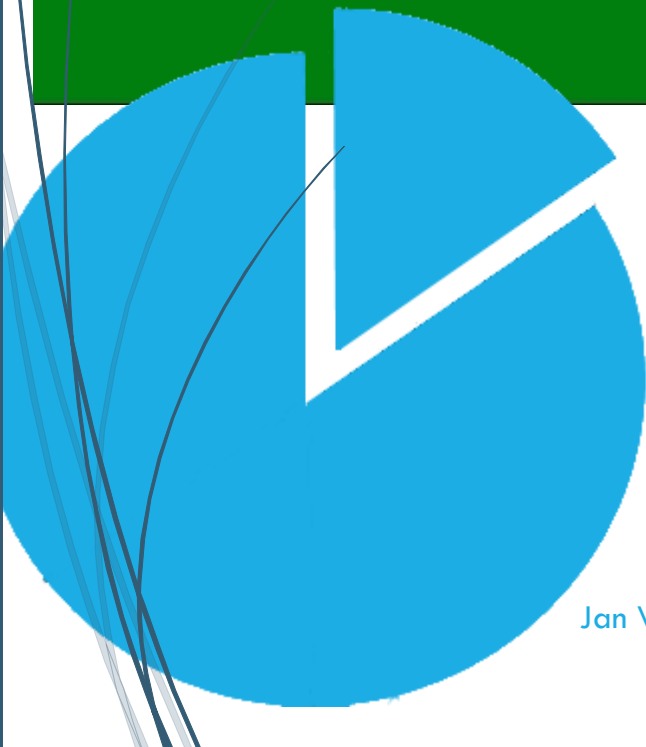
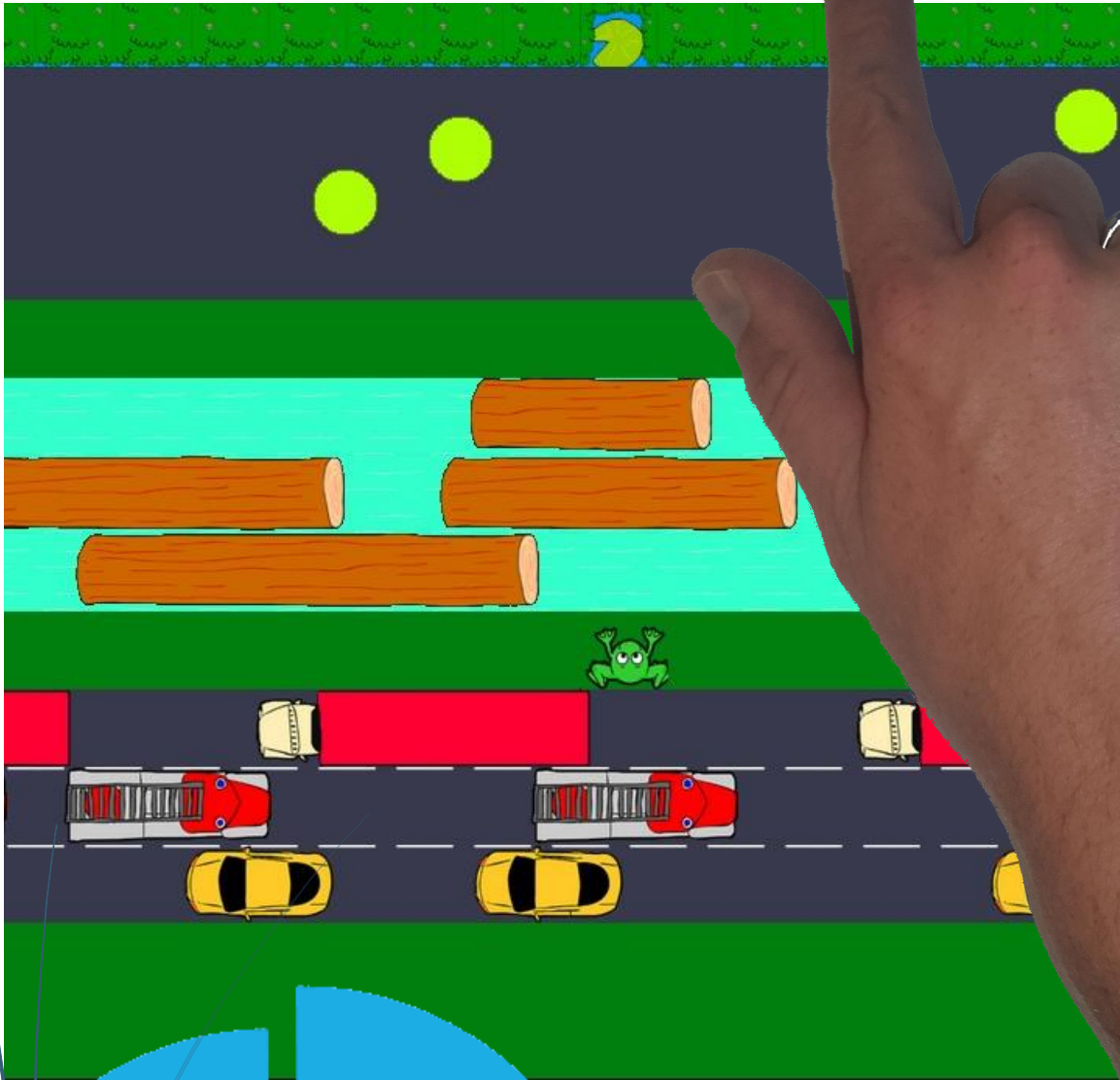


