

29-2-2024

Tupolev Gaming

Dogfight simulation



Figuur 1 A 2D python pygame display with two ww2 planes in it that are doing a dogfight in retro style (DALL-E 3, z.d.)

Bas de Blok, Finn de Graaf, Max Visscher & Joris
Heemskerk
V3A

Inhoudsopgave

1.	Keuze onderwerp.....	2
2.	Experimenten.....	2
3.	Werkwijze	3
3.1.	Projectmethodologie	3
3.2.	Werktijden.....	4
3.3.	Communicatie	4
3.4.	Code	4
4.	PEAS	4
5.	Toepassen van kennisrepresentatie & redenatie.....	5
5.1.	Software-architectuur	5
5.2.	Ontologie.....	5
5.3.	Gebruik van logica.....	5
5.4.	Omgang met onzekerheid	6
6.	Bronvermelding	6

1. Keuze onderwerp

Bij het brainstormen van de groepsopdracht kwamen we op de volgende drie ideeën:

- Zelfrijdende auto's simuleren
- Oorlog simuleren
- Dogfights simuleren

Eigenlijk waren we er al snel over uit dat we de dogfight simulatie toch wel het meest interessant vonden. Finn moest ook direct terugdenken aan spelletjes die hij vroeger op spele.nl (Azerion, z.d.) speelde (*Dogfight 2*, z.d.).

Het zou hierin gaan om een 2-dimensionaal speelveld, waarin we in eerst instantie één enkel vliegtuig agent willen simuleren. In een later stadium van het project zouden er meerdere agenten toegevoegd kunnen worden om er een multi-agent systeem van te maken. Voor een wat specifiekere invulling van wat voor gedrag een agent dan bijvoorbeeld kan vertonen hebben we ook een aantal opties uitgedacht:

- Een vliegtuig heeft een gelimiteerde perceptie van de omgeving, denk aan een cirkel om het vliegtuig heen van een gegeven radius en/of een driehoek vanuit een punt in de cockpit om het gezichtsveld van een piloot te representeren. Deze gelimiteerde perceptie betekent dat de agent een voordeel heeft aan het bishouden van een interne staat/representatie van de omgeving. Denk hierbij aan het onthouden waar de agent een target voor het laatst heeft gezien.
- Gegeven een herkenning van een bepaalde target kan een van de doelen van de agent zijn om zich zodanig te manoeuvreren door de omgeving, dat optimaal schietgedrag uitgevoerd kan worden. Denk hierbij, onder anderen, aan het houden van een bepaalde afstand, het inspelen op eventuele bewegingen van de target, rekening houden met het huidige brandstofniveau, bewust zijn van mogelijke andere vijandelijke targets die terug zouden kunnen schieten, en nog veel meer.
- De agent zou een gelimiteerd aantal kogels of brandstof kunnen hebben. In de omgeving zouden bijvoorbeeld brandstof of kogels in de lucht kunnen verschijnen zoals dit in videogames kan gebeuren. De agent kan in de omgeving de locatie van deze objecten leren onthouden en er beroep op doen op het moment dat de agent bijna geen brandstof of kogels meer heeft.
 - In plaats van deze voorwerpen in de lucht te laten verschijnen kan de omgeving ook vliegvelden bevatten waar een agent kan landen om brandstof en kogels aan te vullen.

2. Experimenten

De simulatie dient natuurlijk ook een doel te hebben; welk experiment proberen we uit te voeren of welk probleem proberen we op te lossen. Gezien onze simulatie, in ieder geval conceptueel, de werkelijke wereld nabootst zouden wij graag een onderzoek uit willen voeren naar optimaal gedrag van een piloot. Een agent die optimaal gedrag vertoont zou, gegeven de restricties van de omgeving, maximaal moeten overleven en maximaal tegenstanders moeten uitschakelen. We realiseren ons dat de simulatieomgeving zoals deze in dit document is omschreven veel abstracties met zich meebrengt. Dit betekent dat een conclusie over optimaal gedrag per noodzaak alleen in het referentiekader van onze simulatie getrokken kan worden. Wij voorzien hier dan ook een stapsgewijze benadering van de realiteit door middel van iteraties over de simulatieomgeving. Door de omgeving elke keer een stukje realistischer te maken komen we stapsgewijs dichter bij een concrete onderbouwing voor de conclusies over optimaal gedrag.

