Commander Keen

Project ICT 3

Lorenz Put

1. Verhaal

Commander Keen, de nieuwe rekruut van het Luvaniaanse ruimtevaartprogramma, werd erop uitgestuurd om een verkenningsaanval uit te voeren op de vijandige cruiser van de GreenSouls, de buitenaardse bewoners van de planeet Green. 5 uur en 27 minuten later kwam de cruiser in zicht, een reusachtige cruiser met bijna ondoordringbare romp van metaal lag recht voor hem. De cruiser moest verkend worden voordat een aanval ingezet kon worden, maar slechts enkele ogenblikken nadat hij geland was, liep het echter helemaal mis. Het ruimteschip waarop hij geland was, vertrok terug naar zijn thuisplaneet Green en doordat zijn schip ontdekt was, kon hij niet meer wegvliegen. Bij de aankomst werd het vijandelijk verkenningsvliegtuig in beslag genomen en opgeborgen op een uiterst geheime plaats. Het duurde dagen voor Commander Keen te weten kwam waar ze zijn vliegtuig naartoe gebracht hadden en er is maar 1 weg om dat vliegtuig terug te bemachtigen, maar het is er één vol met gevaren, zware beveiligingen en daarbij had hij nog maar enkele minuten voor zijn zuurstof op zou raken. Ben jij in staat commander Keen tijdig weg te krijgen van de planeet Green en hem gezond en wel terug op aarde te krijgen?

2. Analyse

2.1 Inleiding

Toen we de opdracht kregen om een platform game te maken, was mijn eerste idee om een kopie te maken van het alom bekende Tarzan. Op het eerste zicht leek mij dit een goed idee omdat het vroeger een zeer populaire game was. Maar na verder opzoekwerk bleek het zeer moeilijk om de juiste afbeeldingen te vinden. De spritesheets die ik vond, waren veel te klein om te gebruiken. Daardoor zou de game van slechte kwaliteit zijn. Na nog een paar tussenfases zoals bijvoorbeeld een straatvechter, was een vriend van mij op een dag commander Keen aan het spelen. Zo is het idee gekomen om een low-level versie te maken van de commander Keen : Aliens ate my baby sister.

In low-level verstaan we dat de held naar links en rechts kan bewegen. Ook moet de held kunnnen springen en vallen. Verder hebben we nog vijanden die in het level kunnen bewegen. Ook beperken we ons tot het maken van 2 levels die bestaan uit een verschillende achtergrond, verschillende soorten blokjes waar de vijand en de held op kunnen lopen, items die punten op leveren, een extra leven en een klok die extra tijd oplevert.

De bedoeling van het spel is dat commander Keen binnen een bepaalde tijd door de deur loopt die naar het volgende level of het einde van het spel leidt. Deze deur bevindt zich uiteraard op het einde van het level en tussen hem en de deur lopen veel vijanden die hem zullen proberen doden. Commander Keen heeft om te beginnen maar 1 leven. Dit kan echter uitgebreid worden door het oprapen van hartjes. Elk hartje zal hem 1 extra leven geven. Hij kan echter maar maximum 3 levens hebben.

Naast de vijanden bevindt er zich nog een moeilijkheid in het spel. Er is namelijk een tijd vastgesteld waarop hij aan de deur moet geraken. Deze tijd staat vastgesteld op een minuut. In het spel bevinden zich ook klokjes, die indien ze opgeraapt worden 10 seconden extra opleveren.

De bedoeling van het spel is om in de opgegeven tijd het level te voltooien. Indien mogelijk door zoveel mogelijk items te nemen.

2.2 Gebruikersaanwijzing

 = naar links lopen

 = naar rechts lopen.

= springen

En voor als het nodig zou zijn is er ook een cheatcode ingebouwd in het spel

 = onsterfelijk. De levens gaan nog wel verminderen maar de commander gaat niet meer dood. Indien dit geactiveerd is, kan dit niet meer uitgeschakeld worden.

2.3 Beschrijving

2.3.1 Algemeen

(Een grotere versie van de afbeelding vindt u in de bijlagen.)

Dit diagramma geeft de algemene structuur van de game weer. In totaal zijn er 16 aanwezig in de game. De meeste van deze zijn met elkaar verbonden door middel van een overervingsstructuur. Deze inhoud, structuur en functies van de gaan één voor één aan bod komen in dit document.

