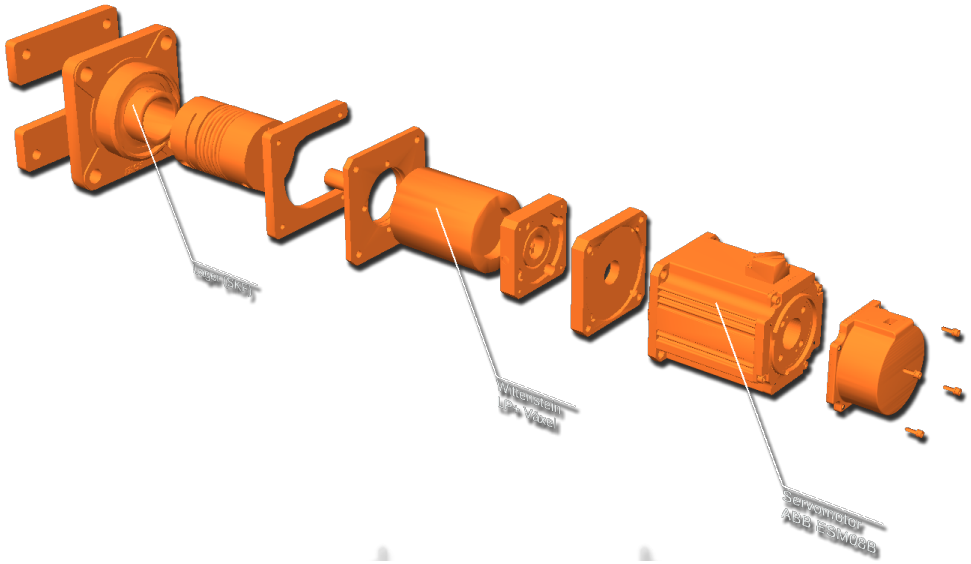
CHALMERS



Interaktiva Fabriker i Augmented Reality

Med hjälp av Microsoft Hololens

Examensarbete inom datateknik

CARL HALVARSSON

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA / GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2017

Examensarbete 2017

Interaktiva Fabriker i Augmented Reality

Med hjälp av Microsoft Hololens

CARL HALVARSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Institutionen för Data- och informationsteknik
Chalmers Tekniska Högskola / Göteborgs Universitet
Göteborg, Sverige 2017

Interaktiva fabriker i Augmented Reality
Med hjälp av Microsoft Hololens
CARL HALVARSSON

Handledare: SVERKER RASMUSON
Examinator: PETER LUNDIN

Examensarbete 2017
Institutionen för Data- och informationsteknik
Chalmers Tekniska Högskola / Göteborgs Universitet
SE-412 96 Göteborg
Telefon +46 31 772 1000

Omslag: Sprängskiss av servosystem för rullstans.

Göteborg, Sverige 2017

Interactive Factories in Augmented Reality
Using Microsoft Hololens
CARL HALVARSSON
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

`Smart Factories` is a publicly funded project aimed towards constructing a pair of prototype connected factories, where visitors can order cardboard VR-glasses using a mobile application. In collaboration with this project, a Hololens application was created which allows the user to interact with a virtutal model of the factory, and obtain information about its different components. The application also served to demonstrate the capabilities of the HoloLens platform on behalf of Cybercom Group, who hosted its development. The result was presented to the project members at the unveiling of the first factory at Lindholmen. This report describes the modular architecture of the application, and lastly, discusses the potential and limitations of using HoloLens for industrial purposes.

Keywords: augmented reality, hololens, visualization

Interaktiva Fabriker i Augmented Reality
med hjälp av Microsoft Hololens
CARL HALVARSSON
Institutionen för Data- och informationsteknik
Chalmers Tekniska Högskola och Göteborgs Universitet

Sammanfattning

`Smarta Fabriker` är ett projekt sponsrat av Västra Götalandsregionen med målet att skapa två moderna, uppkopplade fabriker, där besökare kan beställa VR-glasögon i kartong med hjälp av sin mobil. Som en del av detta projekt skapades en applikation för Microsoft HoloLens, som tillåter användaren att interagera med en virtuell modell av fabriken, och ta del av information kring dess olika komponenter. Syftet med applikationen var att demonstrera potentialen hos HoloLens-applikationen, i uppdrag av Cybercom Group. Resultatet presenterades för projektets medlemmar vid avtäckandet av den första fabriken på Lindholmen. Denna rapport beskriver programmets utveckling och uppbyggnad, samt diskuterar slutligen potentialen och begränsningarna som existerar i att använda HoloLens inom industrin.

Nyckelord: förstärkt verklighet, hololens, visualisering

Förord

Jag vill rikta ett stort tack till Gabriel Ibanez och de andra på Cybercom, som gett mig möjligheten att utforska denna spännande nya teknik hos dem. Jag vill även tacka Sverker Rasmuson, vars råd har varit till stor hjälp, både under arbetet och vid skrivandet av denna rapport. Slutligen vill jag tacka Johan Bengtsson, Ebba Larter, och Adrian Bentland vid GTC för deras stöd och feedback, det har varit fantastiskt roligt och lärorikt att få en inblick i arbetet som sker kring skapandet av en fabrik, och att få jobba med människor med denna nivå av kunskap och engagemang.

Carl Halvarsson, Göteborg, Juni 2017

Innehåll

Förkortningar

AR	Augmented Reality
CAD	Computer-Aided Design
ECS	Entity-Component System
HPU	Holographic Processing Unit
VR	Virtual Reality

Figurer

1

Introduktion

Trots att Augmented Reality varit ett känt koncept sedan 60-talet[1], har tekniken inte förrän på senare år hunnit ikapp visionen. Idag har utbudet av Augmented Reality-applikationer för mobiltelefoner formligen exploderat, och Microsoft lanserar snart sin nya huvudburna holografiska platform HoloLens, som med teknik utvecklad för deras tidigare spelkontroll Kinect[2] kan styras med hjälp av handgester. Volvo Cars har redan inlett ett samarbete med Microsoft för att använda HoloLens som verktyg för att demonstrera nya bilmodeller för sina kunder.[3] Det finns onekligen ett stort intresse hos många företag för vilka användningsområden denna teknik kan tänkas ha inom framtidens industri.

Den senaste trenden kring Augmented- och Virtual Reality ligger även till grund för *Smarta Fabriker*, ett projekt som är sponsrat av Västra Götalandsregionen och som drivs av Göteborgs Tekniska College. Målet med projektet är att bygga ett par uppkopplade fabriker, en i Göteborg och en i Skövde, där besökare kan beställa VR-glasögon i kartong genom en mobilapp. Syftet är att bygga kunskap kring industriell digitalisering samt att locka unga till att arbeta inom industrin. Projektet involverar över 80 studenters examensarbeten, och hittills har dessa lagt ner över 21 000 arbetstimmar i arbetet.[4] Detta är ett fantastiskt tillfälle för att visa hur HoloLens kan utnyttjas som ytterligare en förlängning av en uppkopplad fabrik, och vilken roll virtuell information kan spela för industriell digitalisering — vilket är ämnet för detta examensarbete.

