KU Leuven
Departement Computerwetenschappen

P&O: Computerwetenschappen

Tussentijds verslag

Team:
Geel

Coomans Arno
(Coördinator)
Saniz Bernardo
(Secretaris)
Geens Tomas
Peeters Florian
Theuns Flor
Willemot Toon

Academiejaar 2017-2018

Samenvatting

Elk autonoom systeem moet een aantal opeenvolgende taken kunnen realiseren. Dit rapport beschrijft het ontwerp van een virtuele simulatie, waarin aan autonome drone zo een aantal taken moet uitvoeren. Zo een machine moet zelfstandig opstijgen, naar een doel manoeuvreren, landen en over de grond naar een bestemming rijden. Hier worden deze taken verder in detail uitgewerkt, alsook een algoritme om een pad doorheen een aantal doelen te bepalen. Een aantal verbeteringen en correcties op de oude code worden ook toegelicht. De resultaten in dit verslag zijn belangrijke gegevens voor het verder verfijnen van een autopiloot, en later om verschillende vliegtuigen gecoördineerd te laten vliegen.

Inhoudsopgave

1 Inleiding

Arno

In het nieuwe deel van de opgave worden meerdere drones, luchthavens en een centrale motion planning toegevoegd. Voor deze tussentijdse evaluatie ligt de focus op het opstijgen, landen en taxiën, maar we bereiden ons ook voor om deel 2 van de opgave van dit semester makkelijker te implementeren. Zo zijn bijvoorbeeld de wereld en render klasse voorbereid om meerdere drones te ondersteunen.

Het op voorhand plannen van de bewegingen van de drone wordt dus belangrijker, en daar ligt ook de focus van dit verslag. Tot nu toe kon de drone geen bochten nemen, iets dat broodnodig is voor de nieuwe opgave. Eén van onze doelstellingen was om deze functionaliteit toe te voegen. Zolang bochten nemen niet lukt kunnen verschillende opgelegde taken niet voldaan worden. Zoals altijd waren er verschillende kleine verbeteringen en bug fixes, waarvan de belangrijkste kort vermeld worden.

2 Ontwerp

2.1 Droneparameters

Bernardo

De meeste parameters lagen al vast, maar niet die in verband met de wielen van de drone. Het landingsgestel is gebaseerd op onze berekeningen voor de MQ-1 Predator. dampSlope en tyreSlope zijn verhoogd om het botsen van de drone bij landing te verminderen.

Tabel 1: Gebruikte AutopilotConfig

Parameter	**Vastgelegde waarde**
gravity	9,81
wingX	3,35
tailSize	5,11
engineMass	190,24
wingMass	102,80
tailMass	116,15
maxThrust	4500
maxAOA	15
wingLiftSlope	12,5
horStabLiftSlope	6,25
verStabLiftSlope	4,69
horizontalAngleOfView	120
verticalAngleOfView	120
nbColumns	200
nbRows	200

Parameter	**Gekozen waarde**
wheelY	-1,37
frontWheelZ	-2,10
rearWheelZ	1,00
rearWheelX	1,39
tyreSlope	50000
dampSlope	5000
tyreRadius	0,2
rMax	2000
fcMax	0,7

3 Pathfinding

Toon, Tomas

Vooraleer een drone mag vertrekken moet er een pad dat een aantal doel-locaties bevat aan toegekend worden. De bepaling van zo een pad wordt in deze sectie besproken.