2.3.2 Gameobject

We beginnen met de klasse gameobject. Door deze klasse abstract te maken, kunnen er nog wel instanties gecreëerd worden maar het is niet de bedoeling dit te doen. In het diagram zien we verschillende klasses die overerven van GameObject. Bijgevolg zijn die klasses dus ook allemaal GameObjecten. Deze klasse geeft een algemeen beeld weer van de variabelen, properties en methodes die de overervende klasses zullen bevatten. Er zijn 2 soorten van variabelen en methoden: namelijk degenen die geïmplementeerd moeten worden en degenen die gebruikt kunnen worden. De variabelen of methoden die de overervende moet bevatten zijn de ShowPartImage, Image variabelen en de Update methode.

- ShowPartImage is een rechthoek die bepaalt welk gedeelte van de spritesheet er op scherm geblit zal worden. Door deze rechthoek steeds te verplaatsen ontstaat de illusie van beweging.
- Image is een surface object die de spritesheet in het programma inlaadt.

- De Update methode wordt bij elke tick aangeroepen en dus ook de code die het bevat.

De variabelen of methoden die overervende kunnen bevatten zijn de Position, Leftcollisionrectangle, Rightcollisionrectangle, Uppercollisionrectangle, Lowercollisionrectangle en de Draw methode.

- Position is een point variabele. Een point variabele moet een x- en een y-coordinaat bevatten. De Positionvariabele bepaalt het punt linksboven van de afbeelding m.a.w. dus waar de afbeelding op het scherm geplaatst wordt.

- De left, right, upper, lower collisionrectangle zijn alle vier variabelen van het type rechthoek. Deze rechthoeken worden rond afbeeldingen geplaatst. De collisionrectangles bevatten een positie, een breedte en een hoogte. Deze waarden zijn zo ingesteld dat ze het object mooi omkaderen. Deze collisionrectangles worden gebruik om “botsingen” met andere rechthoeken te detecteren. Indien ze elkaar raken, wordt er bepaalde code uitgevoerd (zie verder).

- De draw methode bevat in de gameobject klasse alleen de video.Blit methode. Indien de draw methode niet geimplementeerd wordt in overvende , zal alleen de afbeelding op het scherm gezet worden. Echter indien methode overschreven wordt, kan er extra functionaliteit toegevoegd worden. Dit wordt bijvoorbeeld gedaan om de collisionrectangles zichtbaar te kunnen maken op het scherm.

De belangrijkste klasse die overerft van gameobject is de Hero klasse. De klasse bevat v-erschillende methoden : constructor, overloaded constructor, Events_KeyboardUp, Events_KeyboardDown, Update, Draw, Movements, Die. Deze methoden en hun functionaliteiten worden één voor één besproken.

- De hero klasse begint met een constructor en een overloaded constructor. We gebruiken de overloaded constructor, maar vanaf het moment dat we zelf een constructor gebruiken vult c# dit niet automatisch aan met een basis constructor wat hiervoor wel gebeurde. Dit betekent dus dat we die zelf nog moeten schrijven. De overloaded constructor gebruiken we om de variabelen x en y toe te kunnen wijzen aan de Position variabele. In deze constructor is exception handeling toegevoegd om het programma niet te laten crashen als er bijvoorbeeld een afbeelding niet correct ingeladen kan worden. In deze exception handeling worden de image, showpartimage, collision rechthoeken en de Positionvariabelen overschreven (uitleg zie hierboven).

- De Events_KeyBoardUp en Events_KeyBoardDown methoden worden uitgevoerd als de gebruiker een input geeft via het toetsenbord. Zo worden bij de KeyBoardUp een aantal bools op vals gezet waardoor de held niet meer naar links of rechts kan bewegen of kan springen. Bij de KeyBoardDown gebeurd juist het omgekeerde. Hier worden bools op true gezet waardoor de held wel naar links of rechts kan bewegen. Enkel bij de bool voor het springen staat er wat meer code. Deze code zorgt ervoor dat de held niet kan springen terwijl hij al aan het springen of vallen is.

- In de Update methode wordt de movements methode aangeroepen. Deze wordt later verder uitgelegd. Verder bevat de Update methode de reactie op de veranderende jump bool in de Events_KeyBoardUp en Events_KeyBoardDown methoden. Zo wordt bijvoorbeeld bij het springen de y-positie aangepast met een bepaalde waarde. Als laatste bevat de Update methode de code om de collision rechthoeken de held te laten volgen en niet ter plaatse te blijven staan. Dit wordt gedaan door de x- en y-coordinaten van het punt linksboven gelijk te stellen aan dat punt van de afbeelding.

