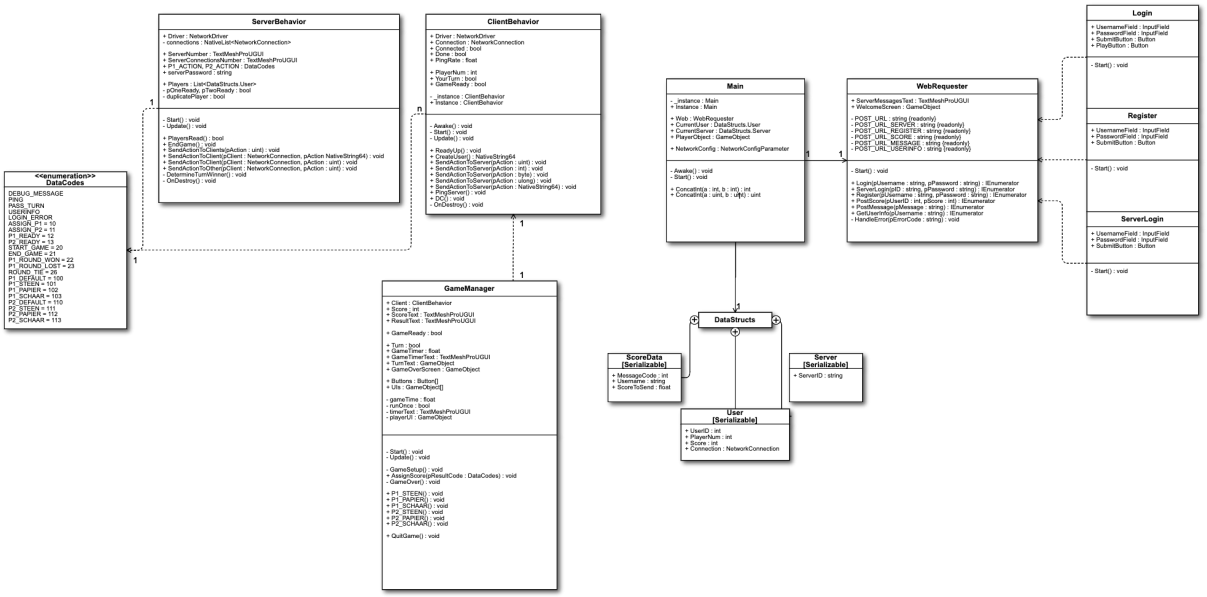
Online Steen, Papier, Schaar.

Mijn multiplayer game is eigenlijk gewoon een Steen, Papier, Schaar clone. Eerst was het idee om er een soort runner/blocker spel van te maken, maar dat bleek uiteindelijk niet zo handig.
Mijn allereerste concept (het runner/blocker spel) was ontstaan uit een quick digital prototype die ik hiervoor had gemaakt voor het gelijknamige vak. Het spel was een 2D platformer waarin je misbruik kon maken van sommige “bugs”. Om zo jumping puzzels op te lossen. Het uiteindelijke doel van het spel was om de prinses te redden, maar die zat in de onderwereld… En hoe kom je als dood normaal mens in de onderwereld? Precies, je moet sterven. Wanneer je dood gaat in het spel wordt je naar de onderwereld gestuurd, en dit is waar het spel pas echt begint. In de onderwereld wordt het spel wat meer een platformer, waarbij je de “bugs” in het spel gaat ontdekken, en gebruiken.
Een voorbeeld van een van de bugs is dat je je snelheid oneindig kan verhogen door tegen een muur aan te blijven springen. Na een gegeven punt kom je een vulkaan tegen waar je overheen moet springen, maar dit is absoluut onmogelijk zonder de bug te gebruiken. Er is een klein stukje in de vulkaan waar je tegen de muur naar beneden kan glijden, voordat je de lava aanraakt (en daardoor doodgaat). Als je vervolgens tegen deze muur aan blijft springen en je input op het juiste moment omdraait, kan je met de gegenereerde extra snelheid over de vulkaan heen springen.
Anyway, deze documentatie gaat over het multiplayer spel, dus laat ik niet te veel afdwalen.
Bij het eerste concept voor de multiplayer game was het de bedoeling dat het een turn-based spel zou zijn, maar dat zou niet werken, omdat speler 1 zou moeten kiezen voor een obstakel, waar speler 2 op zou moeten reageren. Helaas zou het een beetje ingewikkeld zijn, en naar mijn mening ook niet zo heel eerlijk als speler 1 wel punten zou kunnen scoren, maar speler 2 niet.
Hierdoor kwam ik eerst met het idee om ze dan wel allebei punten te kunnen laten scoren, maar dat zou betekenen dat ik 2 verschillende leaderboards bij zou moeten houden en ik wilde het niet te ingewikkeld voor mezelf maken, aangezien het een herkansing was die ik zo snel mogelijk af wilde maken.
Uiteindelijk besloot ik om het toch maar turn based te maken, een om het zo simpel mogelijk te houden ben ik maar gewoon met het welbekende spel “Steen, Papier, Schaar” gegaan. Het is een competitief spel, waarbij het per spel de bedoeling is dat je zo veel mogelijk rondes moet winnen binnen 1 minuut. Ik zeg competitief, maar het is dus eigenlijk ook de bedoeling dat je samen zo snel mogelijk het spel speelt. Dit zorgt er voor dat het iets moeilijker is om een lastige score te halen. Het is in principe natuurlijk niet de bedoeling dat je gaat communiceren met je tegenstander in bijvoorbeeld een local-play situatie.

