Card Game Toolkit

Het Verbeteren van de Workflow in de Initiële Ontwerpfase van een Kaartspel

Auteur: Midas Buitink
Afstudeerbegeleider: Karel Millenaar
Bedrijfsbegeleider: Peter de Jong

codeglue

Card Game Toolkit

Het Verbeteren van de Workflow in de Initiële Ontwerpfase van een Kaartspel

Midas Buitink

Studentennummer, 500660108
Telefoonnummer, 06 42 11 45 08

Hogeschool van Amsterdam
HBO-ICT Game Technology

Afstudeerbegeleider, Karel Millenaar

Codeglue

Goudsesingel 178, 3011 KD, Rotterdam
Telefoonnummer, +31 10 476 4522

Bedrijfsbegeleider, Peter de Jong

Nederland, Rotterdam, 01-07-2020
Stageperiode, derde en vierde semester 2019-2020

Inhoudsopgave

Samenvatting

Voor veel ontwerpers en programmeurs zijn er software tools beschikbaar om hun werk efficiënter en beter te kunnen doen. Er zijn game engines, geluid en image editing software, programma's om 3D modellen te maken en andere, etc. Maar er ontbreken nog tools voor een van de disciplines.

De game designer heeft weinig ondersteuning op het gebied van software tools. Ontwikkelaars van veel gebruikte game engines lijken steeds meer features voor game designers te implementeren. Maar hoe zit het met tools voor ontwerpers van fysieke spelen zoals bord en kaartspellen.

We presenteren de Card Game Toolkit. Een software oplossing om het ontwerpproces van kaartspellen te ondersteunen. We kijken naar twee iteraties van de tool, die aansluiten bij verschillende onderdelen van het ontwerpproces.

We evalueren de twee versies van de CGT aan de hand van de case van Codeglue. Game studio Codeglue maakt voornamelijk digitale games maar is geïnteresseerd in het ontwikkelen van een fysiek kaartspel. De huidige workflow is nog niet rendabel genoeg. Ze zouden het ontwerpproces willen versnellen en meer zichtbaar willen maken.

De CGT 1.0 is ontworpen om de spelregels te kunnen formaliseren zodat de ontwerper na het beschrijven van de regels niet hoeft na te denken welke regels op elk moment van toepassing zijn. Dit geeft de ontwerper meer cognitieve ruimte om zich te richten op de ontwerpproblemen. Helaas blijkt de CGT 1.0 de ontwerper te beperken in het experimenteren met verschillende structuren van het spel. De formele structuur kan maar een klein stukje van de ontwerp ruimte beschrijven. De CGT 1.0 is beperkt tot traditionele kaartspellen zoals pesten.

De CGT 2.0 richt zich op het oplossen van een van de problemen van meneer Kamp, de designer van Codeglue. Meneer Kamp is veel tijd kwijt aan het maken en aanpassen van zijn ontwerp. Daarnaast moet hij zijn ontwerp zowel in zijn design document in Excel en in Photoshop aanpassen. Dit maakt het onduidelijk wat de laatste versie is.

De CGT 2.0 biedt een omgeving om snel kaarten te maken en voornamelijk om deze kaarten snel aan te passen. Een belangrijk onderdeel die dit mogelijk maakt zijn de *card templates.* Kaarten die gebruik maken van een template worden direct aangepast als het template verandert maar ze behouden hun eigen waardes.

De CGT 2.0 zorgt er voor dat de tijd die nodig is om een bepaalde hoeveelheid kaarten aan te passen gelijk is een de hoeveelheid tijd die nodig is om één kaart te veranderen.

1 Inleiding

Het is bijna onmogelijk om een spel te ontwerpen zonder het te spelen [6].

Spelontwerpers maken daarom vaak gebruik van een iteratief ontwerpproces. Deze methoden legt de nadruk op playtesting en prototyping, waarbij ontwerpbeslissingen gebaseerd zijn op de ervaring van het spelen van een spel tijdens het ontwikkelproces.

