**1ste Prototype Gitaarprogramma**

*In dit document beschrijf ik het proces wat ik heb doorlopen om mijn eerste prototype te maken met beschrijving per KPI die in de opdracht geplaatst zijn.*

*Het prototype is een programma om virtueel een gitaar te kunnen bespelen met je eigen handen.*

**Analyseren Software**

Om een keuze te kunnen maken tussen de Software die beschikbaar is voor het maken van het gitaarprogramma heb ik een requirementanalyse uitgevoerd.

Technisch gezien is het programma wat ik wilde maken een game dus moest ik een game engine gaan kiezen. Ik heb geen ervaring met het maken van games dus ben ik gaan kijken naar de bekendste game engines omdat deze de grootste communities hebben, deze hebben dus veel meer informatie en tutorials beschikbaar.

**Knelpunt Analyse**

**++** zeer goed (2)

**+** goed (1)

**o** middel (0)

**-** slecht (-1)

**- -** zeer slecht (-2)

|  |  |  |  |
| --- | --- | --- | --- |
|  | **../../../../Pictures/Unity_Technologies_logo.svg.png** | **../../../../Pictures/1425334231-unreal-engine-logo.png** | **../../../../Pictures/global.logo** |
| **Performance** | + | ++ | + |
| **Platform Compatibility**  **(Development)** | + | + | o |
| **Platform Compatibility**  **(Running)** | ++ | + | o |
| **Leercurve** | + | - | + |
| **Scripting Language** | ++ | + | ++ |
| **Punten** | 7 | 4 | 4 |

Om erachter te komen wat de platform compatibility van elke engine is heb ik gekeken naar de system requirements per game engine op de officiële website.

Verder heb ik onderzoek gedaan door op school aan medestudenten naar ervaringen te vragen en door eigen ervaringen hierop toe te passen.

**Unity**

**Performance**

Deze informatie heb ik gekregen door rond te vragen naar mensen die ervaring hebben met Unity, Bas heeft mij hierin uitgelegd hoe dit zit.

Unity krijgt 1 punt bij dit punt, de performance van Unity is erg goed maar ze maken gebruik van C# die gebruik maakt van een garbage collector. Dit betekent dat C# automatisch zoekt naar code of functies die niet relevant zijn om deze vervolgens te verwijderen. C# is puur script based, dit zorgt ervoor dat het minder efficiënt is.

**Platform Compatibility (Development)**

Het is mogelijk om te developen in Unity wanneer je Windows of MacOS op je laptop/desktop hebt draaien. Hierom heb ik dit kopje 1 punt gegeven, de meeste mensen hebben Windows of MacOS. Als Unity ook op Ubuntu zou runnen zou ik het 2 punten gegeven hebben omdat het dan op alle populaire operating systems voor pc’s zou draaien. Voor mijzelf is het belangrijkste dat het op Windows of MacOS draait dus hierin krijgt het een prima score.

**Platform Compatibility (Running)**

Games/Simulators die gemaakt zijn in Unity kunnen op meer platformen afgespeeld worden dan waarop het gemaakt kan worden. De Unity games/simulators kunnen gespeeld worden op Windows, MacOS, Ubuntu en SteamOS, Android, iOS, Switch, Wii, Nintendo DS3, PS4 en XBOX One. Omdat de producten op zoveel verschillende operating systems afgespeeld kunnen worden (de enige engine die spellen maakt voor SteamOS) heb ik deze 2 punten gegeven.

**Leercurve**

In eerste instantie ben ik door veldonderzoek te doen en rond te vragen naar mensen met ervaring erachter gekomen dat Unity een van de meest toegankelijke game engines zijn om in te werken. Dit wordt ook nog versterkt doordat ik al wat van de scripting language ken. Op het internet vind ik veel mensen die het moeilijk vinden om als beginner hierin te werken maar dat is voornamelijk omdat coding zo uitgebreid is, dit zal dus in een andere game engine niet anders zijn. Iedereen online zegt dat game engineering ongeacht de engine een hoge leercurve heeft maar hierin heeft Unity een erg uitgebreide documentatie en hebben Unity developers erg veel tutorials online staan die je als verwijzing kan gebruiken. Om deze redenen heb ik Unity 1 punt gegeven.

