Software Development

Maandag 8 Juni

Technisch verslag



Hendrik Wiersma

Bauke Jansen

Contents

[Inleiding 3](#_Toc421613083)

[Controller 4](#_Toc421613084)

[Connectie 4](#_Toc421613085)

[Logica 4](#_Toc421613086)

[Client 5](#_Toc421613087)

[Connectie 5](#_Toc421613088)

[Pathfinding 5](#_Toc421613089)

[Waypoints 5](#_Toc421613090)

[Rain 5](#_Toc421613091)

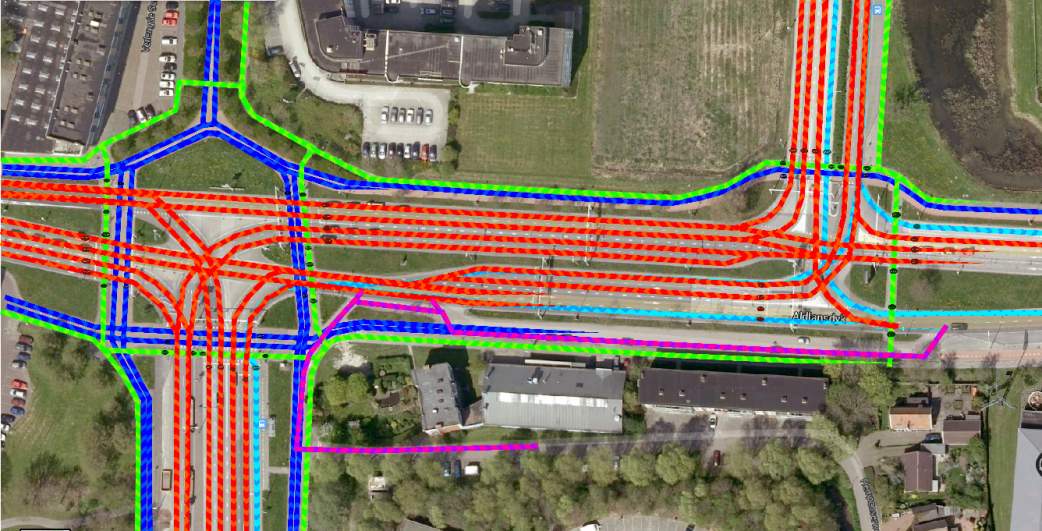
[Spawnpoint 6](#_Toc421613092)

[AI Controller 6](#_Toc421613093)

[Conclusie 7](#_Toc421613094)

# Inleiding

Voor het vak softwaredevelopment werd de kruispunt opdracht gegeven. Hierbij is het de bedoeling dat een bestaand kruispunt word gesimuleerd. Dit jaar is er gekozen voor een dubbelle T-splitsing ten zuiden van Leeuwarden als het te simuleren kruispunt.



Hierbij is het de bedoeling een client onderdeel te maken, welke het kruispunt visueel simuleert en een controller, welke de stoplichten bedient. Deze twee onderdelen moeten onderling kunnen communiceren en moeten bovendien ook met de client of controller van de andere studenten kunnen samenwerken. Verder mogen de onderdelen binnen de groep niet in dezelfde taal worden geschreven. Tijdens deze opdracht moeten we regematig bij elkaar komen om de onderdelen met elkaar te testen. De nadruk van deze opdracht ligt op de samenwerking en het vastleggen van een protocol.

# Controller

De controller is verantwoordelijk voor het spawnen van voertuigen en het besturen van de stoplichten. We hebben gekozen om dit programma te schrijven in java.

## Connectie

De connectie word tot stand gebracht met behulp van sockets. Eerst word een serversocket gemaakt op het gekozen poort nummer. Deze zoekt vervolgens naar een verbinding met een client waaruit een client socket word verkregen. Hiermee worden vervolgens een in- en outputstream aangemaakt. De outputstream word gebruikt om data naar de client te verzenden. Op de inputstream word constant geluisterd of er nieuwe data is ontvangen welke vervolgens verkwerkt word.

Het connecten en luisteren naar nieuwe data gebeurt op een apparte thread. Dit zodat de interface actief blijft tijdens deze acties.

## Logica

De controller volgt een paar simpele regels voor het op groen zetten van de stoplichten. Er word eerst gekeken of er ook bussen op de busbaan staan. Deze zullen altijd voorang krijgen. Als dit niet het geval is word naar de drie “hoofdwegen” gekeken. Bij deze wegen worden de meeste andere wegen afgesneden. Vervolgens worden hierbij uit de overige wegen gezocht naar welke nog open kunnen. Tot slot word gekeken of er inplaats van deze overige wegen ook nog een fiets- of voetpad kan worden gebruikt. Dit samen zorgt voor een optimale solutie voor het doorstromen van het verkeer.

# Client

De Client zorgt voor een visueel beeld van het kruispunt. Deze ontvangt de voertuigen en zorgt ervoor dat ze naar hun locatie gaan. Verder stuurt deze als er voertuigen voor het stoplicht staan en ontvangt de status van de verschillende stoplichten.

De simulatie is geschreven in C# en gemaakt in 3D met behulp van Unity.

