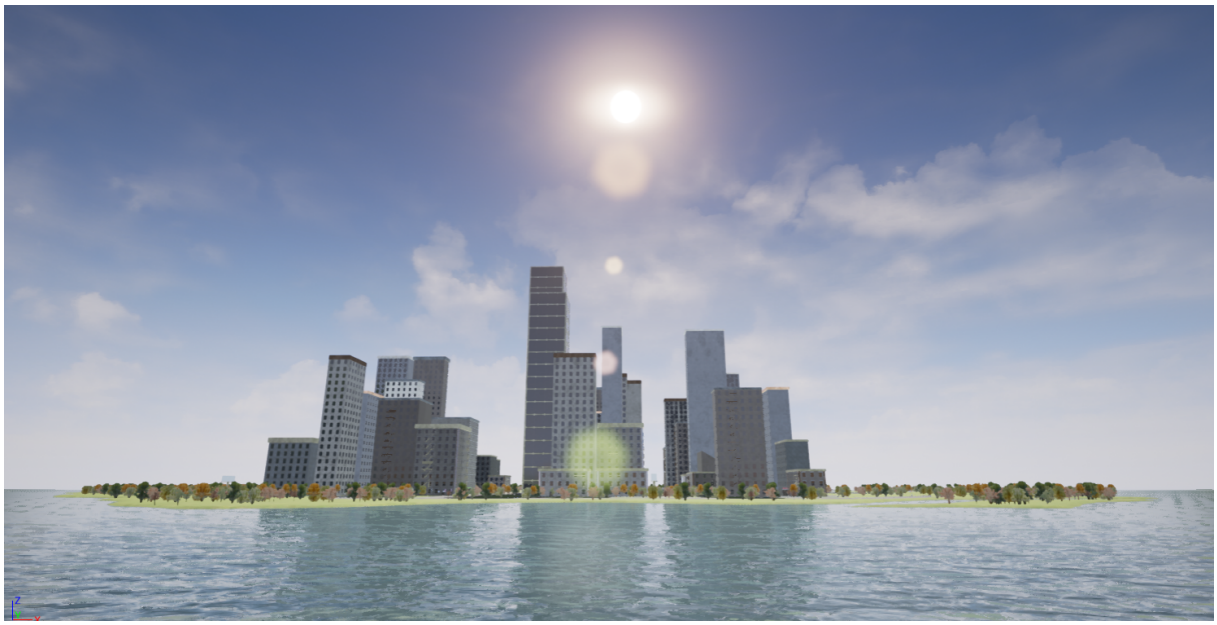


Virtual Reality Car Driving Simulator

Kandidatarbete inom Data- och Informationsteknik

ANTHONY RIZK GUSTAVSSON

FILIP GRANQVIST

JIM BENGTSSON

MANNE ENGELKE

RASMUS LORENTZON

ROBIN SVENINGSON

Chalmers tekniska högskola
Göteborgs universitet
Institutionen för Data- och Informationsteknik
Göteborg, Sverige, Juni 2016

Kandidatarbete

Virtual Reality Car Driving Simulator

ANTHONY RIZK GUSTAVSSON, FILIP GRANQVIST, JIM BENGTSSON,
MANNE ENGELKE, RASMUS LORENTZON OCH ROBIN SVENINGSON.

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborgs universitet

Göteborg 2016

Virtual Reality Car Driving Simulator.
ANTHONY RIZK GUSTAVSSON, FILIP GRANQVIST, JIM BENGTSSON,
MANNE ENGELKE, RASMUS LORENTZON OCH ROBIN SVENINGSON

Examinator : Olof Torgersson

Kandidatarbete 2016 :10

Institutionen för Data- och Informationsteknik
Chalmers tekniska högskola
Göteborgs universitet
412 96 Göteborg
Telefon : 031-772 1000

Omslag :
En bild tagen av Projektgruppen på den Stadskarta som tillverkats åt simulatorn.

Institutionen för Data- och Informationsteknik
Göteborg 2016

Virtual Reality Car Driving Simulator.

ANTHONY RIZK GUSTAVSSON, FILIP GRANQVIST, JIM BENGTSSON, MANNE ENGELKE, RASMUS LORENTZON OCH ROBIN SVENINGSON.
Institutionen för Data- och Informationsteknik, Chalmers Tekniska Högskola och Göteborgs universitet

Kandidatarbete

SAMMANFATTNING

Många företag och högskolor i Göteborgsområdet arbetar på olika sätt med biltillverkning. Dessa företag och högskolor kan uppleva problem när de testar nya koncept för bilar eller då de testar befintlig teknologi. Dessa tester utförs ofta i simulatorer och svårigheterna i processen kan vara att simulatorn är dyr eller att simulatorn inte är tillräckligt verklighetstrogen. Målet med detta projekt var att skapa en realistisk bilsimulator som använder Virtual Reality, och testa om denna teknologi kan användas för att skapa en mer realistisk testprocess, eller jämförbar realism till väsentligt lägre pris, för bilindustrin.

Slutresultatet av detta projekt är en simulator med en realistisk känsla till en relativt låg kostnad, som använder Virtual Reality och ett enkelt bilgränssnitt. Vårt projekt illustrerar hur kommersiell Virtual Reality kan spela en stor roll i bilars framtida testprocess. Den simulator vi tillverkat utgör en bra grund som kan vidareutvecklas för att kunna användas i testprocessen.

Denna rapport diskuterar huruvida VR kan användas i bilindustrins testprocess samt processen för att komma fram till detta. Den beskriver också simulatorns slutgiltiga funktionalitet, processen bakom skapandet av simulatorn och intressanta vidareutvecklingar.

Denna rapport är skriven på svenska.

Nyckelord: Virtuell Verklighet, Bilsimulator, Unreal Engine, VR-HMD

ABSTRACT

Many companies and universities in the Gothenburg region are in different ways working with vehicle manufacturing. These companies and universities can experience difficulties when testing new concepts for vehicles and evaluating existing vehicle technology. The tests are often done in simulators and the difficulties might be that the simulator is expensive or that the simulator is not realistic enough. The goal of this project was to create a realistic car driving simulator using Virtual Reality technology, and test if the technology could be used to create a more realistic and cheaper testing process for the vehicle industry.

The end result of this project is a simulator with a fairly realistic feeling and a relatively cheap price, which is using Virtual Reality and a simple vehicle interface. We show that Virtual Reality can indeed play a big role in the future of the testing process of vehicles, and with more work the simulator developed for this project could possibly be used successfully in the vehicle testing process.

This report discusses the answer to the question whether or not VR can be used in the testing process of vehicles, and the process behind finding the answer. It also describes the final functionality of the simulator created for this project, the process behind creating it and future improvements that can be made.

This report is written in Swedish.

Keywords: Virtual Reality, Car Driving Simulator, Unreal Engine, VR-HMD

FÖRORD

Denna rapport beskriver ett kandidatarbete som skrevs år 2016 under institutionen för Data- och Informationsteknik på Chalmers tekniska högskola. Projektgruppen vill tacka alla de som har varit med och hjälpt till med projektet. Framförallt Daniel Sjölie för hans tid och engagemang under projektets gång.

Ordlista

AI	Artificiell Intelligens, är ett begrepp som syftar på att få datorer att bete sig intelligent.
Actor	Ett objekt i Unreal Engine, som går att placera ut på en karta.
Autodesk Maya	Ett utvecklingsverktyg för att skapa 3D-modeller.
Blender	Ett utvecklingsverktyg för att skapa 3D-modeller.
Blueprint	En beskrivning av hur objekt ska se ut och bete sig i Unreal Engine.
Controller	Ett Blueprint i Unreal Engine som innehåller funktionalitet för hur en Pawn kan kontrolleras.
C++	Ett programmeringsspråk.
Cybersickness	Den åkomma som kan uppstå då en användare använder sig av VR, främst VR-HMD:s. Ett vanligt symptom är illamående.
DK1	Developer Kit 1, syftar på en VR-HMD; Oculus Rift Developer Kit 1.
DK2	Developer Kit 2, syftar på en VR-HMD; Oculus Rift Developer Kit 2.
FPS	Frames Per Seconds, antalet bildrutor som visas per sekund på en skärm.
GIMP	GNU Image Manipulation Program, ett bildbehandlingsprogram.
git	Ett versionshanteringssystem som används för att hålla ordning på olika versioner av samma mjukvara.
GUI	Graphical User Interface, ett grafiskt användargränssnitt.
HMI	Human Machine Interaction, ett område där man studerar interaktionen mellan människor och maskiner.
Interaktionsdesign	Läran om interaktioner mellan system och användare.
Joint	En nod i skelettstrukturen hos en Skeletal Mesh.
Karta	Syftar på en värld i ett datorspel vars storlek är begränsad.
Material	Syftar i det här sammanhanget på information om en texturs egenskaper. Informationen kan exempelvis vara hur solen reflekteras i texturen eller om texturen skall vara osynlig på utvalda ställen.
OSVR	Ett open-source projekt för att försöka standardisera VR-headset. Har även skapat ett eget VR-headset med samma namn.
Pawn	En typ av Actor i Unreal Engine som går att kontrollera av spelaren och AI.

Polygon	Den grundläggande byggsten som används inom datorgrafik för att definiera 3D-modeller.
Static Mesh	En samling av polygoner som används för att beskriva 3D-modeller inom datorgrafik. Static syftar på att de interna vektorerna i modellen inte är rörliga, men hela Meshen kan röra på sig.
Skeletal Mesh	En samling av polygoner som används för att beskriva 3D-modeller inom datorgrafik. Skeletal syftar på att de interna vektorerna i modellen, till skillnad från Static Meshes, är rörliga.
Stadskartan	Den stora kartan innehållandes en stor stad som tillverkades av Projektgruppen till simulatorn.
Starter Content	Ett bibliotek med exempelobjekt, så som Meshes, material, texturer och annat som kommer med från början i UE4.
Textur	Information om hur en yta på en 3D-modell ser ut. En textur är ofta en bild.
UE4	Förkortning för den fjärde versionen av Unreal Engine.
Unreal Engine	Ett utvecklingsverktyg med inbyggd spelmotor som används för spelutveckling.
USD	Amerikanska dollar.
VR	Virtual Reality, virtuell verklighet, är en teknik som används för att projicera en digital värld på ett sätt som får användaren att känna att användaren befinner sig i den digitala världen.
VR-HMD, VR-headset	Virtual Reality Head Mounted Display, är den hjälm du tar på dig för att betrakta den virtuella verkligheten.

Innehållsförteckning

1 Inledning

1.1 Bakgrund

Under 1990-talet var det många teknikintresserade som hoppades på att virtuell verklighet skulle bli nästa stora trend. Det släpptes flera produkter inom denna kategori, bland annat Nintendos Virtual Boy. Det visade sig dock att produkten inte räckte till för att övertyga marknaden, och försäljningen av Nintendo Virtual Boy upphörde efter ungefär ett år [1]. Problemet var att antingen var de för dyra eller helt enkelt inte tillräckligt bra för att slå igenom, och ofta orsakades illamående vid användning [2][3].

Efter en tid i skuggan har utvecklingen av VR tagit fart igen under de senaste åren. Ett av företagen som har utmärkt sig extra mycket är Oculus VR, som år 2014 köptes upp av Facebook för nästan 2 miljarder USD [4]. Oculus VR är ett av flera företag som år 2016 planerar att lansera sina konsumentversioner av VR-HMD:s. En av anledningarna till att produktionen av VR-headsets har ökat kraftigt under de senare åren är att smartphones har introducerats och haft en explosionsliknande utveckling. Utvecklingen av smartphones har lett till att samma skärmar som används i telefonerna kan användas i VR-HMD:s [5], vilket gör det möjligt att producera en kvalitativ VR-HMD till lägre kostnad.

Det uppenbara användningsområdet för VR är i spel och andra underhållningssyften. VR har en stor framtida potential för datorspelsintresserade, och de som ivrigt väntar på att utvecklingen av VR-anpassade spel skall ta fart går en ljus framtid tillmötes [6]. Det finns dock många andra användningsområden av VR-tekniken som kanske inte från början är uppenbara. VR skulle exempelvis kunna användas inom sjukvården, främst i form av terapi, eller exempelvis inom militären, där soldater kan träna i verklighetstrogna scenarion [7]. Bilindustrin är ett exempel på en bransch som kan ha mycket att vinna på VR-teknologin. Testningsprocessen inom bilindustrin är problematisk eftersom det kan vara svårt att genomföra säkra och realistiska tester. Ofta går det inte att sätta en användare i en riktig bil och utföra verkliga tester, eftersom det finns en stor säkerhetsrisk. Istället kan man använda en simulator och simulera verklighetstrogna situationer, men ofta är sådana simulatorer stora och kostsamma. Med vårt kandidatarbete önskar vi kunna bidra till en förändring av hur testprocessen går till med hjälp av VR-teknik, och förhoppningsvis kan göra utrustningen både mindre och billigare.

1.2 Syfte

I detta kandidatarbete vill vi testa vad Virtual Reality har för potential inom bilindustrin och bilindustrins testprocess. Syftet med arbetet är att tillverka en bilsimulator med stöd för Virtual Reality, och undersöka huruvida det går att skapa en simulator som är billigare och mindre än de dyra och stora lösningar som används idag, utan att det påverkar verklighetstrogenheten negativt.

Vår simulator är specifikt till för forskningsgruppen som arbetar med interaktionsdesign på Chalmers tekniska högskola. Där forskas det bland annat om HMI-system i bilar, vilket är interaktionen mellan bil och förare. Simulatorn är tänkt att användas i dessa forskares testprocess.

1.3 Avgränsningar

En av projektets avgränsningar är att ingen hänsyn tas till om användaren kör på ett avvikande sätt. Det kommer inte läggas någon tid på att få det att se naturligt ut om användaren exempelvis krockar med ett träd, en annan bil eller en byggnad. En bra simulatorupplevelse bygger på att användaren kör på samma sätt som i verkligheten. Avgränsningen gjordes eftersom forskarna som är tänkta att använda produkten

antagligen bryr sig mer om att man faktiskt krockade än hur animationen för krocken ser ut.

Under en intervju i vår förstudie, se bilaga B, nämnde en av de intervjuade personerna att det vore bra om vi följde och implementerade någon av de standarder som finns för simulatorer. Exempelvis är OpenDRIVE en standard för att beskriva hur vägarna ska se ut och vilka format de ska ha [8]. Detta hade dock varit oerhört tidskrävande, därför beslutade vi oss för att inte följa någon sådan standard.

Då detta är ett utvecklingsprojekt kommer de flesta delar inte att vara perfekta vid slutförandet av projektet. Vi skapar snarare grunden som det sedan ska gå att vidareutveckla och förbättra. Detta gäller exempelvis vår AI och omgivning.

De VR-HMD:s vi kommer ha möjlighet att testa är OSVR, Oculus Rift DK1 och Oculus Rift DK2. Det är möjligt att andra VR-headset fungerar, men det är inget vi kommer kunna garantera då vi ej har möjlighet att testa dessa.

2 Teknisk bakgrund

2.1 Virtual Reality

Moderna produkter som använder sig av VR-teknologin är ofta någon form av HMD-system som användaren bär på huvudet medan äldre varianter, såsom Sensorama, närmast kan liknas vid en arkadmaskin [9]. HMD:er innehåller en skärm för vardera öga som användaren upplever den virtuella verkligheten genom. Dessutom följer mjukvaran användarens huvudrörelser kring enhetens egna axel. Vissa HMD:er har även medföljande Positional Tracking-enheter vilka gör det möjligt för mjukvaran att följa HMD:ens position i rummet. En nackdel med HMD:er är dock att de i dagsläget ger ett mindre synfält än verkligheten[10]. Andra varianter av VR-produkter som tidigare har använts och som idag används är kub-system; det vill säga ett rum på vars väggar en virtuell verklighet projekteras [11].

Vanliga problem med VR är prestanda[12] och upplösning[13], och det visade sig redan under 1990-talets första generation av VR-HMD:s, då exempelvis Nintendos Virtual Boy inte hade tillräckligt bra hårdvara[14]. Den nya generationen av HMD:s som utvecklas idag fokuserar därför på att motverka dessa problem genom exempelvis högre upplösning [15].

Ett annat problem som VR-teknologin dras med, och i förlängningen även vår simulator, är att avståndsbedömning i VR är svår att göra och ofta underskattas [16][17][18]. Detta beror enligt en undersökning inte på det begränsade synfält som fås med HMD:s [19], men det lämnar ytterligare frågor och forskningen är ej säker på orsaken till fenomenet [20].

Även Cybersickness är ett fortsatt stort problem inom VR-branschen[21] och det är något utvecklare av mjukvara till VR-teknologi behöver ha i åtanke. Cybersickness upplevs ofta som mycket likt åksjuka[22] och därför är illamående ett vanligt symptom. Forskare har inte kunnat bevisa exakt vad som ligger bakom illamåendet[23] men att hitta en lösning är en hög prioritet hos många parter. Forskning har visat att ökad latens mellan användarens rörelser och vad som händer i VR-upplevelsen är en faktor som kan få upplevelsen att resultera i Cybersickness [24]. Det finns även en välkänd teori om att Cybersickness beror på skillnaden mellan det som användaren ser på skärmen och vad de övriga sinnena upplever. Denna teori kallas för Sensory Conflict Theory [22].

2.2 Ratt och pedaler

Ratt och medföljande pedaler har varit en väsentlig del av bilsimulatorer och racingspel under en längre tid. Syftet med ratten och pedalerna är att få en mer verklighetstrogen kontroll, i jämförelse med att exempelvis använda tangentbord. Ett flertal företag producerar den här typen av produkter, några framstående företag är Logitech och Thrustmaster. Det finns flertalet olika modeller att välja på med olika nivåer av verktyg inbyggda. En dyrare modell så som Logitechs G27 kan innehålla koppling, gaspedal, bromspedal, växelspak, extra knappar. Därtill har den en teknologi som kallas Force Feedback, vilket innebär att ratten simulerar att den utsätts för verkliga krafter och därmed känns tung och svårmanövrerad.

2.3 Unreal Engine

2.3.1 Generellt om Unreal Engine

Unreal Engine är namnet på en spelmotor utvecklad av företaget Epic Games, och den senaste utgåvan är Unreal Engine 4, vilken släpptes i maj år 2012. Just nu är Unreal Engine 4 gratis att använda så länge bruttointäkterna för spelet inte uppgår till mer än 3000 USD per kvartal. Om bruttointäkterna är högre än denna summa kräver Epic Games en 5% royaltyinkomst på sålda spel enligt deras licens [25]. Unreal

Engine 4 har ett brett stöd av plattformar; Windows, OS X, Linux, Xbox One, Playstation 4, SteamOS, HTML5, iOS och Android. Dessutom har Unreal Engine 4 ett stort stöd för VR-HMD:s; Oculus Rift, HTC Vive, Steam VR, Playstation VR och Samsung Gear VR. [26]

2.3.2 Unreal Editor

Utveckling med Unreal Engine sker normalt i C++, men med hjälp av programvaran Unreal Editor kan man smidigt och enkelt utveckla ett projekt och arbeta med spelmotorn utan att behöva programmera i C++. Syftet med Unreal Editor är att all utveckling, vare sig man arbetar med material, animeringar, partikelsystem eller användargränssnitt, ska ske på en och samma utvecklingsplattform. Ett tydligt tema Unreal Editor följer är att utveckling av spel ska ske på ett visuellt sätt och att så mycket funktionalitet som möjligt ska vara tillgänglig utan att behöva programmera med kod.