**Scripting Language**

Unity maakt gebruik van een aantal scripting languages, zo kan je kiezen tussen C#, JavaScript of Boo. Om hierin te besluiten wat beter is heb ik naar mezelf gekeken en de populairste taal om in te programmeren omdat ik het zelf moet maken en het zo ook mogelijk is voor iemand anders om gemakkelijk een functie toe te voegen. C# en JavaScript zijn in het algemeen erg populaire talen en ken ik deze zelf ook al (C# verkennend en JavaScript geoefend). Om deze redenen geef ik het 2 punten voor toegankelijkheid en functionaliteit.

**Unreal Engine**

**Performance**

Door de informatie die ik van Bas gekregen heb door rond te vragen naar mensen met ervaring in het programmeren met Unity en/of Unreal Engine ben ik erachter gekomen dat C++ (de scripting language waar Unreal Engine gebruik van maakt) geen garbage collector gebruikt (in tegenstelling tot C#). Hierdoor wordt de performance verbeterd omdat de garbage collector in Unreal Engine zelf zit. Omdat deze in de engine zelf zit kan je zelf bepalen hoe deze werkt en kan je er zelf dus voor zorgen dat dit efficiënter werkt. Door deze redenen krijgt Unreal Engine hier dus 2 punten.

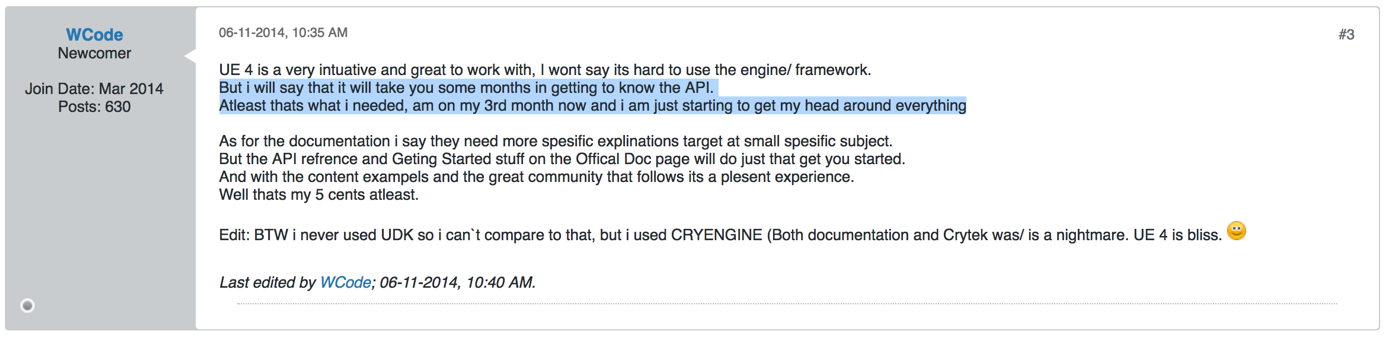
**Platform Compatibility (Development)**

Unreal Engine is beschikbaar op Windows en MacOS. Dit zijn de meest gangbare operating systems hierom heb ik het 1 punt gegeven, om er 2 te krijgen zou het ook nog beschikbaar moeten zijn voor Ubuntu om zo Linux gebruikers hierin mee te nemen.

**Platform Compatibility (Running)**

Het runnen van een game of simulatie die gemaakt is in Unreal Engine is mogelijk op Windows, MacOS en Ubuntu, PS4, XBOX One, Android, iOS en Switch. Dit laat zien dat de games/simulators die gemaakt worden d.m.v. Unreal Engine voor veel doeleinden gemaakt kunnen worden en verdient daarom een score van 1 (hierbij hou ik ook rekening met de score van Unity die nog diverser is). Ik ben achter deze informatie gekomen door rond te vragen op school en online onderzoek te doen naar de system requirements van Unreal Engine.

