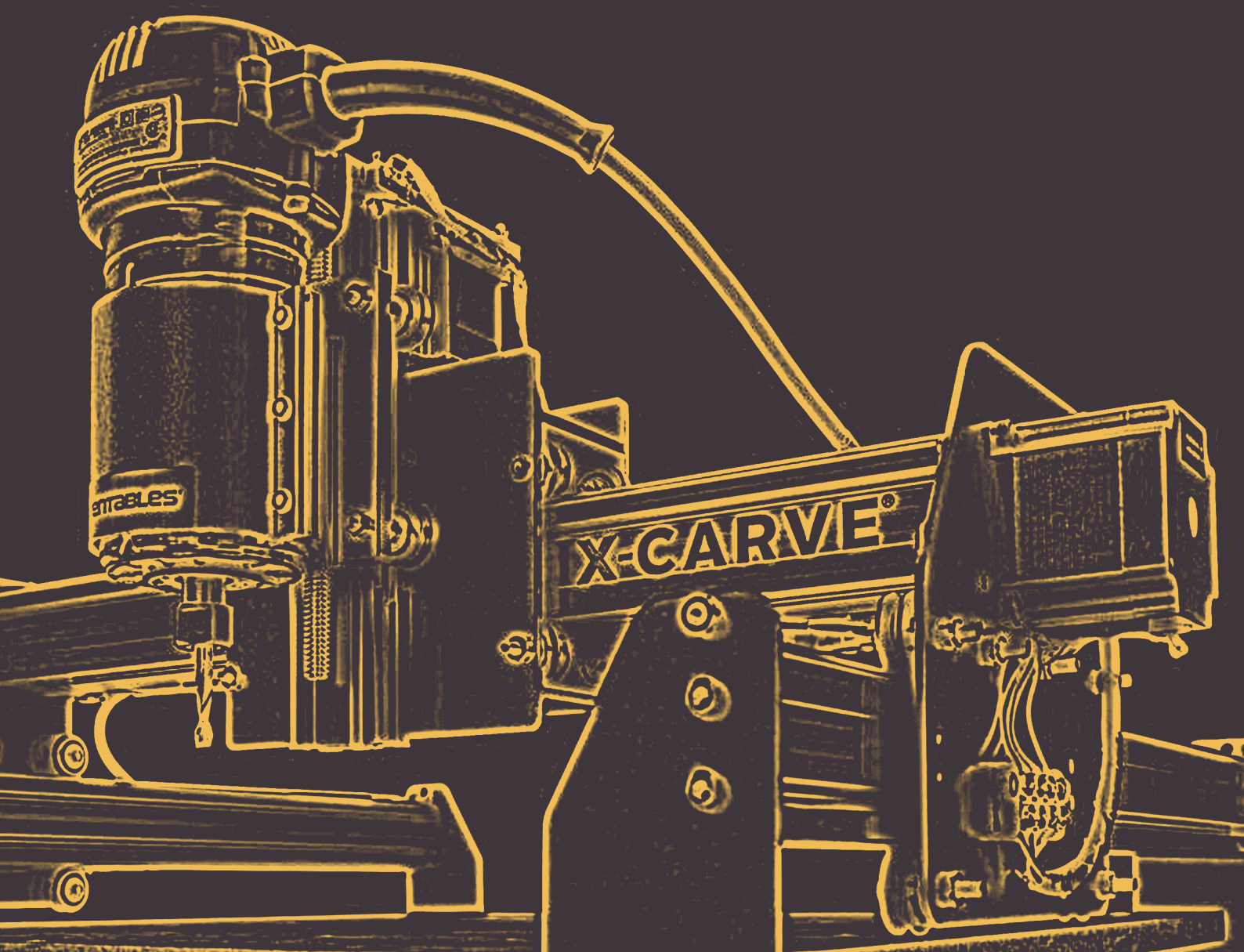
Ontwikkelen van een interface voor de X-Carve

Simon Huijink
19 Augustus 2016
Universiteit Twente
CTW Industrieel Ontwerpen



Titel: Het ontwikkelen van een interface voor de X-Carve
Auteur: Simon Johannes Henricus Huijink
Studentnummer: s1378457
Opleiding: Industrieel Ontwerpen

Datum inleveren verslag: 19-08-2016
Datum eindpresentatie: 26-08-2016

Beoordelings Commissie:
Voorzitter: prof. dr. ir. M.C. van der Voort
UT-Begeleider: ir. R.G.J. Damgrave

Voorwoord

Dit project is op mijn pad gekomen doordat ik op de site van de UT een opdracht staan over de X-Carve. Toen ik de opdrachtgever Roy Damgrave sprak bleek dat er al iemand anders met die opdracht bezig was namelijk Dylan Vogelsang. Ondanks dat mocht ik van Roy toch een project starten met dit apparaat en mocht ik daarvoor zelf de invulling van het project bedenken. Zeker bij het opstarten heb ik daarvoor veel hulp van Roy en Dylan gehad in het bedenken van mogelijke opdrachten en vertrouwd raken met het apparaat. Daarvoor wil ik hen beiden dan ook hartelijk bedanken.

Simon Huijink,
18 Augustus 2016

Inhoudsopgave

Samenvatting

De X-Carve is een open-source 3-assige CNC-frees geproduceerd door het bedrijf Inventables.De Universiteit heeft een X-Carve aangeschaft als tool voor studenten om zichtmodellen gemakkelijk te maken. De doelgroep van Inventables is voornamelijk hobbyisten die een goedkope freesmachine willen hebben voor graveren of uitsnijden van modellen. Hierdoor is enige kennis van hoe een CNC-machine werkt vereist om de X-Carve te besturen. Studenten hebben dit vaak niet dus weten dan niet hoe ze de X-Carve moeten aansturen en wat hij doet als hij eenmaal begonnen is.

Een andere bacheloropdracht word gedaan om een handleiding te ontwikkelen om studenten te leren hoe ze de X-Carve moeten gebruiken. Dit project is gericht op het ontwikkelen van een interface om studenten duidelijk te maken wat de X-Carve nou precies doet tijdens het frezen.

Hiervoor is uitgebreid analyse gedaan over de X-Carve, Cad-programma's en het gebruik van CAD-modellering in het onderwijs. Hier zijn eisen en functies uit voortgekomen en op basis daarvan is de interface gebouwd.

De interface kan de numerieke code genaamd G-Code lezen en visualiseren tot lijnen in een 3D omgeving. Daarnaast kan hij weergeven hoe de freeskop over deze lijnen beweegt. Dit zou nog verder uitgebouwd kunnen worden om een wisselwerking te creeren tussen de interface en de X-Carve zodat hij betrouwbaar de exacte beweging van de X-Carve kan weergeven.

Op basis van deze interface zijn nog een aantal mogelijke toekomstige projecten voortgekomen. Zo kan de interface omgebouwd worden om een regelsysteem te realiseren die de afwijking van de freeskop kan corrigeren. Ook zou hij nog verder omgebouwd kunnen worden om zelf G-Code te interpreteren en daarmee de X-Carve aan te sturen. Als laatste project zou de interface Augmented Reality kunnen gebruiken om via een camera en tablet de G-Code te visualiseren in de echte wereld.

Afbeelding 1 - De X-Carve

Summary

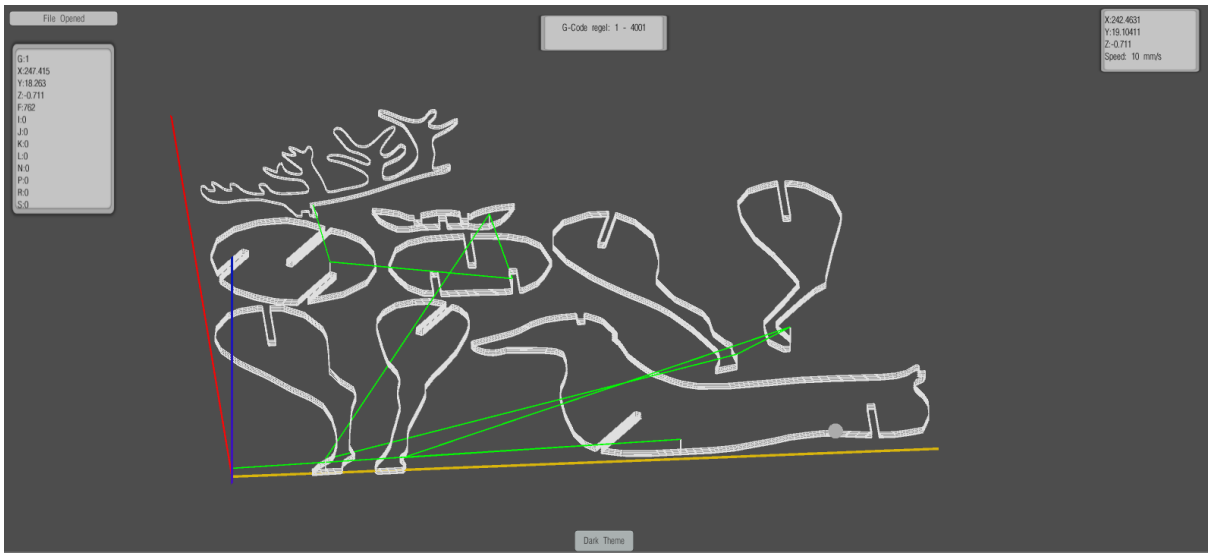
The X-Carve is an open-source 3-axis CNC-mill produced by the company Inventables. The University purchased the X-Carve as a tool for student to easily produce visual models of their projects. The targetgroup of Inventables is primarily hobbyists who want an affordable milling machine to engrave or cut out models. It is required to have some knowledge about how a CNC-machine works to be able to control the X-Carve. Most students do not possess this knowledge and thus do not know how to control it or what it does when they got it started.

Another bachelor assignment is being done to develop a manual of the machine to teach students how they can use the X-Carve. This project is focussed on developing an interface to show students what the X-Carve does while milling.

To achieve this there has been extensive analysis of the X-Carve, CAD-applications and the use of CAD-modelling in the curriculum. From this analysis demands and functions for the interface where produced. Based on these demands and functions the interface was build.

The interface that was build can read numerical code called G-Code and visualise it to lines in a 3D environment. It can also visualise how the mill-head would move over these lines. This could be further developed to create communication between the interface and the X-Carve so the the interface can give a more reliable visualisation of the exact movement of the X-Carve.

On the basis of this interface there is also the possibility of some future projects. The interface can be enhanced to become a control system that could correct the deviation the X-Carve makes. He could even be further enhanced to interpret al the g-code and be able to send it to the X-Carve and so becoming the controller of the X-Carve. The final future project is developing the interface with an Augmented Reality feature that can project the g-code in the real world via a camera and tablet.



Afbeelding 2 - De Interface

Hoofdstuk 1
Inleiding

Veel eerstejaars studenten Industrieel Ontwerpen moeten al bij hun eerste project al meteen beginnen met het maken van zichtmodellen. Vaak worden deze model lasergesneden en in elkaar gevouwen. Dit maakt al deze modellen vierkant en weinig creatief. Om studenten de kans te geven om makkelijk en snel gekromde vlakken te maken is daarom de X-Carve aangeschaft.

De X-Carve is een goedkope en open-source 3-Assige CNC-Frees. Hij is voornamelijk geschikt om hout en schuim uit te frezen naar de vorm die een student wil. Als de student een CAD-model van zijn ontwerp heeft gemaakt zou hij het via een aantal stappen kunnen laten uitfrezen.

De frees heeft op het moment alleen nog een aantal problemen waardoor het voor een student met weinig kennis over CNC machines en numerieke besturing bijna onmogelijk is om hiervan gebruik te maken. Het doel van dit project is om dan ook bij te dragen aan het stroomlijnen van het proces.

Om studenten meer te leren over welke stappen ze moeten ondernemen om de frees te bedienen loopt al een project om een interactieve handleiding te maken. Dit project is dan ook bedoeld om te ondersteunen in het gebruik. Het doel is dan ook om een interface te ontwikkelen die de gebruiker een beeld geeft van wat de frees doet nadat de code er naar toe is gestuurd.

Hiervoor is het project in twee doelen opgesplitst.

- **Analyse doen van de omgeving waarin de interface moet opereren.**

Hiervoor is de X-Carve zelf geanalyseerd op zijn hardware en software. Daarnaast is er een uitgebreide analyse gedaan van welke stappen er nog voor komen om de X-Carve aan te sturen. En omdat het in de context van onderwijsdoeleinden geplaatst moet worden is er analyse gedaan van waar CAD modellering voorkomt in het curriculum en voor welke doeleinden. En om het beeld van de omgeving waar deze interface in moet werken is er een net gemaakt van alle software pakketten die gebruikt worden en hun interactie daarvan.

- ***Het ontwerpen van een interface en daarvan een prototype maken***

Dit is het uiteindelijke doel van het project om een werkend prototype te hebben van de interface die de numerieke code kan lezen en kan visualiseren. In dit verslag zal verder ingegaan worden over hoe de interface is opgebouwd en hoe de code deze functie kan vervullen.

Hoofdstuk 2
Analyse

Om een goed beeld te krijgen van welke factoren in het ontwikkelen van deze interface van belang zijn zijn er een aantal analyses gedaan. Deze analyses en hun subonderdelen zijn:

- Analyse van de X-Carve
  - Hardware
  - Software
  - Huidige problemen
- CAD/CAM toepassingen
  - CAD-toepassingen
  - De beschikbare CAD-pakketten
  - CAM toepassing
- Curriculum Analyse
  - Modules en CAD-toepassingen daarin
  - De beschikbare software voor verschillende toepassingen
  - interactie tussen CAD-pakketten en andere pakketten.
- Gebruikersanalyse

2.1 Analyse X-Carve

De X-Carve wordt geproduceerd door een bedrijf genaamd Inventables. Inventables is een Amerikaans bedrijf gevestigd in Chicago. Het bedrijf is gestart door Zach Kaplan in 2002 en is sindsdien uitgegroeid tot een bedrijf met ongeveer 40 werknemers. Het bedrijf produceert twee CNC frezen genaamd de X-Carve en de Carvey. Daarnaast hebben ze hun eigen aansturingsapplicatie ontwikkeld genaamd Easel. De universiteit heeft een X-Carve aangeschaft dus in de verdere analyse zal dit apparaat en de verschillende software die hem kan aansturen onderzoeken. Carvey zal verder buiten beschouwing gelaten worden.[1]

De X-Carve is een 3-Assige CNC-Frees. CNC staat voor Computer Numerical Control en betekent dat de frees door een computer word aangestuurd in plaats van een klassieke frees die met de hand word bestuurd. De X-Carve werkt met een freeskop die door middel van rails, riempjes en drie steppermotoren die worden aangestuurd vanuit een Arduino. Deze onderdelen zullen wat uitvoeriger worden besproken

2.1.1 Hardware

De freeskop

De X-Carve word aangeleverd met een DeWalt 611 die vast wordt gehouden in een beugel. Frezen is een proces waarbij een frees (en soort van boor) over het oppervlakte word bewogen om materiaal weg te snijden. Twee factoren die hierbij van belang zijn de toeren per minuut en de snijdiepte van de frees. De DeWalt 611 kan een toerental aan tussen de 16.000 en 27.00 toeren per minuut op een diepte van 35 mm. Dit is de frees die met de X-Carve word meegeleverd maar de medewerkers van de werkplaats hebben besloten om er een iets zwaardere frees op te zetten namelijk de Makita RT0700C. Deze kan een toerental tussen de 10.000 en 30.000 toeren per minuut aan ook op een freesdiepte van 35 mm. Veel verschil zit er dus niet tussen deze frezen maar de Makita is iets beter te controleren.

De rails en riemen

De freeskop zit vast in een beugel die bevestigt zit aan een kleine verticale rails waar de kop direct word aangestuurd door een steppermotor zodat hij over de z-as kan bewegen. Deze kopbevestiging leunt met een aantal wieltjes op een rails die horizontaal is geplaats en weer leunt op 2 andere rails. Door middel van steppermotoren die bevestigt zijn aan de riemen kan de freeskop over de rails bewogen worden.

De freeskop kan dus over 3 assen bewegen dus hij heeft drie vrijheidsgraden namelijk translaties over de x-,y- en z-as. De X-Carve is voornamelijk bedoelt om gebruikt te worden als 2.5-assige frees. Dit betekent dat hij in lagen werkt namelijk eerst een beweging in de z-as en daarna bewegingen in de x- en y-as. Hij freest in dat geval dus niet in 3 richtingen tegelijkertijd. Hij kan uiteraard wel deze beweging maken alleen niet in één keer ,deze schuine beweging zou namelijk teveel weerstand opleveren. Wat hij dan doet is eerst een benaderen van de vorm in lagen en daarna de schuine beweging als afronding.

De Steppermotoren

Een Steppermotor is een elektrische motor die zoals de naam als doet vermoeden stappen maakt. Hij werkt dus niet in afstanden maar in hoeveelheden van stappen. Als de freeskop bijvoorbeeld van coordinaat (0,0) naar (10,5) moet dan stuurt de Arduino een signaal om hem zoveel stappen te laten maken om dat te bewerkstelligen. Bij een stapgrootte van 1 zou dat dus 10 stappen voor de X-as steppermotor en 5 stappen voor de Y-As steppermotoren.

Steppermotoren zie je veel bij 3D printers omdat ze een redelijk hoge precisie hebben en makkelijk te besturen zijn. Bij veel goedkope CNC applicaties zie je ze ook. Het probleem is dat een steppermotor niet kan controleren of hij daadwerkelijk een bepaalde afstand heeft gemaakt. Bij een proces als frezen spelen er veel meer weerstandskrachten dan bij 3D printen en dit kan er voor zorgen dat de motor kleinere stappen maakt dan hij zou moeten. Dit is waarom duurdere CNC applicaties meestal servo-motoren gebruiken die niet in stappen maar in rotatieafstand werkt.

De steppermotoren op de X-Carve zijn van de soort Nema 17. Dit zijn 2.8 Volt motoren en zijn met een maximaal moment van 0.44 Nm redelijk licht. Dat bete-

kent dat de frees niet te snel door het materiaal heen kan gaan omdat de motor niet zoveel kracht kan leveren. De X-Carve kan ook besteld worden met een iets zwaardere motor namelijk de Nema 23 deze kan namelijk bijna een moment van 1 Nm aan.

