MECAC WEEKLY REPORT (March 11 – March 17)

This week we have dealt with the complications in the client component. When we have implemented Event Matrix in server network component, we have observed that the client game state crashes. When the bandwidth is small compared to number of players, even close by clients can disappear. We have inspected the source code for possible reasons of this case. Since the game server was sending all client state information to all clients at every game tick before our modification, clients update the opponent positions according to last game tick. Here is a part of the code that generated the bug:

```
public void updateOpponents(List<object> message)
        {
            int howMany = (int)message[1];
            List<bool> connections = new List<bool>();

                for (int i = 0; i < opponents.Count; i++)
                   {
                       connections.Add(false);
                   }
                for (int i = 0; i < howMany; i++)
                {
                    string name = (string)message[i * 13 + 2];
                    int k;

                    for (k = 0; k < opponents.Count; k++)
                    {
                        if (opponents[k].accountName == name && myAccountName !=
opponents[k].accountName)
                            break;
                    }
                    if (k != opponents.Count)//this means this is in my opponent
list
                    {
                        connections[k] = true;
                    ...

                    }
                    else//is not in my opponent list
                    {
                        Client c = new Client();
                    ...
                        }

                    }
                }
                for (int i = 0; i < connections.Count; i++)
                {
                    if (!connections[i])
                    {
                        mNPCManager.RemoveNPC(opponents[i]);
                        connections.RemoveAt(i);
                        opponents.RemoveAt(i);
                    }
                }

        }
```

This code assumes that the server sends position information of all clients at every game tick. We have changed the client game logic component not to rely on this assumtion. Pseudo-code for what we have done is given below:

```
public void updateOpponents(List<object> messages)
        {

              foreach( message in messages )
              {
                     if(message comes from a client I have the last position)
                     {
                           Update cache[message.from] = message.position;
                     } else {
                           Create a cache.add(message.from, message.position);
                     }
              }

              RENDER_ALL_IN_CACHE();
        }
```

This modification is not trivial as it requires each client to identify its opponents uniquely. Therefore, we have modified the server network component to send unique identifiers in each update message. This solved the implications of the modifications on the server network component.

Next, week we will focus on designing the web page for the Virtual Turkey project and prepare the configuration management report.