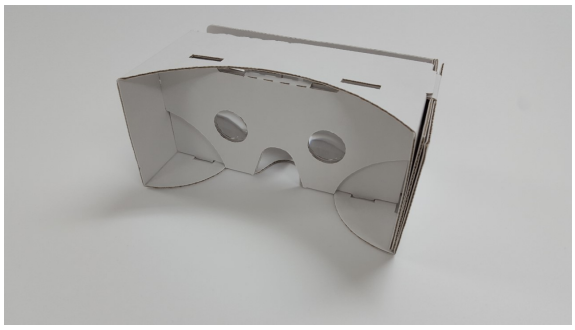
1.1 Bakgrund

Konsultbolaget Cybercom Group är medlemmar i Smarta Fabriker-projektet, och vill som ett led i detta skapa en app för Microsoft Hololens med möjligheten att kunna presentera en virtuell modell av fabriken, där användaren kan ta del av detaljerad information kring fabrikens beståndsdelar och deras funktion. Syftet med detta är att demonstrera HoloLens-plattformens potential — dels som ett medium för att visa upp nya produkter på ett inlevelsefullt sätt, och dels som ett verktyg för att underhålla eller administrera en fabrik på distans. Den färdiga produkten kommer att användas för att marknadsföra Cybercoms satsning inom HoloLens-utveckling.

1.2 Smarta fabriker

Smarta fabriker är ett projekt finansierat av Västra Götalandsregionen med målet att bygga två demonstrationsfabriker, en i Göteborg och en i Skövde, med fokus på digitalisering och uppkoppling. Syftet är, som det står beskrivet på projektets hemsida, att "skapa en plattform för att sprida kunskap om industriell digitalisering för att öka attraktiviteten för tekniska studier och attrahera ungdomar till att vilja arbeta inom industrin."[4] Projektet drivs av Göteborgs Tekniska College (GTC) i samarbete med en rad olika företag inom produktionsindustrin.

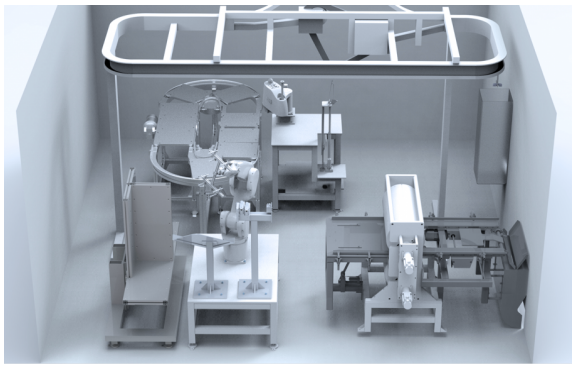
Fabrikernas syfte är att producera vikbara VR-glasögon i kartong, som fungerar som en hållare för smarta mobiltelefoner med stöd för Virtual Reality. Produkten beställs genom en app, utvecklad av Cybercom Group, där kunden får möjlighet att välja en bild och text som under produktionen trycks på kartongen. Glasögonen hämtas sedan ut genom att skanna en QR-kod utanför fabriken, varefter kunden själv får vika ihop dem.



Figur 1.1: VR/AR-glasögonen i kartong som kommer att produceras av fabrikerna. Foto: Johan Bengtsson.

1.2.1 Fabrikens produktionsflöde

Varje fabrik består av tre huvudsakliga moduler - en stans, en 6-axlad industrirobot, samt ett transportsystem för linser. Roboten plockar kartongskivor ur ett magasin och lägger dem i en omtagsfixtur, som används för att garantera att roboten fått rätt grepp. Därifrån hålls de sedan upp framför en skrivare som målar kartongerna med den text och bild som kunden valt. Slutligen läggs kartongen på inmatningen till stansen, där den åker mellan två valsar som skapar den perforering som behövs för att vika glasögonen, innan arket slutligen landar i en utmatningslucka. För att få de plastlinser som krävs för att färdigställa glasögonen visar kunden då upp en QR-kod som finns utskriven på kartongarket framför ytterligare en skanner. Då startas en process där en plockrobot plockar linser från ett transportband och placerar dem på en hydraulisk arm. Därefter lyfts de upp till ett takbaserat transportsystem som tar med sig linserna till avhämtningsplatsen där kunden slutligen kan ta emot dem.



Figur 1.2: En tidig 3D-rendering av fabriken. Bild: Göteborgs Tekniska College.

1.2.2 Tillståndsövervakning

Fabrikerna använder sig av en rad temperatur- och vibrationssensorer för automatiskt tillståndsövervakning. Resultaten från dessa mätningar kan ge en uppskattning på de enskilda maskinernas hälsa och effektivitet. Datan skickas kontinuerligt ut och finns tillgänglig i den mobila applikation som utvecklats av Cybercom.

1.3 Problem

Det finns idag dålig eller ingen kunskap om hur stor potential det finns i att använda HoloLens som ett verktyg inom industrin. Många företag vill visa att de ligger i framkant inom den tekniska utvecklingen, och därför finns det också ett mycket stort intresse för att undersöka vilka möjligheter som plattformen erbjuder.

1.4 Syfte och Mål

Målet med detta arbete var att skapa en applikation för Microsoft Hololens, där användaren ges möjligheten att interagera med en holografisk modell av den fabrik som framtagits i Smarta Fabriker-projektet. Syftet med detta var att skapa en prototypprodukt som Cybercom kan använda i kunddialoger för att demonstrera potentialen i att använda HoloLens som ett verktyg i en digitaliserad fabrik. Målet innefattade att:

- Skapa ett gränssnitt för HoloLens med förmågan att visa en virtuell modell av en fabrik.
- Lägga till information kring modellen i realtid och presentera den på ett tydligt och intuitivt sätt.
- Göra modellen interaktiv, i den mån att man med hjälp av röst- och handgester bör kunna markera individuella delar och visa information kring dem.
- Kunna visa tredimensionella animerade sprängskisser av utvalda delar.

1.5 Avgränsningar

Applikationen skapades i spelmotorn Unity 3D, vilket innebar att inget nytt ramverk behövde skapas för att ladda och rendera 3D-modeller. Ett API för att hantera interaktionen mellan användaren och Hololens-enheten, samt för spatiell kartläggning, fanns utgivet av Microsoft. De modeller som användes är CAD-material från Smarta Fabriker-projektet, men har dock anpassats för realtidsrendering med de begränsade resurser som Hololens-plattformen erbjuder.

2
Teknisk bakgrund

Detta kapitel introducerar några av de tekniker och teoretiska koncept som användes i arbetet. Först ges en bakgrund till begreppet *Förstärkt Verklighet*, samt en introduktion till HoloLens-plattformen, sedan redogörs för några av de decimationsalgoritmer som användes för att bearbeta CAD-materialet. Slutligen ges en kort sammanfattning av ECS-designmönstret, som är ett centralt koncept vid utveckling i Unity 3D.

2.1 Förstärkt verklighet

Begreppet *förstärkt verklighet* (eng. *augmented reality*) myntades av Boeinganställde Tom Caudell år 1990[5], och beskriver kombinationen av virtuella och verkliga sinnesintryck. Till skillnad från *virtuell verklighet* (eng. *virtual reality*, VR), där hela den verkliga världen ersätts med en virtuell sådan, så är syftet med förstärkt verklighet att vara ett komplement till användarens naturliga intryck. Tekniken har på senare år fått bred spridning, och idag stödjer de flesta telefoner enklare former av AR genom geografisk positionering (GPS) och den inbyggda kameran — ett exempel på en applikation som fått stor spridning är Pokémon Go. Andra telefoner är specialbyggda för att stödja Augmented Reality med spårning, såsom Lenovo Phab 2 Pro, som utnyttjar Googles AR Tango-teknik. Djupförståelsen hos dessa skapas genom två eller flera stereoskopiska kameror, och bildigenkänninsalgoritmer kartlägger sedan ett antal referenspunkter i bilden för att spåra telefonens position i rummet.[9]