2.3.3 Blueprints och Blueprint Visual Scripting

Vid utveckling av spel i UE4 är Blueprint Visual Scripting alternativet till att programmera med C++. Detta är ett nodbaserat gränssnitt där händelser, funktioner och variabler agerar som noder, vilka sedan kopplas ihop med kablar mellan varandra. För att skapa de flesta objekt i UE4 så skapar man först en Blueprint-klass, vilket är en beskrivning av hur objekt ska se ut och bete sig. Med hjälp av Blueprint Editor bygger man upp utseendet genom att addera komponenter, till exempel Static Meshes, ljus och ljud. Sedan utvecklas beteendet för Blueprinten med Blueprint Visual Scripting.

2.3.4 Actors, Pawns och Controllers

Ett objekt av typen Actor är ett föremål som kan placeras i världen, vilket också är den vanligast förekommande objekttypen i Unreal Engine 4. Blueprint-klasser som beskriver Actors innehåller två typer av grafer tillämpade för Blueprint Visual Scripting; Construction Script och Event Graph. Den förstnämnda grafen används till att programmera statiska beteenden som inte ändras vid exekvering. Detta kan vara inställningar som ska utföras när Actorn placeras ut i världen i Unreal Editor, till exempel generera ett slumpmässigt material för komponenterna i Actorn. I den andra grafen, Event Graph, sker programmeringen för Actorns dynamiska egenskaper, till exempel funktionalitet hur spelaren ska kunna interagera med Actorn.

En användbar subklass av Actor är typen Pawn. Detta är en Actor som går att styra, till exempel en bil eller människa. Pawns i sin tur går att kontrollera och styra med Actors av subklassen Controller. Dessa Controller Blueprints beskriver styregenskaper och begränsningar hur antingen spelaren eller AI ska kunna kontrollera sin Pawn.

2.3.5 Andra viktiga komponenter

Volym	Volymer är osynliga Actors i Unreal Engine som ofta används till att kontrollera om en Pawn befinner sig i ett givet område.
Navmesh	Detta är en typ av volym med den speciella egenskapen att det omringade området skapar möjlighet för användning av AI:s inbyggda Pathfinding-funktioner, och därmed skapar en navigerbar area för AI.

2.4 3D-modellering

2.4.1 3D-modelleringsprogram

Med hjälp av 3D-modelleringsprogram skapas objekt som kan användas i olika 3D-miljöer. Dessa objekt är uppbyggda av ett set av polygoner. En polygon består av ett antal sammankopplade punkter med olika positionering och tillsammans beskriver de objektets geometriska form. Sträckan mellan två punkter kallas för en kant och området mellan flera kanter i en polygon kallas för en yta. Många 3D-modelleringsprogram har verktyg för såväl skapande och manipulering av polygoner, samt applicering av material och texturer på dess ytor. Funktionalitet för animering är också vanligt förekommande. Några vanliga exempel på populära 3D-modelleringsprogram är 3ds Max, Autodesk Maya och Blender.

2.4.2 Meshes

En Mesh är en samling polygoner som beskriver ett 3D-objekt. Det kan vara allt från en enkel kub till mer avancerade objekt, såsom bilar och människor. Meshes kan sedan i sin tur delas upp i Static Meshes och Skeletal Meshes. Static Meshes behåller alltid samma form medan Skeletal Meshes dynamiskt kan ändra form under programexekvering.

Skeletal Meshes består, utöver den Mesh som beskriver objektet, av en skelettstruktur. Med hjälp av noder, kallade Joints, som kopplas samman byggs en hierarki upp som beskriver ett skelett för objektet. Varje Joint blir sedan en bindningspunkt för var del av Meshen som skall kunna manipuleras individuellt. Då en Joint manipuleras sker en beräkningsprocess som räknar ut hur varje polygon i Meshen skall roteras och translateras för att passa den nya skelettformationen. Under programexekvering appliceras animationer på Skeletal Meshes som bestämmer hur varje Joint skall manipuleras vid varje tidssekvens.

En fördel med Unreal Engine är att animationer sker på skelettstrukturen som i sin tur förändrar Meshen, vilket betyder att flera Meshes kan använda sig av samma skelett och därför kan också samma animationer appliceras. En annan fördel är att man kan köra flera animationer samtidigt på samma objekt. Exempelvis kan en animation för ansiktsuttryck och en animation för kroppspositionering kombineras för att få karaktärer att springa och prata samtidigt. [27]

2.4.3 Texturer och material

En Mesh beskriver bara formen på ett objekt, den innehåller själv ingen information om färg och utseende. En textur är en bild som placeras över Meshen, som definierar Meshens utseende. Ett material beskriver däremot alla andra egenskaper som finns, som inte kan beskrivas med bastexturen. Exempelvis hur mycket ljus som reflekteras på grund av ojämnheter på ytan.

2.5 Artificiell Intelligens

2.5.1 Generellt om AI

Ett datorsystem med intelligent beteende brukar tilldelas namnet AI. Att skapa en intelligent mjukvara, som kan lösa problem och göra val utifrån egna slutsatser är högst aktuellt för speltillverkare, och i vårt fall för en realistisk bilsimulator. AI för spel behöver dock inte vara ett komplext system, utan kan vara begränsade till enkla spelregler. AI:n behöver heller inte ha egenskapen att lära sig själv, utan kan ha förprogrammerade regler och beteenden. Syftet med att implementera AI i spel är att kunna skapa en värld av intelligenta enheter som spelaren kan interagera med och uppfatta som levande, vilket tillför mycket dynamik till spelet.

2.5.2 Beteendeträd

Ett beteendeträd är en hierarkisk trädstruktur av grenar med olika operationer. Ett beteendeträd letar efter Tasks att utföra åt den kontrollerade enheten genom att börja exekveringen vid roten, arbeta sig ner till löv som innehåller uppgifter, och sedan propagera upp genom trädet igen. Denna process upprepas så länge trädet är aktivt.

2.5.3 Artificiell Intelligens i Unreal Engine 4

Unreal Engines primära verktyg till att utveckla AI:s med är beteendeträd, vilket är en av fyra komponenter som behövs för att utveckla en komplett AI. Först behövs en kontrollerbar Actor, en Pawn, som AI:n ska kunna kontrollera. Denna Pawn styrs av en Controller, i detta fall en AI-Controller, och i Controllern initieras sedan vilket beteendeträd och Black Board som ska vara aktivt. En Black Board är platsen där AI:n sparar viktig data som delas mellan en AI:s olika komponenter. Ett förtydligande av vad alla dessa komponenter har för syfte i en komplett AI i Unreal Engine kan ses som en analogi med människokroppen, se Tabell 1.

Människa	AI	Funktion
Kropp	Pawn	En kontrollerbar enhet
Nervsystem	Controller	Funktionalitet för att styra enheten
Hjärna	Beteendeträd	Väljer uppgifter på ett intelligent sätt
Minne	Black Board	Sparar viktiga variabler

Tabell 1 – Analogi mellan människokroppen och en AI i Unreal Engine 4.

Ett beteendeträd i sin tur designas med hjälp av sex olika typer av noder. Trädet kan innehålla ett godtyckligt antal noder och det finns ingen gräns för hur många barn en nod kan erhålla. Dock är det bara så kallade kompositnoder som har grenar. Dessa är de sex olika typer av noder som beteendeträd stödjer:

Task	Alla löv är vanligtvis Task-noder och denna nod kan inte ha några barn. Detta är en vanlig uppgift för AI:n, som exekveras en gång, och sedan returnerar om Task-noden lyckades eller inte till nodens förälder.
Selector (kompositnod)	När exekveringssignalen når denna nod skickas signalen vidare till ett barn i taget, startat från vänster. Ett barn i taget exekveras tills barnet returnerar sant, då avbryts processen och Selector-noden returnerar sant. Denna nod är användbar för att låsa delar av beteendeträdet beroende på variabler och tillstånd.
Sequence (kompositnod)	Likt en Selector så är syftet med en Sequence att exekvera sina barn, ett i taget, fast med annan logik. En Sequence fortsätter nämligen att exekvera sina barn så länge dem returnerar sant. Detta är användbart då flera Task-noder ska exekveras i en sekvens.
Simple Parallel (kompositnod)	Denna nod är Unreal Engines möjlighet till parallell exekvering av noder. En Simple Parallel har två grenar; den ena måste leda till en Task, och den andra kan vara ett helt delträd.
Service	I Unreal Engine sitter en Service ihop med valfri kompositnod. Denna Service exekveras i en loop så länge som kompositnoden den är bunden till är under exekvering.
Decorator	En Decorator, likt Service, binds samman med kompositnoder. Dessutom kan en Decorator bindas till Task-noder. Dessa utgör villkor för när subträdet den blockerar får exekvera.

3 Metod

3.1 Utvecklingsverktyg och programspråk

3.1.1 Unreal Engine 4

När projektet startade bestämde vi oss för att använda UE4 som spelmotor då den har bra inbyggt stöd för VR och är gratis att använda i vårt syfte [25]. Ett alternativ till UE4 hade varit Unity, en annan spelmotor, men då vår handledare till största del arbetat med UE4 såg vi fler fördelar att välja UE4 framför Unity. Ett annat möjligt alternativ hade varit att börja från grunden utan någon spelmotor. Detta uteslöt vi nästan direkt då det hade inneburit extremt mycket extra arbete, som vi nu slapp genom att använda en färdig spelmotor.

3.1.2 Blender, Autodesk Maya och GIMP

När vi skulle modellera de grafiska komponenterna var vi tvungna att använda några andra program än UE4 då detta inte går att göra i UE4. Då använde vi oss av GIMP för bildredigering, på grund av dess väl fungerande funktionalitet och eftersom det är gratis att använda. Vi använde oss av Blender och Autodesk Maya för 3D-modellering. Vilket av dessa program som användes av varje gruppmedlem berodde på personliga preferenser och eventuellt tidigare erfarenheter av något av programmen.

3.1.3 Programspråk

Eftersom vi använde oss av UE4 var tanken att vi skulle använda C++ och Blueprint Visual Scripting som programmeringsspråk. Det fungerade dock så bra med Blueprint Visual Scripting så vi använde aldrig C++.

3.2 Versionshantering

Ett versionshanteringssystem var självklart att använda eftersom vi var så många som arbetade med samma kod. Det system vi har använt oss av var git, eftersom git är välkänt och väl fungerande. Ett alternativ till git hade kunnat vara exempelvis SVN, men då de flesta av oss hade tidigare erfarenheter av git valdes git framför SVN.

3.3 Förarbete

En kravspecifikation togs fram i ett tidigt skede av projektet, baserad på de möten och diskussioner vi haft med personer från målgruppen. Syftet med kravspecifikationen var att hjälpa oss att planera arbetet och underlätta prioriteringen av arbetsuppgifter. Denna kravspecifikation byggdes sedan på och stärktes av en förstudie, bestående av intervjuer och telefonsamtal med personer från målgruppen.

3.4 Användarstöd

För att underlätta användningen av simulatorn och framtida utveckling skapade vi en användarmanual. Syftet med denna användarmanual var att göra det så enkelt som möjligt att kunna köra, men även redigera, simulatorn.

3.5 Testning

Testning och verifiering av det vi arbetat med har varit en fortlöpande process genom hela projektet. Alla mindre delar har testats var för sig innan vi har gått vidare med dem och använt dem i större skala. Detta för att lättare hitta vad det är som är fel om något inte fungerar som förväntat. Själva processen för testen har sett olika ut beroende på vad det är som testats. Förutom den löpande testningen har vi också utfört användartester. Till dessa användartester skapades en blankett som fylldes i efter att användartesterna avslutats. Blankettens syfte var att utvärdera slutprodukten genom att sammanfatta användarnas upplevelse av simulatorn, VR och hur de påverkades av Cybersickness. Blanketten som användes återfinns i Bilaga D.

4 Kravanalys

4.1 Målgruppen

Den primära målgruppen för vår slutprodukt är människor som arbetar med någon form av testning av bilar. Detta kan alltså innebära människor som arbetar för både företag och i forskningssyften. Målgruppen var känd från start och har varit oförändrad genom hela projektet. Som tidigare nämnt är simulatorn specifikt utvecklad åt de på Chalmers tekniska högskola som arbetar med testning av HMI-system i bilar. Dessa forskare är en del av den primära målgruppen, och de är tänkta att använda simulatorn i framtiden, därför har extra fokus legat i att undersöka vad just de efterfrågar.

En sekundär målgrupp har däremot uppkommit under projektets gång. Denna målgrupp är människor som generellt är intresserad av VR och datorspel. Vi insåg med tiden att denna sekundära målgrupp skulle kunna vara intresserad av vår slutprodukt utan att vi behöver göra några förändringar i produkten i sig. Det är viktigt att understryka att den sekundära målgruppen däremot inte har påverkat några beslut under projektets gång, och att den primära målgruppen har legat i fokus.

4.2 Förstudiens format

Förstudien var en viktig del av det initiala arbetet. Eftersom bilindustrin är en bransch som ingen av oss i Projektgruppen vet så mycket om, insåg vi i ett tidigt skede att det var viktigt att vi undersökte ordentligt vad som förväntades av vårt arbete för att slutresultatet skulle vara intressant. Förstudien gick ut på personliga intervjuer samt telefonsamtal med människor inom den primära målgruppen. De som deltog i dessa intervjuer och samtal var personer från olika delar av målgruppen och med olika bakgrunder inom branschen. Personerna som deltog var personer både från företag, ett statligt forskningsinstitut och en högskola. Detta för att ge en så bred bild som möjligt av vad målgruppen var ute efter. Under intervjuerna användes en standardiserad mall med frågor som Projektgruppen tog fram i förväg, se bilaga C. Dessa frågor valdes ut på ett sådant sätt att de inte skulle påverka mottagarens svar och istället ge mottagaren möjlighet att styra konversationen. I slutet av varje intervju berättade vi om vår specifika idé, och tillsammans reflekterade vi och den intervjuade om idéns potential och vad som skulle krävas för ett lyckat resultat. Telefonintervjuerna var mindre formella och innebar samtal där vi diskuterade vår idé, VR, bilsimulatorer och hur den intervjuades organisation arbetade med dessa frågor.

4.3 Förstudiens resultat

4.3.1 Dagens testprocesser

Förstudien bekräftade väldigt många antaganden som gjorts i början av projektet. Både företag och andra organisationer som arbetar med fordonssäkerhet använder sig mycket av simulatorer, eftersom det oftast inte går att göra tester ute i riktig trafik. Dessa simulatorer kan variera i storlek och pris, och beroende på vilken organisation det är så kan simulatorn vara allt från en flera ton tung bil-kaross på hydraulikställning till en enkel projektor framför ett förarsäte.

Det största problemet för dagens simulatorer är bristande verklighetstrogenhet. Att användaren hela tiden är medveten om att det är en simulering och att det inte är på riktigt, tror de intervjuade påverkar användarnas beslutsförmåga. I verkligheten står förarens och andra människors liv på spel om något skulle gå fel. I en simulator står inget på spel, och det vet nog de flesta användare om, vilket resulterar i att de kanske tar andra beslut än vad de hade gjort i en liknande situation i verkligheten.

Projektgruppens uppfattning, efter att ha varit och tittat på en billigare och en dyrare simulator, var att verklighetstrogenheten beror på simulatorns komplexitet. Om hela karossen du sitter i rör sig i flera led samtidigt som du kör simulatorn borde det vara mer realistisk än om du ser att bilen i simulatorn rör på sig, men att du samtidigt känner att du sitter stilla. Vi tror dock att den realistiska känslan avtar när man tittar på statiska skärmar eller projektorer.

Förstudien, ihop med andra diskussioner och möten, gjorde att vi dessutom fick ett konkret exempel på hur testprocessen kunde gå till på en av organisationerna. Denna process innebar att organisationen först tillverkade ett scenario med den specifika trafiksituation de ville testa. Detta scenario såg ut på samma sätt varje gång man körde simulatorn. Organisationen kunde sedan testa detta scenario på många användare och föra statistik på viktiga parametrar. Statistiken kunde sedan analyseras för att dra slutsatser om det som testades.

4.3.2 VR-simulatorer

Alla de som deltog i förstudien var överens om att Virtual Reality är en högst intressant teknik när det kommer till bilsimulatorer. Känslan av att ha på sig ett VR-headset och kunna titta sig omkring i världen kan påverka verklighetstrogenheten för simuleringen markant. Det var dessutom så att en eller flera organisationer redan hade börjat titta på VR-simulatorer och hur de kunde användas i testningssammanhang. Det uppmärksammades däremot många potentiella problem med VR, och många av problemen är de problem som är generella för VR-tekniken, så som latens, låg uppdateringsfrekvens och att användarna riskerar att drabbas av Cybersickness.

4.4 Kravspecifikation

Förstudien, tillsammans med möten, diskussioner och egna analyser gjorde att vi kunde ta fram en kravspecifikation, vars uppgift var att få projektet att röra sig i rätt riktning och att hjälpa oss prioritera vad som var viktigast. De krav vi satte upp för slutprodukten anges i listan nedan.

Modulär bil med realistisk fysik	En bil som användaren kan köra är ett av de mest fundamentala kraven. Denna bil behöver ha en realistisk fysikmotor med exempelvis vikt, hastighet och rattutslag. Bilen måste också vara modulär, så att det på ett enkelt sätt går att ändra dess utseende och funktion.
Stöd för ratt, pedaler och VR	Simulatorn behöver ha stöd för minst ett VR-headset samt en typ av ratt och pedaler.
Vägkomponenter	Olika vägkomponenter skall tillverkas som skall kunna användas för att bygga ihop mer komplexa vägnät. Dessa vägkomponenter skall ha olika antal filer och vara av olika typ; exempelvis korsningar, rondeller, svängar och raksträckor.

Stadskarta med omgivning	Simulatorn behöver ha en stad med vägar, trottoarer, byggnader, terräng, kringliggande natur, AI-styrda människor samt AI-styrd trafik. Staden skall vara uppbyggd med hjälp av de olika vägkomponenterna. I staden skall användaren kunna köra omkring fritt.
Scenarion	Det skall vara möjligt att skapa scenarion, där användaren placerar AI-människor och AI-bilar på specifika ställen och där användaren även kan ange hur de ska röra på sig. Dessa scenarion måste sedan fungera på exakt samma sätt varje gång man kör scenariot. Ett fåtal scenarion skall skapas av Projektgruppen för att demonstrera scenario-funktionaliteten.
AI	AI krävs för människor och bilar.
Användargränssnitt	Ett lättanvänt användargränssnitt krävs så att användarna enkelt kan komma igång med simulatorn. Detta användargränssnitt bör innehålla inställningar, val av scenarion samt statistik från olika scenario-körningar.
Moduläritet	Simulatorn i sin helhet skall vara modulär och enkel att redigera, så att det går att vidareutveckla simulatorn i framtiden.
Verklighetstrogenhet	Det är viktigt att användarna upplever en hög verklighetstrogenhet, så att användarna kör på ett liknande sätt som de hade gjort i verkligheten.
Pris	Det är viktigt att simulatorn inte är för dyr, och relativt den simulator för mångmiljonbelopp Projektgruppen har varit och tittat på, skall priset på utrustningen till detta projektets simulator vara betydligt mindre.
Storlek	Utrustningen som används till simulatorn bör vara liten i storlek, och kunna monteras ner vid behov.