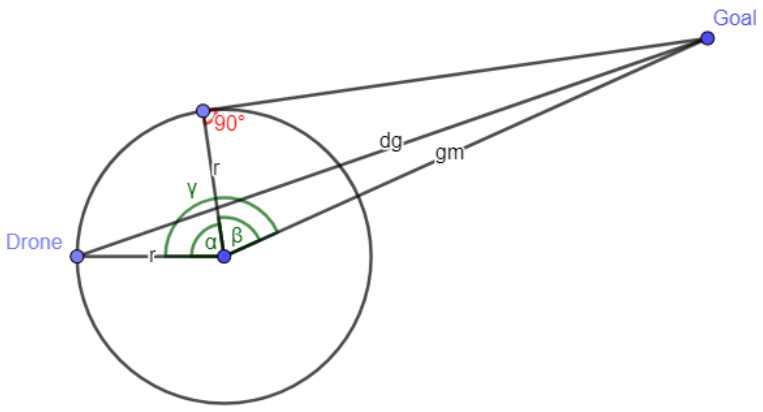
3.1 Implementatie

De volgorde waarin de drone naar de doel-locaties vliegt is op een greedy manier bepaald. Na ieder behaald doel berekent het programma welke locatie het dichtstbij ligt en stelt deze locatie als volgend doel. Het vliegen tussen twee doel-locaties is opgesplitst in drie fases. Bij iedere fase houdt de autopiloot een aantal punten bij die een pad maken dat de drone moet volgen. In de eerste fase stijgt/daalt de drone tot hij op dezelfde hoogte zit als zijn doel-locatie. In de tweede fase draait de drone tot zijn heading richting doel-locatie gericht staat. In de derde fase vliegt de drone rechtdoor tot de doel-locatie bereikt is.

Het uitwerken van de eerste fase is vrijwel eenvoudig. Het hoogteverschil tussen drone en doel-locatie bepaalt of de drone moet stijgen of dalen. De $maxInclination$ of $maxDeclination$ van de drone bepaalt de afstand die nodig is om op dezelfde hoogte als de doel-locatie te komen. Met deze afstand en de heading van de drone wordt een (tussen)positie bepaald en aan het pad toegevoegd. De coördinaten van het punt na stijging zijn in vergelijking 1 berekend, waarbij *current* de beginlocatie is en *goal* de doellocatie. Het dalen gebeurt analoog.

$$\begin{cases} x = current.x + \frac{sin(heading)\cdot(goal.y - current.y)}{maxInclination} \\ y = goal.y \\ z = current.z - \frac{cos(heading)\cdot(goal.y - current.y)}{maxInclination} \end{cases} \tag{1}$$

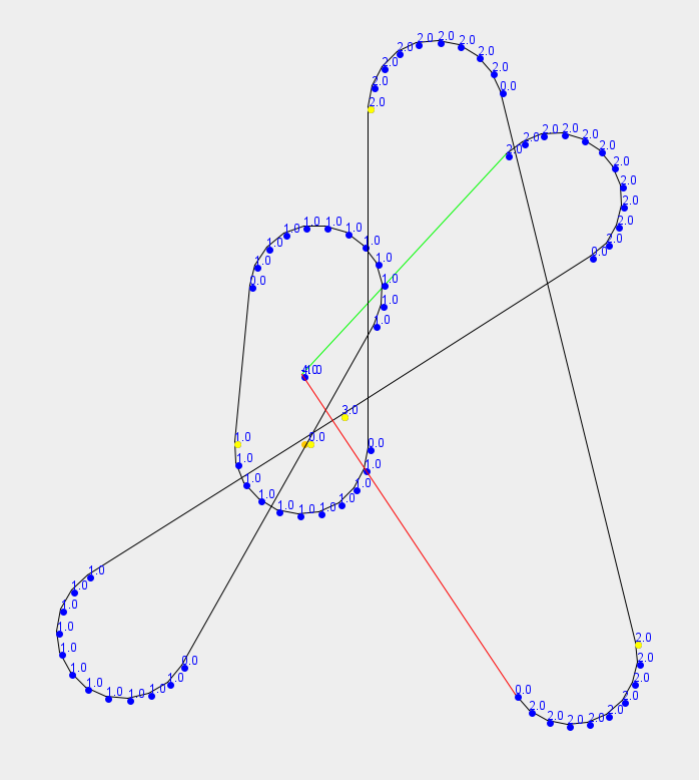
De tweede fase is minder eenvoudig. Eerst wordt gekeken of de doel-locatie binnen het draaibereik van de drone ligt. Dit wordt bepaald a.d.h.v. de draaicirkel van de drone. Is dit niet het geval, laten we de drone rechtdoor vliegen tot de doel-locatie wel binnen draaibereik ligt. Dan moet de drone bepalen hoelang hij de draaicirkel moet volgen zodat zijn heading naar het doel gericht is. in figuur 1 zoeken we α. De straal r is gekend. De afstand tussen het doel en het middenpunt gm en de afstand tussen de drone en het doel dg kunnen we berekenen met de formule voor afstand tussen twee punten. Nu passen we de cosinus-regel toe: $\gamma = acos(\frac{dg^2 - gm^2 - r^2)}{-2\cdot r\cdot gm}$. Omdat de raaklijn van een cirkel een rechte hoek maakt met de straal geldt het volgende: $\beta = acos(\frac{r}{mg})$. En uiteindlijk $\alpha = \gamma - \beta$. Met deze hoek, de initiële heading en de straal van de draaicirkel kunnen we een willekeurig aantal punten op de cirkel aan het pad toevoegen.