Arduino

Arduino is een open source electronica platform. De Arduino bestaat uit twee delen namelijk de hardware in de vorm van printplaten en de software die op de microcontroller staat. De printplaten zijn een Arduino Uno en een GRBLShield. De Arduino Uno is in feite een microcontroller met een aantal poorten. Een van deze poorten is een USB-poort die verbinding maakt met een computer en daarvan numerieke code kan ontvangen.[2] Deze zet de microcontroller om naar een ander signaal dat de GRBLShield kan begrijpen. De GRBLShield is een printplaat die het signaal van de Arduino Uno interpreteert en via elektronische signalen de stepper-motoren aanstuurt. [3]

2.1.2 Software

Easel

Inventables hebben hun eigen software ontwikkeld namelijk Easel. Easel is een gratis webapplicatie waar mensen hun ontwerp kunnen uploaden en om laten zetten naar G-Code die de X-Carve of de Carvey begrijpt. De gebruiker kan eigen vormen of tekst maken in dit programma. Dit is prima als je iets simpels wil laten uitfrezen. Voor ingewikkeldere vormen kan de gebruiker SVG of G-Code bestanden inladen.

SVG staat voor Scalable Vector Graphics. Dat betekent dat het niet meer is dan een vector afbeelding. Normale afbeeldingen werken in het algemeen in pixels terwijl een vectorafbeelding werkt in locaties. Dus als je een vectorafbeelding groter of kleiner maakt blijft de afbeelding scherp omdat de resolutie meeschaalt. Als een gebruiker dus een vector afbeelding maakt van wat hij wil uitfrezen dan kan hij dat prima om laten zetten naar numerieke code. Het probleem hiermee is dat SVG alleen maar X en Y locaties aankan. Hierdoor kan je geen gekromde vlakken maken.[4]

G-Code daarentegen werkt wel over alle drie de assen en kan dus ingewikkeldere vormen uitdrukken. Easel zelf doet dan eigenlijk weinig meer met de code op zich behalve het naar de Arduino sturen. G-Code zal later in dit verslag nog verder worden uitgezocht aangezien het interpreteren van de G-Code een cruciaal onderdeel is van de interface

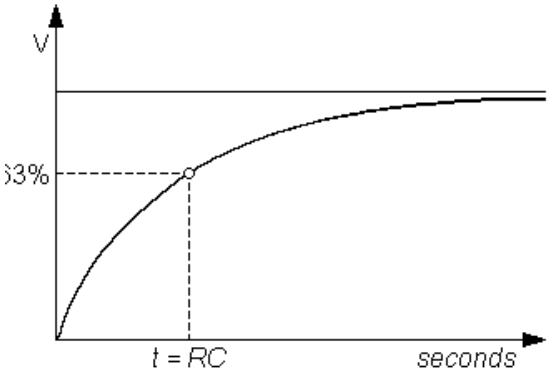
GRBL
GRBL is het besturingsysteem dat op de Arduino staat en daarmee de GRBLShield aanstuurt en daarmee dus de steppermotoren. Er zijn twee grote besturingsystemen voor open source CNC Applicaties namelijk GRBL en TinyG. Beide worden veel gebruikt in 3D Printers en frezen. GRBL kan niet alle G-Code interpreteren maar de standaard elementen begrijpt hij wel. Zo kan hij wel rechtlijnige en cirkelvormige bewegingen maken en kan hij snelheden van de steppermotoren of de frees aanpassen. Ingewikkeldere zaken als koelvloeistof of afzuiging kan hij niet besturen.[5]

ChiliPeppr & Universal G-Code Sender
In het geval dat de gebruiker zelf al G-Code gegenereerd heeft in een CAM programma dan is Easel niet meer dan een doorgeefluik naar de Arduino. Hiervoor moet de gebruiker een speciale compiler downloaden van Easel en kan als het in Easel staat weinig meer doen met de code. Daarom is het ook logischer om een alternatief te gebruiken namelijk ChiliPeppr of UGS (Universal G-Code Sender). Deze laden ook G-Code in maar even de gebruiker nog de kans om dingen als startlocaties of snelheden in te stellen. Daarnaast zijn deze iets inzichtelijker tijdens het frezen aangezien ze aangeven hoe lang de frees nog bezig zal zijn.

2.1.3 Huidige problemen
Na wat testen met de X-Carve zijn er meteen een aantal problemen aan het licht gekomen.

De X-Carve is nog erg onnauwkeurig
Tijdens het frezen komt er een grote onnauwkeurigheid in de frees. De X-Carve corrigeert hier niet zelf voor aangezien de steppermotoren niet zelf kunnen meten wat hun daadwerkelijke positie is en dus niet weten of ze daadwerkelijk de goede afstand hebben afgelegd. Er zijn verschillende mogelijke oorzaken die tot dit probleem hebben kunnen leiden:

- De kop heeft enige speelruimte in zijn beugel en kan tijdens het frezen dus iets bewegen. Dit is alleen niet in dezelfde orde van grootte als de afwijking.
- De riemen kunnen slippen. De riemen zijn van rubber en behoorlijk dun dus bij een te grote kracht kan hij gaan slippen waardoor niet alle rotatie van de steppermotor word omgezet in een translatie van de kop.
- GRBL houd geen rekening met de opstarttijd van de steppermotoren. Als de motor een signaal ontvangt dat hij moet bewegen kost het heel even om de vereiste snelheid te ontvangen (oftewel zijn stapresponsie). GRBL gaat ervan uit dat hij onmiddellijk op snelheid is.
  De kop maakt alleen deze maximale snelheid als hij van punt verplaatst tussen freesacties door. Dit kan voorkomen worden door middel van de maximale snelheid te verlagen aangezien de stapresponsie dan veel minder tijd inneemt. De beweging op zich neemt dan wel meer tijd in.



Afbeelding 3- Stapresponsie

Het gebruik van de X-Carve is ingewikkeld

De X-Carve maakt gebruik van een aantal software pakketten. Het begint bij een CADprogramma die de geometrie van het model bepaalt. hierna word er met een CAM programma een simulatie gemaakt van het bewerken van materiaal om tot het model te komen en word dat uitgedrukt in g-code. Deze g-code word dan in Easel, Chilipeppr of UGS verstuurd naar de Arduino waar GRBL deze g-code omzet in mechanische bewegingen.

Studenten krijgen tijdens de bachelor veel onderwijs in het modelleren in het CAD programma Solidworks maar word weinig aangeleerd over CAM en g-code. Hierdoor weten weinig studenten hoe ze daadwerkelijk gebruik kunnen maken van de X-Carve of andere freesmachines na het maken van een model en kiezen er vaak voor om het dan toch te lasersnijden en na te bewerken.

Het is onduidelijk wat de X-Carve doet na het inladen van de G-Code

Wanneer vanuit Easel g-code verstuurd word kan de gebruiker wel van te voren zien welke bewegingen de freeskop zal maken en hoe lang dat gaat duren. Maar tijdens het frezen is er geen enkele feedback vanuit de X-Carve. De gebruiker weet dus niet waar de frees zou moeten zijn op elk gegeven moment. Hierdoor is het dus ook onduidelijk wat er fout gaat als er iets fout gaat. De oorzaak van het probleem van de onnauwkeurigheid kan dus ook moeilijk gevonden worden omdat er niet gezien kan worden wanneer de onnauwkeurigheid precies optreed. Als daar meer inzicht in gekregen kan worden dan zou er gezien kunne worden of de onnauwkeurigheid optreed in het wisselen van positie (de stapresponsie) of tijdens het frezen (slippen en verdraaien van onderdelen door krachten).

2.2 CAD/CAM analyse

2.2.1 CAD software

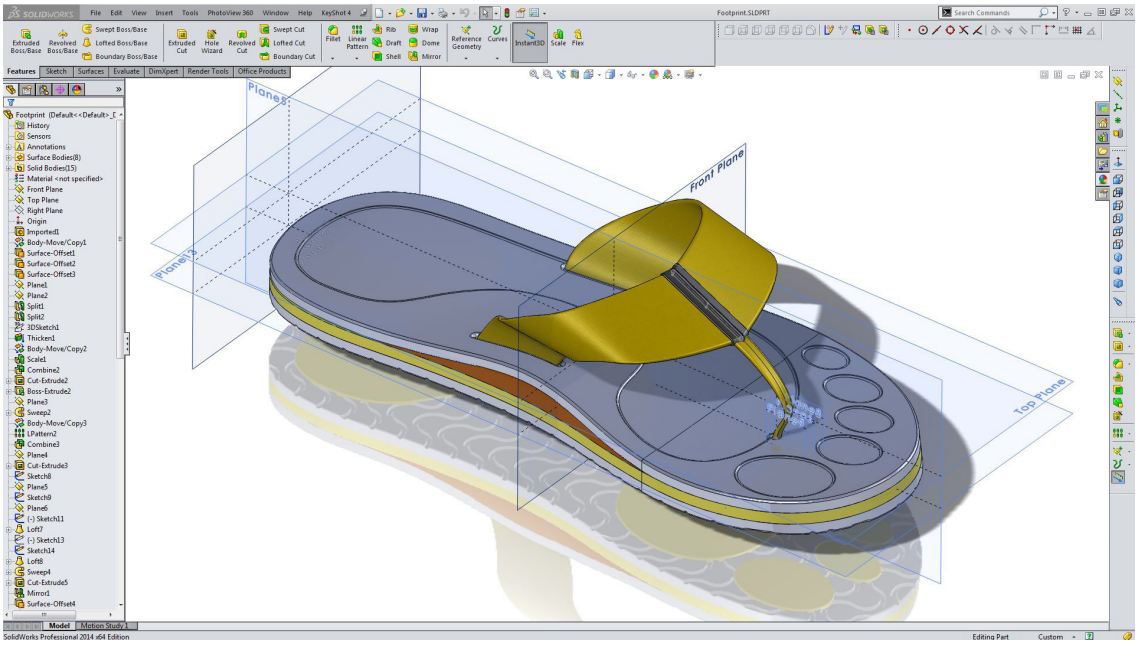
Computer Assisted Design software zijn programma's die ontwerpers in staat stellen digitale 3D modellen van hun ontwerpen te maken. Deze 3D modellen kunnen hierna gebruikt worden voor allerlei verschillende toepassingen als analyses, animaties of visualisaties. Tegenwoordig is CAD-software niet meer weg te denken uit het ontwerpproces (zeker voor industriële toepassingen).

De universiteit bied onderwijs aan voor het CAD-programma 3DS Solidworks. Dit wordt in het eerste jaar van de bachelor al aangeleerd en wordt dan ook door alle Industrieel Ontwerp studenten gebruikt. In de master komen studenten nog in aanraking met een wat groter programma Siemens NX. Omdat het niet zeker is dat de universiteit de komende jaren bij solidworks blijft is er analyse gedaan van welke CAD programma's er zijn en hoe deze interacteren met andere software pakketten die studenten gebruiken. Hiervoor zal er eerst kort de verschillende CAD-pakketten en hun CAM toepassingen langs worden gegaan. Daarna is er nog een analyse van de andere toepassingen van CAD modellen in het onderwijs en de interactie tussen de verschillende softwarepakketten.

De software pakketten die vergeleken zullen worden zijn 3DS Solidworks, Siemens NX en Autdesk Fusion 360. het zijn allemaal Solid Modelling programma's. Dat betekent dat er vanuit schetsen vormen worden opgebouwd. Deze vormen worden daarna als solide blokken gezien. Deze solide blokken kunnen daarna gebruikt worden voor berekeningen.

3DS Solidworks

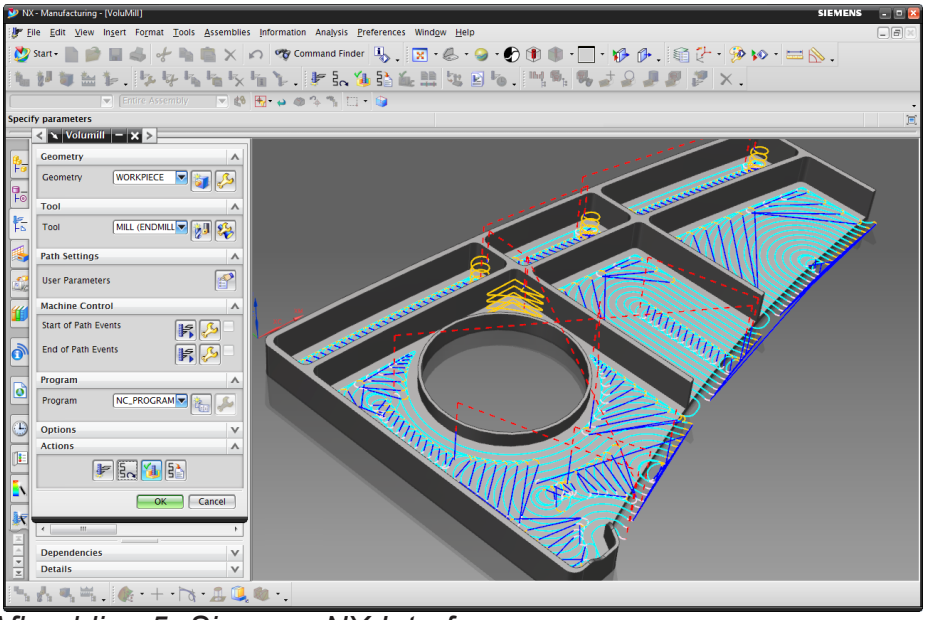
Solidworks word ontwikkeld door een Frans bedrijf genaamd Dassault Systems en heeft op het moment meer dan 2 miljoen gebruikers. Het is op het moment de standaard bij de universiteit. het programma word bij veel kleinere ontwerpbureau's gebruikt. De grotere ontwerpbureau's gebruiken voornamelijk het uitgebreidere CAD-Pakket Catia van Dassault Systems.[6]



Afbeelding 4 -Solidworks Interface

Siemens NX

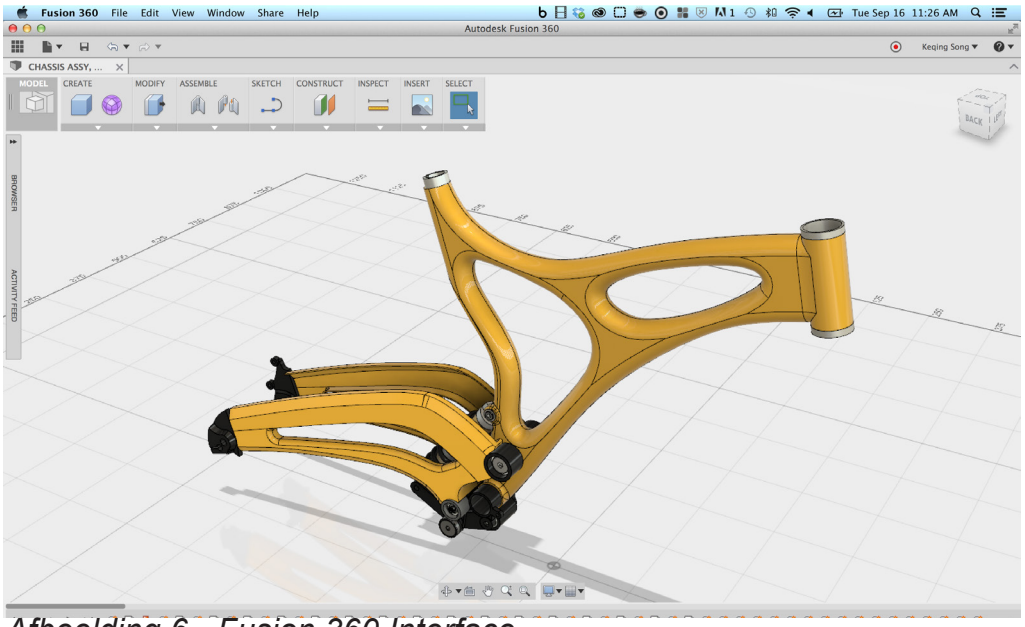
NX is door Siemens zelf ontwikkeld als CAD programma's omdat Siemens niet meer afhankelijk wilde zijn van CAD ontwikkelaars voor functionaliteit in hun proces. NX is in principe een wat makkelijkere en beperktere versie van SolidEdge (ook van Siemens natuurlijk). In vergelijking met Solidworks is het een wat completer en exacter pakket als het gaat om simulaties, analyses en CAM-toepassingen. Maar het is daardoor ook een stuk ingewikkelder dan Solidworks.



Afbeelding 5- Siemens NX Interface

Autodesk Fusion 360

Fusion 360 is een redelijk nieuw programma dat nog steeds hevig in ontwikkeling is. Het is een erg makkelijk CAD programma om in te stappen omdat er niet enorm veel functionaliteit aan vast zit. Daarnaast levert Autodesk veel gratis les- en oefenmateriaal vrij. Voor applicaties als analyses zijn er geen ingebouwde applicaties maar heeft Autodesk veel andere software pakketten. Deze zijn wel goed met elkaar uit te wisselen. Fusion 360 heeft daarnaast in vergelijking met Solidworks en NX een veel gemakkelijkere manier van CAM waardoor tijdens dit project al het CAM-werk in Fusion 360 heb gedaan terwijl veel van het modeleren in Solidworks werd gedaan.

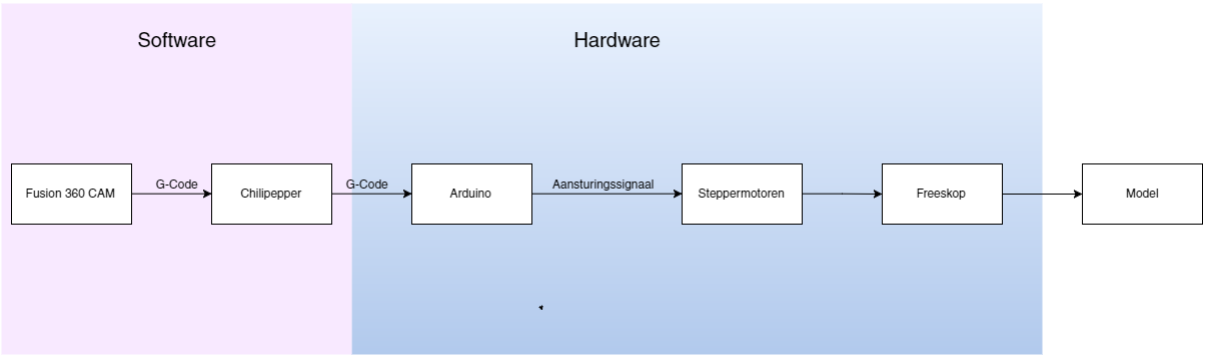


Afbeelding 6 - Fusion 360 Interface

2.2.2 Van CAD-model tot fysiek model

Om van CAD model naar een normaal model te gaan moeten er een aantal stappen doorlopen worden. Deze stappen zijn tijdens dit project op de volgende manier doorlopen

- Maken van een CAD model
  - De CAD-modellen zijn in Solidworks gemaakt. Hier wad de meeste ervaring mee waardoor dit de efficientste manier was.
- CAM genereren
  - Voor het genereren van CAM wordt het Solidworksmodel geëxporteerd naar Autodesk Fusion 360 omdat deze veel makkelijker is om in te stellen voor een CAM project.
  - In Fusion 360 worden alle stappen die doorlopen moeten worden vastgesteld.
  - Deze stappen in de geometrie worden geexporteerd als G-Code.
- G-Code naar Arduino versturen
  - Hiervoor werd voornamelijk Chilipeppr gebruikt.
- De X-Carve het model laten uitfrezen
  - De Arduino ontvangt de G-Code en zet deze om in bewegingen van de frees
  - De Arduino stuurt de steppermotoren aan om het model uit te frezen
  - De frees freest het model uit.