4.5 Slutsats

Förstudien bekräftade många teorier vi haft från start. Vi lärde oss också mycket nytt och fick tips om saker som vi inte hade reflekterat över. Arbetet efter förstudien var sedan väldigt influerat av förstudiens resultat och den framtagna kravspecifikationen, och de hjälpte oss framförallt att veta vilken prioritet saker och ting skulle ha. Högsta prioritet var att först skapa en modulär grund med vägar, omgivning, en förarbil, AI och stöd för ratt, pedaler samt VR. Nästa prioritet var scenarion, verklighetstrogenhet i körupplevelsen och användargränssnitt. Prioriteringen grundade sig delvis i vad som ansågs vara viktigast, men också i att en del koncept var beroende av att andra saker redan var avklarade.

Sammanfattningar av förstudiens intervjuer och telefonsamtal finns i bilaga B.

5 Utförande

5.1 Virtual Reality

Att nyttja VR-teknologin i Unreal Engine har varit högt prioriterat i projektet, särskilt under de första veckorna av implementationsfasen. Inledningsvis använde sig Projektgruppen av ett VR-HMD, OSVR, byggt av Razer. Stödet för detta HMD var dock ej integrerat i Unreal Engine från början och därför användes ett externt plugin vid namn OSVR-Unreal i projektet för att HMD:en skulle fungera.

Vi ansåg att OSVR var relativt svårt att använda sig av då det krävde att en serverapplikation kördes i bakgrunden och vi hade dessutom problem med latens och dålig upplösning. Detta ledde till att vi under projektets gång därför bytte ut OSVR mot Oculus DK1 och DK2 HMD:s eftersom latensen hos dessa produkter inte upplevdes som lika dåliga som OSVR. Unreal Engine har inbyggt stöd för Oculus produkter och därför var det enkelt att byta till DK1 och DK2.

För att minimera de problem med Cybersickness, exempelvis illamående, som upplevs av många användare i VR-miljöer har vi försökt undvika att implementera fenomen som hindrar synkroniseringen av simulatorn och verkligheten enligt Sensory Conflict Theory [22]. Då användaren inte sitter i en fysisk bil gjorde vi valet att låta användaren röra sitt huvud fritt genom den virtuella bilens chassi. Om användaren hade fastnat när huvudet tar emot den digitala bilens chassi, men fortsatt röra sig i verkligheten, så hade det kunnat innebära att den tidigare nämnda synkroniseringen mellan verklighet och simulation hade blivit påverkad. Vidare visar en studie[28] att mycket rörelse gör att användaren upplever mer Cybersickness, detta gjorde att vi valde att undvika att simulera vissa rörelser som normalt sker vid exempelvis en krock.

5.2 Ratt och pedaler

För att få ratten och pedalerna att fungera användes pluginet JoystickPlugin då Unreal Engine ej har stöd för joystick-enheter som standard. Ratten, en Thrustmaster Ferarri 458 Italia, har ett signalspann på -1 till 1, där 1 är maxutslag åt höger. Detta är även det spann som insignalen till styrfunktionen som bilar i Unreal Engine nyttjar. Pedalerna till ratten gav två olika signaler vilka i Unreal Engines system för insignaler, även kallade keybinds, dock ej var konsekventa. Därför användes pluginet för att kringgå detta och få tillgång till alla inkopplade joystickenheters signaler. Pedalsignalerna transformerades sedan så att dessa följer ett mönster där värdena ligger mellan 0 och 1, där 1 är maxutslag.

Unreal Engines inbyggda system för växlar ansåg vi inte kändes verklighetstroget. Exempelvis lades backväxeln i om föraren bromsade för länge. För att förbättra känslan togs ett system fram för att köra med en manuell växellåda. Systemet var likt de som används i racingbilar, där växeln är placerad på ratten, och togs fram med hjälp av den existerande grunden för växelsystem som fanns i Unreal Engine. Då det manuella systemet ej kändes intuitivt att använda implementerades även möjligheten att använda en automatisk växellåda. En knapp för att byta mellan de två olika lägena togs fram. Därtill skapades knappar för att lägga i och ur backväxeln och även för att använda bilens signalhorn.

5.3 Grafiskt användargränssnitt och parametrar

Inledningsvis planerade Projektgruppen att skapa ett VR-kompatibelt grafiskt gränssnitt i 3D. Allt eftersom projektet fortgick framgick dock att ett avancerat grafiskt gränssnitt inte var en hög prioritet för gruppen. Istället bestämdes att ett klassiskt 2D-gränssnitt byggt med Unreal Engines UMG-widgets, vilket är Unreal Engines standardverktyg för att skapa 2D-gränssnitt, skulle utvecklas. Vi upptäckte dock att 2D-menyer ej var VR-kompatibla. Det i kombination med att projektet led mot sitt slut och att det ej finns

någon muspekare att navigera med i en VR-kompatibel 3D-miljö gjorde att vi tog beslutet att avaktivera VR i menyerna. Istället aktiveras det enbart under simulation.

GUI:t implementerades med målet att ha följande menyer: huvudmenyn, kartmenyn, inställningsmenyn, statistikmenyn, informationsmeny och en pausmeny. Dessa implementerades med en simplistisk layout då grafiken ej var ett stort fokus för projektet.

5.4 Bilen

Utvecklingen av den bil som föraren sitter i beslöts, som tidigare nämnt, vara ett av kraven i den kravspecifikation som togs fram. Eftersom många av de andra delarna av projektet var beroende av att ha en bil att tillgå för att kunna utvecklas, exempelvis AI:n, valde vi att utgå från en bil som redan fanns i Starter Content i Unreal Engine. På denna bil bytte vi sedan ut de grafiska komponenterna. De nya komponenterna skapades med hjälp av Autodesk Maya och implementerades i olika faser under projektets gång allteftersom de olika delarna blev klara. Denna process tog längre tid än beräknat på grund av att inlärningsprocessen och utvecklingen var tvunget att ske samtidigt då ingen i projektgruppen hade någon tidigare erfarenhet av 3D-modellering. I första fasen implementerades ett tomt chassi av bilen med fyra hjul som hade samma funktionalitet som startbilen. I nästa skede implementerades interiören av bilen såsom säten och instrumentpanel. I den tredje versionen av bilen implementerades backspeglar, ratt och pedaler med funktionalitet och slutligen implementerades även blinkers, bromsljus och olika material för de olika delarna av bilen.

Ett problem som uppstod i och med att bilen implementerades i faser var att det vid varje ny implementering krävdes omfattande justering av AI:n. Parametrarna som bestämmer bilens förmåga att svänga, accelerera, med mera, är många och då exempelvis interiören lades till innebar det en ökad vikt för bilen vilket påverkade AI:ns förmåga att klara av kurvor. Ett annat problem var att parametrarna kalibrerades för att göra körupplevelsen så bra som möjligt för användaren, vilka då även ändrades för AI:n. Detta löstes genom att separera inställningarna för AI- och förarbilen. Inställningsförändringar i exempelvis maximal svängradie för däcken, däckstorlek, friktionsvariabel och förändringar hos förarens parametrar påverkar därför inte längre AI:n.

När det kommer till testning av bilen så inleddes detta väldigt sent på grund av att den slutliga bilen färdigställdes senare än planerat. Testningen bestod av att köra runt bilen i Stadskartan, och följdes sedan av parameterjustering för att förbättra körupplevelsen.

5.5 Omgivningen

5.5.1 Vägar och broar

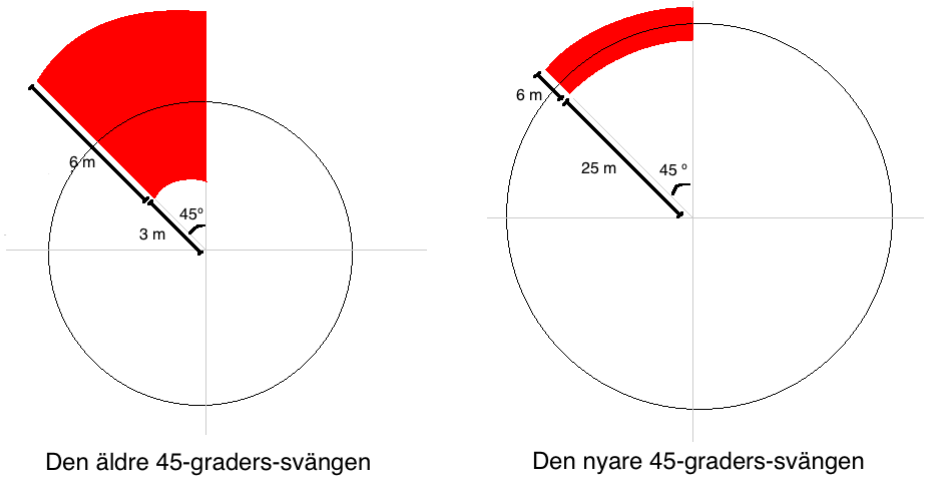
Vägarna var en av de många grupper av komponenter som prioriterades högt från början. Inledningsvis gjordes en uppskattning på vilka typer av vägkomponenter som skulle behöva finnas för att göra en verklighetstrogen karta. Däremot uppkom det under tidens gång flera andra vägkomponenter som glömdes bort i den initiala planeringen.

Till en början undersöktes det huruvida enkla vägkomponenter kunde göras direkt i Unreal Engine. Efter några försök att göra svängar med hjälp av Unreal Engines inbyggda geometriska former togs beslutet att ett modelleringsprogram skulle behövas. Därefter var vi ett par i gruppen som lärde oss att använda modelleringsprogram för att tillverka Meshes.

Vägkomponenterna som tillverkades till slutprodukten var kombinationerna som uppstod av tre olika attribut. Dessa tre attribut var geometrisk form, antal filer samt med eller utan trottoar. Den geometriska formen kunde exempelvis vara raksträcka, 90-graders-kurva, rondell eller korsning. Antal filer kunde variera mellan enkelriktat (1 fil), enkelfilig (2 filer, 1 i vardera riktning) samt dubbelfilig (4 filer, 2 i vardera riktning). Trottoaren valde vi att bygga in i vägkomponenten, eftersom vi insåg att det skulle vara tidskrävande att placera ut trottoarer för hand. Många av alla dessa kombinationer av de tre attributen skapades.

Vägkomponenternas dimensioner baserades på en snabb undersökning över vilka dimensioner som användes i verkligheten. En bils bredd uppskattades vara mellan 190 och 210 cm, och en vägfil uppskattades vara ungefär 300 cm bred. Vägbitarna som tillverkades fick längder som var relativt korta, exempelvis fick en raksträcka längden 300 cm. Detta gjorde det möjligt att kombinera vägbitar som små pusselbitar för att skapa komplexa vägnät. Trottoarers höjd uppskattades, genom att studera trottoarer i närheten av Chalmers, till att vara 12 cm höga och 150 cm breda.

Inledningsvis skapades flera olika typer av svängar till de olika vägtyperna med olika svängvinkel; antingen 45, 60 eller 90 grader. Efter att vi låtit AI-bilar testa att köra på dessa vägkomponenter insåg vi att vi behövde göra om dessa svängar, eftersom AI-bilarna inte kunde ta kurvorna på grund av kurvornas snävhet. Därför skapades nya svängar som inte var lika snäva. Dock tillverkades endast svängar med svängvinkel 45 grader på nytt, på grund av tidsbrist. Problemet med svängarnas snävhet illustreras i figuren nedan.



Figur 1 – Figuren visar hur snävheten på två vägbitar med samma svängvinkel har förändrats genom att göra vägbitens körsträcka längre. Den röda delen är vägen, som är lika bred i de båda svängarna, men den högra svängen är längre vilket minskar dess snävhet.

Efter att Static Meshes för vägkomponenter, texturer samt material skapats var det slutgiltiga steget att anpassa vägkomponenterna för AI:n. Detta arbete gick ut på att placera olika typer av noder på de olika vägkomponenterna. Dessa noder är ej synliga när man kör simulatorn, men de har en viktig funktion eftersom de anger hur AI-bilarna ska bete sig på den givna vägkomponenten. Läs mer om AI-noder i kapitel 5.6. För att minska antalet noder, men framförallt för att göra det lättare att placera ut vägkomponenter fick vi idén om ett system för att “dra“ ut raksträckorna, det vill säga ett system som automatiskt bygger

en rak väg mellan två punkter. Systemet placerade sedan AI noder i ändarna av raksträckan, istället för att varje unik vägbit skulle ha egna noder.

Utöver vägar arbetades det även med att ta fram broar. Broarnas syfte var egentligen av estetisk natur och de gjorde att vi kunde bygga upp avancerade motorvägsavfarter och broar över vattendrag.

5.5.2 Vägskyltar

För att efterlikna verklig trafik gjordes ett antal vägskyltar av olika form och funktionalitet i Blender. Många av skyltarna skapades utan något annat syfte än utökad detaljrikedom i trafiken. En del av vägskyltarna fick däremot en funktion, exempelvis hastighetsskyltarna som ställer in maxhastigheten på AI-bilarna som passerar skylten. Dimensionerna på vägskyltarna baserades på en uppskattning av verkliga skyltar, och skylten blev 50×50 cm på en 200 cm hög påle.

5.5.3 Trafikljus och gatlyktor

Både trafikljusen och gatlyktorna tillverkades i Blender. Vi uppskattade verkliga gatlyktor till att vara 4 m höga, och det blev även våra gatlyktors höjd. Vi skapade också möjligheten att tända gatlyktorna. Denna funktion implementerades främst för framtida användning eftersom vår stadskarta för tillfället har dagsljus, vilket innebär att gatlyktorna aldrig är tända.

Trafikljusens dimensioner baserades även dem på verkliga motsvarigheter. Ursprungligen placerades det tre så kallade Point Lights i trafikljusen, vilket är en vanlig ljuskälla i Unreal Engine 4, en för varje färgindikator. Men eftersom dessa Point Lights är beräkningstunga ersattes de med en enda Point Light som flyttar sig inuti trafikljuset till den position den tända indikatorn har, ändrar färg och som inte lämnar några skuggor.

5.5.4 Byggnader

Vägnätet lades ut innan arbetet med byggnader inleddes. Detta gjorde att vi behövde ett bra och modulärt system för att skapa byggnader. Därför tillverkades byggnaderna med modulära väggbitar. Olika Static Meshes skapades med varierande fönsterkarmar, dörrar och antal våningar. Dessa Meshes användes senare för att skapa 16 olika byggnadsgrupper. En byggnadsgrupp innehöll alltid ett tak, en vanlig tom vägg, en vägg med endast fönster och en vägg med entrédörr och fönster. Det kunde dessutom finnas hörnbitar eller väggar med fönster och brandstegar. För en byggnadsgrupp var alltid materialet som användes och antalet våningar detsamma för alla delar. Alla väggar som skapades gjordes 9 m breda, och konsekventheten i bredd gjorde det möjligt att kombinera byggnadsdelar till ett hus som satt ihop.

Detta system med byggnadsgrupper gjorde att man kunde använda en av de 16 olika byggnadsgrupperna för att skapa ett unikt hus, genom att kombinera gruppens byggstenar på olika sätt. Det innebar alltså att det fanns 16 olika byggnadstyper med varierande material och antal våningar, men att det också kunde finnas byggnader från samma byggnadsgrupp som ändå var olika stora och hade olika former.

Utöver byggnadsdelarna skapades även två stycken skyskrapor med olika antal våningar. Dessa skyskrapor skapades eftersom vi ansåg att en variation och unikhet i byggnader ökade verklighetstrogenheten av simulatorn.

Arkitekturen i staden inspirerades av bilder på byggnader och gator i New York, USA, eftersom det är en stad som många känner igen och kan relatera till. För att lägga till ytterligare en dimension av verklighetstrogenhet tillverkades det också affärer, restauranger, billboards och reklamtavlor. Restaurangerna

och butikerna tillverkades på ett sådant sätt att det gick att placera dem på en befintlig byggnad. För optimal prestanda vore det bättre om vi hade byggt in butiker och restauranger i de befintliga byggnaderna, för att minska överflödiga polygoner, men vi ansåg att det skulle ta för mycket tid jämfört med den lilla prestandaförbättring det eventuellt hade medfört.

Alla byggnader som tillverkades fick bli tomma på insidan, eftersom vi ansåg att det vore allt för prestandakrävande att se någonting innanför varje fönster. Vi studerade dessutom verkliga byggnader i dagsljus, och vi insåg att en solig dag så är de flesta fönster nästan helt reflektiva, och på avstånd kan det vara svårt att avgöra om det är tänt eller släckt i lägenheten. Därför tillverkade vi glasrutor som gjordes mörka och reflektiva, så att användarna inte ser att byggnaderna endast är tomma skal.

5.5.5 Träd och buskar

Eftersom världen hade varit väldigt tom utan natur, valde vi att tillverka ett antal träd och buskar. Naturen var aldrig en hög prioritet, och den gjordes efter att andra viktigare funktioner var avklarade. För att tillverka träd upptäckte vi att det fanns en teknik i Blender för att skapa trädstammar och slumpa fram grenar. Dessa Static Meshes hade dock innehållit väldigt många polygoner, och vi valde istället en annan teknik som innehöll mycket färre polygoner per träd och snabbade upp tillverkningsprocessen. Tekniken gick ut på att använda fyra plan roterade kring ett och samma centrum, och sedan applicera samma textur på alla fyra planen. Dessa träd och buskar gick väldigt snabbt att tillverka, men efter att ha placerat ut alla träd och buskar insåg vi under prestandaoptimeringen att lösningen ur ett prestandaperspektiv kanske inte var den bästa. Problemet som uppstod rörde skuggningen av naturen och därför valde vi att helt enkelt stänga av skuggningen för träd och buskar.

Träden och buskarna placerades ut med ett verktyg i Unreal Engine 4 som kallas för Foliage. Detta innebär att alla träd och buskar tillsammans utgör en Actor, för att det inte ska bli för tungt att arbeta med flera hundratals utplacerade träd och buskar. Processen lät oss måla ut naturen med olika penslar som automatiskt slumpade mellan de olika typerna av träd och buskar.

5.5.6 Terräng

Terrängen i världen tillverkades efter att vägnätet och byggnader redan var på plats. Terrängen placerades på samma nivå som vägkomponenterna, men eftersom trottoarkanterna var 12 cm höga placerades byggnader ut 12 cm i luften, vilket innebar att terrängen under byggnader behövdes höjas manuellt. Terrängen lyftes med olika penselverktyg som finns inbyggda i Unreal Engine. Eftersom en riktig stad ofta innehåller många grönområden och vattendrag, tillverkades dessutom två parker och en flod med hjälp av terrängverktygen som fanns inbyggda i UE4. Det sista som gjordes relaterat till terrängen var att måla på olika material. Detta var också en inbyggd funktion i UE4, som tillät oss att måla på olika material på terrängen med hjälp av olika penselverktyg, så att övergångar mellan olika material inte var raka streck utan att de istället såg mer verklighetstrogna.