Figuur 1: Afbeelding voor het berekenen van de draailengte.

De derde fase is simpelweg rechtdoor vliegen. Het laatste punt dat we toevoegen is het doel zelf.

Voor elke locatie in het uiteindelijke pad wordt ook bijgehouden wat het vliegtuig moet doen om naar de volgende locatie te gaan, e.g. 'turn left/right', 'fly straight', 'ascend/descend'. Zo krijgt de autopilot een idee van wat hij moet doen om het pad succesvol af te leggen. Figuur 2 is een voorbeeld van zo'n pad.



Figuur 2: Een voorbeeld van een pad.

4 Physics engine

Florian

De physics is ten opzichte van vorig semester zeer weinig veranderd. Alle opgestelde vergelijkingen en oplossingsmethodes zijn onveranderd, en worden hier dus ook niet meer vermeld. De enige verandering bestaat uit de toevoeging van wielen aan het vliegtuig. Deze wielen zorgen voor een verticale kracht die het vliegtuig omhoog houdt als het op de grond staat, en eventueel een remkracht om te vertragen.

De verticale kracht in elk wiel is gedefiniëerd met de volgende formule: $F_v = S_t \cdot D + S_d \cdot \frac{dD}{dt}$. Hier is S_t de *tyreSlope* en S_d de dampSlope, de waarde voor deze constanten wordt besproken in sectie 2.1. D is de radiale indrukking van de band. De remkracht op de wielen ligt in het grondvlak, en tegengesteld aan de snelheid van het wiel. Als het wiel geen snelheid heeft, zorgt de remkracht ervoor dat het vliegtuig niet beweegt. De grootte van de kracht wordt door de autopiloot ingesteld, met een waarde tussen 0 en de maximale remkracht. De twee achterste wielen ondervinden ook nog een zijwaartse wrijving, volgens de x-as van de drone, de grootte is gegeven door: $F_w = fcMax \cdot v \cdot N$. Hier is $fcMax$ de wrijvingsconstante van de wielen, v de snelheid van het wiel en N de grootte van de verticale kracht op het wiel.

4.1 Rotatieproblemen

Vorig semester waren er nog problemen met het draaien van de drone, en dit bleef moeilijk in het begin van dit semester. Toen we niet alleen tijdens het vliegen vreemd gedrag opmerkten, maar ook tijdens het taxiën, werd het duidelijk dat het probleem in de physics lag. Dit werd bevestigd door de drone bij het opstarten niet in de richting van de negatieve z-as te zetten, maar bijvoorbeeld de x-as. Dan begon de drone te draaien rond zijn z-as, wat niet de bedoeling is. Na nog een aantal testen werd het duidelijk dat de rotatie van de drone enkel correct was als hij volgens de negatieve z-as ging.

