# *Gemaakt door: Erik Koolen (2099418), Lennard Slabbekoorn (2098691)*

# A\*-algoritme

**Bepalen afstandsfunctie:**

Per knoop wordt bepaald welke afstand er al gelopen is vanuit vorige knopen Deze wordt gebruikt samen met de heuristiek om vast te stellen wat de route uiteindelijk moet zijn. De knopen die op basis van deze prioritering naar voren komen worden toegevoegd een queue. Die daarna weer gebruikt wordt om het kortste pad netjes te construeren in een nieuwe queue.

**Gekozen heuristiek:**

De gekozen heuristiek is de hemelsbrede, absolute afstand in pixels. Deze wordt opgeteld bij de prioriteit die ook wordt bepaald door kosten die er al zijn.

**Rekening houden met rotsen:**

Er wordt door het A\*-algoritme rekening gehouden met de kosten die gemaakt moeten worden om over een bepaalde zijde te gaan naar een volgende knoop. De zijden die zich bevinden tussen de rotsen hebben een veel hogere weight dan gewoon water. De weight van een zijde bepaald als een van de onderdelen (de andere is de totale kosten die er tot nu toe al zijn) hoe veel het kost om naar de knoop te gaan.

**Strategie van Knabbel en de Porcus Vincit:**

De strategie van hoe het A\*-star algoritme wordt gebruikt door Knabbel en de Porcus Vincit houden nauw samen met de eindigetoestandsautomaten. Knabbel zal alleen gebruik maken van het A\*-algoritme als hij in de ‘hunt’ toestand is en de Porcus Vincit als hij in de ‘repair’ toestand is. Meer over de toestanden van Knabbel is te vinden in de volgende paragraaf.

Eenmaal in de hunt-toestand vindt Knabbel o.a. de kortste route naar het dichtstbijzijnde varkentje met het A\*-algoritme in die toestand. Dat blijft hij doen tot dat de toestand over gaat.

De Porcus Vincit gebruikt het A\*-algoritme om de kortste route te vinden naar de geselecteerde haven als de boot gerepareerd moet worden. Dit betekend dus dat het grootste gedeelte van de tijd de Porcus Vincit dus niet het A\*-algoritme gebruikt.

# Eindigetoestandsautomaten

## Toestandsdiagrammen

**KNABBEL**

Within Porcus Vincit range

Hunt

Pig within range

100 steps limit

Within Porcus Vincit range

After 10 steps

At resting place

100 steps limit

100 steps limit

Scared

Tired

Wander

Arrived at pig location

**START**

**PORCUS VINCIT**

33% chance for every dock.  
After 10 repairs chance  
best/worst +/- 3%

**START**

Repaired

Paint damage >= 100

Repair

Wander

## Gedragsbeschrijving

**Knabbel:**-Wander toestand. Knabbel begint in de wander toestand. Hierin gaat hij steeds van een knoop naar een willekeurige andere knoop. Vanaf de wander toestand kan knabbel naar de tired toestand als hij 100 stappen heeft gezet (dit gebeurt via een globale toestand die de stappen bijhoud), naar de scared toestand als de Porcus Vincit binnen 50 pixels is of naar de hunt toestand als er een varkentje binnen 100 pixels is. In deze toestand kunnen varkentjes worden opgegeten als ze in contact komen met Knabbel.

-Hunt toestand. Knabbel komt altijd vanaf de wander toestand in de hunt toestand. Dit gebeurt als er een varkentje binnen 100 pixels is. In de hunt toestand gaat Knabbel naar de knoop die op dat moment het dichtst bij het varkentje is. Mocht de Porcus Vincit tijdens het jagen binnen 50 pixels komen dan gaat Knabbel over naar de scared state. Zo niet en Knabbel is op de knoop aangekomen dan gaat hij terug naar de wander state. Knabbel gaat naar de tired state(via de global state) als ondertussen totaal 100 stappen zijn gezet. In deze toestand kunnen varkentjes worden opgegeten als ze in contact komen met Knabbel.

