Interactive coding wall

onderzoeksvraag

- logiblokken
- coderdojo
- lego mindstorms / blokken**
- max msp
- scratch
- noflo.js
- *icoontjes -> versterking*
- rode knop
- *hardwaremodulles*
- **dries de roeck -> coderdojo**

---

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)?

---

bepalen van:

- doelgroep
- motorische/taalpoblemen met deze doelgroep
-

---

first base

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.

second base

hardware modulles

third base

in deze stap wordt touch geïntroduceerd, hetgeen het kind toelaat variabelen in de code te veranderen. dit gebeurt door naast de magneet een plus- of minteken weer te geven.

---

noodzakelijk onderzoek

magneetlocatie

uitzoeken waar magneten hangen via CV of andere methoden. eens deze gelocaliseerd kunnen worden; is het mogelijk vlot uit te lezen wat erop vermeld staat?

uitlezen met CV

interactive story

het vinden van een storyline die interessant is voor kinderen, en zowel jongens als meisjes aanspreekt. het moet een verhaal zijn dat in verschillende stappen verloopt.

- robot
- zie "her"

codetaal

het (uit)vinden van de juiste codetaal, die in tsaat is errors weer te geven, en live te renderen. dit is nodig om direct feedback te kunnen geven. **minicompiler**

---

beslissingen

uitlezen van een magneet

oplossingen

- via qr
- via een barcode
- **via een kleurcode bovenaan de magneet**
- via ComputerVision

mogelijkheden

- **selectief plakvlak**

- **werken met een grid**
- detectie van plaatsing, uitlezing via een van bovenstaanden

compiling

oplossingen

- eigen codetaal
- **only one solution**
- modulaire code opzoeken
- errorweergave
- **minicompiler**

Onderzoeksvragen

- 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

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

Context

doordat deze sessies begeleid worden door een mentor of leerkracht, hebben de kinderen iemand waarbij ze terecht kunnen met vragen. Hierdoor kan men de geboden hulp limiteren tot het nodige, zonder de kinderen een feitelijk antwoord voor te schotelen.

Waarom

In een rapport van de eu, uitgegeven in 2014, wordt uitvoerend gesproken over het integreren in lager onderwijs. Men bespreekt hierin een aantal landen die dit reeds doen (estland,…) en een aantal die dit van plan zijn in de nabije toekomst (waaronder Belgie). Hierin wordt tevens ook gesproken over het nut van programmeren in een schoolomgeving en de toekomst die eraan gebonden wordt.

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 talk dat sommige kinderen het creatief denken verleerd hebben. Hij geeft hierbij het voorbel van Lego. Dit werdt vroeger verkocht in dozen, zonder instructiehandleiding, terwijl deze vooral verkocht wordt in dozen waarmee men enkel de handleiding kan volgen. 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

het aanleren van code laat zich opdelen in twee onderdelen. Het eerste onderdeel (algoritmisch denken) kan vergeleken worden met metacognitief denken. het tweede onderdeel, dat meestal het eerste opvolgt, beïnvloed dit ook, maar dit is niet het hoofddoel.

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. 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.

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 stijlen dan die van algoritmisch denken. Hierdoor kunnen kinderen gemakkelijk gedemotiveerd geraken en afhaken.

onderverdelingen

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.

installatie

korte beschrijving

het kind krijgt een verhaal voorgeschoteld waarin het op twee punten een uitdaging moet vervullen. Hij/zij moet dit doen door een ministukje code in te geven met behulp van fysieke blokjes waarop functies ed staan ingevuld. in de eerste stap gaat het om een stuk lineaire code, in de tweede stap moeten ze een algoritmische puzzel leggen. de kinderen werken samen aan 1 stuk code (in eerste instantie) of leggen elk een stukje code dat dan gezamenlijk wordt uitgevoerd.