10-5-2019


Technisch Ontwerp

Frogger spel



Jan Willem Lenting, Christiaan Sommer en Anne Torn Broers

Technisch Ontwerp

Datum 10-5-2019	Versie 2.0
Jan Willem Lenting	608047
Christiaan Sommer	622453
Anne Torn Broers	623555
Groep 5	
HBO-ICT (Deeltijd)	2e Semester 2e periode - OOP
Docent: Dhr. Cornelissen	Klas: SEB 1A
	

VERSIEBEHEER

Versie	Persoon	Toepassingen	Datum
0.1	Anne Torn Broers	Document opgezet, gekeken naar de voorwaarden waar document aan moet voldoen, hoofdstukindeling gemaakt, tekst toegevoegd aan de hoofdstukken.	5-04-2019
0.2	Christiaan Sommer	Document verder uitgewerkt, meer inhoudelijke informatie toegevoegd. Diagrammen gemaakt.	10-04-2019
0.3	Christiaan Sommer	Document verder uitgewerkt en overgekeken.	12-04-2019
0.4	Jan Willem Lenting	Document verder bekeken, aangevuld, overgekeken op juistheid.	16-04-2019
0.5	Jan Willem Lenting, Anne Torn Broers	Document doorgelopen op ontbrekende punten. Hoofdpijnen in document toegevoegd welke nog moeten worden uitgewerkt.	20-04-2019
0.6	Jan Willem Lenting	Document verder in de hoofdpijnen uitgewerkt.	23-04-2019
0.7	Jan Willem Lenting	Document verder bekeken, overgekeken op juistheid.	24-04-2019
0.8	Christiaan Sommer	Controle beschreven klassen	25-04-2019
0.9	Jan Willem Lenting	Laatste aanpassingen n.a.v. gesprek met Christiaan Sommer	26-04-2019
1.0	Anne Torn Broers	Document conceptversie gemaakt.	26-04-2019
1.1	Christiaan Sommer	Document overgekeken gecontroleerd op juistheid, klassenbeschrijvingen en conclusie uitgebreid.	27-04-2019
2.0	Christiaan Sommer, Anne Torn Broers	Definitieve versie gemaakt. Ontwerp op kleine puntjes aangevuld en nieuw klassendiagram gemaakt.	9-05-2019

INHOUDSOPGAVE

VERSIEBEHEER	2
INHOUDSOPGAVE.....	3
2. INLEIDING.....	4
3. AANPAK EN UITGANGSPUNTEN.....	5
3.1 AANPAK	5
3.1.1 KLASSENDIAGRAM.....	5
3.1.2 HIGHSCORE (NICE TO HAVE).....	5
3.1.3 HOOFDMENU	5
3.1.4 GAMEOBJECTEN.....	5
3.1.5 SECTIES (WEG/RIVIER/BAL).....	5
3.2 UITGANGSPUNTEN.....	5
4. KLASSENBSCHRIJVING	6
4.1 FROGGER KLASSE.....	6
4.2 PLAYER KLASSE.....	6
4.3 MENUMANAGER KLASSE	6
4.4 MAINMENU KLASSE.....	6
4.5 GAMEMENU KLASSE.....	6
4.6 GAMEOVERMENU KLASSE.....	6
4.7 HIGHSCOREMANAGER KLASSE.....	6
4.8 MAP KLASSE.....	7
4.9 FINISHSECTION KLASSE.....	7
4.10 SECTION KLASSE.....	7
4.11 ROAD/RIVER/BALL SECTION KLASSE.....	7
4.12 ROAD/RIVER OBJECTS KLASSE.....	7
4.13 BUS/CAR/TRUCK/FASTCAR ROADOBJECTS KLASSE.....	7
4.14 TREE/CROCODILE RIVEROBSTACLES KLASSE	7
4.15 TREESIZE ENUM.....	7
4.16 BALL OBJECT KLASSE.....	8
5. KLASSENDIAGRAM	9
6. CONCLUSIE.....	11
7. BRONVERMELDING.....	11

2. INLEIDING

Dit document is een aanvulling op het Functioneel Ontwerp. Het Functioneel Ontwerp kan worden gevonden in de bijlage onder de naam Functioneel Ontwerp Frogger versie 2.0.pdf. In dit document gaan we dieper in op het technische gedeelte van het ontwerp. We gaan hier nadenken over onder andere de klassen en de aanpak. Als basis voor de Technisch Ontwerp verwijzen we naar het Functioneel Ontwerp, het Technisch Ontwerp is een uitbreiding hierop. Begrippen in het Technisch Ontwerp worden uitgelegd in het Functioneel Ontwerp.