2.2 HoloLens-plattformen

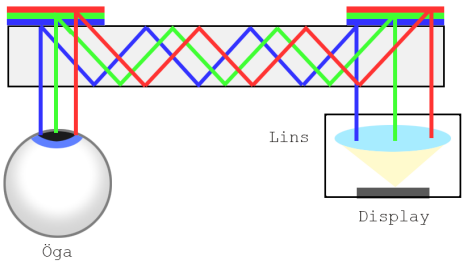
HoloLens är Microsofts senaste huvudburna plattform för förstärkt verklighet, som fungerar genom att leda ljusstrålar igenom en serie glasskivor (s.k. *vågledare*) framför användarens ögon, för att på så vis ge illusionen av att holografiska föremål befinner sig i rummet. (se sektion 2.2.1) Plattformen har stöd för bl.a. spatiell kartläggning, röstigenkänning, och gestigenkänning, varav de två senare är de huvudsakliga medlen för att interagera med enheten. Microsoft har demonstrerat hur plattformen kan användas till såväl spel och rekreation som ett hjälpmedel vid kreativt arbete. Företag som SKF[7] och Volvo[3] har redan börjat undersöka hur HoloLens kan tänkas användas som ett verktyg för tillståndsövervakning och PR-skapande.

HoloLens körs med operativsystemet Windows 10, och uträkningarna utförs utav

en 32-bitars processor samt vad Microsoft kallar för en *Holographic Processing Unit*, eller HPU. Dennas syfte är att behandla sensordata och upprätthålla den spatiella kartläggningen av rummet, samt hantera tolkningen av gester.[10] Enligt Microsoft kan HPU:n genomföra över en biljon flyttalsoperationer per sekund.[11]

2.2.1 Display

Displayen som visar det holografiska materialet består av två s.k. *Vågledare* (eng. *Waveguides*), som är glasskivor med total intern reflektion. Ljuset från en skärm på ena sidan reflekteras in i vardera ledare genom en kollimator som ser till så att alla ljusstrålar är parallella. Strålarna delas upp i frekvensband och speglas in i skivan genom räfflade prismor. Ljusstrålarna transporteras genom glassubstratet enligt samma princip som i en fiberoptisk kabel. Slutligen reflekteras strålarna ut igen genom motsvarande prismor i displayen.[12] Detta är en additiv form av display, vilket innebär att den endast kan lägga till ljus över det som användaren redan ser. På grund av detta lämpar sig HoloLens bäst i dunkla ljusförhållanden.



Figur 2.1: Illustration över vågledarens struktur.

2.2.2 Spatiell kartläggning

Flera sensorer samspelar för att kartlägga och bedöma HoloLens-enhetens position i rummet. En intern mätenhet (eng. *Internal Measurement Unit*, IMU) arbetar tillsammans med en djupmätande kamera samt fyra miljötolkande kameror[8] för att mäta de relativa avstånden mellan närliggande föremål samt bygga upp en rudimentär modell av rummet. Eftersom detta koordinatsystem helt bygger på jämförelser av mätdata rörande rummets utseende så kan HoloLens inte särskilja på två identiska rum, utan betraktar dem som samma. På grund av osäkerheter i mätningarna dessutom kan två punkter i den virtuella miljön komma att flyttas relativt varandra efter att en säkrare mätning gjorts, vilket kan påverka positionen hos virtuella föremål. Mätinstrumenten har en maximal räckvidd på fem meter, vilket innebär att enheten har betydligt svårare att hitta sin position i öppna ytor utan goda referenspunkter. Om inga referenspunkter kan hittas av mätenheten används istället ett relativt koordinatsystem som tillåter menyer att existera relativt användarens huvud. HoloLens har inget stöd för geografisk positionering via GPS.[6]

2.2.3 Gester

Hololens använder sig av Kinekt-systemet, en teknik som tidigare utvecklats för spelkonsolen Xbox[2], för att känna igen handgester. Microsoft kallar dessa för *gestures* (sv. gester), och de två gester som Hololens kan känna igen kallas *bloom* samt *tap*. Bloom utförs genom att att användaren öppnar sin slutna hand framför sig, detta är det universella kommandot för att öppna huvudmenyn och kan inte bindas till någon annan funktion. Tap består av en höjning och sänkning av pekfingret i en nypande rörelse, och kan användas tillsammans med data rörande handens position för att registrera klick eller dragrörelser.[6]

2.2.4 Blick

Relationen mellan användarens blickfång och det holografiska materialet är central för att kunna interagera med Hololens-applikationer. informationen om enhetens position relativt den virtuella världen kan användas för att markera eller belysa holografiska föremål som användaren tittar på, något som Microsoft kallar för *gaze* (sv. *blick*).[6] Att bedöma vilka föremål som ligger framför användaren görs på applikationsnivå med hjälp av *ray casting*.

2.3 Polygondecimation

Inom CAD-modellering är målet generellt att skapa en så komplett digital kopia av motivet som möjligt, då modellerna kan komma att användas som underlag vid produktion. Detta gör dock modellerna dåligt lämpade för användning på mobila plattformer med begränsad renderingskapacitet såsom Hololens. Lyckligtvis finns det idag flera metoder för att automatiskt reducera detaljnivån hos en modell genom iterativa algoritmer. De två metoder som används i detta projekt kallas på Engelska för *Quadric edge collapse decimation* (QECD) samt *Coplanar facets merging* (CFM).

2.3.1 Quadric edge collapse decimation

QECD går ut på att iterativt sammanfoga par av punkter (eng. *Vertices*) med kriteriet att minimera felet, med avseende till ytans topologi. Vid varje steg tilldelas varje punkt den symmetriska 4x4-matrien Q, som är summan av planen hos alla trianglar som möts i punkten, uttryckt som en andragradsyta (eng. *Quadric surface*). Felet ges av $\bar{v}^T(Q_1 + Q_2)\bar{v}$, där $\bar{v}$ är parets mittpunkt, och resultatet representerar kostnaden för att sammanfoga detta par.[14] Paret med minst kostnad sammanfogas, och proceduren upprepas tills att det återstående antalet punkter motsvarar målet.

2.3.2 Coplanar facets merging

Denna algoritm söker i modellen efter sammanhängande koplanära punkter — dvs. punkter som ligger inom samma geometriska plan — och sammanfogar dessa för att skapa en motsvarande yta med färre punkter. Metoden lämpar sig väl för plana ytor eftersom den bevarar skarpa vinklar och konturer. Ofta används ett tröskelvärde

för att tillåta att trianglar vars plan avviker med en viss vinkel kombineras till ett gemensamt plan.[13]

2.4 ECS-designmönstret

ECS står för *Entity–component system*, och är ett designmönster inom objektorienterad programmering (OOP) där ett program delas upp i `Entities` (sv. *Entiteter*) och `Components` (sv. *Komponenter*). Varje entitet kopplas samman med en eller flera komponenter, som var och en bidrar till dess övergripande funktion. Komponenter kommunicerar med varandra antingen genom dubbel länkning, eller genom att skicka meddelanden via sin entitet. Syftet med detta tillvägagångssätt är att minska kopplingen och öka återanvändningsgraden i koden.[15] Designmönstret är mycket vanligt inom spelprogrammering och simulationer där ett logiskt objekt ofta har egenskaper som berör flera domän, exempelvis vid samspelet mellan fysiksimulering, ljud, och rendering då ett fallande föremål träffar marken.[16]

ECS-mönstret är en stark tolkning av principen *Composition over inheritance*, som syftar till att ett objekts syfte bör definieras genom att låta det bestå av mindre, fristående objekt, snarare än att bygga långa arvshierarkier från ett basobjekt.[17] Ett klassiskt exempel på ett problem som kan uppstå i en arvsbaserad arkitektur är om man vill implementera basklassen `Duck`. En tänkbar metod som man kan behöva i denna klass är `Fly()`. Problemet uppstår när man vill implementera en underklass till `Duck` som inte kan flyga. Alternativet är då att flytta metoden `Fly()` ett steg ner i hierarkin, vilket leder till upprepad kod. Ett annat alternativ är att låta `Duck` ärva ifrån `FlightlessDuck` - men då behöver all kod som inte rör själva flygningen istället upprepas i alla grenar förutom `Duck`! En lösning till detta är att låta klassen innehålla ett objekt som implementerar interfacet `IFlying`, som beskriver beteendet. ECS-mönstret drar detta till sin spets genom att låta *alla* egenskaper hos ett objekt vara komponenter.