5.6 Artificiell Intelligens

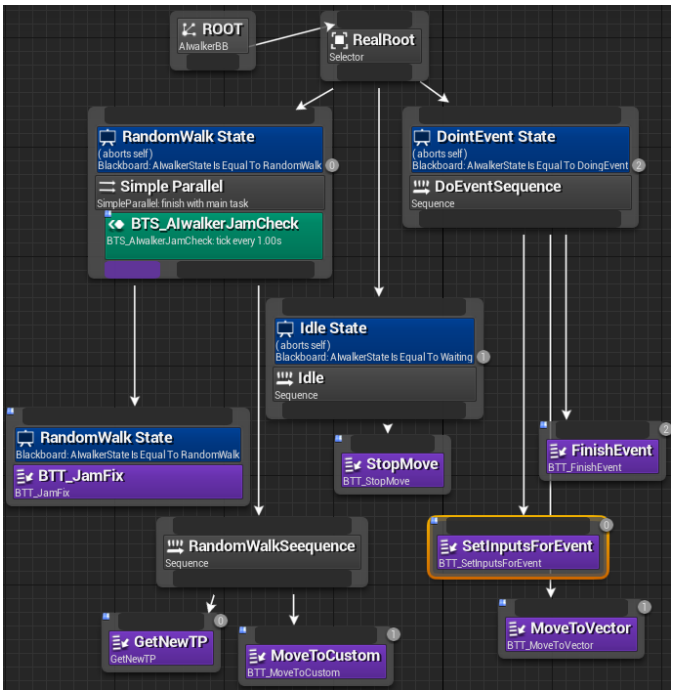
5.6.1 Generellt om AI

Att utveckla bilar styrda av Artificiell Intelligens till bilsimulatorn var en självklarhet. Vi beslöt oss också att implementera AI-styrda gångtrafikanter, vilket vi ansåg skulle erbjuda en mer realistisk upplevelse när föraren befinner sig i stadstrafik. Dessutom möjliggör detta simulation av scenarion med övergångsställen. Både bilarna och människorna utnyttjar Unreal Engines stöd för beteendeträd. Ett krav för både människorna och bilarna var att man skulle kunna skapa beteenden som avviker från den

slumpmässiga körsträckan och triggas av bland annat föraren. Vi kallar dessa förprogrammerade beteenden som är utplacerade på vägarna för events, vilket till exempel kan vara en bil som bryter mot en trafikregel, och tillsammans skapar dessa events ett scenario.

5.6.2 AI-människor

Människorna använder sig av en Pawn som finns i Starter Content för UE4. Detta är en enfärgad människofigur och för att få en mer diversifierad mängd gångtrafikanter implementerade vi en funktion i dess Construction Script som slumpmässigt väljer ut en textur av fördefinierade texturer till sin Mesh-komponent. Människorna kan användas genom att placera ut dem i kartan, men vi implementerade också en Spawn-funktion. Ett antal människor specificeras som input till Spawn-funktionen och när simulatorn startar blir dessa människor utplacerade automatiskt.



Figur 2 – AI-människornas Beteendeträd. Decorators är blåa, Services är gröna och Tasks är lila.

Vi implementerade tre olika tillstånd i AI-människornas beteendeträd: *IDLE*, *RANDOMWALK* och *DOINGEVENT*. De tre olika tillståndens regler definieras med hjälp av tre olika subträd som bara kan exekveras när respektive tillstånd är aktivt. *IDLE*, *RANDOMWALK* och *DOINGEVENT* illustreras i Figur 2. För en genomgående beskrivning om hur de olika noderna i ett beteendeträd fungerar, se kapitel 2.5.

Subträdet *IDLE* består enbart av en Task, se Figur 2, *StopMove*, som avbryter pågående uppgifter och gör så att AI-människan står stilla. Syftet med *IDLE* är att vi behövde ett tillstånd som AI-människan står stilla i, till exempel vid väntan på att ett övergångsställe ska bli passerbart.

Syftet med subträdet *RANDOMWALK* är att AI-människan ska leta efter ett slumpmässigt mål att gå till, gå dit utan att fastna, och sedan börja om sekvensen. *JamCheck* och *JamFix* innehåller funktionalitet för

att AI:n ska undvika att fastna när de går runt i kartan. Med hjälp av en Simple Parallel-nod kan *JamFix* exekveras samtidigt som AI-människan utför resten av sina uppgifter i tillståndet *RANDOMWALK*, se Figur 2.

Det resterande subträdet i *RANDOMWALK* består av en sekvens av två Task-noder:

GetNewTP	Hämtar ett nytt slumpmässigt mål, *TargetPoint*, att gå till och sparar det. Dessa *TargetPoints* är en typ av Actors som är förplacerade på de vägkomponenter med gågator.
MoveToCustom	Utnyttjar Unreal Engines inbyggda navigationslösning, Navmeshes, och navigerar till det nya målet.

När AI-människan når slutdestinationen, specificerad av *GetNewTP*, notifierar *MoveToCustom* att destinationen är nådd och sedan börjar hela processen om så länge AI-människan fortfarande befinner sig i tillståndet *RANDOMWALK*. Ett problem som uppstod var när AI-människorna valde *TargetPoints* att gå till som inte täcktes av en *Navmesh*. Det fanns nämligen ingen navigerbar area till dessa *TargetPoints* och det resulterade i att människorna stod stilla. Lösningen var att implementera stöd för att *JamCheck* också skulle upptäcka dessa tillfällen. Den sista grenen från roten i beteendeträdet exekveras under tillståndet *DOINGEVENT*, se Figur 2. Detta tillstånd aktiveras på förutbestämda platser i ett scenario med en Actor av typen *AIwalkerEvent*. Subträdet för *DOINGEVENT* består av tre Task-noder i en sekvens:

SetInputsForEvent	Ställer in olika inställningar för AI-människan specificerade av *AIwalkerEvent*.
MoveToVector	Har samma funktionalitet som *MoveToCustom* fast med en vektor i *AIwalkerEvent* som mål.
FinishEvent	Räknar ut vilket tillstånd människan ska återgå till, förstör *AIwalkerEvent* om det behövs och återställer inställningar som var specifika för eventet.

Dessa tre Task-noder skapar tillsammans stöd för manuell programmering av AI-människans beteende med hjälp av *AIwalkerEvent*. *AIwalkerEvent* behöver en referens till den AI-människan som ska utföra eventet, och dessutom en referens till vilken Actor som ska kunna trigga eventet. Actorn *AIwalkerEvent* består av två komponenter som går att förflytta var för sig; den ena komponenten är en volym som triggar eventet om den specificerade Actorn överlappar med volymen, och den andra komponenten är målet som AI-människan skall nå om eventet triggas.

AIwalkerEvent innehåller inställningar som beskriver olika beteenden hos AI-människan vilket gäller under den tid människan tar sig till målet. *AIwalkerEvent* bestämmer bland annat i vilken hastighet AI-människan ska gå mot sitt mål och om det ska ske en fördröjning mellan när eventet triggas och när AI-människan faktiskt blir notifierad. Vi implementerade dessutom möjlighet för AI-människorna att ignorera trafikregler när de befinner sig i tillståndet *DOINGEVENT*, vilket skapade möjligheten att designa ett event där AI-människan springer rakt ut på vägen. Vi skapade också en inställning som specificerar om *AIwalkerEvent* ska förstöras efter eventet är genomfört. Detta löser problemet om förarbilen kör på samma plats i ett scenario flera gånger, men ett event ska bara triggas en gång. Slutligen implementerades en inställning som avgör vilket tillstånd, *RANDOMWALK* eller *IDLE*, som AI-människorna ska återgå till efter eventet är genomfört. Detta skapade möjligheten för AI-människorna att återgå till flödet av

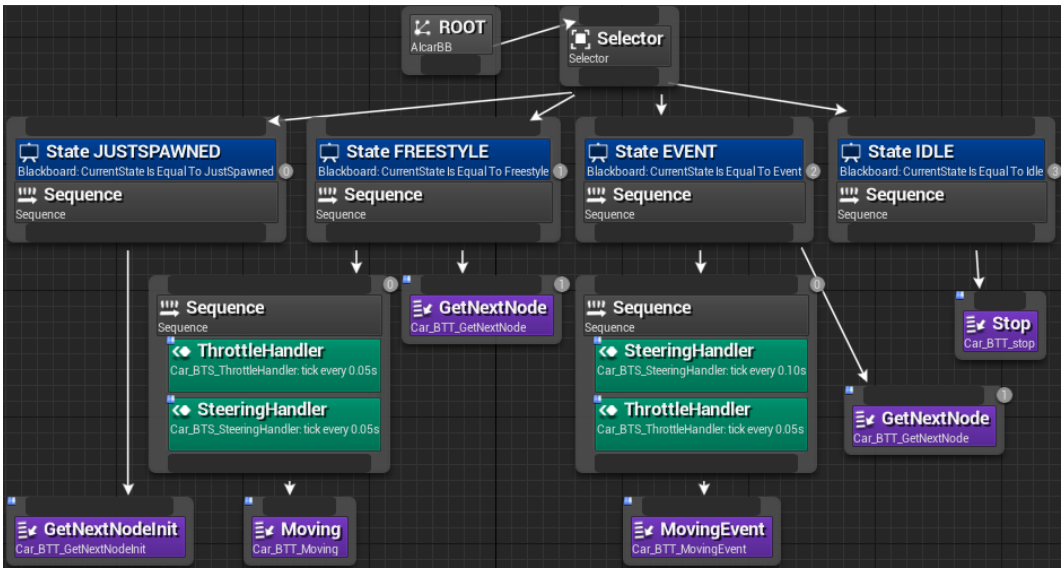
gångtrafikanter eller stå stilla för att undvika att störa andra pågående events.

Dessa inställningar gav oss möjligheten att skapa mer avancerade events, som till exempel säger: “Om 2 sekunder, gå till denna destination i hastigheten 300 units/sekund utan att ta hänsyn till trafikregler, och fortsätt sedan gå runt i tillståndet *RANDOMWALK*“.

5.6.3 AI-bilar

Vi har implementerat vår *AI-Controller* för bilar i två olika bilmodeller. Vi utnyttjade dels en av bilmodellerna i Unreal Engines Starter Content, och dels en modifierad version av förarbilen. Skillnaden mellan AI-bilen och förarbilen är att AI-bilen är lättare att manövrera, med bland annat mindre tröghetsmoment och lika brant styrkurva vid alla hastigheter. Syftet med en mer lättmanövrerad bil för *AI-Controllern* är att minska risken att AI:n hamnar utanför vägen. Vi har inte implementerat stöd för hantering av bilar som hamnar ur kurs, och därför är det extra viktigt att detta inte sker.

Likt människorna så implementerade vi en slumpgenerator som väljer färgen på bilens Mesh-komponent i dess Construction Script för en mer diversifierad trafik. Precis som med förarbilen, så ärver bilmodellerna sina egenskaper från Unreal Engines egna bilimplementation. Detta leder dock till att AI-bilen inte stödjer Unreal Engines inbyggda navigeringssytem med Navmeshes, utan AI-bilarna måste istället reglera gas och styrriktning för att nå målet. AI-bilarnas beteendeträd struktureras upp med ett flertal tillstånd; *JUSTSPAWNED*, *IDLE*, *EVENT* och *FREESTYLE*, se Figur 3.



Figur 3 – AI-bilarnas Beteendeträd. Decorators är blåa, Services är gröna och Tasks är lila.

Likt människorna så har vi implementerat en Spawn-funktion också för AI-bilarna. De blir utplacerade på tillåtna vägnoder som är fria från överlappande bilar. När AI-bilarna blir utplacerade automatiskt av funktionen så startar de alla i tillståndet *JUSTSPAWNED*. Subträdet för detta tillstånd består av en Task kallad *GetNextNodeInit*, se Figur 3. Denna Task räknar ut första målnoden för AI-bilen och sedan roterar AI-bilen i riktning mot denna nod. Om en AI-bil placeras ut manuellt i kartan så måste man själv ange en första målnod för bilen, dessutom måste bilen också roteras manuellt i riktning mot första målnoden.

Task-noden *GetNextNodeInit* avslutar sin exekvering med att byta tillstånd för AI:n till antingen *EVENT* eller *FREESTYLE* beroende på vilken typ av nod som är nästa målnod.

Tillståndet *IDLE* har samma syfte som hos AI-människorna, vilket är att stanna. Subträdets enda Task, *Stop*, lägger i handbromsen och släpper på gaspedalen. Vi använder oss av tillståndet *IDLE* i koden när ett tillfälle kommer då AI-bilen inte vet vad den ska göra härnäst, till exempel när en slinga av vägnoder tar slut eller när bilen inte har någon startnod.

Tillståndet *FREESTYLE* är AI-bilarnas motsvarighet till AI-människornas tillstånd *RANDOMWALK*. Syftet är att man ska kunna generera slumpmässig trafik där enda kravet är att AI-bilarna ska följa trafikreglerna. Vi har utvecklat AI-bilens körförmåga med hjälp av en Service för styrningen, en Service för gaspedalhanteringen och två Tasks, se Figur 3. Den första Task-noden, *Moving*, har som uppgift att pausa exekveringen av sig själv, och sedan vänta på signal att bilen har anlänt till målnoden. Syftet med detta är att vi har implementerat två Services, *ThrottleHandler* och *SteeringHandler*, se Figur 3, som innehåller köregenskaperna och förflyttar bilen till målnoden. När AI-bilen sedan når sin målnod blir *Moving* aviserad och beteendeträdet går sedan vidare till nästa Task. När *Moving* signalerar att noden är nådd hämtar AI-bilen en ny målnod med hjälp av nästa Task, *GetNextNode*. Om inte föregående målnod pekar på någon nästa nod så sätts tillståndet till *IDLE*, annars genereras en nästa målnod utifrån vart i nätverket av noder bilen befinner sig. När AI-bilen har en ny målnod som destination börjar processen i beteendeträdet om med *Moving* igen.

Tillståndet *EVENT* har vi skapat för att kunna hantera de extra tillägg som eventnoderna kräver vid utveckling av scenarion. Samma Tasks och Services används i detta subträd, förutom *Move*, som har ersatts med *MoveEvent*, se Figur 3. Funktionaliteten för *MoveEvent* är densamma som *Move* med tillägget att bilen kan tuta.

SteeringHandler är vår automatiska styrfunktion som reglerar ingående styrsignaler till bilen. En vinkelskillnad från bilens rotation till målnoden räknas ut och transformeras sedan till lämplig styrsignal. Först räknas vinkelskillnaden ut mellan bilen och dess målnod. Sedan kontrolleras om noden ligger åt vänster eller höger om bilen för att utgöra om insignalen för styrningen ska vara negativ eller positiv. Slutligen räknar *SteeringHandler* ut hur stor insignalen för styrningen ska vara. Vi märkte att om vi hela tiden använder maximal styrsignal för att minimera vinkelskillnaden så snabbt som möjligt, så blev bilen instabil och vobblar vid små vinkelskillnader. Därför implementerade vi en gradvis ökning av styrsignal desto större vinkelskillnaden är.

Servicen *ThrottleHandler* fungerar som en automatisk farthållare och är den mest komplexa Blueprinten i AI-bilarnas beteendeträd. Unreal Engine har sin egna implementation av syn för AI:s kallad AIPercetion. Om man lägger till denna komponent så kan AI:n bli aviserad när ett föremål befinner sig inom dess synradie. Efter en del tester uppfattade vi dock denna funktionalitet som långsam när bilen kör fort, och så var det svårt för AI:n att hålla reda på bilar som redan var innanför synradien. Det vill säga ibland blev AI:n aviserad att ett föremål blockerade vägen när det redan var för sent för AI-bilen att hinna bromsa. Vi bestämde oss istället för att utveckla en mer effektiv syn för bilarna genom att använda oss av Unreal Engines egna implementation av Raycasting. Med hjälp av Raycasting-tekniken implementerade vi dynamiska strålar framför bilen som upptäcker eventuella blockerande objekt. Strålarna är dynamiska i form av riktning och längd. Vi kom fram till att om AI-bilen svänger så behöver Raycastingen ske riktat mer åt det hållet svängen sker, och om AI-bilen kör i hög hastighet behöver Raycastingens längd öka för att söka efter blockerande föremål längre fram så att AI-bilen hinner stanna.

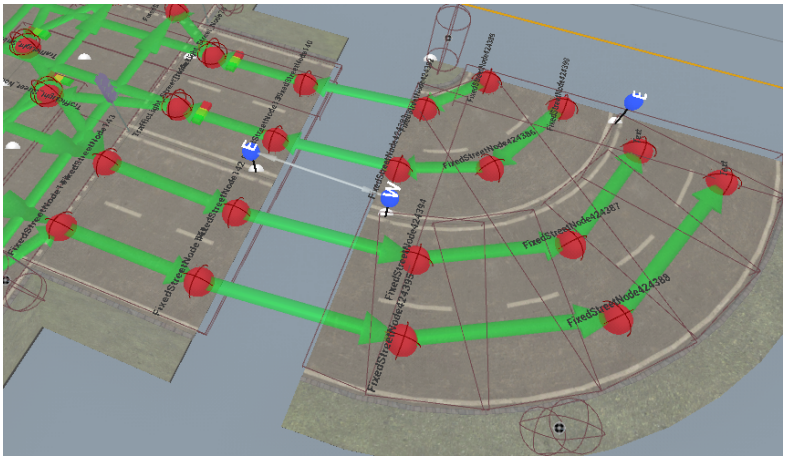
ThrottleHandler använder sig av vår Raycasting som reagerar på andra Pawns och trafikljusnoder. Om

Raycastingen upptäcker en trafikljusnod som är grön så ignoreras den och om trafikljusnoden är röd så stannar bilen. Däremot om noden är gul och nästa tillstånd är röd så stannar bilen bara om den hinner stanna innan AI-bilen når noden, annars så fortsätter den köra, precis som en förare hade agerat i verkligheten. När Raycastingen stöter på en bil som står stilla så stannar AI-bilen enligt samma princip som vid rödljusnoderna. Om den blockerande bilen framför inte står stilla så försöker bilen som kör bakom att matcha hastigheten med den blockerande bilen. Om Raycastingen inte returnerar något blockerande föremål så regleras hastigheten istället med hastighetsgränser från skyltar och noder. Bilen följer den minsta av de två hastigheterna av hastighetsskylten och den rekommenderad hastigheten till nästa målnod, det vill säga om en hastighetsskylt säger 50 km/h men nästa målnod befinner sig i en 90 graders sväng och rekommenderar hastigheten 20 km/h, så byter bilen hastighet till 20 km/h.

5.6.4 Nodsystem

Vi kom fram till att ett nodsystem förplacerat på vägkomponenterna skulle bli en bra grund för regler hur bilarna får köra på vägarna. Ett nät av noder fungerar som en riktad graf där bilarna slumpmässigt navigerar runt beroende på enskilda noders inställningar. Nätet av noder byggs med hjälp av två olika Actors; *FixedStreetNode* och *TrafficLightStreetNode*. Den förstnämnda är av den simplare formen och agerar endast som mål för bilarna, medan *TrafficLightStreetNode* har den extra funktionen att nodens funktionalitet är likvärdigt med ett trafikljus. Hos en nod kan man bland annat specificera en mängd nästa noder som AI-bilen ska kunna välja bland. Dessutom kan man specificera en maximal hastighet som AI-bilen får köra fram till noden med. Detta var en viktig egenskap att implementera hos noderna då vi märkte att AI-bilarna inte kunde hålla sig på vägen i skarpa svängar och med hjälp av noderna kan nu maxhastigheterna sänkas i skarpa svängar så att AI-bilarna inte kör av vägen. Dessutom möjliggjorde detta att maxhastigheten kunde sänkas strax innan korsningar för att säkerställa att AI-bilarna skulle kunna stanna vid eventuellt rött ljus.

