Technisch Rapport

**Project D groep 2**

**INFPRJ01D**

Mohamad Almarawi (0978573)

Parwesh Bhaggan (1005210)

Darshan Alakhramsing (0985709)

Moesab Ajdid (0987916)

Bas Weidmann (1007701)

29-03-2022

Overzicht

De gemeente Rotterdam heeft als opdracht gegeven om in een virtuele omgeving RET-voertuigen te zien bewegen. Deze voertuigen moeten bewegen op basis van de real time RET-dienstregeling in Beurs. Hiermee krijgt de gebruiker een beter beeld van het openbaar vervoer en de omgeving waarin het zich bevindt.

We zijn begonnen met het onderzoeken naar de technische aspecten waar onze ontwikkelomgeving aan moet voldoen. Het belangrijkste aspect waar onze ontwikkelomgeving aan moet voldoen is dat het importeren van SKP-modellen zal ondersteunen. Deze SKP-modellen staan online openbaar beschikbaar op 3DRotterdam.nl. Vervolgens zijn wij gaan kijken naar welke ontwikkelomgeving ons de opties biedt om figuren te laten bewegen. Hier hebben wij naar gekeken, omdat wij dan niet onze eigen animaties moeten maken voor de voertuigen. Door de verschillende ontwikkelingsomgevingen met elkaar te vergelijken en afwegingen te maken, kunnen wij verder gaan kijken naar hoe wij de realtime ov-dienstregeling gaan implementeren. Wanneer wij een keuze hebben gemaakt hoe wij dit gaan doen, zal er als laatst gekeken worden naar hoe wij de realtime data gaan ophalen en gebruiken in de applicatie.

Dit technisch rapport is opgesteld met als doel de lezer te informeren over de gemaakte keuzes en oplossingen. Door dit rapport doorgenomen te hebben zal de lezer een goed beeld hebben van alle technische aspecten en zou op basis hiervan zelf verder kunnen. De technische onderwerpen die behandeld worden in dit document zijn:

* API's
* Version control
* Libraries
* Tests
* Referenties

Abstract

Het doel van dit technisch rapport is om de technische aspecten en implementatie van de applicatie te documenteren. Hoofdstuk 1 geeft een overzicht van de technologieën die voor dit project zijn gebruikt. Het zal de technologieën beschrijven en onze keuzes ondersteunen. Hoofdstuk 2 legt uit welke methoden en tools voor de toepassing wordt gebruikt en ondersteunen deze keuzes. Hoofdstuk 3 gaat over de technische aspecten die er zijn gebruikt met uitleg waarom wij voor deze technieken hebben gekozen. Als laatst volgt er een korte conclusie. Hoofdstuk 4 zal gaan over eventuele tips die wij kunnen meegeven als er verder gewerkt wordt aan het project. Hoofdstuk 5 zal een conclusie zijn van onze onderzoek en implementatie van de technieken die wij gebruikt hebben.

Inhoudsopgave

[Overzicht 2](#_Toc1884279472)

[Introductie 5](#_Toc102049381)

[Probleem 5](#_Toc1411610918)

[Doel 5](#_Toc280450571)

[Uitdagingen 5](#_Toc174384225)

[Scope 5](#_Toc1057129250)

[1 Technologieën 6](#_Toc550050938)

[1.1 Programmeertaal 6](#_Toc676398244)

[1.2 Libraries 6](#_Toc1471717065)

[1.3 Hulpmiddelen 6](#_Toc72608244)

[2 Methodologie 6](#_Toc1469632753)

[3 Onderzoek 7](#_Toc695841510)

[3.1 Onderzoeksrapport 7](#_Toc1782143733)

[3.2 Technische aspecten 7](#_Toc1157932768)

[3.2.1 Unity 7](#_Toc770786873)

[3.2.2 KV78Turbo-OVAPI 7](#_Toc2061718766)

[3.2.3 AI Navigation 7](#_Toc122470042)

[3.2.4 GitHub 7](#_Toc353821406)

[3.2.5 Besturingssysteem 7](#_Toc408580881)

[3.2.6 Colliders 7](#_Toc839927645)

[3.2.7 Checkpoint en finishpoint 7](#_Toc1531961090)

[3.2.8 Tags 7](#_Toc1872764343)

[3.2.9 UI Elements 7](#_Toc223438685)

[3.2.10 Scripts 7](#_Toc1224516791)

[3.2.11 Tijd Systeem 7](#_Toc1179528539)

[3.2.12 Camera Switch 7](#_Toc1808186801)

[3.2.13 Light Change 7](#_Toc602759402)

[3.3 Opzet project 7](#_Toc1357794071)

[3.4 Technische problemen 7](#_Toc2080645830)

[4 Aanbevelingen 7](#_Toc1038296814)

[5 Conclusie 7](#_Toc1620106751)

[Referenties 7](#_Toc314889332)

Begrippenlijst

|  |  |
| --- | --- |
| Begrip | Betekenis |
| OV | Openbaar vervoer |
| API | Application Programming Interface |
| SKP Model | 3D figuur in SKP-formaat. |
| RET | Rotterdam Elektronische Tram |
| JSON | Javascript Object Notation |
| AI | Kunstmatige intelligentie |
| OS | Besturingssysteem (Windows, Linux, MacOS) |
| Version control | De manier waarop het bij houden van veranderingen in de code makkelijk bij te houden is. |
| LTS | Long Term Support |
| OOP | Object Oriented Programming |
| PO | Product Owner (Product Eigenaar) |
| Stakeholders | Iedereen betrokken bij het product |
| Prefab | Aangemaakt object dat als standaard gebruikt kan worden voor het aanmaken van meerdere |
| Empty | Object in unity met als enige standaardwaarden een locatie. |

Introductie

Probleem

Op het moment heeft Gemeente Rotterdam veel data waar nog weinig mee gedaan wordt. Daarom zijn ze op zoek naar innovatie binnen de stad. Hiervoor zijn ze opgekomen met het Digital Twin project. Dit project houdt in dat de stad Rotterdam een digitale kopie krijgt van zijn omgeving. Deze digitale 3D omgeving is al gemaakt en te vinden op https://www.3drotterdam.nl/ (Gemeente Rotterdam, z.d.).

In samenwerking met Gemeente Rotterdam, werken onderzoek- en opleidingsinstituten eraan om de 3D omgeving verder uit te breiden naar andere projecten.

Doel

Om de Digital Twin verder uit te breiden hebben wij de taak gekregen om met realtime OV-data aan de slag te gaan. Dit houdt in dat wij de huidige locatie van de OV-voertuigen met daarbij de statische locaties van de stations moeten weergeven in een 3D omgeving. Deze innovatie zal ervoor zorgen dat mensen een makkelijker beeld kunnen krijgen van de omgeving. Ook geeft het inzicht over waar en hoe laat ze zich moeten bevinden mochten ze een tram of bus willen nemen.

Uitdagingen

* Figuren laten bewegen op realtime positie
* Rekening houden met vertragingen
* Aangeven of trams en bussen eerder aankomen
* Figuren op hun eigen sporen/wegen laten rijden

Scope

De focus zal voornamelijk gelegd worden op het werkend krijgen van de realtime OV-dienstregeling rond Beurs. De OV-voertuigen die geïmplementeerd worden zullen voornamelijk trams en bussen zijn. Er is niet afgesproken dat er ook metro’s in de applicatie komen. Verder zullen de modellen ook heel minimalistisch zijn en niet vele andere objecten naast stations, gebouwen en OV-voertuigen. Er zijn vele verschillende manier hoe wij deze realtime data kunnen ophalen, maar in dit project zal er gebruik gemaakt worden van JSON.

1 Technologieën

1.1 Programmeertaal

**C#:** C# werd ongeveer 20 jaar geleden uitgebracht. Het behoort tot de C familie. Programmeurs die met C en C++ ervaring hebben zullen geen problemen hebben om C# op te pakken. Bijna elke game engine maakt gebruik van één van de C talen en is daar ook op gebouwd. Applicaties ontwikkelen met C# zijn ook heel erg beginner vriendelijk. Applicaties die gemaakt zijn met C# kan je ook op andere platformen uitbrengen naast alleen de gebruikelijke pc met Windows, Mac of Linux. Code schrijven is ook een stuk fijner sinds het gebruik maakt van OOP. Door middel van OOP is het maken van objecten en de structuur van het project ook een stuk overzichtelijker (BairesDev, z.d.).

1.2 Libraries

**Newtonsoft Json.Net:** De Newtonsoft Json.Net library geeft ons de mogelijkheden om JSON-objecten om te zetten naar bruikbare .Net objecten. Met deze library kan ervoor gezorgd worden dat er tijd bespaard wordt om zelf de JSON-objecten om te zetten. Er is geprobeerd om andere JSON libraries te gebruiken, maar die werkte niet zoals verwacht. Er is gebleken dat er sprake was van compatibiliteit en limitatie problemen (Newtonsoft, z.d.).

**System.Net:** Biedt een simpele programmeerinterface voor veel van de protocollen die gebruikt worden om met vele netwerken te communiceren. De klassen WebRequest en WebResponse vormen de basis van zogenaamde protocollen. Met deze protocollen kan er met gemak data opgevraagd worden die niet downloadbaar of lokaal beschikbaar zijn. System.Net maakt het dus mogelijk om data over een netwerk te versturen en ontvangen. (Microsoft, 2022).

1.3 Hulpmiddelen

**Github:** Het project kan veilig worden gewijzigd met Github, omdat het de huidige versie en de originele versie van de code bijhoudt. Het geeft ook meer inzicht op het project, zoals wie wat voor wijzigingen heeft gemaakt en op wat voor datum. Een account aanmaken gaat makkelijk en snel dus iedereen kan gratis een account aanmaken zonder restricties op het project (Github, 2022).