**Leercurve**

****

https://forums.unrealengine.com/development-discussion/c-gameplay-programming/7756-ue-4-udk-4-learning-curve-and-documentation-compared-to-udk-3

Hierbij wordt in de forums van Unreal Engine beschreven dat de leercurve vrij groot is, hij zegt in eerste instantie van niet maar vertelt wel dat het maanden duurt om het onder de knie te krijgen. Ik heb kritisch naar deze review gekeken omdat de persoon zichzelf tegenspreekt (“I won’t say it’s hard to use” en “But I will say that i twill take you some months in getting to know the API”). Er is natuurlijk een mogelijkheid dat hij een paar maanden niet veel vindt maar in mijn geval is het dat wel.

Ook heb ik op school rondgevraagd naar Unreal Engine en kwam ik hier ook tot de conclusie dat de engine een grote leercurve heeft net als de taal die je moet gebruiken. Daar tegenover staat wel dat de engine wel hele mooie visuals levert en goed presteert.

**Scripting Language**

De taal die Unreal Engine gekozen heeft om mee te werken is C++, dit is een taal met een erg hoge leercurve (dit heb ik gevraagd aan Technology studenten die hiermee moeten werken). Om hier goed mee om te gaan wordt ook nog geadviseerd om eerst C te leren wat erg veel tijd zou kosten. Met deze taal heb ik geen ervaring en zou mij dit ook veel tijd gaan kosten om onder de knie te krijgen, daarom heb ik Unreal Engine hier 1 punt gegeven omdat de taal goed in elkaar zit/werkt maar het een hoge leercurve heeft om te gebruiken.

**Cryengine**

**Performance**

Zelf heb ik games gespeeld die gemaakt zijn door Cryengine, deze lopen erg goed maar ze zijn erg zwaar om af te spelen. Hierdoor is de performance heel goed maar is de toegankelijkheid van deze performance lager. Om deze redenen heb ik Cryengine 1 punt gegeven.

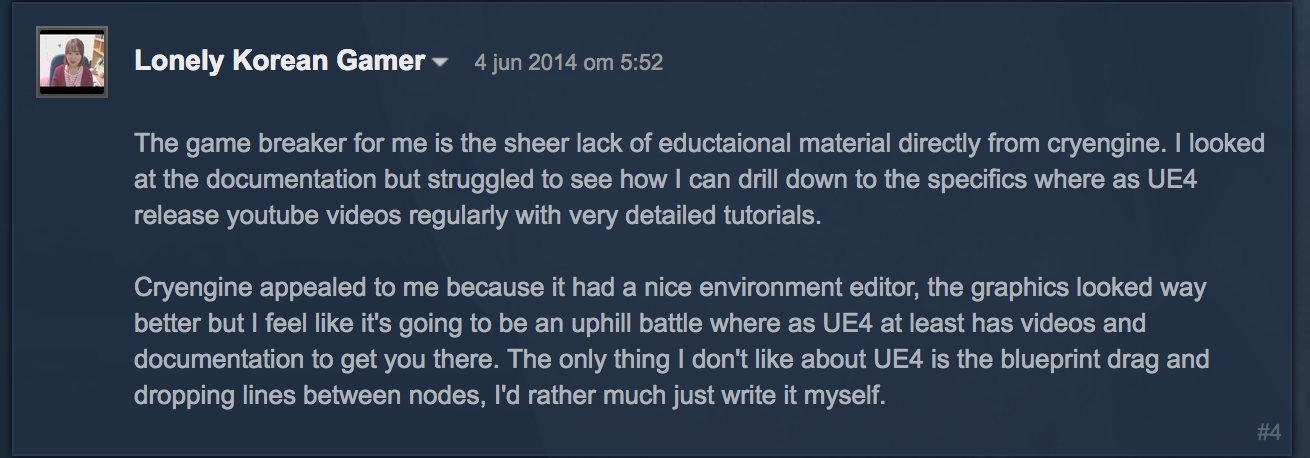
**Platform Compatibility (Development)**

Cryengine heeft van mij 0 punten gekregen omdat het programma alleen op Windows te gebruiken is, hierdoor sluit Cryengine veel developers buiten die MacOS of Ubuntu gebruiken. De informatie heb ik opgezocht online op de officiële webpagina van Cryengine.

