Programmeerproject 1:
Verslag fase 2 "Tower Defence"

Mathis Arthur Boumaza
0555462
Mathis.Arthur.Boumaza@vub.be
Bachelor in de Computerwetenschappen

Academiejaar 2022-2023

Inhoudsopgave

1 Inleiding

Een onderdeel van de Bachelor in de Computerwetenschappen, bestaat eruit om een programmeerproject te maken waarin verwacht word een spel te maken. De opdracht dit jaar is om een tower defence spel te maken. Zo'n spel bestaat uit een spelwereld met een pad erop waarop verschillende soorten monsters kunnen lopen doorheen verschillende levels en ronden die in moeilijkheidsgraad stijgen naarmate het spel vordert. Deze monsters moeten vermoord worden vooraleer ze het einde van het pad bereiken. Om ze te vermoorden wordt gebruik gemaakt van verschillende soorten torens die een soort projectiel afvuren indien er monsters in de buurt zijn. Deze torens kan door de speler in de spelwereld geplaatst worden zolang het niet op het pad is of op een andere toren. Daarbovenop kan de speler ook power-ups gebruiken als hulpmiddel om monsters te vermoorden. Indien het de speler niet lukt om te vermijden dat monsters het einde van het pad bereiken, zal hij levens verliezen en mogelijks sterven, wat het einde van het spel betekent.

Om tot een volwaardig spel te komen werd verwacht om voor fase 2 verschillende functionaliteiten te implementeren om zo tot een afgewerkt spel te komen. Om te starten werd het spel uitgebreid door een Geld en Levens ADT te introduceren. Deze laten toe het geld van de speler en de levens van de speler voor te stellen. Deze werden dan toegevoegd tot het Spel ADT om zo de notie van kost/winst en levens toe te voegen aan het spel. Daarbovenop werd ook een Power-up ADT aangemaakt om de tank en bommenregen power-ups voor te stellen. Deze werd dan ook tot het Spel ADT toegevoegd om de notie van power-ups toe te voegen aan het spel. Het Teken ADT werd dan uitgebreid om deze nieuwe spelelementen te kunnen tekenen op het scherm. De ADTs gemaakt in de vorige fase werden uitgebreid om extra spelelementen toe te voegen. Meer bepaald werd het Monster ADT uitgebreid om groene, paarse en gele monsters voor te kunnen stellen met hun respectievelijke gedrag alsook om random drops voor te stellen. Daarbovenop werd ook het Toren en Projectiel ADT uitgebreid om de net-, vuurbal- en bomwerptorens te kunnen voorstellen. Het Level ADT werd analoog uitgebreid om deze verschillende spelelementen met elkaar te kunnen doen samenwerken om zo een bijna werkend spel te maken. Om deze spelelementen op een dynamische manier te laten samenwerken, werd het Spel ADT uitgebreid om zo een spel te bekomen. Natuurlijk om deze elementen dan op het scherm weer te geven, werdt het Teken ADT uitgebreid om deze zaken op het scherm te tekenen, wat een volwaardig spel oplevert.

Het document zal meer in detail gaan over hoe ik deze verschillende functionaliteiten tot stand doen komen heb. Meer bepaald zal ik het hebben over de ADTs die ontwikkelt zijn en hun operaties en hoe ze gerelateerd zijn met elkaar door middel van een afhankelijkheidsdiagram. Evenzeer zal een overzicht gegeven worden over wanneer wat gedaan werd en ook zullen de verschillen in afhankelijkheidsdiagram en planning tussen de voorstudie en de huidige stand van zaken besproken worden. En om af te sluiten zal een beknopte uitleg volgen over hoe men het spel kan opstarten.

2 ADTs

Om de functionaliteiten te implementeren zijn, zoals uitgelegd, veel ADTs gebruikt geweest om tot een volwaardig spel te komen. Hier worden de ADTs en hun operaties beschreven alsook de operaties hun signaturen.

2.1 Positie ADT

Vermits verschillende functionaliteiten de notie van positie en beweging nodig hebben, bijvoorbeeld een monstertje beweegt vooruit of een toren heeft een bepaalde positie, heb ik een ADT gemaakt die het mogelijk maakt om posities op een 2 dimensional vlak voor te stellen. Dit ADT zal bij aanmaak 2 number datatypes samennemen die dan een positie in een vlak voorstelt. Verschillende operatoren werken op zo'n positie object, namelijk men kan de coordinaten ervan opvragen en veranderen, maar men kan ook vragen als een positie gelijk is aan een andere gegeven positie. Dit ADT vormt de basis voor vele andere ADTs.

Naam	**Signatuur**
maak-positie-adt	(number, number → positie)
x	number
y	number
x!	(number → ∅)
y!	(number → ∅)
gelijk?	(positie → boolean)
ceil	(∅ → positie)
flo	(∅ → positie)
positie-copieer	(∅ → positie)
afstand	(positie → number)

Tabel 1: De operaties bevat in het Positie ADT

De operaties van het positie ADT zijn:

- *maak-positie-adt* is de aanmaakoperatie die toelaat om een positie voor te stellen in een 2 dimensionale ruimte. Het neemt 2 getallen als invoer en zal een positie ADT maken.
- x en y zijn de operaties om respectievelijk de x en y coordinaat van het positie object te verkrijgen.
- *x!* en *y!* zijn operaties om respectievelijk de x en y coördinaat van het positie object te veranderen naar een vooropgegeven waarde.
- *gelijk?* is een operatie die nagaat als het positie object gelijk is aan een ander vooropgegeven positie object. We zeggen dat 2 positie objecten gelijk zijn indien hun respectievelijke x en y waarden gelijk zijn. Uit deze operatie komen booleans, namelijk #t als ze gelijk zijn en #f als ze niet gelijk zijn.
- *ceil* en *flo* zijn operaties die op basis van het beschouwde positie object, een nieuw positie object aanmaken maar met de coordinaten respectievelijk naar boven en naar beneden afgerond.
- *positie-copieer* is een operatie om het huidige positie object "te copieren", meer bepaald zal het een nieuw positie object aanmaken met dezelfde coordinaten als het beschouwde positie object.
- *afstand* is een operatie die gegeven een positie, de afstand tussen het huidige positie en de gegeven positie bereken. Dit word gedaan a.d.h.v. de Euclidische afstand.