## Connectie

Als de client start zal er verbinding gemaakt worden met de server. Hoewel het ipaddres elke keer makkelijk te veranderen is zal de poortnummer altijd op 10000 staan. Dit is volgens de documentatie die we binnen de SDM groep hebben vastgesteld. Er zijn drie verschillende packets die de client kan ontvangen. Het eerste packet is om voertuigen te spawnen. Hierin staat de beginrichting, eindrichting en het type voertuig. Dit is alle informatie de client nodig heeft om voertuigen aan de simulator toe te voegen. Het tweede packet is een stoplicht signaal. Hierin staan alleen het id van het stoplicht en de staat can het stoplicht. Dit kan rood, oranje en groen zijn. Het derde en laatste packet is bedoelt om voertuigen aan te melden. Zo weet de controller dat er bijvoorbeeld een auto voor het stoplicht staat te wachten. Dit packet wordt op twee momenten verzonden. Eens als het voertuig zich aanmeld. En nogmaals als de voertuig onder het stoplicht door rijd.

## Pathfinding

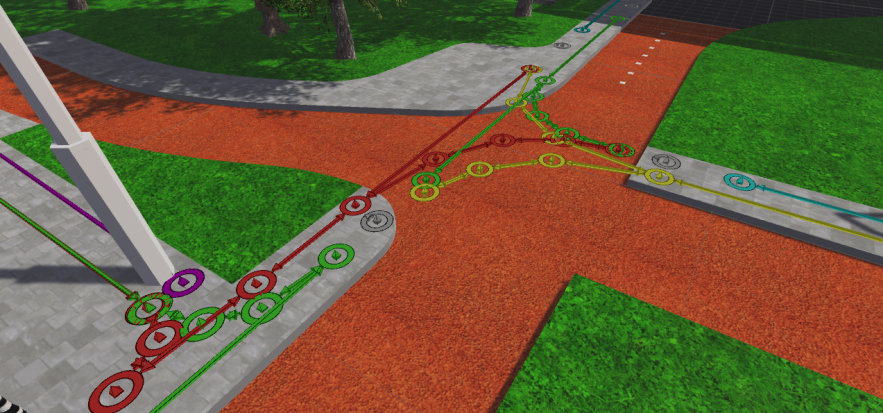
### Waypoints

We zijn begonnen met de pathfinding van de autos. Door middel van waypoints kan een auto een parkour afleggen. Elke waypoint heeft een collider op zich. Dit betekend dat er een functie geactiveerd kan worden waneer er een object er doorheen gaat. Elke keer als de auto door een waypoint rijd worden er een paar variabelen gecheckt. Zo zal de auto het nummer van de baan en de richting verifiëren. Als dit allemaal klopt is de volgende waypoint de huidige target.

De engine zal alle botsingen en alle overige physics calculeren. Dit zorgt ervoor dat de auto’s onder anderen dempende wielen, een remweg en uitlaatgassen hebben. Dit kan voor problemen zorgen sinds dit niet in de client van anderen hoeft te zitten. Een 2d representatie van de client hoeft bijvoorbeeld geen rekening te houden met auto’s en fietsers die vastzitten.

### Rain

Rain is een Unity plugin voor pathfinding. Het geeft een implementatie van waypoints, navigation mesh en een combinatie hiervan. Voor dit project is gebruik gemaakt van waypoints.

Elk voertuig heeft een eigen AI met behaviour tree. Deze zoekt naar een volgende waypoint, en kijkt of deze vrij is. Zoniet, dan word het voertuig bevroren totdat er weer ruimte is.

### Spawnpoint

Als de client een bericht krijgt om een voertuig te spawnen zullen er een paar dingen gebeuren. Ten eerste zal er onderscheid gemaakt worden in het voertuig. Een auto gebruikt namelijk een ander systeem dan de fietsers en voetgangers. Als een auto of bus gekozen is zal de client kijken naar de richtingen die in de packet zitten. Eerst zal de client kijken of de spawnpoint beschikbaar is. Als er al een voertuig op deze plek staat zal er niks gespawned worden. Anders zullen de models in elkaar zitten en uiteindelijk uit elkaar springen. De volgende stap is om de juiste waypoint collectie te kiezen. De client doet een kleine zoekfunctie om de juiste eindpunt te vinden. Elk richting heeft maximaal twee manieren om er te kunnen komen. De client kiest hierbij de collectie die beschikbaar is.

### AI Controller

Wanneer een auto op de weg rijd moet er niet alleen bijgehouden worden waar de auto heen moet rijden. Door middel van raycasts in Unity3d “kijkt” de auto voor zich. Wanneer een ander voertuig in de weg zit zal de auto afremmen. En de auto zal rechts van zich blijven kijken. Als er twee auto’s willen invoegen op dezelfde weg zal de rechter auto afremmen en de linker auto voorrang geven. Hiermee proberen we ophoping tegen te gaan. Naar het registreren van andere voertuigen moet de auto ook de stoplichten in de gaten houden. Als de auto een stoplicht collider voor zich ziet zal het op dezelfde manier als de waypoint gaan checken. Wanneer de client een stoplicht command binnen krijgt zal niet alleen het stoplicht model veranderen. Ook de stoplicht colliders die door de auto’s worden gecheckt krijgen een seintje. De auto ziet deze verandering en beslist dan om wel of niet door te rijden.

# Conclusie

De opdracht ging erg goed. We hebben regelmatig met elkaar afgesproken. Doordat we vaak hebben getest werken de clients en servers goed met elkaar samen. Dit is ook te danken aan de duidelijke regels die we aan het begin hebben opgesteld. De communicatie was relatief snel ingebouwd. Daarna was voor ons de pathfinding het moeilijkst. Omdat we dit systeem zelf hebben moeten maken zat er erg veel tijd in. Uiteindelijk hebben we een goed werkende server en client gemaakt, die alle verkeersregels handhaafd.