- In deze klasse wordt de Draw methode overschreven. De code base.Draw(video) wil zeggen dat de code van de Draw klasse in de klasse gameobject ook geïmplementeerd wordt. Verder worden de boxen gelijkgesteld aan de collision rechthoeken, wordt er een breedte en hoogte meegegeven en wordt de box getekend in een rode kleur.

- In de movements methode wordt er ook gereageerd op de veranderende bool uit de Events_KeyBoardUp en Events_KeyBoardDown methoden. Echter deze keer gaat het om het naar links en rechts bewegen en niet om het springen. Ook wordt in deze methode het beginpunt van de ShowPartImage rechthoek veranderd zodat we de illusie krijgen dat de afbeelding beweegt.

- Als laatste hebben we de Die methode. Deze methode checkt elke keer of het aantal levens van de held niet op nul staat. Indien dit het geval is, wordt de held uit de lijst met gameobjecten verwijdert en bijgevolg dus ook niet meer getekend.

Naast de Hero klasse is er nog een belangrijke klasse die overerft van gameobject nl. die van de vijanden. De klasse van de vijanden heeft voor een deel dezelfde methode als de Hero klasse. Zo bevat deze bijvoorbeeld een Update methode waar de collision rechthoeken ge-updatet worden en een Draw methode waar de afbeelding en de rechthoeken op het scherm getekend worden. Naast deze gelijkenissen zijn er ook verschillen. Zo heeft de overloaded constructor geen 2 maar 4 argumenten. Het bevat nog altijd een x – en y – positie, maar er is een extra moveleft en moveright bool bij gekomen. Deze worden gebruikt om te kunnen kiezen naar welke kant de vijand loopt wanneer hij voor het eerst geïnitialiseerd wordt. Een ander verschil is dat vijand klasse een EnemyMovement methode bevat. Deze methode doet verschillende dingen.

- Ten eerste zorgt deze ervoor dat het beginpunt van de ShowPartImage rechthoek verandert zodat de illusie gecreëerd wordt dat de afbeelding beweegt.

- Ten tweede wordt de x-positie ge-updatet a.d.h.v. het feit of de held naar links of rechts loopt.

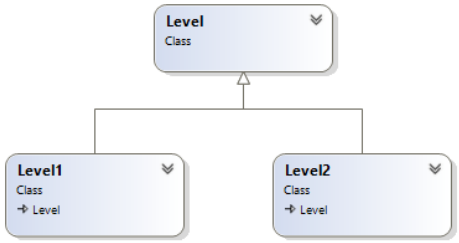
- Als laatste wordt er door de lijsten MetalObjects en StoneObjects gelopen en wordt er gecheckt of de rechthoeken van de MetalObjects en de StoneObjects elkaar niet raken. Indien de vijanden links of rechts een blokje raken, keren ze om en als ze langs de onderkant geen blokje raken, vallen ze naar beneden.

Naast hero en vijand klasses hebben we nog verschillende andere die overerven van gameobject. Deze bevatten bijna allemaal dezelfde code. Enkel de namen van de

constructors en de bestandsnaam gedinfinieerd in het image object verschillen. Om deze reden beperk ik mij tot het uitleggen van 1 klasse namelijk die van het Item.

- De item klasse begint met een constructor en een overloaded constructor. Dit wordt zo gedaan om dezelfde redenen als in de hero klasse.
- De Update methode zorgt ervoor dat de collision rechthoeken mee bewegen over het scherm en niet ter plaatse blijven staan. Meer uitleg zie hero klasse.
- Er worden ook nog 4 boxen aangemaakt om op het scherm te kunnen tekenen.
- In deze klasse wordt ook de Draw methode overschreven. Dit wordt op exact dezelfde manier gedaan als in de Hero klasse.

2.3.3 Level



De structuur tussen level, level1 en level2 is gebaseerd op het principe dat ook toegepast wordt bij de gamobjecten namelijk overerving en abstractie. Level is abstract gemaakt en dus mogen er geen instanties van gemaakt worden. De level klasse wordt gebruikt om het wisselen tussen de level1 en level2 klasse in de manager te vergemakkelijken. Om deze reden bevat de level klasse dus verschillende variabelen en methoden die in de level1 en 2 klasses ook zullen terugkomen.