Het doel van een prototype is het evalueren van een ontwerp als werkend systeem, in plaats van een theoretisch model. Het prototype wordt gespeeld, geëvalueerd, aangepast en opnieuw gespeeld. Dit cyclische ontwerpproces helpt de ontwerper succesvolle dan wel spelbrekende elementen te identificeren, waardoor hij beter onderbouwde ontwerp beslissingen kan nemen.

Het is daarom belangrijk om zo snel mogelijk een ruwe versie van het spel te maken. Het boek *Rules of Play* (Salen, K. & Zimmerman, E.) [12] noemt de volgende vuistregel; het eerste prototype moet gemaakt en getest zijn voordat 20 procent van de projectduur verstreken is.

Er zijn weinig software tools beschikbaar voor het ontwerpen van spellen. Dan hebben we het over tools gericht op het oplossen van ontwerpproblemen, het documenteren en vastleggen van een spel of andere onderdelen van het ontwerpproces. Tegenwoordig zijn game engines voor digitale games redelijk toegankelijk zijn spelontwerpers zonder technische achtergrond, we vroegen ons af waarom er geen software tools zijn voor ontwerpers van fysieke spellen zoals kaart en bordspellen.

We zijn in contact gekomen met Codeglue vanuit een exploratief onderzoek naar een software tool voor het ontwikkelen van kaartspellen. Dit afstudeerproject is een directe opvolging van de resultaten van het vorige project. De richting van deze voortzetting hangt af van de problematiek van game studio Codeglue.

1.1 Codeglue

Codeglue is een game development studio gevestigd in Rotterdam. Het bedrijf telt elf werknemers: waaronder zes programmeurs, één spelontwerper, en één artiest. Naast het ontwikkelen van hun eigen spellen leveren ze een dienst genaamd porting aan grote internationale uitgevers (o.a. Microsoft, EA, THQ). Porting is het omzetten van een spel van ene platform naar een ander platform. Dit is voornamelijk een technische uitdaging waarbij de werklast grotendeels bij de programmeurs ligt. Hierdoor heeft de spelontwerper meer tijd beschikbaar voor andere activiteiten.

Codeglue is geïnteresseerd in het ontwikkelen van een fysiek kaartspel, geïnspireerd op hun digitale card game spelbenders. Het prototypen en playtesten neemt te veel tijd in beslag waardoor het op dit moment nog niet goed te realiseren is.

1.2 Probleemstelling

Codeglue loopt tegen een aantal problemen aan omtrent hun ontwerpproces van hun kaartspel. Ze omschrijven de volgende problemen:

- Er is een gebrek aan overzicht;
- Er is te weinig documentatie;
- Iteraties zijn niet zichtbaar;
- Het produceren van een prototype kost veel tijd;
- De terugkoppeling van een fysieke versie naar een digitale versie neemt veel tijd in beslag.

Ze zijn geïnteresseerd in het ontwikkelen van een software tool om hun kaartspel digitaal te kunnen prototypen. Dit zou niet alleen het ontwerpproces moeten versnellen maar het ook meer zichtbaar moeten maken.

1.3 Doelstelling

Het doel is het ontwikkelen van een software tool voor game studio Codeglue om het ontwerpproces van hun kaartspel te verbeteren, door het huidige ontwerpproces te vertalen naar een digitale omgeving, om hiermee tijdrovende en mentaal intensieve taken te automatiseren, waardoor de ontwerper meer aandacht kan besteden aan het kritisch evalueren van zijn ontwerp.

1.4 Onderzoeksvraag

Hoe kan een software tool de verschillende fases van het ontwerpproces ondersteunen waardoor de ontwerper meer cognitieve energie beschikbaar heeft voor het oplossen van ontwerpproblemen?

2 Methoden

2.1 Literatuuronderzoek

Als voorbereiding op een interview met de game designer van Codeglue hebben we literatuur verzameld over het ontwerpproces van (kaart)spellen. We hebben gekeken welke methoden bestaan om een spelontwerp tot stand te brengen, en welke specifieke technieken het ontwerpen van kaartspellen kan ondersteunen. Ook hebben we vastgesteld wat we verstaan onder kaartspellen.