**Platform Compatibility (Running)**

De programma’s die met Cryengine gemaakt worden zijn net als Cyrengine zelf alleen beschikbaar voor Windows. Dit laat net als bij de Development van de game nog te wensen over omdat het veel eindgebruikers buitensluit. Vandaar dat ik Cyrengine ook hiervoor 0 punten heb gegeven.

**Leercurve**

****

Deze comment komt van de Steam community bij de comment section van Cryengine. Een groot obstakel bij het produceren in Cryengine is dat ze zelf weinig documentatie leveren om je te ondersteunen bij het programmeren.

Toch heb ik Cryengine 1 punt gegeven hier omdat het beide C++ en C# ondersteunt (C# wordt bij Unity gebruikt en C++ bij Unreal Engine). Naast deze talen kan er ook in Lua geprogrammeerd worden binnen Cryengine wat het diverser maakt wanneer je over de Scripting Language praat. De documentatie mag dus iets te kort schieten maar de leercurve wordt wel gecorrigeerd doordat het met beide C# en C++ werkt.

**Scripting Language**

De talen waarmee je als Cryengine developer kan werken zijn C++, C# en Lua. Waarom ik hiervoor 2 punten gegeven heb is omdat dit allebei de talen waar Unity en Unreal Engine mee werken ondersteunt. Naast deze 2 kan je ook nog met Lua programmeren wat zorgt voor een brede keuze waarvan 2 zeer gewilde programmeertalen zijn.

**Adviseren Software**

Doordat ik een knelpunt analyse uitgevoerd heb kan ik een onderbouwd advies geven over mijn software keuze. Hieruit komt dat voor mij Unity de beste keuze is, de grootste redenen zijn persoonlijk zoals scripting languages waar ik al een begin van ken maar ook gericht op de gebruiker door compatibility met vele operating systems. Naast een onderbouwde keuze voor mijn project kan ik ook adviezen geven aan andere personen die iets vergelijkbaars zouden willen maken. Dit komt omdat ik de zwaktes en sterke kanten van elke engine in kaart heb gebracht en dit kan toepassen op een situatie.

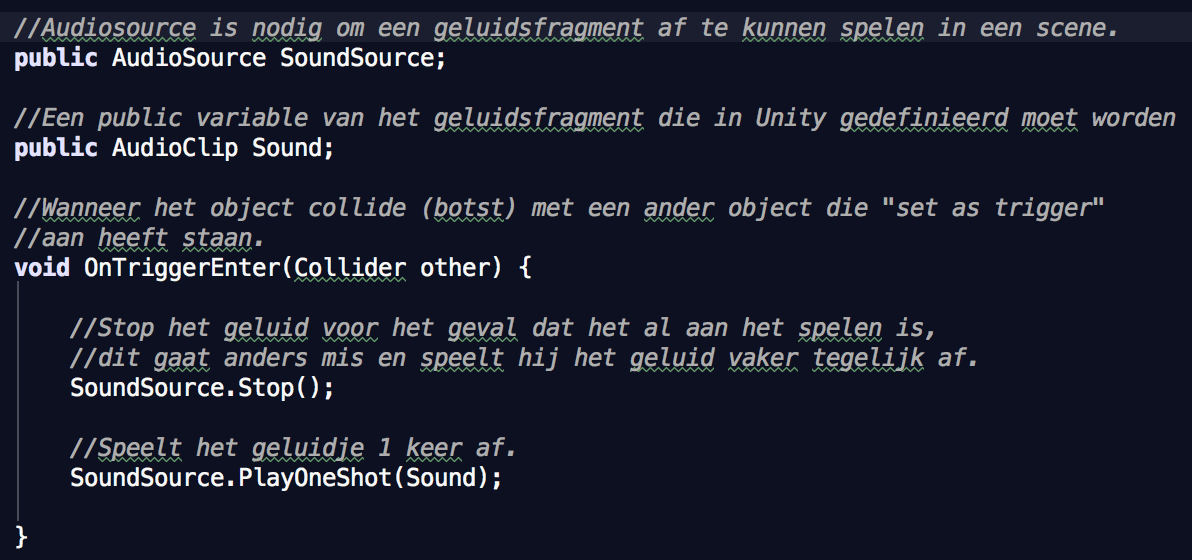
**Ontwerpen Software**

Ik heb gebruik gemaakt van de documentatie die Unity levert om zo een eigen functie te schrijven met C#. Ook heb ik een C# library gedownload genaamd “BezierCurve”, deze helpt bij het maken van een pad waar ik elementen aan vast kan maken zodat ik het pad een “wave” beweging kan laten maken zodat het lijkt op een gitaar snaar.

