Verslag programmeer project I fase II

Aïko schürmann
aiko.schurmann@vub.be
computerwetenschappen 2023-2024
0620275

Inhoudsopgave

1 Introductie

Dit document beschrijft de implementatie van een uitbreiding op het spel "Duck Hunt" in de programmeertaal Scheme. Dit project is onderdeel van het vak "Programmeer Project I", waarin een retrogame met extensies moet worden geïmplementeerd. Dit jaar is gekozen voor het spel "Duck Hunt" uit 1984 voor de NES.

"Duck Hunt" (Deluxe) is een first-person shooter waarin de speler in elke ronde een minimumaantal doelen moet raken. Deze doelen worden weergegeven als vogels. Er zullen ook obstakels zijn waar deze vogels achter kunnen schuilen. En valse doelwitten waar je niet op mag schieten.

1.1 Spelonderdelen fase I

Al deze onderdelen zijn geïmplementeerd in mijn code voor fase I

1.1.1 Spelwereld Concreet zal in mijn implementatie de spelwereld alles voorstellen dat actief gerenderd wordt tijdens het spelen van het spel. De spelwereld is dus de omgeving waar het spel zich in afspeelt, het zal dus de meeste andere spelelementen bevatten. De obstakels, vogels, geweren maar ook de GUI kan worden aanzien als deel van de spelwereld. Menu's en GUI dat niet gebruikt wordt tijdens het spelen zijn hier dus geen deel van.

1.1.2 Obstakels Obstakels (boom) zullen grafische elementen zijn waar de vogels zich achter kunnen verstoppen om op die manier veilig te kunnen zijn voor de meeste wapens. Er zullen verschillende soorten obstakels zijn met verschillende afmetingen die elk ander gedrag kunnen induceren bij bepaalde vogelsoorten.

1.1.3 Vogels Fase I bevat 1 soort vogel die in het begin van hun levenscyclus elk een willekeurige richting zullen uitvliegen. Wanneer deze de rand van de spelwereld raken zullen ze van richting veranderen. Na meerdere keren worden de vogels verwijderd en zijn deze ontsnapt. Later zullen de verschillende vogelsoorten zullen variatie hebben in snelheid, het aantal kogels dat ze kunnen overleven en hun interactie met obstakels en meer.

1.1.4 Geweren Fase I bevat 1 soort geweer. Geweren schieten een projectiel af, wanneer deze één van de vogels raakt dan verliest de vogel levenspunten. Later zullen de verschillende geweren zullen variatie hebben in herlaadsnelheid, hoeveelheid levenspunten schade en hun interactie met de spelomgeving.

2 ADTs

2.1 Point2D ADT

- point2D::new : neemt twee numbers (x en y coördinaat) en geeft een Position (2D positie) terug.
- point2D::x, point2D::y : nemen elk een Position en geven het corresponderende coördinaat terug.
- point2D::x!, point2D::y! : nemen elk een number en veranderen hun corresponderende coördinaat naar het meegegeven number.
- point2D::equal? : neemt twee Positions, verlelijkt of deze gelijke x en y waarden hebben en geeft true terug indien dit zo is anders false.
- point2D::+!, position::-! : nemen elk een Vec2D, beiden voeren respectievelijk de som of het verschil hiertussen uit, en zetten de x en y waarde van het originele point2D hiernaar.
- point2D::distance : neemt twee Postions, berekent de afstand hiertussen en geeft dit number terug.
- point2D::copy : geeft een identiek kopie van het originele Point2D terug.

Procedure	**Signatuur**
$point2D :: new$	$number, number \rightarrow Point2D$

$point2D :: x$	$\emptyset \rightarrow number$
$point2D :: y$	$\emptyset \rightarrow number$
$point2D :: x!$	$number \rightarrow \emptyset$
$point2D :: y!$	$number \rightarrow \emptyset$
$point2D :: equal?$	$Point2D \rightarrow bool$
$point2D :: +!$	$Vec2D \rightarrow \emptyset$
$point2D :: -!$	$Vec2D \rightarrow \emptyset$
$point2D :: distance$	$Point2D \rightarrow number$
$point2D :: copy$	$\emptyset \rightarrow Point2D$