3

Systemkonstruktion

I detta kapitel redogörs först för de metoder som användes i arbetet — hur upplägget såg ut, vilka material som krävdes, och en kort introduktion av de utvecklingsverktyg som utnyttjades. I sektion 3.4 beskrivs sedan hur skeendet såg ut då applikationen realiserades, vilka problem som uppstod och hur dessa löstes, samt slutligen en översiktsbild av programmets struktur.

3.1 Upplägg

Arbetet utfördes enligt Kanban-metoden, en SCRUM-liknande metodik, där arbetet delades upp i vertikala delmål. Möten med produktägaren GTC skedde löpande varannan vecka där statusen för utvecklingen diskuterades och resultatet för perioden visades upp. Delmålen beskrevs som följande:

- Konstruera en klass som representerar en enskild modul och dess funktion.
- Designa en kontextmeny som kan innehålla två nivåer av information.
- Skapa en algoritm för att placera menyn på sidan av en modul, på ett sådant sätt att den alltid sitter på den sida som är närmast användaren.
- Skapa ett system för att flytta och manipulera föremål med hjälp av gester.
- Konstruera ett system för att hantera sprängskisser.
- Välja en metod för att läsa information om modulerna från en fil.

För att tillåta snabb prototypning beslutades att endast en av fabrikens moduler skulle implementeras till en början, så att resten av funktionerna sedan kunde testas på denna.

3.2 Utvecklingsverktyg

Att från grunden utveckla ett ramverk med stöd för 3D-rendering och integration med HoloLens är i sig ett mycket stort projekt — på grund av detta är utvecklingsverktyg från en tredje part en förutsättning för att kunna skapa en applikation inom rimlig tid. Idag finns flera kostnadsfria alternativ för såväl spelutveckling som 3D-modellering som underlättar denna process. I detta fall användes spelmotorn Unity 3D tillsammans med Microsofts egna bibliotek HoloToolkit som bas för applikationen, och modelleringsprogrammet Blender användes för bearbetning av modellerna.

3.2.1 Unity 3D

Unity 3D är ett gratis program för snabb och enkel utveckling av spel och simulationer. Det är också den enda plattform som har officiellt stöd för utveckling av Hololens-applikationer, vilket gjorde den till ett naturligt val för detta projekt. Unity använder sig av en ECS-baserad programarkitektur (se sektion 2.4), där allt byggs upp utav hierarkier av s.k `GameObjects`. `GameObjects` är minimala byggstenar som i stort sett endast består av en position och orientation i spelvärlden. Varje GameObject kan associeras till flera `MonoBehaviors` (även kallade `Components`), som vardera bidrar med en specialiserad funktionalitet och tillsammans samspelar för att definiera objektets funktion.[18] En klass som ärver från `MonoBehavior` kan valfritt implementera särskilda metoder såsom `Start`, `Awake`, och `Update`, som då registreras till att initiera och uppdatera beteendet. Unity använder sig av reflektion för att hitta dessa metoder. Hierarkier av `GameObjects` kan serialiseras och sparas till vad som Unity kallar för `Prefabs`, som kan laddas in i applikationen under exekvering. Programmering för Unity sker i C#, Javascript, eller Boo — för detta projekt användes C#. `MonoBehaviors` refereras hädanefter till som *beteenden*.

3.2.1.1 Coroutines

Unitys API stödjer inte trådning, men däremot tillhandahåller s.k. *Coroutines* möjligheten att köra en rutin som kan skjuta upp sin exekvering tills nästa varv i spelloopen. För att kunna köras som en coroutine behöver metoder returnera typen `IEnumerator`. Rutinen kan suspenderas när som helst med `yield return null`, och avslutas när exekveringen når metodens slut. En rutin kan suspenderas under en bestämd tid genom att returnera ett objekt av typen `WaitForSeconds`, vars konstruktor tar önskat antal sekunder.

3.2.2 HoloToolkit

Microsoft erbjuder ett fritt resurspaket för att utveckla för Hololens som kallas Holotoolkit. Detta innehåller bl.a. utbildningsmaterial, en fokusmarkör, samt ett API med hjälpklasser för att hantera input, spatiell kartläggning, spatiellt ljud, och nätverksdelning. HoloToolkits inputsystem är byggt omkring *Managers*, där `GazeManager` hanterar vilket objekt som för tillfället ligger i användarens fokus, och `InputManager` hanterar input från gester samt tangentbordet. Varje objekt, som besitter minst ett beteende som implementerar interfacet `IFocusable`, kommer att hanteras av GazeManager då objektet hamnar i fokus. Objekt kan även registrera sig själva som globala lyssnare hos `GazeManager` genom delegatmetoden `OnFocusChanged`. Vilket objekt som ligger i fokus bedöms med hjälp av s.k. *ray casting*, dvs. att en linje projiceras från användarens ögon som sedan testas mot de olika objektens geometri. `InputManager` fungerar på ett liknande sätt, där objekt med beteenden som implementerar `IInputHandler` exponerar metoderna `OnInputDown` samt `OnInputUp` och därmed blir klickbara. Även här har objekt möjligheten att registrera sig själva som globala lyssnare av inputevent.[19]

3.2.3 Blender

Blender är ett gratis open source-verktyg för 3D-modellering, rendering, och animering. Användare har möjligheten att själva skapa insticksmoduler i programspråket Python, vilket innebär att nya metoder snabbt blir tillgängliga.[20] Programmet används i detta projekt för att bearbeta CAD-modellerna.

3.3 Material

Eftersom modulerna i applikationen bygger på verkliga produkter ligger stor vikt i att informationen som visas är korrekt. Informationen visas intill hologram bestående av modifierade CAD-modeller, och kompletteras i vissa fall med produktritningar.

3.3.1 CAD-modeller

CAD-modeller (CAD, eng. *Computer-Aided Design*) är en del av produktionsunderlaget för medparten av komponenterna och fanns därmed redan färdiga hos företagen som producerar dem. Produktionsunderlag har även skapats av elever inom CAD-konstruktion specifikt för Smarta Fabriker-projektet. Dessa bearbetades med de reduceringsalgoritmer som beskrivs i sektion 2.3.

3.3.2 Informationstexter

Leverantörerna bidrog själva med informativa reklamtexter om sina produkter. Dessa anpassades för att passa i sammanhanget, och i vissa fall behövede texterna även översättas från Engelska. Det skrevs dessutom korta ingresser som beskriver komponenternas specifika roll i fabriken.

3.3.3 Produktritningar

För att komplettera den textbaserade informationen om modulerna användes produktionsritningar i vissa av fallen. Syftet med detta var att demonstrera hur olika informationsmedium kan utnyttjas på Hololens för att lagra teknisk information. Ritningarna kommer ifrån datablad som tillhandahavs av produkternas respektive leverantörer, och bearbetades i bildbehandlingsprogram för att få en transparent bakgrund.