---

end of readable text

---

wat moeten ze kunnen

- hoe denkpatronen interpreteren
- storyline -> op basis van een bestaand kinderboek?
- minicompiler -> slechts beperkt aantal blokjes
- woordenschat scratch -> vertalen en overnemen
- aan elkaar snappen -> magneetjes langs de rand
- lineair duidelijk onderscheid tussen algoritmisch -> andere blokjes
- aanpassingen op vlak van beslissingen in het verhaal

- fysieke startknop
- avi leesmethodes
- gecombineerde leesniveaus overnemen naar code?
- molletje -> to space -> stijn favourited this

- chris o’shea

- karim amrani

- game -> tegen elkaar? / sandbox
- story en code simultaan
- flexibel bouwen

---

workflow:

1.a basiscode voor het interpreteren van blokjes (modulle? plugin?) 1.b uitproberen van code via legoblokjes

2.a mini interpreter 2.b storyline uitwerken

to code;

adjustable board size story uitlezen coding -> dif story, make lib

- fysieke boek
- ouders minder bezorgd
- geen pagina switch probleem
- nieuwe markt
- lees gemak
- gebruik van bestaand boek
- mark de bel

interpreter

composite design pattern

codeblock

var voorwaarde fysiek console functie for if else switch

paper on table, and drawing on it?

---

changing format (again)

Tijdens Resonate in Belgrado, heb ik het privilege gehad een gesprek te kunnen hebben met Zach Lieberman. Deze raadde me aan in plaats van blokjes over te schakelen naar een kaartspel. Dit heeft een aantal gevolgen:

1. de code's die op het kaartje moeten komen kunnen groter gemaakt worden.
2. de ruimte rond de tag wordt wit, waardoor dit vlotter uit te lezen valt.
3. naargelang de stijl van illustraties kan dit kaartje ook aangepast worden aan de

gangbare stijl.
4. het gehele project wordt een stuk opener. Mensen met kennis van programmeren kunnen deze took gebruiken om hun of andere kinderen gemakkelijk en zonder extra middelen te leren programmeren.
5. (indien tijd over) er kan misschien gewerkt worden met een soort pixelated vorm van illustreren. zolang de hoeken aangeduid zijn, kan dit ook nog steeds uitgelezen worden. dit moet natuurlijk appart definieerbaar zijn.

deze verandering probeer ik vanavond nog doorgevoerd te krijgen in het systeem.

---

own interpretor

designing a new code language

the entire system contains 4 main classes. these classes have subgroups and the function ones have a system of functions. the main goal is to be able to code in a sandbox, while the code is being read out of an array of id's. this is necessary to be able to work with a set of physical blocks.

one of the solutions might be to "build" a number of classes, which are linked to the earlier array, to create a total function, which will be executed at the end. after that, these variables can be changed while running, and the function should adapt itself.

to be able to do that, it might be better to run through the array, and linking the id's to a class or function. this could solve the if-else structure problem. This problem states the program should now before what the end of the if/else will be, before being able to execute it.

in short: the program runs over the array the first time, grouping if/else structures, while and for loops, and putting those into different classes. if it is an if/else structure, these classes will save the array elements in a corresponding array, to execute these later on, or skip them if necessairy. if the id means its an for loop, the class will save a variable, to be able to go over a set of id's the predefined number of times. in this first loop, it wil try to figure out if the set it saved contains any other set, to make separate classes for these actions.this brings along the problem we should save the number of if structures, to know where to stop. the program goes over the array a second time, this time, it sets the variables, and connects to the physical blocks (if possible) the program executes the array now, being able to perform actions and if else structures.

the first groups is a set of vars. these are dynamically added and changed throughout the array. this means this class has a set of vectors, which can contain some kind of variable. the main issue here is the addressability of the variables. maybe it is possible that this class has two main variables, and works as a container for a certain variable.

meeting beyond.io 15/05

- uitleenmanier
- overloop documenteren
- verhaalintroductie
- helpfunctie
- stap voor stap
- groter projectievlak
- animatieopbouw
- sound helpt om acties aan te duiden, feedback,
- kaartjes in het boek
    - tutorialgewijs
- gestaag opbouwen
- uitschrijven van tweede opdracht
- duidelijk verhaal --> testen
- robbie de robot
    - storyboard
-