TABEL 1: Point2D ADT mutators en operatoren

2.2 Vec2D ADT

- vec2D::new : neemt twee numbers (x en y richting) en geeft een Vec2D (2D vector) terug.
- vec2D::x, vec2D::y : geven een number van de corresponderende richting terug.
- vec2D::x!, Vec2D::y! : veranderen hun corresponderende richting naar het meegegeven number.
- vec2D::equal? : neemt twee Vec2Ds, verlelijkt of deze gelijke x en y waarden hebben en geeft true terug indien dit zo is anders false.
- vec2D::+!, vec2D::-! : nemen elk een een Vec2D voeren respectievelijk de som en het verschil hiertussen uit, en zetten de x en y waarde van het originele vec2D hiernaar.
- vec2D::*! : neemt een number voert het produxt uit, en zet de x en y waarde van het originele vec2D hiernaar.
- vec2D::distance : neemt een Vec2D, berekent de afstand hiertussen en geeft dit number terug.
- vec2D::normalize! : maakt van de vector een genormaliseerde vector.
- Vec2D::copy : geeft een identiek kopie van het originele Vec2D terug.
- Vec2D::magnitude : berekent de lengte van de vector en geeft dit number terug.

Procedure	**Signatuur**
$vec2D :: new$	$number, number \rightarrow Vec2D$
$vec2D :: x$	$\emptyset \rightarrow number$
$vec2D :: y$	$\emptyset \rightarrow number$
$vec2D :: x!$	$number \rightarrow \emptyset$
$vec2D :: y!$	$number \rightarrow \emptyset$
$vec2D :: equal?$	$Vec2D \rightarrow bool$
$vec2D :: +!$	$Vec2D \rightarrow \emptyset$

$vec2D :: -!$	$Vec2D \rightarrow \emptyset$
$vec2D ::* !$	$number \rightarrow \emptyset$
$vec2D :: distance$	$Vec2D \rightarrow number$
$vec2D :: normalise$	$\emptyset \rightarrow \emptyset$
$vec2D :: copy$	$\emptyset \rightarrow \emptyset$
$vec2D :: magnitude$	$\emptyset \rightarrow number$

TABEL 2: Vec2D ADT mutators en operatoren

2.3 Bird ADT

- bird::new : neemt symbol (type), number (health), een Pos2D (coördinaat), Vec2D (vliegrichting), number (speed), number (score bij dood) en number (z-index 1 of 2) en geeft een Bird terug.
- bird::position, bird::vector, bird::health, bird::score, bird::speed, bird::type, bird::z-index : geven elk de huidige waarde terug.
- bird::dead? Geeft een true terug als de bird zijn levens < 0 anders false.
- bird::nextframe? : geeft true terug als de interene gepaseerde tijd groter is dan de tijd tussen animaties.
- bird::hasdied? : geeft true terug wanneer de vogel zijn val animatie heeft bereikt (wanneer hij neergeschoten is)
- bird::hasdied! : veranderd deze state naar de meegegeven waarde.
- bird::position!, bird::vector!, bird::health! : nemen elk een type gedefinieerd in bird::new en geven de aangepaste Bird terug.
- bird::update : neemt als argument een number (tijd sinds laatste frame) update interne staat (reflectie) en berekent frame-times
- bird::move! : neemt als argument een number (tijd sinds laatste frame) en voert de beweging van de vogel uit.
- bird::life-list: neemt geen argumenten en geeft de huidig gegenereerde lijst van tiles terug (dit is een lijst met (health) keer een levens icoon)
- bird::generate-life-list: neemt geen argumenten en genereerd de vorige lijst opnieuw (wanneer health veranderd)

Procedure	**Signatuur**
$bird :: new$	$Position, Vec2D, symbol, number, number, number \rightarrow Bird$