Vi valde att förplacera ut alla trafiknoder på vägkomponenterna med hjälp av dess Construction Script, se Figur 4. På detta sätt kunde vi undersöka vilka inställningar som passar bäst för noderna en gång och sedan sätta det som en statisk implementation på de olika vägkomponenterna. Vi valde detta sätt att implementera trafiknoderna på vägkomponenterna på grund av att nu när en korsning placeras ut så finns all logik för hur korsningen fungerar redan utplacerat på vägkomponenten, och designern av scenariot sparar därmed massvis med tid på att inte behöva implementera denna logik från grunden på varje utplacerad vägkomponent.



Figur 4 – Exempel på de förplacerade noderna (röda) och hur det ser ut efter man har kopplat ihop dem med väganslutningarnas Connectors (blåa).

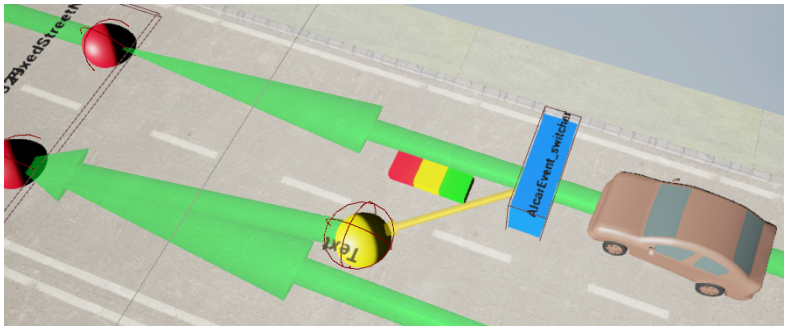
Unreal Engines stöd för Blueprints inuti andra Blueprints var dock begränsad. Det gick bland annat inte att nå inställningar hos de noder som var förplacerade på vägarna genom Construction Script. Detta medförde att vi skapade Actors kallade Connectors med syftet att kunna koppla ihop vägkomponenterna med varandra och på så sätt koppla ihop alla noder däremellan automatiskt. Dessa Connectors finns som tre olika typer, vilka representerar de tre olika anslutningar en vägkomponent kan ha; enkelriktad väg med en fil, dubbelriktad väg med en fil och dubbelriktad väg med två filer. En Connector är förplacerad på varje anslutning i en vägkomponent. Dessa används sedan för att koppla ihop andra anslutningar till vägar som har samma typ av Connector, se Figur 4. Vi kom på att *TrafficLightStreetNodes* inte bara kunde användas för logik vid trafikljuskorsningar, utan de kunde också användas som logik för att implementera trafikregler på vägkomponenter som innehöll bland annat övergångsställen och väjningsplikter. För att kunna styra regleringen av grönt, gult och rött ljus av trafikljusnoderna utvecklade vi två Actors som skötte detta på olika sätt:

TrafficLightController — Syftet med denna Actor är att vi ville ha en enhet som reglerar en grupp trafikljus enbart beroende av tid, som vid en trafikljuskorsning. *TrafficLightController* grupperar ihop en mängd trafikljusnoder i två grupper. Ena gruppen byter tillstånd till grönt samtidigt som den andra gruppen byter till rött och sedan roterar efter ett tidsintervall.

CarChecker — Denna Actor reglerar inte trafikljusnoder beroende av tid, utan sätter en grupp med trafikljusnoder till grönt eller rött beroende på om en bil befinner sig inom en volym. Detta behövde vi för att till exempel sätta noderna till rött på ett övergångsställe då AI-människor vill passera och sätta noder röda vid de anslutningar som har väjningsplikt då det finns inkommande bilar.

Förutom en stabil förplacerad nodstruktur på vägarna så hade vi också kravet att kunna designa avvikande beteenden i form av events för att kunna konstruera scenarion med AI-bilar. Vi utvecklade dessa tre Actors som tillsammans kunde uppfylla alla våra krav på vad som skulle kunna vara möjligt i ett scenario:

AIcarEventNode	Flera Actors av denna typ bygger upp slingor av events för AI-bilarna och är den primära byggstenen vid design av scenarion med AI-bilar. *AIcarEventNode* har samma funktionalitet som en *TrafficLightStreetNode* med en del extra tillägg specifikt för scenarion, till exempel möjlighet att säga till AI-bilen att använda biltuta.
AIcarEventTriggerbox	Denna Actor är en vidareutveckling av *Carchecker*. Vi använder *AIcarEventTriggerbox* till att skifta tillstånden hos de eventnoder som är sammankopplade med denna Actor. Detta resulterar i att vi kan starta och stanna olika flöden av events beroende på till exempel vart förarbilen befinner sig.
AIcarEvent Switcher	Vi behövde utveckla en enhet som kunde hantera ett byte mellan de förplacerade trafiknoderna på vägkomponenterna och eventnoder. Genom att en bil överlappar med denna Actor så byts dess målnod till en ny nod specificerad av *AIcarEvent Switcher*. För ett exempel på hur eventnoder och de förplacerade noderna kan kombineras med hjälp av en *AIcarEvent Switcher*, se Figur 5.



Figur 5 – Exempel på ett event där AI-bilen byter fil från höger till vänster. En *AIcarEvent Switcher* (blå) notifierar AI-bilen att byta målnod till eventnoden (gul), vilket sedan leder tillbaka till det förplacerade nätet av noder (röda).

5.7 Scenarion

5.7.1 Generellt om simulatorns scenarion

Då tanken med detta projekt är att målgruppen ska kunna utföra olika tester och samla in data från dessa tester, valde vi att skapa tre olika scenarion för att visa hur man kan skapa scenarion i vårt projekt. Med ett scenario menar vi att föraren kör en bestämd slinga där det finns något förprogrammerat event som inträffar som vi då samlar in data och statistik kring. Forskarna på HMI-avdelningen har sagt att det är viktigt att kunna ha bra kontroll över vart i scenariot det kan finnas variation mellan varje gång man utför samma test, för att kunna samla in så konsekvent data från testerna som möjligt. Därför är det viktigt att kunna styra AI:n i varje scenario, för att påverka så lite som möjligt på just det som ska testas. Då dessa är tänkta att visa hur man skapar scenarion är de även relativt korta och avskalade. Det är fullt möjligt att göra både längre och mer avancerade scenarion, men då vi inte vet vad det är de vill testa har vi valt att implementera det på detta sätt.

De två första scenariona är skapade med vår Stadskarta som grund. Vi kopierade helt enkelt Stadskartan och tog bort de delar som vi inte ville ha med i respektive scenario. Det vi tog bort var dels vägkomponenter från de områden föraren inte ska köra i, men även utsmyckning såsom byggnader och reklamskyltar som föraren ändå inte kommer att se. Detta gjorde vi eftersom de inte används i respektive scenario och då endast drar ner prestandan. Det tredje scenariot skapade vi helt från grunden genom att använda de komponenter vi har skapat.

I scenario 1 och 2 var nästa steg att placera ut avspärrningar vid de korsningar och rondeller som föraren inte skulle köra in på. Tanken med scenarion är att föraren alltid ska köra samma slinga, och då måste vi på något sätt avgränsa så att föraren inte kör in på andra vägar än de vi har tänkt oss. Detta hade inte behövts om vi implementerat våra scenarion med vägkomponenter där det endast går att köra åt ett håll, alltså inte ha med korsningar och rondeller. Vi ansåg dock att det upplevs mer naturligt om det faktiskt finns dessa variationer av vägkomponenter.

Sedan placerade vi ut en startpunkt för föraren. Från våran huvudmeny i simulatorn går det att välja något av våra scenarion, och då är det vid den utplacerade startpunkten som föraren kommer att starta. Vid slutet av körslingan placerade vi ut en så kallad *EndOfSimulationBox* som täcker hela vägen men som inte är synlig för föraren. När föraren kör in i denna box så avslutas scenariot och en ruta med statistik visas. När scenariot väl är avslutat sparas exempelvis vilken hastighet och vilken styrvinkel föraren hade vid varje sekund av körningen, som användarna sedan har tillgång till.

5.7.2 Scenario 1

Tanken med detta scenario var att visa hur man kan använda AI-människor till att göra något förutbestämt beroende på när föraren når en speciell plats i scenariot. Därför valde vi att implementera så att när föraren passerar en specifik punkt i scenariot springer en AI-människa ut från trottoaren och stannar mitt på vägen i 4 sekunder. Därefter fortsätter människan över till den andra trottoaren. Förutom denna AI-människa finns det ingen annan trafik med alls, vilket vi valde eftersom det blir ett enkelt testscenario.

5.7.3 Scenario 2

Med detta scenario ville vi visa hur det går att använda AI-bilarna till att dels köra exakt så som skaparen av scenariot vill, men även att bilarna sedan kan övergå till att köra runt automatiskt genom vårt inbyggda nodsystem i vägkomponenterna. Detta visade vi genom att implementera ett event där en bil svänger ut framför föraren vid en specifik tidpunkt. Se Figur 6. Denna bil följer först eventnoder som vi har placerat ut, och övergår sedan till att följa nodsystemet som finns på vägkomponenterna.

Figur 6 – Scenario 2 där AI-bil svänger ut framför föraren

5.7.4 Scenario 3

Scenario 3 har vi skapat helt från grunden, det vill säga utan att kopiera Stadskartan, för att testa att vår process för att skapa scenarion verkligen fungerar. Vi ville även testa att ha ett scenario där AI-bilarna kör runt fritt, därför har vi inte satt ut några eventnoder i detta scenario. AI-bilarna använder sig bara av nodsystemet på vägkomponenterna. För att bilarna ska kunna köra runt fritt implementerade vi denna karta så att vägarna inte har ett plötsligt slut. Vi har inte heller avgränsat denna karta så att föraren endast kan köra en specifik slinga, utan föraren kan istället svänga in på olika vägar. Därför har vi satt ut flera *EndOfSimulationBox*:ar så att scenariot garanterat tar slut.

5.8 Testning

Processen för att testa har sett olika ut beroende på vad det är som testats. När vi testade att VR-HMD:en OSVR fungerade i vårt projekt bestod testet i att använda OSVR:s egna programvara för testning. Vid testning av de olika vägdelarna och AI så fick vi bygga små testfall för att se om det fungerade som planerat. Testerna för våra olika kartor bestod i att placera ut alla vägkomponenter i miljön vi hade byggt upp, och sedan placera ut AI-människor och AI-bilar för att se hur allting fungerade tillsammans. Stadskartan testades genom att köra runt i den och iaktta hur den upplevdes både prestandamässigt och även hur det gick att ta sig runt i den. Användargränssnittet testades genom att försöka navigera genom de olika menyerna, ändra värden, öppna kartor och se att resultatet blev det förväntade. Statistikloggarna testades genom att ändra statistikparametrarna i menyn och därefter kontrollera att loggarna var korrekta.

6 Resultat

6.1 Slutprodukten

6.1.1 Vägkomponenter

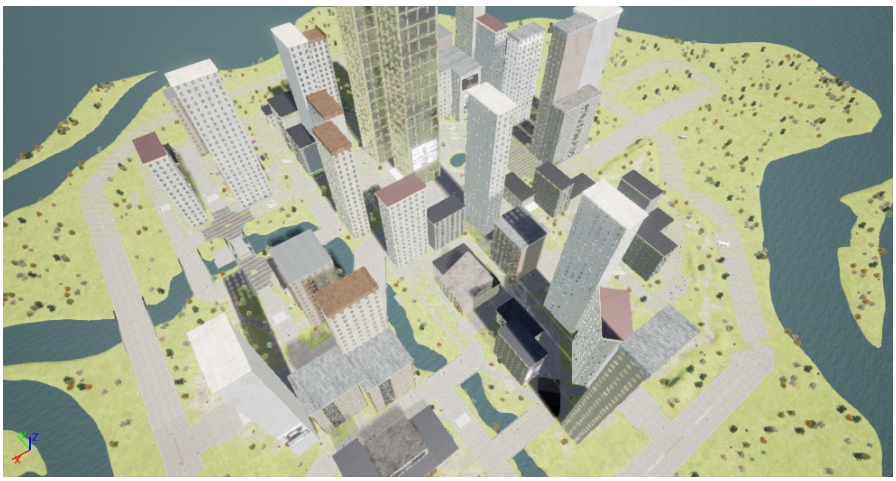
Det finns ungefär 40 olika vägkomponenter som kan vara av typen enkelfiliga, dubbelfiliga, enkelriktade, motorvägar och övergångar mellan dessa olika typer. Inom varje olika typ av väg finns det raksträckor, raksträckor med olika hastighetsgränser, 45-, 60- och 90-grader svängar, övergångsställen, rondeller och korsningar med och utan rödljus. Det har även skapats broar. Utöver detta har vi skapat 17 olika vägskyltar, exempelvis stoppskylt, väjningsplikt, huvudled och olika hastighetsskyltar.

För vägarna finns ett väl fungerande system för att placera ut raksträckor. Detta system innebär att en väg placeras ut, och därefter drar man i en så kallad *Endpoint*-vektor som avgör vart raksträckan skall sluta. Systemet placerar sedan ut så många vägbitar som behövs för att skapa en raksträcka mellan den första vägen och vektorn.

6.1.2 Stadskarta och omgivning

För att fylla omgivningen har vi konstruerat ett system för att skapa byggnader genom att sätta ihop olika byggnadskomponenter. Som utsmyckning till byggnaderna har vi även skapat affärer, en restaurang och brandstegar som placeras vid en byggnads vägg. Vidare finns det 6 olika buskar, 6 olika träd, reklamskyltar, uteserveringar, parkbänkar, gatljus och soptunnor för utsmyckning.

Stadskartan är uppbyggd med hjälp av de flesta delar som vi skapat. Kartan består av en ö, där det på ön finns en motorväg som omsluter staden. Figur 7 visar en översiktsbild där staden syns i mitten med motorvägen runt omkring den. Staden är uppbyggd av olika byggnader och utsmyckning såsom reklamskyltar, parkbänkar, gatljus och soptunnor. Utanför staden finns det utsmyckning i form av träd och buskar. Vägnätet inne i staden är uppbyggt av de olika vägkomponenterna som vi har skapat. Det finns även AI-människor och AI-bilar utplacerade på stadskartan. Människorna rör sig slumpmässigt mellan olika punkter på kartan, medan bilarna följer det nodsystem som är implementerat för vägkomponenterna.



Figur 7 – Översiktsbild av stadskartan

6.1.3 Scenarion

Vi har skapat tre enkla scenarion och tanken med dessa scenarion var endast att utgöra exempel på hur användaren kan tillverka scenarion. I de två första scenariona har vi använt oss av delar av Stadskartan, medan det tredje är skapat helt från grunden. Det har även placerats ut startplats och slutdestination som avslutar scenariona. Efter ett scenario är avslutat visas statistik upp på förarens hastighet och styrvinkel vid olika tidpunkter. I det första scenariot har vi med en AI-människa som visar hur det går att använda AI-människors event, medan det andra scenariot använder sig av AI-bilar för att visa deras event-funktionalitet. Det tredje scenariot visar hur det går att ha övrig trafik, där det finns AI-bilar som kör runt fritt enligt nodsystemet.

6.1.4 Artificiell intelligens

Vi har implementerat AI för både bilar och människor. Dels så de kan röra sig automatiskt i en karta, och dels där användaren på förhand kan bestämma exakt hur de ska röra sig. Bilarna kan placeras i en karta och kan sedan köra runt fritt i denna karta med hjälp av nodsystemet. De kommer då att köra på vägarna och följa trafikreglerna, så länge denna karta är skapad enligt användarmanualen. Människorna är implementerade så att de kan gå runt slumpmässigt på trottoarer och i naturen. När de kommer fram till ett övergångsställe och en bil är närvarande, så passerar AI-människorna efter att bilen har bromsat in och släpper förbi gångtrafikanterna.

En del buggar kvarstår hos AI:n som inte hann färdigställas innan projektet avslutades. Majoriteten av dessa buggar leder till att AI-bilarna hamnar ur kurs och ingen funktionalitet för hur bilarna ska bete sig under denna situation har implementerats. Därför kommer AI-bilarna oftast fastna vid en byggnad eller vägskylt om de hamnar ur kurs.

6.1.5 Bil med realistisk fysik

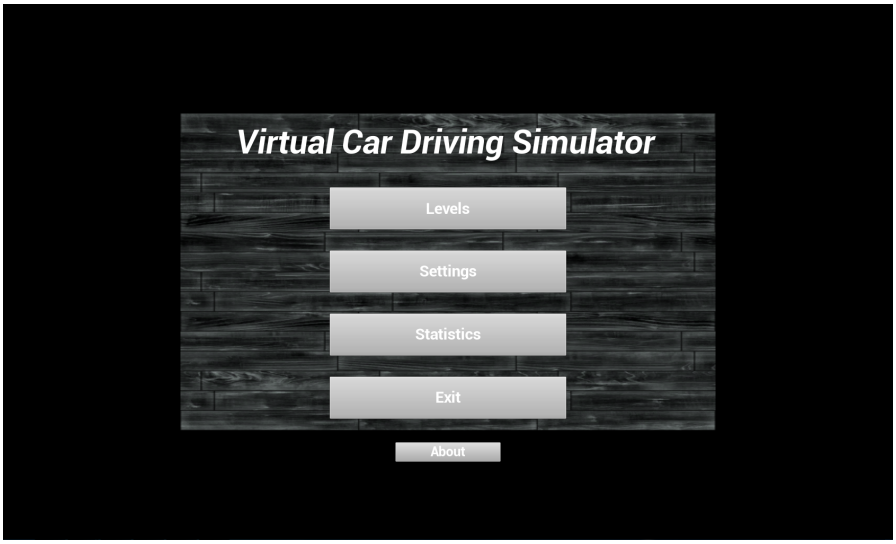
Vi har skapat en fullt fungerande bil som har bakhjulsdrift och har en styrvinkel på 60 grader, delvis genom att bygga på en befintlig bilmall. Dess instrumentpanel består av hastighetsmätare och varvmätare. Den har ett fullständigt chassi förutom rutor, vindrutetorkare och registreringsskylt. Vissa av dess fysiska egenskaper blev inte fullt realistiska. Exempelvis kan bilen snurra runt vid hög hastighet om föraren svänger för hastigt.

6.1.6 Implementation av hårdvara

Vi har implementerat stöd för OSVR, Oculus Rift DK1 och DK2 i vårt projekt som möjliga VR-HMD:s. Även för ratt och pedaler har vi implementerat stöd för den hårdvara som vi hade tillgänglig för testning, nämligen Thrustmaster Ferarri 458 Italia. Det finns dock vissa begränsningar med hårdvaran, exempelvis att man ser pixlarna i VR-HMD:n, att rattutslaget är endast ca 100 grader åt varje håll, att det saknas kopplingspedal och växelspak.

6.1.7 Grafiskt gränssnitt

Vi har skapat ett grafiskt användargränssnitt där användaren kan navigera i menyerna med hjälp av muspekaren. I huvudmenyn finns fem olika menyknappar; Levels, Settings, Statistics, Exit och About vilka illustreras i figuren nedan.



Figur 8 – Bilden är tagen i simulatorns huvudmeny och visar de olika menyalternativen i denna.

I Levels-menyn går det att välja bland de olika kartorna som finns skapade. Genom Settings-menyn går det att ändra inställningar, bl.a. förarens höjd i simulatorn. I Statistics-menyn går det att ändra var filen med diverse statistik ska sparas. Exit avslutar helt enkelt applikationen. About-menyn visar information om projektet, vilka som skapat simulatorn, varför vi skapade simulatorn och särskilda tack. Under pågående simulation går det att nå en pausmeny via Escape-knappen på tangentbordet i vilken användaren kan välja att gå tillbaka till huvudmenyn eller fortsätta simulationen. Simulatorn kommer även ihåg inställningar från tidigare körsessioner genom användande av konfigurationsfiler.