Na uitgebreid zoeken in de code, hebben we opgemerkt dat de rotatiesnelheid, die we intern berekenen in het drone-assenstelsel, niet getransformeerd werd om de rotatie van de drone te berekenen in het wereldassenstelsel. Dit levert inderdaad juist gedrag op als beide assenstelsels niet gedraaid staan ten opzichte van elkaar, maar vanaf ze gedraaid werden liep het mis. De oplossing was uiteindelijk zeer gemakkelijk, namelijk de rotatiesnelheid transformeren naar het wereldassenstelsel voordat we de rotatie ermee berekenen.

5 Motion Planning

De motion planning van de drone is in tegenstelling tot vorig semester niet meer gebaseerd op de image recognition maar op de gegeven coördinaten. De gegeven coördinaten worden met behulp van de pathfinding omgezet tot een reeks van commando's die in de motion planning zijn geïmplementeerd. Verder gaat de image recognition nog gebruikt worden om de locatie van de kubussen die niet helemaal nauwkeurig is beter te bepalen en zo de vlucht lichtjes aan te passen. Dit is nog niet geïmplementeerd. We splitsen de vlucht op in 4 delen: het taxiën, het opstijgen, het vliegen en het landen. Hieronder wordt elk onderdeel verder beschreven.

5.1 Taxiën

Bernardo

Taxiën is het van en naar de landingsbaan rijden van een vliegtuig. Bij het taxiën kan de drone enkel manoeuvreren door gebruik te maken van zijn motor en remmen. Een eenvoudige en courante strategie hiervoor is differentiële sturing. Het principe hiervan gaat als volgt. Als het linker- en rechterwiel van een robot aan verschillende snelheden draaien, beschrijven ze allebei cirkels met verschillende straal. Op deze manier volgt de robot een cirkelboog tussen deze twee draaicirkels. De linker- en rechterrem asymmetrisch aanzetten zorgt voor dit snelheidsverschil.

Het rijden bestaat uit een aantal stappen. Eerst richt de drone zich naar zijn doel in het xz-vlak, tot op een bepaalde hoek nauwkeurig. De thrust wordt op een lage waarde gezet en één van de remmen op het maximum. Eén van de wielen blijft volledig stil, terwijl het massacentrum van de drone licht versnelt. Hierdoor gaat het vliegtuig strak draaien. Als het vliegtuig naar het doel gericht is versnelt het tot ongeveer 4 m/s. Zijn initiële uitlijning is natuurlijk niet perfect. Als de afwijking van de drone tegenover de gewenste richting te groot wordt, slaat één van de remmen kort aan om te compenseren. Wanneer de drone binnen 15 meter van zijn doel komt vertraagt hij tot 1 m/s. Hierdoor kan hij weer scherpe bochten nemen, moest dat nodig zijn. Deze snelheid is ook een vereiste voor het succesvol bereiken van het doel.

Omdat de wielen weinig wrijving ondervinden, moet de drone aan de hand van de voorrem vertragen. PID-controllers sturen zowel de thrust als de remkracht aan. Deze zijn goed afgesteld, en de snelheid van de drone ligt consistent binnen 0.2 m/s van de gewenste snelheid. De drone is in staat opeenvolgende doelen te bereiken en doet dit snel. Wanneer de richtingsverandering als gevolg hiervan heel groot is, treedt er wel soms een AOAException op. De exacte omstandigheden waarbij dit gebeurt zijn echter moeilijk te bepalen.

5.2 Opstijgen

Florian

Tijdens het opstijgen moet de drone uit stilstand vertrekken, om uiteindelijk een bepaalde hoogte te halen. Deze hoogte wordt bepaald door de pathfinding, en vanaf dan begint het volgende deel van de vlucht: het vliegen. Het volledige opstijgen verloopt in 1 richting, de autopiloot kan nog niet al draaiend klimmen. Dit lijkt ons wel perfect haalbaar, maar bleek voor de huidige taak niet nodig.

