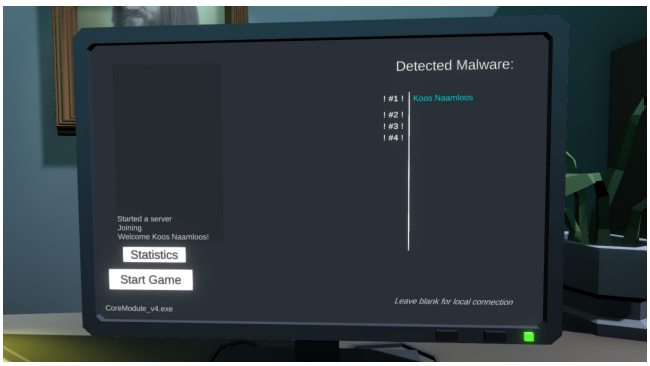
CoreModule_v4.exe

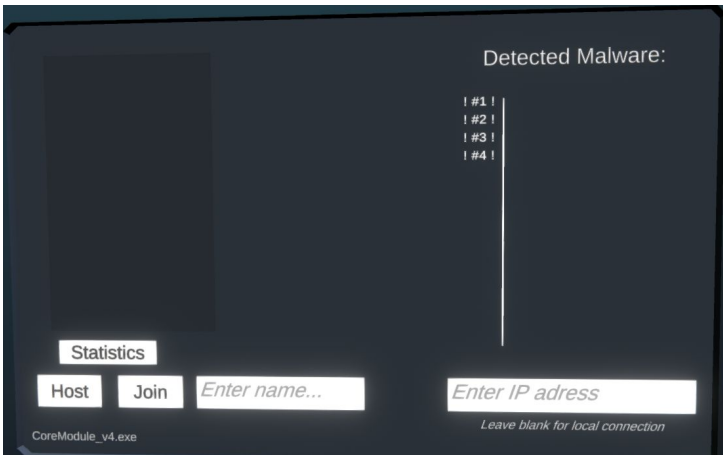
Inleiding:

Voor deze kernmodule kregen we de opdracht om een online multiplayer game te maken in Unity. Daarbij hebben we ook de opdracht gekregen om een database te maken waar de scores van de game in kunnen worden opgeslagen.

Het moeten sturen in lezen van messages gaf mij een “hacker” gevoel. Dit gevoel werd mijn inspiratie om het spel te designen in het thema “hacker”. Het idee is dat de speler een computer virus is die belangrijke bestanden (bank inlog gegevens) van de computer moet stelen. De monsters die de bestanden verdedigen zijn firewalls! De naam van het spel is toepasselijk: **“CoreModule_v4.exe”** en hieronder kun je zien hoe het er uiteindelijk uit is komen te zien.



Game Flow



De game flow gaat als volgt. De speler begint het spel met het scherm wat je op de afbeelding rechts kunt zien. Er zijn 3 knoppen waar de speler op kan drukken. Als de speler op de “Statistics” knop drukt zal de camera verplaatsen naar het statistics scherm (ik had helaas geen tijd om de statistieken in de game te implementeren, maar ik had het scherm al wel af dus deze heb ik er maar ingelaten). Hiervandaan kan de speler weer terug naar het beginscherm.

De speler kan zijn of haar naam in voeren en heeft ook de optie om een IP-adres in te voeren. Als deze leeg worden gelaten zal hier een standaard waarde aan worden gegeven. Drukt de speler op “Join” dan zal er een “ClientBehaviour” script aangemaakt worden. Dit script probeert te connecten naar een server met het gegeven IP-adres. Drukt de speler op “Host” dan zal er een “ServerBehaviour” script aangemaakt en vervolgens ook een “ClientBehaviour” die met zichzelf connect.

Bij zowel de host als de client worden alle knoppen onbruikbaar, maar de host krijgt een “Start Game” knop erbij. Als de host hierop drukt word het spel gestart (wie had dat verwacht).

Het spel word gespeeld volgens het Game Protocol. Hier is niet veel interessants aan op te merken dus het lijkt me vrij zinloos om in mijn eigen woorden te vertellen wat het Game Protocol eigenlijk al glashelder uitlegt.

Als elke speler dood of uit de “dungeon” is stopt het spel en krijgt iedereen de uiteindelijke score te zien met hun naam erbij (en krijgen ze een close-up van het schilderij in de achtergrond ;)).

