P&O Computerwetenschappen
2016–2017 Semester 1
Een automatische piloot en virtuele testomgeving voor drones

3 oktober 2016

1 Algemene praktische informatie

1.1 Context

In het OPO *Probleemoplossen en ontwerpen: Computerwetenschappen* oefent de student zich, onder begeleiding van de assistenten en de docenten, en in interactie met de mede-groepsleden, in het ontwerpen van oplossingen (met een overwegende software-component) voor technische problemen, onder gegeven omstandigheden wat betreft tijdsbestek, beschikbare middelen, voorkennis, en human resources (mede-groepsleden). Niet alleen vergaart de student zo technische domeinkennis en past hij de technische kennis en vaardigheden toe die hij in andere OPO's vergaard heeft, maar ook en vooral versterkt hij zijn algemene vaardigheden zoals analytisch denken, projectbeheer, risicobeheersing, self-management, communicatie, en people skills.

Het is dus expliciet de bedoeling dat de student te maken zal krijgen met, en professioneel om zal moeten gaan met o.a.:

- Technische problemen van allerlei aard die niet door het didactisch team voorzien waren en waar zij geen pasklare oplossing voor hebben.
- Een heterogeen team waarvan de leden verschillende vaardigheden en voorkennis hebben.
- Beperkte tijd en middelen.
- Onduidelijke of zelfs contradictorische vereisten, die dan in interactie met de opdrachtgever (in casu het didactisch team) uitgeklaard moeten worden, waarbij de student proactief oplossingen voorstelt. (Requirements elicitation and analysis is een significant onderdeel in elk software-ontwikkelingsproces.)

Als kader voor deze oefening bepaalt het didactisch team een opgave. Voor de opgave voor dit academiejaar beschouwen we het probleem van het bouwen van een Drone Autopilot, een systeem dat een drone (en meer bepaald een quadcopter) aanstuurt om allerlei taken (bv. het bezorgen van pakjes of het winnen van een drone race) te vervullen.

1.2 Didactisch team

Docenten: Hendrik Blockeel (coördinator), Bart Jacobs en Dirk Nuyens.
Assistenten: Juan Alvarado, Vincent Coppé, Roel Matthysen, Tim Van hamme, Fan Yang.
Student-assistenten: Jasper Deflander, Sam Landuydt, Jeroen Tempels.

1.3 Structuur van het practicum gedurende het semester

Het practicum bestaat uit één begeleide sessie per week. **Aanwezigheid op deze sessies is verplicht.** Indien je wegens overmacht of ziekte niet aanwezig kan zijn, meld je dit aan je groepscoördinator, je begeleiders en prof. Blockeel (Hendrik.Blockeel@kuleuven.be). Bij ziekte breng je achteraf een ziektebriefje binnen.

Naast het werk dat verricht wordt in de sessies wordt er ook verwacht dat je buiten de sessies werkt aan de taken die aan jou zijn toevertrouwd. Op basis van de 9 studiepunten worden er iets meer dan 6 uur extra werk per week per student buiten de sessie verwacht (in totaal meer dan 11 uur per week). Deze tijd moet opgemeten worden.

1.4 Teamwerking

Samenstelling team

Elk team is samengesteld uit vijf of zes studenten. Deze samenstelling wordt door de docenten vastgelegd aan het begin van het semester en wordt tijdens de eerste sessie meegedeeld.[1] In elk team wordt er een "CEO" en een "CAO" aangesteld, zie hieronder, die in nauwe samenwerking het werk van het team coördineren. De namen van de CEO en CAO worden doorgegeven aan de begeleidende assistent van je team op het einde van de eerste zitting.

Subteams

Voor de uitvoering van het gehele project kunnen er subteams gevormd worden, die elk een onderdeel van het project zullen uitwerken. De taakverdeling kan, in onderling overleg, vrij gekozen worden binnen het team. De subteams, hun samenstelling en hun taken moeten mee beschreven worden in het eindverslag.

Chief Executive Officer (CEO)

Gedurende de eerste sessie duidt elk team in onderling overleg een CEO aan. De functies van de CEO (naast het eigen werk in het team) zijn de volgende:

- De CEO is de voornaamste contactpersoon tussen het team en het didactisch team (professoren en assistenten).
- De CEO brengt het didactisch team op de hoogte indien er zich problemen voordoen binnen het team die een invloed kunnen hebben op de goede teamwerking. Bv. langdurige ziekte van een medestudent, regelmatige afwezigheden, conflicten die een vlotte samenwerking in de weg staan.
- De CEO is ook verantwoordelijk voor het *coördineren* van de verschillende subteams en zorgt ervoor dat iedereen zich houdt aan de opgelegde planning zodat het project binnen de opgelegde tijd beëindigd wordt.
- De taken van de CEO dienen mee in beschouwing genomen te worden bij het verdelen van het feitelijke projectwerk in de subteams.

