abstract

inleiding

De inspiratie voor dit onderzoek komt oorsprongkelijk van een van de installaties op het brusselse evenement "Nuit Blanche". Een van de docenten liet een video zien waarin een connectie werdt gemaakt tussen fysieke componenten en een projectie. 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 van het project: kinderen zin geven in programmeren.

Kinderen krijgen in dit project een verhaal aangeboden, 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.

Ik kreeg bij dit poject hulp van een aantal personen, Waaronder mijn ouders, Wouter Van Den Broeck, Pieter Feys, Zach Lieberman, en Beyond.io.

--> problem --> one page --> i am going to --> bullet list of contributions --> paper provides claim

---

de onderzoeksvraag

onderzoeksafbakening

probleemschets

Er zijn reeds een aantal applicaties met het doel kinderen te leren programmeren. Dit gebeurd vaak in de vorm van een spel of kleine opdrachten. Deze applicaties missen helaas vaak een doel, naast het ontwikkelen van een klein spel. Een narratief kan ervoor zorgen dat kinderen langer gemotiveerd zijn, en minder snel afhaken omdat opdrachten op elkaar beginnen lijken. In deze applicaties worden verschillende programeeronderdelen ook duidelijk een voor een aangeboden, in plaats van dit op een minder opvallende manier te doen. Elke opdracht is ook verschillend van de vorige. Het kind doorloopt dus een aantal opdrachten, en weet na de laatste opdracht elk onderdeel correct te gebruiken. Het doel van het narratief is deze onderdelen aan elkaar te binden, zodat het geheel een grote opdracht wordt, zonder ht kind "opdrachten" op te leggen.

De doelgroep van dit project ligt vooral tusen 8 en 12. Dit is een leeftijdklasse waarin lezen

vaak niet veel moeite vraagt, en de focus dus kan liggen op het schrijven van de code.

doel van het onderzoek

Het onderzoek dient duidelijk te maken of een narratief effectief kan bijdragen aan een leukere leerervaring. Dit via een hands-on ervaring, en een empathische band met het hoofdpersonage van het verhaal. Het resultaat van dit onderzoek zal ook een POC (proof of concept) bevatten in de vorm van een applicatie die een extra dimensie geeft aan een verhaal, en het toestaat meerdere verhalen met dezelfde appicatie te kunnen gebruiken.

onderzoeksvragen

centrale vraag

Hoe kunnen we kinderen aanzetten tot coderen in een narratieve omgeving, en aan welke voorwaarden moet de digitale omgeving voloen om succesvol te zijn.

deelvragen

De onderzoeksvragen laten zich indelen in een aantal categorieen. De eerste categorie bevat vooral vragen omtrend het leerproces van het kind, en welke elementen bij kunnen dragen aan een betere, leukere leerervaring. Het gaat hier dan vooral over de volgende vrgaen:

- heeft handelend leren een positieve invloed, en hoe kan dit geïntegreerd worden
- heeft empathie effect op leren
- leren in klasverband; welke randvoorwaarden hangen hieraan vast
- op welke manier kunnen kinderen samenwerking in deze applicatie

Verder wordt het aanleren van programmeren aan kinderen in vraag gesteld.

- waarom zou je kinderen leren programmeren
- welke invleoden heeft programmeren op kinderen
- hoe kan dit technisch gerealiseerd worden

Naarmate het onderzoek vorderde, kwamen een aantal andere onderzoeksvragen aan bod. Deze gaan vooral over de functionaliteit van het programma, maar eveneens over de interactie die het programma aanbied tegenover de gebruiker

- welke software en hulpmiddelen zijn reeds beschikbaar
- hoe staat men tegenover het aanleren van programmeren aan kinderen
- zijn er motorische beperkingen bij de doelgroep
- welke stimulus kan programmeren bieden
- hoe kan een kind gemotiveerd worden een doel te bereiken

onderzoeksresultaat

Het vooropgestelde resultaat zal een verhaal bevatten, dat op bepaalde momenten het kind zal vragen een stukje te programmeren alvorens verder te gaan met het verhaal. Dit proces wordt ondersteund met een projectie en de nodige tips. Het aantal stukjes code dat hiervoor

gebruikt kunnen worden zal in dit onderzoek beperkt worden. Het programma zal gebruik maken van empathie en andere technieken om het kind te motiveren.