Afbeelding 7 - Diagram van het CAD/CAM proces

2.3 Curriculum analyse

Binnen het onderwijs word er veel gebruik gemaakt van CAD modellering. Voornamelijk bij projecten is het nodig dat er ofwel een visualisatie van een product moet komen. Dit kan ofwel door middel van het renderen van het model of er in de werkplaats een fysiek zichtmodel van te maken. De X-Carve is aangeschaft als onderdeel in het maken van een fysiek model. Op het moment maken student voornamelijk op 2 manieren fysieke zichtmodellen:

- Opbouwen in lagen uit de lasersnijder. De student deelt zijn model op in lagen en snijd deze uit MDF uit in de lasersnijder. Hij lijmt ze op elkaar en schuurt daarna bij. Dit proces is redelijk exact als de student bij het schuren niet teveel materiaal weghaalt.
- Schuimblokken bewerken. Voor een wat sneller zichtmodel kan de student uit schuim zijn vorm schuren. Ook al meet de student de afmetingen goed uit is het erg moeilijk om gekromde vlakken hierbij goed te krijgen.

Beide methodes zijn vrij tijdsintensief. Het opbouwen in lagen heet een lijmlaag die tijd nodig heeft om op te drogen en voor het schuim is een student lang bezig om alles precies goed te schuren. De X-Carve is aangeschaft om dit probleem op te lossen. De freesmachine kan namelijk ook in 3D werken in plaats van de lasersnijder die alleen in 2D werkt. Het is zo dat er al een CNC frees in de werkplaats aanwezig is maar deze is erg duur en ingewikkeld aangezien het een 5-assige frees is. Studenten kunnen deze niet zomaar gebruiken zonder veel begeleiding van werkplaatsmedewerkers.

2.3.1 Module analyse

Om een volledig beeld te krijgen waar en wanneer CAD toepassingen worden gebruikt in het curriculum zal hieronder elke module van de bachelor worden doorlopen en aangestipt hoe CAD in deze module voorkomt.

Module 1 Introduction ID

studenten moeten in deze module al een model bouwen voor project “Kick Start”. Ze hebben alleen nog geen ervaring met CAD software dus velen maken gebruik van Google Sketch Up (een makkelijk gratis 3D modeleerprogramma) of van ontwerptekeningen die ze omzetten naar werkplaatstekeningen.

Module 2 Ideation

In deze module krijgen de studenten het vak Technisch Product Modelleren wat ze leert om modellen te bouwen in Solidworks en die om te zetten naar bouwtekeningen. Voor het project Ideation moeten ze ook weer een zichtmodel bouwen. sommige studenten maken hiervoor al gebruik van Solidworks.

Module 3 Realisation of Products

in deze module word een ontwerp van het vorige project uitgebouwd tot een werkend prototype. Dit betekent dat er gedetailleerde CAD modellen moeten komen en deze uitgebouwd moeten worden tot uitgebreide werkplaatstekeningen.

Module 4 Smart Products
In deze module word weinig gewerkt met CAD toepassingen. Sommige studenten zouden er voor kunnen kiezen om gebruik te maken van Solidworks voor hun model voor het vak Design and Styling maar weinigen doen dat.

Module 5 Human Product Relations
in deze module zijn twee onderdelen die gebruik maken van CAD namelijk weer het project voor een zichtmodel/rendering en het vak Production 3 wat CAD modellen gebruikt om spuitgietmodellen te creeeren.

In combinatie met het programma Autodesk Moldflow maakt de student analyses over hoe het model spuitgegoten kan worden. Hier kan hij informatie uit halen over hoe de flow van het materiaal gaat en waar problemen op kunnen treden.

Module 6 Consumer Products
In deze module word Technisch Product Modelleren 2 gegeven waarbij de student meer leert over het opzetten van een grootschaliger modelleer project en de beginselen leert van in-context-modelling. In-context-modelling houd in dat het 3D model zichzelf kan aanpassen aan de context. Dus als 1 onderdeel van afmeting vernadert de rest van de onderdelen mee kunnen veranderen.

Daarnaast is zit er in deze module weer een project waarbij een zichtmodel moet worden gemaakt en dit is meestal een redelijk complex product dus word er een CAD model gemaakt met veel verschillende onderdelen.

Module 7 Design for Specific Users
In deze module zit een project waarvoor een model gemaakt moet worden maar hiervoor hoeft niet per se een CAD model te worden gebruikt. Wel word er bij het vak Design Sketching aandacht besteed aan hoe CAD modellen gebruikt kunnen worden om snel vormen na te kunnen tekenen in perspectief.

Module 8 Virtual Product Development
het vak “Introduction to Finite Element Methods” leert studenten hoe de theorie van krachtberekeningen voor ingewikkelde structuren werken. Voor krachtberekeningen in Solidworks of andere CAD programma’s word deze methode veel gebruikt.

Het project geeft daarnaast veel vrijheid in wat de opdracht en deliverables zijn. Wel worden studenten geïntroduceerd met Unity wat een programma is bedoeld om games te maken. unity kan ook gebruikt worden om een model of omgevingn te visualiseren.

Module 9 & 10 Minor

Omdat alle minors verschillen zal hierbij niet gekeken worden naar CAD toepassingen in alle minors

Module 11 Systems in Context

In deze module word weinig gebruik gemaakt van CAD modellering. Wellicht om wat te visualiseren in het project maar hier hoeft geen compleet model van te komen.

Module 12 Bachelor Assignement

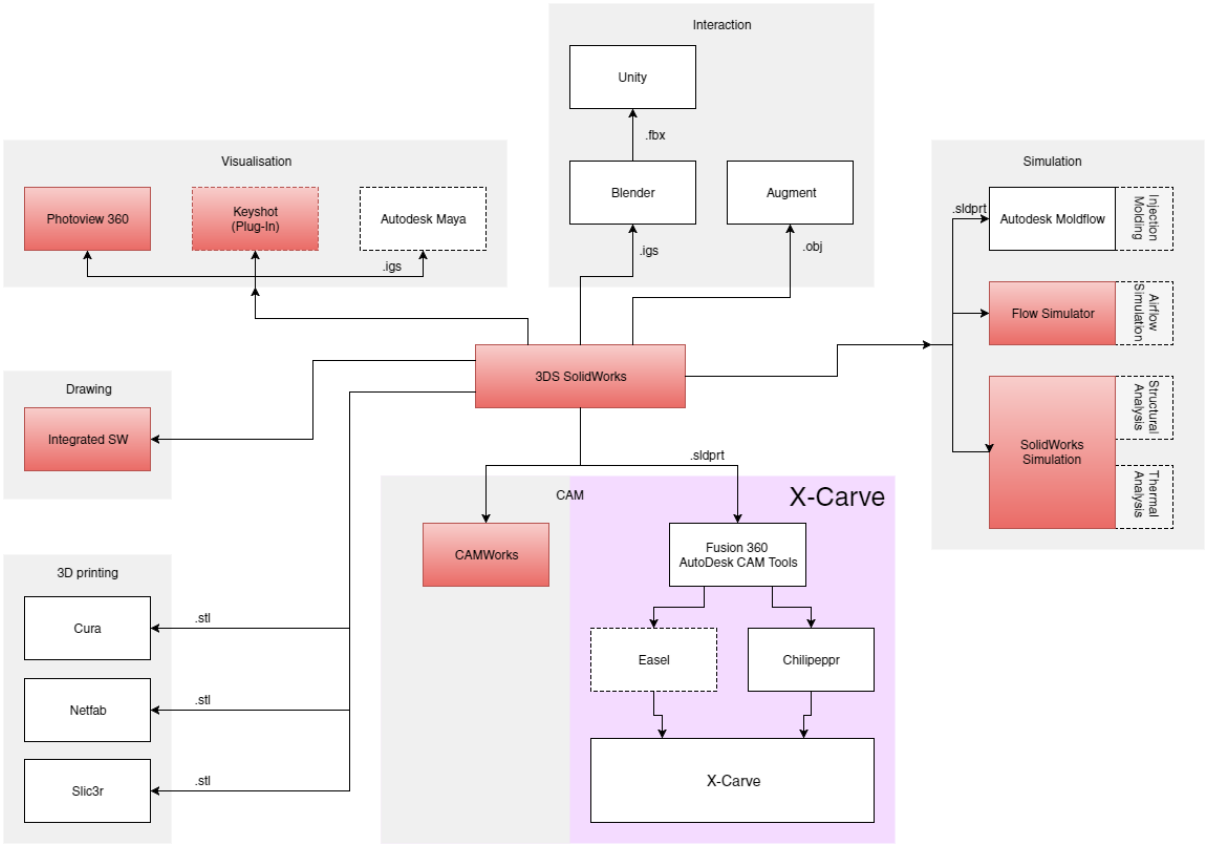
Omdat de Bachelor Assignment compleet verschillend is voor elke student kan hier niet met zekerheid gezegd worden dat er gebruik gemaakt zal worden van CAD programma's

In de master zijn er nog maar een aantal vakken die echt dieper in gaan op CAD modellering. Deze leren studenten verder over 3D print applicaties, Simulaties, Analyses en Collaboration in grote modelleerprojecten.

Afbeelding 8 - Het Bachelor Industrieel Ontwerpen Curriculum

2.3.2 Software Interactie

Om een compleet beeld te krijgen van hoe al eze toepassingen van CAD in het onderwijs onderling samenwerkt zijn er diagrammen gemaakt waarin alle software pakketten verzameld zijn. Het diagram voor solidworks is op de volgende pagina te vinden. De diagrammen voor alle drie de CAD programma's zijn te vinden in de bijlage.



Afbeelding 9 - CAD interactie diagram van Solidwors

2.4 Gebruikersanalyse

Het is belangrijk bij het ontwerpen voor de interface om te weten wie gebruik zal maken van de X-Carve. We hebben het tot nu toe voornamelijk over studenten gehad en dit zal ook de primaire groep gebruikers zijn. Deze zullen weinig ervaring hebben met freesmachines en weinig ervaring met CAM-applicaties of G-code. Een tweede groep gebruikers zijn de werkplaatsmedewerkers. Zij hebben daarentegen juist wel veel ervaring met computergestuurde machines en zullen daarom misschien ook meer informatie uit de interface willen halen.

Na gesprekken met werkplaatsmedewerkers blijkt dat zij een aantal onderdelen van belang vinden als het gaat om de X-Carve:

- Hij moet veilig zijn aangezien zij verantwoordelijk zijn voor de veiligheid van de studenten.
- Hij moet door studenten alleen aangestuurd kunnen worden zodat zij er geen extra tijd aan moeten besteden om te begeleiden. Dus als de interface daarin kan werken
- Informatie over snelheid van de frees zouden ze interessant vinden. De X-Carve kan alleen niet vanuit de Arduino de freessnelheid aanpassen dus dat is niet te implementeren in de interface.

Aangezien de interface voornamelijk bedoelt is voor studenten die niet per se ervaring hebben met dan wel CNC machines of modelbouwen moet de interface redelijk voor zich spreken. Als de student het bestand heeft ingeladen moet meteen duidelijk zijn wat de frees gaat doen en hoe dat er uit gaat zien. Er moeten daarnaast geen ingewikkelde vragen of instellingen zijn die de student moet invoeren. Dit zou ook geen probleem moeten zijn omdat de student bijna alle informatie al heeft ingevoerd bij het maken van de g-code.

Hoofdstuk 3
Eisen Functies

3.1 Eisen

Uit de analyse is het proces van het frezen in kaart gebracht. Hierdoor kunnen we de interface hier naast plaatsen in het schema waardoor we kunnen zien hoe de wisselwerking tussen de interface en de andere onderdelen van het systeem zal zijn. Door middel van dit schema kunnen we eisen opstellen voor het visuele gedeelte van de interface (de front-end) en de code die daarachter ligt (de back-end).

3.1.1 Front End Eisen

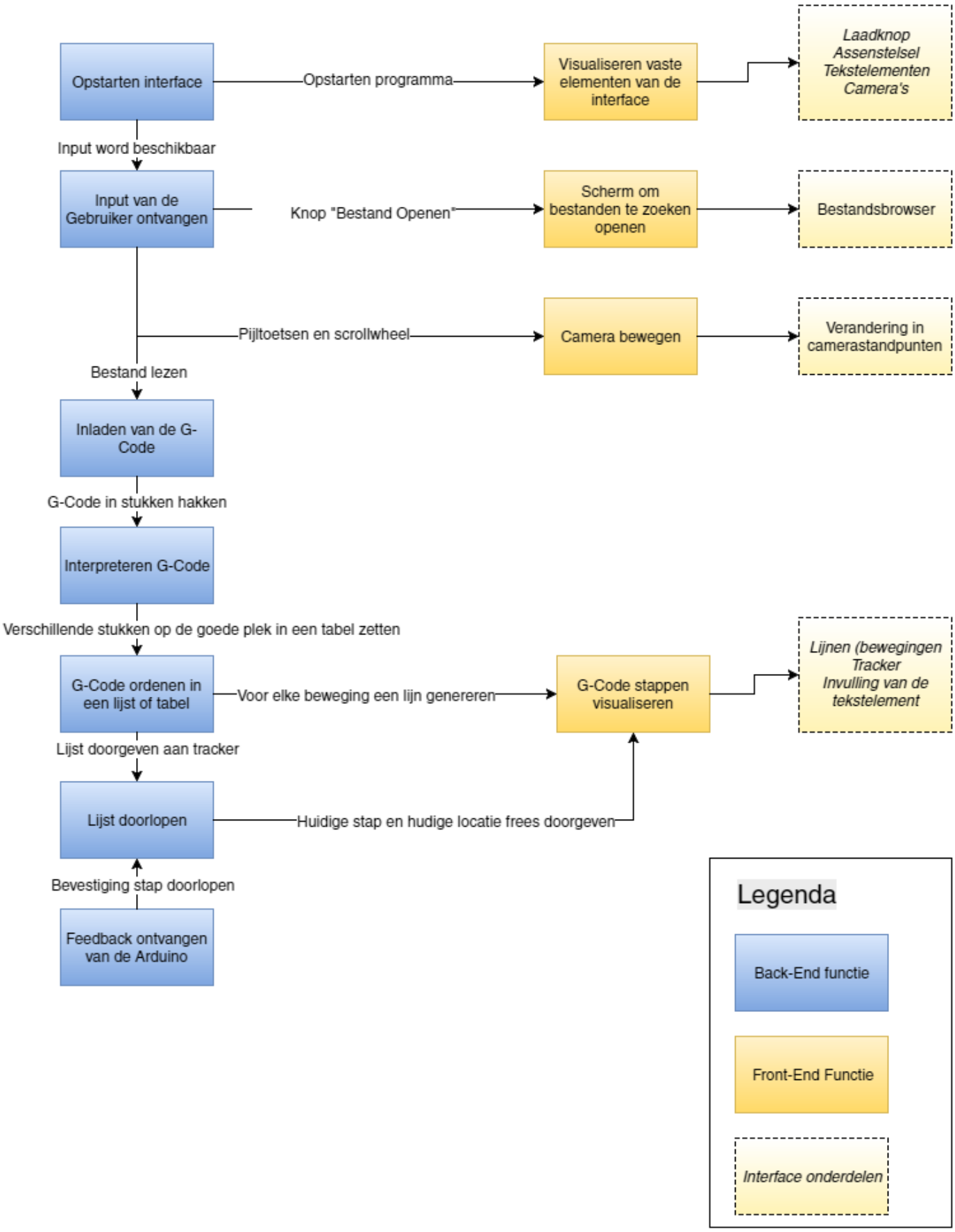
- De interface moet ten minste geopend kunnen worden op een windows computer.
- De interface
- De interface moet G-Code kunnen inladen door middel van een knop.
- De interface moet de beweging van de frees kunnen visualiseren.
- De interface moet de huidige locatie van de frees kunnen visualiseren.
- De interface moet de snelheid van de frees aangeven.
- De interface moet zowel een 2D als een 3D aanzicht kunnen geven.

3.1.2. Back End Eisen

- De Back-End moet G-Code kunnen inladen en deze interpreteren
- De interpretatie van de G-Code moet kunnen worden omgezet in een lijst van acties
- Acties relevant voor de interface moeten doorgegeven worden aan de fronte-end om te visualiseren
- De back-end moet door deze acties heen kunnen lopen om de locatie van de frees te kunnen bepalen.
- De back-end moet verbinding hebben met de Arduino om feedback te ontvangen om van elke regel g-code bevestiging te hebben dat hij uitgevoerd is.
- De back-end moet voldoende commentaar bevatten zodat volgende mensen die er mee willen werken begrijpen wat er in de code gebeurt.

3.2 Functies

Vanuit deze eisen kunnen functies worden opgesteld. Deze functies zijn weergegeven in het volgende diagram met hun bijbehorende interfaces tussen functies. Het diagram is opgebouwd uit back-end functies, front-end functies en interface onderdelen.



Afbeelding 10 - functie diagram

Hoofdstuk 4
De interface

4.1 Unity

Om de interface te bouwen moet hiervoor een geschikt ontwikkelplatform gekozen worden. De keuze voor deze interface hiervoor is gevallen op Unity ,een game engine. Unity word tijdens de studie kort belicht tijdens module 8 (Virtuele Product Ontwikkeling). Hieronder eerst wat informatie over Unity en de ontwikkelaars ervan en daarna word er dieper ingegaan op waarom Unity gekozen is.