-Scared toestand. Knabbel kan vanaf de hunt en de wander toestand in de scared toestand komen als de Porcus Vincit binnen 50 pixels is. In de scared toestand zet Knabbel 10 willekeurige stappen naar een andere knoop. Na die tien willekeurige stappen gaat Knabbel weer terug in de wander toestand. Knabbel gaat naar de tired state(via de global state) als ondertussen totaal 100 stappen zijn gezet.

-Tired toestand. Knabbel kan vanaf iedere toestand naar de tired toestand via de global state. In de tired toestand zoekt Knabbel de snelste weg naar zijn rustplaats. Eenmaal op de rust plek aangekomen gaat hij weer terug naar de wander toestand.

-Globale toestand. Deze toestand wordt iedere update call aangeroepen en komt daarom ook niet voor in het diagram. Knabbel kan namelijk niet in de global state zitten, die is er gewoon altijd. De global state houdt bij hoeveel stappen er gezet zijn en zet Knabbel in de tired state als dat er 100 zijn.

**Porcus Vincit**

-Wander toestand. De Porcus Vincit begint in de wander toestand. Hier gaat hij van een knoop naar een willekeurige andere knoop. Vanaf de wander toestand kan de Porcus Vincit in de repair toestand komen als de lakschade 100 is.

-Repair state. De Porcus Vincit komt in de repair toestand als de lakschade 100 is. In de repair toestand selecteert de boot eerst 1 van de 3 havens. Eenmaal gekozen gaat hij daar met de kortste weg naar toe. In de haven wordt de boot voor een bepaalde waarde gerepareerd waarna de Porcus Vincit weer in de wander toestand komt.

-Globale toestand. Ook de Porcus Vincit heeft een globale toestand. In deze globale toestand wordt bijgehouden of de lakschade 100 is. Zo ja dan gaat de Porcus Vincit in de repair toestand.

## Aanpassingsstrategieën transitiekansen

Naar welke haven de Porcus Vincit heen gaat voor reparatie hangt af van kansen. De havens beginnen allemaal met 33% kans. Na 10 reparaties worden de reparatie aantallen geëvalueerd. De haven met de kleinste reparatie waarde krijgt -3% kans tot een minimum van 15%. De haven met de grootste reparatie waarde krijgt +3% kans tot een maximum van 69%. Let op, dit kan dezelfde haven betreffen. In dat geval veranderd er dus niks aan de kansen. Een tweede opmerking is dat de kansen bij elkaar 99 zijn en niet 100. Er wordt een willekeurig getal gekozen uit 99. Dus effectief hoort achter ieder genoemde kans .33%

# Force-driven entities

**Verantwoording van gebruikte krachten:**

* *Wander:* De wander-kracht wordt gebruikt om de varkentjes min of meer willekeurig te laten bewegen over de kaart. Wander is echt een basisonderdeel voor het laten bewegen van de varkentjes. Wander werkt aan de hand van een ‘wander-target’ die zich voor het varkentje onzichtbaar min of meer random beweegt. Het varkentje zal proberen te sturen naar deze wander-target.
* *Wall-avoidance:* De wall-avoidance-kracht wordt gebruikt om de eilanden in de hoeken van de kaart te ontwijken en om niet van de kaart af te bewegen. De kracht begint pas te werken als een varkentje in de buurt van zo’n onzichtbare muur komt en begint dan als het ware terug te duwen zodat het varkentje afremd en een andere kant op gaat bewegen.

De detectie van een muur zit op 3 vectoren in het varkentje zelf. Eentje recht vooruit en 2 aan op een hoek van 45 graden daarvan.