---

onderzoek

---

programmeren

In 2014 gaf de EU een rapport uit dat het integreren van programmeren in het lager onderwijs bespreekt. Het gaat hier vooral over landen die dit reeds doen (zoals estland) waar 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 van zowel de economie (meer programmeurs), maar belangrijker nog, met het oog op het kind.

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 momenteel 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.

21st century skills stelt dat onze maatschappij van een industriele samenleving naar een kennis-samenleving veranderd is. Hun doel is het onderwijs aan te passen om beter aan te sluiten bij dat beeld van de 21ste eeuw. De europese unie is hier reeds mee bezig, samen met bv unesco en een aantal amerikaanse organisaties. Nederlandse beleidsmakers zijn hier zeker reeds in betrokken in de context waarin ze werken. Deze discussie wordt helaas nog niet helemaal doorgetrokken naar het onderwijs. Deze skills zouden even belangrijk moeten zijn dan wiskunde of nederlands. Joke Voogt is voorstander van een publiekdebat die deze discussie opentrekt waarin ook ouders en studenten hun mening kunnen uiten.

digitale samenleving

de digitalisering van de samenleving is enorm merkbaar in de tewerkstelling. Het aantal beroepen waarin productiewerk vericht wordt, daalt snel, en de beroepen waarin men

creatief moet nadenken en probleemoplossend moet werken, zit in de lift. In die laatste categorie is een degelijke ITkennis natuurlijk noodzakelijk. Vooral voor communicatie, maar eveneens om de efficientie te verhogen. Daarom is het belangrijk dat we dit al van jongsafaan kunnen meegeven.

21st century skills

De term 21st century skills heeft betrekking op een geheel van vaardigheden die nodig zijn te functioneren in de huidige samenleving. Deze samenleving wordt onderverdeeld in een aantal groepen. De belangrijkste groepen zijn "digital natives" en "digital immigrants".

De digital natives zijn in dit onderzoek de belangrijkste doelgroep. Zij zijn mee opgegroeid met de technologie, en ontdekken nieuwe ontwikkelingen en toepassingen, volledig onbevangen. Deze groep heeft weinig tot geen moeilijkheden met het gebruiken van nieuwe technologie, en zijn hier snel mee weg.

Tot de groep van "digital immigrants" behoren de volwassenen die de technologie hebben zien groeien. Sommigen zijn snel weg met vernieuwing, anderen twijfelen eerder alvorens deze te omarmen. Ook deze groep gebruikers leert omgaan met de nieuwere technologieën, maar hebben soms meer moeite deze vlot te integreren.

Het grote verschil tussen deze twee groepen is het gemak waarmee men overschakeld. Dit is vooral merkbaar in de manier van interactie. Terwijl de immigrants vooral vasthouden aan mail en formelere communicatie (ook face-to-face), maakt de groep van digital natives gebruik van sociale media zoals whats-app en facebook, en schakeld zeer snel over wanneer een nieuwe vorm beschikbaar is (vb: snapchat).

21st century skills worden onderverdeeld in een aantal vaardigheden. Deze worden vaak ondersteund door een degelijk gebruik van ICT. We spreken hier bijvoorbeeld over vaardigheden zoals:

- samenwerking
- kennisconstructie
- ICT gebruik voor leren
- probleemoplossend denken en creativiteit
- planmatig werken

De nadruk in dit onderzoek zal liggen op het probleemoplossend denken en creativiteit. Dit omdat deze vaardigheid gestimuleerd wordt door programmeeractiviteiten en de denkwijze die hiermee gepaard gaat.

Algoritmisch denken

Algoritmisch denken (ook beschreven als algoritmisch denken) betekend letterlijk "denken over denken" (meta=over, cognos=denken). Algoritmisch denken wordt door de EU omschreven als "de mogelijkheid een probleem op te lossen via een stap-voor-stap oplossing (algoritme), waarin deze bestaat uit een eindige set duidelijk gedefinieerde stappen".