$bird :: positon$	$\emptyset \rightarrow Position$
$bird :: vector$	$\emptyset \rightarrow Vec2D$
$bird :: health$	$\emptyset \rightarrow number$
$bird :: score$	$\emptyset \rightarrow number$
$bird :: speed$	$\emptyset \rightarrow number$
$bird :: position!$	$Point2D \rightarrow Bird$
$bird :: vector!$	$Vec2D \rightarrow Bird$
$bird :: health!$	$number \rightarrow \emptyset$
$bird :: type$	$\emptyset \rightarrow symbol$
$bird :: z - index$	$\emptyset \rightarrow number$
$bird :: hasdied?$	$\emptyset \rightarrow bool$
$bird :: hasdied!$	$bool \rightarrow \emptyset$
$bird :: nextframe?$	$\emptyset \rightarrow bool$
$bird :: update$	$number \rightarrow \emptyset$
$bird :: move!$	$number \rightarrow \emptyset$
$bird :: dead?$	$\emptyset \rightarrow bool$
$bird:: life - list$	$\emptyset \rightarrow (list\ bitmap \ldots)$
$bird:: generate - life - list$	$\emptyset \rightarrow (list\ bitmap \ldots)$

TABEL 3: Bird ADT mutators en operatoren

2.4 Gun ADT

- gun::new : neemt een symbol (type), number (damage), number (range), number (cooldown), number (ammo), bool (splash?), bool (burst?)
- gun::ammo, gun::damage, gun::range, gun::type, gun::splash?, : nemen elk een Gun en geven hun respectievelijke waarde terug met als type gedfinieerd in bird::new
- gun::ammo! : neemt een Gun en een number (bullets), het veranderd het aantal kogels naar het meegegeven number en geeft het aangepaste Gun terug.
- gun::shoot! : neemt geen argumenten en geeft een bool terug of het geweer heeft kunnen schieten (update ook interne waarden, ammo, cooldowns etc)
- gun::time-since-last-shot: neemt geen argumenten en geeft een number terug (tijd in ms sinds laatste schot)
- gun::update!: neemt een number als argument (tijd sinds launch game) en geeft niets terug (update interne waarden)
- gun::cooldown: neemt een number als paramter (herlaad factor) en geeft de tijd in ms terug om te herladen
- gun::reset! Neemt geen parameters en geeft niets terug (reset cooldown)

Procedure	Signatuur
$gun :: new$	$symbol, number, number, number, number, bool, bool \rightarrow Gun$
$gun :: type$	$\emptyset \rightarrow symbol$
$gun :: ammo$	$\emptyset \rightarrow number$
$gun :: damage$	$\emptyset \rightarrow number$
$gun :: range$	$\emptyset \rightarrow number$
$gun :: reload\_time$	$\emptyset \rightarrow number$
$gun :: ammo!$	$number \rightarrow Gun$
$gun :: shoot!$	$\emptyset \rightarrow bool$
$gun:: time - since - last - shot$	$\emptyset \rightarrow number$
$gun :: cooldown$	$number \rightarrow number$
$gun :: reset!$	$\emptyset \rightarrow \emptyset$

TABEL 4: Gun ADT mutators en operatoren

2.5 Game ADT

- game::new : neemt een symbol (start state) en geeft een Game terug.
- game::start! Neemt geen parameters en geeft niets terug (start het spel)

Procedure	Signatuur
$game :: new$	$symbol \rightarrow Game$
$game :: start!$	$\emptyset \rightarrow \emptyset$

TABEL 5: Game ADT mutators en operatoren

2.6 Engine ADT

- engine::new :neemt twee numbers (width, height) en geeft een Engine terug.
- game::new-bird: neemt een symbol (type), Bird, en number (tile amount: hoeveel frames de animatie heft)
- engine::render-birds: neemt geen parameters en geeft niets terug (roept render bird up voor elke bird)
- engine::update-birds: neemt een number (dt) en geeft niets terug (roept update bird op voor elke bird.
- engine::active-birds!: neemt een list (list (cons Bird TileSequence)…) als parameter en geeft niets terug
- engine::next-duck-animation!: neemt een Bird TileSequence pair als parameters en geeft niets terug (veranderdt de animatie van flying naar falling (TileSequence moet dus veranderdt worden))
- engine::collision?: neemt een bitmap en een Point2D als parameters en geeft een bool terug (true als dit punt binnen de bitmap valt anders false)
- engine::add-drawable!: neemt een Bitmap en een number (z-index) als parameters en geeft niets terug (voegt Bitmap toe aan de correcte layer)