3.4 Genomförande

Arbetet delades upp i flera delmoment, och genomfördes i ordning efter hur kritiska de var för applikationens funktion. Ett separat beteende för modulen, fabriken, samt kontextmenyn implementerades, och sedan skapades en fokusmarkör med feedback samt ett system för att flytta och rotera godtyckliga objekt. Slutligen färdigställdes fabrikens återstående delar en efter en. I detta avsnitt beskrivs i detalj de moment som här nämnts, samt vilka beslut som ledde fram till programmets slutgiltiga struktur.

3.4.1 Förberedning av utvecklingsmiljö

För att utnyttja HoloToolkits funktionalitet krävdes en viss förberedelse i Unity. Inputsystemet fungerar genom att placera ett tomt `GameObject` med beteendena `InputManager` samt `GazeManager`. Dessa håller sedan reda på när kameran tittar på ett objekt innehållande ett beteende som implementerar `IFocusable`, respektive när användaren trycker på ett objekt med ett `IInputClickHandler`-beteende. För att representera användarens huvud i spelvärlden räckte det med att placera ut ett kameraobjekt — HoloLens anpassar automatiskt kameran efter enhetens orientation i världen.

3.4.2 Bearbetning av modeller

Det första steget i arbetet var att utveckla ett arbetsflöde för att decimera modellerna till en detaljnivå som HoloLens-plattformen klarar av. Modellen skickades modulvis i sex filer av Stefan Rydén, som var ansvarig över CAD-arbetsgruppen på yrkeshögskolan YRGO. Modul 1, som bestod av en industrirobot med tillhörande magasin, valdes som första modul att implementeras i applikationen. Ett tidigt problem var att filen var för stor för att öppnas i Blender med den hårdvaran som fanns tillgänglig. Lösningen var att förbehandla modellen i det kostnadsfria 3D-programmet MeshLab, om är skapat för att hantera stora datamängder. Förbehandlingen bestod av att sammanfoga punkter som låg inom ett visst avstånd från varandra, samt applicera ett QECD-filter (se sektion 2.3) över modellen som i genomsnitt minskade polygonantalet med 20% och antalet punkter med 50%. Sedan öppnades modellen i Blender där den delades upp (med automatiserade metoder) efter sammanhängande element. Varje del behandlades separat med olika grader av CFM-decimation, och omtriangulerades för att undvika de visuella artefakter som kan uppstå när en polygon är för smal. Slutligen rensades onödiga detaljer bort för hand. Tester visade att HoloLens har kapaciteten att rendera ungefär 900 000 polygoner med en bildfrekvens på ungefär 30 bilder per sekund.

3.4.3 Moduler

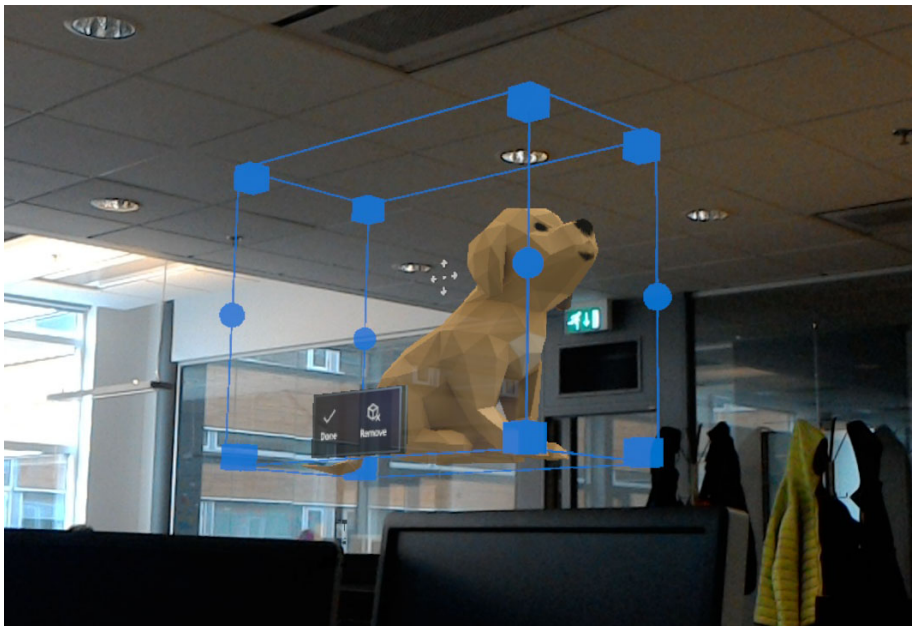
Fabriken består av ett stort antal komponenter, och en tidig fråga var i vilken granularitet dessa skulle uppdelas. För att minska mängden text som skulle behöva skrivas beslutades att fabrikens komponenter skulle delas upp efter hur urskiljbara deras uppgifter var i produktionsflödet. Detta resulterade i nio delar, som kom att kallas *Moduler*. Varje modul gavs beteendet `Module`, som kontrollerar modellens färg och transparens. Klassen implemnterar interfacen `IFocusable` och `IInputClickHandler`, där `IFocusable` tillför metoderna `OnFocusEnter()` respektive `OnFocusExit()`, som kallas av GazeManager då komponenten ligger under användarens markör. `IInputClickhandler` gör objektet klickbart, genom att `InputManager` anropar metoden `OnInputClicked(InputEventData eventData)` från interfacet.

3.4.4 Moduldefinitioner

JSON (Javascript Object Notation) är ett textbaserat datautbytesformat som har internt stöd av Unity. Detta format valdes för att lagra informationen rörande modulerna, eftersom det är lätt att skapa och redigera JSON-dokument även utan vidare programmeringskunskaper. Varje modul söker automatiskt i mappen `HoloFactory/Assets/StreamingAssets` efter en JSON-fil med samma namn som sitt `GameObject`, för att ladda sin definition ifrån. En definition består av ett visningsnamn, en kort beskrivning, samt en mer omfattande informationstext. Definitionen kan även innehålla en lista på sökvägar till andra objekt som kan laddas och fästas i menyn, till exempel bilder eller 3D-modeller. Programmet refererar till dessa som `Widgets`, och de sparas lämpligast som s.k. `Prefabs`, vilket är Unitys namn på serialiserade objekthierarkier. En JSON-fil med rätt format kan direkt serialiseras till en `ModuleDefinition` med hjälp av Unitys egna serialiserare. En stor fördel är att fält kan tas bort eller läggas till i filen utan att serialiseringen fallerar, så länge texten använder korrekt JSON-syntax. Serialiseraren ignorerar fält som inte finns med i målklassen, och använder standardvärdet när fältet saknas i filen.