2 Methodologie

**SCRUM**: Voor het project hebben we gebruik gemaakt van de SCRUM methode. Deze manier van werken hebben wij al gebruikt in eerdere projecten. Het voordeel van SCRUM is dat er transparantie en duidelijkheid binnen het team zit. Hiermee weet iedereen ten alle tijden waar iedereen mee bezig is. De momenten dat wij feedback krijgen van de PO en andere stakeholders komen dan ook goed van pas. Door al deze eigenschappen kan er een product tot stand komen waar de PO blij mee zal zijn (Scrum, z.d.).

**Microsoft Teams**: Voor vergaderingen met de PO en stakeholders hebben wij Microsoft Teams gebruikt. Een voordeel hiervan is dat niet iedereen per se op locatie hoeft te zijn als ze het niet kunnen redden qua tijd. Dit helpt ook met het vastleggen van afspraken en verwachtingen (Microsoft, 2022).

**Trello:** Met Trello kan het team makkelijk een overzicht hebben waar iedereen kan zien: wat er gebeurd is, wat er nog moet gebeuren en waar iemand nog eventueel vastloopt. Het is makkelijk om mensen in het Trello bord uit te nodigen. Je kan tags, labels en commentaar plaatsen als deze nodig zijn om een overzichtelijk bord te creëren (Atlassian, z.d.).

**Discord:** Als groep hebben wij Discord gebruikt voor vergaderingen binnen de groep zelf. Met Discord hebben wij een project server aangemaakt waarin wij kanalen kunnen maken om verschillende onderwerpen te delen zoals: code, linkjes, vragen en feedback hiermee is alles netjes georganiseerd en kan iedereen alles makkelijk terugvinden (Hornshaw, 2022).

**Contact met PO en stakeholders:** Gedurende het project hebben we een paar keer contact gehad om vergaderingen en interviews in te plannen met de stakeholders en de PO. Dit hebben we voornamelijk gedaan via Office 365 Outlook.

**Sharepoint:** Office 365 geeft de mogelijkheid om bestanden online met elkaar te delen en op te slaan. Hier hebben wij gebruik van gemaakt om alle documentatie op te slaan (Microsoft, 2022).

3 Onderzoek

3.1 Onderzoeksrapport

Wij hebben een onderzoeksrapport gemaakt waarin staat waarom wij voor bepaalde technieken en tools hebben gekozen. In dit rapport staat waarom wij voor bepaalde technieken en tools hebben gekozen. Voor het volledige rapport zie [[Onderzoeksrapport](Onderzoeks%20Rapport.docx) ].

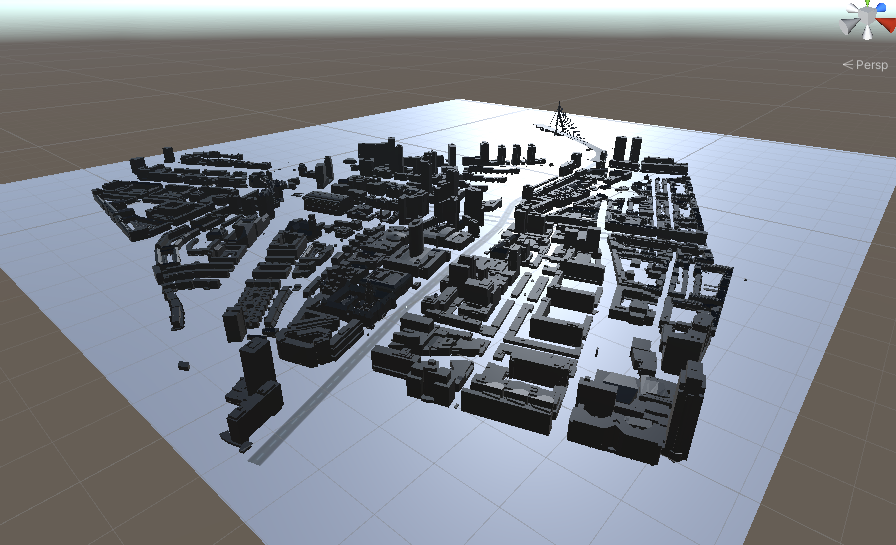
3.2 Technische aspecten

|  |  |  |
| --- | --- | --- |
| Tool | Versie | Gebruik |
| Unity | 2020.3.X | Ontwikkelomgeving |
| KV78Turbo-OVAPI | 28 Apr 2020 | RET ov-dienstregeling ophalen |
| NavMesh | 2021.3 | Bewegen voor objecten in Unity |
| GitHub | n.v.t. | Version Control |
| Besturingssysteem | Windows 10 en Windows 11 | Besturingssysteem |
| Json.Net | 13.0.1 | Converteren van JSON types naar .Net objecten |

3.2.1 Unity

Voor het gebruiken van Unity kan Unity hub gedownload worden via <https://unity3d.com/get-unity/download>. In Unity hub kan je de verschillende Unity versies downloaden.

De versie waarmee we het project maken is Unity 2020.3.X (LTS). Voor het tonen van de verschillende Rotterdam locaties zullen wij gebruik gaan maken van SKP-modellen die gedownload kunnen worden via <https://www.3drotterdam.nl/.> Deze SKP-modellen zijn rechtstreeks te importeren naar Unity zonder dat daar enige omzetting voor nodig zijn.



Afb 1. Overzicht Stadsdriehoek en cool skp modellen

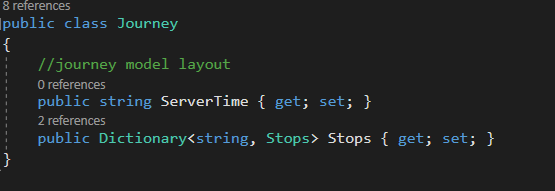
3.2.2 KV78Turbo-OVAPI

Voor de implementatie van de OV API binnen ons project hebben we een RET-data script aangemaakt.

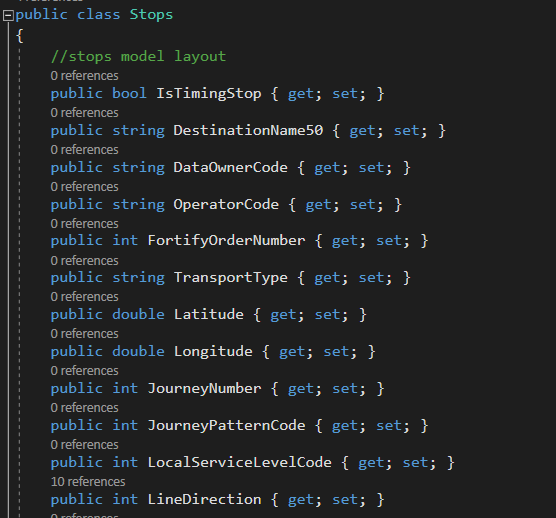
De RET-data script bevat:

* Stops model
* Journey model
* API webrequest
* Tram objecten en attributen

In afbeelding 1 en afbeelding 2 zijn 2 modellen te zien: Stops en Journey modellen.  
De *“Stops”* model bevat de eigenschappen van een station die opgeslagen is in de API in dit formaat.  
De *“Journey”* model bevat de eigenschappen van een route die opgeslagen is in de API in dit formaat.



Afb 2. Journey structuur

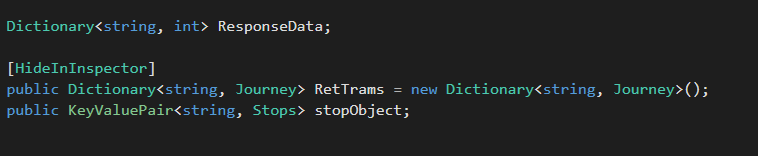
  
Afb 3. Stops structuur

In afbeelding 4 zijn er variables geplaats die gebruikt worden voor het ophalen van de OV -data uit de API.

*“ResponseData”* is een dictionary dat gebruikt wordt om een overzicht van al het openbaar vervoer in kaart te brengen. Dit maakt het mogelijk de specifieke voertuigen na het filteren op te vragen.

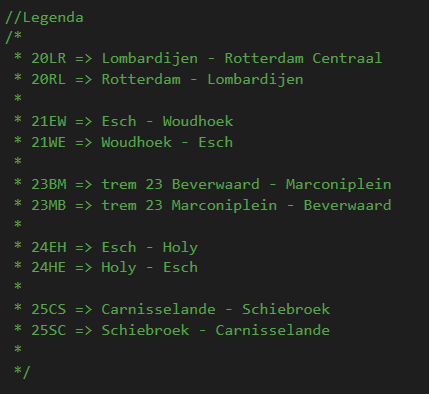
*“RET-trams"* is een dictionary dat gebruikt wordt voor het opslaan van RET-tram objecten uit de API.

*“StopObject”* is een variabel met KeyValuePair data type waarin de stations opgeslagen worden uit de API.