Dit soort denken wordt vaak gebruikt tijdens het programmeren. Het komt erop neer dat de programmeur (in dit geval ook de gebruiker) een probleem dient te verdelen in set kleinere deelproblemen, om daarna deze deelproblemen op te kunnen lossen via een stappenplan. Men moet hierbij natuurlijk ook rekening houden wat de beginsituatie is, en het einddoel.

Het aanleren van deze vaardigheid gebeurd reeds in de alternatieve programma's. Onder andere Scratch leert dit aan door gebruik te maken van een serie uitdagingen die het kind moet doorlopen. Tijdens dit proces leert hij het probleem op te delen in kleine acties. Nadat de gebruiker alle stappen heeft doorlopen, kan deze aan de slag met alle blokjes die Scratch bevat, om zelf creaties te maken.

Scratch kan een opstap zijn naar programma's als java, visual basic of javascript, zonder de leerling al direct te confronteren met het zoeken naar bugs als komma's of aanhalingstekens. Dit kan namelijk een grote drempel vormen, omdat en programmeur veel tijd doorbrengt een code te debuggen. Het werken met voorgemaakte blokjes neemt deze drempel weg.

programming

Men kan het natuurlijk niet hebben over programmeren zonder de meest pure vorm te bespreken. Programmeren kent vele definities. de volgende is diegene die de EU gebruikt in zijn rapport:

“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 soort programmeren heeft en stijle leercurve. Het duurt even vooraleer men een eigenlijke output heeft, en het aanleren van de syntax neemt de nodige tijd in beslag. Kinderen kunnen hierdoor gedemotiveerd geraken, en haken af als gevolg. Daarom wordt in dit project gebruik gemaakt van een abstractere vorm hiervan. Een vorm waarin de syntax niet zichtbaar is voor de gebruiker, en deze ook niet aan de nodige haakjes, komma's of punt-komma's moet denken.

Programmeren laat zich onderverdelen in een aantal categorieen.

algoritmisch denken

dit omvat het programmeren zonder eefectief code te schrijven. Enkel standaard acties zijn hierin opgenomen. Variabelen en andere structuren komen hierin niet aan bod. Het oplossen van een probleem (meestal het bereiken van een bepaalde plek op het speelveld) kan hier opgelost worden door een opeenvolging van stappen zoals "stap naar rechts", "stap naar boven", en meer van dit sort blokken. Zoals eerder vermeld is deze vorm zeer geschikt voor jonge kinderen. Ze kunnen hier snel mee aan de slag en zien zeer snel resultaat. Dit is echter wel een ideale manier om snel en duidelijk een eerste probleem onder te verdelen in kleine stappen, en de drempel naar het volgende niveau is zeer laag. Voor ouderen is dit echter niet heel uitdagend, omdat de gelijkheid met het volgende level vaak te groot is en het als bandwerk aanvoelt.

metaforisch programmeren

In deze stap (die meestal volgt op de vorige) komen de eerste programmeerstructuren naar boven. Kinderen en volwassenen maken kennis met een "for loop", en een "if else" structuur. Ook variabelen komen hier aan bod. De gemakelijkste manier om deze vorm te gebruiken is door de code onder te verdelen in kant en klare blokjes. Deze kan de gebruiker dan aaneenschakelen en een groter geheel vormen. In veel applicaties kan deze vorm voorgesteld worden als een efficiënter alternatief voor algoritmisch denken.

flow programming

Flow programming wordt voorgesteld in progammas zoals pure data. De gebruiker verbind blokjes die een actie of voorwaarde voorstellen met elkaar, om zo het programma uit te voeren. Deze vorm laat echter slechts moeilijk de implementatie van programmeerstructuren toe.

programming

Dit is de meest bekende vorm van programmeren. In deze vorm is de syntax van groot belang, en vele programmeertalen maken gebruik van deze vorm van programmeren. De nadruk ligt in deze vorm op ht correct aanroepen van de juiste functies, en er moet vooraf heel hard nagedacht worden over de manier van implementatie. Deze vorm neemt ook de nodige tijd in beslag.

in conclusion

Programmeren leert het kind op een andere manier nadenken over een probleem. Het bevorderd het opdelen van een groot probleem in kleinere deelproblemen, en vlotter naar een oplossing te zoeken. Het algoritmisch nadenken staat het kind ook toe bepaalde acties die herhaald moeten worden misschien efficienter uit te voeren.

