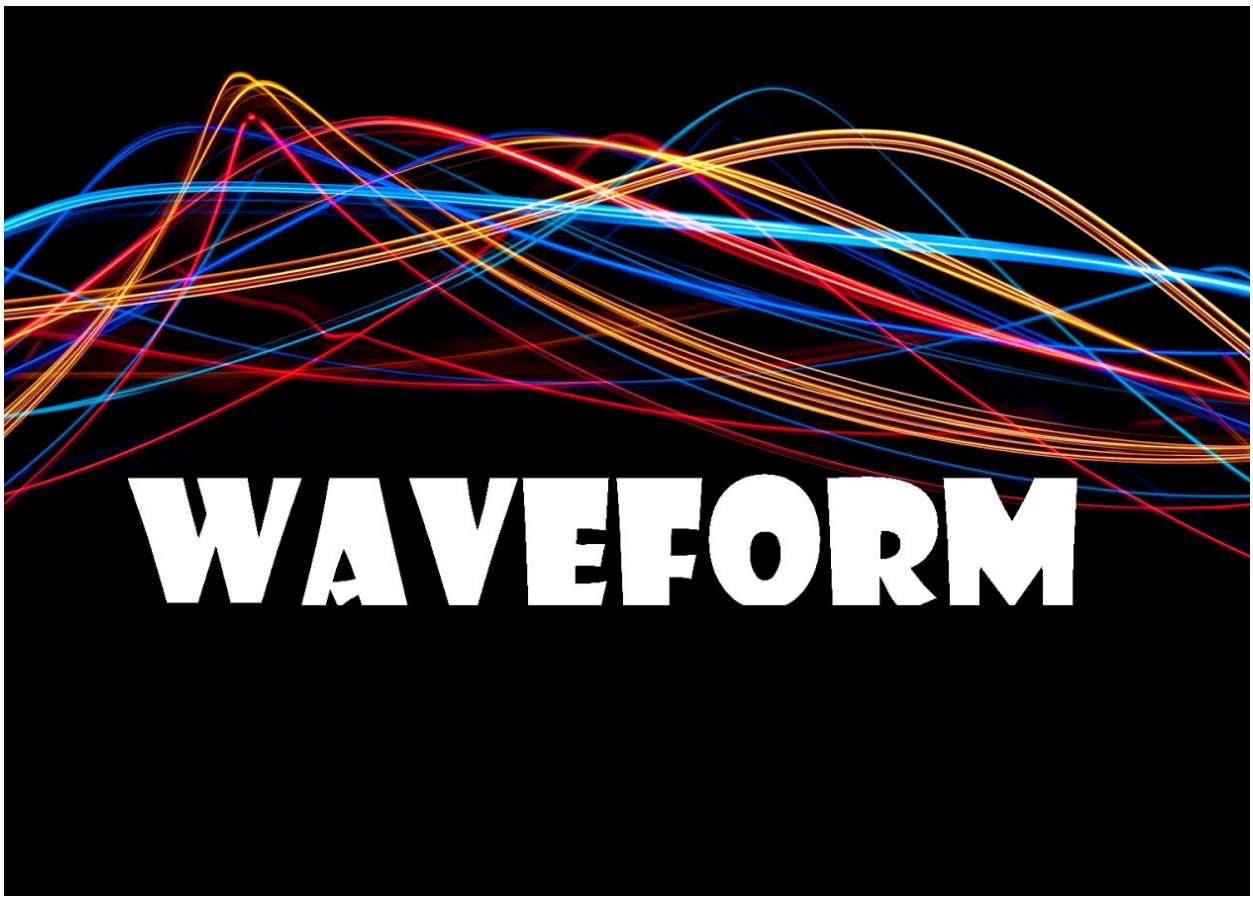
Game Development

Technisch Verslag

Inside the waveform



Team skilled eagle **concept v4.20 05-02-17**

Bob Thomas	1685710
David Driessen	1677976
Max Beunk	1691309
Jeremy Ruizenaar	1686577
Mike Hilhorst	1676029
Robbie Valkenburg	1666428

Managementsamenvatting

Dit verslag bevat alle technische details betreffende de game waveforms. Het doel van het maken van deze opdracht is het bestuderen van game programmeer technieken en deze kunnen toepassen. Mede Als het leren werken met een ontwikkelmethode zoals Scrum.