3.4.5 Manipuleringsläget

En utmaning med plattformen var att när programmet väl startar finns det inget sätt att veta hur koordinatsystemet är orienterat i relation till enheten, förutom att uppåt-riktningen alltid står i motsatt riktning till gravitationskraften. I den första versionen av applikationen placerades fabriken alltid i origo, dvs. på användarens huvud, vilket innebar att denne behövde ställa sig i en strategisk ställning medan programmet laddade för att fabriken skulle hamna på rätt plats. En tidig lösning till detta var att placera fabriken fem meter framför och 1,8 meter nedanför enheten vid programmets start, vilket alltid placerade den på bekvämt avstånd från början; dock var det uppenbart att ett system skulle krävas för att kunna flytta och rotera fabriken. Som förlaga till detta system användes Microsofts egna applikation *Holograms*, som följer med enheten som standard.
Systemet består av komponenten `BoundaryBox`, som placeras på ett tomt `GameObject` tillsammans med en `BoxCollider`, vilket är en matematisk representation av ett rätblock som används vid kollisionstestning i Unity. `BoundaryBox` genererar sedan automatiskt ett objekt med beteendet `BoundaryCorner` i varje hörn av rätblocket, och ett med beteendet `BoundaryEdge` på varje vertikal kant. Dessa länkas sedan samman såtillvida att ett klick på något av hörnen eller kanterna kallar på metoden `StartInteraction` hos `BoundaryBox`, och skickar som argument en `BoundaryOperation` som representerar interaktionstypen. Beteendet sköts sedan inuti `BoundaryBox` tills dess att metoden `StopInteraction` kallas. Operationen utförs automatiskt på objektet ovanför `BoundaryBox` i hierarkin, vilket innebär att vilket objekt som helst kan utnyttja systemet. I den första implementationen var vardera av de tre ovanstående klasserna ansvariga för att fokusmarkören visade deras respektive feedbacksymbol. Detta visade sig dock vara en bristfällig lösning då ansvaret över markören blev distribuerat över för många klasser, och ingen klass hanterade markören då den inte pekade på något av dessa objekt. Istället implementerades en separat `CursorManager` som samlade hela detta ansvar i en klass, vilket beskrivs i



Figur 3.1: Manipuleringsläget i Microsofts program *Holograms*

nedanstående kapitel.

3.4.6 Fokusmarkören

HoloToolkits API innehåller klasser för att hantera en fokusmarkör, som visas i mitten av användarens synfält. Denna används som en slags muspekare, dels för att ge användaren högre precision i sina gester, och dels för att visa feedback baserat på vilka interaktioner som finns tillgängliga. HoloToolkit implementerar inte denna feedback, men låter utvecklaren utnyttja klassen `ObjectCursor` där det finns möjlighet att specifiera ett antal `ObjectCursorDatum`, som representerar olika grafiska varianter. För att tydligare visa för användaren när och hur det går att interagera med ett objekt skapades klassen `CursorManager`, som anpassar markörens utseende baserat på vilket objekt den står över. Denna klass registrerar sig som global lyssnare hos `GazeManager` och `InputManager`, och blir på så vis fristående från resten av koden.

3.4.7 Kontextmenyn

För att kunna visa informationen om de olika modulerna behövde ett menysystem utvecklas som kunde stödja flera olika former av media. Något som bestämdes tidigt var att texten skulle projiceras intill modellen på ett sätt som påminner om textplakat på ett museum. Tanken bakom detta var att förstärka intrycket av att användaren befinner sig i ett rum som den utforskar. En annan anledning var att det andra alternativet, att låta menyn finnas relativt användarens huvud, skulle ge ett instängt intryck — att ha informationen i en s.k. *Heads Up Display* (HUD) är dessutom något som Microsoft avråder ifrån av just denna anledning.[6]

Menyn består av en plan yta med en titeltext, informationstext, och en knapprad. När användaren ser på en modul så visas dess namn, tillsammans med upppmaningen att trycka på modulen för att markera. Efter att modulen markerats visas knappar med alternativ, samt en kortare ingress som beskriver dess funktion i fabriken. Knapparna designades för att likna de som används i HoloLens egna gränssnitt. För att avmarkera en modul trycker man på den igen — detta visade sig dock vara otydligt för många, och därför lades senare en stäng-knapp till. I knappraden finns också alternativet att visa mer info, förutsatt att modulen har extra info sparad i sin definitionsfil. Denna knapp visar dessutom eventuella `Widgets` (se sektion 3.4.4).

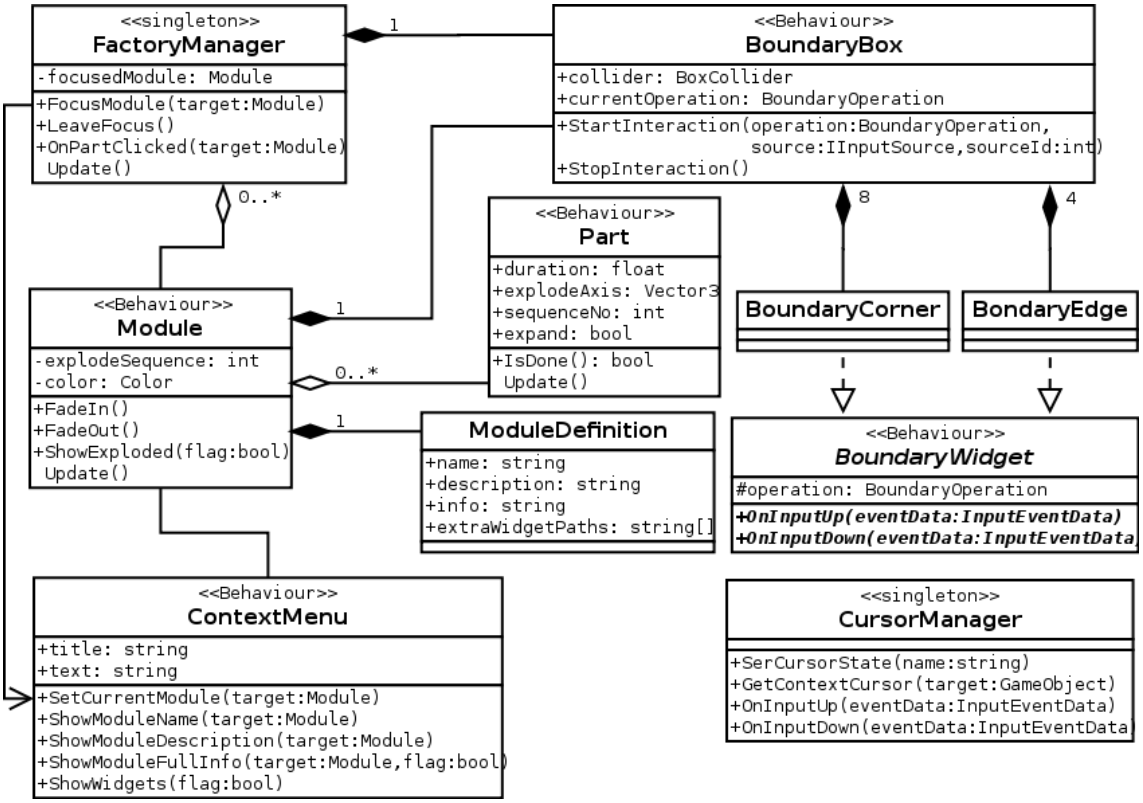
3.4.8 Prestanda

I den första iterationen av applikationen renderades modellerna med Unitys standardshader, vilket gav en renderingshastighet på 14-16 bilder per sekund. Detta medförde uppenbart jitter vid användning. Detta problem löstes genom att skapa en ny shader, som renderar modellen i en solid färg och enbart tar en ljuskälla i beräkning. Alla beräkningar görs dessutom med lägsta möjliga precision. Som en vidare optimering programmerades modellerna till att skifta mellan ett transparent material — då de tonade in och ut — till ett solitt material i övriga fall. Med dessa ändringar kunde bildfrekvensen fördubblas.