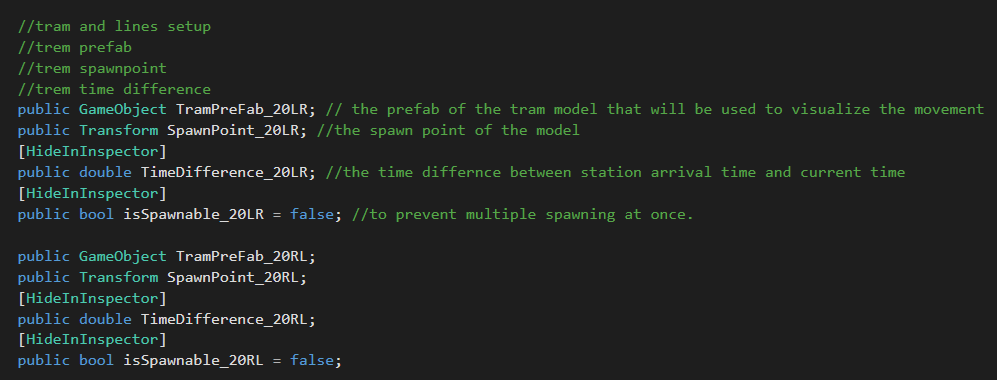
Afb 4. Overzicht van ResponseData, RetTrams en stopObject

In dit script hebben we een schrijf formaat gebruikt voor de tram objecten. In afbeelding 5 wordt er uitgelegd wat de verschillende codes betekenen van de tram objecten



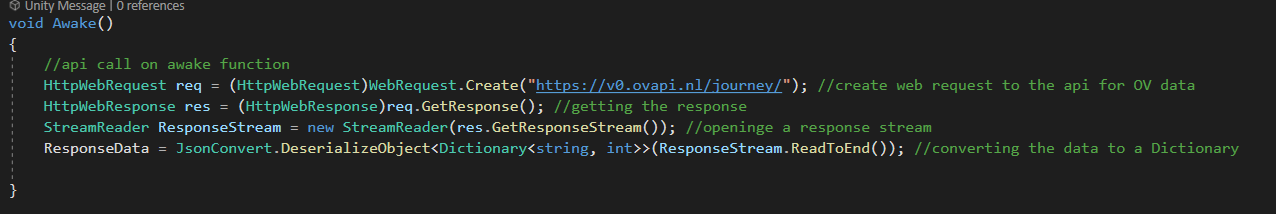
Afb 5. Legenda tram codes

Afbeelding 6 geeft aan de eigenschappen die er nodig zijn voor een tram object voor dit project.  
*“TramPreFab\_20LR”*: Is het model van tram 20 Lombardijen – Rotterdam Centraal.  
*“SpawnPoint\_20LR”*: Is het start locatie van de tram wanneer hij ingeladen wordt in de scene.  
*“TimeDifference\_20LR”:* Is de variabel waarin het tijdverschil tussen huidige en aankomsttijd wordt opgeslagen.  
*“IsSpawnable\_20LR”*: Geeft aan of het mogelijk is voor de tram om in geladen te worden. Deze variabel is gemaakt zodat er niet heel veel objecten van de tram in geladen wordt in 1 keer.



Afb 6. Tram object variables

In afbeelding 7 is er in de *“Awake()”* functie van Unity de API-call gedaan. Een Awake() functie wordt automatisch geroepen door Unity bij het in laden van de scene.  
*“HttpWebReqeust req”*: is de variabel waarin een WebRequest gecreëerd wordt voor het connecten naar de API.  
*“HttpWebResponse res”*: is de variabel waarin het antwoord terug op de aanvraag van data opgeslagen wordt.  
*“StreamReader ResponseStream”*: is de variabel waar een stream geopend wordt zodat de data leesbaar is.  
*“ResponseData”*: is een variable met dictionary data type. In deze variable wordt de data geconverteerd naar dictionary data type en opgeslagen.

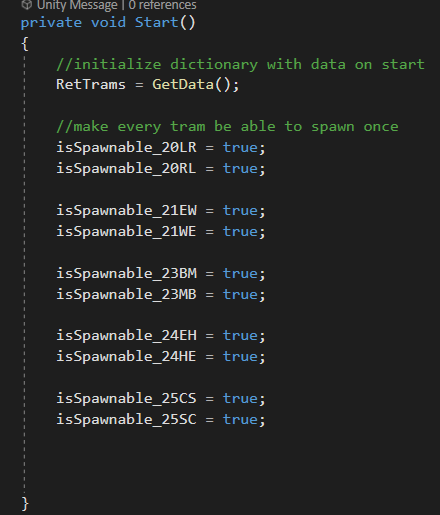


Afb 7. Api Awake() call

Afbeelding 8 geeft de *“GetData”* functie aan. Deze functie zorgt ervoor dat de data gefilterd worden op RET-trams en bussen. Vervolgens wordt het gefilterd op basis van specifieke tram lijnen die voorkomen op specifieke tramhaltes.  
Tenslotte wordt deze data opgeslagen in de variabel *“var tramData”* wat de functie als output terugstuurt.  


Afb 8. GetData functie

In afbeelding 9 is de *“Start()”* functie te zien. De *“Start()”* functie wordt automatisch opgeroepen in Unity tijdens het laden van de scene. Deze functie wordt na de *“Awake()”* functie geroepen.  
*“RetTrams = GetData()”* slaat alle RetTrams die gefilterd zijn op.  
Vervolgens worden alle trams geplaats op *“isSpawnable = true”*. Hierdoor zijn de tram objecten in staat in te laden op het juiste tijdstip.

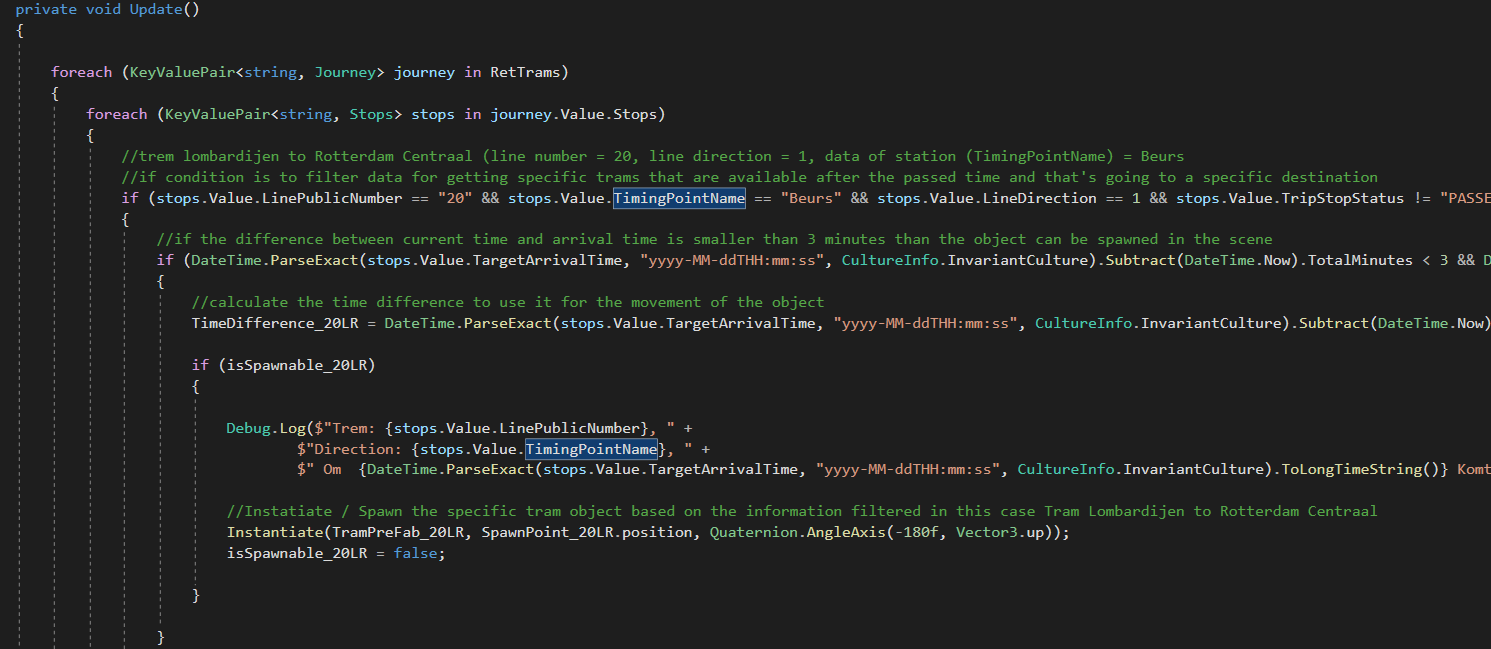


Afb 9. Api Start functie

In afbeelding 10 is de *“Update()”* functie te zien. De *“Update()”* functie wordt geroepen in elke frame zodra de Unity scene is ingeladen. In deze *“Update()”* functie wordt er eerst een loop gemaakt door de RetTrams object heen. Daarna wordt er gekeken of er een specifieke tram voorkomt die gefilterd wordt op:

* LinePublicNumber: de lijn nummer
* TimingPointName: het station waar de tram moet aankomen (Beurs station)
* LineDirection: de richting van de tram bijv. Lombardijen – Rotterdam Centraal
* TripStopStatus: ervoor zorgen dat de tram Beurs nog niet is gepasseerd
* Het verschil van de aankomsttijd en huidige tijd moet groter zijn dan 0 seconden

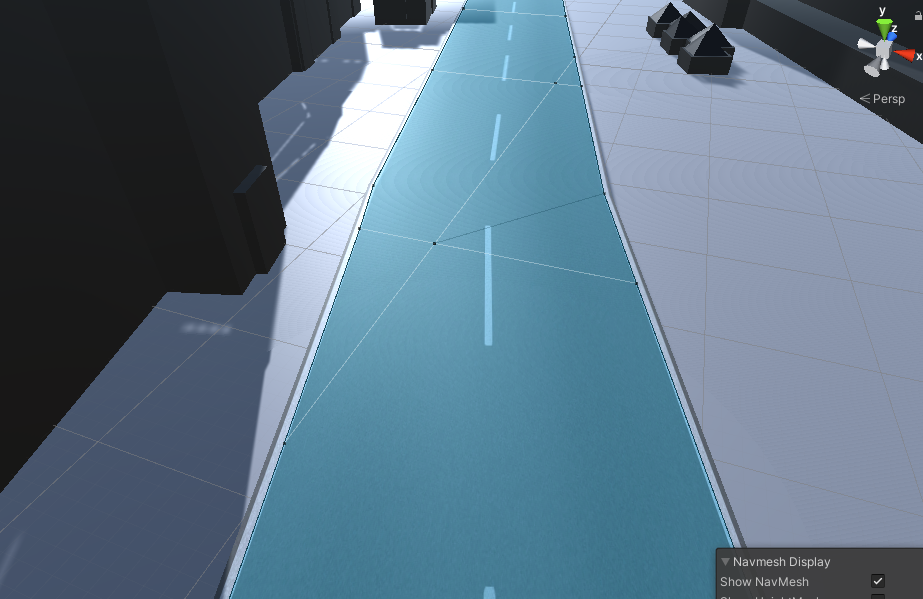
Daarna wordt er gekeken naar het verschil van aankomsttijd en huidige tijd als dat kleiner is dan 3 minuten, wordt dat verschil opgeslagen in de TimeDifference variabel en door de Instantiate functie wordt de tram object in de scene geladen.