4.1.1 Ontwikkelaar

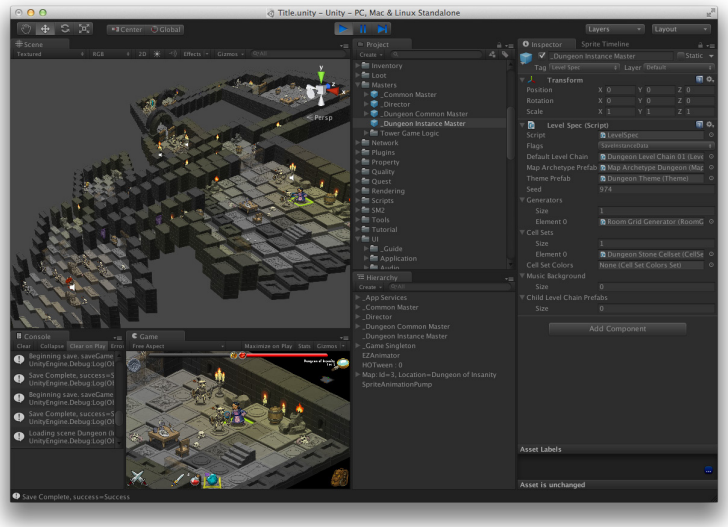
Unity word ontwikkeld door het bedrijf Unity Technology. Het programma is een gamingplatform waar ontwikkelaars hun game in kunnen maken. Het programma is zoals dat heet een game-engine. Unity heeft een 3D omgeving waarin objecten geplaatst kunnen worden. Deze objecten kunnen daarna door middel van code bepaalde functies en gedragingen uitvoeren. De objecten kunnen verschillen van fysieke objecten, camera's, lichten en interface-onderdelen. Code hiervoor word geschreven in C# of Javascript.

Unity is voornamelijk gefocust op de kleine game-ontwikkelaars aangezien ze gratis licenties uitgeven voor non-profit gebruik of ondernemers met een omzet van minder dan $100.000 (ong. €89.000). Veel grote game ontwikkelaars zoals Ubisoft, EA of Valve gebruiken hun eigen game-engines omdat ze die dan specifiek voor een bepaald soort game kunnen inrichten en daarmee de prestaties van hun game kunnen optimaliseren. Voor kleine ontwikkelaars zijn deze engines niet beschikbaar, niet te betalen of veel te complex.[7]

Unity heeft daarentegen een vrij eenvoudige opbouw en werkt met programmeertalen waar veel ontwikkelaars wel bekend mee zijn. Daarnaast bieden ze veel lessen aan over hoe hun engine werkt en hoe een game er in moet worden opgezet.

4.1.2 Waarom voor Unity gekozen

De belangrijkste reden om voor Unity te kiezen om deze interface te maken is dat er voor deze interface in een 3D omgeving gewerkt moet worden. Veel ontwikkelingsprogramma's voor interfaces hebben wel een 2D omgeving waarin getekend kan worden maar geen 3D omgeving. Functies als het uitlezen van G-Code of berekeningen van snelheden of locaties kunnen prima gedaan worden in andere programma's zolang er in C# geprogrammeerd word. Voor het visualiseren kan Unity dan prima gebruikt worden aangezien deze zowel interface-onderdelen in 2D kan weergeven als lijnen in een 3D omgeving.



Afbeelding 11 - De Unity interface

4.2 De Front-End

De Front-End is opgebouwd in een aantal objecten. Deze objecten kunnen componenten bevatten als locaties, afbeeldingen ,meshes colliders en nog vele meer. Ook kunnen objecten weer child-objecten bevatten. Een child-object is afhankelijk van zijn parent. Als de parent beweegt of op non-actief word gezet dan word dat ook gedaan voor zijn children. Bij alleen een verzamel parent-object is het dan ook belangrijk dat zijn posities en rotaties allemaal op 0 staan ander staan zijn child-objecten altijd verkeerd of moeten daar voor compenseren.

4.2.1. De objecten

De interface is opgebouwd uit gameobjects die verschillende functies vervullen:

Canvas

Unity heeft een 3D omgeving waar overheen een canvas geplaatst kan worden. Op het canvas zijn onderdelen die niet bewegen met de camera. Hierdoor blijven interface-elementen op dit canvas stil staan op het scherm tijdens wat er ook gebeurt met de 3D-omgeving. Het canvas is dus een parent object van alle 2D elementen. De Gameobjecten hieronder zijn de tekstvakken,de tekst en de knoppen. Deze zijn allemaal gepositioneerd naar verhouding van het canvas.

LinePLaceHolder

Dit is een Gameobject dat bij het opstarten nog compleet leeg is. Het is de bedoeling dat alle lijnelementen hierin geplaatst worden als child-objecten. Dit is zodat als er iets verandert moet worden aan de lijnen hier simpelweg gezegd kan worden dat alle child-objecten van de LinePlaceHolder aangeroepen kunnen worden.

LinePrefab

De LinePrefab is een gameobject dat niet aanwezig is in de interface bij het opstarten. Het is een gameobject dat in een aparte map bewaard is en word gecreeerd als de code daarom vraagt. Het is namelijk het object dat een lijn die een g-statement visualiseert bevat. Het bestaat uit nog 3 child-objecten namelijk het lijnelement en nog 2 transform-objecten die het begin en eindpunt van de lijn beschrijven. Het bevat ook een script-component die bij het opstarten van dit object eerst de transform-objecten op de goede positie zet en daarna de lijn laat verwijzen naar de transform-objecten.

ScriptManager

De ScriptManager heeft alleen als doel om alle scripts op een centrale plek te hebben staan. Alle scripts behalve die van de lijnen zijn aan dit GameObject als component geplaatst.

EventSystem

Dit is een Object dat door unity zelf is ontwikkeld. Het is een object met een vast stuk code die als doel heeft om input van de gebruiker te verwerken. Alle andere code kan wel aanvragen of er een knop ingedrukt word maar de EventSystem zorgt ervoor dat dit signaal daadwerkelijk word geregistreerd.

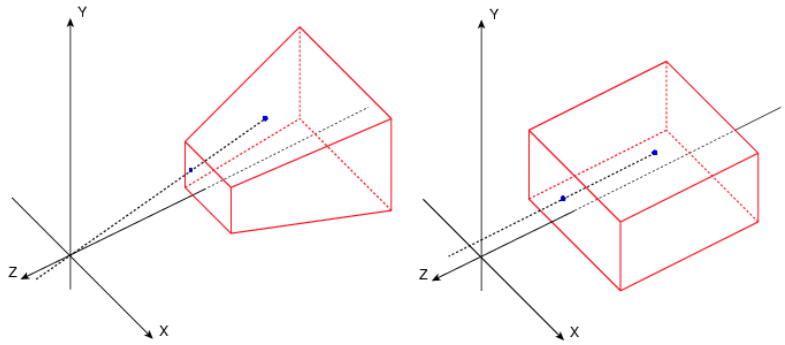
Tracker
De tracker is een bol die simpelweg zijn positie constant verandert over de tijd om de lijnen van de G-Code te volgen. Hij geeft aan waar de freeskop een bepaald moment zou moeten zijn.

Assenstelsel
Het assenstelsel spreekt redelijk voor zich het geeft aan hoe de assen geplaatst zijn en geeft wat meer idee van het perspectief dat de lijnen zijn geplaatst. Het object bevat 3 child-objecten met linerenderers voor de verschillende assen.

De camera's
Er zijn 2 camera's toegevoegd namelijk een 2D en een 3D camera. De 2D camera is een bovenaanzicht wat zeker handig is als de g-code in lagen is opgebouwd. Het 3D camera standpunt is kijkt van schuin voren naar het model toe. Hier kan de gebruiker ook zien hoe diep de frees zal gaan. Zeker bij dubbelgekromde vlakken is dit een belangrijk aanzicht omdat het 2D aanzicht dit waarschijnlijk als rechte lijnen zal visualiseren.

Er is voor de camera's wat interactie ingebouwd. Zo kan over de assen bewogen worden door knoppen op het toetsenbord in te drukken. En kan er in en uit gezoomd worden door te scrollen met het muiswiel. Rotatie is nog niet ingebouwd omdat dat erg goed gekalibreerd moet worden om goed en natuurlijk over te laten komen.

Beide camera's werken op een ortographische manier. Dat betekent dat ze een rechthoek op hun locatie tekenen en van daaruit rechte lijnen trekken. Dit in vergelijking met een perspectiefcamera. Deze trekken vanuit 1 punt allemaal lijnen naar buiten toe. Dit betekent dat wat ver weg is kleiner word. Dit is niet het geval bij een ortographische camera. Omdat perspectief camera's snel een erg vertekend beeld geven van grootte is voor dit soort lijnenwerk een ortographische camera een stuk realistischer.[8]



Afbeelding 12 - Links: Perspectief camera, Rechts : Ortographische camera

4.2.2 De visualisatie

Afbeelding 13 - De interface met een 2D camera in het donkere thema

Op deze twee pagina's zijn twee verschillende standen van de interface weergegeven. Het verschil tussen een 2D en 3D camera is hier te zien en het verschil tussen het lichte en donkere thema is weergegeven.

Afbeelding 14 - De interface met een 3D camera in het lichte Afbeelding 13

Alle interfaceobjecten

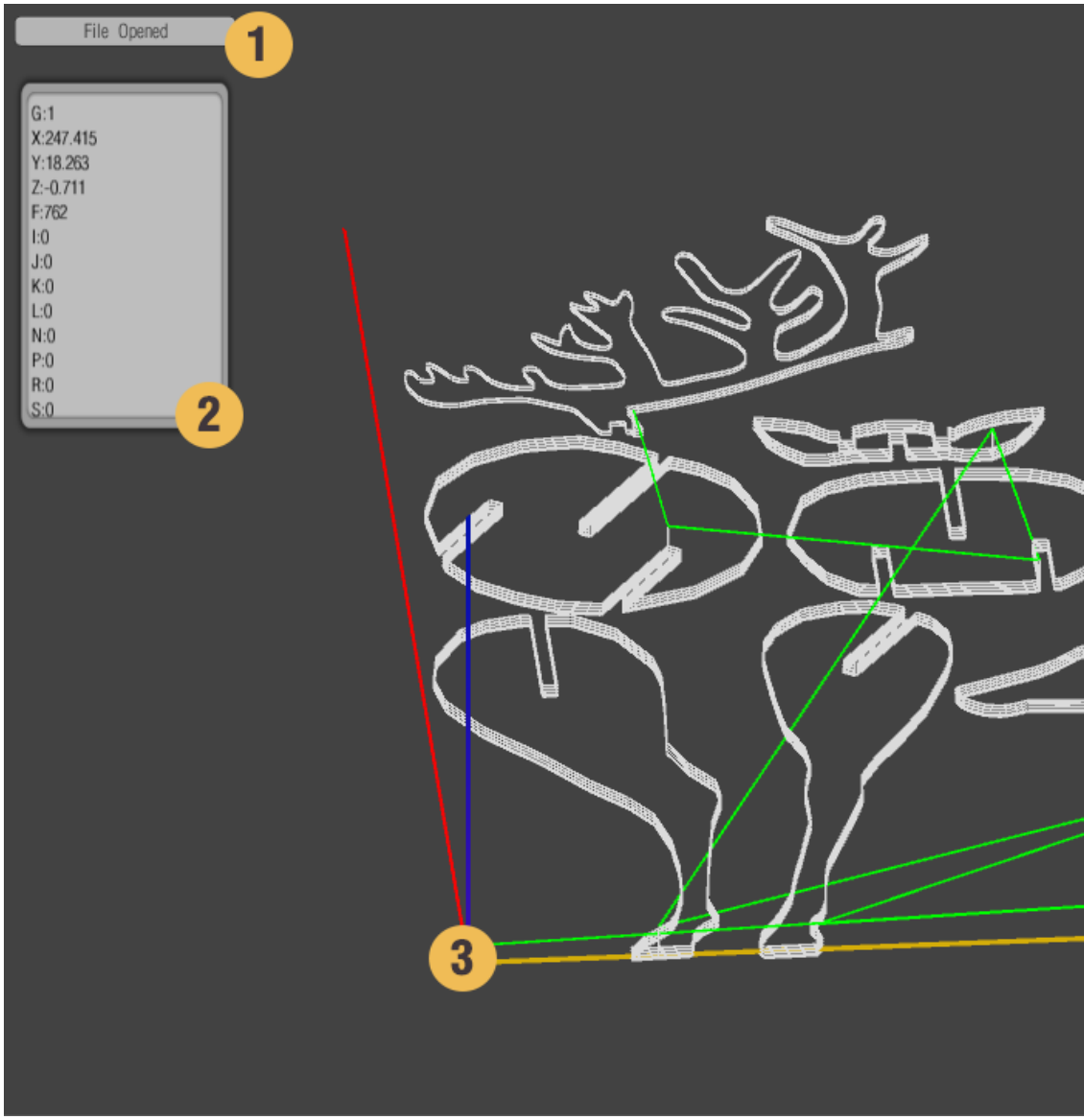
1. De laadknop. Eeen druk op deze knop open de filebrowser om bestanden mee te openen

2. De GCode. Dit tekstvak geeft de huidige GCode die doorlopen word aan.

3. Het assenstelsel. Het assenstelsel bestaat uit een x,y en z-as die zichzelf schalen naar de gcode.

4. De lijnen. Dit zijn de lijnobjecten die de verschillende tappen van de g-code visualiseren.

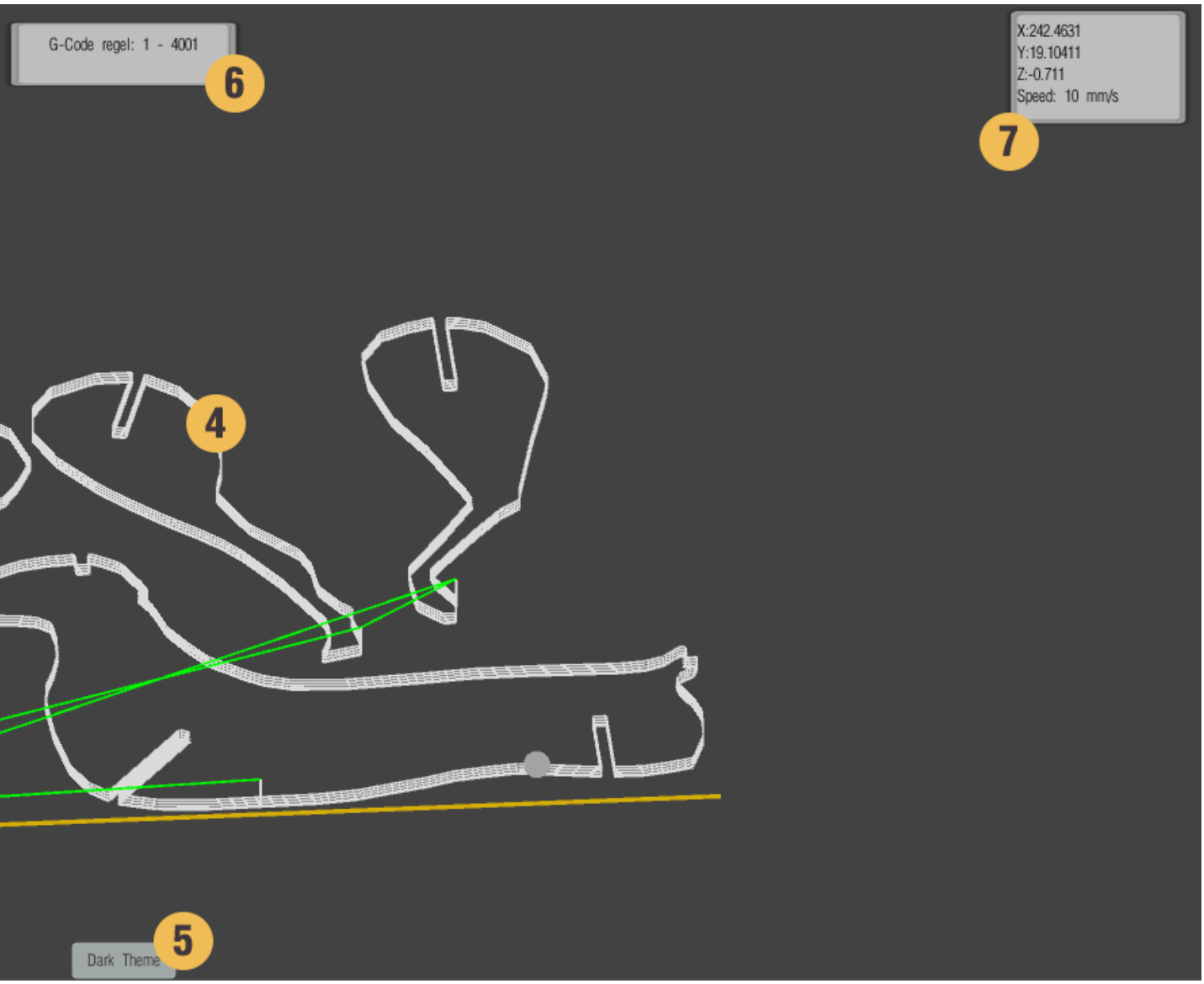
5. De themaknop. Door middel van het klikken op deze knop kan de gebruiker wisselen tussen een donker en een licht thema



Afbeelding 15 - De interface met al zijn objecten

6. De segmenttekst. Deze tekst laat zijn welke groep lijnen in beeld zijn. In dit geval zijn de eerste 4000 lijnen in beeld.

7. De hudige locatie van de freeskop. Dit tekstobject geeft aan welke locatie en welke snelheid de freeskop heeft op het moment.



4.3 Back-End

4.3.1 G-Code

Om goed te kunnen begrijpen wat de back-end verwerkt en daarna omzet tot een visualisatie moet duidelijk zijn wat g-code is en doet. G-Code is numerieke code die door middel van “statements” aan een machine duidelijk maakt wat hij moet doen. G-Code heet G-Code omdat het primair draait om G-Statements. G-Statements bepalen wat voor actie gedaan zal worden en statements die hierna komen geven informatie over hoe deze actie uitgevoerd moet worden. Een voorbeeld van een stuk G-Code tekst is:

G21
G0 X15 Y12 Z20
G1 X17 Y25 Z20

De G-Code Parser (het programma dat de G-Code uitleest) leest dit als 3 statements. Hij begint met G21 wat een statement is die alleen maar zegt dat deze code in milimeters is. Daarna leest hij G0 wat een beweging van de freeskop is die niet door het materiaal gaat. Deze beweging kan daarom ook snel gemaakt word en G0 staat dan ook voor Rapid Positioning. Hij zet de frees kop hiermee op coordinaat (15,12,20) en dit coordinaat is dus in milimeters. Hierna gaat hij echt frezen dat is namelijk wat G1 betekent een lineaire beweging door het materiaal heen. Hij beweegt dan van zijn huidige positie (15,12,20) in een rechte lijn naar zijn nieuwe positie (17,25,20).[9]

4.3.2 De opbouw van de code

De interface is opgebouwd uit objecten in de game-engine. De code bestaat uit verschillende scripts die allemaal als component zijn toegevoegd aan 1 object. Dit zorgt er voor dat er een centraal punt is voor alle code en deze wel allemaal gedraaid worden in de game-omgeving. Veel code heeft invloed op verschillende objecten en deze objecten zijn dan ook gekoppelt aan variabelen van de code.