[1] Eventueel kunnen, met onderlinge toestemming, studenten van team wisselen. Dit kan echter enkel gebeuren tijdens de eerste sessie van het semester en dient persoonlijk gemeld te worden aan prof. Blockeel.

Chief Administrative Officer (CAO)

Gedurende de eerste sessie duidt elk team in onderling overleg een CAO aan. De functies van de CAO (naast het eigen werk in het team) zijn de volgende:

- De CAO is verantwoordelijk voor het gestructureerd bijhouden van alle documenten en verslagen m.b.t. het project, en dit zowel in hardcopy formaat als elektronisch.
- De CAO (samen met de CEO) zorgt er voor dat er steeds een up-to-date versie is van de planning en het verslag. Zie ook §**??** voor meer informatie betreffende het verslag. De CAO is ook de eerste verantwoordelijke om de gevraagde verslagen samen te stellen en in te dienen. Dit wil niet zeggen dat de CAO de inhoud van alle verslagen zelf schrijft, maar wel dat hij zorgt dat elk verslag een coherent geheel vormt.
- De CAO houdt tevens een overzicht bij van hoeveel tijd (in uren) elk lid van de groep bijgedragen heeft tot het project. Dit overzicht dient om de bijdrage van elk teamlid individueel bij te houden, en tevens om de totale werkhoeveelheid gepresteerd door het volledige team te meten.
- De taken van de CAO dienen mee in beschouwing genomen te worden voor het verdelen van het feitelijke projectwerk in de subteams.

1.5 Planning

Er wordt verwacht dat er steeds een algemeen planningsdocument is dat de status en vooruitgang van het project laat zien. Dergelijke planning is voornamelijk een tool om bottlenecks en eventuele problemen, die een negatieve invloed kunnen hebben op het tijdig voltooien van het project, te detecteren.

In principe is het gebruik van planningstools reeds in vorige P&O's aan bod gekomen: taakstructuur & verantwoordelijkheidsstructuur, Gantt-chart, teamkalender, enz. Maak gebruik van je ervaring uit vorige P&O's om een functionele en nuttige planning uit te werken.

Zoals reeds vermeld, houdt de CAO het planningsdocument up-to-date, de CEO zorgt ervoor dat iedereen zich aan de planning houdt en lost problemen op waar nodig.

Het kan tevens nuttig zijn om binnen een enkele sessie een goede werkmethode te hebben:

- Zit samen met de ganse groep aan het begin van de sessie om te overlopen wat iedereen die week plant te doen.
- Op het einde van elke sessie kan je opniew samenzitten en kan de CAO volgende puntjes overlopen en registreren:
  - Wie heeft deze sessie waaraan gewerkt?
  - Welke resultaten zijn er deze sessie bereikt?
  - Wat wordt er gepland in de loop van de week?
  - Welke zijn de belangrijkste problemen en aandachtspunten?

1.6 Begeleiding

Begeleiders

Elk team beschikt in principe over twee vaste begeleiders van het didactisch team. De begeleiders volgen het project op en gaan na of de gemaakte vorderingen het toelaten om het project binnen de tijdspanne van een semester klaar te krijgen. Tijdens elke begeleide sessie zal er een begeleider langskomen om de vooruitgang te bespreken. De begeleiders dienen voornamelijk om vragen aan

te stellen m.b.t. de planning, uitvoering en praktische problemen gerelateerd aan het project. Ze zullen het project niet voor jou maken.

Hoe problemen bespreken?

- Voor kleine problemen die zich tijdens een sessie zelf voordoen, kan je rechtstreeks één van de aanwezige begeleiders aanspreken.
- Voor problemen die betrekking hebben op de inhoud of planning van het project zelf, spreek je best je eigen vaste begeleider aan. Indien hij niet in de sessie zelf aanwezig is, aarzel dan niet om hem telefonisch of per email te contacteren.
- Belangrijke en grotere problemen meld je best voor de sessie (per email) aan je begeleider, zodat hij ook tijd heeft om eventueel naar een oplossing te zoeken. Je kan niet verwachten dat uitgebreide problemen ogenblikkelijk ter plaatste kunnen opgelost worden.
- Wees je ervan bewust dat problemen oplossen een inherent deel vormt van dit vak, cfr. de titel van het vak. Je zal dus geconfronteerd worden met tijdsverlies tengevolge van onverwachte problemen.

Wat kan je niet verwachten?

Je bent in eerste instantie zelf verantwoordelijk voor het beheersen en oordeelkundig gebruik van verschillende software-tools die in het project zullen gebruikt worden. Je mag niet verwachten dat de begeleiders een technische helpdesk vormen die allerlei kleine probleempjes oplossen. We verwachten dat je team eerst een serieuze poging onderneemt om dit soort problemen weg te werken.