- engine::new-power-up: neemt een symbol (type) en een Point2D (locatie) als parameters en geeft niets terug (genereerd bitmap en voegt deze toe aan layer)
- engine::add-bird-life-tiles!: neemt een Bird als parameter en geeft niets terug (voegt de door Bird gegenereerde life tiles toe aan de correcte layer)
- engine::(layer-name): heeft geen parameters en geeft een layer terug (engine::ui-layer zou ui-layer terug geven) (mogelijke layers, ui-layer, bird-layer, bird-layer2, stone-layer, scenery-layer)
- engine::active-birds: heeft geen parameters en geeft een (list (Bird . TileSequence) …) terug.
- engine::active-power-ups: heeft geen parameters en geeft een (list (symbol . Bitmap) …) terug.
- engine::reset!: heeft geen parameters en geeft niets terug (reset interne variabelen)
- engine::initialise-mouse-click-callback!: Neemt een procedure als parameter en geeft niets terug.
- engine::initialise-mouse-move-callback! : Neemt een procedure als parameter en geeft niets terug.
- engine::initialise-update-callback! Neemt een procedure als parameter en geeft niets terug.
- engine::initialise-draw-callback!: Neemt een procedure als parameter en geeft niets terug.
- engine::initialise-key-callback!: Neemt een procedure als parameter en geeft niets terug.

Procedure	**Signatuur**
$engine :: new$	$number, number \rightarrow Engine$
$engine :: new - bird$	$symbol, Bird, number \rightarrow \emptyset$
$engine::render - birds$	$\emptyset \rightarrow \emptyset$
$engine::update - birds$	$number \rightarrow \emptyset$
$engine::active - birds!$	$list \rightarrow \emptyset$
$engine::next - duck - animation!$	$(Bird\ .TileSequence) \rightarrow \emptyset$
$engine::collision?$	$Bitmap, Point2D \rightarrow bool$
$engine::add - drawable!$	$Bitmap, number \rightarrow \emptyset$
$engine::new - power - up$	$symbol, Point2D \rightarrow \emptyset$
$engine::add - bird - life - tiles!$	$Bird \rightarrow \emptyset$
$engine :: (layer - name)$	$\emptyset \rightarrow Layer$
$engine::active - birds$	$\emptyset \rightarrow (list\ (Bird\ .TileSequence) \ldots)$
$engine::active - power - ups$	$\emptyset \rightarrow (list\ (symbol\ .Bitmap) \ldots)$
$engine::reset!$	$\emptyset \rightarrow \emptyset$
$engine::initialise - mouse - click - callback!$	$(symbol, symbol, number, number) \rightarrow \emptyset$
$engine::initialise - mouse - move - callback!$	$(number, number) \rightarrow \emptyset$
$engine::initialise - update - callback!$	$(number) \rightarrow \emptyset$
$engine::initialise - draw - callback!$	$(\emptyset) \rightarrow \emptyset$
$engine::initialise - key - callback!$	$(symbol, symbol) \rightarrow \emptyset$

2.7 Round-manager ADT

- round-manager::new : heeft geen parameters en geeft een Round-manager terug.
- round-manager::spawn-power-ups!: heeft geen parameters en geeft niets terug. (spawnt een powerup als genoeg tijd heeft verstreken)
- round-manager::get-next!: heeft geen parameters en geeft niets terug (spawnt vogel(s) indien genoeg tijd heeft verstreken)
- round-manager::level: heeft geen parameters en geeft een number terug (level)
- round-manager::next-level?: heeft geen parameters en geeft een bool terug (true als het volgende level moet beginnen)
- round-manager::next-level!: neemt een bool als parameter en geeft niets terug (wordt gebruikt om de next-level? te resetten)
- round-manager::powerup!: neemt een symbol (type) als parameter en geeft niets terug (activeerd intern een powerup)
- round-manager::update!: neemt een number (dt) als paramater en geeft niets terug (update interne waarden cooldown etc)
- round-manager::focus?, round-manager::double-score?: nemen geen parameters en geven een bool terug (true als de powerup actief is).
- round-manager::reset!: neemt geen argumenten en geeft niets terug (reset interne waarden)