Er zal door verschillende scripts heen gelopen worden en hun functies worden belicht.
De volgende classes zijn aanwezig in het programma en bevatten de functies en variabelen die de interface besturen:

Fileopener	Deze class is puur gericht op het openen en uitlezen van een g-code bestand
Arraymaker	Deze clas converteert de g-code naar variabelen in een array waar de andere classes mee overweg kunnen
Linebuilder	Deze class bouwt de lijnen en geeft ze de goede waardes
LineChanger	Elke lijn word door deze class nog individueel goed ingesteld met de waardes vanuit Linebuil-der
Tracker	Deze class laat de tracker van positie veranderen.
Trackerlocation	Deze class leest de huidige positie van de tracker uit
CameraManager	Deze class verwerkt alle input over de camera’s
ColorManager	Deze class verandert de interfa-ce elementen van kleur als een ander thema word gekozen

4.3.3 De werking van de code

Het interpreteren van de G-Code

Om G-Code in te laden is er een aparte class gemaakt genaamd FileOpener deze bevat een functie die een scherm opent waar de gebruiker een .nc (numerical control) bestand kan kiezen. Deze word compleet gelezen en omgezet in een String. Deze String is een lange lijst van statements waar nog weinig mee gedaan kan worden omdat het gezien word als 1 entiteit. Deze String word daarna doorgegeven aan de class Arraymaker.

Arraymaker splitst de verschillende statements in losse statements en zet deze in een Array (tabel). Dit is dan nog een tabel met maar 1 rij en vele kolommen. Hierna gaat de Arraymaker kijken naar de inhoud en bepaalt op basis van de eerste letter van een statement wat voor statement het is en zet deze in de goede plek in een andere Array. Deze Array bevat meerdere rijen namelijk 1 voor elke soort statement. Elke kolom bevat sowieso een G-Statement en voor de rest zijn bijbehorende informatie statements.

4.3.4 G-Code omzetten naar lijnen

De array met alle statements word hierna ingeladen in een class die er door heen gaat lopen. Hij gaat elke kolom langs en bekijkt of het een G0 (Rapid Positioning) of G1 (lineare beweging) betreft. Als dit het geval is dan maakt hij een nieuw lijnsegment met als beginpunt de XYZ-waarden van het vorige statement en als eindpunt de XYZ-waarden van het huidige statement. Omdat veel verschillende objecten renderen behoorlijk zwaar is voor een computer maakt hij maar een gedeelte van de lijnsegmenten. Op het moment staat dat getal op maximaal 4000. Door deze groepen van lijnen kan dan gewisseld worden. Hij verwijdert dan eerst de 4000 aanwezige lijnen en bouwt de volgende 4000 op.

De rest van de G-Code statements leest hij op het moment nog niet uit. Dit zou natuurlijk uitgebreid kunnen worden met een library die verschillende functies per statement aanroept.

Tracker over lijnen heen en weer laten bewegen

Naast de lijnen is er nog een tracker aanwezig die de huidige locatie van de frees laat zien. Deze is niet direct verbonden met de lijnen maar wel met dezelfde array voor locaties waar ook de lijnsegmenten hun gegevens vandaan halen. De tracker lees dus op dezelfde manier als de lijnsegementen locaties uit en loopt daar doorheen. Op basis van of het een G0 of G1 statement is past de tracker zijn snelheid aan. In het optimale geval wacht de tracker met het beginnen van zijn volgende beweging na feedback van de Arduino maar om redenen die later uitgelegd zullen worden is dat niet mogelijk.

Om ervoor te zorgen dat de snelheid klopt maakt Unity gebruik van een Time. deltaTime functie. Deze functie zorgt ervoor dat het programma niet naar frames kijkt maar naar secondes om deze funcite uit te voeren. Normaliter bekijkt unity elke frame wat een nieuwe positie zou zijn. Bij een snelheid van 10 zou dat dus 10 punten per frame zijn. Aangezien de hoeveelheid frames niet altijd constant is deze snelheid dus ook niet constant. Door het gebruik van Time.DeltaTime kijkt Unity juist naar de tijd die het duurde van de vorige frame tot de huidige en bepaald

daarvoor de nieuwe locatie. Als er dus 10 punten per seconden nodig zijn in plaats van 10 punten per frame dan is de functie Time.deltaTime*10.

In deze beweging is stapresponsie van de steppermotoren nog niet meegenomen. Dit zou de code een stuk ingewikkelder maken aangezien de snelheid dan ook tijdsafhankelijk word. Er zou dan een controle moeten worden ingebouwd worden die controleert of de snelheid al maximaal is en de snelheid opbouwen. In het huidige commanda dat gebruikt word is dat niet mogelijk omdat die eerst zijn gehele beweging doet voordat hij naar de volgende regel gaat. Een benadering kan ook gedaan worden door het begin en einde van een lijn los te nemen en daarvoor een gemiddelde snelheid voor te nemen. Hierdoor zijn er 3 bewegingen namelijk begin en eind op ongeveer de helft van de snelheid en daarna een groot middenstuk op maximale snelheid.

Camerabewegingen

De Camera's in de interface hebben een positie en rotatie in de virtuele wereld. Door middel van input van de muis en het toetsenbord kan de gebruiker kan deze positie verandert worden. Aangezien we werken met ortographische camera's zoals uitgelegd in het front-end gedeelte werkt in- en uitzoomen neit door middel van het verplaatsen van de camera. In plaats daarvan moet de groote van het scherm verandert worden. Hoe groter het beeld hoe verder de camera is "uitgezoomd". Wel kan er over de assen bewogen worden door middel van wat toetscombinaties. In het geval de 2D camera maakt een beweging over de z-as dus niet uit behalve als de camera onder de lijnen terecht komt want dan vallen de lijnen buiten het beeld.

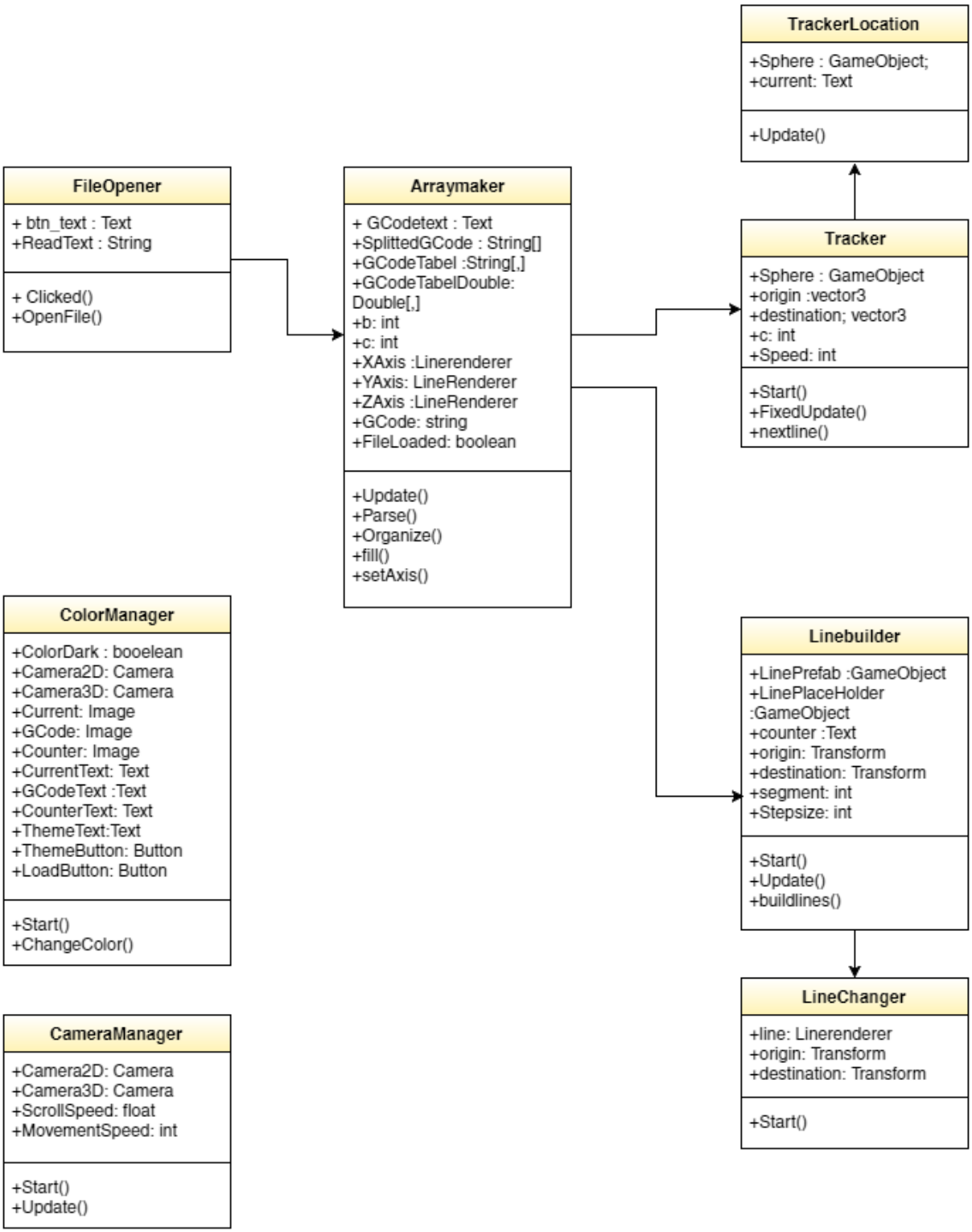
Rotatie van de camera's is nog niet in het prototype meegenomen omdat daar nog wat problemen aan vast zitten. De 2D camera zou prima om de z-as kunnen roteren als dat nodig zou moeten zijn. De 3D camera daarentegen staat gericht op het nulpunt in het assenstelsel.De camera staat dus in verhouding van dit punt en moet dus dit punt geroteerd worden waarmee de positie van de camera mee beweegt. Omdat de camera niet automatisch mee roteert verliest hij al snel het nulpunt en de lijnen uit het oog. De meeste intuitive actie is om met de muis op een positie te klikken en dan de muis te bewegen om de camera te roteren om dat punt heen. Dat is op het moment nog niet mogelijk omdat hiervoor uit deze muisklik een coordinaat in de virtuele wereld moet voortkomen en deze voor de camera als referentiepunt te zetten in plaats van het nulpunt. Dit zou mogelijk moeten zijn met een signaal waarmee een soort van straal uit de muisklik komt (raycast functie in Unity) en als die door het xy vlak beweegt dat als coordinaat te nemen. Dit betekent alleen wel dat er altijd om een punt word geroteerd waarvan de z-component nul is.

Kleurthema's veranderen

Deze code doet niet meer dan alle interface elementen inladen en daarvan de waarde van hun kleur te veranderen. Een probleem op het moment is dat hij dit alleen doet als er een knop word ingedrukt om van thema te wisselen. Nieuwe lijnen die gegenereerd worden staan dan nog steeds in hun originele kleur. Er zou dus nog een wisselwerking tussen de class ColorManager en Linebuilder moeten zijn.

4.3.4 Class Diagram

Deze classes kunnen ook weer uitgezet worden in een class diagram om hun interactie met elkaar weer te geven. Sommige calsses hebben wel interactie met de objecten in de interface maar verder geen interactie met andere classes. Deze zijn wel weergegeven in het diagram.



4.4 Vervolgstappen

4.4.1 Cirkelbogen

Op het moment kan het prototype nog geen cirkelbogen maken. Dit is voor G-Code vanuit Easel geen probleem aangezien deze alleen vectorbestanden inlaad en deze omzet naar echte lijnen omdat Easel ook geen code heeft om cirkelbogen te herkenen. Hierdoor kan het prototype alsnog cirkelbogen maken maar die bestaan dan uit heel veel kleine rechte lijnen die een cirkelboog benaderen.

Een cirkelboog in G-Code heeft zijn eigen G-Statement:G02 en G03 voor respectievelijk een boog met de klok mee of tegen de klok in. Een G02-statement ziet er bijvoorbeeld als volgt uit:

G02 X10 Y30 I0 J-2

Wat dit betekent is dat er een cikel boog gemaakt moet worden van het huidige punt naar coordinaat (10,30). Om te weten wat voor cirkelboog het hier precies om gaat moet het middelpunt van de cirkel bepaald worden. Dat gat door middel van I en J deze zijn de afstanden van het beginpunt tot aan het middelpunt van de cirkel. Stel het start punt is (8,32) Dat betekent dat er een boog ontstaat die 2 stappen in de x-richting gaat en 2 stappen naar beneden in de y-richting. Dit maakt dan een kwart cirkel.

De lijnrenderer van Unity kan cirkelbogen nog niet maken aangezien hij alleen rechte lijnen kan weergeven. Een curve kan net als wat Easel doet dan toch weer te geven als hij opgebroken word in kleine stukken en als lijnen weergegeven. Om de vorm te bepalen is het dan mogelijk om op twee manieren een hoop locaties te bepalen op deze curve als start- en eindpunten van de lijndelen

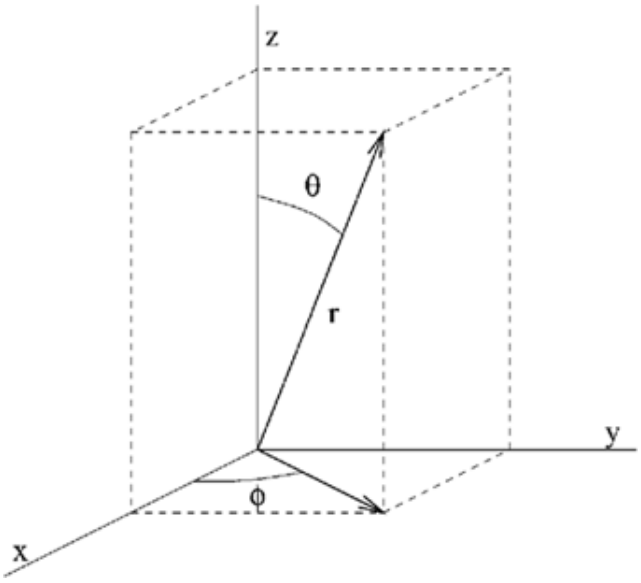
Pool- en Bolcoordinaten

Poolcoordinaten zijn een wiskundige manier om een cirkel niet in X en Y coordinaten uit te drukken maar in een parametrisatie. In het geval van een cirkel met een straal van 1 het middelpunt in het nulpunt kan hij uitgedrukt worden in

X^2 +Y^2 = 1

Dit kan omgeschreven worden in:

X= cos(theta) en Y = sin(theta)



Afbeelding 16 - visualisatie bolcoordinaten

Hiermee kunnen alle locaties op de cirkel worden beschreven. Bij een gehele cirkel loopt t van 0 tot 2pi. Als je deze cirkel bijvoorbeeld in 100 delen wil opsplitsen dan laat je t stappen maken 2pi/100 en vult elke stap in voor X en Y en zo heb je 100 coordinaten.

Dit principe kan uitgebreid worden voor een bol als er ook een Z-component in de beweging zit. De bol kan dan uitgedrukt worden in:

X=cos(phi)*cos(theta)
Y=cos(phi)*sin(theta)
Z=sin(phi)

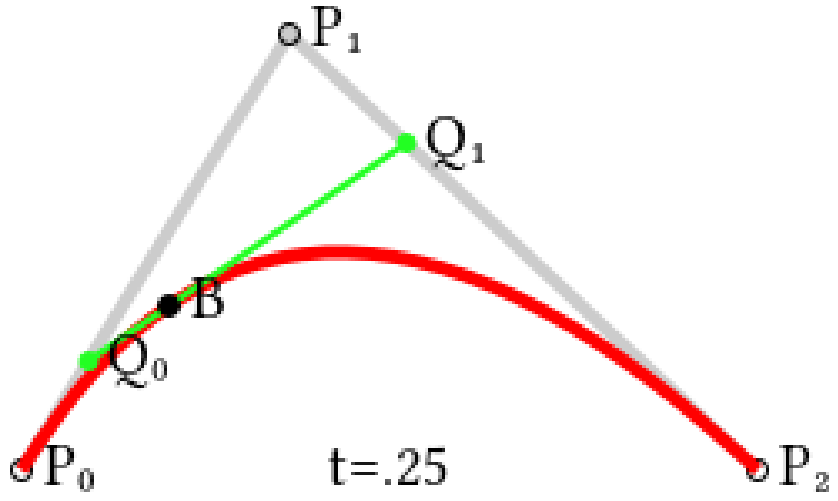
Hiermee draait een punt een gehele cirkel over het XY vlak met theta van 0 tot 2pi en deze cirkel draait daarna nog een half rondje rond zijn as om een bol te vormen door middel van phi van 0 tot pi in te stellen. Hiermee kunnen dan ook weer op dezelfde manier coordinaten voor de lijnen bepaald worden.

Bezier Curve

Een Bezier curve heeft een begin en eindpunt en een steunpunten waar de curve aan gerefereerd word. Deze curve zou kwart cirkels kunnen beschrijven als met 1 steunpunt. In het geval van een kwart cirkel van (0,1) tot (1,0) zou het steunpunt zijn (1,1). Voor een cirkelboog moet de afstand van het beginpunt tot het steunpunt hetzelfde zijn als van het eindpunt tot het steunpunt. Het steunpunt behoud daarnaast altijd een 90 graden hoek tusssen de lijn vanaf het beginpunt en de lijn vanaf het eindpunt.[10]

De formule voor de bezier curve is:
B(t) = ((1-t)^2)*P0 + 2(1-t)*t*P1 + (t^2)*P2

Met P0 =Startpunt , P1 = steunpunt en P2 = eindpunt



Afbeelding 17 - visualisatie Bezier Curve

4.4.2 Arduino comunicatie

Het is prima mogelijk om een Unity programma te laten communiceren met de Arduino. Het is een stuk code in Unity die de USB poort herkent en daar signalen naar ka sturen en uit kan lezen. En bij de Arduino is het een stuk code die op input van de Unity een reactie weet te geven. Er treedt alleen een probleem als de interface tijdens het frezen moet communiceren. De poort word dan namelijk al gebruikt door de G-Code Sender (Easel, Chilipeppr, UGS) die zelf ook weer feedback ontvangen van de Arduino. Omdat de Arduino maar 1 poort heeft kan de interface daar niet tussendoor signalen opvangen. Er zijn hier twee mogelijke oplossingen voor.

Ten eerste zou er een extra poort aan de Arduino gebouwd kunnen worden. Qua electronica is dit niet heel ingewikkeld. Het word pas ingewikkeld wanneer er dan in de code van GRBL veranderingen gedaan moeten worden om er voor te zorgen hij zijn output twee keer doet in beide poorten.