Als kernvraag willen we neerzetten; hoe gaan we de game Frogger opzetten en welke klassen en variabelen hebben we nodig om tot een werkende game te komen?

Hierna zal het Technisch ontwerp in een paar grove penningen worden beschreven in deze inleiding.

De kern van de hoofdstukken is voor het overzicht puntsgewijs beschreven:

- *In Hoofdstuk 3* gaat over de aanpak en de uitgangspunten waarvandaan we werken;
- *In Hoofdstuk 4* is de klassenbeschrijving, deze is zo veel mogelijk up to date maar kan afwijken;
- *In Hoofdstuk 5* is het klassendiagram zelf;

Uiteraard zijn hiervoor de Inleiding en hierna de Conclusie beschreven.

3. AANPAK EN UITGANGSPUNTEN

3.1 AANPAK

Omdat het maken van een game een proces is zal niet alles wat in de loop van het maken van de game bedacht wordt, hierin beschrijven. Dit dient dus vooral als startup van het programmeren, hierdoor kan het zijn dat bepaalde classes ontbreken. Concreet, het document wordt achteraf niet gewijzigd tenzij met een kleine aanpassing het geheel weer volledig is.

3.1.1 KLASSENDIAGRAM

Eerst wordt er een klassendiagram gemaakt voor een goed totaaloverzicht. Hierin komen de associaties en de interfaces en klassen met hun methoden en variabelen in voor. Aan de hand van deze opzet wordt de applicatie opgebouwd.

3.1.2 HIGHSCORE (NICE TO HAVE)

De score zullen we opslaan in een .txt bestand welke door de game ingelezen zal worden aan het begin van het spel. De methode `GetHighScores()` zal zorg dragen voor het laden hiervan en `SetHighScores(sting)` voor het opslaan

3.1.3 HOOFDMENU

Er is bewust gekozen voor een simpel start menu met alleen de keuze Start en Afsluiten. Een extra uitbreiding zou kunnen naar gelang de einddatum nog niet in zicht is en de game naar behoren werkt. Er kunnen knoppen toe worden gevoegd om de Higscore in te zien en de mogelijkheid te geven om een naam in te geven.

3.1.4 GAMEOBJECTEN

Elk game onderdeel is van het type `SpriteObject` (zie het Klassediagram in Figuur 1) en heeft door middel van overerving eigen eigenschappen. Zo is per object bijvoorbeeld een eigen snelheid of path te maken die het object zal afleggen.

3.1.5 SECTIES (WEG/RIVIER/BAL)

Elke sectie van het spel heeft zijn eigen instellingen, zoals de positie/limieten en breedte hiervan. Een spawner (zie het Klassediagram in Figuur 1) zal hier dan de juiste objecten in aanmaken.

3.2 UITGANGSPUNTEN

Niet alles wat in het functioneel ontwerp is bedacht zal ten uitvoer worden gebracht, dit vooral in verband met de tijd en geen te behalen punten hiervoor volgens het beoordelingsformulier. Dit is dan wel aangegeven in het ontwerp als een nice-to-have of als extra werk. Een voorbeeld hiervan is highscore maar bijvoorbeeld ook sound effecten en achtergrond muziek wordt gezien als extra werk.

4. KLASSENBSCHRIJVING

Hieronder zullen in de volgende paragrafen de klassen welke nodig zijn, kort worden beschreven. Sommige punten die worden gezien als extra werk zullen toch worden opgenomen in de klassen, aangezien het dan gelijk duidelijk is waar deze klasse had moeten komen te staan.

4.1 FROGGER KLASSE

Dit is de hoofdklasse van het spel, deze zal alle onderdelen van het spel (bijvoorbeeld hoofmenu en het speelgebied) starten.

4.2 PLAYER KLASSE

Deze klasse zal bijhouden wat er gebeurt door de user (player). Bijvoorbeeld als er op de knop pijltje naar links of rechts gedrukt wordt. Tevens zal hier de logica plaatsvinden met betrekking tot botsingen met de gameobjecten.

4.3 MENUMANAGER KLASSE

Deze klasse zorgt voor het laden van de juiste menu klasse en zal dan bijvoorbeeld het MainMenu object inladen. Maar ook eventuele extra menu's zoals het menu voor de game zelf.