De begeleider zal ook geen rol spelen in het maken van keuzes; elke ontwerpkeuze moet gewikt en gewogen worden door het team zelf. Discussieer en overtuig met argumenten en onweerlegbare testresultaten wat de beste ontwerpkeuze is. De begeleider zal vaak zeer kritische vragen stellen om te toetsen of de keuze inderdaad gefundeerd is.

1.7 Leerlijn Engels (enkel voor Bachelor Informatica)

De Faculteit Wetenschappen heeft doorheen de opleiding een “leerlijn Engels” uitgewerkt. Dit houdt in dat in verschillende vakken het gebruik van het Engels ingeoefend moet worden. Om die reden vragen we dat sommige elementen van het project in het Engels uitgevoerd worden.

2 Opgave

Een automatische piloot met virtuele testomgeving voor drones

Voor de opgave voor dit academiejaar beschouwen we het probleem van het bouwen van een Drone Autopilot, een systeem dat een drone (en meer bepaald een quadcopter) aanstuurt om allerlei taken (bv. het bezorgen van pakjes of het winnen van een drone race) te vervullen.

In eerste instantie beschouwen we meer bepaald het probleem van het aansturen van een indoor quadcoptertje dat binnen een opstelling in een lokaal van een beginpositie naar een eindpositie moet bewegen. Het te bouwen systeem moet hiervoor dus bewegings-opdrachten voor de drone-hardware (bv. "volle kracht omhoog" of "zwenk naar rechts halve snelheid") genereren. Een moeilijkheid hierbij is dat de reactie van de drone op dergelijke inputs niet precies voorspelbaar is. Daarom is het noodzakelijk een controlelus in te voeren waarbij op elk ogenblik de positie en oriëntatie van de drone gemeten wordt en op basis daarvan de nieuwe inputs voor de drone bepaald worden. Hiertoe moet dus de positie en de oriëntatie van de drone gemeten kunnen worden. Voor drones die over grote afstanden opereren kan een GPS-ontvanger hiervoor voldoende zijn; voor onze setting zijn echter niet meteen gelijkaardige sensoren voorhanden. In deze opgave zullen we de plaatsbepaling van de drone dan ook doen aan de hand van camerabeelden. Meer bepaald zullen we twee camera's vast bevestigen op de drone, op een zekere afstand van elkaar, beide voorwaarts georiënteerd; twee ogen dus. Deze laten via dieptezicht toe de positie van de drone te bepalen ten opzichte van voorwerpen die in de camerabeelden herkend worden. Het te bouwen systeem krijgt de camerabeelden als invoer, moet daarop de opstelling herkennen en de positie van de drone ten opzichte van de opstelling berekenen, en op basis daarvan nieuwe bewegings-opdrachten voor de drone genereren.

De opgave behelst uiteraard ook dat het gebouwde systeem getest en gedemonstreerd wordt. Echter, op dit moment zijn programmatorisch aanstuurbare drones waarop meerdere camera's gemonteerd kunnen worden nog niet gemakkelijk beschikbaar. Daarom gaan we dit academiejaar een **puur software-systeem** zonder enige hardware-interactie bouwen en behelst de opgave, naast het bouwen van een Drone Autopilot, het bouwen van een Drone Autopilot Virtual Testbed. Dit laatste is een softwaresysteem dat een fysieke opstelling met een drone en camera's **simuleert**. In het bijzonder neemt het als invoer de bewegings-opdrachten gegenereerd door de Drone Autopilot, berekent het de nieuwe positie van de drone, en genereert het camerabeelden voor consumptie door de Drone Autopilot. Het systeem moet ook een GUI ter beschikking stellen van de gebruiker zodat die de voortgang van de simulatie kan volgen en allerlei informatie (positie en snelheid van de drone, beelden van de opstelling uit willekeurige standpunten) kan opvragen. Ook de Drone Autopilot moet een GUI ter beschikking stellen van de gebruiker, zodat de gebruiker de uit te voeren taak kan opgeven (bv. de coördinaten van de bestemming) en om de gebruiker op de hoogte te houden van de voortgang (in het bijzonder de voltooiingsgraad) van de taak.

Elk team zal een Drone Autopilot en een Virtual Testbed bouwen. We leggen hier de API tussen deze twee stukken software vast, zodat de Autopilot van het ene team getest kan worden op het Testbed van het andere team. We beschrijven de API hier onder de aanname dat beide systemen zullen geïmplementeerd worden als Java-modules die zullen draaien in eenzelfde Java virtual machine-proces, waarbij dus de Autopilot gerealiseerd wordt als een Java-bibliotheek, en het Virtual Testbed als een Java-programma dat gebruik maakt van deze bibliotheek. Beide systemen mogen echter in willekeurige programmeertalen en -platformen gerealiseerd worden,