De andere oplossing zou zijn om van de interface zelf een G-CodeSender te maken. Dit is een van de Future Concepts die later in het verslag nog worden belicht. IN het kort zou dit beteken dat in plaats van het gebruiken van Easel of Chilipeppr de interface gebruikt word om G-Code te converteren naar GRBl toegankelijke signalen en daarvan feedback te ontvangen. Dit is zeker mogelijk aangezien zoals eerder vermeld het wel degelijk mogelijk is om verbinding te maken met de Arduino door je USB poort uit te lezen. Dit is nogal een omslachtige en ingewikkelde manier dus het zou handiger zijn om UniDuino aan te schaffen. UniDuino is een applicatie die puur bedoelt is voor communicatie tussen een Arduine en Unity. Het is een investering van 30 euro maar bespaart zeker een hoop werk omdat er dan normale serial commands gegeven kunnen worden aan de Arduino. Dit is nu niet het geval omdat de Arduino geen eigen library heeft met Serial Commands.

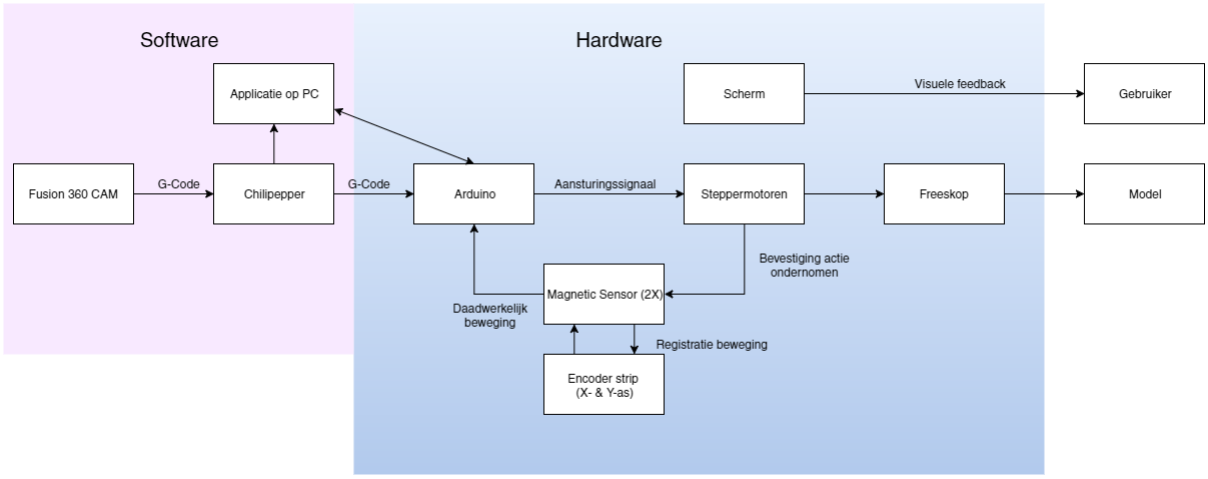
Hoofdstuk 5
Future Concepts

Vanuit deze interface en zeker als de probleempunten die in de voorgaande hoofdstukken genoemd zijn nog opgelost worden kunnen er wat volgende projecten aan toegevoegd worden.

5.1 Regelsysteem

Een groot probleem van de X-Carve is nog steeds zijn onnauwkeurigheid. Aangezien als de interface goed gekalibreerd op de snelheid van de frees en alle g-statements correct geimplementeerd zijn zou de interface op elk moment precies moeten weten op elk moment waar de frees is. Dat betekent dat als er op een manier de daadwerkelijk locatie van de frees gemeten kan worden hij hiermee gecorrigeerd kan worden. De afwijking kan bepaald worden en de interface zou signalen naar GRBL kunnen sturen om zich daarop aan te passen.

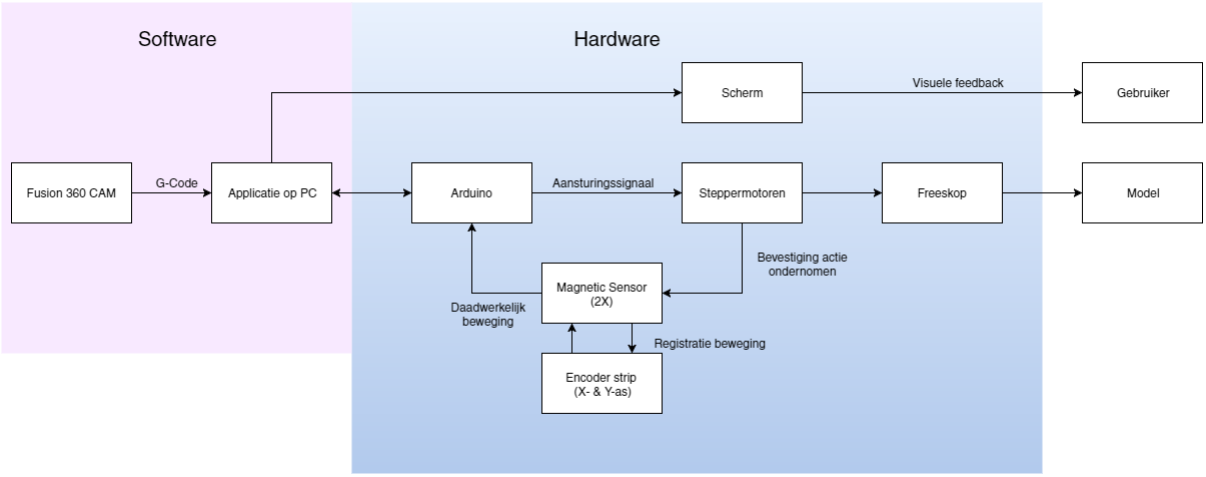
Het meten van de freeskop kan redelijk eenvoudig door magnetische encoder. Dit betekent dat er magnetische strips aan de rails geplaats word en door middel van sensoren word de verplaatsing ten opzichte van deze strips bepaald. Het systeem zou er dan als volgt uit komen te zien.



Afbeelding 18 - Systeem met correctie loop

5.2 G-Code Sender

Als de interface alle G-Code kan interpreteren en kan communiceren met de Arduino is het heel goed mogelijk om van de interface een g-code sender te maken. In feite doen Chilipepper en UGS niet veel anders dan g-code van tekst omzet naar commando's en na elke bevestiging van de Arduino een nieuwe regel omzetten en naar de Arduino sturen. Als duidelijk word hoe deze omzetting gedaan word zou de interface dit prima kunnen bewerkstelligen. Het maakt daarnaast het regelsysteem veel gemakkelijker omdat het dan niet tussen g-code commando's van een g-code sender zijn correctie te doen maar kan dat zelf direct doen. Het schema van eht systeem zou dan weer veranderen.



Afbeelding 19 - Systeem met correctie loop en de interface als G-Code sender

5.3 VR/AR uitbreiding

Omdat unity al eigen plugins heeft voor VR en Ar apparaten zou een concept die dit hierin zou verwerken zeker kunne. Een mogelijkheid zou kunnen zijn om een tablet te nemen die door middel van zijn camera de G-Code kan projecteren op de daadwerkelijke X-Carve op een manier die Augment ook wel gebruikt. Dit maakt het voor de gebruiker meteen duidelijk welke bewegingen de frees nou echt zou gebruiken en of zijn materiaal groot genoeg is om dit te bewerkstelligen. En natuurlijk of zijn materiaal op de goede plek is ingeklemd.

Hiervoor zou de interface wel wat aangepast moeten worden voor de tablet maar niet veel omdat Unity zijn programma's voor meerdere platforms kan ontwikkelen.

De lastigheid hiervan zit hem voornamelijk in het positioneren van de lijnen over het camerabeeld heen. Er zullen een aantal referentiepunten nodig zijn op de X-Carve waarmee het programma zijn beeld kan calibreren. Hij hoeft dan alleen maar zijn camera in Unity moeten verplaatsen naar de plek die hij bepaald heeft vanuit die referentiepunten. Hier moet ook weer rekening gehouden met perspectief camera's dus dit is nog wel een behoorlijk ingewikkeld project. Een gemakkelijkere toepassing zou zijn om een aantal camera's op vaste punten te hangen en die op de interface te kunnen bekijken. Door middel van de vaste punten kan je ze 1 keer calibreren en hoeft dat niet meer met referentiepunten in real-time. Nog veel eenvoudiger zou zijn om boven de X-Carve een beamer te hangen en die het 2D bovenaanzicht te laten projecteren.

Afbeelding 20 - Augment

Hoofdstuk 6
Conclusie
Discussie

6.1 Conclusie

Aan het begin van het project zijn er twee doelstellingen gesteld namelijk a) Analyse doen van de omgeving waarin de interface moet opereren en b) het daadwerkelijk maken van een interface. Door deze analyse erg breed te nemen en alle onderdelen die er ook maar iets mee te maken kunnen hebben te bekijken is er een compleet beeld gecreeerd over hoe de interface in het gehele systeem past. Uit deze analyse zijn hierna eisen en functies gekomen waar de interface aan zou moeten voldoen.

Het prototype voldoet niet aan alle eisen die gesteld zijn. Een aantal belangrijke onderdelen zijn nog niet werkzaam in het prototype. Doordat er nog geen communicatie mogelijk is met de Arduino geeft de interface alleen nog maar een simulatie van hoe de X-Carve door de stappen heen zou moeten lopen en niet daadwerkelijk wat hij doet.

De doelstelling om g-code te kunnen lezen en te visualiseren is grotendeels wel gelukt. In het prototype kunnen rechte lijnen weergegeven worden en in het verslag worden er voorstellen gedaan om ook bogen te visualiseren. Dat deze doelstelling behaald is maakt de weg vrij voor nog andere mogelijke uitbreidingen van de interface zoals het ombouwen tot een g-code sender of een AR-applicatie die de g-code over de X-Carve projecteert.

6.2 Discussie

Veel tijd van dit project is gaan zitten in het uitzoeken hoe de CAD programma's werken. Het is dan ook jammer dat uit de analyse blijkt dat het voor dit project weinig verschil uitmaakt welk CAD-programma gebruikt word. Wel vind ik nog steeds dat het van belang was dat deze analyse gedaan werd omdt het de levensduur van de interface bevestigt.

Op een gegeven moment moest er een beslissing gemaakt worden wat het daadwerkelijk eindproduct zou zijn dat opgeleverd moest worden. Het kon een mock-up zijn van een interface die heel gebruiksvriendelijk was en waar veel usability testen over waren gedaan of een werkend prototype die de het programma daadwerkelijk kan draaine. Er is voor gekozen om een werkend prototype te maken. Dit betekent dat het proces niet zoals normaal was met een ideefase, conceptfase en detaillering maar al redelijk snel er gewoon gebouwd moest worden. Een vervolgstap na dit project zou dan ook zeker een onderzoek zijn naar hoe gebruikers met de interface omgaan en waar qua usability nog verbeter punten inzitten

Referentielijst

Bronnen

1. Retrieved May 2016, from Inventables.Com: https://www.inventables.com/

2. Retrieved May 2016, from Sparkfun.com: https://learn.sparkfun.com/tutorials/what-is-an-arduino

3. Retrieved May 2016, from github.com: https://github.com/synthetos/grblShield/wiki/What-is-grblShield%3F

4. Retrieved May 2016, from hongkit.com: http://www.hongkiat.com/blog/scalable-vector-graphic/

5. Retrieved May 2016, from Github.com: https://github.com/grbl/grbl

6. Retrieved May 2016, from Solidworks.nl: http://www.solidworks.nl/sw/6453_NLD_HTML.htm

7. Retrieved June 2016, from Unity3d.com: https://unity3d.com/public-relations

8. Using Camera's. Retrieved July 2016, from Unity3d.com: https://unity3d.com/learn/tutorials/topics/graphics/using-cameras

9. Retrieved June 2016, from cnccookbook.com: http://www.cnccookbook.com/CCCNCGCodeRef.html

10. Retrieved July 2016, from Wikibooks.org: https://en.wikibooks.org/wiki/Cg_Programming/Unity/B%C3%A9zier_Curves

Afbeeldingen

Sommige afbeeldingen zijn van het ineternet gehaald. hieronder welke afbeeldingen van welke sites.