Om een concrete vergelijking te kunnen doen maken wij gebruik van David Solinger et al. (April 2005). Deze thesis is gemaakt met het doel om een agent te kunnen maken die een competitieve speler kan zijn in een 1 tegen 1 dogfight in Microsoft Combat Flight Simulator (MSCFS). MSCFS is een 3D flight simulator terwijl wij onze simulatie in 2D gaan doen, het doel van ons onderzoek zal dan ook zijn om te onderzoeken of we de resultaten van de agents uit de paper kunnen benaderen in onze 2D omgeving.

3. Werkwijze

3.1. Projectmethodologie

We hebben vaker in deze groepssamenstelling gewerkt, waardoor we onderling een goed beeld hebben van elkaars competenties en werkwijze. In voorgaande projecten hebben we maximaal resultaat kunnen leveren met behulp van een ScrumBan werkwijze. We hebben gedurende deze projecten echter ook ervaren dat het professioneel aanpakken van een ScrumBan werkwijze vrij veel *overhead* creëert. Het kost veel tijd om User Stories goed uit te werken, om een sprint te plannen en om Daily Standups, Sprint Reviews en Retrospectives te houden. Gezien dit project relatief klein is in verhouding tot voorgaande projecten willen we daarom ook onze werkwijze op bepaalde vlakken versimpelen. Zo willen we User Stories alleen op globaal niveau aanmaken. Taken mogen dus groter zijn dan traditioneel toegestaan in ScrumBan. Daarbij willen we deze taken ook niet onderverdelen in sub-taken, omdat dit relatief veel tijd kost en wij inmiddels het onderlinge vertrouwen hebben dat we zonder dit uit te voeren ook tot een valide resultaat kunnen komen. Standups willen we er wel in houden, tweemaalweeks, in de vorm van een fysieke bijeenkomst op de dinsdag, voorafgaand aan het ABD-college en online in Discord op de donderdag. De Sprint Reviews en Retrospectives vatten we samen in een korte bijeenkomst waarin we elkaar aangeven wat we wel en niet goed vonden gaan in het proces en wat we graag anders zouden willen doen. Deze bijeenkomst vindt plaats voorafgaand aan de Standup op de eerste dinsdag of donderdag vanaf de voorgaande deadline. Daarbij vervangen we de eerste Standup van een nieuwe Sprint door een korte Sprint Planning, waarvoor we gezamenlijk nieuwe User Stories aanmaken en deze prioriteren in de Sprint Backlog. De Sprint Review, Sprint Retrospective en Sprint Planning vinden dus opeenvolgend plaats op het eerst mogelijke moment vanaf de voorgaande deadline. Wij hebben er het vertrouwen in dat we op deze manier ons werk- en leerproces kunnen maximaliseren en tegelijkertijd de *overhead* te kunnen minimaliseren.

3.2. Werkijken

Om aan de opdrachten van ABD te werken hebben we als team een aantal vaste werktijden afgesproken.

- Dinsdag
 - 13:00 – 16:00 | ABD-les
- Donderdag
 - 09:00 – 13:00 | Werksessie gezamenlijk, om de week als het groepje geen gilde heeft
 - 13:00 – 16:00 | Werksessie gezamenlijk (Joris kan niet altijd)

Wij worden verwacht zo'n 12 uur in de week aan de kennisroute te besteden, door conflicterende roosters kunnen wij niet elke week 12 uur vinden waarin wij allemaal vrij zijn om aan ABD te werken. Buiten deze vaste werktijden om is het dus de verantwoordelijkheid van de individu om verder te werken aan zijn toegewezen taken die niet afgerond waren tijdens deze tijden.

3.3. Communicatie

De communicatie tussen groepsgenoten vindt volledig plaats via Discord, waar een chat is gestart voor alle groepsgenoten die specifiek is bedoeld voor overleg omtrent dit project.

3.4. Code

De code van de simulatie wordt allemaal opgeslagen in deze private-repository op GitHub:
<https://github.com/Finnyy012/TUPOLEVGAMING/>

4. PEAS

Agent type		Performance measure	Environment	Actuators		Sensors
Dogfight-vliegtuig		Maximaliseer aantal targets geraakt door agent, Maximaliseer tijd overleefd in de lucht.	De lucht, grond, balonnen	Schielen, inclinatie veranderen		Ogen piloot
Agent type	Fully observable vs. Partially observable	Single agent vs. Multiagent	Deterministic vs. stochastic	Episodic vs. sequential	Static vs. dynamic	Discrete vs. continuous
Dogfight-vliegtuig	Partially observable	Single agent	Stochastic	Sequential	Semidynamic	Continuous