Voor dit project is er voor de beat detection veel informatie verzameld op het internet.
Verder is er bij het opstellen van het technisch zowel als het functioneel ontwerp altijd uitgegaan van zo min mogelijk overbodige zaken toevoegen. Dit om ingewikkelde constructies en dependencies zoveel mogelijk te vermijden.

Na het doorlopen van dit project zijn de volgende belangrijkste conclusies en aanbevelingen opgesteld.

1. Geheugenbeheer is zeer belangrijk en moet niet vergeten worden.
2. Het strak bijhouden van de sprints is belangrijk om alles zo soepel mogelijk te laten verlopen en moet dus ook strak gevolgd worden.

De belangrijkste resultaten zijn het opgeleverde spel mede dit verslag.

Index

Inleiding

Voor het thema opdracht gaming wordt een 2D-game gerealiseerd. Het doel van deze opdracht is het leren werken volgens de ontwikkelmethodes Scrum & Agile binnen een kleine ontwikkelgroep. Hierbij is het gebruik van en versiebeheersysteem een zeer belangrijk aspect. De projectweken worden opgedeeld in korte sprints van drie tot dagen. Na elke sprint worden er een reeks deel producten opgeleverd waarop verder voortgebouwd kan worden, aan het eind van het project wordt de game gedemonstreerd d.m.v een demo. Tevens wordt dit technisch verslag ingeleverd.

Doel

Het doel van dit verslag is de lezer informeren over de gebruikte technieken binnen de architectuur van deze game. Elk belangrijk component van de game heeft een beschrijving bevattende waar het voor dient, hoe het werkt, en waarom voor die oplossing gekozen is. Ook worden er een aantal knelpunten die tijdens het project aan het licht kwamen toegelicht. Dit verslag is bestemd voor mensen die voort zouden willen bouwen op dit project.

Vooruitblik

In de komende hoofdstukken kunt u de volgende onderwerpen verwachten.

1. Technisch ontwerp
    -beschrijving architectuur
2. Realisatie
    -ontwikkelmethodes
    -installatie
    -knelpunten
    -oplossingen
    -algoritmen en datastructuren
    -tests en experimenten
3. Evaluatie
    - verbeterpunten
4. Conclusies en aanbevelingen

Technisch Ontwerp:

De game Waveforms bestaat uit de volgende componenten:

De game state machine:

De game state machine is de state machine die tussen de verschillende game states en hun bijbehorende functies switched . Hieronder vallen: de loading state, settings state, pause state, level select state en de playstate.

De menu klasse:

De menu klasse wordt gebruikt om de verschillende menu's die opties bevatten uniek voor elke state aan te maken.

De debug overlay:

De debug overlay klasse wordt gebruikt om real time informatie over het spel te weergeven aan de gebruiker.

De sound manager:

De sound manager regelt het decoderen van de ingeladen geluidsbestanden die gebruikt worden in-game.

ScreenObject klasse:

De abstracte klasse screenObject zorgt ervoor dat alle afgeleiden subklassen op een uniforme manier gebruikt kunnen worden. hun gemeenschappelijke functies: update() , draw() en input() worden hierin ondergebracht.

Weapon klasse:

Abstract klasse Weapon is een interface voor de melee en ranged weapon. Deze interface zorgt ervoor dat elke type wapen op een uniforme manier gebruikt kan worden. Een ranged weapon bevat tevens een std::Vector met unique-pointers naar bullets. Een bullet heeft een boolean waarmee aangegeven kan worden of er een collision heeft plaatsgevonden. Op basis van deze eigenschap kunnen de bullets verwijderd worden uit de vector.