Het opstijgen wordt intern nog verder opgedeeld in twee eenvoudige onderdelen: versnellen en klimmen. Bij het versnellen wordt de motor op volle kracht ingesteld, en staan alle vleugels in neutrale stand om geen tegenkracht te veroorzaken. Vanaf het moment dat de drone genoeg snelheid heeft, wordt er omhoog gepitcht met de verticale stabilisator om te beginnen klimmen. De benodigde snelheid om op te kunnen stijgen hebben we experimenteel bepaald, en ligt momenteel op 120 km/u. De drone kan al vanaf een aanzienlijk lagere snelheid opstijgen, maar dan bleek het moeilijker om snel omhoog te pitchen. Een andere reden voor de ruime marge op de opstijgsnelheid is dat er momenteel geen limiet ligt op de lengte van de landingsbaan.

Men kan de benodigde afstand om een bepaalde snelheid te halen benaderen met formule 3. Deze formule komt uit het oplossen van het stelsel met de bewegingsvergelijkingen (zie vergelijking: 2). Hierin is m de massa, F de kracht van de motor, v de te behalen snelheid. Hiermee berekent men de versnelling a, de benodigde tijd t en de benodigde afstand x. Onze drone bijvoorbeeld zou 133.33 m nodig hebben. Experimenteel blijkt dat de vermelde formule eerder een absoluut minimum geeft, want om echt 120 km/u te halen is er eigenlijk ongeveer 160 m nodig. Dit komt doordat de vleugels niet perfect horizontaal staan tijdens het versnellen, en dus weerstand opleveren.

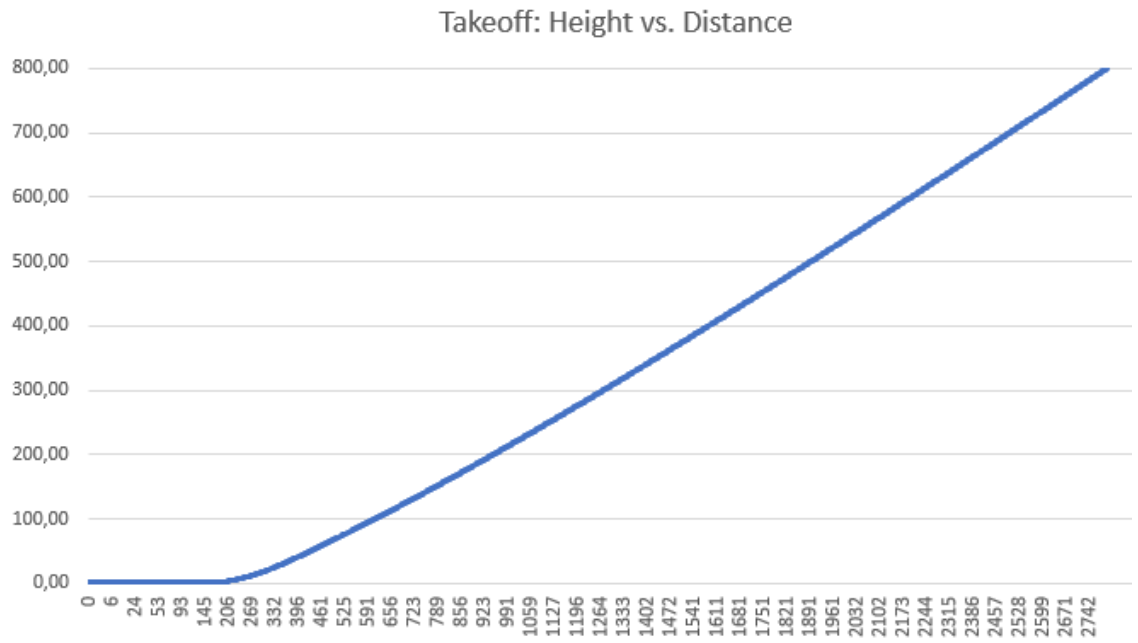
$$\begin{cases} F = m \cdot a \\ v = a \cdot t \\ x = \frac{a \cdot t^2}{2} \end{cases} \tag{2}$$