De functionaliteit van het programma is de mogelijkheid om d.m.v. motion tracking met je eigen hand verschillende snaren aan kunnen raken die na aanraking een mp3 afspelen en tegelijk een “wave” beweging maken.

**Realiseren Software**

In het begin had ik de verkeerde indruk van de hand objecten die de Leap assets toevoegen aan Unity. Nadat ik een tutorial over een ander project gezien had dacht ik namelijk dat ik nog vanalles toe moest voegen zoals rigidbodies etc. maar dit was dus niet het geval. Toen Bas mij liet zien dat ik gewoon functies kon schrijven en deze toekennen aan de handen snapte ik hoe ik verder moest. Ik heb dus een functie geschreven met behulp van de Unity documentatie die geluid afspeelt wanneer de Leap Motion hand collide met een object die als trigger geplaatst is. De code ziet er zo uit:



**Analyseren Hardware**

Om een goede hardware keuze te maken heb ik een requirementanalyse gemaakt waarin een vergelijking gemaakt wordt tussen hardware om het doel van mijn project te bereiken. Omdat ik weet dat we hier op school Leap Motion hebben om dit te doen heb ik op het internet alternatieven gezocht voor Leap Motion om een vergelijkbaar product te vinden die dit zou kunnen realiseren. Deze heb ik met elkaar vergeleken om uit de beste hardware keuze te komen voor mijn project.

**Knelpunt Analyse**

**++** zeer goed (2)

**+** goed (1)

**o** middel (0)

**-** slecht (-1)

**- -** zeer slecht (-2)

|  |  |  |
| --- | --- | --- |
|  | **../../../../Pictures/white_background_logo.png** | **../../../../Pictures/765px-Kinect_logo.svg.png** |
| **Performance** | o |  |
| **Platform Compatibility** | + |  |
| **Leercurve** | ++ |  |

**Leap Motion**

**Performance**

Omdat Leap Motion een relatief klein oppervlakte aan sensoren heeft is het gebied om interactie uit te voeren gelimiteerd. De sensor ziet vaak de hand verkeerd waarna je je hand overnieuw er overheen moet houden. Ook kan je sommige bewegingen niet maken omdat de sensor alleen van de onderkant je hand scant, dit zorgt ervoor wanneer je je hand rechtop houdt de sensor je beweging niet kan zien. Wanneer je je hand horizontaal houdt kan de sensor je beweging wél gemakkelijk meten. Leap Motion heeft zijn issues maar wanneer de UI een beetje rekening houdt met deze flaws werkt de Leap Motion hiermee. Omdat de Leap zijn issues heeft maar desondanks voor mijn project prima zou zijn geef ik het 0 punten.

**Platform Compatibility**

Leap Motion is bedoeld voor gebruik op Windows of MacOS, het is mogelijk deze ook te gebruiken op Playstation of Xbox maar daar is deze niet voor bedoeld. Windows en MacOS zijn de meest populaire operating systems dus dit is positief voor Leap maar omdat het alleen maar de 2 populairste heeft geef ik het 1 punt in de knelpunt analyse.

**Leercurve**

Als je Leap Motion wilt gebruiken moet je de assets downloaden waarna je deze gemakkelijk in een engine kan laden. Met deze ingeladen assets krijg je een aantal hand objecten waarvan ze allemaal al werken en interacties kunnen uitvoeren met andere objecten. Als je wilt dat er iets anders gebeurt dan dat je het object kan verplaatsen moet je de code zelf schrijven. Hierdoor is de leercurve erg laag voor Leap Motion en ligt de moeilijkheidsgraad alleen bij de engine waarin je deze laadt. Omdat het zo makkelijk in gebruik is krijgt Leap 2 punten bij Leercurve.