Omslagfoto: Inventables.com

Afbeelding 1: http://x-carve-instructions.inventables.com/

Afbeelding 3: http://www.referencedesigner.com/

Afbeelding 4: http://enhancepd.com/

Afbeelding 5: http://dev.volumill.com/

Afbeelding 6: http://www.cncrouterparts.com/

Afbeelding 8: https://www.utwente.nl/io/curriculum/

Afbeelding 11: https://spellbindstudios.files.wordpress.com

Afbeelding 12: http://blender.stackexchange.com/

Afbeelding 16: http://www.sterrenkunde.nl/

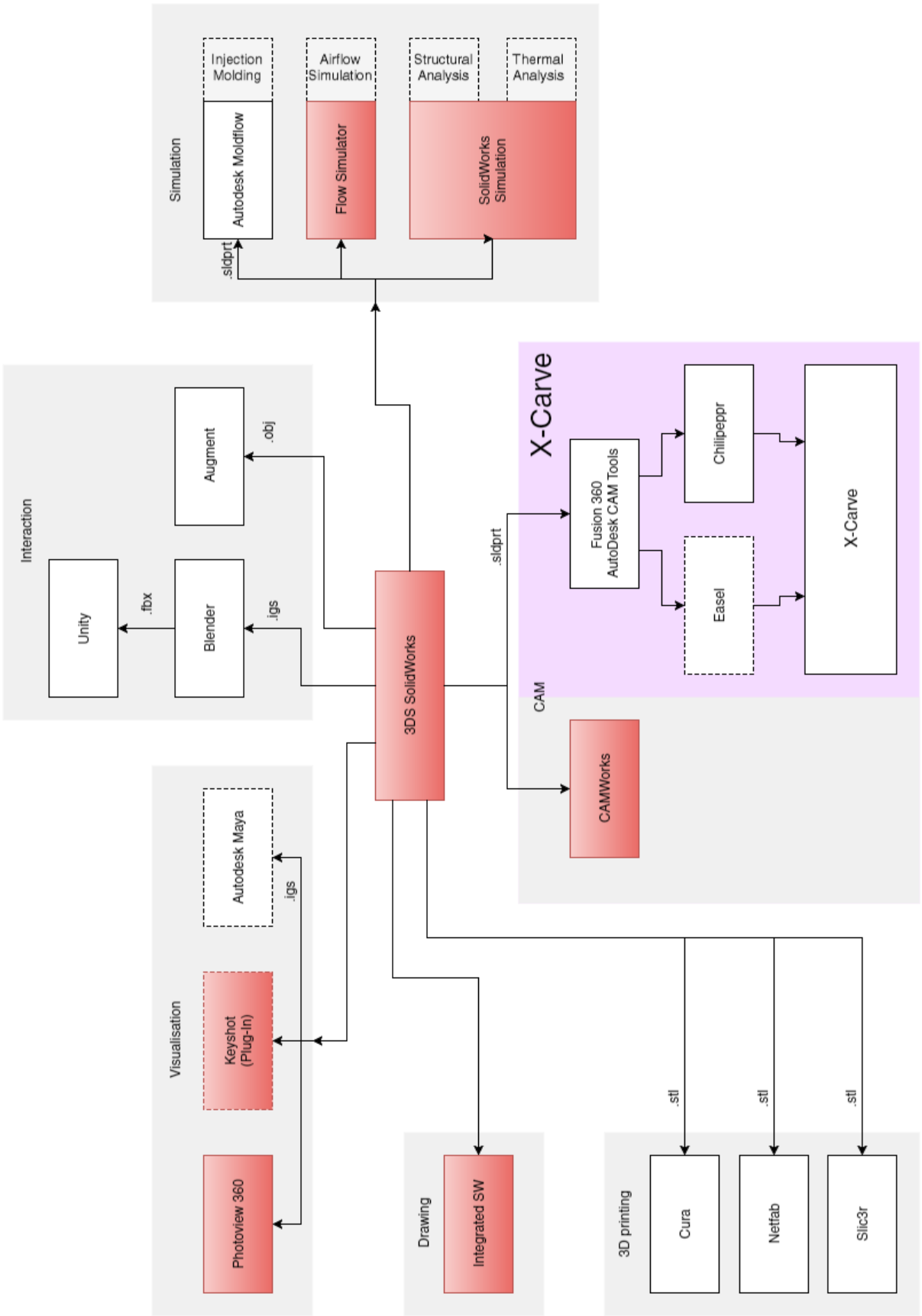
Afbeelding 17: https://en.wikibooks.org

Afbeelding 20: http://www.augment.com/

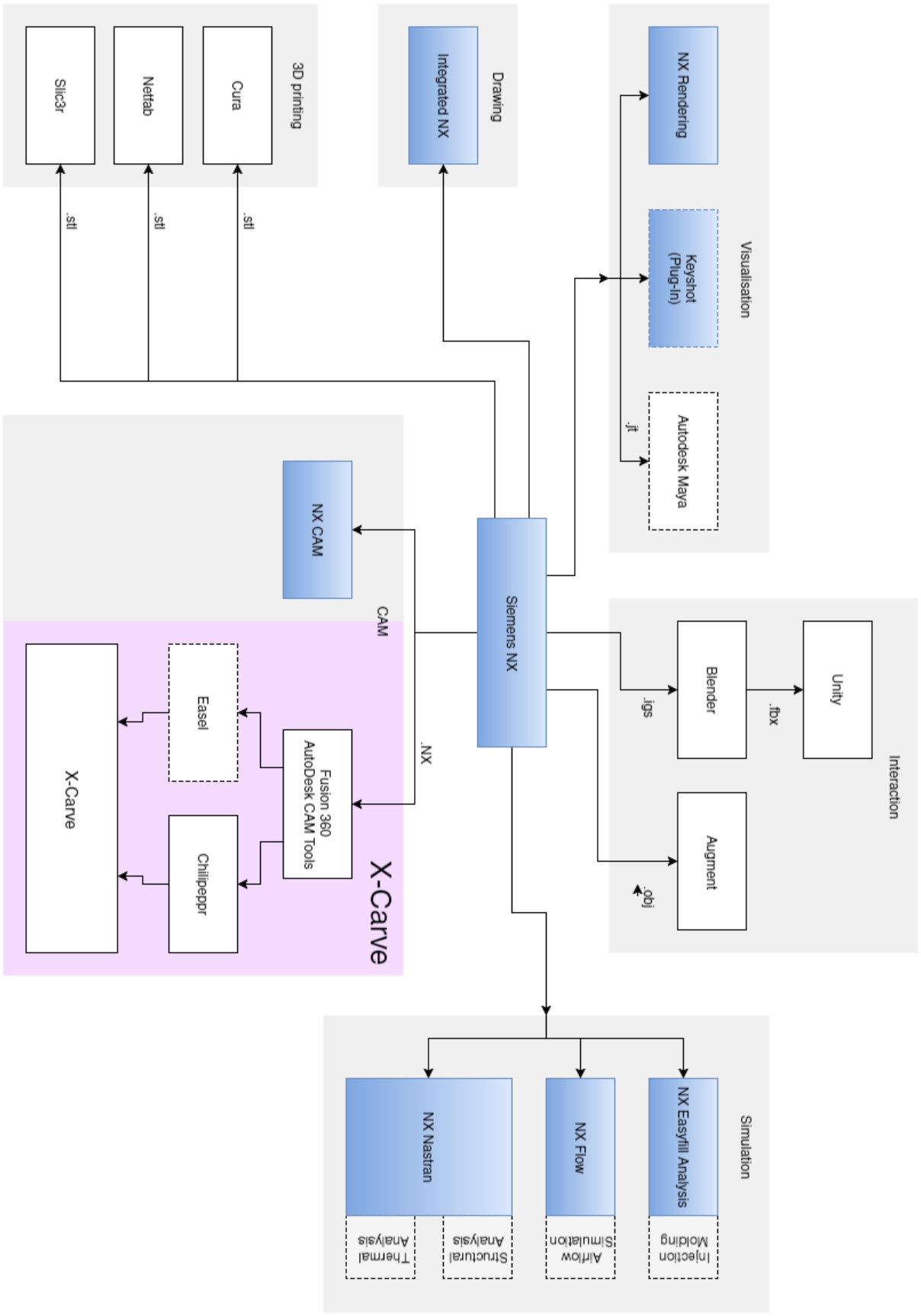
BIJLAGEN

1. CAD-Diagrammen
2. Functiediagram
3. Classdiagram
4. Code

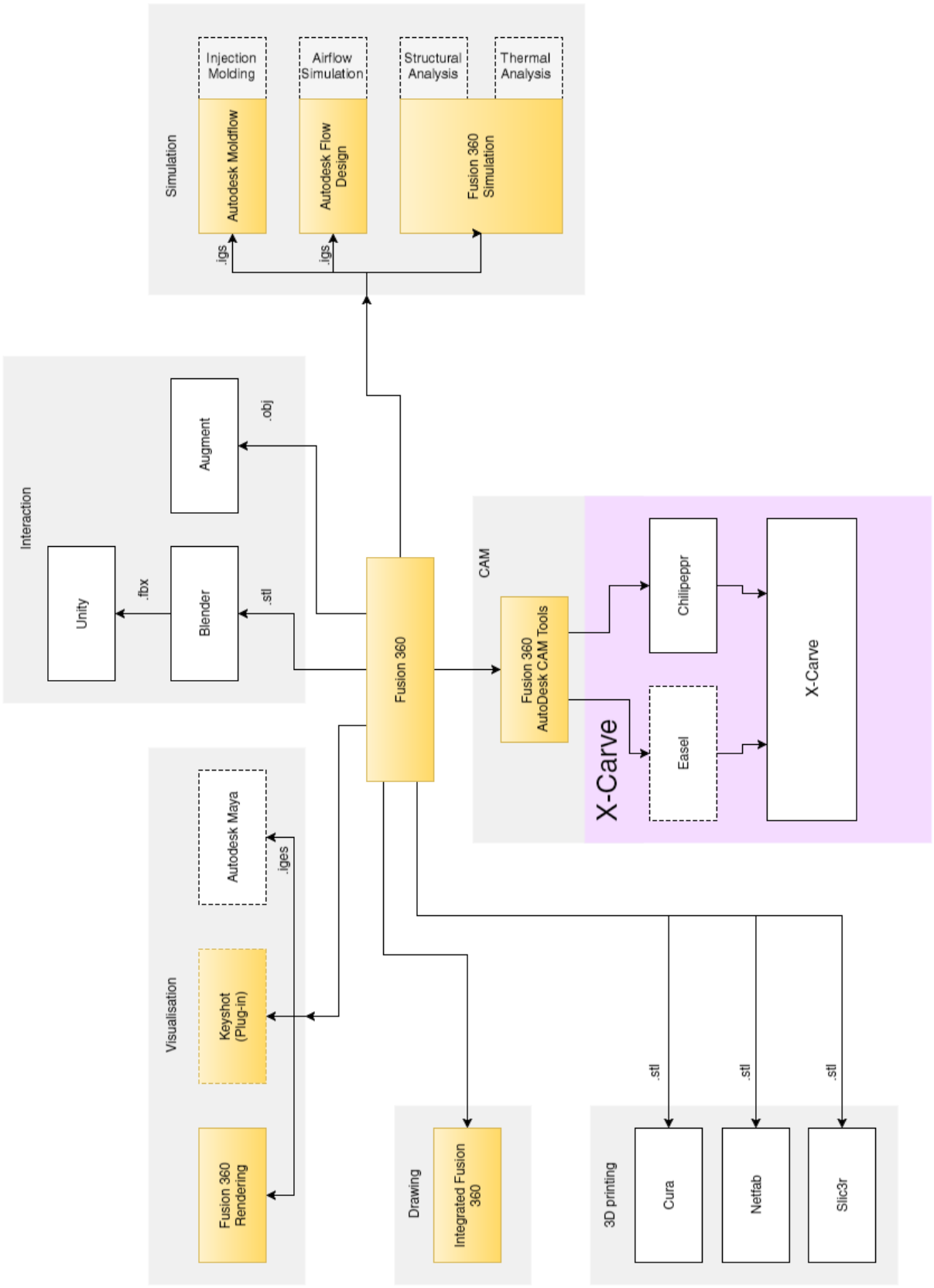
1.1 SolidWorks Diagram



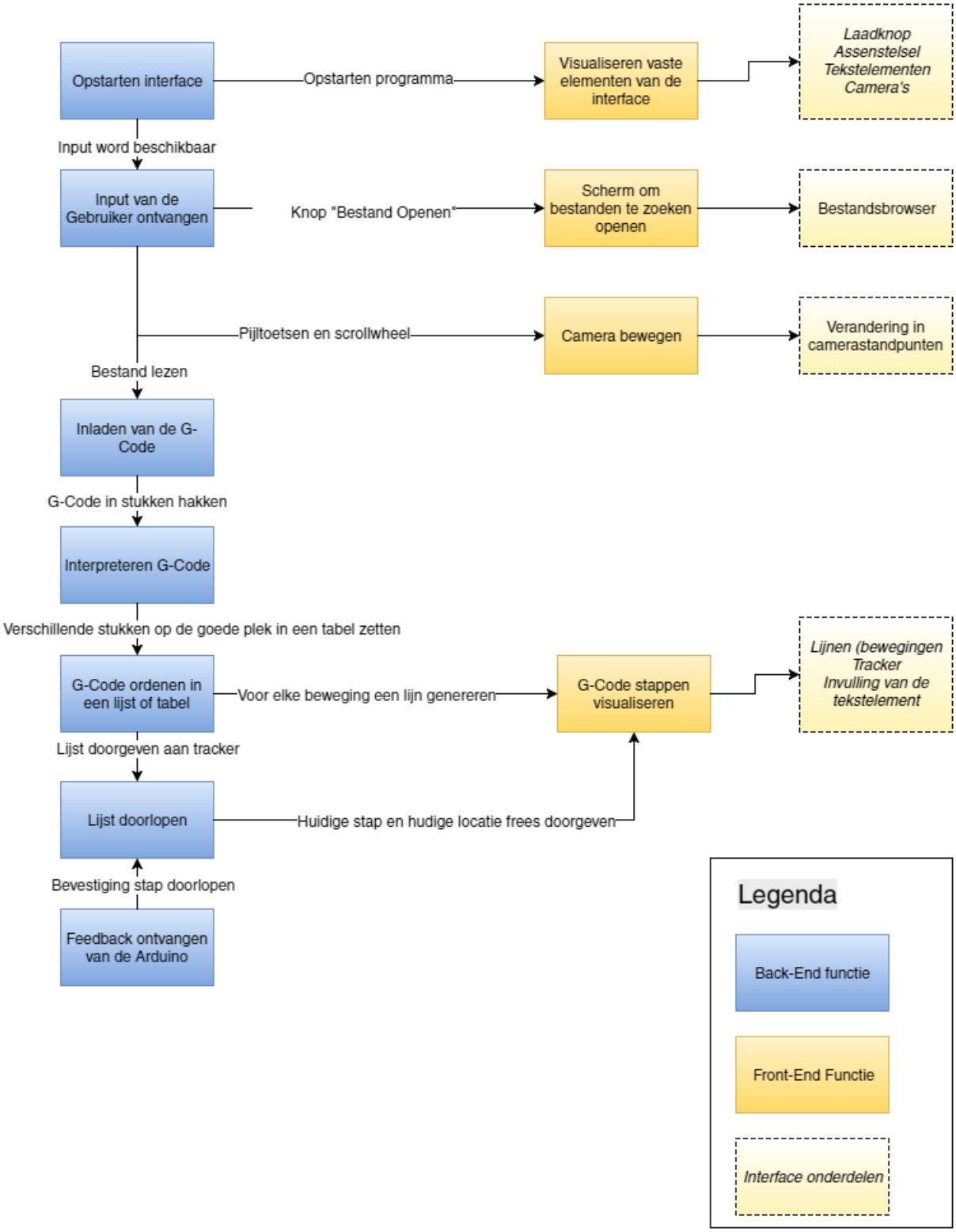
1.2 Siemens NX Diagram



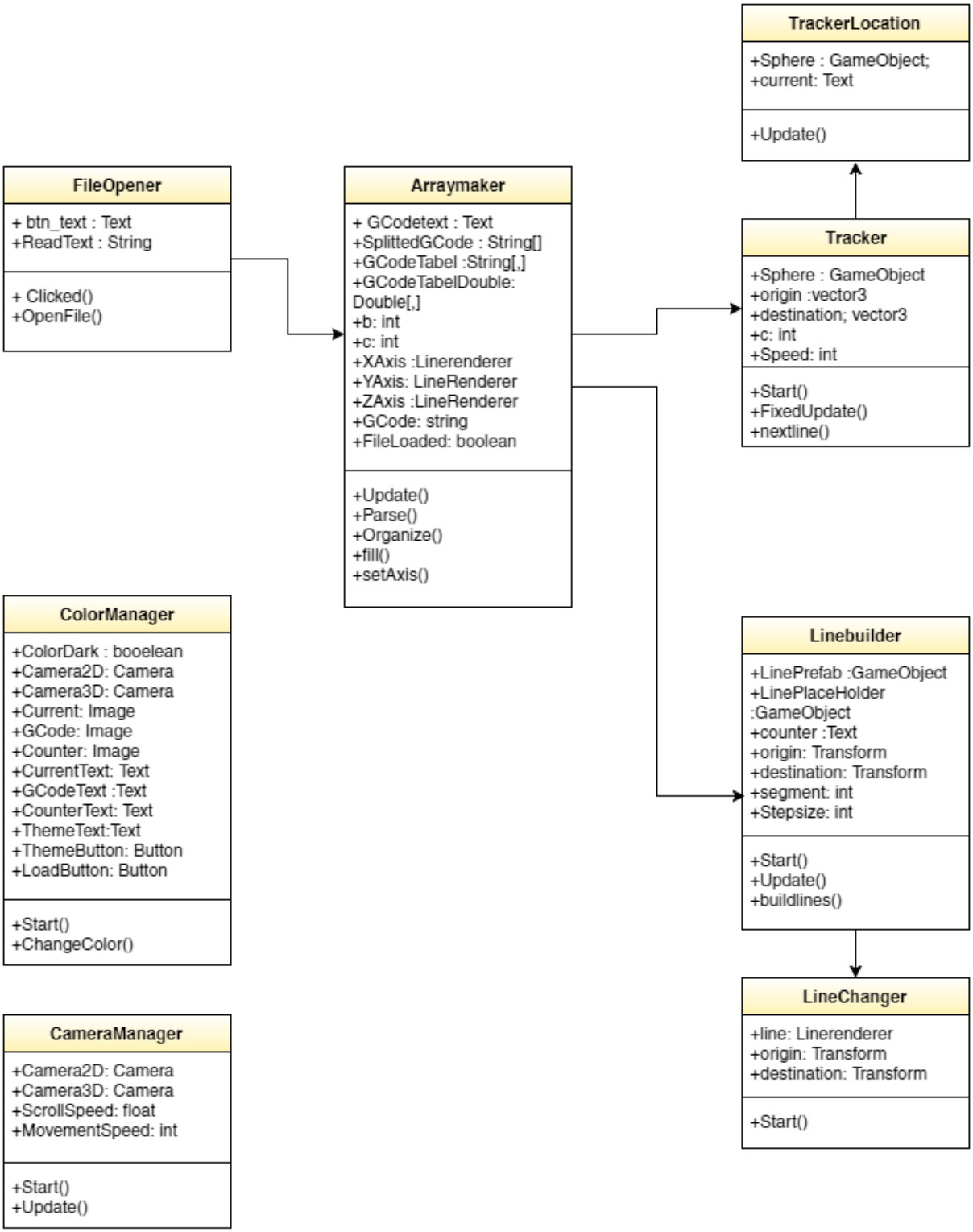
1.3 Fusion 360 Diagram



2.1 Functiediagram



3.1 Class-Diagram



4.1 Fileopener

```
// These are the libraries used in this code
using UnityEngine;
using UnityEngine.UI;
using System.IO;
using UnityEditor;

public class FileOpener : MonoBehaviour {
	// calling upon different classes, objects and variables
	public Arraymaker Arraymaker;
	public Text btn_text;
	string ReadText ;



	public void Clicked () {
		//This code will change the text of the button and call for the openfile
function
		btn_text.text = “opening File”;
		openfile ();
		return;
	}

	void openfile(){
		string path = EditorUtility.OpenFilePanel(“Open GCode”, “”, “nc”); //
opens a filebrowser. The chosen files path will be stored as String
		ReadText = File.ReadAllText (path); // Reads out the file on the speci-
fic path
		Arraymaker.GCode = ReadText; // Calls for two function of the Array-
maker Class
		Arraymaker.Parse ();
		btn_text.text = “File Opened”; // Changes the buttontext again
		return;
	}


}
```

