Bachelorproef begeleidend document

abstract

Inhoudstafel

inleiding

van de 3 voorstellen omtrent een bachelorproef die ik indiende, werd geen enkele goedgekeurd. het uiteindelijke idee is kwam naar boven tijdens een aansluitende lunch. Een van de docenten vernoemde een project dat fysieke voorwerpen en projectie combineerde tot een interessant geheel. In het gesprek dat volgde werd hier al snel een educatief doel aan gekoppeld. Fysieke componenten, een systeem om zogenaamde tags uit te lezen en te volgen, en een narratief. Het doel: kinderen zin geven in programmeren.

Kinderen krijgen in dit project een verhaal voorgeschoteld, in de vorm van een boek. Bepaalde pagina's zijn leeggelaten. Op deze pagina is het de bedoeling dat het kind een stukje code legt om het hoofdpersonage verder te helpen, en zo het verhaal te laten verdergaan.

Doorheen de maanden heeft dit project vele vormen aangenomen. Zo heeft vooral het projectieoppervlak vele vormen gekend, van een tafeloppervlak naar een fysiek boek. Ook zijn de codeblocks (kleine componenten die de kinderen moeten inzetten om een doel te bereiken) van een plexiglazen blokje naar een speelkaartformaat gegaan.

opstellen van de onderzoeksvraag

onderzoeksvraag

> *is het voor kinderen aantrekkelijker code te leren schrijven wanneer deze tastelijk elementen bevat, en er gebruik wordt gemaakt van een personage waarmee het kind zich kan identificeren (of empathie mee kan hebben)?*

onderzoeksafbakening

dit project wordt uitgewerkt voor kinderen in een “coderdojo-omgeving”. dit wil zeggen dat de kinderen altijd in een groep aanwezig zullen zijn, onder leiding van een mentor. de aantallen gaan van 8 tot 15 kinderen. Dit project kan ook gebruikt worden in een klasomgeving, maar niet voor een aantal consecutieve lessen. men beperkt dit best tot één of twee lessen, afhankelijk van de grootte van de groep.

het eindresultaat zal een proof of concept zijn dat weergeeft hoe het volledige systeem zal werken. In eerste instantie zal de gebruiker op 2 punten in het verhaal code moeten schrijven, en hiermee het hoofdpersonage verderhelpen.

Inhoud bachelorproef zoals vermeld in aanvraag onderzoek naar de doelgroep en mogelijke hindernissen die hieraan verbonden zijn het maken van een interactief verhaal dat de brug maakt tussen verschillende code-challenges uitwerken van fysieke blokken code, waarmee kinderen snel overweg kunnen (deze worden ingelezen door Computer Vision Eindresultaat bachelorproef zoals vermeld in aanvraag een oppervlak (tafel of muur) waarop kinderen het verhaal kunnen volgen en op gegeven momenten in dit verhaal het hoofdpersonage moeten helpen door stukjes code te schrijven. fysiek aanwezige blokken code, die op een intinuitieve manier aan elkaar gelinked kunnen worden. een proof of concept van een kortverhaal dat de “leermomenten” aan elkaar lijmt, waarmee de focus van het kind behouden blijft

subvragen van het onderzoek

om dit onderzoek uit te voeren, moesten een aantal vragen eerste opgelost worden.
dit ging vooral om de motorische en cognitieve eigenschappen van de doelgroep, maar eveneens rond de rol die storytelling kan spelen in het leerproces.

verder moesten een aantal functionaliteiten onderzocht worden.

- Welke software en hulpmiddelen zijn er reeds in omloop
- hoe staat men tegenover het aanleren van code aan kinderen
- welke skills zijn vereist
- welke skills worden gestimuleerd door code
- hoe kan men kinderen motiveren een doel te bereiken

eindresultaat

aanzet

een muur waarop kinderen kunnen leren coden door middel van een verhaal. het verhaal wordt verteld in verschillende stappen, en aan het begin van elke stap kan een kind (alleen of in groep) een stukje code schrijven om het hoofdpersonage te helpen. Dit kan hij/zij doen door magneetjes waarop codestructuren staan aan of onder elkaar te hangen. dit wordt in real life uitgevoerd/ uitgevoerd wanneer het kind op play drukt/de playmagneet in het vakje hangt.

tips kunnen gegeven worden door deze te projecteren op de juiste plaats.

er kunnen ook inputs gegeven worden door middel van een controller ed. meerdere devices kunnen later toegevoegd worden.

een kort verhaal, met twee momenten waarop er code kan ingevoerd worden. dit gebeurd op een tafel, maar wel met magneten. alles wordt van bovenaf geprojecteerd. CV kan magneetjes analyseren, en de functie erop uitvoeren.

resultaat

omschrijving van werkelijk resultaat

hypothetisch verdere uitwerking

what we could've done

deze onderzoeksvraag veranderde doorheen het process, na gesproken met Dries De Roeck en Beyond.io. Ook Pieter Feys leverde een aantal interessante documenten doorheen het proces.