Afb 10. Api Update functie

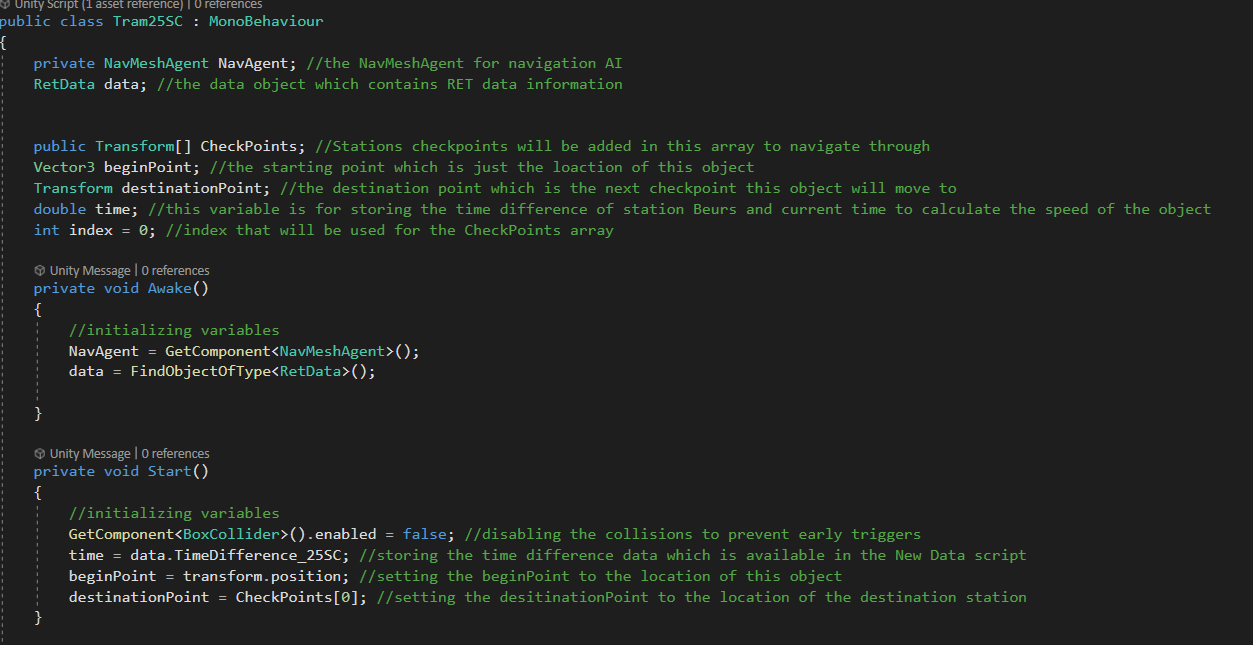
3.2.3 AI Navigation

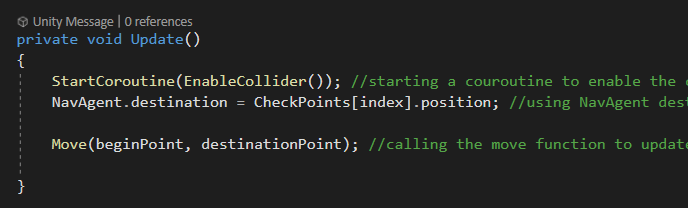
Unity heeft een AI-feature genoemd Navigation. Hierdoor kunnen objecten zelf bewegen op platformen waarop er een NavMesh geplaatst wordt. Deze NavMesh zorgt ervoor dat het platform gezien wordt als een "loopbare" platform. Op de objecten die kunnen bewegen wordt een NavMeshAgent component geplaatst. Dit component zorgt ervoor dat de objecten kunnen bewegen op het loopbare platform en met een aangegeven snelheid. Met behulp van deze AI kan de NavMeshAgent een NavMesh detecteren en uit zichzelf navigeren. Aangezien dit een ingebouwde functie is in Unity hoeft hier niks extra’s voor gedownload te worden.



Afb 11. Voorbeeld navmesh

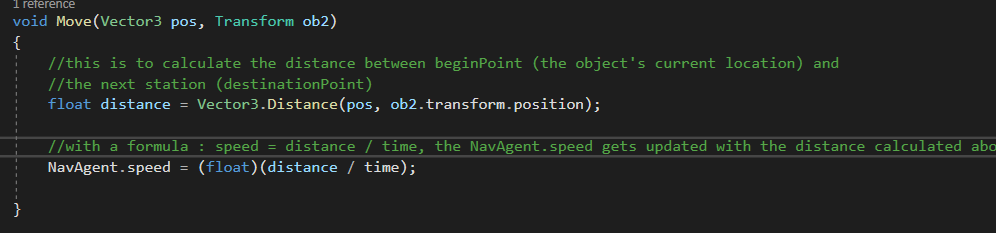
In ons project hebben alle tram objecten een NavMeshAgent component. Om gebruik te maken van de NavMeshAgent op de tram objecten zijn er enkele scripts geschreven voor de mechanica en logica van de trams.   
Elk tram script is bijna identiek aan elkaar met kleine verschillen. De verschillen liggen in de time variabel en *“isSpawnable”* variabel.  
NavMeshAgent NavAgent: Is een variabel die nodig is voor de Navigation AI van Unity. Hiermee kan de tram object op een object met een NavMesh component bewegen.  
*“RetData data”*: is de data object waarin de tijdverschillen en isSpawnable eigenschappen voorkomen die nodig zijn voor de tram objecten.  
*“Transform[] CheckPoints”*: Hierin worden de stations checkpoints en finish checkpoints opgeslagen die nodig zijn voor het navigeren door checkpoints heen.  
*“Vector3 beginPoint”*: deze variabel wordt gebruikt om de begin punt van de tram locatie op te slaan.  
Transform destinationPoint: deze variabel wordt gebruikt om het eindpunt van de station en finish checkpoints geplaatst.  
*“Time”*: in deze variabel wordt het tijdverschil opgeslagen.  
*“Index”*: deze variabel dient voor de index van de checkpoints array.

  
Afb 12. Tram opstart script

In afbeelding 13 is de Updatefunctie te zien. In deze functie wordt er eerst de colliders aangeschakeld die uitgeschakeld worden in de start functie. Dit zodat de triggers niet geactiveerd worden zodra het object in geladen is.  
NavAgent.destination eigenschap wordt update naar de checkpoint locatie.  
Om de tram in beweging te krijgen wordt de *“Move()”* functie geroepen.  


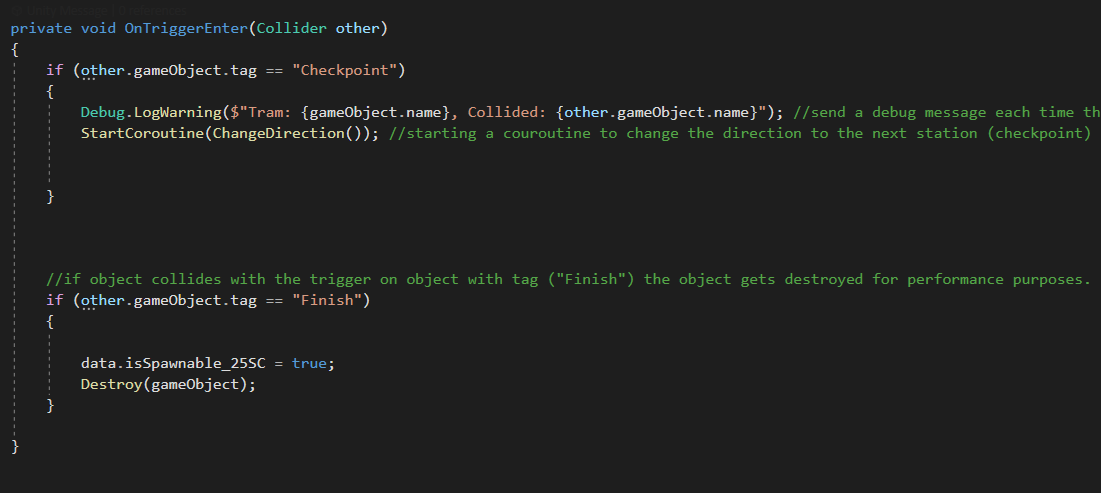
Afb 13. Tram update script voor beweging

In de *“Move()”* functie wordt er eerst de afstand berekend tussen Vector3 pos en Transform ob2. In deze parameters worden op Vector3 pos de beginPoint geplaatst en op Transform ob2 wordt de destinationPoint geplaatst. Hierdoor kan de afstand tussen de 2 punten berekend worden en samen met het tijdverschil in time variabel kan de snelheid van de tram berekend worden.  
De formule voor snelheid is:   
Snelheid = afstand/ tijd => NavAgent.speed = distance / time.



Afb 14. Tram beweging script berekening

In afbeeldingen 15 en 16 zijn de *“OnTriggerEnter()”* en *“ChangeDirection()”* functies. Het moment dat de tram object tegen een checkpoint object bots wordt de OnTriggerEnter functie opgeroepen. In deze functie wordt de *“ChangeDirection()”* functie opgeroepen die de destinatie van de tram update. Als de tram tegen de finish point bots dan wordt in de “OnTriggerEnter()” functie de *“Destroy()”* functie opgeroepen die ervoor zorgt dat de tram object verwijderd wordt uit de scene.