* *Seek:* de seek-kracht wordt gebruikt om de varkentjes aan te trekken richting de Porcus Vincit. Ze zullen zich iedere tick naar de locatie van de Porcus Vincit op dat momentproberen te bewegen. Deze kracht werkt echter wel alleen als de Porcus-Vincit zich op dat moment in een de absolute, hemelsbrede afstand van 150 pixels bevindt. Anders staat deze kracht gelijk aan een vector met twee 0 waarden.
* *Flee:* Dit is de inverse van de seek-kracht. Hij werkt ook alleen binnen de absolute hemelsbrede afstand van 125 pixels.
* *Cohesion:* Deze kracht is een van de krachten die het flocking gedrag van de varkentjes bepaald. Hij werkt aan de hand van seek, en wil daardoor bewegen naar het centrum van de massa van de getagde buren van dit varkentje.
* *Separation:* Deze kracht is een van de krachten die het flocking gedrag van de varkentjes bepaald. Deze kracht stuurt de varkentjes weg van zijn getagde buren.
* *Alignment:* Deze kracht is een van de krachten die het flocking gedrag van de varkentjes bepaald. Deze kracht probeert de richting waarnaar het varkentje zich beweegt gelijk te houden met zijn buren. Dit wordt gedaan aan de hand van een middeling van alle richtingen van de getagde buren.

**Strategie om krachten te combineren**

Eerst wordt de kracht berekend zoals de desbetreffende kracht berekend hoort te worden. Daarna wordt de kracht getweaked aan de hand van een weight doormiddel van een vermenigvuldiging. Dit zorgt ervoor dat de totale grootte van de kracht veranderd, daarna wordt deze kracht nog een keer vermenigvuldigd met de waarde uit het chromosoom van het varkentje (door mogelijke negatieve waarden hier kunnen krachten zelfs tegenovergesteld gaan werken).

Na het vaststellen van elke kracht wordt het doormiddel van de methode “accumulate force” naar rato meegerekend in de totale ‘steeringforce’ van het varkentje. Als alle krachten toegepast zijn heeft de steeringforce zijn eindwaarde bereikt en is het resultaat in combinatie met velocity, speed e.d. op het scherm zichtbaar.

Wall avoidance wordt in deze applicatie heel zwaar meegerekend omdat we niet willen dat de varkentjes het scherm afvliegen en ze juist van de 12 muren in de kaart af bewegen. Seek en Flee gedrag van de varkentjes wordt ook zwaar meegerekend, mede omdat deze krachten alleen werken binnen een bepaalde afstand van de Porcus Vincit en Knabbel. Dit geeft de varkentjes tegelijkertijd wat meer kans om aan boord te komen of om juist te ontsnappen aan Knabbel. Wander is daarentegen juist relatief licht meegerekend.

# Genetisch algoritme

-Initiële populatie. De applicatie begint met generation\_0. Generatie 0 wordt aangemaakt met een willekeurige positie en een chromosoom met willekeurige waarden. Dit chromosoom bestaat uit 5 genen. Ieder representatief aan tabel 2 van de opdracht(aantrekking tot haai, aantrekking tot boot, cohesion, separation en alignment). Vervolgens worden deze varkens in het level neergezet.

- De fitness van de chromosomen wordt heel open bepaald. Namelijk iedereen die de ronde overleeft, heeft het effectief goed gedaan. Iedereen die opgegeten is doet niet meer mee, ze zijn immers dood. De kans is echter wel groter dat een ouder uit de boot komt dus de boot heeft wel een kleine voorkeur.

- De ouder selectie wordt daarmee de volgende: Ouder A komt altijd uit de boot en is een willekeurig varken uit de boot (tenzij er geen varkens gered zijn, dan komt de ouder van de overige overlevenden). Ouder B komt altijd uit de overige overlevenden, ook compleet willekeurig(tenzij er geen overige zijn, dan komt de ouder uit de boot). Mocht niemand het overleeft hebben dan begint er een nieuwe generatie 0.

- De nakomelingen worden als volgt gemaakt: Er wordt een willekeurig getal gekozen tussen 1 en 4(er zijn namelijk 5 genen). Het chromosoom van het kind is dan van het begin tot het willekeurige getal van ouder a en van het willekeurig getal tot en met het einde van ouder b.

- De nieuwe populatie heeft een 1 op 100 kans dat er een mutatie optreed. Een willekeurig kind van die populatie krijgt dan voor een willekeurig gen een willekeurige nieuwe waarde.   
Er zijn na al deze stappen 100 chromosomen aangemaakt bestaand uit: iedere overlevende op de boot + 100 – boot overlevenden kinderen. Deze populatie wordt vervolgens weer op een willekeurige plaats op de kaart gezet.