en hoeven niet in hetzelfde proces te draaien. Een vertaling van de onderstaande API naar een wire protocol dat communicatie toelaat tussen Autopilot en Testbed via bv. sockets is beschikbaar op Toledo, alsook Java-modules genaamd adaptors die toelaten een Autopilot die geschreven is in Java en gebruikmaakt van de onderstaande Java-API te draaien op een Testbed dat het wire protocol implementeert, en andersom.

```
package p_en_o_cw_2016;

/** A camera that can be thought of as a pinhole camera with a flat sensor. Each
    pixel is the same size (in meters), so they are not the same angular size when
    perceived from the pinhole. */
public interface Camera {
    /** Angle of view in degrees. */
    float getHorizontalAngleOfView();
    /** Angle of view in degrees. */
    float getVerticalAngleOfView();
    /** The width in pixels. */
    int getWidth();
    /** Returns an array of width x height elements. The element at index r * width + c
        represents the pixel at row r and column c, where row 0 is the topmost row
        (bottommost on the sensor) and column 0 is the leftmost column (rightmost on
        the sensor). A pixel with value (R,G,B) is encoded as R | G << 8 | B << 16. */
    int[] takeImage();
}

public interface Drone {
    /** The distance between the cameras, in meters. Note: the (pinholes of the)
        cameras are on the pitch axis, so pitching rotates the cameras but does not
        translate them. */
    float getCameraSeparation();
    Camera getLeftCamera();
    Camera getRightCamera();
    /** The weight in kg. */
    float getWeight();
    /** The gravity in newtons. */
    float getGravity();
    /** The drag force per unit of speed, in kilograms per second. This drone is highly
        aerodynamic, so the drag force is linear in the speed. */
    float getDrag();
    /** The current pitch (the angle between the direction of view and the horizontal
        plane), in degrees. Positive pitch means the drone is looking down. */
    float getPitch();
    /** The current roll (the angle between the line that connects the cameras and the
        horizontal plane), in degrees. Positive roll means the drone is banking to the right. */
    float getRoll();
    /** Gets the maximum of the absolute value of the thrust, in newtons. */
    float getMaxThrust();
    /** Sets the thrust (the force exerted by the propellors) in newtons. This force is
        perpendicular to the plane defined by the direction of view and the line connecting
```

```
        the cameras. Positive means upward. */
    void setThrust(float thrust);
    float getMaxPitchRate();
    /** Sets the pitch rate, in degrees per second. */
    void setPitchRate(float rate);
    float getMaxRollRate();
    /** Sets the roll rate, in degrees per second. */
    void setRollRate(float rate);
    float getMaxYawRate();
    /** Sets the yaw rate (rate at which the drone rotates around the vertical axis (parallel to
        gravity) through the center between the cameras), in degrees per second. Positive
        means the drone turns to the right. */
    void setYawRate(float rate);
    /** The amount of simulated time that has passed since the start of the simulation,
        in seconds. */
    float getCurrentTime();
}

public interface Autopilot {
    /** Called by the testbed in the AWT/Swing GUI thread at a high (but possibly variable)
        frequency, after simulated time has advanced and the simulated world has been
        updated to a new state. Simulated time is frozen for the duration of this call. */
    void timeHasPassed();
}

public interface AutopilotFactory {
    /** Called by the testbed in the AWT/Swing GUI thread to create and start an Autopilot.
        At this point, the drone exists in the virtual world, with zero pitch and roll. The
        Autopilot can request initial camera images and set initial values of thrust and
        rotation rates. It can also set up a Swing GUI. Simulated time is frozen for the
        duration of this method call; simulated time starts running after this call returns.
        The Drone and Camera objects are not thread-safe; calls of methods of these
        objects should occur only in the AWT/Swing GUI thread. */
    Autopilot create(Drone drone);
}
```

De belangrijkste technische uitdagingen zijn dus:

- In de Drone Autopilot
  - Positiebepaling door middel van beeldherkenning
  - Planning om de hoog-niveau taak om te zetten naar bewegings-opdrachten
- In de Virtual Testbed
  - Physics simulation om uit de bewegings-opdrachten de positie en snelheid van de drone in functie van de tijd te berekenen
  - Collision detection om te bepalen of de drone een obstakel raakt
  - Het genereren van 3D-beelden

De moeilijkheidsgraad van die uitdagingen hangt sterk af van de volgende factoren:

- De mate van realisme van de gehanteerde drone-fysica. Bv.: houden we rekening met mogelijk tocht in het lokaal? Houden we rekening met mogelijke ongewenste rotatie van de drone?
- De complexiteit van de opstelling: beweegt de drone in de vrije ruimte of in een complexe omgeving met complexe, eventueel bewegende, obstakels?
- De mate van realisme van de te genereren en te verwerken camerabeelden: Is de belichting realistisch of vlak? Kan er overbelichting, onderbelichting, schaduw optreden? Kan er ruis op de camera zijn?
- De complexiteit van de uit te voeren taak. Moet de drone naar een opgegeven positie bewegen, of moet hij racen met andere drones doorheen een ingewikkeld 3D-parcours dat hij zelf moet leren kennen uit de camerabeelden?