Afb 15. Checkpoint herkenning script

Afb 16. Route wijziging script

3.2.4 GitHub

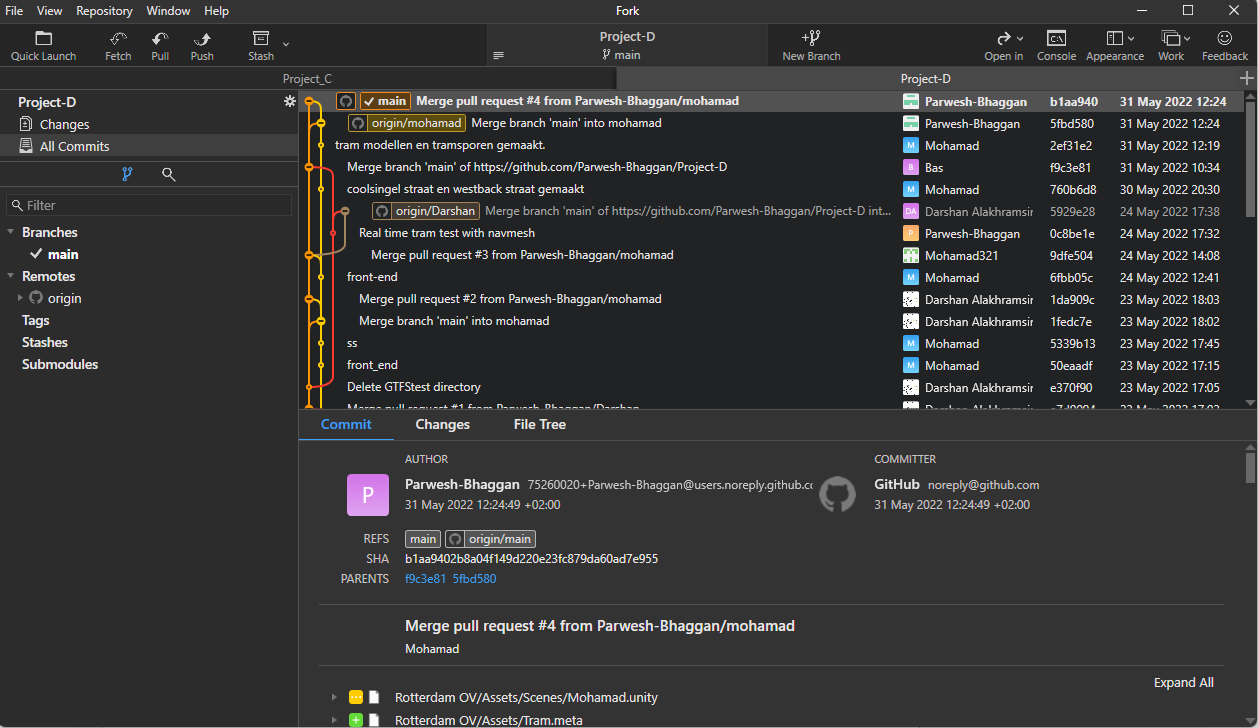
Voor het pushen en pullen naar een remote branch maken wij gebruik van twee soorten GIT clients genaamd Fork en GitHub desktop. Om van deze clients gebruik te maken hoef je alleen in te loggen met je GitHub gegevens en dan kan je de link van het project toevoegen. Als de link naar het project in deze clients toegevoegd is dan kan je beginnen lokale branches te maken om vervolgens code te toe te voegen. Links boven bij "File" krijg je een drop down menu met als optie "clone". Als je daarop hebt geklikt krijg je het volgende te zien.

Fork:



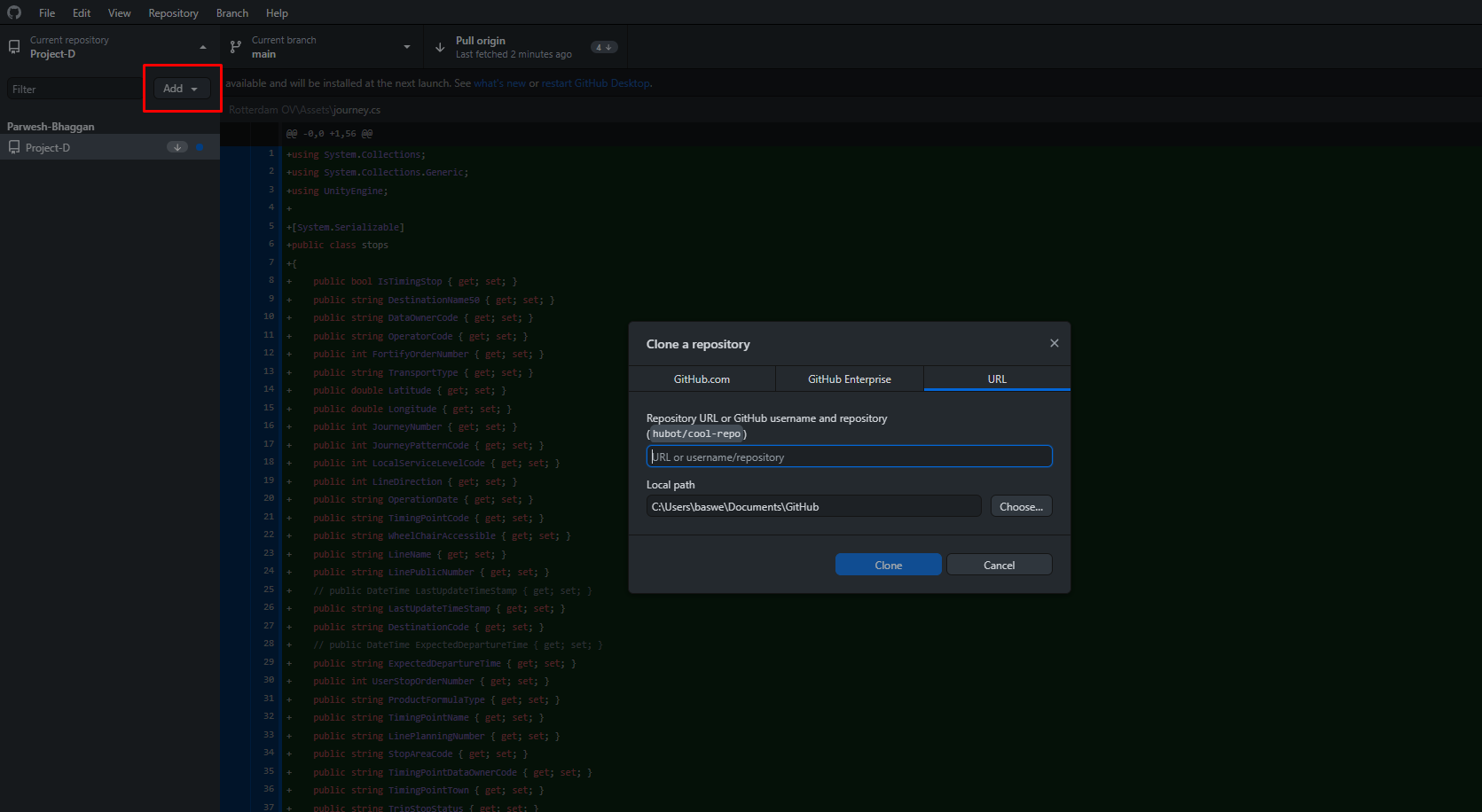
Afb 17. Github Fork

Als je de url naar de repositiry op GitHub hebt dan kan je dat bij "Repository url" plakken en vervolgens clonen.