Level klasse:

De level klasse bevat meerdere std::Vectoren die naar alle level objecten verwijzen, inclusief de speler. De vectoren en attributen van deze klasse worden aangemaakt met de level factory die onder is gebracht in deze klasse. De level factory haalt zijn input uit een txt file.

Character klasse:

De character klasse handelt alle animatie, verplaatsing en status veranderingen af van een character. De cyber-enforcer klasse bevat een character, met het verschil dat de stamina bar niet getekend wordt en de health bar zich boven de character bevindt. Bij een player character worden de health and stamina bars ook niet getekend maar opgehaald door de playerHUD klasse die het weergeven van de speler-info afhandelt.

StatusBar klasse:

De statusbar klasse handelt het omrekenen van currentValue naar currentValue in procenten af.
Op basis van deze omrekening kan er een rechthoekig vlak opgespannen worden met als breedte de currentValue in procenten vermenigvuldigd met de geschaalde size van de window.

Realisatie

De softwareontwikkeling

De Game Waveforms wordt ontwikkeld door een team bestaande uit 6 leden. Voor dit project wordt er gebruik gemaakt van een agile scrum board genaamd Youtrack en de ontwikkelmethode Scrum. Binnen Youtrack kunnen leden zogeheten user-story aanmaken. Een user-story bevat een zoveel mogelijk losstaand onderdeel wat binnen het kader van het project gerealiseerd moet worden. Bij de start van het project is er met alle leden een sizing meeting gehouden. Hierin worden met het eindproduct in gedachten, zoveel mogelijk user-story's aangemaakt. Deze worden vervolgens gesorteerd op de tijd die gemiddeld verwacht wordt om deze user-stories te volbrengen. Vervolgens kan de eerste sprint gestart worden. Hierin kiest elk projectlid een of meerdere user-stories uit waar vervolgens aan gewerkt kan worden. Een sprint duurt voor dit project drie dagen. Aan het einde van een sprint kiest elk projectlid weer nieuwe user-stories uit die van nut zijn voor de volgende sprint. User-stories die niet voltooid zijn worden meegenomen naar de volgende sprint. Verder wordt er tevens een retrospective gehouden waarin wordt teruggekeken naar de afgelopen sprint.

Platform

Voor dit spel is er gekozen voor het platform PC dit omdat dit voor iedereen wel toegankelijk is

Installatie van software

Binnen dit project zijn er een aantal software standaarden in gebruik:

-Clion: De gebruikte IDE.

-SFML: The simple fast media library. Deze library wordt gebruikt om de audio en grafische componenten af te handelen.

-GIT: Het versiebeheersysteem.

-FMOD: De library om geluidsbestanden te kunnen decoderen.

-Tiled: Een level editor dat een level naar een .txt file kan schrijven.

Knelpunten

Bij het ontwikkelen van de beat detection kwamen de volgende problemen aan het licht.

1. Hoe onderzoek je iets wat je totaal niet kent
2. Hoe gaan we algoritmes implementeren en wat gaan we doen met de grote hoeveelheid samples die muziek bevat

Oplossingen

Beat detection

Na 2 dagen research gedaan te hebben over bpm detection en beat detection kwamen we erachter dat de mensen die het eerder gemaakt hebben niet graag hun geheimen bloot stellen. Na wat gespeeld te hebben met de sfml audio samples vonden we dat we te weinig data eruit vangde hebben we besloten om FMOD audio engine te implementeren.
Dit ging gelukkig met weinig problemen bij het begin totdat we aan het cross platform gedeelte begonnen. Toen kwamen we erachter dat de FMOD c++ versie alleen bestaat voor visual studio compilers en dus niet onze MINGW compiler. Gelukkig na wat google kwamen we er ook achter dat de C Bindings wel gewoon werken en hebben onze FMOD api omgeschreven naar FMOD met C bindings het ziet er wat minder mooi uit maar het werkt wel stabiel.
Nadat we FMOD werkend hadden kwamen we weer op het probleem dat er weinig of geen informatie is over wat wij wouden doen en de informatie die we vonden was vol met wiskunde en digital signal processing iets wat ons allemaal boven ons hoofd ging. Ulteindelijk hadden we een realtime fft modulering werkend en daarmee ook per ongeluk een audio visualizer gebouwd. Toen dat allemaal opeens werkte kregen we onze motivatie weer terug en gingen we zoeken naar beat detection algorithms en met wat zoeken vonden we een c++ sfml beat detection systeem gemaakt door een fransman op youtube. Zonder veel van de code te snappen konden we een werkend prototype maken dat op de beat stuiterde en de bpm uitrekende van verschillende liedjes. We hebben redelijk wat herschreven van de orginele code met franse comments het algorithm wat gebruikt wordt zal verder worden uitgelegd in het volgende hoofdstuk.