2.2 Pad ADT

Het pad ADT zal een pad voorstellen waarop de monstertjes zullen wandelen om tot de basis te geraken. Het is dit pad object dat de speler zal moeten beschermen met behulp van torens die geplaatst kunnen worden

Naam	Signatuur
maak-pad-adt	(pair → pad)
posities	vector
lengte	number
keer-punten	pair
keer-tekens	pair
begin	positie
einde	positie
pad-positie	(number → positie)
toren-in-pad?	(toren → boolean)
dichste-punt	(positie → positie)

Tabel 2: De operaties bevat in het Pad ADT

door de speler. Dit ADT is een abstractie bovenop het positie ADT die de posities van het pad zal bijhouden zodanig dat monsters over dat pad kunnen lopen en dat er geen torens op gebouwd kunnen worden.

De operaties van het pad ADT zijn:

- *maak-pad-adt* is de aanmaakoperatie van het ADT, het zal het pad object aanmaken. Om dit te doen zal het een lijst van 3 elementen innemen die respectievelijk, het aantal keerpunten (punten waarbij het verloop van het pad van richting veranderd), de keertekens (tekens die zeggen in welke zin het verloop van het pad veranderd is) en de werkelijke vector van posities van het pad voorstelt.
- *posities* is een operatie die een vector bestaande uit de posities van het pad object teruggeeft.
- *lengte* is een operatie die de lengte van de vector van posities waarop het pad object gebouwd is, teruggeeft.
- *keer-punten* is een operatie die de lijst van keerpunten van het pad teruggeeft.
- *keer-tekens* is een operatie de de lijst van keertekens van het pad teruggeeft.
- *begin* en *einde* zijn operaties die respectievelijk de begin en einde positie van het pad teruggeven.
- *pad-positie* is een operatie die gegeven een getal, die een index voorstelt, de positie geeft op die index in een vector van posities.
- *toren-in-pad?* is een operatie die, gegeven een toren, nagaat als die toren zich bevind op het pad. Het resultaat van deze operatie is #t als de toren zich op het pad bevind en #f als de toren zich niet op het pad bevind.
- *dichste-punt* is een operatie die gegeven een positie, de dichste positie op het pad teruggeeft relatief tot de gegeven positie.

2.3 Monster ADT

Het monster ADT laat toe om monsters voor te stellen die op het pad zullen lopen en het einde proberen te bereiken. Dit ADT is een abstractie bovenop het positie, pad en projectiel ADT. Het positie ADT wordt gebruikt om de positie van het monster object bij te houden, maar deze posities zijn dan beperkt tot posities op het pad, wat betekent dat het ook gebouwd is bovenop het pad ADT. Daarbovenop word ook het projectiel ADT gebruikt om de snelheid van het monster object te verhogen na een vermindering.

Naam	**Signatuur**
maak-monster-adt	(symbol, pad . positie → monster)[1]
positie	positie
type	symbol
volgende-positie!	(∅ → ∅)
einde?	(∅ → boolean)
gestorven?	(∅ → boolean)
geen-actie-groen-monster?	(∅ → boolean)
actie-monster-sterven!	(∅ → monster ∪ vector)
actie-monster-levend!	(symbol . projectiel ∪ symbol→ ∅)
verhoog-levens!	(∅ → ∅)
voeg-net-projectiel-toe!	(projectiel → ∅)
update-tijd-net-projectielen!	(number → ∅)
net-al-vertraagd?	(projectiel → pair ∪ {#f})
haal-weg-verlopen-net-projectielen!	(∅ → ∅)
soort	symbol

Tabel 3: De operaties bevat in het Monster ADT

De operaties van het monster ADT zijn:

- *maak-monster-adt* is de aanmaakoperatie die toelaat om een monster object aan te maken. Het neemt op zijn minst 2 dingen als invoer, het type van monster, en een pad. Daarbovenop is het ook mogelijk om nog meer argumenten toe te voegen, meer bepaald een positie waarvan een monster object moet beginnen voortbewegen.
- *positie* is een operatie die de huidige positie van het monster object teruggeeft.
- *type* is de operatie die het type van het monster object teruggeeft.
- *volgende-positie!* is een operatie die toelaat om het monster object naar de volgende positie te laten gaan, dit is gebaseerd op het pad ADT.
- *einde?* is de operatie die nagaat als het monster object de eind positie bereikt heeft, namelijk het einde van het pad.
- *gestorven?* is een operatie die nagaat als het monster object gestorven is.
- *geen-actie-groen-monster?* is een operatie die nagaat als een groen monster object een actie moet doen bij een projectiel aanraking. Meer bepaald is het resultaat van deze operatie #t als een groen monster object geen actie moet uitvoeren en #f als hij wel een actie moet uitvoeren.
- *actie-monster-sterven!* is een operatie die, bij het sterven van het monster object, afhankelijk van het soort monster object een actie uitvoert.
- *actie-monster-levend!* is een operatie die afhankelijk van de gegeven actie en de inhoud van de optionele variabele het monster object verminderen van levens of vertragen.
- *verhoog-levens!* is een operatie die het monster object zijn levens met 1 naar boven doet.

[1]Alles die na het puntje komt stellen optionele parameters voor, in mijn geval verwacht ik maar maximum 1 optionele parameter. Vermits ik niet wist hoe ik deze signatuur moest neerschrijven heb ik het zo gedaan.

- *voeg-net-projectiel-toe!* is een operatie die een gegeven netprojectiel object toevoegd aan de lijst van netprojectiel objecten die het monster vertragen. (De lijst is een associatelijst met de tijd verlopen sinds het toevoegen van het netprojectiel object.)
- *update-tijd-net-projectielen!* is een operatie die in de assocatie lijst van netprojectiel objecten en verlopen tijd, de verlopen tijd update van elke netprojectiel.
- *net-al-vertraagd?* is een operatie die nagaat als een bepaalde netprojectiel object het monster object al vertraagd heeft. Meer bepaalde is het resultaat van deze operatie een pair indien het netprojectiel object de monster al vertraagd heeft en #f indien dit niet zo is.
- *haal-weg-verlopen-net-projectielen!* is een operatie die de verlopen netprojectiel objecten uit de lijst haalt.
- *soort* is een operatie die het soort van het object teruggeeft (in dit geval een monster symbool).