Het is de bedoeling dat het werk dat binnen deze opgave gebeurt, een basis legt voor een Drone Autopilot die uiteindelijk met echte drones complexe taken kan uitvoeren. De principes van risicobeheersing vereisen weliswaar dat we dit einddoel iteratief en incrementeel benaderen door middel van een opeenvolging van haalbare mijlpalen, zodat we telkens na beperkte tijd een tussenresultaat bekomen dat we kunnen testen en aan de hand waarvan we ervaring kunnen opdoen met de materie.

De te realiseren mijlpalen zijn als volgt. Voor elke mijlpaal beschrijven we de uitbreiding/aanpassing ten opzichte van de vorige mijlpaal.

Te realiseren tegen half november

Mijlpaal 1.1

- Opstelling: vrije ruimte; witte achtergrond. Een kleine rode bal geeft de doelpositie aan.
- Belichting: uniform, mat
- Taak: beweeg naar de rode bal en blijf daar hangen.
- Drone physics: de drone voert de opgegeven pitch, roll, en yaw rates onmiddellijk uit. Er is dus geen rotationele inertie. Ook thrust settings nemen onmiddellijk effect.
- Demonstreerbaar systeem (autopilot + testbed, elk met GUI). Zie hierboven voor welke informatie en functionaliteit elke GUI moet aanbieden.

Mijlpaal 1.2

- Drone physics: de drone is voortdurend onderhevig aan een kleine (wind)kracht waarvan de grootte en de richting willekeurig varieert in de tijd. Ook de oriëntatie van de drone verandert voortdurend lichtjes. Bijgevolg zal hij zonder bijsturing steeds verder afwijken van het bedoelde pad. De autopilot moet dus voortdurend de positie van de drone bepalen en bijgewerkte bewegings-opdrachten doorgeven.

Mogelijke verdere mijlpalen (tentatief)

Mijlpaal 1.3

- Er hangen N ballen van verschillende kleuren in de ruimte. Ze zijn allemaal zichtbaar en niet-overlappend in de initiële camerabeelden. De drone moet de ballen zo snel mogelijk doorprikken (bij aanraking verdwijnt de bal), in willekeurige volgorde.

Mijlpaal 1.4

- De ballen kunnen zich achter, onder, boven de drone bevinden, kunnen verborgen zitten achter andere ballen, maar zijn niet zo ver weg dat ze niet zichtbaar zijn.

Mijlpaal 2.1

- Opstelling: een zeer dunne zwarte muur met een gat erin. De drone moet door het gat vliegen; dan zal hij de rode bal zien die de doelpositie aangeeft.

Mijlpaal 2.2

- Opstelling: een 3D-doolhof. Een bewegende rode bal leidt de drone erdoor. De rode bal zal voldoende traag vliegen dat de drone, door de bal te volgen, niet tegen de zijwanden zal vliegen.

Mijlpaal 2.3

- Opstelling: een 3D-doolhof. De drone bevindt zich in het inwendige van een veelvlak waarvan elk zijvlak een driehoek is met een andere kleur. Elk hoekpunt wordt dus volledig bepaald door de kleuren van de aanliggende zijvlakken. Dit laat toe eenzelfde hoekpunt terug te vinden op de twee camerabeelden. De drone moet de uitgang (ontbrekend zijvlak) vinden. Aanraking met de wand leidt tot diskwalificatie.

Mijlpaal 2.4

- Opstelling: een 3D-circuit. De drone moet zo snel mogelijk N rondjes voltooien. De start/finish is herkenbaar (wit zijvlak). De eerste keer moet hij de weg zoeken; door de weg te onthouden gaat het daarna sneller.

Mijlpaal 2.5

- Multi-drone race. Drones zijn onzichtbaar en zijn immaterieel; ze kunnen dus door elkaar heen vliegen. (Deze mijlpaal vereist dus enkel uitbreiding van het testbed.)

Mijlpaal 2.6

- Drones zijn zichtbaar als ballen, waarvan de kleur verschilt van de kleuren van de wanden en de andere drones. (Drones hebben een volledig gesatureerde kleur; wanden niet.) Een andere drone aanraken leidt tot diskwalificatie van beide drones.

Mijlpaal 2.7

- Deathmatch. Drones kunnen door ballen vliegen die een tijdelijk schild doen ontstaan waardoor aanraking met een andere drone enkel fataal is voor de andere drone.

Uitwerking

- Zoals gezegd moet elk team voor elk van beide systemen een GUI voorzien die toelaat alle relevante toestandsvariabelen van het systeem te inspecteren.
- De Testbed GUI laat toe een of meerdere virtuele camera's op te stellen in de opstelling en daarvan de beelden te bekijken.