Onderzoek

nut van programmeren

In 2014 gaf de EU een rapport uit dat het integreren cvan programmeren in het lager onderwijs bespreekt. Het gaat hier vooral over landen die dit reeds doen (zoals estland) aar eveneens over landen die dit van plan zijn in de nabije toekomst (waaronder België). Ook het nut van programmeren in een schoolomgeving komt hierin aan bod, met het oog op de toekomst.

Mitch Resnick (MIT, medeoprichter van Scratch) vergelijkt programmeren met schrijven; slechts weinig van de kinderen worden professioneel schrijver, maar kunnen schrijven is steeds handig. Hij vindt dat dit ook geldt voor code. Men moet niets computergerelateerd kiezen als beroep om de opgedane kennis te gebruiken. Programmeren stimuleert volgens Mitch het creatief denken, het systematisch redeneren en het samenwerken in teamverband. Verder kan het kind ideeën naar buiten brengen op deze manier.

Dit inzicht komt terug bij Derek Cabrera. Hij beschrijft in zijn TED-talk dat sommige kinderen het creatief denken verleerd hebben. Hij geeft hierbij het voorbeeld van Lego. Dit werd vroeger verkocht in dozen, zonder instructiehandleiding, terwijl deze vooral verkocht wordt in dozen waarmee men enkel de handleiding kan volgen, doordat de blokjes slechts 1 enkel model toelaten. volgens hem verleren kinderen hierdoor het creatief denkproces (metacognitief denken). Dit soort denken houdt in dat men gemakkelijk nieuwe problemen kan oplossen, en ongestructureerde problemen can structureren en er iets zinvol mee kan doen.

algemeen nut van algoritmisch denken

Als een kind een probleem oplost via een gedefinieerde en eindige set stappen, kan men spreken over algoritmisch denken. deze vorm van programmeren vindt men terug bij applicaties die zich vooral richten op jongere kinderen (eerste 3 jaren van het basisonderwijs).

“The ability to propose a step by step solution to a problem (an algorithm), consisting of a finite, clearly defined set of simple (definite, unambiguous) steps.” -EU rapport-

In deze stap wordt gebruik gemaakt van software die de manier van denken achter code weergeeft. Het kind hoeft in dit onderdeel nog geen feitelijke code schrijven, maar krijgt deze voorgeschoteld als blokken code. Een van de veelgebruikte programma’s op dit vlak is Scratch (later in dit werk uitgebreid besproken). Het abstracter maken van code, door gebruik te maken van blokken bestaande code, maakt de drempel minder hoog. Vaak worden de bruikbare blokken ook stapsgewijs aangebracht, waardoor de leercurve onder controle blijft. Toch maakt dit het geheel niet minder beperkt. In veel van deze applicaties kan men gebruik maken van een webcam, microfoon en andere multimediale apparaten. Door gebruik te maken van een kleurcode die deze blokken onderscheid van andere bouwblokken, blijft het voor de gebruiker zeer overzichtelijk.

Programming

Dit is vooral bedoeld voor kinderen en volwassenen die het algoritmisch denken onder de knie hebben. Deze stap wordt door het rapport van de EU gedefinieerd als volgt:

“Het kunnen realiseren van een algoritme in machinetaal, de stappen die genomen zijn kunnen interpreteren, [...] het kunnen compileren, runnen en debuggen van een programma, alsook het kunnen identificeren en herbruiken van designpatterns”.

Dit betekend dat men meer met feitelijke code werkt dan met metaforische software. Doordat deze vorm veel complexer is en de nodige kennis vereist alvorens men ermee kan beginnen, is dit ongeschikt voor jonge kinderen. Dit ligt tevens ook aan het debugprocess, dat de nodige inzichten vereist. De leercurve van programmeren is ook veel stijler dan die van algoritmisch denken. Hierdoor kunnen kinderen gemakkelijk gedemotiveerd geraken en afhaken.