4.2 Arraymaker

```
// These are the libraries used in this code
using UnityEngine;
using System.Collections;
using UnityEngine.UI;
using System.IO;
using System;
using System.Linq;


public class Arraymaker : MonoBehaviour {
	// calling upon different classes, objects and variables
	public Text GCodetext;
	string []SplittedGCode;
	string [,]GCodeTabel;
	int b;
	int count;
	public int c;
	public Linebuilder Linebuilder;
	public LineRenderer XAxis;
	public LineRenderer YAxis;
	public LineRenderer ZAxis;

	[HideInInspector]
	public string GCode;
	public bool FileLoaded;
	public double [,]GCodeTabelDouble;

	// Initializing start values
	void Start () {
		FileLoaded = false;
		c = 1;
		b = 0;
		count = 0;
	}


	void Update () {
		//This will show the current g-code statement the X-Carve or tracker is
working with
		if (FileLoaded == true) {
			GCodetext.text = “G:” + GCodeTabelDouble [c, 0] + “\n” +”X:” +
GCodeTabelDouble [c, 1] + “\n”
			+ “Y:” + GCodeTabelDouble [c, 2] + “\n” + “Z:” + GCodeTa-
belDouble [c, 3] + “\n”
			+ “F:” + GCodeTabelDouble [c, 4] + “\n” + “I:” + GCodeTa-
belDouble [c, 5] + “\n”
			+ “J:” + GCodeTabelDouble [c, 6] + “\n” + “K:” + GCodeTa-
belDouble [c, 7] + “\n”
```

```
                    + “L:” + GCodeTabelDouble [c, 8] + “\n” + “N:” + GCodeTa-
belDouble [c, 9] + “\n”
                    + “P:” + GCodeTabelDouble [c, 10] + “\n” + “R:” + GCodeTa-
belDouble [c, 11] + “\n”
                    + “S:” + GCodeTabelDouble [c, 12] + “\n” + “T:” + GCodeTa-
belDouble [c, 13] + “\n”+ count;

            }

        }

        public void Parse() {
            //This will split the string into seperate parts into an array
            SplittedGCode = GCode.Split (new string[] {“ “, “\n”, “” }, StringSpli-
tOptions.RemoveEmptyEntries);
            Organize ();
        }

        void Organize(){
            //Initializing arrays to fill
            GCodeTabel = new string[SplittedGCode.Length, 14];
            GCodeTabelDouble = new double[SplittedGCode.Length, 14];

            for (int i = 0; i < SplittedGCode.Length; i++){
                //This checks what kind of g-statement it is
                if (SplittedGCode [i].StartsWith (“G”)) {
                    b++; //every g is placed in a new column of the array
                    GCodeTabel [b - 1, 0] = SplittedGCode [i]; //places the
code in the riht position of the array
                    GCodeTabel [b - 1, 0] = GCodeTabel [b - 1, 0].Remove
(0, 1); //removes the G in front of it
                    GCodeTabelDouble [b - 1, 0] = double.Parse (GCodeTa-
bel [b - 1, 0], System.Globalization.CultureInfo.InvariantCulture); // changes it into
an double
                } else if (SplittedGCode [i].StartsWith (“X”)) {
                    GCodeTabel [b - 1, 1] = SplittedGCode [i];
                    GCodeTabel [b - 1, 1] = GCodeTabel [b - 1, 1].Remove
(0, 1);
                    GCodeTabelDouble [b - 1, 1] = double.Parse (GCodeTa-
bel [b - 1, 1], System.Globalization.CultureInfo.InvariantCulture);
                } else if (SplittedGCode [i].StartsWith (“Y”)) {
                    GCodeTabel [b - 1, 2] = SplittedGCode [i];
                    GCodeTabel [b - 1, 2] = GCodeTabel [b - 1, 2].Remove
(0, 1);
                    GCodeTabelDouble [b - 1, 2] = double.Parse (GCodeTa-
bel [b - 1, 2], System.Globalization.CultureInfo.InvariantCulture);
                } else if (SplittedGCode [i].StartsWith (“Z”)) {
                    GCodeTabel [b - 1, 3] = SplittedGCode [i];
                    GCodeTabel [b - 1, 3] = GCodeTabel [b - 1, 3].Remove
(0, 1);
```

```
                        GCodeTabelDouble [b - 1, 3] = double.Parse (GCodeTa-
bel [b - 1, 3], System.Globalization.CultureInfo.InvariantCulture);
                    } else if (SplittedGCode [i].StartsWith (“F”)) {
                        GCodeTabel [b - 1, 4] = SplittedGCode [i];
                        GCodeTabel [b - 1, 4] = GCodeTabel [b - 1, 4].Remove
(0, 1);
                        GCodeTabelDouble [b - 1, 4] = double.Parse (GCodeTa-
bel [b - 1, 4], System.Globalization.CultureInfo.InvariantCulture);
                    } else if (SplittedGCode [i].StartsWith (“I”)) {
                        GCodeTabel [b - 1, 5] = SplittedGCode [i];
                        GCodeTabel [b - 1, 5] = GCodeTabel [b - 1, 5].Remove
(0, 1);
                        GCodeTabelDouble [b - 1, 5] = double.Parse (GCodeTa-
bel [b - 1, 5], System.Globalization.CultureInfo.InvariantCulture);
                    } else if (SplittedGCode [i].StartsWith (“J”)) {
                        GCodeTabel [b - 1, 6] = SplittedGCode [i];
                        GCodeTabel [b - 1, 6] = GCodeTabel [b - 1, 6].Remove
(0, 1);
                        GCodeTabelDouble [b - 1, 6] = double.Parse (GCodeTa-
bel [b - 1, 6], System.Globalization.CultureInfo.InvariantCulture);
                    } else if (SplittedGCode [i].StartsWith (“K”)) {
                        GCodeTabel [b - 1, 7] = SplittedGCode [i];
                        GCodeTabel [b - 1, 7] = GCodeTabel [b - 1, 7].Remove
(0, 1);
                        GCodeTabelDouble [b - 1, 7] = double.Parse (GCodeTa-
bel [b - 1, 7], System.Globalization.CultureInfo.InvariantCulture);
                    } else if (SplittedGCode [i].StartsWith (“L”)) {
                        GCodeTabel [b - 1, 8] = SplittedGCode [i];
                        GCodeTabel [b - 1, 8] = GCodeTabel [b - 1, 8].Remove
(0, 1);
                        GCodeTabelDouble [b - 1, 8] = double.Parse (GCodeTa-
bel [b - 1, 8], System.Globalization.CultureInfo.InvariantCulture);
                    } else if (SplittedGCode [i].StartsWith (“N”)) {
                        GCodeTabel [b - 1, 9] = SplittedGCode [i];
                        GCodeTabel [b - 1, 9] = GCodeTabel [b - 1, 9].Remove
(0, 1);
                        GCodeTabelDouble [b - 1, 9] = double.Parse (GCodeTa-
bel [b - 1, 9], System.Globalization.CultureInfo.InvariantCulture);
                    } else if (SplittedGCode [i].StartsWith (“P”)) {
                        GCodeTabel [b - 1, 10] = SplittedGCode [i];
                        GCodeTabel [b - 1, 10] = GCodeTabel [b - 1, 10].Remo-
ve (0, 1);
                        GCodeTabelDouble [b - 1, 10] = double.Parse (GCode-
Tabel [b - 1, 10], System.Globalization.CultureInfo.InvariantCulture);
                    } else if (SplittedGCode [i].StartsWith (“R”)) {
                        GCodeTabel [b - 1, 11] = SplittedGCode [i];
                        GCodeTabel [b - 1, 11] = GCodeTabel [b - 1, 11].Remo-
ve (0, 1);
                        GCodeTabelDouble [b - 1, 11] = double.Parse (GCode-
Tabel [b - 1, 11], System.Globalization.CultureInfo.InvariantCulture);
```

```
                } else if (SplittedGCode [i].StartsWith (“S”)) {
                        GCodeTabel [b - 1, 12] = SplittedGCode [i];
                        GCodeTabel [b - 1, 12] = GCodeTabel [b - 1, 12].Remo-
ve (0, 1);
                        GCodeTabelDouble [b - 1, 12] = double.Parse (GCode-
Tabel [b - 1, 12], System.Globalization.CultureInfo.InvariantCulture);
                } else if (SplittedGCode [i].StartsWith (“T”)) {
                        GCodeTabel [b - 1, 13] = SplittedGCode [i];
                        GCodeTabel [b - 1, 13] = GCodeTabel [b - 1, 13].Remo-
ve (0, 1);
                        GCodeTabelDouble [b - 1, 13] = double.Parse (GCode-
Tabel [b - 1, 13], System.Globalization.CultureInfo.InvariantCulture);
                }

            }
            fill ();
            setAxis ();
            FileLoaded = true;

            Linebuilder.buildlines ();

        }

        void fill(){
            // every g-statement without coordinates will be given the coordinates
of the last statement to ensure the tracker can go through coordinates
            for(int i = 1; i<GCodeTabelDouble.GetLength(0); i++){
                if (GCodeTabelDouble [i, 1] == 0d) {
                    GCodeTabelDouble [i, 1] = GCodeTabelDouble [i-1, 1];
                }
                if (GCodeTabelDouble [i, 2] == 0d) {
                    GCodeTabelDouble [i, 2] = GCodeTabelDouble [i-1, 2];
                }
                if (GCodeTabelDouble [i, 3] == 0d) {
                    GCodeTabelDouble [i, 3] = GCodeTabelDouble [i-1, 3];
                }
            }
        }

        void setAxis(){
            //The minimal and maximal values of the Axis
            double Xmin = 0.0;
            double Xmax = 100.0;
            double Ymin = 0.0;
            double Ymax = 100.0;
            double Zmin = 0.0;
            double Zmax = 100.0;

            //If a value of a coordinate is below the min or max value it will ebco-
me the new value
```

```
                for (int i = 0; i < GCodeTabelDouble.GetLength (0); i++) {
                        if (GCodeTabelDouble [i, 1] < Xmin) {
                                Xmin = GCodeTabelDouble [i, 1];
                        } else if (GCodeTabelDouble [i, 1] > Xmax) {
                                Xmax = GCodeTabelDouble [i, 1];
                        }
                        if (GCodeTabelDouble [i, 2] < Zmin) {
                                Zmin = GCodeTabelDouble [i, 2];
                        } else if (GCodeTabelDouble [i, 2] > Zmax) {
                                Zmax = GCodeTabelDouble [i, 2];
                        }
                        if (GCodeTabelDouble [i, 3] < Ymin) {
                                Ymin = GCodeTabelDouble [i, 3];
                        } else if (GCodeTabelDouble [i, 3] > Ymax) {
                                Ymax = GCodeTabelDouble [i, 3];
                        }

                        //setting the axis
                        XAxis.SetPosition (0, new Vector3 ((float)Xmin, 0, 0));
                        XAxis.SetPosition (1, new Vector3 ((float)Xmax, 0, 0));
                        YAxis.SetPosition (0, new Vector3 (0,(float)Ymin, 0));
                        YAxis.SetPosition (1, new Vector3 (0,(float)Ymax, 0));
                        ZAxis.SetPosition (0, new Vector3 (0,0,(float)Zmin));
                        ZAxis.SetPosition (1, new Vector3 (0,0,(float)Zmax));
                }
        }

}
```

4.3 Linebuilder

```
// These are the libraries used in this code
using UnityEngine;
using System.Collections;
using UnityEngine.UI;


public class Linebuilder : MonoBehaviour {

	// calling upond idfferent classes, objects and variables
	public Arraymaker Arraymaker;
	public GameObject LinePrefab;
	public GameObject LinePlaceHolder;
	public Text counter;
	Transform origin;
	Transform destination;
	int i;
	int segment;
	int StepSize;
	GameObject[] lines;


	// initialization of startvalues
	void Start () {
		segment = 1;
		StepSize = 4000;

	}

	// Update is called once per frame
	void Update () {
		// this code will interpret the pressing of the up or down key to change
between segments of lines
		if (Input.GetKeyDown (KeyCode.UpArrow) ) {
			segment = segment + StepSize;
			buildlines ();

		}

		if (Input.GetKeyDown (KeyCode.DownArrow) ) {
			segment = segment - StepSize;
			buildlines ();

		}

	}

	public void buildlines (){
```

```
            lines =GameObject.FindGameObjectsWithTag(“lines”); //this finds all
the lines
            for (int i = 0; i < lines.Length; i++) {
                  Destroy (lines [i]); // this destroys all existing lines
            }

            //this code will initialise line-object as many as the stepsize
            for (int i = segment; i < segment + StepSize; i++) {
                  if (Arraymaker.GCodeTabelDouble [i, 0] == 0 || Arraymaker.
GCodeTabelDouble [i, 0] == 1) {
                        GameObject line = (GameObject)Instantiate (LinePre-
fab);
                        line.transform.GetChild (0).position = new Vector3
((float)Arraymaker.GCodeTabelDouble [i - 1, 1], (float)Arraymaker.GCodeTabelDou-
ble [i - 1, 3], (float)Arraymaker.GCodeTabelDouble [i - 1, 2]);
                        line.transform.GetChild (1).position = new Vector3
((float)Arraymaker.GCodeTabelDouble [i, 1], (float)Arraymaker.GCodeTabelDouble
[i, 3], (float)Arraymaker.GCodeTabelDouble [i, 2]);

                        // if it is a rapidpositioning statement it will be made
green
                        if (Arraymaker.GCodeTabelDouble [i, 0] == 0) {
                              LineRenderer linerenderer = line.gameObject.
GetComponent<LineRenderer> ();
                              linerenderer.SetColors (Color.green, Color.green);
                        }

                        //this will set Lineplaceholder as parent
                        line.transform.SetParent (LinePlaceHolder.transform);
                  }

            }

            //this updates the countertext
            counter.text = “G-Code regel: “ + segment + “ - “ + (segment + Step-
Size);
      }


}
```

4.4 Linechanger

```
using UnityEngine;
using System;
using System.Collections;
using UnityEngine.UI;

public class linechanger : MonoBehaviour {
	public LineRenderer Line;
	public Transform origin;
	public Transform destination;

	// Use this for initialization
	void Start () {
		//initialisez the position of this specific line
		Line.SetPosition (0, origin.position);
		Line.SetPosition (1, destination.position);

	}
}
```

4.5 Tracker

```
// These are the libraries used in this code
using UnityEngine;
using System.Collections;
using UnityEngine.UI;


public class Tracker : MonoBehaviour {
	// calling upon different classes, objects and variables
	public GameObject Sphere;
	Vector3 origin;
	Vector3 destination;
	int c;
	public int Speed;
	public Arraymaker Arraymaker;



	// Here the starting values will be determined
	void Start () {
		c = 1;
		Speed = 10;
		destination = Sphere.transform.position;
		}

	// This is fixed update so it will look at the time between frames instead of
the count of frames
	void FixedUpdate () {
		if (Arraymaker.FileLoaded == true) {
```

```
                        nextline ();

                        if (Sphere.transform.position == destination) {
                                c++;
                                Arraymaker.c++; // this will tell arraymaker it can continue to the next line of g-code
                                //makes a new destination for the tracker based on the g-code
                                destination = new Vector3 ((float)Arraymaker.GCodeTabelDouble [c, 1], (float)Arraymaker.GCodeTabelDouble [c, 3], (float)Arraymaker.GCodeTabelDouble [c, 2]);
                                //based on the type of g-code the speed will vary
                                if (Arraymaker.GCodeTabelDouble [c, 0] == 0) {
                                        Speed = 20;
                                } else {
                                        Speed = 10;
                                }

                        }


                }
        }

        void nextline(){
                // this will move the tracker towards its endpoint with a constant speed
                Sphere.transform.position = Vector3.MoveTowards (Sphere.transform.position, destination, Time.deltaTime*Speed);

        }

}
```

4.6 TrackerLocation

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;

public class TrackerLocation : MonoBehaviour {
        public Arraymaker Arraymaker;
        public Tracker Tracker;
        public GameObject Sphere;
        public Text current;


        // Update is called once per frame
        void Update () {
```

```
		//this looks every frame where the tracker is and sends it to a text
component
		if (Arraymaker.FileLoaded == true) {
			current.text = “X:” + Sphere.transform.position.x + “\n”
				+ “Y:” + Sphere.transform.position.z + “\n” + “Z:” + Sphe-
re.transform.position.y + “\n” + “Speed: “ + Tracker.Speed + “ mm/s “;
		}
	}
}
```

4.7 CameraManager

```
using UnityEngine;
using System.Collections;

public class CameraManager : MonoBehaviour {
	// calling upon different classes, objects and variables
	public Camera Camera2D;
	public Camera Camera3D;
	float ScrollSpeed;
	int MovementSpeed;


	// Initializing start values
	void Start () {
		ScrollSpeed = 100f;
		MovementSpeed = 5;
		}

	// Update is called once per frame
	void Update () {
		//changes to 2D camera and resets it
		if (Input.GetKeyDown (KeyCode.Alpha1)) {
			Camera3D.gameObject.SetActive (false);
			Camera2D.gameObject.SetActive (true);
			Camera2D.orthographicSize = 180;
		}

		//changes to 3D camera and resets it
		if (Input.GetKeyDown (KeyCode.Alpha2)) {
			Camera2D.gameObject.SetActive (false);
			Camera3D.gameObject.SetActive (true);
			Camera3D.orthographicSize = 180;
		}
		 //if it is scrolled the size/zoom of the camera will change
		if (Camera3D.isActiveAndEnabled) {
			Camera3D.orthographicSize += Input.GetAxis (“Mouse Scroll-
Wheel”) * ScrollSpeed;
		}
		//if it is scrolled the size/zoom of the camera will change
		if (Camera2D.isActiveAndEnabled) {
```

```
                Camera2D.orthographicSize += Input.GetAxis (“Mouse Scroll-
Wheel”) * ScrollSpeed;

        }
        //moves the camera over the axis
        if (Input.GetKey (KeyCode.W)) {
                if (Camera2D.isActiveAndEnabled == true) {
                        Camera2D.transform.position += new Vector3 (0,0,Mo-
vementSpeed);
                }
                if (Camera3D.isActiveAndEnabled == true) {
                        Camera3D.transform.position += new Vector3 (0,0,Mo-
vementSpeed);
                }
        }

        if (Input.GetKey (KeyCode.S)) {
                if (Camera2D.isActiveAndEnabled == true) {
                        Camera2D.transform.position -= new Vector3 (0,0,Move-
mentSpeed);
                }
                if (Camera3D.isActiveAndEnabled == true) {
                        Camera3D.transform.position -= new Vector3 (0,0,Move-
mentSpeed);
                }
        }

        if (Input.GetKey (KeyCode.D)) {
                if (Camera2D.isActiveAndEnabled == true) {
                        Camera2D.transform.position += new Vector3 (Move-
mentSpeed,0,0);
                }
                if (Camera3D.isActiveAndEnabled == true) {
                        Camera3D.transform.position += new Vector3 (Move-
mentSpeed,0,0);
                }
        }

        if (Input.GetKey (KeyCode.A)) {
                if (Camera2D.isActiveAndEnabled == true) {
                        Camera2D.transform.position -= new Vector3 (Move-
mentSpeed, 0, 0);
                }
                if (Camera3D.isActiveAndEnabled == true) {
                        Camera3D.transform.position -= new Vector3 (Move-
mentSpeed, 0, 0);
                }
        }
        if (Input.GetKey (KeyCode.Q)) {
                if (Camera2D.isActiveAndEnabled == true) {
```

```
                Camera2D.transform.position += new Vector3 (0,Move-
mentSpeed,0);
            }
            if (Camera3D.isActiveAndEnabled == true) {
                Camera3D.transform.position += new Vector3 (0,Move-
mentSpeed,0);
            }
        }

        if (Input.GetKey (KeyCode.E)) {
            if (Camera2D.isActiveAndEnabled == true) {
                Camera2D.transform.position -= new Vector3 (0,Move-
mentSpeed,0);
            }
            if (Camera3D.isActiveAndEnabled == true) {
                Camera3D.transform.position -= new Vector3 (0,Move-
mentSpeed,0);
            }
        }
    }
}
```

4.8 ColorManager

```
// These are the libraries used in this code
using UnityEngine;
using System.Collections;
using UnityEngine.UI;

public class ColorManager : MonoBehaviour {
    // calling upon different classes, objects and variables
    bool ColorDark;
    public Camera Camera2D;
    public Camera Camera3D;
    public Image Current;
    public Image GCode;
    public Image Counter;
    public Text CurrentText;
    public Text GCodeText;
    public Text CounterText;
    public Text ThemeText;
    public Button ThemeButton;
    public Button LoadButton;

    GameObject[] lines;



    // This will set all the starting colors
```

```
        void Start () {
                Camera2D.backgroundColor = new Color (0.2588f,0.2588f,0.2588f);
                Camera3D.backgroundColor = new Color (0.2588f,0.2588f,0.2588f);
                Current.color = new Color (1f,1f,1f, 0.75f);
                GCode.color = new Color (1f,1f,1f, 0.75f);
                Counter.color = new Color (1f,1f,1f, 0.75f);
                ThemeButton.GetComponent<Image>().color = new Color
(0.74f,0.74f,0.74f);
                LoadButton.GetComponent<Image>().color = new Color
(0.74f,0.74f,0.74f);
                ColorDark = true;
        }

        public void ChangeColor(){
                //This code will switch between the dark en light theme and change
the colors of each object seperately
                if (ColorDark == true) {
                        Camera2D.backgroundColor = new Color (0.933f, 0.933f,
0.933f);
                        Camera3D.backgroundColor = new Color (0.933f, 0.933f,
0.933f);
                        Current.color = new Color (0.74f, 0.74f, 0.74f, 0.75f);
                        GCode.color = new Color (0.74f, 0.74f, 0.74f, 0.75f);
                        Counter.color = new Color (0.74f, 0.74f, 0.74f, 0.75f);
                        ThemeButton.GetComponent<Image>().color = new Color
(0.74f,0.74f,0.74f);
                        LoadButton.GetComponent<Image>().color = new Color
(0.74f,0.74f,0.74f);

                        lines =GameObject.FindGameObjectsWithTag(“lines”); //This
will find all lines and change their color
                        for (int i = 0; i < lines.Length; i++) {
                                lines[i].GetComponent<LineRenderer>().SetColors (new
Color (0.27f,0.35f,0.39f),new Color (0.27f,0.35f,0.39f));
                        }

                        ThemeText.text = “Light Theme”; //this will change the text of
the button
                        ColorDark = false;
                }
                else if (ColorDark == false) {
                        Camera2D.backgroundColor = new Color
(0.2588f,0.2588f,0.2588f);
                        Camera3D.backgroundColor = new Color
(0.2588f,0.2588f,0.2588f);
                        Current.color = new Color (1f,1f,1f, 0.75f);
                        GCode.color = new Color (1f,1f,1f, 0.75f);
                        Counter.color = new Color (1f,1f,1f, 0.75f);

                        ThemeButton.GetComponent<Image>().color = new Color
```

```
(0.74f,0.74f,0.74f);
                        LoadButton.GetComponent<Image>().color = new Color
(0.74f,0.74f,0.74f);


                        lines =GameObject.FindGameObjectsWithTag("lines");
                        for (int i = 0; i < lines.Length; i++) {
                                lines[i].GetComponent<LineRenderer>().SetColors (new
Color (0.933f, 0.933f, 0.933f),new Color (0.933f, 0.933f, 0.933f));
                        }

                        ThemeText.text = "Dark Theme";
                        ColorDark = true;
                }

        }
}
```