2.4 Toren ADT

Het toren ADT zal een toren voorstellen die projectiel objecten zal afschieten naar dichtbijzijnde monsters op het pad. Het ADT is opnieuw een abstractie die gebouwd is bovenop het positie, projectiel en pad ADT. Het positie ADT zal gebruikt worden om de toren zijn positie bij te houden, het projectiel ADT word gebruikt om projectiel objecten af te vuren en het pad ADT wordt gebruikt om bomprojectiel objecten af te vuren.

Naam	**Signatuur**
maak-toren-adt	(positie, symbol → toren)
positie	positie
type	symbol
toren-posities	vector
in-toren?	(toren → boolean)
in-buurt?	(monster → boolean)
schiet!	(monster, pad → ∅)
projectiel-update!	(level, number → ∅)
projectielen	pair
update-afvuur-tijd!	(number → ∅)
schieten?	(∅ → boolean)
initialiseer-tijd!	(∅ → ∅)
soort	symbol

Tabel 4: De operaties bevat in het Toren ADT

De operaties van het toren ADT zijn:

- *maak-toren-adt* is de aanmaakoperatie die toelaat om een toren object aan te maken. Het neemt een positie als invoer, die de positie van het toren object voorstelt in de spelwereld, en een symbool die het type toren voorstelt.
- *positie* is een operatie die de positie van de toren in de spelwereld teruggeeft.
- *type* is een operatie die het type van de toren teruggeeft.
- *toren-posities* is een operatie die 4 posities teruggeeft, die de toren posities voorstellen. We doen dit omdat 1 positie niet voldoende is om een toren zijn positie voor te stellen, vermits een toren groter is dan 1 centrale positie.

- *in-toren?* is een operatie die nagaat als een bepaalde gegeven toren in de toren zit, meer bepaald het kijkt voor overlap van posities tussen de 2 torens gebruikmakend van de toren-posities van beide. het geeft een #t terug als de torens overlap hebben, en #f als dit niet zo is.
- *in-buurt?* is een operatie die nagaat als een bepaalde monster in de buurt van de toren zit, het zal #t teruggeven als de monster zich in de buurt bevind, en #f als dit niet zo is.
- *schiet!* is de operatie die toelaat om een toren te doen schieten naar een monster op het pad.
- *projectiel-update!* is een operatie die al de projectielen die de toren geschoten heeft en die nog niet aangekomen zijn aan hun bestemming, hun posities updaten om zo dichter te komen bij het monster die ze moeten raken.
- *projectielen* is de operatie die al de afgevuurde projectielen teruggeeft die nog niet toegekomen zijn bij het monster die ze moeten raken.
- *update-afvuur-tijd!* is een operatie die de afvuur tijd van het toren object update. De afvuur tijd stelt de verlopen tijd voor sinds het vorige schot.
- *schieten?* is een operatie die nagaat indien het toren object mag schieten naar een volgend monster object.
- *initialiseer-tijd!* is een operatie die de afvuurtijd van het toren object op 0 zet.
- *soort* is een operatie die het soort van het object teruggeeft (in dit geval een toren symbool).

2.5 Projectiel ADT

Het projectiel ADT laat toe om projectielen voor te stellen in ons spel en deze door toren objecten te laten gebruiken om monster objecten te vermoorden. Dit ADT is een abstractie bovenop het positie, monster en level ADT. Het positie ADT zal de positie van het projectiel object bijhouden alsook de bestemming positie ervan. Daarbij zal het ook verder bouwen op het monster ADT om bij het naar afgevuurd monster zijn positie aan te komen en zijn actie uit te voeren. Daarbovenop is het ook gebouwd op het level ADT om bepaalde projectiel objecten op het pad object te laten liggen.

Naam	**Signatuur**
maak-projectiel-adt	(positie, symbol, monster ∪ positie . number → projectiel)
positie	positie
type	symbol
bestemming-bereikt?	(∅ → boolean)
afgehandelt?	(∅ → boolean)
volgende-positie!	(∅ → ∅)
actie-te-raken-monster!	(∅ → ∅)
actie-na-positie-bereik!	(level, number → {#f} ∪ projectiel ∪ ∅)
binnen-rand?	(monster → boolean)
niet-bereikt&&afgehandelt?	(∅ → boolean)
soort	symbol

Tabel 5: De operaties bevat in het Projectiel ADT

De operaties van het projectiel ADT zijn:

- *maak-projectiel-adt* is de aanmaakoperatie die ons in staat stelt om een projectiel object aan te maken. Het neemt op zijn minst 3 dingen als invoer, namelijk de begin positie van het projectiel object, het type van het projectiel object en een monster of positie object. Daarbovenop kan men ook een snelheid meegeven waarmee het projectiel object zijn snelheid geinitaliseerd kan worden.
- *positie* is de operatie die de huidige positie van het projectiel zal teruggeven.
- *type* is de operatie die het type afgevuurd projectiel teruggeeft.
- *bestemming-bereikt?* is een operatie die nagaat als het projectiel object zijn bestemming, die bij de aanmaak bepaald wordt, al bereikt heeft. Het resultaat van deze operatie is #t indien het projectiel object zijn bestemming bereikt heeft en #f indien dit niet zo is.
- *afgehandelt?* is een operatie die nagaat indien een projectiel object volledig afgehandelt is, m.a.w zijn acties uitegevoerd heeft. Het geeft #t terug indien dit het geval is en #f indien dit niet zo is.
- *volgende-positie!* is de operatie die het projectiel object zijn positie naar zijn volgende positie zal zetten. De volgende positie wordt bepaald door bij de aanmaak van het object constanten te maken, a.d.h.v de bestemming en start positie, die de verandering van positie zullen voorstellen en deze dan op te tellen bij de huidge positie van het projectiel object, iedere keer als die operatie opgeroepen word.
- *actie-te-raken-monster!* is een operatie die een actie uitvoert op het monster object, wanneer het projectiel object zijn positie bereikt.
- *actie-na-positie-bereik!* is een operatie die een actie uitvoert na het projectiel object zijn positie bereikt heeft.
- *binnen-rand?* is een operatie die nagaat als een monster object zich binnen de rand van een projectiel object bevind.
- *niet-bereikt&&afgehandelt?* is een operatie die nagaat indien het projectiel object nog niet zijn bestemming bereikt heeft en afgehandelt is geweest. Meer bepaald, het geeft #t indien het object nog niet afgehandelt geweest is of zijn positie bereikt heeft en #f indien hij wel alles afgehandelt heeft en zijn bestemming bereikt heeft.
- *soort* is een operatie die het soort van het object teruggeeft (in dit geval een projectiel symbool)