Mogelijke uitbreidingen (tentatief)

- Uitbreiding A.1: De Autopilot GUI laat toe manuele controle te verkrijgen over de drone.
- Uitbreiding A.2: De Autopilot GUI laat toe de drone te bedienen door middel van het touch screen en de accelerometer en de gyroscoop van een smartphone. (Dus: je kan de drone doen pitchen door de smartphone vooruit te kantelen.) Voor de latere mijlpalen zijn doolhoven/circuits nodig, ook "level maps" genoemd in de game-wereld.
  - Uitbreiding B.1: Ontwikkel een GUI om 2D level maps (level maps die volledig gekarakteriseerd worden door een bovenaanzicht) te ontwikkelen.
  - Uitbreiding B.2: Breid de level map editor uit om gemakkelijk willekeurige 3D level maps te ontwikkelen.
  - Uitbreiding B.3: Ontwikkel een algoritme om automatisch bruikbare level maps te genereren.
  - Uitbreiding B.4: Importeer 3D-modellen van steden uit Google Maps en pas aan om te kunnen dienen als doolhof of circuit.
- Uitbreiding D: Belichting: de virtuele beelden kunnen realistischer gemaakt worden door belichting toe te laten, zonder de beeldherkenning te bemoeilijken, door ervoor te zorgen dat de belichting enkel de value (V in de HSV-kleurenruimte) beïnvloedt en alle nuttige informatie in H en S zit.
- Uitbreiding E: Textures: je kan de level maps een interessant uitzicht geven door textures te voorzien. Om te voorkomen dat dit de beeldherkenning bemoeilijkt: reserveer een deel van de HSV-kleurenruimte (nl. saturation < 0.75) voor textures en de rest (saturation > 0.75) voor de randen van de driehoeken. De beeldherkenning gebruikt immers enkel de randen. (De Testbed kan eventueel adaptief dikkere en dunnere randen tekenen naarmate de drone verder weg of dichter bij de driehoek is, zodat de randen altijd goed zichtbaar zijn voor de drone.)

3 Fasering, demo's en verslag

3.1 Fasering gedurende het semester

Om de uitwerking van het project te faseren, wordt er verwacht dat je verschillende kleine doelstellingen realiseert en een tussentijdse demonstratie geeft aan het didactisch team vooraleer de eindpresentatie plaatsvindt, zie §**??** voor details. Op deze wijze leer je naar deadlines toe te werken en kan je ook je vooruitgang vergelijken met de andere teams.

De week vóór de tussentijdse en finale demonstratie dien je een verslag in. Dit verslag informeert het didactisch team over de achtergrond, de verantwoording van de gemaakte keuzes, de wetenschappelijke onderbouw, de details over softwaredelen, enz. Het eindverslag bevat naast dit wetenschappelijk gedeelte ook een administratief gedeelte met tabellen over de werkverdeling, het aantal verrichte uren, etc.

Om de opbouw van een groter verslag gradueel te laten verlopen, worden er in het eerste deel van het semester kortere schrijfopdrachten uitgevoerd. Je verslag bevat ook enkele screenshots.

Elk team komt individueel zijn project voorstellen op de tussentijdse demo en finale presentatie. Bij de tussentijdse demo ligt de nadruk op de demonstratie zelf, met een beknopte uitleg voor en tijdens de demonstratie. Bij de eindpresentatie ligt de nadruk op de presentatie (een uitgebreid mondeling verslag over de werkzaamheden tijdens het semester en de behaalde resultaten), en dient de demonstratie vooral als bevestiging dat de doelstellingen bereikt werden. Na het demonstreren van de gevraagde functionaliteit worden er een aantal vragen gesteld worden door het didactisch team.

3.2 Algemene planning

Meer details over de exacte inhoud van de demonstraties en schrijfopdrachten vind je in volgende secties. Hieronder een globaal, per week, georganiseerd schema.

- 3/10: Openingssessie
- 10/10: Sessie
  11/10 (Di): Inleveren: max. 1 pagina planning/ontwerp/oplossingssuggesties
- 17/10: Sessie
- 24/10: Sessie
- 31/10: Sessie
- 07/11: Sessie
  09/11 (Wo): Inleveren tussentijds verslag, max. 10 pagina's
- 14/11: Tussentijdse demonstratie
- 21/11: Sessie
- 28/11: Sessie
- 05/12: Sessie
- 12/12: Sessie
  14/12: Finaal wetenschappelijk verslag
- 19/12: Finale demo en presentatie (ganse dag)

3.3 Schrijfopdrachten en verslagen

Algemene richtlijnen

Een goed verslag maakt het de lezer zo eenvoudig mogelijk. Breng een klare boodschap en schrijf bondig. “Leid” de lezer door je tekst: gebruik inleidende zinnen en verwijs nadien. Definieer begrippen en laat de lezer niet met vragen zitten. Houd figuren en tabellen leesbaar, benoem assen en gebruik eenheden. Voorzie figuren van een verklarend onderschrift en refereer ernaar in je tekst. Vergeet ook de appendices niet van de nodige uitleg te voorzien.