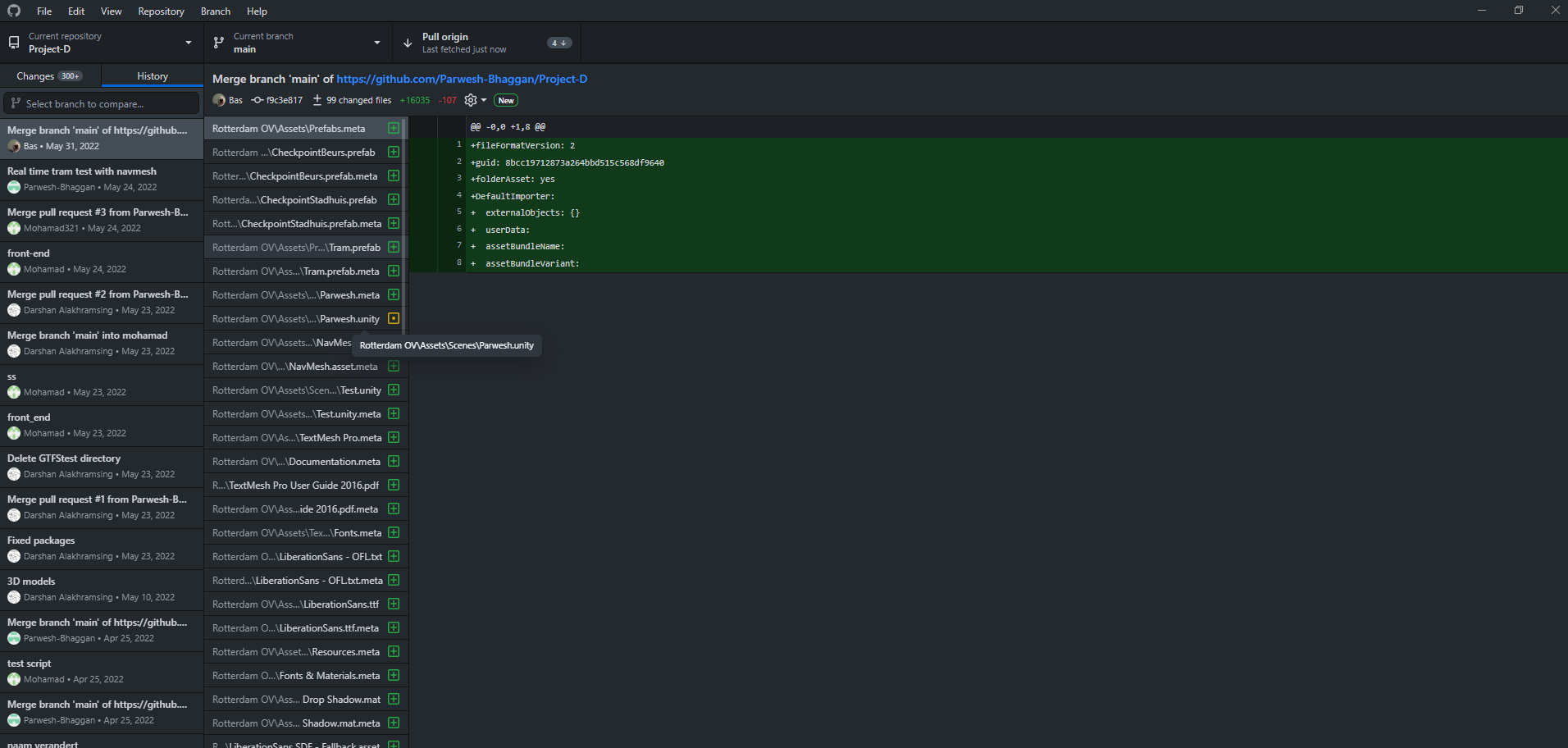
Afb 18. Github Fork clone

GitHub Desktop:



Afb 19. Github Desktop

Door linksboven op Add te drukken kan je de url van de github toevoegen om het project te clonen



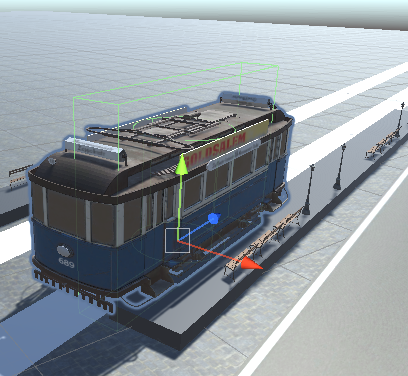
Afb 20. Github desktop clone

3.2.5 Besturingssysteem

De versie van Unity die wij gebruiken is 2020.3.X (LTS). Deze wordt ondersteunt op Windows 10 en 11. De meeste besturingssystemen gebruiken tegenwoordig een van deze Windows varianten en wordt het nog voor de komende jaren door Microsoft ondersteunt.

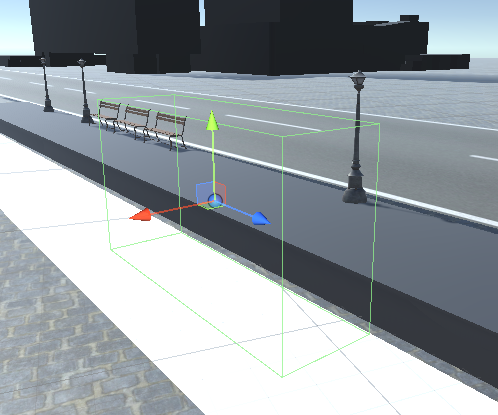
3.2.6 Colliders

In het project hebben we gebruik gemaakt van colliders. Alle 3d modellen bezitten een collider. De tram object collider zorgt ervoor dat de botsing tegen een checkpoint of een finish point collider gedetecteerd kan worden. Het verschil tussen de tram collider en de checkpoint collider is dat de checkpoint collider als een trigger reageert. Hierdoor wordt de *“OnTriggerEnter()”* functie opgeroepen om enkele eigenschappen van de tram te updaten. De twee eigenschappen die geüpdatet worden zijn het volgende checkpoint waar de tram naar toe moet navigeren en de snelheid die vervolgens berekend wordt.



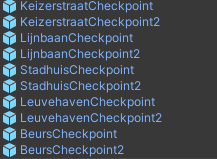
Afb 21. Tram collider

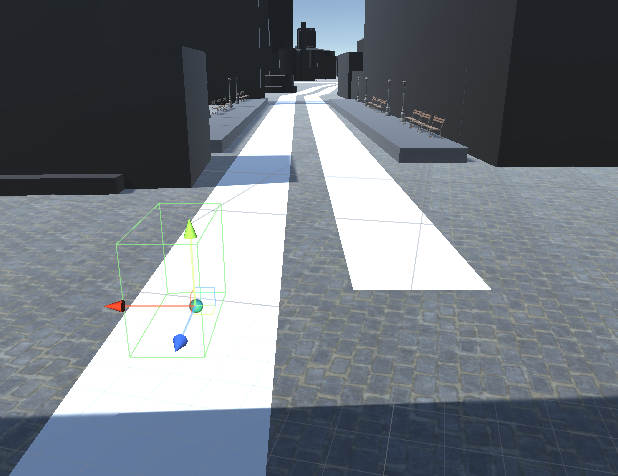
3.2.7 Checkpoint en finishpoint



Afb 22. Checkpoint collider

In de scenes is er bij elk station een checkpoint geplaatst. Elk checkpoint is gecreëerd door een empty game object aan te maken. Aan elk checkpoint is er een box collider geplaatst en is de *“isTrigger”* knop ingeschakeld. *“IsTrigger”* zorgt ervoor dat de objecten dwars door een checkpoint heen kunnen gaan en kunnen de objecten gedetecteerd worden door middel van botsing. Hierdoor wordt er geen natuurkundige acties plaatsgevonden bijvoorbeeld: Als het object met dit checkpoint bots dan gaat het checkpoint niet van positie veranderen maar zal het object dwars door het checkpoint bewegen. In afbeelding 23 zijn de checkpoints te zien die aanwezig zijn in de scene.

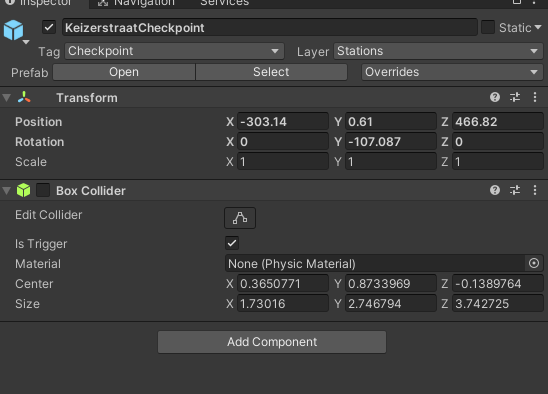


Afb 23. Overzicht station en perron checkpoints  
  
Het doel van deze checkpoints is dat de tram kan navigeren van checkpoint naar checkpoint, dus van station naar station. De trigger collider is geprogrammeerd op een manier dat wanneer de tram het checkpoint heeft bereikt wordt de *”OnTriggerEnter()”* functie ingeschakeld. Hierdoor verandert de locatie naar waar de tram moet navigeren.Afb 24. Finishpoint overzicht Afb 25. Finishpoint collider

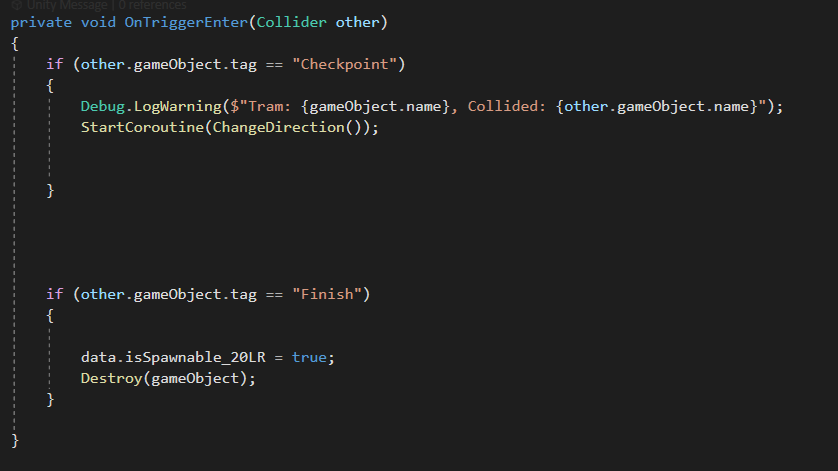
De finish points werken met dezelfde principes als van een normaal checkpoint. Deze zijn geplaatst aan het einde van een tram route. Elk finish point beschik ook van een box collider met de *”isTrigger”* ingeschakeld. Een ander eigenschap van een finish point is dat deze met behulp van de *”OnTriggerEnter()” functie* de tram weghaalt van de scene. Deze tram is niet meer nodig en helpt het weghalen met performance.

3.2.8 Tags

Een tag is een ander eigenschap die een object kan hebben in Unity. Om verschillende soorten objecten te herkennen worden er tags geplaatst. Hierdoor kunnen objecten onderling makkelijker interacteren door dat ze met de tags elkaar kunnen herkennen. In afbeelding 26 is er een voorbeeld van een tag die geplaatst is op een checkpoint.



Afb 26. Checkpoint in unity inspector met checkpoint tag

Elk checkpoint hebben een tag “Checkpoint” en elke finish points hebben een tag “Finish”. Hierdoor kan er via colliders gedetecteerd worden welke object tegen wat is gaan botsen. Een voorbeeld van de code is te zien in de afbeelding 27.  


