnectMessage*)m)->GetUsername(), ci);
30
31              if (player->GetPlayerName().length() < 2)
32              {
33                  Message* reply = new RejectMessage("Error: Please pick a longer username to connect (at least 3 letters).");
34                  ci->SendString(reply->Serialise());
35                  delete reply;
36                  delete player;
37              }
38              else if (!Lobby::Inst().GetGame()->AddPlayer(player))
39              {
40              }
41          }
42      }
43  }
44
45  }
46

```

Figure 2 - Source.cpp

In the source.cpp file player info is collected from a connection message through the use of a network server, it gathers user's usernames and adds the players to the game. The source file has events on what to display depending on the user entering non-accepted information, these come in the form of accepted messages for approved actions and reject messages for unapproved actions.

```

1  }
2
3  int main(int argc, char** argv)
4  {
5      INet4Address* address = new INet4Address(50018);
6      ServerConnection* server = new ServerConnection(address);
7
8      server->StartServer(100);
9
10     while (true)
11     {
12         ConnectionInstance* ci = server->AcceptClient();
13         thread(ClientHandler, ci).detach();
14     }
15
16     system("pause");
17 }

```

Figure 3 - Source.cpp 2

In the main, the INet4Address calls the IP Address and connects it to the server, this is so that the players can connect to the server together.

```
void QuizGameNetwork::PlayGame(ConnectedPlayer* tPlayer)
{
    Player* winner = 0;
    int player2readycheck = 0;
    int player1score = 0;
    int player2score = 0;

    while (winner < 0)
    {
        Player* currentPlayer = qg->GetCurrentPlayer();

        // Question 1
        cout << currentPlayer->GetPlayerName() << " What's the capitol of France? ";
        string answer;
        cin >> answer;

        if (player2readycheck == 1)
        {
            if (answer == "Paris")
            {
                player2score = player2score + 1;
            }
        }
        else if (answer == "Paris")
        {
            player1score = player1score + 1;
        }

        // Question 2
        cout << currentPlayer->GetPlayerName() << " What country was Tetris created? ";
        cin >> answer;
```

Figure 4 - QuizGameNetwork.cpp

In the QuizGameNetwork.cpp file then is where the quiz game logic is stored. A variable named "winner" is set to 0, this variable is set to 1 once both of the players have taken their turns, and this is how the game figures out that both players have taken their turns. Player2readycheck has a similar purpose except it checks that player 1 has taken their turn so that the turns can swap for player 2 to take their turn. Then questions are asked to the players.

```

21
22 Player* QuizGame::GetPlayer(SquareState ss)
23 {
24
25     switch (ss)
26     {
27     case player1:
28         return p1;
29         break;
30     case player2:
31         return p2;
32         break;
33     default:
34         return nullptr;
35     }
36 }
37
38 void QuizGame::SwapTurn()
39 {
40     switch (playerTurn)
41     {
42     case player1:
43         playerTurn = player2;
44         return;
45     case player2:
46         playerTurn = player1;
47         return;
48     }
49 }
50
51 //Player* QuizGame::GetCurrentPlayer()

```

Figure 5 - QuizGame.cpp

QuizGame.cpp has the role of getting and swapping players. It very simply uses a case system so that if the case checks that player 1 is the current player then player 2 is what the player state switches to. There are many other network based .cpp files in this application such as Message.cpp and INet4Address.cpp, many of these are class examples with a few changes to allow them to be compatible with the quiz game whether that be by having them use different variables and classes relevant to the quiz game. Some of this code may not work or be correct however this was untestable on the author's side as an error was present in the code that the author struggled to fix, restarting the development of the project also failed to correct this issue and so when converting to a network version of the app the code was untestable. In the next heading a demo will be shown of how the application should operate.

Demo Testing

The following section is not representative of the final application, unfortunately an error in the original application meant that the program did not operate due to an unresolved external symbol error, this is likely a circular dependency issue that the author has tried and failed to correct. However, a demo project has been created to demonstrate how the application should operate.

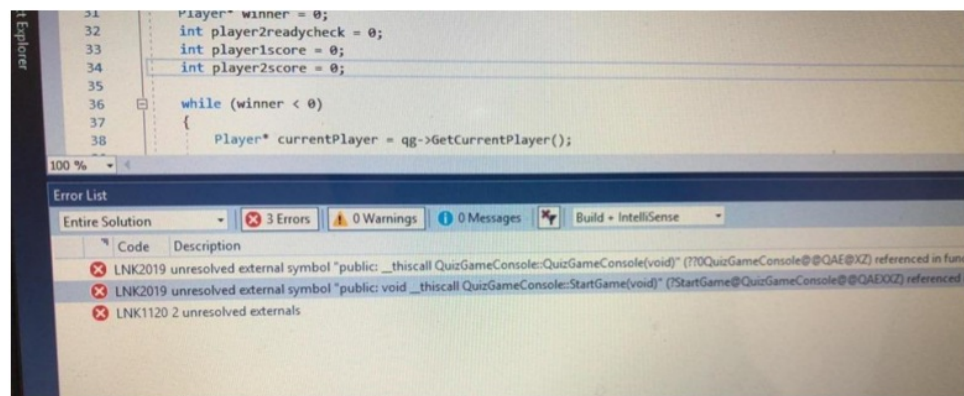


Figure 6 - Error Messages present in code