---

storytelling

Om volledig te kunnen begrijpen wat een narratief kan betekenen voor een gebruiker, en hoe we dit kunnen gebruiken, moeten we natuurlijk eerst definnieren hoe een verhaal in elkaar zit.

oorsprong van verhalen

Verhalen zijn doorheen de tijd vaak veranderd. Ze zijn introverter geworden (mensen lezen verhalen zelf, zonder ze aan anderen te vertellen), zijn zijn van mondelinge vorm naar een geschreven vorm gegaan. Doorheen de jaren zijn er ook meer en meer geworden.

We vertellen al verhalen sinds het begin van mensenheugenis, denk maar aan de rotstekeningen in frankrijk. In die tijd werden verhalen vooral verteld, om geschiedenis door te geven, of fenomenen te verklaren. Met de komst van een geschreven taal zijn we deze verhalen ook beginnen neerschrijven. Het is echter pas sinds de uitvinding van de

boekdrukkunst dat we deze in geschreven vorm ook beginnen verspreiden zijn. De tijd tussen deze eerste vooruitgangen spreid zich echter over duizenden jaren. Pas met de komst van de film is het echt snel beginnen gaan. Na film kwam radio en tv, waarna computers en het internet het rijtje voegden om verhalen door te geven.

Elke van deze uitvindingen brachten met zich mee dat artiesten een nieuwe manier hadden om verhalen te vertellen, maar ook hun verhalen aan te passen naar het gebruikte medium. Denk maar aan de uitvinding van de stop-motion truck in de filmwereld, waarmee het plots mogelijk werdt verhalen anders vorm te geven, en andere verhalen te vertellen. Acties moesten niet meer mogelijk zijn, men kon truckeren.

In 2005 kwam youtube uit. Dit is een van de grotere milestones in het vertellen van verhalen. Het werd mogelijk voor iedereen met een camera (elk soort camera) eigen verhalen en films te verspreiden. Dit in combinatie met het goedkoper worden van de filmcamera en smartphones zorgt voor een enorme boost in het aantal en de kwaliteit van verspreidde films, en dus ook verhalen.

Nieuwe technologie gaat ook vaak gepaard met een nieuwe vorm van verhaalvertelling. Denk maar aan Vines, een van de recentere vormen hiervan. een online platform waarin de gebruiker een luttele 7 seconden kreeg om zijn punt, of verhaal, in duidelijk te maken.

nut van storytelling

Het vertellen van verhalen heeft geen echte regels. Eerder wegwijzers. Er zijn een aantal guidelines waar aan gedacht kan worden, maar die niet nodig zijn om het schrijven van verhaal tot een goed einde te brengen. Vaak zijn de verhalen die afwijken van deze regels eerder succesvol, omdat de lezer verast wordt.

fundamentals of digital storytelling

Als we spreken over digital storytelling, bedoelen we vooral een verhaal dat ondersteund wordt door een xtra laag, die vaak computergestuurd is. Die laag kan aanvullend zijn, zoals het gebruik van augmented reality in een boek. Hij kan echter ook de enige laag zijn die we zien. Dit is vaak het geval in games of films. De gebruiker ziet het verhaal slechts afgespeeld op zijn scherm, en heeft hierop niet veel invloed.

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).

een narratieve context voegt tevens betekenis toe, en legt verbindingen in je hersenen naar contexten die je herkent, waardoor je het beter opneemt en dus beter leert. Leren is verbindingen leggen in de hersenen, en een betekenisvolle context helpt daarbij.

De sleutel tot een goed verhaal (volgens andrew stanton)

het vertellen van een verhaal loopt gelijk aan het vertellen van een mop. Je weet op voorhand hoe het verhaal to stand zal komen, en hoe het zal aflopen. Alles moet leiden tot 1 uiteindelijk resultaat. Veelal is dit een doel van het hoofdpersonage. Soms is dit doel niet specifiek uitgesproken. Het kan dieper liggen dan waar het verhaal over gaat. Dit doel zorgt niet altijd voor de goede beslissing. De hoofdrol kan volledig de verkeerde richting uitgestuurd worden.