Procedure	**Signatuur**
$round-manager :: new$	$\emptyset \rightarrow Round-manager$
$round-manager :: spawn-power-ups!$	$\emptyset \rightarrow \emptyset$
$round-manager :: get-next!$	$\emptyset \rightarrow \emptyset$
$round-manager :: level$	$\emptyset \rightarrow number$
$round-manager :: next-level?$	$\emptyset \rightarrow bool$
$round-manager :: next-level!$	$bool \rightarrow \emptyset$
$round-manager::powerup!$	$symbol \rightarrow \emptyset$
$round-manager::update!$	$number \rightarrow \emptyset$
$round-manager :: focus?$	$\emptyset \rightarrow bool$
$round-manager :: double-score?$	$\emptyset \rightarrow bool$
$round-manager::reset!$	$\emptyset \rightarrow \emptyset$

TABEL 7: Round-manager ADT mutators en operatoren

2.8 Input ADT

- input::new : neemt geen argumenten en geeft een Input terug
- input::key-pressed? : neemt een char en geeft een bool terug of deze net ingeduwt werd.
- input::key-down? : neemt een char en geeft een bool terug of deze ingeduwt is.
- input::key-up? : neemt een char en geeft een bool terug of deze net losgelaten werd.
- input::mouse-location : geeft de huidige muis locatie terug in een conscell (x . y)
- input::button-pressed? : neemt een symbol (button) en geeft een bool terug of deze net ingeduwt werd.
  input::button-down? : neemt een symbol (button) en geeft een bool terug of deze ingeduwt is.
- input::button-up? : neemt een symbol (button) en geeft een bool terug of deze net losgelaten werd.
- input::get-key-callback: geeft de key-callback procedure terug.
- input::get-mouse-click-callback: geeft de mouse-click-callbackprocedure terug.
- input::get-mouse-move-callback: geeft de mouse-move-callback procedure terug.

Procedure	**Signatuur**
$input :: new$	$\emptyset \rightarrow Input$
$input :: key-pressed?$	$char \rightarrow bool$
$input :: key-down?$	$char \rightarrow bool$
$input :: key-up?$	$char \rightarrow bool$
$input :: mouse-location$	$\emptyset \rightarrow pair$
$input :: button-pressed?$	$symbol \rightarrow bool$
$input :: button-down?$	$symbol \rightarrow bool$
$input :: button-up?$	$symbol \rightarrow bool$
$input :: get-key-callback$	$\emptyset \rightarrow$ (symbol, symbol)
$input :: get-mouse-click-callback$	$\emptyset \rightarrow$ (symbol, symbol, number, number)
$input :: get-mouse-move-callback$	$\emptyset \rightarrow$ (number, number)

TABEL 8: Input ADT mutators en operatoren

2.9 Game-mode ADT

- game-mode::new : neemt een Engine als parameter en geeft een Game-mode terug.

- game-mode::init : heeft geen parameters en geeft niets terug

  (voert de procedure init uit)

- game-mode::get-draw-update : heeft geen parameters en geeft een procedure terug
- game-mode::get-logic-update : geeft de logic update terug van de game mode

Procedure	**Signatuur**
$game - mode :: new$	$Engine \rightarrow Game - mode$
$game - mode :: get - draw - update$	$\emptyset \rightarrow (\emptyset)$
$game - mode :: get - logic - update$	$\emptyset \rightarrow (\text{number})$
$game - mode :: init$	$\emptyset \rightarrow \emptyset$