```
Enter your name Player 1: Daniel
Enter your name Player 2: Cameron
What's the capitol of France?
```

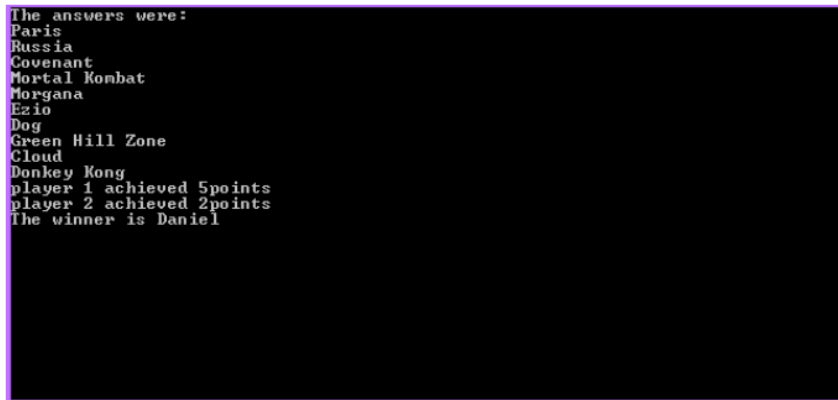
Figure 7 - Demo - Part 1

In the above image, the system has asked Player 1 for their name and then asked for the second player's name. Then the first question is asked to the first player.

```
Enter your name Player 1: Daniel
Enter your name Player 2: Cameron
What's the capitol of France? Paris
What country was Tetris made in? Russia
What is the name given to the alien race in the first 3 HALO games? Covenant
What game series is the character Sub-Zero from?
```

Figure 8 - Demo - Part 2

Above is a demonstration of how the questions are to be asked to the user, and an example of the user answering questions. The quiz does not say whether they are right or wrong as this is revealed to both players at the same time at the end.



```
The answers were:
Paris
Russia
Covenant
Mortal Kombat
Morgana
Ezio
Dog
Green Hill Zone
Cloud
Donkey Kong
player 1 achieved 5points
player 2 achieved 2points
The winner is Daniel
```

Figure 9 - Demo - Part 3

The next image is what appears when both users have answered their questions. The answers are displayed and the points are shown to the users and the winners name is shown.

Best Practice

Best practice is a set of methods or common solutions to problems that are generally accepted to be the best methods of achieving a goal. For example, if it's generally accepted that using functions to cut down on development time is the best method of cutting costs then that would be considered the best practice. [1]

How does the application adhere to best practice?

Businesses all over the world are using networks and networking tools to help their products stand out, whether it be mobile phone companies allowing customers to talk to each other or retail stores sending emails to customers, networking is one of the most useful practices in development. The application developed adheres to this practice by allowing networking between users so that they can connect with each other and play a short quiz together. [2] As well as this it's accepted that the majority of internet users are using IPv4 addresses and not IPv6. IPv4 addresses are compatible with the quiz game, though in a future development IPv6 could be added but it's considered best practice to develop for IPv4 first as far more users exist for it. [3]

Evaluation

The author believes that some things in this assignment could have gone better and some things went well. In terms of development due to a few errors present in the original version it meant that code became untestable and unable to run in general. Many different attempts to fix these issues were made however no fix worked, the author also attempted to start from scratch and redo their project, however the same error messages were present in the second attempt too. This does unfortunately mean that the code isn't perfect and a lot of the networking code could very well be incorrect however at least in the author's case no error messages beyond the ones that were present in a previous image would appear so there was no way to tell what was and was not accurate. With that being said a lot of effort was made by the author to attempt to fix these shortcomings and the code that has been produced, though untestable have logic used in them that

is accurate and could potentially work if the initial errors were fixed. The author worked very hard to get a good application and though could not get the main one working, the author did take the initiative to create a simpler version of the project to demo as a proof of concept and a demonstration of how the project was meant to operate.

Conclusion

To conclude this report the author has demonstrated the features and the design of the application, shown how the program is programmed and how it should work in the form of a demo. In terms of best practices this assignment allowed the author to experiment with many different practices and demonstrate an understanding of how they work and why they're important.

References

- [1] Cambridge Dictionary, "best practice," Cambridge Dictionary, [Online]. Available: <https://dictionary.cambridge.org/dictionary/english/best-practice>. [Accessed 19 April 2020].
- [2] K. Kassabov, "Creative Networking Tips For Small Businesses," Forbes, 4 September 2018. [Online]. Available: <https://www.forbes.com/sites/theyec/2018/09/04/creative-networking-tips-for-small-businesses/#2adce4ae6b88>. [Accessed 19 April 2020].
- [3] J. Gold, "6 years on, IPv4 still dominates IPv6," Network World, 6 June 2018. [Online]. Available: <https://www.networkworld.com/article/3278256/isoc-ipv6-usage-still-growing-after-six-years-on-the-march.html>. [Accessed 19 April 2020].

1702330 - Networked Applications 2

GRADEMARK REPORT

FINAL GRADE

29/100

GENERAL COMMENTS

Instructor

Unfortunately, this is clearly a long way from finished - you mention the issue with a linker error, but I couldn't get the network version of the application even that far - it seems very early in development. The non-networked version works OK, but doesn't quiet meet the brief. Still, an attempt was made, with a reasonable report.

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9