$$x = \frac{v^2}{2 \cdot a} = \frac{m \cdot v^2}{2 \cdot F} \tag{3}$$

Vanaf het moment dat de drone omhoog begint te pitchen, wordt de instelling van de horizontale stabilisator geregeld door een PD-controller. Dit is een PID-controller waarbij er geen integrale component is. Deze wordt ingesteld op de gewenste klimhoek, en zorgt er dan voor dat de drone naar die klimhoek zal pitchen. Experimenteel bleek 20° een goede klimhoek te zijn, en waarden van respectievelijk 1.1 en 0.3 voor de P en D constanten in de controller gaven een goed klimgedrag. Om verder een constante lift te genereren met de vleugels, worden die vast ingesteld op een inclinatie van 6° tijdens het klimmen. Zo blijft de drone verdergaan totdat de gevraagde hoogte bereikt is.

Voor de pathfinding hadden we een schatting nodig van de afstand die de drone nodig heeft om tot een bepaalde hoogte te komen. Hiervoor is een grafiek (zie figuur 3) opgesteld van de bereikte hoogte in functie van de afstand. Uit die grafiek bleek dat het verband tussen beide praktisch lineair is als de hoogte hoog genoeg is. De functie bekomen met regressie levert dan de benodigde afstand in functie van de gevraagde hoogte op, te zien in vergelijking 4. Voor een hoogte onder 50m is geen van beide formules accuraat, maar dit zal normaal nooit voorkomen.

$$x = \begin{cases} 254.0966 + 3.5519 \cdot h & 50 < h <= 150 \\ 343.0420 + 3.0874 \cdot h & 150 < h \end{cases} \tag{4}$$



Figuur 3: Grafiek van het verloop van het opstijgen, met de bereikte hoogte op de y-as in functie van de afgelegde afstand.

5.3 Vliegen

Flor

Het vliegen bevat het deel nadat het opstijgen het vliegtuig tot op een zekere hoogte heeft gebracht totdat het vliegtuig volledig wordt gestabiliseerd en een zo laag mogelijke snelheid heeft zodat het landen kan beginnen. Tussen opstijgen en landen moet het vliegtuig verschillende kubussen raken waarvan de volgorde en de weg die hij aflegt volledig bepaald worden door de pathfinding. Het vliegen is ook opgedeeld in verschillende states, het draaien in beide richtingen en het stijgen en dalen zoals voorheen al was geïmplementeerd. Er is ook een state toegevoegd die het landen voorbereidt door het vliegtuig op een lage hoogte te brengen en de snelheid te verlagen. Door de nieuwe parameters van de drone moest de tuning van de PID's opnieuw gebeuren en moesten er nieuwe waarden voor de snelheid en thrust worden bepaald.

5.4 Landen

Tomas

Voor het vliegtuig begint met landen is het vliegtuig gestabiliseerd en heeft het geen roll en geen verticale snelheid meer. Dit wordt in het deel van het vliegen bekomen. Nu is een eerste stap om de maximale snelheid te bepalen waarmee het vliegtuig mag landen. We maken een opstelling waar de drone begint op de grond en een snelheid heeft volgens de negatieve y-as. Het vliegtuig crasht wanneer we deze snelheid boven 1.8 m/s kiezen, dit is dus de maximale landingssnelheid. Ten tweede moet de drone landen op de wielen die het dichtst bij het massacentrum liggen. Anders zou er een te groot moment veroorzaakt worden. In dit geval zijn dit de achterste wielen. Deze doelen kunnen we bereiken door het vliegtuig een positieve pitch te geven. Zo landt het vliegtuig op zijn achterwielen en als het vliegtuig te snel daalt kan het zijn thrust aanzetten en is er een acceleratie naar boven. Experimenteel vonden we dat een goede waarde voor deze pitch 5° is.