6.2 Virtual Reality och prestanda

Virtual Reality fungerar i simulatorn och vi har testat tre olika HMD:s som alla stöds. VR används enbart under simulationen och är avstängt i menyerna.

Prestandan i simulatorn, när VR inte är aktiverat, är relativt hög, och på en dator med bra grafikkort, processor och arbetsminne körs simulatorn utan några större prestandaproblem. Den mest krävande delen av simulatorn är Stadskartan på grund av sin storlek och komplexitet. Om användaren tillverkar egna scenarion går det att själv avgöra hur komplex kartan skall vara, vilket också direkt påverkar hur prestandan vid körning kommer att bli. Alla objekt som används i Stadskartan är inte helt perfekta när det kommer till prestandan, och oftast är det ljussättning och skuggning som är det största problemet. De objekt som har problem med skuggningen är idag vägskyltar, trafikljus, gatlyktor, träd och buskar. Då detta är ett känt problem och som inte hinns med att åtgärdas innan deadline är skuggor för dessa objekt inaktiverade i simulatorn tills vidare.

När VR däremot är aktiverat, uppstår stora prestandaproblem vilket resulterar i en bilduppdateringsfrekvens, FPS, som är väldigt låg. Denna låga FPS resulterar i en näst till olidlig upplevelse för användare när de kör i Stadskartan, men upplevelsen är något bättre när mindre detaljerade scenarion körs istället.

För att underlätta för framtida simulatoranvändare med prestandarelaterade problem gjordes under projektet en prestandaanalys, vilket var en serie tester för att identifiera eventuella prestandaproblem. Resultatet av dessa tester visar på vad i stadskartan som är prestandamässigt tungt under körning av simulatorn.

Resultatet ger också antydningar om vad det är som är flaskhalsen för ett givet objekt, till exempel om det är renderingen på grafikkortet eller beräkningar som är problemet. Framtida användare kan med hjälp av denna analys avgöra vad som är bra att ta bort ur världen, eller göra om, för att optimera prestandan.

6.3 Användartester och Cybersickness

Vi utförde ett fåtal användartester efter att simulatorn tillverkats. På grund av tidsbrist var dessa personer studenter på Chalmers tekniska högskola och alltså inte personer från vår primära målgrupp. Testpersonerna hade olika mängd tidigare erfarenhet av VR. Resultatet av användartesterna var att många ansåg att prestandan var för låg för en bra körupplevelse och de flesta påverkades i olika grad av Cybersickness.

6.4 Verklighetstrogenhet

Den realistiska känslan, både i Stadskartan och för de scenarion som skapats, blir bättre tack vare detaljrikedomen och estetiskt tilltalande grafik. I vår Stadskarta har vi lagt till så många detaljer som möjligt för att ge illusionen av att det är en riktig stad där människor lever, där det finns byggnader, affärer, restauranger, parkbänkar, papperskorgar och reklampelare. Terrängen i staden är varierande och ihop med naturen skapar det en ökad verklighetstrogenhet. En annan viktig detalj är den detaljerande och tilltalande bilen. Helhetsintrycket blir att det känns ungefär som en riktig stadskörning.

6.5 Moduläritet

Tack vare ett system vi tillverkat, där bilens komponenter är gjorda av olika del-meshes i 3D-modelleringsprogrammet som använts, är komponenterna i bilen enkla att flytta på och förändra. Detta ger bilen en moduläritet och gör det enkelt att redigera den. Vägkomponenterna är skapade med moduläritet i åtanke, och denna moduläritet uppnås tack vare att vägkomponenterna består av små vägbitar som på ett smidigt sätt går att koppla ihop med varandra. Att vägbitarna också innehåller förprogrammerade AI-beteenden ökar också moduläriteten avsevärt. Därför blir också skapandet av ett scenario väldigt lätt då det i princip bara är att lägga ut vägkomponenterna och sedan koppla ihop dessa. Tack vare att små komponenter pusslas ihop för att skapa en stor byggnad, det enkla system som används för att tillverka ny natur samt Unreal Engines inbyggda funktioner för att placera ut objekt fick slutligen omgivningen en hög moduläritet. Det grafiska gränssnittets välarbetade grund gör också att simulatorns inställningar och statistik är modulära och enkla att förändra.

6.6 Storlek och pris

Storleken på hårdvaran som används till simulatorn är liten, då både VR-HMD:t, ratt och pedaler är relativt små till storleken. Det är enkelt att montera ner hårdvaran och förvara den i ett skåp eller liknande. Priset på hårdvaran, exklusive den dator som används, är också låg i jämförelse med de simulatorer vi tittade på under förstudien. Uppskattningsvis är priset 5000-20 0000 kronor beroende på kvaliteten av den hårdvara som väljs.

6.7 Användarstöd

För att underlätta för vår framtida användare att sätta sig in i vårt projekt skapades, som tidigare nämnt, en användarmanual, se Bilaga A. Användarmanualen beskriver i detalj hur simulatorn installeras, vad som krävs för att köra simulatorn, hur användaren kommer igång och hur användaren kan börja redigera olika typer av element.

7 Diskussion

7.1 Slutprodukten

7.1.1 Vägkomponenter

Vägkomponenterna är vi överlag nöjda med. Vi anser att vi lyckades få bra variation på de olika vägkomponenter som vi skapade. Under projektets gång har vi itererat och uppdaterat vägkomponenterna många gånger. Vi har upptäckt nya saker som behövs finnas på varje komponent och då har vi fått implementera detta på alla komponenter. Detta hade kunnat utföras effektivare om vi från början tänkt genom allt som skulle finnas med på vägkomponenterna. Samtidigt är det väldigt svårt att veta detta från början. Exempelvis upptäckte vi efter ett tag att det var väldigt jobbigt att koppla ihop alla AI-noder med varandra på varje vägkomponent vi placerade ut i en karta. Då fick vi implementera detta i vägkomponenterna istället.

7.1.2 Stadskarta och omgivning

Stadskartan, den karta där det skulle ingå stadstrafik, landstrafik och motorvägar, blev inte riktigt så bra som vi hoppades på. Grafiskt sett ser den bra ut, men funktionaliteten nådde inte upp till våra mål. Dels har vi märkt att det är svårt att vara stadsarkitekt och verkligen skapa en stad som är väl genomtänkt och med bra placeringar av vägar så att blir ett bra flöde i trafiken. Detta leder till att det blir lite konstiga kombinationer av olika vägdelar. Sen har vi även märkt att det är väldigt svårt att bygga en relativt stor karta där allt inom AI ska fungera utan problem. Ju större kartan är desto fler punkter är det som någonting kan gå fel med våra AI-människor, men framförallt med AI-bilarna. Om någonting går fel med AI-bilarna, att de missar en nod de skulle köra till eller att de inte riktigt hinner stanna i en korsning, är det ganska stor risk att detta påverkar andra AI-bilar. Detta är ett problem med Stadskartan, att det finns risk för kedjereaktioner från vår AI som i värsta fall leder till att man får starta om stadskartan.

På grund av att vi i ett tidigt skede placerade ut vägkomponenterna på stadskartan var vi tvungen att lägga om många av dessa vägkomponenter när vi upptäckte stora problem gällande vägbitarna. Ett exempel på ett sådant stort problem var när vi gjorde förändringar i hur AI-noderna fungerade. Alla dessa uppdateringar av Stadskartan har tagit ganska mycket tid, något vi hade kunnat utföra effektivare om vi exempelvis väntat med att implementera Stadskartan till ett senare skede i projektet.

7.1.3 Scenarion

Vi är nöjda med hur våra olika scenarion blev. Vi tycker vi fick bra blandning av scenarion, där scenario 1 och 2 bygger på vår stadskarta och scenario 3 är skapat från grunden. Då vi skapade scenario 3 helt från grunden kunde vi även verifiera att själva processen för att skapa scenarion faktiskt fungerar. Samtidigt som vi genom att skapa scenario 1 och 2 från vår Stadskarta visade hur det går att skapa ett scenario med fin grafisk omgivning utan att behöva lägga så mycket tid på det. Vi anser även att vi på ett bra sätt visat att det går att skapa scenarion med olika nivåer av AI implementerad. De två första scenariona är minimala i avseende med AI medan det tredje scenariot innehåller både AI-bilar och AI-människor, där AI-bilarna kör runt helt fritt.

7.1.4 Artificiell intelligens

Att implementera AI i bilsimulatorn var för oss en utmaning, speciellt underskattade vi hur komplexa AI-bilarna behövde vara. De kunde inte använda sig av Unreal Engines inbyggda navigeringssystem med Navmeshes på grund av att bilarna har gas och styrvinkel som input istället för de automatiska navigeringsfunktionerna som utnyttjar Navmeshes. Detta försvårade arbetet att kunna förflytta bilen till en given

position. AI-bilarna är heller inte helt konsekventa i sitt beteende under tillståndet *FREESTYLE*. Det finns nämligen en sannolikhet att bilen hamnar ur kurs och fastnar. Detta kan undvikas genom att inte ha för mycket trafik på vägarna och inte ge världen eller scenariot en alltför komplex design. Till exempel bör inte flera svängar och korsningar användas i rad, utan det bör skapas utrymme mellan dessa blueprints med hjälp av raksträckor. Vi kom också fram till att vår egna bilmodell har en större sannolikhet att fastna än den andra bilmodellen från UE4:s Starter Content. Dessutom skapar AI-bilarnas styrfunktion *Steeringhandler* instabilitet vid för långt avstånd mellan två noder, vilket skapar stor risk för AI-bilarna att köra av vägen. Detta kan lösas genom att dela upp extremt långa raksträckor i flera mindre raksträckor.

AI-människornas gränser för vart de inte har tillåtelse att gå, till exempel på vägar, designades om flera gånger. Men den slutliga implementationen av AI-människorna var väl fungerande.

AI-bilarnas och AI-människornas möjlighet att utföra events för uppbyggnad av scenarion blev en lyckad implementation utan några nämnvärda buggar. Det finns mycket funktionalitet och möjlighet till att kombinera dessa eventnoder med varandra och vi har själva inte kommit på alla tänkbara sätt de kan byggas med i scenarion. Längden på slingorna som bilarna ska köra i ett scenario är oftast mycket kortare än i vår Stadskarta, och därför löper AI-bilarna mycket mindre risk att fastna och har ett mer konsekvent beteende. Ett av kraven från forskarna på HMI-avdelningen var att scenarion ska vara konsekventa och att bilarna ska göra samma saker varje gång scenariot simuleras. Vi anser att AI-bilarna är tillräckligt konsekventa för detta ändamål så länge vägarna i scenariot inte har en allt för svår och komplex design, som till exempel förekommer på vissa ställen i Stadskartan.

Sammanfattningsvis tycker vi att AI-människorna, AI-bilarna och dess förmåga att både kunna köra runt slumpmässigt i en karta och kunna utföra förprogrammerade events, är väl implementerade. Med undantaget att sannolikheten finns att AI-bilarna kan fastna om det förekommer för mycket trafik och inte tillräckligt med utrymme mellan vägkomponenter som inte är raksträckor.

7.1.5 Bil med realistisk fysik

Realistisk fysik för bilen lyckades vi till viss del att åstadkomma. Mycket beroende på att det redan finns inbyggt stöd för bil-klasser i Unreal Engine. Eftersom vi utgick från en bilmall, fanns funktioner för hur en bil gasar, bromsar och hur däcken skall bete sig när man svänger, vilket underlättade arbetet en del. Mycket av vårt arbete för att skapa realistisk fysik för bilen var justering av parametrar såsom däck-friktion, svängradie för däcken och vikten på bilen för att ge en så bra och realistisk körkänsla som möjligt. Det finns fortfarande en hel del finslipning som skulle behöva göras, men vi anser att vi har kommit en god bit på väg och är nöjda med resultatet.

7.1.6 Implementation av hårdvara

Vi anser att vi har lyckats bra med implementationen av hårdvaran. De begränsningar som vi nämnde i kapitel 6, att man ser pixlarna i VR-headsetet, rattutslaget är endast ca 100 grader åt varje håll och att det saknas kopplingspedal, är helt beroende av hårdvaran i sig och inte hur den har implementerats.

7.1.7 Grafiskt gränssnitt

Det grafiska användargränssnittet vi skapat fyller sin funktion men går tyvärr ej att använda tillsammans med VR eftersom de är tvådimensionella vilket gör att vi fått slå av VR inuti menyerna. Det förtar känslan och inlevelsen av simuleringen men har fördelen att det är naturligt att navigera för användaren då de kan navigeras med datormus. Eftersom gränssnittet inte var ett fokus i projektet så får lösningen ändå anses

vara helt acceptabel. Därutöver är grunddesignen av gränssnittet enkelt att förstå och göra förändringar i vilket leder till högre användbarhet.

7.2 Virtual Reality och prestanda

En överväldigande majoritet av de vi pratat med och låtit testa simulatorn under projektets gång har pratat om hur verklighetstroget VR känns jämfört med en vanlig bildskärm. Att kunna se sig om i världen på det sätt som VR-HMD:s möjliggör upplevs av de vi pratat med, och av oss själva, öka verklighetstrogenheten avsevärt. Vi anser därför att VR bidrar väldigt mycket till simulatorupplevelsen, och jämfört med den simulator Projektgruppen testat tidigare anser vi att VR-tekniken tillför en helt ny dimension som gör användaren mer övertygad om att simuleringen är på riktigt.

Prestandan i vår slutprodukt är ett stort problem och det påverkar körupplevelsen negativt. Eftersom Projektgruppen inte har en bakgrund eller utbildning inom spelutveckling, uppstår därför många prestandaproblem som en erfaren spelutvecklare möjligen hade kunnat undvika. Den dåliga prestandan gör också att det är svårt att helt och hållet utvärdera VR:s potential i framtida testprocesser inom bilindustrin. Vi tycker också att dålig upplösning och dålig avståndsbedömning i de VR-HMD:s vi har använt påverkar prestandan negativt.

Trots att simulatorn har tunga prestandarelaterade problem när man använder VR, och att det gör det lite svårare att bedöma VR:s framtida potential, anser vi ändå att den ökade effekten av verklighetstrogenhet som VR bidrar med är stor nog för att kunna dra slutsatsen att VR har en stor potential inom framtida testprocesser. Förhoppningsvis kommer utvecklingen av nya VR-HMD:s motverka, eller till och med eliminera, dessa problemen med hårdvaran. Om simulatorn vidareutvecklas i framtiden skulle man också kunna eliminera de problem som skapats på grund av vår oerfarenhet av spelutveckling.

7.3 Användartester och Cybersickness

Användartesterna kunde ha gjorts bättre eftersom antalet tester vi utförde var allt för få och eftersom vi inte utförde testerna med personer från målgruppen, utan istället studenter från Chalmers. På grund av dessa brister anser vi att resultaten från användartesterna inte är helt tillförlitliga. Men vi ser ändå en trend att simulatorn i olika hög grad orsakar Cybersickness bland användarna. Detta beror sannolikt främst på problem med prestandan. När simulatorn används med VR aktiverat så blir det för låg FPS, vilket lätt leder till Cybersickness [24]. Användartesterna som utfördes låg även tidsmässigt relativt sent i projektet, vilket gjorde det svårt att hinna justera och åtgärda simulatorn beroende på den feedback vi fick från användartesterna. Exempelvis hade en åtgärd kunnat vara att försöka öka prestandan ännu mer för att på så sätt förhoppningsvis minska Cybersickness.

7.4 Verklighetstrogenhet

Bilsimulatorns verklighetstrogenhet upplever vi som för det mesta bra, men det finns områden som behövs förbättras. För att det ska upplevas helt realistiskt är det många olika delar som ska fungera. Några delar fungerar bättre och andra fungerar lite sämre.

En av sakerna som fungerar lite sämre anser vi vara själva VR-headsetet. Detta upplevs som för lågupplöst och det är svårt att se tydligt när man kör. Detta gör det svårt att se framåt på vägen och kunna planera sin körning, vilket drar ner den realistiska känslan. Samtidigt tillför VR-headsetet ganska mycket till upplevd realistisk känsla bara genom att föraren tror att denne befinner sig i bilen, tack vare VR-tekniken. Så ett bättre VR-headset hade troligtvis förbättrat denna punkt.

Bilens köregenskaper kan i vissa situationer ta bort från den realistiska känslan. Exempelvis om föraren kör i ganska hög hastighet och gör en skarp sväng så kan bilen snurra runt, vilket inte känns helt realistiskt. För det mesta upplevs bilen bra att köra, men detta är ett område som skulle behöva lite mer finjustering för att få till en ännu mer realistisk känsla.

Grafiken i simulatorn anser vi bidra till en realistisk känsla. Omgivningen är varierande och framförallt i Stadskartan finns det många detaljer som höjer realismen. Även bilen som föraren kör är verklighetstrogen rent grafiskt, både invändigt och utvändigt. Då AI-bilarna använder sig av denna modell i olika färger känns det som verkliga bilar som föraren möter i trafiken.

AI-bilarna anser vi höja den realistiska känslan i de flesta situationer då de för det mesta fungerar väldigt bra. När de däremot inte fungerar, om de exempelvis missar en nod som de skulle köra till, kan det upplevas som väldigt konstigt då en AI-bil helt plötsligt kan köra av vägen för att passera denna nod.

7.5 Moduläritet

Vikten i moduläritet grundar sig i, som tidigare nämnt, att simulatorn skall kunna vara enkel att modifiera så att det passar användarens behov, och detta är något Projektgruppen känner sig väldigt nöjda med. Detta beror bland annat på valet av grafikmotor, eftersom Unreal Engine från början erbjuder en hög grad av moduläritet och utbytbarhet av modeller, texturer, material med mera. Att objekten som används ofta skapas i 3D-modelleringsprogram gör också att de är enkla att redigera. Grafikmotorn, 3D-modelleringsprogrammen tillsammans med de väl fungerande systemen för att skapa nya vägnät och byggnader gör att simulatorn är enkel att förändra, vilket gör att vi kan känna att vi nådde upp till målet gällande moduläriteten.

Det finns dock en tydlig nackdel gällande moduläriteten, och det är att användaren behöver ha god kännedom av Unreal Engine samt modelleringsprogram, så som Autodesk Maya eller Blender, för att kunna göra ändringar i simulatorn. Optimalt vore om det var så enkelt att ändra i simulatorn så att du inte behövde vara en avancerad datoranvändare, men detta ligger utanför vår omfattning och det var aldrig relevant att under projektets gång utveckla ett sådant system på grund av dess komplexitet och vår begränsade tid.

7.6 Användarstöd

Detta är ett ganska omfattande projekt där det finns väldigt mycket att sätta sig in i. Vårt mål har hela tiden varit att göra det så lätt som möjligt för framtida användare att använda vår simulator, men även att vidareutveckla den och lägga till nya funktioner. Vi anser att användarmanualen blev väldigt bra där vi på ett systematiskt sätt går genom vad som krävs för att använda vår simulator och hur den ska användas. För att förtydliga vad vi menar har vi valt att inkludera många bilder som visar exakt hur våra olika komponenter ska användas.