**Kinect**

**Performance**

**Platform Compatibility**

Windows/

Xbox

**Leercurve**

**DUO**

Duo is een vergelijkbaar met Leap Motion, ik heb deze gevonden op Kickstarter maar helaas voor de developer heeft het product niet voldoende funding gekregen om het product uit te brengen. Om deze reden was er ook niet voldoende informatie te vinden online om hier een goed onderbouwde vergelijking van te maken. Natuurlijk had dit niet veel nut gehad voor mijn product maar het is evengoed interessant om te bekijken of deze motion sensor beter of slechter zou zijn voor mijn product als het uitgebracht zou worden. De link naar de kickstarter pagina staat hieronder:

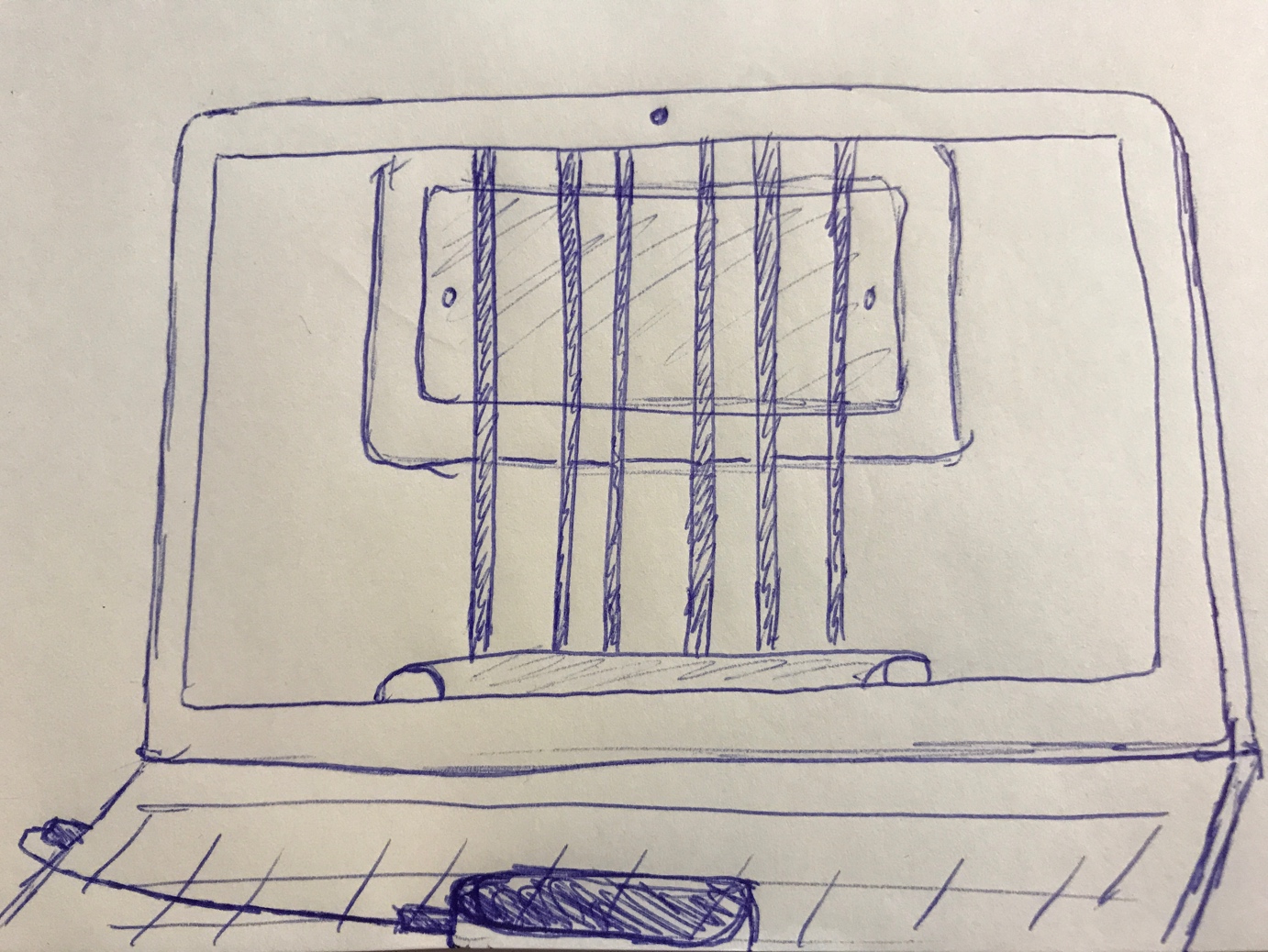
**https://www.kickstarter.com/projects/codelabs/duo-the-worlds-first-diy-3d-sensor/?ref=kicktraq**