Eenmaal de drone de grond raakt, begint hij met remmen. Nu kunnen we berekenen hoe ver de drone zal bollen als alle remmen (3 in totaal) maximaal staan.De formule berekend in vergelijking 6 volgt uit stelsel 5. Als we de waarden van onze drone invullen met een startsnelheid van 40m/s, vinden we 192m.

$$\begin{cases} 3F = m \cdot a \\ 0 = a \cdot t + v_0 \\ x = \frac{a \cdot t^2}{2} + v_0 \cdot t \end{cases} \tag{5}$$

$$x = -\frac{v_0^2}{2 \cdot a} - \frac{v_0^2}{a} = -\frac{3 \cdot v_0^2}{2 \cdot a} = \frac{3 \cdot m \cdot v_0^2}{2 \cdot F} \tag{6}$$

6 Rendering

Arno

Naast het toevoegen van landingsbanen, wielen en een grond is er op het vlak van rendering niet veel veranderd. Deze onderdelen toevoegen heeft maar één probleem veroorzaakt, namelijk dat objecten die dicht bij elkaar staan alternerend op de voorgrond zouden gerenderd worden. Een oplossing hiervoor was de Z_{near} value van de view frustum verder weg zetten[1]. Zo verliest een camera detail van objecten die zich dichtbij bevinden. In OpenGL wordt het gebied dichtbij de camera gerendered met veel precisie, waardoor de Z_{buffer} vol staat met gedetailleerde informatie die in onze applicatie nooit gebruikt gaat worden. In onze huidige render engine betekende dat het volgende:

$$Z_{near} : 0.01 \rightarrow 0.7 \tag{7}$$

Aangezien de 70 centimeter voor de drone zo goed als verwaarloosbaar zijn voor image recognition is het geen probleem om het Z_{near} plane zo ver naar voor te duwen. Later kunnen we nu met gemak het Z_{far} plane tot $25km$ verhogen. Momenteel wordt die waarde wel op $10km$ gehouden om het renderen lightweight te houden.

6.1 Skybox en grond

De top en right cam zijn heel onoverzichtelijk geworden door het landen en opstijgen, ook het verschill tussen vliegen en naar beneden vallen was niet helemaal duidelijk. Naast de grond die in de opgave gespecifieerd is hebben we ook, in de right ortho cam, een skybox toegevoegd met een texture waardoor we hoogteverschillen van de drone ook duidelijk kunnen maken. Deze texture in de lucht wordt niet getekend in de andere camera's voor efficiency en minder clutter in de grafische voorstelling.

7 Besluit

Bernardo

Alle afzonderlijke componenten van de simulatie werken zoals verwacht. Pathfinding geeft een correct en haalbaar pad terug. Elke afzonderlijke stap van de motion planning werkt goed, en de drone gedraagt zich correct en stabiel als stappen gecombineerd worden. Textures voor de grond en landingsbaan maken de positie en snelheid van de drone duidelijk zichtbaar. De visuele representatie van de drone heeft nu wielen die contact met de grond duidelijk maken. Het grootste resultaat tot nu toe, is de recentste update aan de physics engine. Alle transformaties van snelheden en hoeksnelheden gebeuren nu correct, wat het correct werken van alle aspecten van de motion-planning toelaat. Dit klaart problemen op die sinds het eerste deel van dit project aanwezig waren.

De volgende stap is alle componenten samenbrengen. Daarbij zullen ongetwijfeld nieuwe problemen ontstaan, maar die zullen hoogstwaarschijnlijk te wijten zijn aan de integratie van de componenten, en niet aan hun implementatie zelf.

Referenties

[1] S. Baker, *Learning to love your z-buffer.* sjbaker.org, 2002.