Vi tror att det är ganska lätt för målgruppen att börja använda vårt projekt och att simulatorn fungerar bra i deras syfte. Att exempelvis skapa ett scenario med diverse AI är en ganska rättfram process och borde inte ta lång tid om användarmanualen används. Eftersom det inte krävs någon kunskap i hur de olika delarna fungerar, bara hur de används. Däremot att vidareutveckla vårt projekt och lägga till egna funktioner kräver djupare kunskap, både i hur vårt projekt är implementerat och hur UE4 fungerar. Att vidareutveckla vårt projekt är dock något som vi har haft i åtanke under tiden vi har arbetat med det, så det borde inte vara några större problem för framtida användare.

7.7 Fortsatt utveckling

Som tidigare nämnt, så var det tänkt att bilsimulatorn skulle bli en stabil grund för vidareutveckling, och därmed finns det enorma möjligheter att arbeta vidare på detta projekt. Allt kan i princip förbättras och optimeras, bland annat AI:ns beteenden och grafikmässig prestanda.

Leap Motion är ett företag som tillverkar en produkt, Leap Motion Controller, som med hjälp av en IR-kamera kan spåra användarens händer för att sedan användas som input till olika program. Händerna kan då genereras visuellt i en VR-miljö. Vid fortsatt utveckling kan stöd för Leap Motion läggas till i simulatorn. Att föraren kan se sina händer i bilen med hjälp av Leap Motion skulle kunna öka verklighetstrogenheten, till exempel genom att det öppnar upp möjligheter för interaktion mellan föraren och eventuella HMI-system i förarkabinen. Exempel på HMI-system som föraren då skulle kunna interagera med är radio, möjlighet att veva ner glasrutorna, justera backspeglarna och en virtuell handbroms. Detta öppnar också upp för nya forskningsmöjligheter, det vill säga undersökningar om hur föraren interagerar med bilens inredning.

Förarbilen har för tillfället bara de mest väsentliga funktionaliteter i förarkabinen, såsom varvmätare och hastighetsmätare. Med Leap Motion-teknologin implementerad skulle vi kunna utveckla fler komponenter på instrumentbrädan och runt om i förarkabinen som föraren kan interagera med.

Simulatorns menysystem är implementerat med 2D-grafik. Detta resulterar i att användaren bara kan navigera i menyn utan att använda sig av VR-headset. En möjlig vidareutveckling av menysystemet är att göra om det till 3D-grafik och därmed göra menyn tillgänglig för VR-headsetet också. Om vi dessutom implementerar Leap Motion-teknologin så skulle vi kunna göra navigering tillgänglig med händerna istället för med datormusen.

Möjligheten att logga statistik under simulationen är bara en grundläggande design och det är meningen att det i framtiden ska läggas till fler parametrar som ska loggas beroende på vad slutanvändaren är ute efter. Om framtida utvecklare implementerar interaktiva komponenter i förarkabinen med Leap Motion, vilket tidigare nämnts, så öppnar detta upp möjlighet till att också logga statistik om hur människan interagerar med dessa komponenter.

Växelspak var en diskussion som kom upp tidigt i projektet. Dock märkte vi att utbudet på elektroniska växelspakar med USB-uttag var begränsat och vi valde att skjuta upp det tills vidare. Vid vidareutveckling av projektet är det rimligt att ta upp denna diskussion igen. Växelspak är en av de mest essentiella komponenterna i en bil som vi har valt att inte implementera. Genom att implementera en fysisk växelspak till simulatorn anser vi att körupplevelsen skulle bli mer komplett.

7.8 Ekologisk hållbarhet

Även om hållbarhet inte har varit ett stort fokus i projektet är det ändå något viktigt som bör beröras. Vi anser att det här projektet, och dess resulterande simulator, har en hög ekologisk hållbarhet eftersom vår slutprodukt använder hårdvara som är mindre i storlek och har en lägre energiförbrukning än de simulatorer vi har varit och tittat på. Då jämför vi framförallt vår simulator med den mångmiljonsimulator vi var och tittade på, som bestod av en flera ton tung fordonskaross på en stor hydraulikställning.

7.9 Utvecklingsverktyg

7.9.1 Unreal Engine 4

Det var med hjälp av UE4 som menyerna, AI, spellogiken, alla vägkomponenter och alla kartor skapades. Då UE4 har väldigt bra inbyggt stöd för dessa delar var det inte så tidskrävande som man skulle kunna tro. Oftast fanns det väldigt bra utvecklingsverktyg som vi kunde använda, till exempel för att skapa menyer och AI. I vissa fall fanns det till och med en färdig grund som vi kunde använda och arbeta vidare på så att det skulle passa just vårt syfte. Så var fallet med bilen vi använder oss av där mycket av funktionaliteten redan fanns. Det var implementerat hur den gasade, att den visade bilens hastighet och att den visade bilens växel. Sedan har vi ändrat själva 3D-modellen av bilen till den vi själva skapat och även finjusterat dess svängradie och vikt, allt för att det ska bli så verklighetstroget som möjligt. Unreal Engine fungerade väl i vårt syfte, även om det finns en del mindre problem med spelmotorn som förhoppningsvis förbättras framöver.

7.9.2 Blender, Autodesk Maya och GIMP

Under projektets gång har vi, som tidigare nämnts, använt 3D-modelleringsprogrammen Blender och Autodesk Maya. Ingen i projektgruppen hade någon tidigare erfarenhet av dessa program vilket ledde till att modellering tog lång tid i början av projektet. Lyckligtvis fanns en mycket hjälpsam användarbas som gjorde att lösningar på de problem vi stötte på oftast gick att hitta på olika forum. Då modelleringsprogram valdes på preferensbasis resulterade detta i att två personer i projektgruppen valde att modellera byggnader och omgivning i Blender medan en person valde att göra bilen i Autodesk Maya. Eftersom båda programmen stödjer ett filformat som gör det enkelt att importera modellerna till Unreal Engine stötte vi inte på många implementationsmässiga problem. I efterhand inser vi dock att det antagligen varit bättre att använda oss av samma program. Det hade blivit lättare att hjälpa varandra, inlärningsperioden hade antagligen kortats ned och hela modelleringsprocessen hade gått snabbare.

GIMP, som användes för att redigera texturer, fungerade felfritt vilket vi är väldigt nöjda med.

7.9.3 Programspråk

Eftersom användandet av Blueprint Visual Scripting fungerade så väl i Unreal Engine, behövde vi aldrig använda oss av C++. Detta var väldigt fördelaktigt eftersom det resulterade i att vi inte behövde lära oss ordentligt om hur C++ fungerar, och därav sparade mycket tid.

7.9.4 Versionshantering

Fördelarna vi har fått ut av att använda versionshantering är att det har varit relativt smidigt för oss att hela tiden kunna arbeta med den senaste versionen av vårt projekt. Alla i projektgruppen har även på ett överskådligt sätt kunnat se vilka filer som de övriga i projektgruppen arbetat med. Då det är binärfiler man arbetar med i UE4 så har vi dock inte kunnat arbeta parallellt med samma filer, vilket vanligtvis är en av de största fördelarna med versionshantering. Git klarar helt enkelt inte av att sammanfoga två olika binärfiler, utan man blir tvungen att välja en av dessa. Detta har lett till att om två personer arbetat med samma binärfil parallellt har den enas jobb gått mer eller mindre helt förlorat.

8 Slutsats

Syftet med det här kandidatarbetet var att undersöka vilken påverkan Virtual Reality kan ha på bilindustrins testprocess, och huruvida den moderna VR-tekniken kan användas för att skapa mindre och billigare simulatorer. Under projektets gång har vi tillverkat en bilsimulator innehållandes komplexa vägnät, byggnader, natur, avancerade AI-människor och AI-bilar, en detaljerad förarbil samt en rad andra detaljer. En simulator som exempelvis har stöd för OSVR eller Oculus Rift DK2 som VR-headset, och Thrustmaster Ferarri 458 Italia som ratt och pedaler. Vi visade, med hjälp av en förstudie, att målgruppen efterfrågar en modulär, verklighetstrogen och grafiskt tilltalande simulator, även om de inte ansåg grafiken vara det absolut viktigaste.

Simulatorn vi har skapat är avsevärt billigare och mindre i storlek än de simulatorer vi var och tittade på vid projektets start. Simulatorn har en hög moduläritet eftersom vi har haft moduläritet i åtanke när de olika komponenterna tillverkats. Tack vare alla de detaljerade komponenter som skapats, men främst tack vare VR, så uppnår simulatorn även en hög verklighetstrogenhet. Framförallt eftersom VR i sig bidrar till en effekt som gör att användare känner sig som att de befinner sig i den digitala världen.

Dock har vår simulator stora problem när det kommer till prestandan, både problem som beror på att simulatorn inte är optimerad till fullo, men också problem som uppkommer på grund av exempelvis låg upplösning i VR-HMD:s. Många av användarna upplever olika grad av Cybersickness, och det tror vi beror på de prestandaproblem som idag finns.

Prestandaproblemen gör det också svårare att avgöra vad VR har för potential i framtidens testprocesser inom bilindustrin. Men tack vare den fantastiska känsla av verklighetstrogenhet som VR bidrar med så anser vi att Virtual Reality har en stor potential inom bilars framtida testprocess. Vår målgrupp kommer sannolikt att ha en stor nytta av den här tekniken. Vi tror framförallt att forskarna på interaktionsdesignsavdelningen på Chalmers kommer att ha nytta av vår simulator, om de har tid att arbeta bort de problem som idag finns.

Referenser

[1] theverge.com, *'Nintendo Virtual Boy'*. [Online]. Tillgänglig: http://www.theverge.com/products/virtual-boy/1672. [Hämtad: 25/5 2016].

[2] C. Kohler, *'Virtual Boy, Nintendo's Big 3-D Flop, Turns 15'*, 2010. [Online]. Tillgänglig: http://www.wired.com/2010/08/virtual-boy/. [Hämtad: 25/5 2016].

[3] D. McFerran, *'Feature: The Making of the Nintendo Virtual Boy'*, 2010. [Online]. Tillgänglig: http://www.nintendolife.com/news/2010/03/feature_the_making_of_the_nintendo_virtual_boy. [Hämtad: 25/5 2016].

[4] F. Nelson, M. Yam, *'The Past, Present, And Future Of VR And AR: The Pioneers Speak'*, 2014. [Online]. Tillgänglig: http://www.tomshardware.com/reviews/ar-vr-technology-discussion,3811-2.html. [Hämtad: 25/5 2016].

[5] R. Amadeo, *'Oculus Rift DK2 includes the entire screen assembly from a Galaxy Note 3'*, 2014. [Online]. Tillgänglig: http://arstechnica.com/gaming/2014/07/oculus-rift-dk2-includes-the-entire-screen-assembly-from-a-galaxy-note-3/. [Hämtad: 25/5 2016].

[6] H. Winchester, *'Best VR games 2016: The titles set to blow your mind in the next 12 months'*, 2015. [Online]. Tillgänglig: http://www.wareable.com/gaming/top-vr-games-for-oculus-rift-project-morpheus-gear-vr-and-project-cardboard. [Hämtad: 25/5 2016].

[7] A. Rizzo, G. J. Kim, *'A SWOT Analysis of the Field of Virtual Reality Rehabilitation and Therapy'*, In Presence, vol. 14, no. 2, pp. 119-146, April 2005.

[8] opendrive.org, *'OpenDrive, managing the road ahead'* [Online]. Tillgänglig: http://www.opendrive.org/index.html. [Hämtad: 25/5 2016].

[9] engadget.com, *'The sights and scents of the Sensorama Simulator'*, 2014. [Online]. Tillgänglig: http://www.engadget.com/2014/02/16/morton-heiligs-sensorama-simulator/. [Hämtad: 25/5 2016].

[10] F. Steinicke, G. Bruder, S. Kuhl, P. Willemsen, M. Lappe and K. Hinrichs, *'Natural Perspective Projections for Head-Mounted Displays'*, In IEEE Transactions on Visualization and Computer Graphics, vol. 17, no. 7, pp. 888-899, July 2011.

[11] visbox.com, *'Cave Automatic Virtual Environment'*. [Online]. Tillgänglig: http://www.visbox.com/products/cave/. [Hämtad: 25/5 2016].

[12] J. Walton, *'Evaluating VR performance and latency with Futuremark's VRMark'*, 2016. [Online]. Tillgänglig: http://www.pcgamer.com/evaluating-vr-performance-and-latency-with-futuremarks-vrmark/. [Hämtad: 25/5 2016].

[13] riftinfo.com, *'Oculus Rift History – How it All Started'*, 2015. [Online]. Tillgänglig: http://riftinfo.com/oculus-rift-history-how-it-all-started. [Hämtad: 25/5 2016].

[14] ifixit.com, *'Nintendo Virtual Boy Teardown'*, 2010. [Online]. Tillgänglig: https://www.ifixit.com/Teardown/Nintendo+Virtual+Boy+Teardown/3540 [Hämtad: 25/5 2016].

[15] digitaltrends.com, *'Spec Comparison: The Rift is less expensive than the Vive, but is it a better value?'*, 2016. [Online]. Tillgänglig: http://www.digitaltrends.com/virtual-reality/oculus-rift-vs-htc-vive/ [Hämtad: 25/5 2016].

[16] T. Nguyen, *'Estimating distances and traveled distances in virtual and real environments'*, 2011.

[17] D. Waller, A. R. Richardsson. *'Correcting distance estimates by interacting with immersive virtual environments: effects of task and available sensory information.'*, In Journal of Experimental Psychology: Applied, Vol. 14 Issue 1, pp. 61-72, Mar. 2008.

[18] F. Ahmed, J. D. Cohen, K. S. Binder and C. L. Fennema, *'Influence of tactile feedback and presence on egocentric distance perception in virtual environments'*, In Proceedings of the 2010 IEEE Virtual Reality Conference (VR), Waltham, MA, 2010, pp. 195-202.

[19] J. M. Knapp and J. M. Loomis, *'Limited Field of View of Head-Mounted Displays Is Not the Cause of Distance Underestimation in Virtual Environments'*, In Presence, vol. 13, no. 5, pp. 572-577, Oct. 2004.

[20] B. Bodenheimer, J. Meng, H. Wu, G. et al, *'Distance Estimation in Virtual and Real Environments using Bisection'*, In Proceedings of the 4th Symposium on Applied Perception in Graphics and Visualization (APGV'07), Tübingen, 2007, pp 35–40.

[21] R. M. Doctor , *'Cybersickness Affects Up To 80 Percent Of Smartphone, Tablet Users: Things To Know About This Digital Motion Illness'*, 2016. [Online]. Tillgänglig: http://www.techtimes.com/articles/107934/20151118/cybersickness-affects-up-to-80-percent-of-smartphone-tablet-users-things-to-know-about-this-digital-motion-illness.htm. [Hämtad: 25/5 2016].

[22] J. LaViola Jr, *'A discussion of cybersickness in virtual environments'*. SIGCHI Bull. 32, 2000, pp 47-56.

[23] T. Lewis, *'When Will Virtual-Reality Headsets Stop Making People Sick?'*, 2015. [Online]. Tillgänglig: http://www.livescience.com/50129-virtual-reality-nausea-sickness.html. [Hämtad: 25/5 2016].

[24] S. Kawamura and R. Kijima, *'Effect of HMD latency on human stability during quiescent standing on one foot'*, 2016 IEEE Symposium on 3D User Interfaces (3DUI), Greenville, SC, 2016, pp. 141-144.

[25] unrealengine.com, *'Unreal Engine FAQ'*. [Online]. Tillgänglig: https://www.unrealengine.com/faq. [Hämtad: 25/5 2016].

[26] unrealengine.com, *'VR is the future'*. [Online]. Tillgänglig: https://www.unrealengine.com/vr-page. [Hämtad: 25/5 2016].

[27] unrealengine.com, *'Animation Blueprints'*. [Online]. Tillgänglig: https://docs.unrealengine.com/latest/INT/Engine/Animation/AnimBlueprints/index.html. [Hämtad: 25/5 2016].

[28] S. Bruck and P. A. Watters, *'Estimating Cybersickness of Simulated Motion Using the Simulator Sickness Questionnaire (SSQ): A Controlled Study,'* In Proceedings of Computer Graphics, Imaging and Visualization, 2009. CGIV '09. Sixth International Conference on, Tianjin, 2009, pp. 486-488.

Bilagor

Översikt över rapportens bilagor:

— Bilaga A: Sammanfattning av användarmanual
— Bilaga B: Sammanfattning av intervjuer och telefonsamtal utförda i samband med förstudien
— Bilaga C: Förstudiens intervjumall
— Bilaga D: Blankett för användartester

Bilaga A: Sammanfattning av användarmanual

Användarmanualen i sin helhet finns här:

https://github.com/sveningsonrobin/VRDrivingSimulator/tree/master/Documentation

1. Hårdvarukrav

Detta kapitel beskriver vad användaren behöver för hårdvara för att komma igång med simulatorn. Den beskriver vilka systemkrav användarens dator behöver ha, men även hur användaren får igång ratt & pedaler samt VR.

2. Installation

Innehåller information om hur användaren laddar ner källkoden, vilka program som behövs för att köra simulatorn och hur användaren kommer igång.

3. Prestandaförbättring

Detta kapitel berättar vad användaren kan göra om användaren har problem med prestandan i simulatorn. Bland annat finns generella tips om hur prestandan kan öka utan att detaljrikedomen minskar, men det finns även information om vad i omgivningen användaren kan kapa utan att förstöra funktionaliteten.

4. Mappstruktur

Förklarar mappstrukturen i projektet, så att användaren kan få en överblick över hur alla filer är sorterade.

5. Instruktioner

Innehåller instruktioner om hur användaren kan arbeta med och förbättrar olika viktiga komponenter i simulatorn. Komponenterna är; vägar, byggnader, natur, terräng, AI-bilar & AI-människor, scenarion, parametrar, statistik och förarbilen.

6. Kända problem

Det sista kapitlet i användarmanualen beskriver vilka kända problem som existerar när Projektgruppen är klar med projektet, dvs. problem som inte hunnits åtgärdas.

Bilaga B: Sammanfattning av intervjuer och telefonsamtal utförda i samband med förstudien

Intervju 1

Personens bakgrund

Forskningsinjengör på statligt ägt forskningsinstitut. Har expertis inom bildsystem, allt från hårdvara och mjukvara med VR, bilskärmar etc. Har kopplingar till svenskt universitet, skall disputera och bli doktor. Skall forska kring bakgrunden av simulatorer.

Utbildad till civilinjengör inom medieteknik. Gjorde examensarbete på stort fordons-företag. Har varit lite konsult inom IT-branschen och har varit på forskningsinstitutet i 8 år.

Hur går processen till när ni testar en fordonsanvändares beslut i olika situationer och användarupplevelsen generellt?

Bilsimulatorer - man låter en testperson köra en "normal körning". Om det är något i bilen man testar får de göra något i gränssnitten medan de kör och så mäts olika saker. Kan spela in personen på video, logga kroppsdata, ögontracking etc. Har tre simulatorer totalt. Är nationella resurser så vem som kan hyra in sig för ~30k per dag. Stora anläggningar, typ en halv fordonskupé. Föraren sitter i kupén och har en projektorduk framför sig. Hela systemet rör sig. Hela bilens inredning finns i verkligheten, och alla komponenter är fysiska. Vissa saker är utbytbara, exempelvis instrumentklustret som är en skärm. Ofta kanse man vill montera en separat skärm (typ iPad) i instrumentpanelen.

Vad har er testprocess för brister?

