Pmst1T

Concept Document

Never Board

Contents

[Algemeen 2](#_Toc513448294)

[Technology 3](#_Toc513448295)

[Software 5](#_Toc513448296)

[Media 7](#_Toc513448297)

[Inleiding 7](#_Toc513448298)

[Concept omschrijving 7](#_Toc513448299)

[Doelgroep 7](#_Toc513448300)

[Website 7](#_Toc513448301)

[Stijl 7](#_Toc513448302)

# Algemeen

Het Never-Board project is een interactief digitaal bord dat thuis gebruikt kan worden om verschillende bordspelen digitaal tot leven te wekken. Het Never-Board maakt gebruik van een beamer om het speelveld op het bord te projecteren. Hiermee kunnen we het speelveld live-updaten. Een boven hangende camera ziet wanneer de hand van de speler het speelveld aanraakt. Zo kan de gebruiker zo soepel mogelijk met het spel interacteren. Onder het bord staat een mechanisme dat in staat is om via magneten de speel stukken op het bord te bewegen als het spel hierom vraagt.

Het Never-Board is in staat om klassieke bordspellen aantrekkelijker te maken voor de huidige generatie. Zo kunnen mensen van jong en oud genieten van de mogelijkheden die het Never-Board opent. Ook kunnen er compleet nieuwe bordspellen gemaakt worden met het Never-Board die voorheen fysiek onmogelijk waren. Er kunnen zelfs real time strategy games geïmplementeerd worden in het Never-Board zodat bijvoorbeeld de nieuwste Civilisation game gespeeld kan worden op tafel in de huiskamer.

Het Never-Board is voor andere bedrijven (zoals Hasbro en 99 Games) ook erg interessant. Zij kunnen in samen met ons hun bordspellen en andere familie spelletjes publiceren via het Never-Board. Wij zouden met onze eerste winsten ook zelf projecten bij indie studio’s kunnen financieren als wij zelf niet aan de vraag naar nieuwe games kunnen voldoen.

# Technology

* De I/O technieken digitale input, digitale output, PWM en analoge input zijn betekenisvol toegepast.

-Digitale output, de bewegingssensor wordt vanuit de Arduino aangestuurd met een digitale pin

-Digitale input, de Arduino ontvangt in een digitale pindata vanuit de bewegingssensor

-PWM, gebruik van een potentiometer, de waarde van de potentiometer wordt in Pulse Width Modulation doorgegeven aan de Arduino

-Analoge input, de digitale pins die ook gebruikt kunnen worden om analoge input te simuleren worden gebruikt voor de input van de potentiometer, en de potentiometer bestuurt een onderdeel van het eindproduct

* Er is seriële communicatie of een goedgekeurd alternatief voor communicatie toegepast, met berichtenverkeer in beide richtingen en op basis van een zelf opgesteld en gedocumenteerd protocol.

Er is verbinding via de seriële poort tussen de Arduino en Visual Studio, Arduino stuurt data en informatie en instructies door naar Visual Studio, en in Visual Studio kun je commando’s en instructies doorsturen naar de Arduino, die daar iets nuttigs mee doet.

* Het programma is opgedeeld in modules en herbruikbare functies.

Herhaalcode in de Arduino wordt zo veel mogelijk beperkt door het gebruik van methods. Dit houdt de code kort en overzichtelijk, en ook herbruikbaar.

* De code is leesbaar en gedocumenteerd (duidelijke naamgeving van variabelen en functies, weinig duplicatie van code).

Elke method krijgt een korte commentaar beschrijving en er is een sequentiele commentaar beschrijving in de Setup en de Loop functies, zodat duidelijk is wat in welke volgorde wordt uitgevoerd door de Arduino.

* Het Arduino programma werkt zoals verwacht. Eventuele programmafouten kunnen verklaard worden.

De Arduinocode is vooraf goed bedacht en er wordt gebruik gemaakt van herbruikbare methods. Voorderest wordt er goed van te voren gebugtest om te zorgen dat de Arduino foutvrij is.

* Er zijn andere sensoren en actuatoren gebruikt dan alleen die bij de OIT12 les gebruikt worden.

Er wordt een bewegingssensor gebruikt die geleend wordt bij de ISSD, en een luidspreker die geluiden die relevant zijn voor het eindproduct afspeelt en aangestuurd wordt door de Arduino.

# Software

Fouthandeling is overal geprogrammeerd middels exception handeling

Voor standaard problemen kunnen we exceptions goed gebruiken. In het geval van “memory out of space” of bepaalde gegevens die niet kloppen is het een handige optie om exceptions te gebruiken en zo het probleem efficiënt en snel op te lossen. Naast deze standaard problemen kan je ook andere problemen oplossen zoals een speler die een verkeerde zet doet. Dan kan je voor vrijwel elke foute zet dezelfde foutmelding geven naar de gebruiken in plaats van dat je alle foute zetten individueel moet implementeren. Ook in het geval dat er andere problemen zijn kan je gemakkelijk bekijken of een exception een goede oplossing is of niet. Voor herhaaldelijke problemen is het een prima oplossing.