2.6 Geld ADT

Het Geld ADT zal het geld van de speler voorstellen. Het zal de notie van kost en winst toevoegen in het spel voor de speler. Meer bepaald zal het de level, ronde en monster winst mogelijk maken en de kost voor power-ups en torens mogelijk maken.

De operaties van het geld ADT zijn:

- *maak-geld-adt* is de aanmaakoperatie die ons toelaat om een geld object aan te maken. Het neemt een initieel bedrag als invoer.
- *voldoende-geld?* is de operatie die, gegeven een symbool (type object), nagaat indien men voldoende geld heeft om dit object te kopen. Meer bepaald, geeft het #t terug indien men voldoende geld heeft en #f indien dit niet het geval is.

Naam	Signatuur
maak-geld-adt	(number → geld)
voldoende-geld?	(symbol → boolean)
verwijder-geld!	(symbol → ∅)
voeg-geld-toe!	(symbol, boolean . boolean → ∅)
status	number
reset!	(∅ → ∅)

Tabel 6: De operaties bevat in het Geld ADT

- *verwijder-geld!* is de operatie die gegeven een symbool (type object), de respectievelijk kost van dat object aftrekt.
- *voeg-geld-toe!* is de operatie die gegeven een symbool (type object), de respectievelijke winst van dat object bij het bedrag optelt. (Een aantal extra invoervariabelen zijn nodig om te bepalen indien men extra geld moet toevoegen voor een tank en indien een groen monster het geld object mag bedrag verhogen.)
- *status* is de operatie die toelaat om het huidig bedrag terug te krijgen.
- *reset!* is de operatie die het bedrag terug op het initieel bedrag zet.

2.7 Levens ADT

Het levens ADT zal de levens van de speler voorstellen. Het zal de notie van sterven en endgame toevoegen aan het spel.

Naam	Signatuur
maak-levens-adt	(number → levens)
levens-verminder!	(pair → ∅)
dood?	(∅ → boolean)
status	number
reset!	(∅ → ∅)

Tabel 7: De operaties bevat in het Levens ADT

De operaties van het levens ADT zijn:

- *maak-levens-adt* is de operatie die ons toelaat om een levens object aan te maken. Het neemt het initieel aantal levens als invoer.
- *levens-verminder!* is de operatie die een speler zijn levens doet verminderen met een hoeveelheid. (Deze hoeveelheid is afhankelijk van het soort monsters dat de basis bereikt hebben.)
- *dood?* is de operatie die nagaat indien de speler al zijn levens verloren is.
- *status* is de operatie die het huidig aantal levens van de speler teruggeeft.
- *reset!* is de operatie die het aantal levens terug op het initieel aantal levens zet.

2.8 Power-up ADT

Het power-up ADT zal een power-up in het spel voorstellen. Meer bepaald een tank en bommenregen. Het is een abstractie bovenop het positie en pad ADT. Het positie ADT wordt gebruikt om de positie van het power-up bij te houden en het pad ADT word gebruikt om een tank object voort te bewegen op het pad object.

Naam	**Signatuur**
maak-power-up-adt	(pad, symbol . positie → power-up)
positie	positie
einde?	(∅ → boolean)
update!	(number → ∅)
bommen	pair
tijd-afgelopen?	(∅ → boolean)
bom-explosie!	((vector, symbol → ∅) → ∅)
drop-positie!	positie ∪ {#f}
in-drop-rand?	(positie → boolean)
verander-drop-status!	(∅ → ∅)
type	symbol
soort	symbol

Tabel 8: De operaties bevat in het Power-up ADT

De operaties van het power-up ADT zijn:

- *maak-power-up-adt* is de aanmaakoperatie die ons toelaat om een power-up object aan te maken. Het neemt op zijn minst 2 dingen als invoer, een pad object en een symbool die het type van het power-up object voorstelt. Daarbovenop kan ook een positie meegegeven worden om de positie waar het power-up object gedropt is geweest voor te stellen.
- *positie* is een operatie die de positie van het power-up object teruggeeft.
- *einde?* is een operatie die nagaat als een tank power-up object aan het eind van het pad object gekomen is.
- *update!* is een operatie die de staat van het power-up object zal updaten.
- *bommen* is een operatie die de lijst van bommen van het bommenregen power-up object teruggeeft.
- *tijd-afgelopen?* is een operatie die nagaat als de tijd dat de bommenregen power-up objecten moeten blijven liggen, al verlopen is. Het geeft #t indien die tijd afgelopen is en #f indien dit niet het geval is.
- *bom-explosie!* is een operatie die een bommenregen power-up object zijn explosie uitvoert.
- *drop-positie!* is een operatie die de positie waar een power-up object gedropt is, indien het een gedropte power-up object is, teruggeeft.
- *in-drop-rand?* is een operatie die gegeven een positie nagaat als de positie in de drop rand van een gedropt power-up object zit.
- *verander-drop-status!* is een operatie die de drop status van een gedropt power-up object verandert naar #f.

- *type* is een operatie die het type van het power-up object teruggeeft.
- *soort* is een operatie die het soort van het object teruggeeft (in dit geval een power-up of drop-power-up symbool).

2.9 Level ADT

Het level ADT zal een level van het spel voorstellen. Het zal een heel level beheren van start tot einde en zal al de elementen die geupdate moeten worden updaten. Dit ADT is gebouwd op veel andere ADTs (bv. monsters, geld, levens, ...) en het gebruikt deze ADTs om ze te doen samenwerken tot een groter geheel.