Om een beetje duidelijk te maken hoe de code in elkaar zit heb ik even snel een UML diagram in elkaar gezet. Zie hieronder.



Voor het geval dat het plaatje niet te lezen is [klik hier](#) voor een duidelijker overzicht.

De executable start op en je wordt als eerste in een main menu gezet. Je kan niet spelen zonder in te loggen. Als je nog geen account hebt moet je je eerst registreren. De `Main` class is de container voor alle “login” functionaliteit. Deze class voegt events toe aan de knoppen in het spel die er voor zorgen dat de juiste Enumerators worden aangeroepen in de `WebRequester` class zodat de speler kan inloggen of registreren. Alle info die teruggegeven wordt aan het spel wordt opgeslagen in `DataStructs` structs zodat deze gebruikt kunnen worden voor latere calls terug naar de server. Als de speler is ingelogd dan kunnen ze op start drukken. Zodra ze dit doen worden ze in een laadscherm gezet, waarna er een `ClientBehavior` op de `PlayerObject` in de Main class wordt gezet. Deze gaat automatisch als het wordt aangemaakt verbinden met de server.
Eenmaal verbonden met de server, dan gaat de speler op zoek naar een andere speler. Of wat er *eigenlijk* gebeurt is dat de speler wacht totdat de server aangeeft dat er een 2e speler is verbonden met de server. Als dit is gebeurt wordt er vanuit de `ServerBehavior` (server) een signaal gestuurd naar beide `ClientBehavior` objecten (de spelers). Dit gaat doormiddel van het versturen van `uints`. Om ervoor te zorgen dat de `ClientBehavior` en `ServerBehavior` objecten weten wat ze moeten doen bij het ontvangst van specifieke messages worden deze onderscheiden met `DataCodes`.
Het spel wordt hierna gestart.
De spelers worden allebei naar een nieuwe scène gestuurd, met daarin de `GameManager`. De `ClientBehavior` objecten worden niet vernietigd, dus deze blijven persistent als ze de nieuwe scène in worden gezet. Deze client objecten worden vervolgens direct aan de `GameManager` verbonden, voor future reference. Na een korte countdown kunnen de spelers om de beurt een van de 3 opties kiezen; steen, papier of schaar. Alle 3 de opties hebben hun eigen button function die een message naar de server stuurt met de gekozen actie. Als beide spelers dit hebben gedaan wordt er in de `ServerBehavior` met de `DetermineTurnWinner()` functie vergeleken en bepaald welke speler de ronde wint. Dit

wordt vervolgens met een message terug naar beide spelers gecommuniceerd, waarna de punten worden opgeslagen in de server en lokaal worden aangepast.
Na 60 seconden wordt het spel stopgezet en de scores vanuit de `ServerBehavior` naar de database gestuurd!