---

concentratie

Concentratie is een belangrijk onderdeel in programmeren. Het voorgestelde probleem moet volledig ontleed worden in kleine acties.

onderverdelingen van programmeren

algoritmisch denken

in deze fase of onderverdeling leert het kind vooral problemen opbreken in verschillende stappen. het leert bijvoorbeeld een minidoolhof oplossen door middel van eenvoudige commando's (draai links, draai rechts, vooruit). Deze vorm is vrij beperkt in mogelijkheden, maar geeft het jonge kind wel de nodige inzichten en leert het probleemoplossend denken. deze vorm is zoals eerder vermeldt zeer geschikt voor zeer jonge kinderen. men kan dit terugvinden in Ipad-applicaties die gericht zijn naar 3 jarigen. voor oudere leeftijden is dit geen uitdaging, waardoor dit minder geschikt is. Het kan echter wel dienen als eerste stap, waarbij alles zeer snel duidelijk is.

metacognitief denken

in deze stap komen de eerste programmeringsstructuren naar boven. het kind leert bijvoorbeeld werken met een if else structuur, onder verscheidene vormen. soms zijn deze zeer variabel opgesteld, op andere momenten kan dit tevoorschijnkomen onder vastgelegde blokjes. afhankelijk van de doelgroep kan men hiertussen kiezen. Deze vorm van programmeren is echter bijzonder geschikt voor de beoogde doelgroep. de nadruk ligt hier namelijk op de denkwijze achter de code, en de syntax is niet heel belangrijk.

flow programming

bvb: puredata. in deze vorm van programmeren zal de programmeur kleine blokjes toevoegen (die een functie of variabele voorstellen), en deze met elkaar verbinden. deze manier van programmeren is eveneens vrij simpel aan te leren en op kleine schaal relatief gemakkelijk, het nadeel is dat programeerstructuren moeilijker te implementeren zijn.

programming

dit omvat programmeren onder meest pure vorm. men spreekt hier over talen zoals C++ en java, waar de syntax een zeer belangrijke rol speelt. Hierdoor is dit veel minder geschikt voor kinderen. De nadruk in deze vorm van programmeren ligt op het correct schrijven van de code, in plaats van achter de logica erachter. Het merendeel van de tijd zal de programmeur hier bezig zijn met het corrigeren van typfouten.

Andere applicaties

Scratch

Scratch is een initiatief van MIT, onder meer van Mitch Resnick. Scratch kan gebruikt worden voor het animeren van een verhaal, het interactief maken hiervan, het recreëren van games, polls, tutorials, artwork en veel meer. In de nieuwe versie zal het tevens ook mogelijk zijn te werken met een webcam en de microfoon.

nut van storytelling

fundamentals of digital storytelling

de focus mag niet enkel liggen op het medium, er moet ook gedacht worden aan de manier van interactie, aan de hoeveelheid ervan, of het aantal variabelen waar de lezer mee te werk kan gaan. Men neemt hier vaak het voorbeeld van games. Deze werken eveneens met een narratief, dat vaak de input van users nodig heeft om te vorderen. Omdat dit proces zeer persoonlijk is, is het moeilijk een algemeen beeld op te stellen hiervan. de ervaring van de interactiviteit varieert van speler tot speler.

De meeste focus in dit stuk ligt op de interactiviteit tussen speler en narratief. Ik heb lang gespeeld met het idee het verhaal te laten variëren naarmate andere stukken code neergelegd werden, maar dit bracht met zich mee dat het fysieke boek meerdere delen moest bevatten. Naarmate de speler andere code neerlegde zou hem/haar meegedeeld worden door te gaan naar een bepaalde pagina, waar het verhaal dan een andere wending nam. Dit is mogelijk, maar vergt extra tijd. Vooral omdat het verhaal dan via een boomstructuur verloopt, en verschillende zijverhalen moeten geschreven worden.

digitale verhalen (zoals games en interactive stories) hebben het voordeel abstracte onderwerpen meer toegankelijk te maken voor verscheidene doelgroepen. zo zijn er bijvoorbeeld veel youtube kanalen die gebruik maken van een narratief om ingewikkelde wetenschappelijke onderwerpen op een klare manier uit te leggen. Ze kunnen ook helpen leerstof aantrekkelijker te maken, en zo minder gemotiveerde studenten toch aan te zetten tot leren. Het gebruik van de creativiteit van de gebruiker en zijn diepere gedachten creëren een sterker gevoel van doel in het leren (Hull and Katz, 2006).

verhaal

zelf schrijven

In eerste instantie dacht ik het verhaal zelf te schrijven. dit om het zo volledig mogelijk te doen aansluiten bij de noden van het programma. Beyond.io raadde me aan het verhaal eerst te schrijven alvorens te beginnen programmeren.Dit om te voorkomen dat ik anders functionaliteiten zou ontwikkelen die achteraf onnodig zouden zijn.