Voor elke demonstratie wordt een verslag verwacht. Deze verslagen worden opgemaakt in LaTeX, vertrekkende van een template die op Toledo zal worden geplaatst, en ze worden ingediend via Toledo. Voor het verslag bij de tussentijdse demonstratie mogen jullie **maximaal 10 bladzijden** indienen, voor de finale demo **maximaal 20 bladzijden**. Op het eerste verslag krijgen jullie feedback van het didactisch team.

Op het einde van het semester wordt een eindverslag verwacht. Dit verslag omvat grofweg twee grote delen. Het eerste luik licht op een *wetenschappelijke, onderzoeksgebaseerde en duidelijk gefundeerde wijze* de gemaakte keuzes en het project toe. Bronvermelding is hierin ook essentieel.

> *Wanneer tekst en/of code gebruikt wordt die overgenomen is van, of sterk geïnspireerd op, ander werk, moet dit duidelijk aangegeven worden in het rapport. Het nalaten van dergelijke vermelding kan als plagiaat beschouwd worden, en tot ernstige sancties leiden, overeenkomstig het examenreglement.*

Het tweede luik bevat een meer ervaringsgerichte bespreking van het verloop van het project, dit deel maakt op zich geen deel uit van de wetenschappelijke bespreking van het project. Dit tweede deel bevat een beschrijving van het proces, de werkverdeling, een kritische analyse, enz.

Het is niet de bedoeling dat dit verslag pas op het einde gemaakt wordt, maar wel dat je het opbouwt aan de hand van de schrijfopdrachten en het tussentijds verslag.

Toelichting bij de verschillende in te dienen teksten

De doelstelling van de eerste, kleine schrijfopdracht is om snel heel gerichte en duidelijke feedback te verkrijgen. Het beperkt aantal pagina’s zorgt ervoor dat je relatief veel schrijftijd hebt. Het is sterk aanbevolen dat elk groepslid zorgvuldig en uiterst kritisch de in te dienen tekst naleest, rekening houdend met de opmerkingen uit vorige sectie.

- Dinsdag 11/10, vóór 24u wordt max. 2 pagina’s verwacht. Beschrijf op een zeer hoog niveau wat je gaat realiseren, hoe je ontwerp er verwacht wordt uit te zien, een ruwe schets van de planning en geef de verantwoordelijkheden die daar aan gekoppeld zijn weer. Al het belangrijke van jullie eerste brainstormsessie zou hierin vermeld moeten zijn. De tekst wordt geacht op één bladzijde te passen, en bijhorend wordt er een schets/tabel/figuur verwacht die duiding geeft bij een bepaald aspect uit deze tekst.
- Woensdag 9/11, vóór 24u wordt het eerste tussentijds verslag max. 10 pagina’s ingediend. Het tussentijds verslag stemt overeen met het finale verslag; alleen zijn bepaalde secties nog niet geschreven (zie template). Voor stukken software die nog niet in detail uitgewerkt zijn wordt wel al een schets van de structuur verwacht, bv. in de vorm van klassediagrammen.

3.4 Overzicht demonstraties

Kleinere demonstraties voor de begeleiders

Wekelijks wordt elk team geacht om tijdens de sessies op maandag een aantal zaken te demonstreren aan het begeleidersteam. De volgorde van realisatie en de tijdstip waarop dit getoond wordt is van geen belang. Echter, de begeleiders zullen de realisatie beoordelen en hun bevindingen meedelen; deze kritiek kan je dan meenemen in je ontwerp en aanpassingen voor de tussentijdse demonstratie. Een voorbeeld:

- Sessie 2: Aansturen van de drone
- Sessie 3: Basis-ontwerp GUI
- Sessie 4: De simulator
- ...

Tussentijdse demo

Tijdens de tussentijdse demo moet het team demonstreren dat **Mijlpalen 1.1 en 1.2** behaald zijn.

Voorzie dat iemand van je team tijdens de demo kan uitleggen wat er gebeurt, zodat het didactisch team daar niet naar moet vragen. Verwacht je ook aan enkele vragen over de werking van je software en de keuzes die jullie gemaakt hebben.

Deze demo gaat in het Engels door.

3.5 Finale presentatie

Op de dag van de finale presentaties geeft elke groep een presentatie van het project: een voordracht van 10 minuten, gevolgd door een demo van 5 minuten. Daarop volgen ongeveer 10 minuten vragen en discussie. Het hele team is op deze presentatie aanwezig.

De presentatie bevat minstens volgende onderdelen:

- beschrijving van de software;
- gebruikte algoritmes en oplossingen;
- kort overzicht van behaalde resultaten;
- werken in groep: ervaringen, moeilijkheden;
- eigen conclusies: sterktes en zwaktes van het verwezenlijkte project.

De informatie op de slides mag beknopt zijn, voor details kan je verwijzen naar het verslag. De bedoeling is om zoveel mogelijk dubbel werk tussen presentatie en verslag te vermijden. Het didactisch team heeft zowel het verslag gelezen als de demonstraties gezien: creëer een meerwaarde.