Gebruikte algoritmen en datastructuren

Beat detection

Ons BeatDetection algorithm komt met een groot deel uit een willekeurig filmpje wat we gevonden hadden op youtube [https://www.youtube.com/watch?v=jZoQ1S73Bac](Beat detection c++). de code was helaas voor ons helemaal gedocumenteerd in het frans en vrij onduidelijk gestructureerd.
Na wat variabelen heen en weer schuiven en de code doorspitten hebben we een groot gedeelte van het algoritme ontcijferd. Hoe het systeem werkt is via FMOD en daar de raw data uit de muziek. Dit wordt gedaan door de muziek te locken via FMOD_Sound_Lock.
Als de muziek gelocked is kan binnen een paar milliseconden de left en right channel data van de muziek worden ingelezen in een twee int pointers. En daarna unlocken we de muziek met FMOD_Sound_Unlock. Nu de muziek weer is vrijgegeven kan de worden afgespeeld door de geluidskaart van de computer. Ondertussen hebben 2 int pointers met een grootte van de hoeveelheid samples in de muziek. Deze gegeven zijn allemaal makkelijk op te halen via de FMOD api met bijf FMOD_Sound_getLength. Nu onze data beschikbaar staat wordt het doorgegeven aan de beatDetector klas.
Deze klasse bezit over een paar helper functies genaamd.

- Normalize - normaliseert het signaal en verwijdert ruis
- Energie - geeft de energie van het signaal terug
- Search_max - zoekt de hoogste waarde float pointer

De energie berekening gaat als volgt.

```
for (int i = offset; (i < offset + window) && (i < length); i++) {
   energie = energie + data[i] * data[i] / window;
}
```

Of te wel
Voor alle samples met een offset van ongeveer een beat
Doe een RMS (Root Mean Square) calculatie en deel het door het aantal samples om de median te berekenen tel die bij elkaar op en je hebt de energie van het sample.

Nu de helper functies duidelijk zijn kunnen we beginnen met de berekeningen.
Eerst zetten initialiseren we onze start variabelen.

```
found_beats = 0; - gevonden aantal beats
length = sound->length; - hoeveelheid samples
energie1024 = new float[length / 1024];  - energie van 1024 samples
energie44100 = new float[length / 1024]; - energie van 44100 samples gebaseerd op de 1024 samples
conv = new float[length / 1024]; - convolution word gevuld met de energie gebaseerd op onze training samples en  offset
beat = new float[length / 1024]; - word een 0 of 1 in opgeslagen of er in beat in de sample zit
```

```
energie_peak = new float[length / 1024 + 21]; - de kracht van de peak in de muziek(zal later kunnen gebruikt worden voor meer damage of hoger springen)
```