Generellt är ett stort problem att folk bli åksjuka, eller simulatorsjuka. Man blir illamående på ett sätt som man inte blir i en bil. Resultaten kan påverkas av detta. Att veta att man sitter i en simulator innebär att man tänker annorlunda. Kör du in i en bil så dör du inte, det gör också att folk kanske omedvetet ändrar sitt beteende. Exempelvis när lastbilschaufförer fick köra i fyra timmar på en tråkig väg, så somnade de, även om de kanske inte hade gjort det i verkligheten. Simulatorn kan aldrig vara en riktig version av verkligheten.

Vad tror du att fordonsindustrin generellt har för brister när det kommer till denna typ av testning?

Beror lite på vad man testar. Svårt att svara på. Generella problem med illamående och att man ändrar sitt beteende. En skärm som täcker större delen av synfältet ökar mängden simulatorsjuka.

Vad krävs av en bra bilsimulator för att testa vad användaren tar för beslut i olika situationer och hur den generella användarupplevelsen är?

För att få fram resultat man kan lita på måste man ligga så nära den riktiga upplevelsen som möjligt. Varje steg bort från det är också ett steg ifrån att överföra resultaten till verkligheten. Man vill kunna validera simulatorn mot verkligheten. Det man gör i simulatorn skall kunna gå och göra i verkligheten, och sen kan man jämföra hur man agerar olika i verkligheten och simulatorn. Då kan man se en skala på hur mycket simulatorn skiljer sig. Man kanske exempelvis vrider på ratten mer i simulatorn jämfört med verkligheten.

Vad är de mest generella bristerna med dagens bilsimulatorer?

Ganska mycket tekniska problem. Bildåtergivning är personens generella område, och personen tycker att det är svårt att representera både riktigt ljusa förhållanden men också riktigt mörka. Ljusstyrkan kan inte vara likt verkligheten. Exempelvis dagsljus eller nattmörker.

Har du någon erfarenhet inom VR?

Ja, har arbetat ganska uteslutande med det de senaste tre åren. Har jobbat mycket med Oculus Rift, men inte i simulatorn, utan har tittat på den om man skulle kunna använda en VR-hjälm i en riktig bil för att kunna köra på en testbana. Tanken är att du skall kunna utnyttja fördelarna med att köra en riktig bil, men du skall också kunna simulera fordon kring om dig. Du kan köra scenarion som är för farliga för att testa i verkligheten. Du kan exempelvis ha en långtradare som kör ut fram för dig, men det hade du inte kunnat göra på en testbana. Kan orsaka skador i verkligheten.

Vad kan VR bidra med till testningen?

Du får en annan inlevelse, iom. att du får hela synfältet med. Det finns 3d-projektor-system men de når inte riktigt fram.

Vilka problem kan testningen medföra?

Fortfarande ganska låg upplösning om du jämför med en projektorbaserad lösning, deras simulator har 6 full HD projektorer och det blir ganska hög upplösning. En Occulus är sämre och du ser pixlarna. Har både devkit 1 och 2 av oculus.

Vad tycker du om vår idé?

Unreal Engine är ett bra steg, eftersom det är dumt att börja från scratch. Det är bra om vi tittar på de standarder som finns inom simulatorbranchen. Den ena heter “Open Drive” som är en standard för att beskriva hur vägen ser ut. Det är egentligen en tysk standard. “opendrive.org”. Folk från en annan

simulator kan ta med sig sin väg om vi har den standarden. Ganska mycket jobb att implementera fullt, men kan vara bra att ha i bakgrunden. Finns också “Open Scenario” som är en systerstandard, som är en öppen scenariobeskrivning så att man kan byta scenarion mellan simulatorrer. Tanken är att ett bilföretag skall kunna köra en studie hos dem, och sen vill de testa i forskningsinstitutets simulator, då skall de kunna flytta över sin simulator. De flesta simulator är custom lösningar och det är svårt att flytta lösningar mellan simulatorer.

Om det är HMI-människor som skall använda det så bör man tänka på eye tracking, finns det till VR? Kanske vi kan kolla på? Finns för “pro”-lösningar av VR, sensix gör VR-hjälmar i miljon-klassen. Kanske kan vara värt att representera händerna i simulatorn?

Finns ett franskt företag som gör körskole-simulatorer

Man kan testa mer än HMI. Bland annat trötthet, droger, alkohol och ovana förare. I Göteborg lät man barn köra i simulatorer och jämförde sedan med hur föräldrarna körde.

Något man för närvarande inte kan testa i våra simulatorer är studier som har kopplingar till en riktig bils hårdvarusystem.

Intervju 2

Bakgrund

Är forskare på statligt forskningsinstitut. Jobbar med fordonsteknik och simulering. Har tidigare jobbat 4-5 år på ett stort fordonsföretag. Där arbetade personen på uppdrag som någon form av expertkonsult. Har arbetat med flera olika saker på detta företag.

Hur går processen till när ni testar en fordonsanvändares beslut i olika situationer och användarupplevelsen generellt?

Den vanliga processen är att när processen är satt, och man vet vilket scenario man vill prova, så är processen ganska förbestämd. Man kallar in försökspersoner via de kanaler de har (en databas, annonser) till simulatorn, det kan vara olika personer beroende på vad man vill prova. Tillräckligt många personer för att få ett statistiskt underlag. Sedan utför de försöket i simulatorn och sparar all vesentlig information från de kriterier som är intressanta. Sedan drar man ett statistiskt resultat.

Vad har er testprocess för brister?

Hela konceptet med att rekrytera försökspersoner… Dels finns det inte tillräckligt med pengar för att rekrytera tillräckligt många, ofta har man typ 20 till 40 personer som skall representera den stora massan. Det är ibland inte representativa personer, ofta de personer som tycker att det är kul och intressant att vara med. En annan svaghet är den teknik som används, de använder dock inte alltid simulatorer, ibland kör man på vägar eller provbanor, men validiteten i körsimulatorn är kanske inte alltid representativ. Man kan alltid ifrågasätta resultatet. Det vanligaste sättet att mildra den effekten är att titta på relativa skillnader, tex. om man testar en sak och en annan sak och jämför skillnaden. Finns ingen snabb fix.

Vad tror du att fordonsindustrin generellt har för brister när det kommer till denna typ av testning?

Se tidigare fråga. Fordonsindustrin använder nog själva de här metoderna som vi beskrev, man gör inga produkttester i den här typen av miljö och anläggning, utan man gör förstudier för att få tillräckligt med information för att starta igång ett projekt som leder till en produkt. Denna metoden använder man i tidiga faser. Man gör det ofta i samspel med doktorandprojekt på högskolor.

Vad krävs av en bra bilsimulator för att testa vad användaren tar för beslut i olika situationer och hur den generella användarupplevelsen är?

Skulle nog säga att det beror helt på syftet, såklart, om man är ute efter en absolut validitet (då man inte jämför två fall), då har man väldigt mycket större krav på att allt återges på ett så korrekt sätt som

möjligt. Men om man mäter relativa skillnader så kan man komma iväg med bilsspelsliknande simulatorer. Det gäller att man inte kör extremt, då är rörelsesystemet viktigare. Kräver det att man drömmer sig in i sitt scenario för att bli överraskad krävs andra egenskaper, exempelvis ljud. Frågan är ett eget ämne i sig.

Vad är de mest generella bristerna med dagens bilsimulatorer?

Egentligen alla komponenter. Timing mellan vissa synintryck, att grafik inte släpar efter rörelse, och att ljud hänger med och sådär är allmäna brister man inte har så bra koll på. Det finns studier som visar att du ökar inte validiteten nödvändigtvis för att du ökar detaljrikedomen i grafiken. Vissa saker är inte så uppenbara, blir det närme verkligheten så blir det bättre. Ganska komplext system. Man är känslig för förskjutningar i synintryck.

Har du någon erfarenhet inom VR?

Genom att han är handledare till en person som doktorerar. Annars har han ingen bakgrund inom bildbehandling eller den typen av teknik.

Vad kan VR bidra med till testningen?

Ganska mycket. Om man löser latensproblem, som han och personen som doktorerar har tittat på, det handlar mycket om att matcha intryck. Får man bara synintrycket timat så kan man komma långt. Som exempel så skriver han på en ansökan till en ny doktorand som skall jobba med att kombinera VR och AR teknik ihop med fysisk simulering.

Vilka problem kan testningen medföra?

Latens är väldigt viktigt. VR i det här provningssammanhanget har att göra med interaktionen mellan människa och maskin, och VR-teknik… Att få på en klumpig hjälm med glasögon och grejjer så kommer det ta bort verklighetstrogenheten. Att ha mindre hjälmar för att inte det ska bli klaustrofobiskt är viktigt. Grafiken är nog inte dödsviktig för att testa, men ett företag kan tycka det är bra i demonstrationssystem. En studie så kan det nog duga med lägre grafik. Grafiken är inte central, men det finns andra aspekter som gör att den inte kan vara för dålig.

Vad tycker du om vår idé?

Bra idé. Den skulle lika gärna kunna varit på deras forskningsinstitut, ihop med deras simulator. Vi bör fundera på om vi vill gå vidare med kandidatarbetet som exjobb.

Vad krävs för att resultatet skall bli bra?

Latens är svagheten och det är väldigt viktigt att fixa det problemet. Uppdateringsfrekvens och sånt kan spela roll, man blir ju känsligare ju närmre skärmen är ansiktet, om man har fokus längre bort blir man mer förlåtande. Det som är väldigt känsligt är nackrörelser. Om man gör scenarion som inte kräver att föraren hastigt rör på huvudet, kan man exempelvis låta föraren göra små korta rörelser med huvudet? Om man är i en spel-situation så kanske man som användare är mer förlåtande än om man sätter sig i en simulatorn, hjärnan kanske är mer förlåtande om man går in med inställningen att det är ett spel.

Intervju 3

Bakgrund

Biträdande lektor, forskning och utbildning inom VR och mixed reality. Organisationen arbetar med diverse forskningsprojekt som är relaterade till fordonsindustrin. Har diskuterat projekt inom området och varit med och konstruerat olika forskningsprojekt.

Hur går processen till när ni testar en fordonsanvändares beslut i olika situationer och användarupplevelsen generellt?

Varierar från fall till fall.

Vad har er testprocess för brister?

Dyrt och långsamt att använda fysiska bilar, går ej att testa allt med fysiska bilar. Traditionella VR-simulatorer ej tillräckligt realistiska för att få kvalitet. Produkter som köps är låsta.

Vad tror du att fordonsindustrin generellt har för brister när det kommer till denna typ av testning?

Svårtolkad fråga, ej säker vad jag ska svara. Snarare utmaningar?

Vad krävs av en bra bilsimulator för att testa vad användaren tar för beslut i olika situationer och hur den generella användarupplevelsen är?

Fokus på en avsiktlig realism med avseende på det som testas. Fokus på rätt saker. En begränsning hos det vi gör kan vara fokus på HMI-delen.

Vad är de mest generella bristerna med dagens bilsimulatorer?

Finns ett glapp mellan de som är helt virtuella simulatorer och de som är mer realistiska.

Har du någon erfarenhet inom VR?

Biträdande lektor inom området mixed reality och VR.

Vad kan VR bidra med till testningen?

Realistiska miljöer och situationer. Beror på vad man jämför med. Kan vara snabbare att implementera och testa.

Vilka problem kan testningen medföra med VR?

Cybersickness. Svårt med validering. Svårt att få in händer och andra verkliga delar.

Intervju 4

Bakgrund

Doktor inom HMI-system på universitet. Arbetat med olika forskningsprojekt inom fordonsindustrin, bla. designer för 3D-moddellering.

Hur går processen till när ni testar en fordonsanvändares beslut i olika situationer och användarupplevelsen generellt?

Många olika sätt beroende på vilket test. Brukar först utföra någon förstudie, oftast i form av intervjuer. Sen komma på ett koncept som ska testas och sedan implementera det i simulatorn. Sedan låta testpersoner testa detta i simulatorn och observera hur de reagerar och analysera deras subjektiva data, alltså hur deras upplevelse var.

Vad har er testprocess för brister?

Det är en statisk simulator. Användarna känner inte hastigheten, de rör sig inte. Simulatorn skiljer sig från riktig körning. Det finns en inlärningskurva när man använder simulatorn. Alla har olika inlärningskurvor, så det är viktigt hur lång tid användarna behöver på sig för att lära sig.

Vad tror du att fordonsindustrin generellt har för brister när det kommer till denna typ av testning?

Känslan när man kör, att det skiljer sig mellan en riktig bil och en simulator. Tester håller ofta på i en timme, de kan inte mäta långvariga effekter och se om det påverkar beteendet. Om de till exempel testar varningssystem så testar de reaktionstiden. Om du testar användarvänligheten och hur det känns subjektivt, så påverkas det under testets gång.

Vad kan VR bidra med till testningen?

VR kan ändra formatet av testningen. Bättre känsla för miljön. En mer rörlig simulator som inte är statiskt i labbet. Användarna känner sig mer självsäkra om de exempelvis kan testa hemma. Mer flexibilitet med VR.

Vilka problem kan testningen medföra?

Testade det en gång och blev illamående. Det är lätt att illamående uppstår. Känn konstigt att ha VR-hjälmen på sig, det begränsar användaren.

Vad tycker du om vår idé?

Det är en bra idé.

Vad krävs för att resultatet skall bli bra?

Det mesta har redan nämnts. Gör testerna konsekventa. Om de vill mäta mönster måste testet ge konsekventa resultat. Alla måste se samma saker i testen och det får inte finnas för många variabler i testet.

Intervju 5

Bakgrund

Teknikchef på en organisation som äger en biltestningsbana. Arbetat där i nästan 3 år. Startades som init från fordonsindustrin. Fanns ett strategiskt behov för fordonsindustrin för att testa grejor. Arbetat på ett stort fordonsföretag sedan 1997.

Hur går processen till när ni testar en fordonsanvändares beslut i olika situationer och användarupplevelsen generellt?

Beror på experimentet och vad fordonet gör. Tar säkerhetsåtgärder. Beror på det dom provar och vart. Spärra av etc.

Vad har er testprocess för brister?

Planering med kunden. Försöker identifiera risker som kan bli fel. Missar risker.

Vad tror du att fordonsindustrin generellt har för brister när det kommer till denna typ av testning?

Tycker inte om ordet brister. Föredrar utvecklingspotential. Brister = fel, han tycker det inte ska vara så.

Vad krävs av en bra bilsimulator för att testa vad användaren tar för beslut i olika situationer och hur den generella användarupplevelsen är?

Måste vara mer verklighetstrogna. Mindre realistiska svar, mer syntetiska svar. Mer dynamiska tester har haft för dålig realistisk känsla. slipp. svängingar gäller att sätta rätt.

Vad är de mest generella bristerna med dagens bilsimulatorer?

Grafik och kinematik.

Har du någon erfarenhet inom VR?

Lite. Flyger drönare med VR privat. Mest privat.

Vad kan VR bidra med till testningen?

Åstadkomma en omvärldsuppfattning utan en 90gradersdisplay. Rapid prototyping. Utveckla hmi snabbt. Mer nedsänkt upplevelse. shorter time to market.

Vilka problem kan testningen medföra?

Åksjuka. Aldrig upplevt det själv. Låg upplösning. Eftersläpningar (vrider på huvudet). Undrar hur man ska arbeta med hmifrågor, klicka på knappar osv.

Vad krävs för att resultatet skall bli bra?

Uppmaningar med att få allt att funka. Underskatta inte tiden.

Vad ser du för brister?

Uppmaningar med att få allt att funka. Definera use cases.

Telefonsamtal 1

Bakgrund

Anställd på ett stort företag inom fordonsindustrin. Arbetar inom aktiv säkerhet.

Anteckningar

- Företaget har simulatorer. Den information som är publik går att söka fram på internet.
- Intresserade av vårt projekt och av slutprodukten.
- Efterfrågar att vi utforskar vad som finns idag
    - Sök på internet för att se vad andra håller på med
    - Försök hitta standarder för simulatorer och använd dem

Telefonsamtal 2

Bakgrund

Arbetar på ett stort företag inom fordonsindustrin.

Anteckningar

- 2 eller 3 simulatorer
- Simulatorerna är “medium-fidelity”
- En simulator har en riktigt kaross, 3 skärmar och modifierbara displays och sensorer
- Företaget har VR-simulatorer med både UE4 och Unity som grund
- Använder Oculus Rift och HTC Vive
- Har en master thesis på gång inom VR och fordonsindustring
- Beställer simulatorerna av andra
- Har/jobbar på eye tracking till VR-headsetet
- De håller på och testar vad VR-simulatorer kan åstadkomma
- Inga krav på sina VR-simulatorer än, eftersom de skapades för att börja undersöka hur det kan användas
- Hade problem med lagg
    - Behövde göra en del modeller till simulatorn enklare eftersom de var för detaljerade

Bilaga C: Förstudiens intervjumall

Inledning

- Vill du vara anonym?
- Vill du att din position och organisationen du representerar skall vara anonymt?

Informera: Vi skickar ett utkast för godkännande innan något publiceras.

Frågor

- Vart arbetar du och vad är din position?
- Vad har din organisation för bakgrund inom fordonsindustrin?
- Vad har du för personlig bakgrund inom fordonsindustrin?
- Hur går processen till när ni testar en fordonsanvändares beslut i olika situationer och användarupplevelsen generellt?
- Vad har er testprocess för brister?
- Vad tror du att fordonsindustrin generellt har för brister när det kommer till denna typ av testning?

[Om den intervjuade personen inte kommer in på simulatorer själv:]

- Vad krävs av en bra bilsimulator för att testa vad användaren tar för beslut i olika situationer och hur den generella användarupplevelsen är?
- Vad är de mest generella bristerna med dagens bilsimulatorer?

- Har du någon erfarenhet inom VR?
- Vad kan VR bidra med till testningen?
- Vilka problem kan testningen medföra?

[Berätta om vårt projekt]

- Vad tycker du om vår idé?
- Vad krävs för att resultatet skall bli bra?
- Vad ser du för brister?

Bilaga D: Blankett för försöksperson

Virtual Reality Car Driving Simulator - VR Cybersickness

Namn:	Ålder:	Yrke:

Kryssa för huruvida du vill kunna bli kontaktad av Projektgruppen beträffande dina resultat i undersökningen:

Ja ☐ Nej ☐

(Fyll endast i om du kryssat för Ja som svar på ovanstående fråga)
Telefonnummer:_____

Hur lätt har du för att drabbas av så kallad "åksjuka" i det vardagliga livet, dvs vid transport i båtar, bilar och liknande (ej skärm-relaterat)?

1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Kommentar:

_____

_____

Hur många timmar om dagen spenderar du framför en dator/tv-skärm med att:

Spela:_____ Övrigt:_____

Kommentar:

_____

_____

Hur lätt har du för att drabbas av så kallad "åksjuka" när du gör 3D-upplevelser (exempelvis biograf eller spel)?

Ej inträffat ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Kommentar:

Övrig information :

Upplevde du någon gång under simulationen “åksjuka”?

Ja ☐ Nej ☐

(Fyll endast i om du kryssat för Ja som svar på ovanstående fråga)
Var vänlig och precisera när under simulationen åksjukan uppstod och även när den avtog, om den gjorde det:

(Fyll endast i om du kryssat för Ja som svar på ovanstående fråga)
Hur illa upplevde du åksjukan som? (1: mindre obehag; 5: magsjuka)

1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Kommentar:

Övriga kommentarer på simulationen (inlevelse, prestanda, verklighetstrogenhet etc):

Tack för din medverkan!

Anthony | Filip | Jim | Manne | Rasmus | Robin