3.6 Peer-assessment

Op het einde van het semester zal ook via peer-assessment gevraagd worden naar ieders evaluatie m.b.t. de andere groepsleden. Deze peer-assessment maakt geen deel uit van het eindverslag, maar zal afzonderlijk via Toledo opgevraagd worden.

3.7 Evaluatie

De evaluatie van dit vak zal gebeuren op basis van de volgende criteria:

- Teamwerk
- Kwaliteit en inhoud van tussentijdse verslagen en demonstraties.
- Kwaliteit en inhoud van de eindpresentatie en eindverslag.
- Functionaliteit en correctheid van de ontwikkelde applicatie.
- Kwaliteit van het ontwerp van de ontwikkelde software.
- Individuele evaluatie van elke student, op basis van prestaties gedurende het semester en peer-assessment. Dit impliceert dat elke student een individuele quotering krijgt.

Om jullie te motiveren om de deadlines voor de tussentijdse demo's te halen, wordt voorzien dat een team dat de doelstellingen van de demo niet haalt tot 2 strafpunten kan krijgen. Strafpunten worden op het einde afgetrokken van de totaalscore.

De teams worden als geheel beoordeeld op de globale kwaliteit van het verslag, de finale demo, en de presentatie. De teamleden worden individueel beoordeeld op de kwaliteit van hun eigen bijdrage in het project: de tekst (verslag) en programmacode die ze zelf geschreven hebben, hun inhoudelijke bijdrage, de mate waarin ze het team gesteund hebben. Dit wordt o.a. via peer-review beoordeeld.

4 Planning 28 september 2015

- 14u00-15u00: Introductie, Dirk Nuyens, lokaal 00.144
- 15u00-15u30: Verslagen schrijven, Tim Van hamme, lokaal 00.144
- 15u30-19u00 Afspraak met je team, Sol-N, Sol-Z, PC-klas.
  - Kennismaking Informatica- en CW-studenten
  - Kies CEO en CAO
  - Bediscussieer planning, enz.

5 Groepsverdeling

Dit is een voorlopige verdeling. Namen voor de puntkomma zijn studenten 3de bachelor informatica. Deze worden zo gelijk mogelijk over de groepen verspreid.

00.124 – Begeleiders: Tim Van hamme, Juan Alvarado

Indigo: Kristof Achten, Wouter Baert; Dieter Balis, Yves-Cédric Bauwelinckx, Rik Bauwens, Ranec Belpaire
Goud: Jonas Baes, Michiel Bollen; Tijs Bergmans, Tim Beyne, Marte Biesmans, Alexander Boucquey
Oranje: Sven Cuyt, Bauwen Demol; Bart Breuls, Ruben Cartuyvels, Bart Claessens, Joke Claeys
Brons: Yana Dimova, Thor Galle; Valentin Claeys, Ruben Coppens, Maarten Craeynest, Antoon De Cleen
Platinum: Bert Goemans, Tim Hofmans; Bram De Cooman, Antony De Jonckheere, Toon Deburchgrave, Lauren Devrieze
Groen: Robbe Keters, Aykut Koc; Philippe Dewart, Siebe Diels, Willem Engels, Zeno Gillis

Sol-N – Begeleiders: Vincent Coppé, Fan Yang

Paars: Bavo Luysterborg, Mathieu Monsecour; Moran Gybels, Stijn Hendrikx, Elisabeth Heremans, Jesse Hoobergs
Blauw: Nick Schouten, Jonas Soenen; Vincent Janssen, Simon Lauwers, Urbaan Lemmens, Bert Leroy
Rood: Sofie Theys, Minh Tri Tran; Jasper Mertens, Pieter Mertens, Joa Oris, Roman Peeters
Geel: Sebastiano Valente, Niels Van Belle; Maxime Pittomvils, Ruben Praets, Jeroen Robben, Joris Schrauwen
Wit: Matthias Van Ceulebroeck, Arne Van Den Kerchove; Thijs Steel, Andreas Stieglitz, Damian Szkuat, Victor Van Eetvelt

Sol-Z – Begeleiders: Roel Matthysen, X

Robijn: Rick van Hek, Mathias Van Herreweghe; Gilles Van Gestel, Florian Van Heghe, Eleanor Van Looy, Sebastiaan Van Overschee
Koper: Maaike Van Roy; Jules Van Reempts, Martijn Van Rooy, Tristan Van Thielen, Andreas Vande Voorde
IJzer: Nick Vandeborg; Thomas Vandendijk, Steven Vander Eeckt, Kasper Verhulst, Jan Vermaelen
Zilver: Bram Vandendriessche; Matthias Van der Heyden, Jef Versyck, Vincent Vliegen, Arne Vlietinck, Laura Vranken
Saffier: Kwinten Verhelst; Emile Valcke, Thomas Vranken, Timothy Werquin, Roeland Wyns, Kevin Zhang