3.4.9 Part-klassen

Det sista steget i arbetet var att implementera sprängskiss-funktionen. Den första tanken var att göra sprängskisserna helt procedurella, och bara låta delarna röra sig ifrån varandra med ett konstant avstånd från sin gemensamma mittpunkt. Detta visade sig dock vara otillräckligt, då vissa delar behöver plockas isär i en viss ordning och på ett visst sätt, och detta givetvis är något man vill illustrera om sprängskisserna ska vara till någon nytta. Den nya lösningen gick ut på att skapa ett `Part`-beteende, som kopplas till varje del som ska ges en rörelse under expanderingen. `Part`-beteendet har tre publika fält: `explodeAxis`, som är expansionens lokala riktning och magnitud, `duration`, som är expansionens varaktighet, och `sequenceNo`, som är denna dels sekvensnummer i expansionsförloppet. Varje modul listar automatiskt alla objekt med detta beteende i sin hierarki — alla moduler som har minst en `Part` betraktas ha en sprängskiss tillgänglig. När sedan `ShowExploded(true)` anropas startas en `Coroutine` där alla `Parts` med sekvensnummer 0 expanderas. Expanderingen sker genom att interpolera objektets position lokalt över `explodeAxis`, inom tiden som anges i `duration`. Rutinen väntar sedan tills att alla delar nått sin slutposition innan den ökar sekvensräknaren och startar nästa sekvens efter en kort fördröjning. När det inte längre finns några delar kvar att expandera avslutas tråden. Vid `ShowExploded(false)` genomförs istället expansionen baklänges tills att sekvensnumret är 0. Genom att bygga hierarkier av `Parts` kan man låta flera delar röra sig tillsammans och expandera på ett trädartat vis, vilket gör systemet mycket flexibelt.

3.4.10 Programstruktur



Figur 3.2: UML-diagram över programmets struktur.

Det slutgiltiga programmet innehåller tre huvudsakliga *Managers*, som styr över övergripande funktioner. (se figur 3.2) Dessa tre är `FactoryManager`, `CursorManager`, och `SoundManager` (ej synlig i diagrammet). Utöver dessa finns dessutom HoloToolkit-klasserna `InputManager` och `GazeManager`. Samtliga av dessa följer ett singleton-mönster. FactoryManager listar automatiskt alla moduler i sin hierarki och fungerar som en mediator mellan dessa. Varje modul listar i sin tur alla `Parts` i sin hierarki, om där finns några, och hanterar sekvensordningen mellan dessa då en sprängskiss genomförs. `SoundManager` hanterar förfrågningar om att spela upp ljudfeedback, exempelvis när en knapp klickas.

4

Resultat

En applikation med möjligheten att utforska en virtuell fabrik skapades. Fabriken är indelad i moduler som användaren kan interagera med och visa information om; en av delarna kan dessutom expanderas till en tredimensionell sprängskiss där de huvudsakliga komponenterna namnges. Slutligen utvecklades ett generellt system för att flytta och manipulera 3D-objekt som tillåter användaren att placera fabriken i rummet.

4.1 Applikationen

När applikationen startas placeras fabriken automatiskt fem meter framför och 1,8 meter nedanför användarens huvud. Totalt finns nio moduler i fabriken som kan markeras, varav tre har sin information kompletterad med produktionsritningar, och en har stöd för sprängskisser.

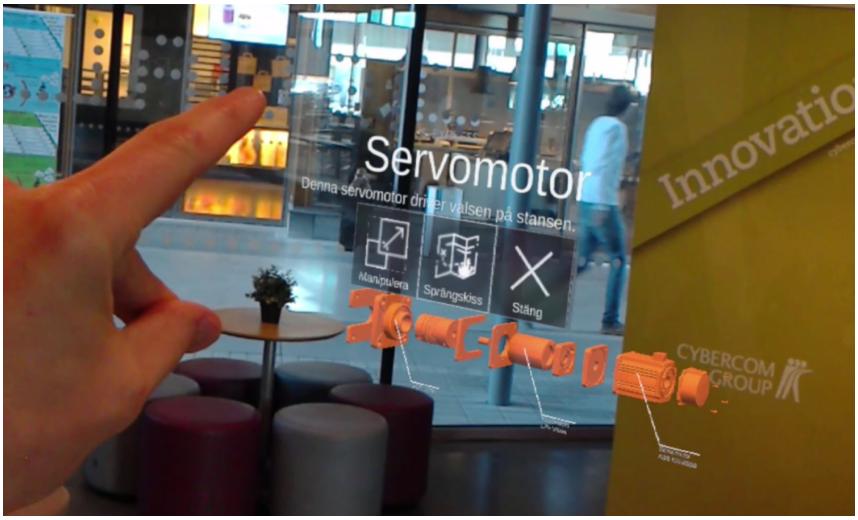
Figur 4.1: Kontextmenyn med utökad information.

Kommunikationen melan användaren och fabriken sker huvudsak på tre olika sätt: Färgmarkering, fokusmarkören, och kontextmenyn. Färgmarkering används för att visa vilken modul som användaren tittar på, och för att visa att denna går att interagera med. Modulen visas då i en orange färg, i kontrast till den gråa färg som är standard. Fokusmarkören fungerar som en muspekare, som ändrar utseende baserat på vilken form av interaktion som kan utföras i situationen — exempelvis visar markören en hand med utsträckt finger då användaren har möjlighet att klicka

på ett objekt. Kontextmenyn är en plan textyta som visas när en modul ligger i fokus eller är markerad. Menyn placeras över sidan på modulen och flyttas automatiskt till den sida som är vänd mot användaren.

4.1.1 Sprängskisser

En av modulerna har alternativet att expanderas till en sprängskiss. Alternativet att visa sprängskissen dyker automatiskt upp kontextmenyn för de moduler som stödjer detta och aktiveras/avaktiveras med samma knapp.



Figur 4.2: Sprängskissfunktionen.

4.1.2 Manipuleringsläget

Genom att trycka på `P` på det trådlösa tangentbordet startas manipuleringsläget, och en blå kontur visas runt fabriken. När fokusmarkören pekar över sidan på fabriken visar den pilar i fyra riktningar som indikerar att användaren kan gripa tag för att flytta modellen. Genom att på samma sätt gripa tag i ett av de fyra hörnen och dra kan fabriken förstoras eller förminskas, och de runda kulorna på sidorna kan användas tillsammans med en horisontell dragrörelse för att rotera fabriken. (se figur 4.3)

4.2 Presentationen

Fredagen den 2/6 2017 hölls en mässa i fabrikens lokaler på Lindholmen för alla projektets deltagare och sponsorer. Under mässan demonstrerades den färdiga HoloLens-applikationen, och varje deltagare fick en chans att prova på den själv. HoloLens gav upphov till många intressanta diskussioner, och applikationen mottogs övervägande positivt.

Figur 4.3: Manipuleringsläget, med den fullständiga fabriken i bild.

Figur 4.4: Applikationen demonstreras på Lindholmen. Foto: Gabriel Ibanez.

5
Slutsats

Detta arbete utfördes i samarbete med projektet *Smarta Fabriker*, som kretsar kring att bygga en uppkopplad, digitaliserad fabrik. Målet var, att på uppdrag av konsultbolaget Cybercom Group, skapa en applikation för Microsoft HoloLens där användaren har möjligheten att utforska en virtuell version av fabriken. Detta för att visa möjligheterna med plattformen, samt för att skapa en prototypprodukt som Cybercom kan demonstrera för sina kunder. Applikationen som konstruerades har förmågan att visa två nivåer av information om de olika komponenterna i fabriken, och några av dessa har dessutom stöd för att visa produktionsritningar och animerade sprängskisser. Ett system för att flytta objekt med hjälp av handgester skapades även. Resultatet demonstrerades slutligen vid avtäckandet av den första fabriken på Lindholmen för alla som medverkat i projektet.