TABEL 9: game-mode ADT mutators en operatoren

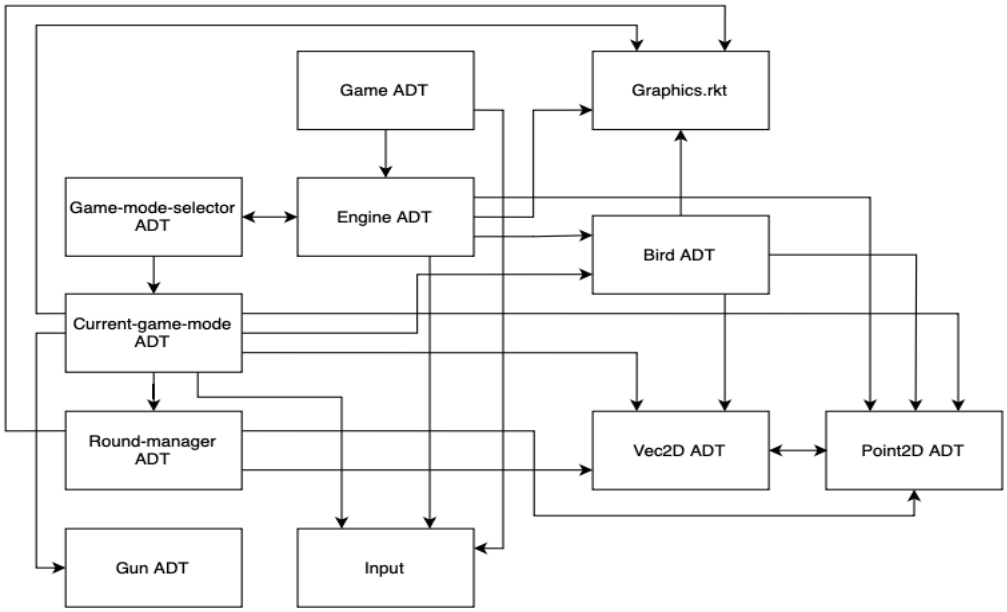
2.10 Game-mode-selector ADT

- game-mode-selector::new : neemt een Engine als parameter en geeft een Game-mode-selector terug
- game-mode-selector::get-draw-update :heeft geen parameters en geeft de draw procedure terug van de actieve game
- mode game-mode-selector::get-logic-update : heeft geen parameters en geeft de logic update terug van de actieve game mode

Procedure	**Signatuur**
$game - mode - slector :: new$	$Engine \rightarrow Game - mode - selector$
$game - mode - slector :: get - draw - update$	$\emptyset \rightarrow (\emptyset)$
$game - mode - slector :: get - logic - update$	$\emptyset \rightarrow (\text{number})$

TABEL 10: game-mode-selector ADT mutators en operatoren

3 Afhankelijkheidsdiagram



TABEL 11: game-mode-selector ADT mutators en operatoren

- **Vec2D** is afhankelijk van **Position** omdat *vec2D::get_normalised_vec* gebruik maakt van twee **Positions** als parameters. En **Position** is afhankelijk van **Vec2D** omdat *position::+ en position::-* gebruik maken van een vector om bij een positie op te tellen.
- **Bird** maakt gebruik van **Vec2D** om de huidige vliegrichting op te slaan en **Bird** maakt ook gebruik van **Position** om zijn huidige locatie op te slaan.
- **Game** maakt geen gebruik van Engine en **Engine** om de **Input** op te vragen nadien worden aan **Input** callbacks opgevraagd
- **Gun** maakt ook geen gebruik van andere ADTs omdat het enkel wat variabelen moet kunnen bijhouden en manipuleren.
- **Game-mode-selector** is afhankelijk van **Engine** omdat het deze doorgeeft aan **current-game-mode.** En is daardoor ook van **current-game-mode** afhankelijk
- **Current-game-mode** is afhankelijk van **Engine** omdat het de teken-logica verder uitbreidt.Ook maakt het gebruik van **Vec2D**, **Point2D Round-manager** Gun en **input**
- **Engine** is afhankelijk van **Input** omdat het ADT hier geïnstantieerd wordt dit zodat het overal

toereik baar is. Ook maakt het gebruik van **Bird,** en **Game-mode-selector** om callbacks op te vragen.

- **Round-manager** is afhankelijk van **Bird** omdat het nieuwe vogels aanmaakt, **Point2D** en **Vec2D** als parameters voor **Bird**

4 Planning

Week	**Taak**
Week 10	*Indienen voorstudie*
Week 11	*Implementeren Vec2D en Position*
Week12-13	*Implementeren Gun Bird Obstacle*
Week 14 – 16	*Implementeren gameloop en Game*
Week 17-21	*Examenperiode + lesvrije week*
Week 18 - 22	*Afwerken game/ bugfixes*
Week 23	*Schrijven verslag + indienen verslag en game*
Week 26	*Indienen voorstudie*
Week 27	*Implementeren round-manager*
Week 29	*Implementeren vogels*
Week 31	*Implementeren guns*
Week 33	*Implementeren powerups*
Week 36	*Afwerken game / bugfixes*
Week 37	*Schrijven verslag + indienen verslag en game*

TABEL 12: Planning

De planning verliep volgens plan, in fase I waren er enkele vertragingen dit was niet het geval in fase II