Afb 27. Checkpoint en finishpoint scripts

Deze script is geplaatst aan een tram object. Elke keer als een trem object met een checkpoint met tag “Checkpoint” bots wordt er enkele informatie gestuurd en een functie opgeroepen. De informatie dat gestuurd wordt naar de applicatie is:

Debug.LogWarning($"Tram: {gameObject.name}, Collided: {other.gameObject.name}");

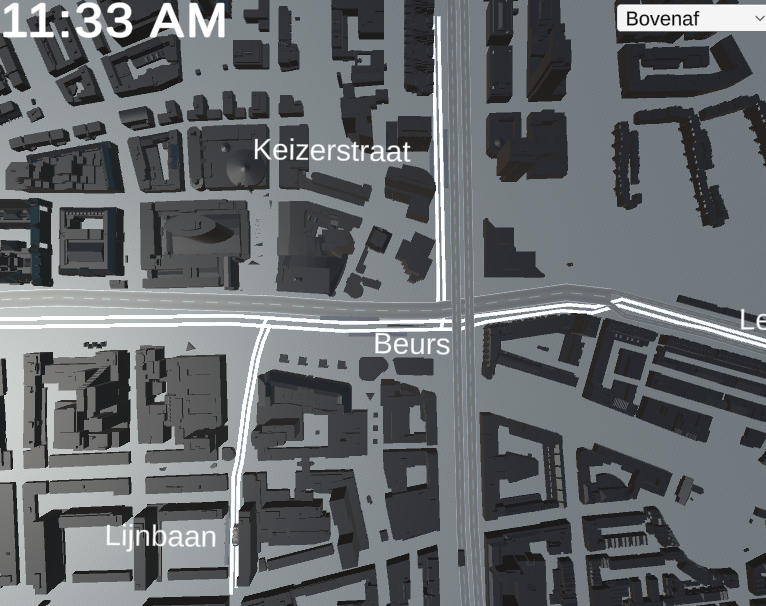
Dit stukje code stuurt informatie van welke tram is gebotst tegen welk checkpoint.

De functie die geroepen wordt heet *“StartCourotine(ChangeDirection())”*. Deze functie zorgt ervoor dat de trem 5 seconden wacht en vervolgens verandert het de locatie waar de trem naartoe zal bewegen.

Verder is er een andere situatie die kijkt naar wat er moet gebeuren als een tram object botst tegen een object met als tag “Finish”. In dit geval wordt er gesproken over de handelingen als een tram een finish point bereikt. Ten eerste wordt een *“isSpawnable”* attribuut geactiveerd die voorkomt in de main data script (RetData). Hierdoor kan de volgende tram object van dit soort tram voorschijnen. Daarna wordt er een *“Destroy(gameObject)”* functie opgeroepen. Hierdoor wordt de tram object weggehaald uit de scene.

3.2.9 UI Elements

In Unity is het mogelijk om UI (User Interface) elementen te plaatsen. In de afbeelding 28 is te zien dat er enkele tekstelementen zijn geplaats die aangeven welke buurt op welke locatie staat, de tijd en een drop down menu waarmee de gebruiker kan navigeren tussen verschillende camera zichten.



Afb 28. Camera perspectief

Voor het opzetten van UI elementen kan er in Unity UI game objecten gemaakt worden. Het moment dat er een UI object gemaakt wordt komen 2 andere objecten tevoorschijn namelijk:

* Canvas
* Eventsystem

Een canvas is de ruimte die elk UI element bevat. Door middel van zo een canvas kunnen de UI elementen zichtbaar komen in de camera view.  
Een event system beheert acties van UI elementen bijvoorbeeld als er knoppen geplaatst worden kan via de event systeem de acties van het drukken van een knop beheerd worden.

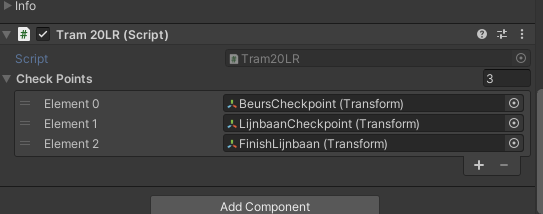
3.2.10 Scripts

In ons project zijn er verschillende scripts gemaakt voor het ophalen van OV-data, beweging van trams, een tijdsysteem, verschillende camera zichten en licht veranderingen. In afbeelding 29 zijn de scripts van ons project te zien.



Afb 29. Scripts in unity overzicht

Als er een script aan een object geplaatst wordt dan ziet het eruit als in afbeelding 30.

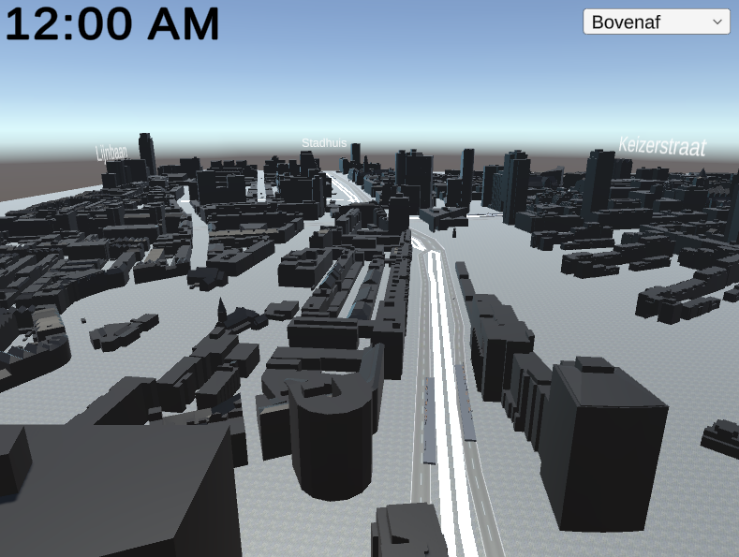


Afb 30. Tramhaltes als checkpoints in de unity inspector

Hieruit kan geconcludeerd worden dat een script als een component werkt en zo zijn ook de rest van de componenten in Unity.

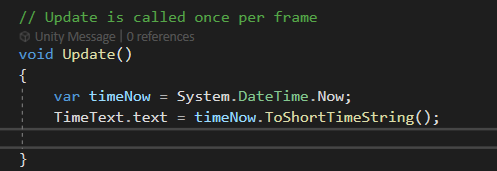
3.2.11 Tijd Systeem

In het project hebben we een klok geprogrammeerd als een UI element.



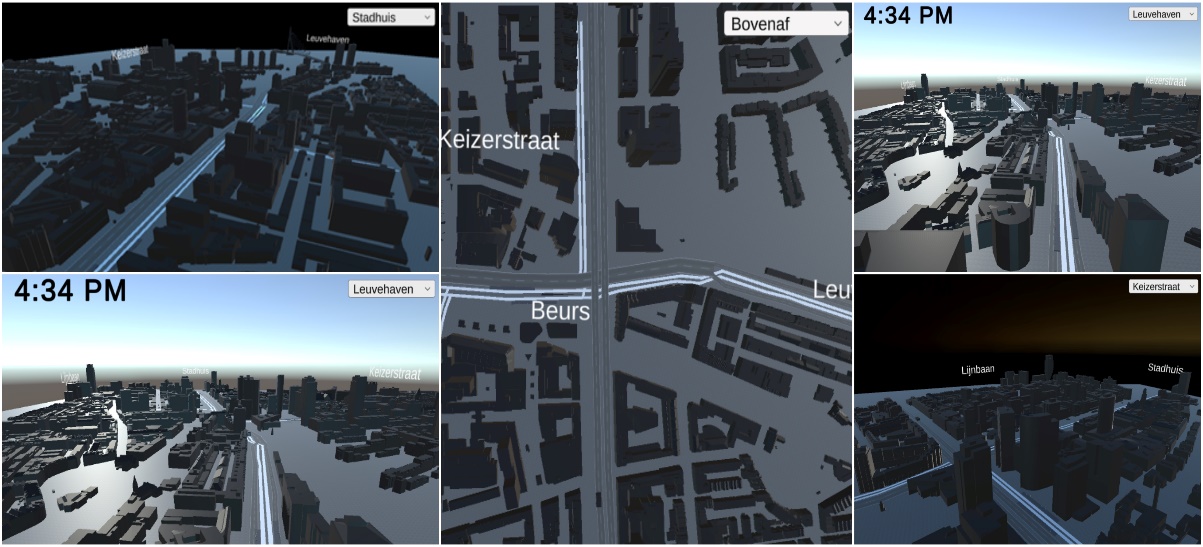
Afb 31. Klok in Unity camera perspectief

In de TimeSystem script wordt de UI tekstelement update naar de huidige tijd.

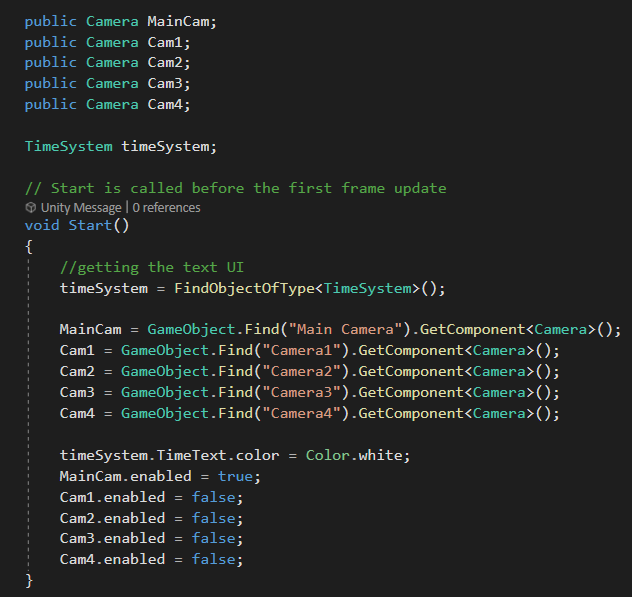


Afb 32. Klok script

3.2.12 Camera Switch



Afb 33. Alle camera overzichten

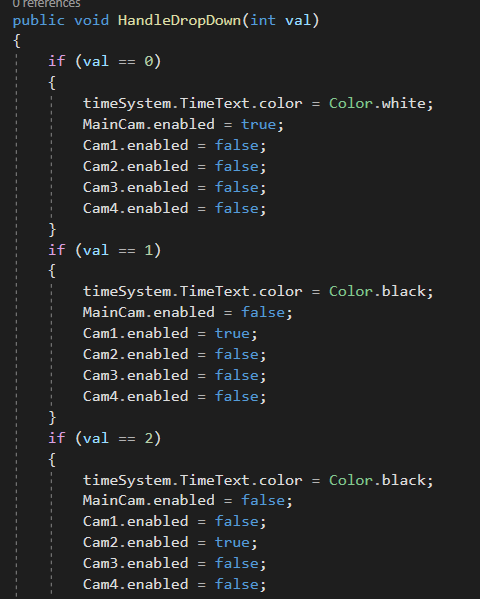
In de CameraSwitch script worden de verschillende camera’s die geplaatst zijn in de scene geïnitialiseerd in de *“Start()”* functie.  