We kijken verhalen om te voelen, meevoelen met de protagonist. Volgens Andrew Stanton willen we ook vooral onszelf in de schoenen van de protagonist kunnen plaatsen. Verhalen laten ons dus toe gelijkheden tussen onszelf en anderen (in een andere tijd of ruimte) te kunnen voelen, of meemaken.

“there isn’t anyone you couldn’t learn to love, once you’ve heard their story”

Andrew spreekt in zijn talk ook grotendeels over “a promise worth your time”. Een belofte die aan het begin van het verhaal gemaakt worden en waar naartoe geleefd kan worden. Een belofte die de lezer beloofd dat zijn tijd waardig besteed zal worden. Een van de meest gebruikte zinnen om dit te doen is “er was eens”. Andrew bschrijft het belang van deze belofte als een steentje, in een katapult. het katapulteert de lezer/luisteraar door het verhaal, tot aan het einde.

Lezers, luisteraars, of kijkers willen best werken voor hun verhaal. ze willen het alleen niet expliciet weten. Het voelt meer aan als een eigen werk als je zelf de gaten kan invullen, onbewust. Als voorbeeld kan Wall-E aangehaald worden. een film over een afvalrobot die achterblijft op een vervuilde planeet(pixar). Wall-e kan niet praten, en dus moet het publiek zelf werken voor hun eten (uitspraak).

quotes

“drama is anticipation mingled with uncertainty” “technology is as ambiguous as the air we breath” godfrey Reggio “work with technology that exists, not with the promises of tomorrow” tawny schlieski

“we zijn op een punt in digital storytelling waar we niet goed weten wat te vertellen, maar

enthousiast zijn over de mogelijkheden.” dit was een situatie die voorkwam in het begin van film, waar mensen filmpjes animeerden waarin geen verhaal verteld wordt, maar vooral de effecten getoond werden.

work with tools of today, not with the promises of tomorrow

Resultaat

algemene structuur

Het programma werkt via een aantal fasen. Deze laten toe dat er toevoegingen kunnen gebeuren zonder te moeten zoeken in een ingewikkelde structuur van als-dan loops. Om de code duidelijk en gestructureerd te houden wordt er natuurlijk ook gewerkt met klassen. Niet alle klassen zullen in dit document uitgebreid besproken worden.

De interfaceklasse

Deze klasse neemt de visuele output voor zijn rekening. Het zorgt ervoor dat de gebruiker weet wara het boek gelegd moet worden om gescanned te worden, of welke codeblokjes met elkaar verbonden zijn. Om dit overzichtelijk te houden heeft elke status een eigen functie.

Deze klasse is pas vrij laat toegevoegd, omdat de nood aan een uitgebreide visuele output mij pas duidelijk werd na een gesprek met Beyond.io. Deze output bestond al reeds maar werdt vooral door de hoofdklasse afgehandeld.

De verhaalklasse

Deze klasse wordt pas aangeroepen als een nieuw boek gescanned wordt. Indien dit zich voordoet haalt deze klasse de nodige bestandenuit een online file-structuur. Deze zal eveneens een foutmelding weergeven als het gescannede boek niet herkend wordt. Eens deze bestanden opgeslagen zijn zal deze klasse dienen als een centrale verzamelplaats waar andere klassen informatie uit kunnen halen, zoals bijvoorbeeld de compilatieklasse en de visuele weergave van het desbetreffende level.

De compilatieklasse

Deze klasse heeft een aantal zeer verschillende vormen ondergaan. Dit omdat de eigenlijke werking vrij ingewikkeld in elkaar zit. De klasse krijgt aanvankelijk een serie getallen door, die deze omzet in acties. Dit was natuurlijk een gemakkelijke functie, tot het gebruik van for-loops (herhaling van een aantal acties) in het blikveld terecht kwam. Hieronder wordt het gehele proces opgedeeld in een aantal stappen.