Naam	**Signatuur**
maak-level-adt	(number, pair, geld, levens → level)
pad	pad
monsters	pair
torens	pair
voeg-toren-toe!	(toren → ∅)
update-monsters!	(number . symbol → ∅)
update-torens-projectielen-positie!	(number → ∅)
update-torens-projectielen-afschieten!	(pad, number → ∅)
initialiseer-toren-tijden!	(∅ → ∅)
update-power-ups!	(number → ∅)
monster-na-monster	(monster → monster ∪ {#f})
voeg-net-projectiel-toe!	(projectiel → ∅)
voeg-power-up-toe!	(symbol, projectiel → ∅)
verkrijg-projectielen	(∅ → pair)
vekrijg-tank-power-ups	pair
verkrijg-bommen-regen-power-ups	pair
verkrijg-gedropte-power-ups	pair
drop-opraap!	(number, number → pair)
explodeer-monsters-in-buurt!	(vector, symbol → ∅)
ronde-einde?	(∅ → boolean)
ronde-einde!	(∅ → ∅)
zet-monster-lijst!	(pair → ∅)

Tabel 9: De operaties bevat in het Level ADT

De operaties van het level ADT zijn:

- *maak-level-adt* is de aanmaakoperatie die toelaat om een level object aan te maken. Het neemt 4 dingen als invoer, namelijk het pad index, een lijst van monsters, een geld object en een levens object.
- *pad* is de operatie die toelaat om het pad van het level object terug te krijgen.
- *monsters* is de operatie die de gespawnde monsters teruggeeft in een lijst.
- *torens* is de operatie die de gezette torens op de spelwereld teruggeeft in een lijst.
- *voeg-toren-toe!* is de operatie die toelaat om torens toe te voegen tot het level object.
- *update-monsters!* is de operatie die zal zorgen dat de monster objecten zullen voortbewegen en zo naar het einde van het pad voortbewegen.

- *update-torens-projectielen-positie!* is de operatie die de afgeschoten projectielen hun posities zal updaten, om zo voort te bewegen naar hun bestemming.
- *update-torens-projectielen-afschieten!* is de operatie die er voor zal zorgen dat torens zullen schieten naar het eerste monster dat ze tegenkomen, namelijk het eerste monster dat in hun buurt is.
- *initialiseer-toren-tijden!* is de operatie die de toren hun tijd verlopen sinds de laatste afvuur tijd, terug op 0 zetten.
- *update-power-ups!* is de operatie die de power-ups updaten om bijvoorbeeld verder te bewegen of te exploderen.
- *monster-na-monster* is een operatie die het monster teruggeeft die volgt op een gegeven monster.
- *voeg-net-projectiel-toe!* is een operatie die toelaat om een netprojectiel object toe te voegen aan de actieve netprojectiel objecten op het pad object.
- *verkrijg-projectielen* is de operatie die al de afgevuurde projectielen zal teruggeven in een lijst.
- *verkrijg-tank-power-ups* is een operatie die de actieve tank-power-up objecten teruggeeft.
- *verkijg-bommen-regen-power-ups* is een operatie die de actieve bommenregen power-up objecten teruggeeft.
- *verkrijg-gedropte-power-ups* is een operatie die de gedropte power-up objecten teruggeeft.
- *drop-opraap!* is een operatie die een gedropte power-up object van het pad object haalt en teruggeeft.
- *explodeer-monsters-in-buurt!* is een operatie die alle monsters in de rand van een bom object, hun levens vermindert.
- *ronde-einde?* is een operatie die nagaat indien de ronde beindigt is.
- *ronde-einde!* is een operatie die de ronde ten einde brengt.
- *zet-monster-lijst!* is een operatie die een nieuwe lijst van monster objecten in het level object brengt.

2.10 Teken ADT

Het teken ADT is het ADT die het mogelijk zal maken om de voorgaande objecten op het scherm te tekenen, namelijk de spelwereld, het pad, de monsters, de power-ups, het geld, de levens, de torens en de projectielen. Daarbovenop is het ook het ADT die het mogelijk zal maken om bepaalde toetsen, spellus en muisklik operaties in te stellen. Dit ADT is een abstractie gebouwd op de grafische bibliotheek van Scheme. Het gebruikt deze bibliotheek om alles te tekenen op het scherm.

De operaties van het teken ADT zijn:

- *maak-teken-adt* is de aanmaakoperatie die het teken object zal aanmaken, en dus het venster waarop we tekenen aanmaakt, waarnaar we boodschappen kunnen sturen om bepaalde elementen te tekenen op het scherm en om bepaalde toetsen of een spellus in te stellen. Deze operatie neemt 2 getallen binnen, namelijk het aantal horizontale en verticale pixels groot dat het venster moet zijn.
- *teken-pad!* is de operatie die het zal toelaten een pad op het venster te tekenen.

Naam	**Signatuur**
maak-teken-adt	(number, number → teken)
teken-pad!	(pad → ∅)
verwijder-pad!	(∅ → ∅)
teken-toren!	(toren → ∅)
toren-selectie	(number, number → symbol ∪ {#f})
verwijder-torens!	(∅ → ∅)
power-up-selectie	(number, number → symbol ∪ {#f})
teken-monsters!	(pair → ∅)
teken-projectielen!	(pair → ∅)
teken-tank-power-up!	(pair → ∅)
teken-bommen-regen-power-up!	(pair → ∅)
teken-gedropte-power-ups!	(pair → ∅)
verwijder-gedropte-power-ups!	(∅ → ∅)
verwijder-bommen!	(∅ → ∅)
teken-afkoeling-acties!	(symbol → ∅)
teken-begin-scherm!	(∅ → ∅)
teken-eind-scherm!	(∅ → ∅)
teken-game-over!	(∅ → ∅)
verwijder-scherm!	(∅ → ∅)
set-muis-toets!	((symbol ∪ char, symbol, number, number → ∅) → ∅)
set-spel-lus!	((number → ∅) → ∅)
set-toets-procedure!	((symbol, symbol ∪ char → ∅) → ∅)
update-tekst-teken!	(symbol, number → ∅)
pas-aan	(number, number → pair)
buiten-beperking?	(number, number → boolean)