Omdat ik zelf niet de capaciteiten bezit een kinderverhaal te schrijven, deed ik beroep op andere schrijvers. Ik contacteerde onder andere Marc De Bel en Antony Horowitz. Marc De Bel antwoord positief, en stond toe dat ik een van zijn verhalen mocht gebruiken en zelfs aanpassen. Dit Leek echter geen goede optie, omdat zijn verhalen zowel te lang als te gevarieerd zijn om een degelijke programmeer oefening op te leveren.

Verder kon hij mij ook niet helpen aan digitale versies van tekeningen, vaak gebruikt in zijn boek. Daarom contacteerde ik Het Geluidshuis.

geluidshuis

het geluidshuis is een productiehuis dat vooral luisterverhalen (de zogenaamde hoorspelen) uitbrengt. Op het moment dat ik hen contacteerde zaten de kinderen volgens het concept nog aan een grote tafel. Het gebruiken van een hoorspel loste het kijkhoekprobleem op. Dit wil zeggen dat de beeldende voorstelling van het verhaal niet naar een bepaalde richting kon wijzen, omdat kinderen errond zaten. Het luisteren lostte dit probleem op.

Ik maakte met het geluidshuis en afspraak op 22 januari, in de voormiddag. Na een korte wachttijd kon ik met hen de volledige situatie bespreken. Al snel werd het duidelijk dat het moeilijk zou worden auteursrechtelijke toegang te krijgen tot de werken. Een nieuw werk maken ging zeker ook niet lukken, daarvoor waren de productiekosten veel te hoog. Er werd me wel hulp beloofd moest ik zelf een verhaal opzetten. Ik kon rekenen op intellectuele bijstand, maar infrastructuur was te duur om te mogen gebruiken.

Werken met een hoorspel bracht helaas ook nieuwe problemen met zich mee. Onder andere dat de verhalen lang waren, en kinderen op elkaar zouden moeten wachten alvorens verder te gaan met het verhaal. Daarom besloot ik niet verder te werken met het geluidshuis.

open sourcing

in plaats daarvan zou ik lesgevers, schrijvers en derden de mogelijkheid geven zelf een eigen verhaal in te voeren. Dit zou via een wegomgeving mogelijk zijn. De levels zouden ingevoerd kunnen worden via een webplatform, en zijn downloadbaar door het systeem.

tag recognition

de tags hebben een cruciale rol in het geheel. zowel voor het herkennen van het boek, de gebruikte kaartjes en posities van deze kaartjes. Daarom moest dit stukje software bijzonder efficiënt kunnen lopen, en de nodige ID's als output hebben.

Ik ging natuurlijk eerst op zoek naar een soortgelijke plugin, die me verder zou helpen. De gevonden systemen waren helaas slechts in staat met eigen tags te werken. het aanpassen om bijvoorbeeld enkel bepaalde kleuren te tracks, het beeld negatief te maken, of de positie in een bepaalde vorm weer te geven zou teveel tijd hebben genomen. Verder was het ook een bijzonder zware plugin, omdat deze heel het cameraveld analyseert. Ik besloot zelf een eigen plugin te schrijven dit dit waar kon maken.

eerste versie

Eerlijk is eerlijk, mijn eerste versie liep zo mogelijk nog minder efficient dan de eerder vermeldde plugin. Ik maakte gebruik van een systeem dat de volgende stappen onderneemt:

1. de vorige positie als centrum voor de huidige scan
2. 360 graden rond deze positie op verscheidene afstanden de kleur bepalen
3. indien het punt donker genoeg was, werd dit opgeslagen in een lijst.
4. De lijst wordt overlopen om het verste punt te bepalen.
5. tussen deze vier punten werd dan geïtereerd om de desbetreffende code te bepalen.
6. de rotatie werd opgeslagen in een object.

Natuurlijk namen al deze stappen bijzonder veel tijd in beslag. Ik moest dus op zoek naar een efficiëntere versie die dezelfde output kon geven.

Tweede versie

de tweede versie ondernam veel van deze stappen, maar had slechts een beperkt scan oppervlak, zodat de benodigde aantal calculaties minder hoog was.

Finale versie

sinds ik toch al werkte met een beperkt oppervlak, leek het me onnodig vanuit het midden te beginnen scannen. Bij gevolg verliep de scan nu in een veel efficiëntere vorm.

Elk quadrant wrest apart behandeld. Er wert van buitenaf naar binnen gescanned, tot er een eerste donkere punt werd gedetecteerd. op dit moment wordt dit opgeslagen als verste hoek, en werd dit quadrant verder genegeerd. Deze actie alleen haalde de nodige processor kracht al met een 20tal percent naar beneden. Verder verder slechts een scan uitgevoerd als de camera detecteerde dat de kleur van het veld drastisch veranderd was.

code