- In het begin van de level klasse worden eerst een aantal variabelen en lijsten aangemaakt. Deze lijsten worden verder gebruikt in de level klasse zelf maar ook in level1 en 2.
- De Createworld is gedeclareerd als virtueel en doorloopt de TileArray. Tijdens dit doorlopen creëert de methode verschillende objecten a.d.h.v. het cijfer dat zich in de TileArray bevindt en worden de gecreëerde objecten respectievelijk aan de juiste lijst toegevoegd.
- Naast de Createworld functie is er ook de Draw functie. Deze doorloopt alle lijsten (buiten die van MetalObjects en StoneObjects) en tekent de respectievelijke objecten op het scherm door hun Draw functie aan te roepen.
- Verder is er ook nog de MoveLevel methode maar deze wordt volledig geïmplementeerd in de level1 en 2 .
- Als laatste bevat de level klasse ook een functie CheckCollisionsWithBackground. Ook hier wordt weer door de lijsten gelopen met een foreach en wordt er gecheckt of de held niet botst met een van de objecten.

- Wanneer hij tegen een metaal object loopt, kan hij niet meer naar links of rechts bewegen en wanneer hij op een blokje springt, zal hij er op blijven staan. Dit geldt ook voor de steen objecten.
- Wanneer hij tegen een leven loopt, wordt het leven uit de lijst verwijdert en wordt er een leventje meer getoond. Dit geldt ook voor de items en klokjes, alleen wordt de score vermeerderd of worden er tien seconden bijgeteld bij de tijd in plaats van het tonen van een extra leven.
- Wanneer hij tegen de deur naar het volgende level loopt, wordt er een bool op true gezet waardoor het volgende level getekend zal worden.

De level1 klasse erft over van level en bevat bijgevolg de variabelen en methode van de level klasse.

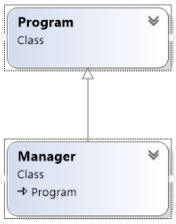
- De constructor bestaat uit de TileArray, het instantiëren van de object lijsten en het aanroepen van de Createworld methode van de level klasse. De TileArray bevat eigenlijk alle informatie nodig om het level te kunnen tekenen.
- Verder wordt de Draw methode overschreven. Deze implementeert de Draw methode van de level klasse en voegt zelf nog een extra lijst toe die getekend moet worden.
- Ook de MoveLevel methode wordt overschreven. Deze methode bestaat eigenlijk uit 2 verschillende delen. Het eerste deel wordt gebruikt wanneer de held naar rechts loopt, telkens dit gebeurt, wordt er een teller verhoogd en worden de x-posities van de objecten vermindert. In het tweede deel wordt de teller verlaagt en worden de x – posities van de objecten verhoogd. Dit gebeurt tot de teller op nul staat. Indien dit gebeurt, wordt het level niet meer verschoven.
- Als laatste wordt de CheckCollisionsWithBackground methode van level overschreven. Deze implementeert ook weer de CheckCollisionsWithBackground functie van de level klasse maar voegt een extra object toe om te checken of er “botsingen” voor komen.

De level2 klasse erft net zoals level1 over van level en implementeert ook dezelfde methoden en variabelen. De level2 klasse verschilt niet zoveel van de level1 klasse buiten het feit dat er in de Draw methode en in de CheckCollisionsWithBackground methode een ander object wordt toegevoegd en dat de IntTyleArray verschillend is.

2.3.4 HeroLives

HeroLives wordt gebruikt om het aantal levens van de held op het scherm te kunnen tonen. De klasse erft niet over van gameobject. Dit omdat het aantal levens ook niet echt als een gameobject beschouwd kan worden. De klasse implementeert echter wel ongeveer dezelfde structuur. Zo bevat de klasse een constructor met daarin exception handeling voor het initialiseren van de afbeelding, punt en ShowPartImage rechthoek. Verder hebben we ook een Draw methode. In deze methode wordt 1,2 of 3 levens op het scherm getoond a.d.h.v. het aantal levens van de held. Dit wordt gedaan door de ShowPartImage rechthoek te vergroten of verkleinen.

2.3.5 Manager en program



In deze afbeelding zien we dat de manager klasse overerft van de program klasse. Het enige dat gebeurt in de program klasse is het aanmaken van een manager object in de main. Door deze lijn code kunnen de verdere aangemaakt en uitgevoerd worden.

- De manager klasse bevat verschillende variabelen die in het begin aangemaakt of aangemaakt en geïnstantieerd worden. Daarna hebben we de constructor van de manager. Ook hier wordt weer exception handeling toegepast. In deze exception handeling wordt de sdl app aangemaakt en worden verschillende afbeeldingen ingeladen. Na de exception handeling worden 2 font objecten geïntialiseerd. Dit wordt gebruikt om tekst op scherm te kunnen tonen.