Het literatuuronderzoek vormt een context waarmee we het ontwerpproces van Codeglue kunnen vergelijken. Daarnaast helpt het om doelgerichte vragen te kunnen stellen die dieper op het proces in gaan.

2.2 Semi-Structured Interview

Een eenvoudige manier om snel persoonlijke informatie te vergaren is door middel van een interview. Het is belangrijk om eerst een duidelijk beeld te bewerkstelligen van de huidige problemen van de game designer. Het stellen van vragen omtrent het ontwerpproces en bijbehorende taken waarbij de interviewer de mogelijkheid heeft om dieper in te gaan op nieuwe informatie zou moeten leiden tot een heldere beschrijving van de huidige situatie.

Uit een interview met meneer Kamp Kamp, game designer bij Codeglue, moet een overzicht komen van zijn ontwerpstappen en waarom deze onderdeel uitmaken van zijn proces. Hoeveel tijd besteed hij aan deze taken? Waar denkt hij dat de problemen liggen en wat heeft hij in gedachten als een mogelijke oplossing?

2.3 Dag-in-het-leven

Om de informatie van het interview te verifiëren zal de game designer zijn volledige ontwerpproces toepassen onder observatie. Het schaduwen van het werkproces moet het beeld op de problematiek scherpstellen en eventueel ongenoemde problemen zichtbaar maken. Het dient ook om aanvullende concrete data te meten zoals bijvoorbeeld de exacte tijdsduur van verschillende taken.

De ontwerper krijgt een opdracht om binnen drie halve dagen een gehele doorloop van zijn ontwerpproces te laten zien. Door de lockdown van covid-19 is het niet mogelijk om fysiek aanwezig te zijn bij het ontwerpproces. Er is besloten om fysieke taken te verplaatsen naar een digitale omgeving, bijvoorbeeld het maken van schetsen op papier zal de ontwerper nu via een tekenprogramma doen. De observant zal de tijdsduur van individuele taken noteren, vragen om handelingen toe te lichten en het gehele proces opnemen via screensharing met audio opnames.

2.4 Usability Test

Het resultaat van het vorige project is de Card Game Toolkit 1.0 (CGT): een software tool waarin het mogelijk is om een formele beschrijving te geven van de spelregels van een kaartspel. De CGT 1.0 is ontworpen als een digitale omgeving om traditionele kaartspellen te kunnen beschrijven waarna de gebruiker het spel kan spelen.

We zijn nieuwsgierig in hoeverre de huidige tool bruikbaar is om een of meerdere problemen van Codeglue op te lossen. Hiervoor gebruiken we een usability test, deze moet ook inzicht geven in de gebruiksvriendelijkheid van de tool.

Een usability test is een kwaliteitsevaluatie van de gebruiksvriendelijkheid van een product. De hieronder beschreven criteria vormen de verschillende perspectieven van waaruit we kritisch naar het ontwerp gaan kijken [16].

Ease of Learning
Hoe gemakkelijk is het om de software tool te leren gebruiken?

Efficiency
Nadat gebruikers de functionaliteit van de software tool geleerd hebben, hoe snel kunnen ze taken volbrengen?

Error Tolerance
Hoeveel fouten maken gebruikers? Hoe zwaar is de impact van deze fouten? Hoe gemakkelijk kunnen ze een fout herstellen?

Engagement
Hoe fijn is het om de software tool te gebruiken?

Memorability
Hoe gemakkelijk kunnen gebruikers een proces hervatten als ze na een lange tijd terugkeren naar het programma? Kunnen ze nog even behendig met het programma omgaan?

Daarnaast moet de usability test antwoord geven op de vraag; doet het product wat het moet doen?

3 Het Ontwerppproces

3.1 Wat is Game Design?

Voordat we een nieuwe workflow kunnen ontwerpen is het essentieel om een duidelijk beeld te krijgen van *game design* zowel als een concept als een beoefening.

Een *game designer* is een specifieke vorm van een ontwerper, net als een grafisch ontwerper of industrieel ontwerper. Een game designer ontwerpt *gameplay* door het bedenken en voorleggen van regels en structuren die een ervaring vormen voor spelers [12].