Code Structure

Het implementeren van alle messages was een moeizame taak en leverde enorm veel regels code op. Hierdoor heb ik het zo gestructureerd mogelijk geprobeerd aan te pakken. Zowel de “ClientBehaviour” en “ServerBehaviour” hebben een sectie voor “Send Messages” en “Handle Incoming Messages”. Voor elke message die binnenkomt of verstuurd moet worden heb ik een aparte functie aangemaakt. Soms roept een functie maar een paar regels code aan, maar door dit consistent te doen werd mijn code een stuk duidelijker om te lezen.

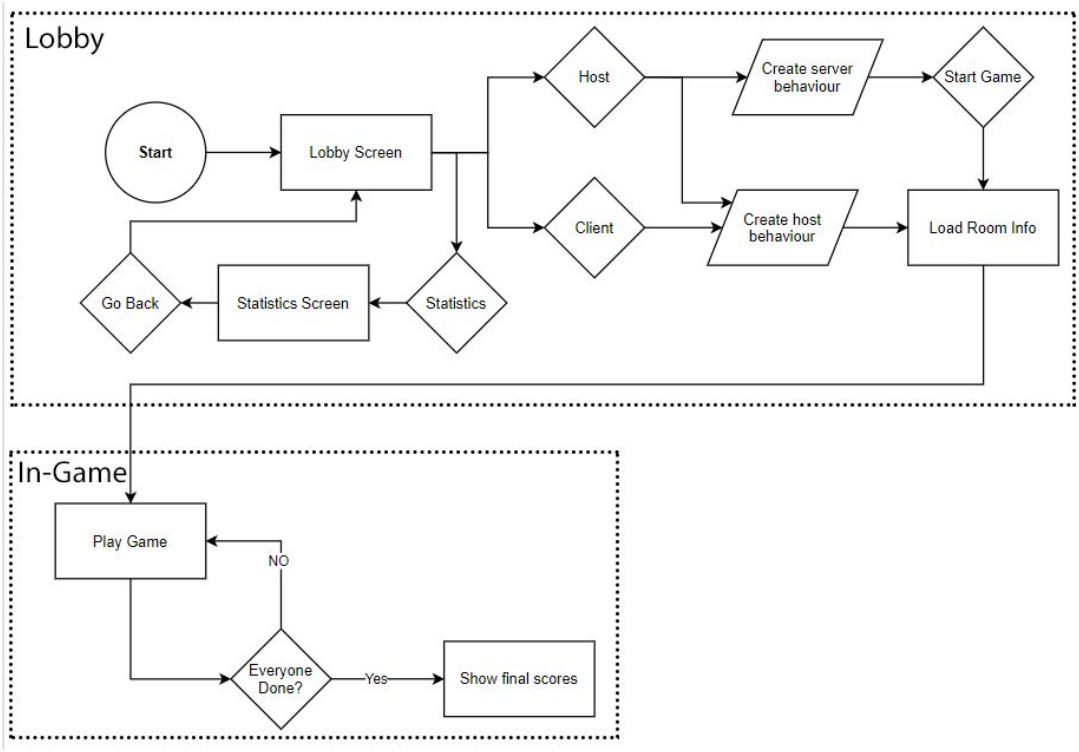
Het encoden en decoden van een message handel ik af met het script “MessageHandler”. Deze heeft de functie “SendMessage” en “ReadMessage” die ik aanroep wanneer ik een message ontvang of wil sturen.

Ik heb een “GameManager” script die een hoop losse front-end functionaliteit regelt (denk hierbij aan: functionaliteit voor knoppen, het printen van message details, switchen van lobby naar game view, etc.).

Ook heb ik een “PlayerManager” script die alle info over de spelers bijhoudt. Zowel de client als de host maken gebruik van dit script.

Het script “HostDataManager” regelt alle data die alleen de host nodig heeft. Denk hierbij aan het grid van rooms, welke speler er aan de beurt is, etc.).

En verder is er nog een “UIManager” script die alle knoppen aan of uit zet en de juist informatie aan de speler weergeeft (naam, aantal ander spelers, etc.).



PMI

Plus

- Ik heb enorm veel geleerd over het werken met een netwerk in Unity.

Min:

- Het was enorm veel werk om alles werkend te krijgen.

Interessant:

- Over een aantal jaar zal er vast weer een nieuw systeem zijn voor multiplayer in Unity