- **Verder hebben we ook nog polymorfisme dat wordt toegepast door een level object te instantiëren met de level1 klasse. Hierdoor kan het level object gebruik maken van alle methodes en variabelen in de level1 klasse. Dit vermijdt het schrijven veel dubbele code.**

- Ook wordt er een object aangemaakt van de HeroLives klasse, maken we een lijst aan van vijanden, wordt er een timer object met een tijd van 1 seconde plus callback methode aangemaakt, zetten we de hoofding van de sdl app op Commander Keen, en wordt er een Events_Tick en een Events_KeyBooardUp met callback methodes aangemaakt. Als laatste element in de constructor wordt Events.Run aangeroepen. Dit start eigenlijk de applicatie.

- De callback methode van de timer wordt uitgevoerd van zodra de timer op 0 staat. In dit geval wordt de tijd verminderd met 1.

- In de Events_KeyBoardUp wordt er gecheckt of bepaalde knoppen ingedrukt zijn. Indien dit het geval is, wordt er een bepaalde code uitgevoerd zoals het sluiten van de applicatie of het op true zetten van een bool.

- In de callback methode van de tick wordt er eerst gekeken of de held niet dood is. Indien dit niet het geval is, wordt er gekeken of de gebruiker op de s - toets geduwd heeft.

    - Indien dit het geval is, worden de methodes CheckCollisionsWithEnemy, level.held.Die, level.MoveLevel, level.CheckCollisionsHeldWithBackground; UpdateBackGroundTiles en TimeCheck uitgvoerd. Ook wordt er een achtergrond op scherm getoond a.d.h.v. het feit of de gebruiker in level 1 of level 2 aan het spelen is. Indien het level 2 is, worden eerst de lijsten van de gameobjecten gecleard en opnieuw polymorfisme uitgevoerd om het level object toe te wijzen aan de level 2 klasse. Na dit worden level, levens, score en tijd op het scherm getoond.

    - Indien dit niet het geval is, wordt het startscherm getoond.

- Indien de held dood is, wordt het gameover getoond en heeft de gebruiker de optie om opnieuw te starten of af te sluiten. Indien er op de r- toets geduwd wordt, wordt de methode reinitialize aangeroepen en kan het spel terug opnieuw starten.

- Als laatste in de callback van het tick event wordt de sdl app ge-updatet. Dit is zeer belangrijk. Indien dit niet gebeurt, verschijnt er een lege sdl app.

In de functie CheckCollisionsWithEnemy() wordt er door de lijst van vijanden gelopen en wordt er gecontroleerd of er geen “botsingen” zijn tussen held en vijand. Indien de held langs de zijkanten geraakt wordt, vermindert zijn aantal levens met 1. Indien de held op de vijand springt, wordt de vijand gedood en wordt hij uit de lijst verwijderd, waardoor hij niet meer op het scherm verschijnt.

De functie UpdateBackGroundTiles roept van elk object in de levels de update functie aan. Deze functie wordt eigenlijk pas nuttig als het level moet bewegen. Indien dit niet gebeurt, verschuiven de afbeeldingen wel maar hun collisionrectangles niet en gaat de collision detectie volledig verloren.

De functie ReIntialize zet alle waarden weer naar de start waarden. Zo worden alle lijsten gecleard, wordt de held terug tot leven gewekt en krijgt hij terug 1 levens, wordt level 1 terug opnieuw gegenereerd en wordt de tijd terug op 60 seconden gezet.

Als laatste methode in de manager hebben we TimeCheck. Deze methode checkt 2 dingen :

- Of de tijd niet op 0 staat. Indien dit het geval is, gaat de held dood.

- Of de held een klokje heeft opgeraapt.Indien dit het geval is, krijgt hij 10 extra tijd.

3. Reflectie

Toen ik de opdracht kreeg, wist ik totaal niet hoe ik er aan moest beginnen. De opdracht leek mij onbegonnnen werk omdat mijn kennis van programmeren vrij beperkt was. Ik moest veel opzoekwerk verrichten, maar dat heeft uiteindelijk wel geloond want dingen die je zelf uitzoekt, onthoud je beter. Moest ik nu een gelijkaardige opdracht krijgen, zou ik nog altijd wel wat tijd nodig hebben, maar zou het al veel vlotter gaan omdat bepaalde denkwijzen die ik heb aangeleerd tijdens dit project, mij nu vast een sneller en beter inzicht zouden geven. Ondanks het vele werk vond ik het toch een boeiende opdracht.

4. Bijlagen