Dogfight-vliegtuig is partially observable, omdat de agent niet het gehele luchtruim kan zien op een gegeven moment, inclusief alle overige informatie die zich hierin verborgt. Het is Single agent, omdat er maar 1 agent is in onze huidige simulatie. Het is stochastic aangezien er randomness geïntroduceerd wordt bij het inladen van de ballonnen. Het is sequential, omdat de agent afhankelijk is van voorgaande acties zoals onthouden waar fuel is. Het is semidynamic, omdat de environment niet veranderd, maar de agent en de ballonnen beïnvloeden wel de omgeving. Het is continuous, omdat er niet een eindig aantal staten is

5. Toepassen van kennisrepresentatie & redenatie

5.1. Software-architectuur

Bij Simulation Programming hebben we verschillende tools leren te gebruiken om een simulatie in te bouwen. Zo hebben we gewerkt met Mesa, Unity en NetLogo. Bij het Simulation Project hebben we geconcludeerd dat al deze tools voor ons eigenlijk niet prettig werkten. We hebben voor dat project toen ervoor gekozen om zonder framework aan de slag te gaan, wat ons wel beviel. Dit willen we dus ook zo gaan doen voor ABD. Wel willen we uiteraard gebruik maken van packages als Pygame voor de grafische elementen van de simulatie. Drie van de groepsgenoten hebben al eerder gewerkt met Pygame en hebben hier positieve ervaringen mee. Verder werken we exclusief in GitHub om onze code en projectmanagement zaken in bij te houden. Voor documenten gebruiken we een gedeelde OneDrive omgeving.

5.2. Ontologie

Singleagent:

- Niet tegen grond aan vliegen
- Niet tegen ballonnen aanvliegen
- Op stationair voorwerp richten
- Op stationair voorwerp schieten
- Refuelen wanneer nodig
- Herladen wanneer nodig

Multiagent:

- Richten bewegend voorwerp
- Schieten bewegend voorwerp
- Manoeuvres
 - Straight flight
 - Looping
 - Immelman
 - Split-S
 - Stall turn

5.3. Gebruik van logica

Ballon(x)	x is een ballon
InSight(x)	x is in het gezichtsveld van het de agent
IsKnown(x)	x is bekend bij de agent
Tick(x)	x is een tick
Load(x)	x wordt ingeladen
Exist(x)	x bestaat in de environment
Direction(x)	x is de richting waar het vliegtuig naartoe kijkt
Move(x)	x is een beweging

Alle ballonnen in het zicht van het vliegtuig zijn bekend bij dat vliegtuig.

- $\forall x(Ballon(x) \wedge InSight(x) \rightarrow IsKnown(x))$

Elke tick wordt er gekeken of er 10 ballonnen aanwezig zijn, zo niet worden er meer ingeladen.

- $\forall x(\text{Tick}(x) \rightarrow ((\exists y_1, \dots, y_{10} (\text{Ballon}(y_1) \wedge \dots \wedge \text{Ballon}(y_{10}) \wedge \text{Exist}(y_1) \wedge \dots \wedge \text{Exist}(y_{10}))) \vee (\forall z(\text{Ballon}(z) \wedge \neg \text{Exist}(z) \rightarrow \text{Load}(z))))$

Elke tick beweegt het vliegtuig in de richting waar hij naartoe kijkt.

- $\forall x(\text{Tick}(x) \rightarrow \text{Move}(x) = \text{Direction}(x))$

5.4. Omgang met onzekerheid

Op dit moment is de enige onzekerheid voor de agent waar de ballonnen zich bevinden. Wanneer hij in beweging is kan de agent wel nieuwe ballonnen vinden

6. Bronvermelding

DALL-E 3. (z.d.). <https://openai.com/dall-e-3>

Azerion. (z.d.). *Spelletjes - gratis spelletjes online spelen op Spele.nl*. Spele.nl. <https://spele.nl/>

Dogfight 2. (z.d.). Play Online On SilverGames 🛡️. <https://www.silergames.com/en/dogfight-2>

David Solinger et al. (April 2005). Creating a dogfight agent
http://www.kbs.twi.tudelft.nl/docs/MSc/2005/Solinger_David/thesis.pdf

Commented [BB1]: TODO:

Werk README bij:

opdracht

“Vermeld wel je bronnen in de readme van je Git repo.”

Inspiratie ander project:

<https://github.com/Casper-Smet/LOAN/blob/main/README.md>