4.4 MAINMENU KLASSE

Deze klasse toont het daadwerkelijke menuscherm en zal ook de highscores inladen via de highscoremanager klasse. Dit laatste wordt overigens gezien als extra werk.

4.5 GAMEMENU KLASSE

Deze klasse toont alle info die de speler nodig heeft zoals de score of de hoeveelheid levens de speler nog over heeft.

4.6 GAMEOVERMENU KLASSE

Deze klasse toont het menu als de speler geen levens meer heeft. Hier wordt de score de die speler heeft gekregen en kan hij zijn naam invullen.

4.7 HIGHSCOREMANAGER KLASSE

Deze klasse zal de highscores bijhouden en kan deze ook inladen en eventueel opslaan. Dit wordt overigens gezien als extra werk.

4.8 MAP KLASSE

Deze klasse zal definiëren hoe het spel eruit zal zien en bijvoorbeeld ook zorg dragen voor het laden van de juiste sectie klassen.

4.9 FINISHSECTION KLASSE

Deze klasse zal bijhouden hoeveel kikkers in het finish gedeelte staan en daarmee ook controleren of het spel gewonnen is en de juiste dingen aanroepen hiervoor.

4.10 SECTION KLASSE

Dit is de hoofd klasse voor alle secties, hierin zit bijvoorbeeld de gezamenlijke functionaliteit die elk sectieobject gebruikt zoals spawn entity en getbounds voor de grenzen van het sectieveld (zie het Klassediagram in Figuur 1 voor de definities).

De spawner in de section klasse creert game objecten die een vast path volgen. Als voorbeeld, het verkeer, in de weg sectie maar ook de boomstammen in de rivier sectie. In het object zelf is gedefinieerd hoe het obstakel of object zal reageren.

4.11 ROAD/RIVER/BALL SECTION KLASSE

Dit is de specifieke sectie klasse voor de sectie objecten, hierdoor zijn de secties en hun objecten makkelijk uit elkaar te halen en ook makkelijk te killen als er bijvoorbeeld naar een nieuwe sectie gegaan wordt.

4.12 ROAD/RIVER OBJECTS KLASSE

Hierin zitten alle objecten die voor dat specifieke onderdeel nodig zijn, bijvoorbeeld de Road objects heeft alle wegobjecten zoals Bus/Car/Tuck en eventueel FastCar(extra werk) en de River objects heeft alle rivier elementen.

4.13 BUS/CAR/TRUCK/FASTCAR ROADOBJECTS KLASSE

Dit zijn de specifieke objecten met hun eigen eigenschappen voor de wegelementen, zo zal een bus langzamer gaan dan een auto bijvoorbeeld.

4.14 TREE/CROCODILE RIVEROBSTACLES KLASSE

Dit zijn de specifieke elementen met hun eigen eigenschappen voor de rivier elementen, Zoals een boom of krokodil.

4.15 TREESIZE ENUM

De Tree Klasse maakt gebruik van deze TreeSize Enum om de juiste afbeelding in te laden. Er hoeft alleen naar een grootte aangegeven te worden, de rest regelt deze Enum.

4.16 BALL OBJECT KLASSE

Dit is het element voor de bal met zijn eigen snelheid en instellingen.

5. KLASSENDIAGRAM

Hieronder wordt het klassendiagram in Figuur 1 getoond. Ook kan deze klassendiagram gevonden worden in de bijlage onder de naam: Frogger Klassediagram.pdf. De klassen in het diagram welke geel zijn, zijn optioneel.

6. CONCLUSIE

In deze conclusie komen we allereerst terug op de kernvraag welke gesteld is in de inleiding, namelijk: hoe gaan we de game Frogger opzetten en welke klassen en variabelen hebben we nodig om tot een werkende game te komen?

We hebben gezien dat er best veel verschillende klassen nodig zijn om tot een goed spel te komen. Ook hebben we gebruik gemaakt van de gameengine. Dit heeft ons wel verschillende hoofdbrekends geleverd, zoals:

- Het uitzoeken hoe de gameengine precies werkt (kostte veel tijd);
- De gamepositie start tegen alle verwachting in bij 0, 0 bovenaan;
- Menu's zijn allemaal gameobjecten die variabelen hebben welke nooit worden gebruikt;
- Direction wordt niet gebruikt bij het tekenen van de sprite, daardoor kan de afbeelding niet draaien;
- Een player is niet gemakkelijk aan een tilemap te binden.

Het is goed om na een Functioneel Ontwerp te gaan verdiepen in de materie en daar dan een duidelijk document over te maken die als richtlijn kan worden gebruikt om de opdracht tot een goed einde te brengen.

7. BRONVERMELDING

Het huidige document heeft geen bronnen.