Tabel 10: De operaties bevat in het Teken ADT

- *verwijder-pad!* is de operatie die het huidige getekent pad verwijdert van het scherm.
- *teken-toren!* is de operatie die het mogelijk maakt om een toren op het venster te tekenen.
- *toren-selectie* is de operatie die gegeven x en y pixels, bepaalt welke toren de speler geselecteerd heeft.
- *verwijder-torens!* is de operatie die de getekende torens van het scherm verwijdert
- *power-up-selectie* is de operatie die gegeven x en y pixels, bepaalt welke power-up de speler geselecteerd heeft.
- *teken-monsters!* is een operatie die op het venster monsters tekent.
- *teken-projectielen!* is een operatie die op het venster de afgeschoten projectielen tekent.
- *teken-tank-power-up!* is de operatie die tank power-ups op het scherm zal tekenen.
- *teken-bommen-regen-power-up!* is de operatie die bommenregen power-ups op het scherm zal tekenen.
- *teken-gedropte-power-ups!* is de operatie die toelaat om gedropte power-ups te tekenen op het scherm.
- *verwijder-gedropte-power-ups!* is de operatie die de getekende gedropte power-ups van het scherm verwijdert.

- *verwijder-bommen!* is de operatie die de getekende bommen van het bommenregen power-up verwijdert van het scherm.
- *teken-afkoeling-acties!* is de operatie die de afkoeling van power-ups op het scherm tekent.
- *teken-begin-scherm!* is de operatie die het begin scherm op het scherm tekent.
- *teken-eind-scherm!* is de operatie die het eind scherm op het scherm tekent.
- *teken-game-over!* is de operatie die game-over op het scherm tekent.
- *verwijder-scherm!* is de operatie die een game-over, begin- of eindscherm van het scherm verwijdert.
- *set-muis-toets!* is de operatie die toelaat om bepaalde muis toetsen in te stellen.
- *set-spel-lus!* is de operatie die toelaat om een spel lus te maken.
- *set-toets-procedure!* is de operatie die toelaat om bepaalde toetsenbord toetsen in te stellen.
- *update-tekst-teken!* is de operatie die toelaat om de tekst op het scherm up te daten.
- *pas-aan* is de operatie die van pixel posities de equivalente coordinaten posities maakt.
- *buiten-beperking?* is de operatie die nagaat als een speler geklikt heeft buiten of binnen de geldige spelwereld. Meer bepaald geeft het #t terug indien hij in de geldige spelwereld heeft geklikt en #f indien dit niet het geval is.

2.11 Spel ADT

Het spel ADT is de kern van dit project, dit ADT zal alles beheren om tot een volwaardig project te komen. Het ADT is een abstractie bovenop alle voorgaande besproken ADTs. Het gebruikt deze ADTs om alles te doen samenwerken en een eindproduct te bekomen.

Naam	**Signatuur**
maak-spel-adt	$(\emptyset \rightarrow \emptyset)$
start!	$(\emptyset \rightarrow \emptyset)$

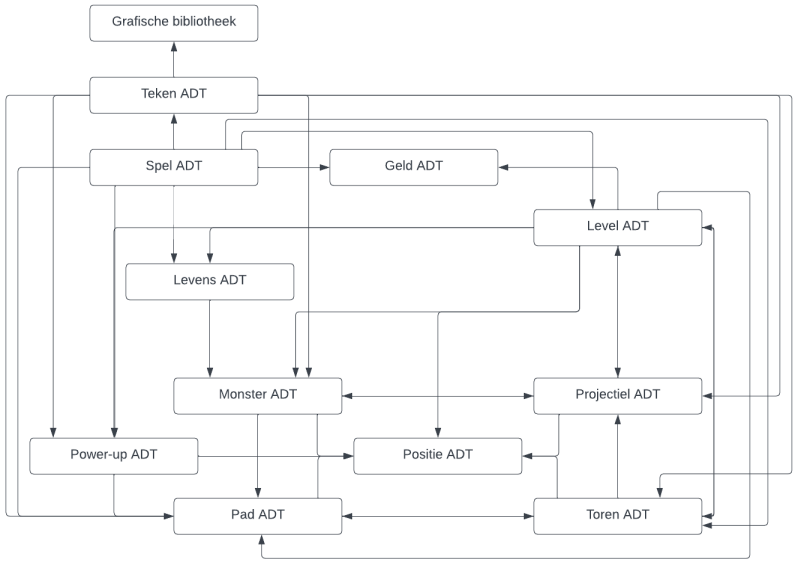
Tabel 11: De operaties bevat in het Spel ADT

De operaties van het spel ADT zijn:

- *maak-spel-adt* is de aanmaakoperatie van het spel object. Het zal de spelwereld aanmaken en de intiele zaken op het scherm tekenen (zoals het pad, achtergrond en menu).
- *start!* is de operatie die ervoor gaat zorgen dat men het spel kan starten indien men de juiste toetsen indrukt.

3 Afhankelijkheidsdiagram

Figuur 1 toont het afhankelijksdiagram van mijn ADTs die gebruikt geweest zijn om het project te implementeren.



Figuur 1: Afhankelijkheidsdiagram

Bij de implementatie van de functionaliteiten van mijn spel, heb ik 11 ADTs gebruikt zoals gezien op het afhankelijkheidsdiagram. Om zo'n afhankelijkheidsdiagram te lezen moet men de dozen zien als mijn ADTs en de pijlen tussen de dozen betekenen dat een ADT afhankelijk is van een ander ADT. Op de figuur kan men zien hoe mijn ADTs van elkaar afhangen:

- Het *Level*, *Monster*, *Pad*, *Power-up*, *Projectiel* en *Toren* ADT zijn afhankelijk van het *Positie* ADT omdat ze allemaal positie objecten gebruiken om posities bij te houden.
- Het *Monster* en *Power-up* ADT zijn afhankelijk van het *Pad* ADT vermits zij deze gebruiken om hun objecten op het pad voort te bewegen. Daarbovenop wordt het *Pad* ADT gebruikt door het *Toren* ADT om bomwerpprojectiel objecten naar het pad te schieten. Het *Level* en *Spel* ADT brengen de spelelementen samen en dus gebruiken ze beide het *Pad* ADT.
- Het *Level* en *Spel* ADT zijn beide afhankelijk van het *Geld* ADT. Ze gebruiken het *Geld* ADT om respectievelijk monsterwinsten erbij te tellen en dingen te bekostigen.
- Het *Level* en *Spel* ADT zijn beide afhankelijk van het *Levens* ADT. Het *Level* ADT gebruikt het *Levens* ADT om levens af te trekken indien monster objecten de basis bereiken. Terwijl het *Spel* ADT gebruikt maakt van het *Levens* ADT om na te gaan indien de speler dood is.
- Het *Toren*, *Monster* en *Level* ADT zijn afhankelijk van het *Projectiel* ADT. Het *Toren* ADT gebruikt het *Projectiel* ADT om projectiel objecten te schieten, terwijl het *Level* en *Monster* ADT het ADT gebruiken om bij netprojectiel objecten, de monster objecten te vertragen.

- Het *Leven*, *Level* en *Projectiel* ADT zijn afhankelijk van het *Monster* ADT. Het *Level* ADT gebruikt het *Monster* ADT om de monster objecten voort te bewegen. Daarbovenop word het *Monster* ADT gebruik door het *Projectiel* ADT om het monster object zijn levens te verminderen bij aankomst. Als laatst gebruikt het *Leven* ADT het ADT om na te gaan hoeveel levens afgetrokken moeten worden van de speler indien een monster object de basis bereikt.
- Het *Level*, *Pad* en *Spel* ADT zijn afhankelijk van het *Toren* ADT. Het *Pad* ADT is afhankelijk van het *Toren* ADT om na te gaan indien een toren object mogelijks op het pad object staat. Het *Spel* ADT hangt af van het *Toren* ADT om gebruikmakend van het *Level* ADT na te gaan als een toren object mogelijks op een ander toren object staat.
- Het *Level* en *Spel* ADT zijn beide afhankelijk van het *Power-up* ADT. Het *Spel* ADT houdt de gekochte en opgeraapte power-up objecten bij, terwijl het *Level* ADT de geactiveerde power-ups bijhoud en uitvoert.
- Het *Projectiel*, *Toren* en *Spel* ADT zijn afhankelijk van het *Level* ADT. Meer bepaald gebruikt het *Projectiel* en *Toren* ADT het *Spel* ADT om bom- en netprojectielen op het pad object te leggen. Daarbovenop gebruikt het *Spel* ADT het ADT om alles up te daten en het spel dynamisch te maken.
- Het *Spel* ADT is afhankelijk van het *Teken* ADT om al de spelelementen te tekenen op het scherm. Daarbovenop is het *Teken* ADT afhankelijk van de *Pad*, *Toren*, *Monsters*, *Projectielen* en *Power-up* ADTs om al de spelelementen te tekenen op het scherm.

4 Tijdsschema

In dit deel bespreken we in welke weken er aan het project gewerkt werd en wat er elke week gedaan werd om zo tot een volwaardig tower-defense spel te geraken. Meer bepaald kan je in Tabel 12 zien hoe het project aangepakt werd.

Week	Actie
Week 27	Op basis van de feedback van fase 1 een aantal ADTs verbetert alsook het implementeren van het Levens en Geld ADT. Het Teken ADT werd ook uitgebreid om deze objecten te kunnen visualiseren op het scherm.
Week 28	Het uitbreiden van het Monster ADT om groene, gele en paarse monster objecten te kunnen voorstellen. Het Teken ADT werd ook uitgebreid om deze monster objecten te kunnen visualiseren.
Week 29	Het uitbreiden van het Toren, Projectiel en Level ADT om netprojectielen en nettorens te kunnen voorstellen. Het Teken ADT werd ook uitgebreid om dit soort netprojectiel en nettoren objecten te kunnen visualiseren op het scherm.
Week 30	Het uitbreiden van het Toren en Projectiel ADT om vuurbalprojectielen en vuurbaltorens te kunnen voorstellen. Het Teken ADT werd ook uitgebreid om dit soort projectiel en toren objecten te kunnen visualiseren op het scherm.
Week 31	Het uitbreiden van het Toren, Projectiel, Level en Pad ADT om bomwerp-projectielen en bomwerptorens te kunnen voorstellen. Het Teken ADT werd ook uitgebreid om dit soort projectiel en toren objecten te kunnen visualiseren op het scherm.
Week 32	Het aanmaken van het Power-up ADT om een tank power-up object te kunnen voorstellen. Het Teken ADT werd ook uitgebreid om dit soort power-up object te kunnen visualiseren op het scherm.
Week 33	Het uitbreiden van het Power-up ADT om een bommenregen power-up object te kunnen voorstellen. Het Teken ADT werd ook uitgebreid om dit soort power-up object te kunnen visualiseren op het scherm.
Week 34	Het uitbreiden van het Spel.rkt file en Monster ADT om respectievelijk randomness en random drops te kunnen implementeren. Het Teken ADT werd ook uitgebreid om deze drops te kunnen visualiseren.
Week 35	Het uitbreiden van het Spel ADT om een continu en herstartend spel te bekomen.
Week 36	Schrijven van het verslag en code review alsook het indienen van het project