**Adviseren Hardware**

Hier komt het onderbouwde hardware advies.

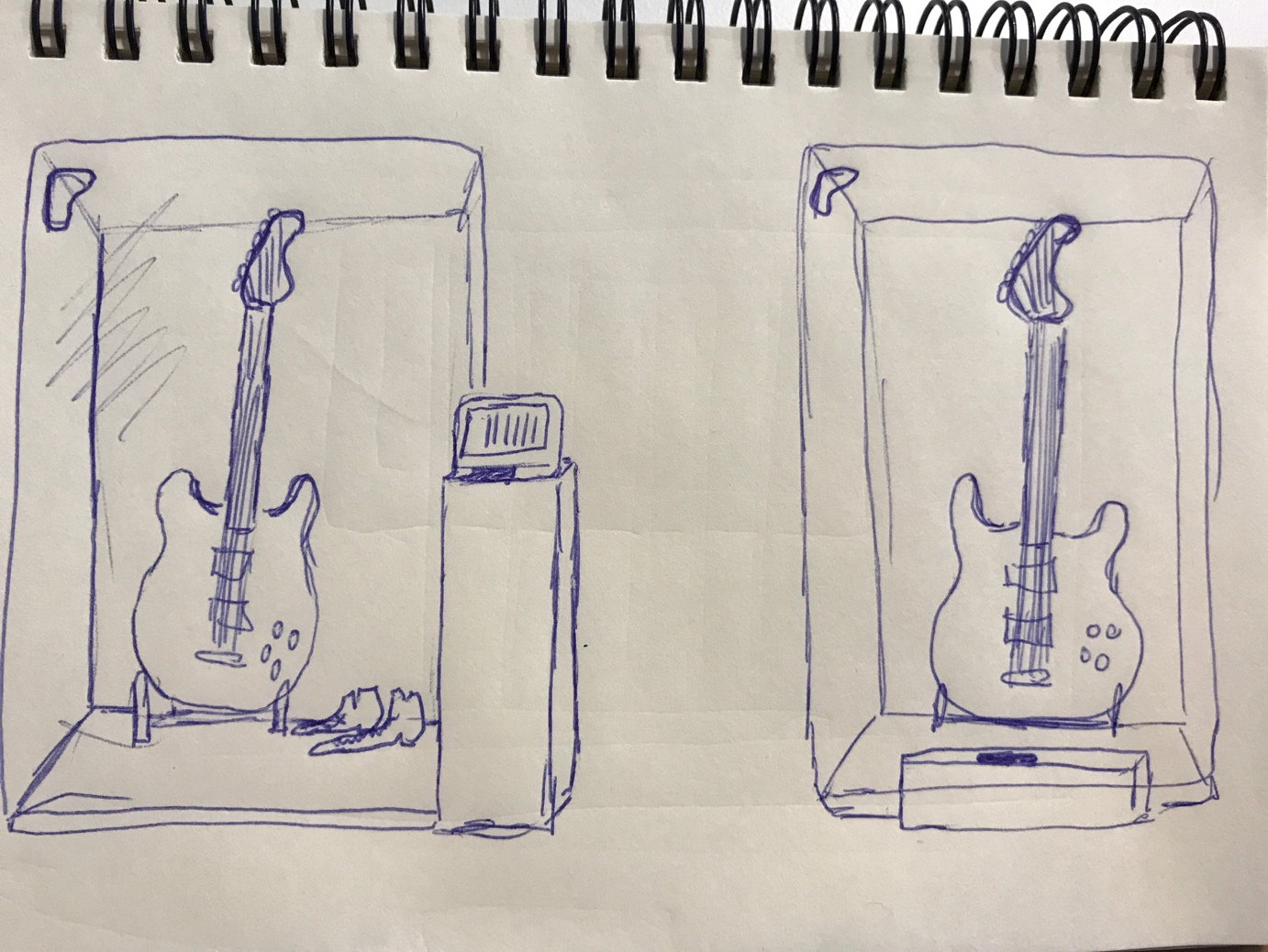
**Ontwerpen Gebruikersinteractie**

De installatie is een toevoeging op een display en de display mag geen aandacht verliezen door de installatie. Hierdoor wil ik de activiteit erg simpel houden en de functionaliteit van de gitaar niet te ingewikkeld maken. Om dit simpel te houden heb ik alleen een UI gemaakt voor de snaren zonder de fretten (de strepen die je indrukt om de toonhoogte aan te passen) zodat je alleen de snaren aan kan raken en geen tonen aan kan passen. Ik heb nagedacht over de rotatie van de UI of dat je deze plat zou leggen zodat het meer op een gitaar lijkt die je bespeelt of rechtop, na beide opties geprobeerd te hebben heb ik hem rechtop gezet omdat het zo beter opvalt in de winkel en anderen hierdoor ook beter mee kunnen kijken met de gebruiker. De snaren heb ik rechtop gezet omdat na het testen duidelijk was geworden dat dit een fijnere beweging is om te maken dan verticaal. Je ziet in de volgende UI dat ik het als een laptopapplicatie heb getekend, na een gebruikerstest ben ik erachter gekomen dat een toetsenbord lastig zou zijn voor een installatie dus heb ik ervoor gekozen om de applicatie te runnen op een tablet of een soortgelijk apparaat zoals een notebook waarvan het toetsenbord verwijderd kan worden.



Na de tests werd ook duidelijk dat de snaren verder uit elkaar moeten staan zodat het makkelijker is om elke snaar apart aan te slaan. De arm van het Leap model moet ook verwijderd worden omdat deze het zicht belemmerd.