Afb 34. Camera wisseling script

In afbeelding 35 is het mechanisme van de DropDown menu te zien. Door middel van de dropdown menu worden er op specifieke waarden van het menu bepaalde camera’s ingeschakeld en alle ander camera's uitgeschakeld.

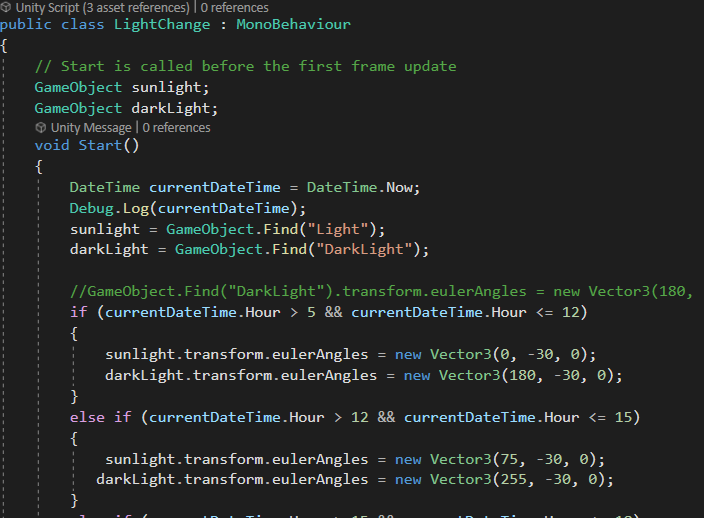


Afb 35. Camera selectie

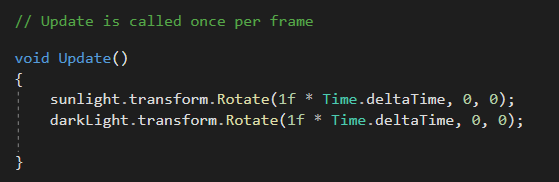


Afb 36. Camera selectie script

3.2.13 Light Change

In de LightChange script worden de licht bronnen geprogrammeerd. In de start functies worden de licht bronnen geïnitialiseerd. Daarna wordt er gekeken hoe laat het is en dan worden de licht bronnen geroteerd naar een specifieke rotatie waarde.  


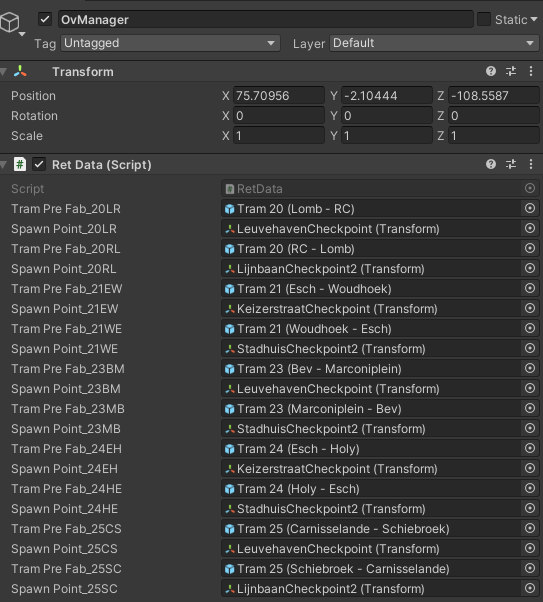
Afb 37. Lightchange script

De updatefunctie zorgt ervoor dat de licht bronnen bewegen in een roterende beweging.  
  


Afb 38. Dag en nacht wisseling script

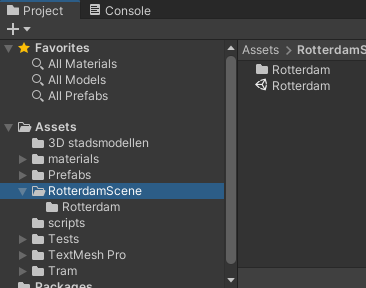
3.2.14 OV-manager

OV Manager is een empty object in Unity waaraan de RET-data script gekoppeld is. Dit object zorgt ervoor dat de Ret Data script gerund wordt voor initialiseren van de data en ophalen van de OV-data uit de API. Vervolgens zijn de prefabs van de Trams en het laad punt van de locaties geplaatst.

  
Afb 39. Ovmanager script in Unity inspector

3.3 Opzet project

1. Zorg voor een Windows 10 of 11 installatie op de pc. (Meerdere stappen mogelijk)
2. Download vervolgens Unity Hub van <https://unity3d.com/get-unity/download>
3. Na het downloaden is het verplicht om een Unity account aan te maken. Dit kan je doen bij <https://id.unity.com/en/conversations/3ec72c18-bc38-4276-9c25-7e7c64bad638006f?view=register>
4. Na het aanmaken van een account en downloaden van Unity Hub start je de Unity Hub client op. In de client zelf moet je inloggen met je Unity account.
5. Zorg ervoor dat je de editor versie 2020.3.X installeert. Dit kan je doen aan de linkerkant van de client bij de sectie “Installs”.
6. Als dat gebeurd is kan je op vele verschillende manieren het project van Github downloaden. Het is aan te raden om een GIT client te gebruiken, zoals Fork of Github Desktop. Het project zelf kan je clonen via de link: <https://github.com/Parwesh-Bhaggan/Project-D>.
7. Wanneer je teruggaat naar de Unity Hub client kan je rechts boven een project openen. Hier kan je het gekopieerde project openen en vervolgens verschijnt het eronder.
8. Als je het project geopend hebt dan aan de linker kan je naar verschillende mappen navigeren



9. Vervolgens klik je op RotterdamScene en vervolgens op de scene waar "Rotterdam staat."

10. Nu komt het 3D model met alle OV-modellen tevoorschijn als je de applicatie afspeelt.

3.4 Technische problemen

|  |  |  |
| --- | --- | --- |
| Onderwerp | Probleem | Oplossing |
| VS naar Unity | In VS worden er verschillende packages geïnstalleerd. Deze werken vervolgens niet in Unity | Packages ook nog installeren in Unity. |
| Data ophalen | Door de hoeveelheid data die opgehaald moet worden duurt het inladen van het Unity project lang | De data filteren zodat het niet allemaal meerdere keren opgehaald hoeft te worden |

4 Aanbevelingen

In het onderzoeksrapport staat beschreven wat en hoe er allemaal is onderzocht. Als tip kan er meegegeven worden om voornamelijk naar de OV API gekeken worden en naar de Unity scripts die er geschreven zijn. In het onderzoeksrapport staat ook wat er precies wel en niet gelukt is tijdens het onderzoeken van de API en implementeren van de Unity scripts. Een alternatief voor Github is Unity Teams. Github werkt niet 100% vlekkeloos met Unity en daarom heeft Unity hun eigen version control ervoor gemaakt. Dit is geen gratis optie voor teams die groter zijn dan drie man.

5 Conclusie

Van alle bevindingen die er zijn gedaan is Unity 2020.3.X een uitstekende keus om het project in te maken. Het heeft veel opties en vrijheid gegeven om met libraries en API’s te werken die wij onderzocht hebben. Alhoewel het lastigste gedeelte het ophalen van de ov-data was, omdat er nergens duidelijke documentatie is. Is er na heel veel trial en error gebleken dat KV78Turbo-OVAPI één van de beste open source realtime OV API’s die er is.

Referenties

Atlassian. (z.d.). *What is Trello?* From Atlassian Support: https://support.atlassian.com/trello/docs/what-is-trello/

BairesDev. (z.d., z.d.` z.d.). *The C# Programming Language*. Opgehaald van BairesDev: https://www.bairesdev.com/technologies/csharp/

Gerrist. (2020, 04 28). *KV78Turbo-OVAPI*. Opgehaald van Github: https://github.com/koch-t/KV78Turbo-OVAPI/wiki

Github. (2022). *Hello World*. From Github docs: https://docs.github.com/en/get-started/quickstart/hello-world

Hornshaw, P. (2022). *What is Discord?* From Digital Trends: https://www.digitaltrends.com/gaming/what-is-discord/

Microsoft. (2022). *System.Net Namespace*. Opgehaald van Microsoft Docs: https://docs.microsoft.com/en-us/dotnet/api/system.net?view=net-6.0

Microsoft. (2022). *What is Microsoft Teams*. From Microsoft Support: https://support.microsoft.com/en-us/topic/what-is-microsoft-teams-3de4d369-0167-8def-b93b-0eb5286d7a29

Microsoft. (2022). *What is SharePoint*. From Microsoft Support: https://support.microsoft.com/en-us/office/what-is-sharepoint-97b915e6-651b-43b2-827d-fb25777f446f

Newtonsoft. (z.d., z.d. z.d.). *Json.Net*. Opgehaald van NewtonSoft: https://www.newtonsoft.com/json

Scrum. (n.d.). *What is Scrum?* From Scrum: https://www.scrum.org/resources/what-is-scrum

Gemeente Rotterdam. (z.d.). *De Digitale Stad*. Opgehaald van Rotterdam: https://www.rotterdam.nl/wonen-leven/digitaal/