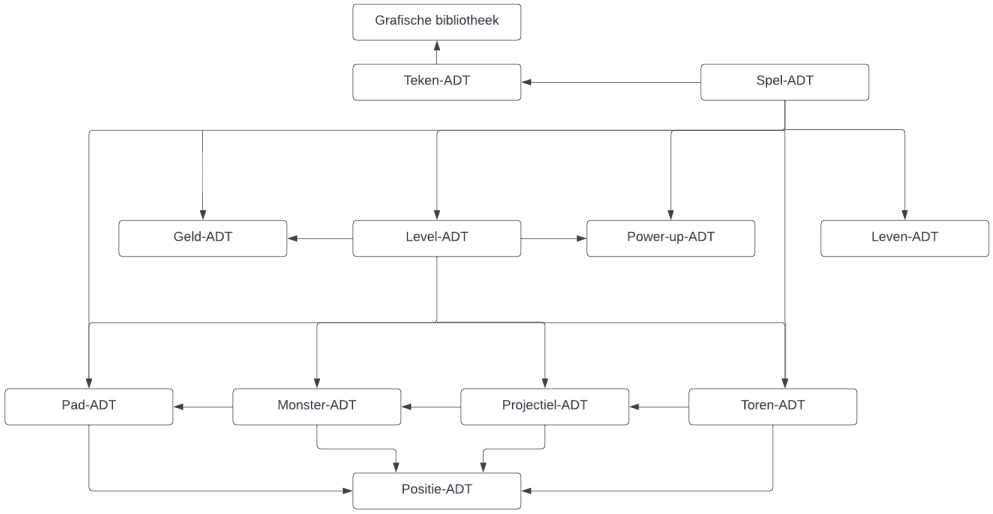
Tabel 12: Mijn gevolgd schema tijdens het implementeren van de functionaliteiten

5 Voorstudie

In dit deel zal teruggekeken worden naar de voorstudie die men moest maken voor het begin van het uitbreiden van het spel. Meer bepaald zullen het afhankelijkheidsdiagram en de planning van de voorstudie vergeleken worden met de nieuwe afhankelijkheidsdiagram en het tijdsschema die werkelijk gevolgd geweest is.

5.1 Afhankelijkheidsdiagram

Terugkijkend naar het afhankelijksheidsdiagram van de voorstudie (Figuur 2) kan men zien dat de huidige afhankelijksheidsdiagram (AD) complexer in elkaar zit dan de vorige. Eerst en vooral, zijn er in de AD van de voorstudie minder afhankelijkheden dan nu. Er waren een aantal afhankelijkheden die misten. Meer bepaald, beweerde het AD van de voorstudie dat het Teken ADT onafhankelijk was van alle spelementen, wat niet correct was. Deze niet bestaande afhankelijkheden tussen een aantal ADTs werden toegevoegd aan het huidige AD. Daarbovenop waren er nog vele andere afhankelijkheden die misten, bijvoorbeeld tussen het Power-up en Pad ADT of tussen het Levens en Monster ADT. Deze werden ook toegevoegd.



Figuur 2: Afhankelijkheidsdiagram opgemaakt voor voorstudie

5.2 Planning

Als we de planning van de voorstudie (Figuur 3) vergelijken met de echte tijdschema die gevolgd werd (Tabel 12) dan zien we dat we voor het maken van de functionaliteiten van de tweede fase, een lichtjes verschillende aanpak genomen hebben. In plaats van eerst onze huidige Monster, Projectiel en Toren ADT uit te breiden werden eerst een aantal nieuwe ADTs toegevoegd (geld en levens). Het verdere verloop van het project was gelijkaardig. Meer bepaald, het Monster ADT uitbreiden, het Toren ADT uitbreiden en de aanmaak van het Power-up ADT.

Bij het aanmaken van de verwachte functionaliteiten heeft alles de verwachte tijd ingenomen, behalve voor het implementeren van de verschillende soorten projectielen en torens in de spellogica. Het implementeren van het bomwerp en de netprojectielen heeft langer geduurd dan verwacht vermits men gemakkelijk compile-time of run-time errors kon veroorzaken

Week	Actie
Week 27	Op basis van feedback van fase 1 een aantal ADTs verbeteren.
Week 28	Aanpassen Monster ADT en Teken ADT om meerdere soorten monsters te kunnen voorstellen en op het scherm te tekenen.
Week 29	Aanpassen Toren ADT en Projectiel ADT om meerdere soorten torens te kunnen voorstellen die een specifieke projectiel afvuren.
Week 30	Het Teken ADT aanpassen om zo de verschillende torens en hun bijbehorende projectielen op het scherm te tekenen.
Week 31	Aanmaken van een Leven ADT om zo levens te kunnen voorstellen, en daarbij het Teken ADT aanpassen om die levens op het scherm te kunnen tekenen.
Week 32	Aanmaken van een Geld ADT om zo geld voor te stellen alsook het Teken ADT om geld op het scherm te kunnen zetten.
Week 33	Aanmaak van een Power-ups ADT om Power-ups te kunnen voor stellen en daarbovenop het Teken ADT aanpassen om deze Power-ups op het scherm te plaatsen
Week 34	Aanpassen Monster ADT om monsters power-ups te kunnen doen droppen, geld afgeven indien ze sterven en levens van de speler doen zakken indien ze het einde van het pad bereiken. Daarbovenop ook het Toren ADT aanpassen zodat torens enkel gebouwd kunnen worden als er voldoende geld is.
Week 35	Aanpassen van het Level ADT om verschillende rondes te kunnen voorstellen en geld af te geven bij succesvolle rondes. Evenzeer zal het Spel ADT aangepast worden om uiteindelijk alles samen te brengen tot een volwaardig spel.
Week 36	Code nakijken en opmaken van het verslag
Week 37	Indienen code en verslag

Figuur 3: Planning opgemaakt voor voorstudie

6 Spel uitleg

Om mijn tower defence game op te starten moet men het bestand met naam *Spel.rkt* openen en deze runnen. Wanneer men dit gedaan heeft zal de spelwereld aangemaakt worden en kan men beginnen spelen door op de "return" toets te toesten, het spelscherm opent dan. Het scherm bestaat uit 2 delen, een spelwereld deel en een menu deel. Om torens te plaatsen in de spelwereld moet men op de gewenste toren klikken in de menu en dan op de gewenste plaats in de spelwereld klikken om de toren werkelijk te plaatsen. Indien men een power-up wil kopen, moet men gewoon drukken op de gewenste power-up in de menu. Om deze te activeren moet men respectievelijk op de toetsen "T" en "B" drukken om een tank of een bommenregen te activeren . Als men dan de ronde wil starten moet met op de "spatie" toets klikken. Indien de ronde gedaan is, kan men de ronde herstarten door nogmaals op "spatie" toets te drukken. Tenslotte, indien men de ronde wil beeindigen moet geklikt worden op de ëscape" toets (dit is een cheat toets). Bij een game-over of game-win kan men het spel opnieuw starten door op de "return"toets te drukken.