Naast een UI design heb ik ook een ontwerp gemaakt van de installatie, hoe deze in de winkel zou moeten komen staan.



Hier heb ik een tekening gemaakt van de laptop die naast de vitrine staat zodat je de gitaar op het scherm zou kunnen bespelen en ik heb een tekening gemaakt hoe het eruit zou kunnen zien zonder UI op een scherm, namelijk alleen de Leap in een kastje onderin zodat je de gitaar zelf als UI kan gebruiken. Hoewel de tweede optie waarschijnlijk cleaner eruit ziet is het hier minder duidelijk wat de bedoeling zou zijn en zal de Leap minder accuraat werken dan in een laptopapplicatie. Ook heb ik nagedacht over een UI waarbij er fysieke snaren achter de Leap zouden staan i.p.v. een scherm. Voor dit idee ben ik niet gegaan omdat je hierbij iets extra’s zou moeten maken om duidelijk te maken wat de bedoeling is, dit is veel beter mogelijk op een scherm door bijvoorbeeld een bewegende hand als de applicatie even niet gebruikt is. De schetsen zijn erg ruw en alleen voor een beeld te geven aan een idee, daarom heb ik in Illustrator de schets waar ik voor ga uitgewerkt.

*De uitgewerkte versie van Illustrator*

**Realiseren Gebruikersinteractie**

Ik heb het gemaakte design uitgevoerd, de interactie en audio werkt goed samen. Ook heb ik de UI getest waarna ik bruikbare info verkregen heb. Deze informatie heb ik gedocumenteerd en verwerkt in mijn nieuwe UI om zo het programma beter te laten werken en het esthetisch mooier te maken. In deze nieuwe UI worden de snaren geanimeerd bij interactie.