Nu onze start variable klaar staan kunnen we beginnen met het algoritme.
Het algoritme wat we gebruiken wordt uitgebreid beschrijven op deze website..
http://www.flipcode.com/misc/BeatDetectionAlgorithms.pdf
Maar in een wat simpele taal gaat het als volgt.
Hij berekent de pieken van de muziek samples en als die peak over de treshold gaat is er een beat. De threshold wordt bepaald door een kleine training systeem te bouwen over het liedje. Wat bijhoud waar de pieken zit en daaraan zijn data normaliseert naar een meer binaire vorm van data. Via de genormaliseerde data gaat te kijken hoeveel stappen er tussen de voorspelde beats en als die stappen overeenkomen met de threshold en verwachte stappen tussen beats gaan we er vanuit dat het een echte beat in de muziek is en wordt er bij de beat float op de positie een 1 gezet anders komt er op die positie een 0. Helaas kwamen we erachter dat dit niet altijd even precies is en heb ik besloten het half te brute force. Het berekenen van bpm doet die erg stabiel en daarmee kunnen we berekenen hoeveel beats er zouden moet voorkomen.
Dus ik laat de decoder net zo vaak over zijn samples heen gaan met alle opnieuw geïnitialiseerd totdat die bij het verwachte aantal beats aan komt met een foutmarge van 10 beats.
Dit word gedaan door een simpele formule
Gevonden beats<=(lengte van geluid in ms / 1 minuut in ms/bpm van de muziek))-10
Als die meer beats of een gelijk aantal beats vinden betekent dat die de juiste beats heeft gevonden en mag die door naar de beats in 0 en 1 schrijven naar een txt.file die we later kunnen hergebruiken voor het zelfde liedje in plaats van opnieuw decoderen.
Als die minder beats vind begint die weer helemaal opnieuwe met het decode net zolang totdat die het verwachte aantal beats vind. Dit algoritme kan zeker verbeterd worden alleen wegens gebrek aan tijd en ervaring zijn we met huidige systeem erg tevreden.

Resource loader

Dit systeem is heel simpel en komt uit het boek (SFML GAME DEVELOPMENT,2013).

Het maakt een paar template klasse voor Muziek,Textures,Fonts,Shaders etc.

Waar je uit een vector gevuld met unique pointers van het aangegeven typen kan lezen en schrijven.

Deze waarde worden gekoppeld aan een ENUM zodat je gemakkelijk textures kan onthouden en niet meer hoeft te onthouden in welke map of welke folder ze staan. De textures worden in ons spel bij het opstarten allemaal geladen dit duurt een tijdje maar wij vonden het acceptabel.

Tests en experimenten

De beat decoder is getest met een assortiment aan gevarieerde muziek. Hieruit zijn de volgende resultaten verkregen:

-Nummers langer dan tien min kunnen het decodeerprocess vast laten lopen,
-Nummers met het aantal beats per minute hoger van 170 worden niet altijd gedetecteerd.
-Nummers die bestaan uit complexe geluiden worden ook niet altijd gedetecteerd.

Het geheugengebruik van het spel is ook getest en bedraagt 2,7 GB aan werkgeheugen. Dit is nog best aan de hoge kant

Evaluatie

Het gebruik van smart pointers en het opruimen van deze smart pointers had bij nader inzien beter gekund. Dit omdat 2,7 GB gebruikt werkgeheugen relatief hoog is.

Conclusies en aanbevelingen

Op het functioneel vlak is het strak bijhouden van de sprints handig om vast te houden. Binnen dit project kon er relatief losjes mee omgegaan worden, Maar zo nu en dan kon er toch wat onduidelijkheid ontstaan. Om de doorloop van het project zo soepel mogelijk te laten verlopen is het handig om dit gewoon aan te houden zodat iedereen op dezelfde lijn zit.

Op het technisch vlak is geheugenmanagement een belangrijk aspect. Het nalatig omgaan met pointers en smart pointers kan slechte performance of undefined behavior met zich mee brengen. Het is dus altijd een goede gewoonte om dit tot in de puntjes in de gaten te houden. Dit wordt ook ten strengste aanbevolen.

Bronvermelding

1. Jan Haller Henrik Vogelius Hansson, (2013). SFML Game Development, Packt Publishing Limited.
2. Barsichou, (2011), Beat detection C++, https://www.youtube.com/watch?v=jZoQ1S73Bac.