Stap 1

De eerste stap gebeurd reeds voor de gebruiker kaarten heeft neergelegd. In deze stap zal de compiler het speelveld opdelen in een aantal blokken. Sommige blokken zorgen ervoor

dat de speler niet verder kan, anderen zijn zogenaamde valkuilen, waar de speler opnieuw moet beginnen indien hij deze raakt.

Stap 2

De speler heeft zijn kaarten gelegd, en heeft een groene kaart in de desbetreffende zone gelegd om duidelijk te maken dat hij zijn sequentie wil testen. De compiler zal hierop een eerste keer over de serie getallen lopen, om op zoek te gaan naar een for-loop. Indien deze tegengekomen wordt, zal hij de acties binnen deze loop herhalen. Het aantal herhalingen hangt af van wat de gebruiker ingegeven heeft.

Stap 3

Nadat alle speciale acties zijn omgezet in een lange serie afzonderlijke acties, zal deze klasse uitzoeken of de uitvoer mogelijk is. Omdat deze reeds weet waar de valkuilen en solide blokken zitten, kan deze stap voor stap nagaan of de speler een code heeft gelegd die lukt. Hij doet dit door actie voor actie uit te zoeken waar de speler staat, en of deze positie een gevolg heeft. Elke stap zal opgeslagen worden in een nieuwe Array (opeenvolging van acties).

Stap 4

Hierna (onafhankelijk van het resultaat van voorafgaande functie) zal de laatstgenoemde array uitgevoerd worden. Dit gebeurd traag, en stap voor stap, zodat de gebruiker kan volgen via de kaarten die hij reeds heft neergelegd. Op deze manier kan de gebruiker ook terugvinden waar hij een fout heeft gemaakt. Terwijl deze acties worden uitgevoerd, wordt de kaart die overeenkomt met de actie aangeduid via de projectie.

De tag recognition

Tags zijn snel herkenbare afbeeldingen die eruit zien als een vereenvoudiging van een QR code. Ze worden algemeen gebruikt om informatie door te geven aan het programma. Onder andere het overgaan naar een codemoment, maar eveneens de positie van blokken code worden via deze tags geregeld.

Deze tags hebben een cruciale rol in het geheel. 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.

Versie een

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.

Versie twee

Het probleem met deze eerste versie is dat deze het gehel oppervlak moet afzoeken naar een tag. Dit kon efficiënter. Ik besloot het probleem op te lossen door gebruik te maken van bepaade scanvelden. Dit zijn kleine velden die geprojecteerd worden waarin een kaart gescanned kan worden. Het programma moest in plaats van 1400*900 dus nog maar 200*200

Finale versie

Omdat 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 wordt apart behandeld. Er werd 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 wordt slechts een scan uitgevoerd als de camera detecteerde dat de kleur van het veld drastisch veranderd was.

Vooruitang

Beyond.io raadde me aan te beginnen met het schrijven van het verhaal. Op deze manier zou ik voorkomen dat stukken code nooit gebruikt werden in de finale versie, omdat het verhaal dit bijvoorbeeld niet toestond. Ook zou dit de kans dat ik structurele aanpassingen zou moeten doen drastisch verlagen. Na het aanbrengen van een degelijk verhaal zou ik dan kunnen overschakelen tot het schrijven van een algemene versie van het programma, een die nog geen cameraondersteuning had, en gewoon de basis weergaf.

Zoektocht naar een verhaal

De zoketocht naar schrijvers

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.

Het 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 kijkhoekproblem 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.

Deze oplossing brengt een aantal voordelen met zich mee

1. Het laat creatieve leerkrachten toe verder te werken met bijvoorbeeld een klasmascotte. Via deze weg zijn kinderen al vanin het begin gemotiveerd en is het gevoel van empathie al vanaf pagina een aanwezig.
2. Het staat toe dat schrijvers eigen boeken aanpassen of uitbreiden met programmeeropdrachten in het achterhoofd. Men bevestigd de desbetreffende tags aan het boek, op de juiste plaats, en het kind kan ermee beginnen werken.
3. Het staat toe dat gevorderde kinderen uiteindelijk zelf verhalen kunnen toevoegen, en elkaar kunnen uitdagen.

Vorm van het project

Allen tesamen

conclusions and further work