Att utveckla för HoloLens visade sig vara mycket enkelt, och den goda tillgången till verktyg och hjälpmedel hjälpte till att påskynda processen. Den färdiga produkten har en modulär utformning (se sektion 3.4.10) och kan lätt konfigureras om till att visa en annan fabrik eller maskin. Enligt Cybercom kommer applikationens vidareutveckling att vara föremål för framtida examensarbeten.

5.1 Potential inom industriella tillämpningar

Den begränsande faktorn i utvecklingsprocessen var möjligheten att förenkla CAD-modellerna till den nivån att de kunde renderas tillsammans i realtid med en godtagbar hastighet, och samtidigt bibehålla den detaljrikedom som eftersöks hos en CAD-modell. Något som visade sig ha stor inverkan på prestandan var hur avancerat shaderprogrammet som användes för renderingen var, vilket beskrevs i sektion 3.4.8. Hur mycket tid som behövdes för att behandla modellerna var dock ett större problem. Algoritmerna som användes (se sektion 2.3) har bägge som mål att till största möjliga grad bibehålla topologin hos modellen, men skiljer inte mellan små och stora detaljer, vilket ofta innebar att exempelvis modellerade serie- och patentnummer fick större prioritet än den överordnade formen hos motorn de stod skrivna på. Algoritmerna tar givetvis inte heller hänsyn till huruvida en detalj är synlig eller ej, vilket innebar att manuell redigering fortfarande spelade en stor roll i processen. Det bästa alternativet för att påskynda utvecklingen av denna typ av AR-applikationer vore sannolikt att från början skapa CAD-modeller med en lägre detaljupplösning som ett led i den vanliga modelleringsprocessen. Att visa färre delar i taget är en annan möjlig lösning — det visade sig att HoloLens under rådande omständligheter

har kapaciteten att rendera strax under en miljon polygoner i samma vy (se sektion 3.4.2), vilket skulle vara tillräckligt för att visa enskilda modeller i sitt ursprungliga skick.

Eftersom Unitys programstruktur uppmuntrar återanvändning av kod, kan utvecklingskostnaderna för en praktisk applikation av denna sort bli förhållandevis låga. Ett möjligt användningsområde finns inom bilindustrin, där HoloLens skulle kunna användas för att instruera mekaniker vid verkstäder i hur underhåll av en ny bilmodell går till. Detta skulle ha fördelar ur ett hållbarhetsperspektiv, eftersom inga testbilar behöver produceras och skickas ut för detta ändamål. På ett liknande sätt kan sensordata relaterad till maskinstatus i en fabrik skickas till en applikation av den typ som skapats i detta projekt, vilket skulle innebära att fabrikens administratör slipper resa till plats för att tillse fabriken.

5.2 Vidare arbete

IT-företaget Prevas har, i samarbete med Göteborgs Tekniska College, utvecklat ett API för att ta emot livedata rörande maskinhälsa hos fabrikens olika delar. Maskinhälsa är ett mått baserat på tillståndsövervakning i form av temperatur- och vibrationsmätningar som tillsammans används för att uppskatta graden av slitage hos maskinen. Då HoloLens har stöd för trådlös uppkoppling finns potentialen att visualisera denna typ av data i realtid över maskinens virtuella motsvarighet. Applikationen kan dessutom integreras med Cybercoms industriella IoT-plattform `MachineBook`, där maskinerna publicerar sin driftstatus i ett format inspirerat av sociala media. [21]

Litteraturförteckning

[1] Krevelen, D. W. F (2007), *Augmented Reality: Technologies, Applications, and Limitations*, International Journal of Virtual Reality vol. 9, nr. 2, sid. 1-20.

[2] McBride, S. (2016, 23/5), *With HoloLens, Microsoft aims to avoid Google's mistakes*, Reuters. Webbsida. Hämtat 8/6 2017, från `http://www.reuters.com/article/us-microsoft-hololens-idUSKCN0YE1LZ`

[3] Thompson, C. (2015, 20/11), *Here's what it's like to buy a car with Microsoft's HoloLens*, Business Insider. Hämtad 8/6 2017 från `http://www.businessinsider.com/volvo-partners-with-microsoft-to-use-hololens-2015-11?r=US&IR=T&IR=T`

[4] Göteborgs Tekniska College (2017), *Smarta Fabriker*. Webbsida. Hämtad 2/6 2017, från `http://www.smartafabriker.se/info/projektet`

[5] Lee, K. (2012), *Augmented Reality in Education and Training*, TechTrends vol. 56 nr. 2, sid. 13.

[6] Microsoft (2017), *Mixed Reality*. Webbsida. Hämtad 8/6 2017, från `https://developer.microsoft.com/en-us/windows/mixed-reality/`

[7] SKF Group (27/2 2017), *Augmented Reality — Engineering access at your fingertips*. Blogginlägg. Hämtad 8/6 2017, från `http://skfgroup.tumblr.com/`

[8] Microsoft (2017), *HoloLens Hardware Details*. Webbsida. Hämtad 8/6 2017, från `https://developer.microsoft.com/en-us/windows/mixed-reality/hololens_hardware_details`

[9] Google (2016, 18/5), *Introducing Project Tango Area Learning - Google I/O 2016*. Youtube-video. Hämtad 8/6, 2017 från `https://www.youtube.com/watch?v=NTZZCtmR3OY`

[10] Holmdahl, T. (2015, 30/4) *BUILD 2015 — a closer look at the HoloLens hardware*. Blogg. Hämtad 8/6 2017 från `https://blogs.windows.com/devices/2015/04/30/build-2015-a-closer-look-at-the-microsoft-hololens-hardware/#qxyswSPQsw77syAh.97`

[11] Bright, P. (2016, 24/8), *Microsoft sheds some light on its mysterious holographic processing unit*. Webbtidning. Hämtad 8/6 2017 från `https://arstechnica.com/information-technology/2016/08/microsoft-sheds-some-light-on-its-mysterious-holographic-processing-unit/`

[12] Mukawa, H. et al. (2008)*A Full Color Eyewear Display using Holographic Planar Waveguides*, SID Symposium Digest of Technical Papers, Vol. 39 nr. 1, sid. 89–92.

[13] Cignoni, P. et al. (1997), *A comparison of mesh simplification algorithms*, Computers & Graphics vol. 22 nr. 1, sid. 37–54.
[14] Garland, M. och Heckbert, P. S. (1997) *Surface simplification using quadric error metrics*, Proceedings of the 24th annual conference on Computer graphics and interactive techniques, sid. 209-216.
[15] Wiebusch, E. och Latoschik, M. E. (2015), *Decoupling the entity-component-system pattern using semantic traits for reusable realtime interactive systems*, IEEE 8th Workshop on Software Engineering and Architectures for Realtime Interactive Systems, Art. nr. 7854098, sid. 25-32.
[16] Nyström, R. (2014), *Game Programming Patterns*, Genever Benning, sid. 213.
[17] Freeman, E. et al. (2003), *Head first design patterns, 1st edition*, O'Reilly Media, sid. 4.
[18] Unity Technologies (2017), *GameObjects*. Webbsida. Hämtad 7/6 2017 från `https://docs.unity3d.com/Manual/GameObjects.html`
[19] Microsoft (2017), *Readme.md*. GitHub-dokumentation. Hämtat 7/6 2017 från `https://github.com/Microsoft/HoloToolkit-Unity`
[20] Blender Foundation, *About Blender*. Webbsida. Hämtad 8/6 från `https://www.blender.org/about/`
[21] Cybercom Group (2017), *Machinebook*. Webbsida. Hämtad 7/6 2017 från `http://www.cybercom.com/what-we-do/internet-of-things/machinebook/`