Code bestaat uit minimaal 6 functionele klassen

Voorlopige klassen

* Speelstukken
* Instellingen
* Menu
* Dobbelstenen
* Arduino
* Beurten
* TBD

Om aan deze eis te voldoen moeten we voorafgaand bedenken welke klassen we nodig hebben, wat er in deze klassen moet komen te zitten en hoe wij deze gaan toepassen. Misschien moeten we een klasse opsplitsen om aan 6 klassen te komen. Het is waarschijnlijk dat we tijdens het realisatie nieuwe klassen aan moeten maken om een of andere reden, zelfs met uren planning kun je namelijk makkelijk iets over het hoofd zien.

Het vooraf gemaakte UML-klassendiagram komt overeen met later geprogrammeerde code

Voordat wij daadwerkelijk gaan programmeren moeten wij dus bedenken wat de eigenschappen van elke klasse moet zijn, en welke methodes erin moeten komen, dus wat ze allemaal precies doen.

Ook moeten we, als we erachter komen dat we nieuwe klassen nodig hebben, deze klassen ook via een UML-klassendiagram uitwerken.

Het programma maakt op meerdere plaatsen gebruik van file handeling

We kunnen voor onze proftaak file handling voor een aantal problemen goed gebruiken. File handeling kunnen we toepassen in de vorm van spelregels van de verschillende spelletjes, het onthouden van een schaakpartij door coördinaten op te slaan en op te slaan zijn. Er zullen ook nog wel wat andere mogelijkheden zijn voor het gebruik van file handeling wat betreft andere methodes zodat ze makkelijker kunnen worden gemaakt.

Er is goede scheiding tussen code en user interface (dus alle berekeningen en alle andere acties staan gecodeerd in klassen en niet in forms).

Om alle berekeningen in klassen te doen, moeten we dus ervoor zorgen dat de juiste berekeningen in de juiste klassen terecht komen. Bij dobbelstenen moeten we natuurlijk een random getal gaan maken, bij speelstukken, de locatie, kleur/teams het type speelstuk etc.

# Media

## Inleiding

Met een interactieve beamer willen wij bordspellen een nieuw leven in blazen.

## Concept omschrijving

Ons idee is om een interactief spelbord te maken. Hier op kan je verschillende klassieke en moderne bordspellen spelen. Het spelbord wordt geprojecteerd op een tafel, door het gebruik van een tracking sensor wordt deze interactief. Door het toepassen van animaties worden de klassieke bordspellen een nieuw leven in geblazen. Hierdoor worden de spellen leuker voor de jongere generatie maar blijven ze ook interessant voor de huidige generaties.

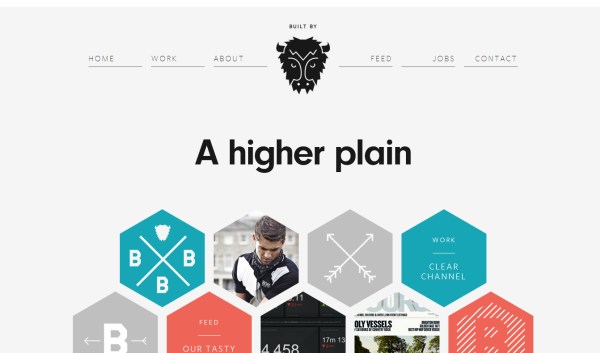
Als eerste spel dat wij zullen realiseren is schaken. Mocht er tijd over zijn zouden wij hier e.v.t. nog andere spellen aan toe kunnen voegen. Het is later de bedoeling dat er een keuze gemaakt kan worden tussen verschillende spellen.

## Doelgroep

Onze doelgroep is voornamelijk gezinnen met thuiswonende kinderen. We hebben er ook over nagedacht om ons spel in andere omgevingen zoals restaurants toe te passen, maar dit is voor nu niet de door ons gekozen doelgroep.

## Website

Als toevoeging aan het spel willen wij een website gaan maken. Deze website willen wij gaan gebruiken om het spel te promoten, maar ook voor handleidingen van de verschillende spellen. We willen er voor zorgen dat de website ook optimaal in gebruik is voor smartphone & tablet.



## Stijl

We gaan voor een strakke stijl, zodat het duidelijk is en simpel te gebruiken is voor alle leeftijden. Ook omdat onze plannen erg modern zijn past een strakke huisstijl goed bij het concept