Er bestaat geen recept voor een succesvol spel. Het ontwerpen van een spel is vergelijkbaar met het componeren van een muziekstuk. Het is een creatief proces waarbij elke ontwerper zijn eigen manieren heeft om tot een goed resultaat te komen. Maar er zijn wel technieken die het ontwerpproces kunnen ondersteunen.

3.2 Iteratief Design

Het is bijna onmogelijk om een spel te ontwerpen zonder het te spelen. Niemand kan precies voorspellen welke ervaring een speler zal hebben. Om een een spel te ontwerpen moet je het kunnen ervaren. *Iteratief design* maakt dit mogelijk.

Het boek Rules of Play (Salen, Zimmerman, 2004) [12] definiëert *iteratief design* als een *play-based* design proces. Het is een cyclisch ontwerpproces waarbij de nadruk ligt op het constant testen van een concept. Een game designer maakt een prototype dat hij kan playtesten, evalueren, aanpassen, en nog een keer spelen. Ontwerpbeslissingen zijn hiermee gebaseerd op succesvolle iteraties van het spel.

3.2.1 Doelstellen

De eerste stap in het ontwikkelen van een spel is het vaststellen van een of meerdere ontwerpdoelen. De doelen dienen als een maatstaf voor het evalueren van succesvolle ontwerpbeslissingen. Alle ontwerpbeslissingen moeten het nastreven van de ontwerpdoelen ondersteunen. Hiermee vormen ze ook een afbakening voor de ontwerpruimte waarin de ontwerper zich kan bewegen. Deze restrictie is noodzakelijk om ideeën te kunnen genereren.

3.2.2 Brainstormen

Na het vaststellen van de ontwerpdoelen probeert de spelontwerper zo veel mogelijk ideeën naar boven te halen en te omschrijven. Het vastleggen hierin is essentieel, concepten in het hoofd van de ontwerper zijn kwetsbaar. Ze kunnen makkelijk veranderen of vergeten worden. Daarnaast geeft het noteren de ruimte voor nieuwe gedachten.

De ontwerper gaat op onderzoek uit. Hij wil zo veel mogelijk relevante informatie verzamelen en om betere ideeën te creëren. Hij kan gebruik maken van spel componenten zoals kaarten, dobbelstenen of tokens om mee te experimenteren.

Hierbij kan het helpen om tussendoor afstand te nemen van het project door bijvoorbeeld ergens anders aan te werken. Als hij terugkeert naar zijn ontwerp kan hij nieuwe associaties leggen. Daarom is het documenteren zo belangrijk voor het ontwerpproces.

De ontwerper evalueert zijn ideeën waarbij hij de ontwerpdoelen in acht neemt. Hij maakt een selectie van concepten die hij gaat testen.

3.2.3 Prototyping

Een ontwerper kan een intuïtie ontwikkelen voor de effecten van specifieke spelelementen maar hij is niet instaat om precies te voorspellen welke ervaring de spelers zullen hebben. De interacties van alle onderdelen van een spel zijn alleen zichtbaar te maken door het in de context te beleven. De ontwerper moet zijn ontwerp testen door middel van een prototype.

Hij maakt het prototype met een doel. Deze liggen vaak in lijn met de eerder gedefinieerde ontwerpdoelen. Het prototype moet de intuïtie van de game designer kunnen bevestigen. Het prototype hoeft niet altijd een directe weerspiegeling te zijn van het eindproduct, het moet dienen als een schijnwerper op de belangrijkste ontwerpproblemen. Alle aspecten die niet direct helpen om het doel van het prototype te bereiken laat hij buiten beschouwing. Als hij de economie van zijn spel wilt onderzoeken is bijvoorbeeld de vormgeving van de kaarten niet belangrijk. De ontwerper probeert zo snel mogelijk een speelbaar ontwerp te ontwikkelen.

3.2.4 Playtesten

Het testen van een prototype of een versie van het uiteindelijke product is een van de belangrijkste fases van het ontwerpproces. Het playtesten benaderd de beleving van het uiteindelijke resultaat.

De ontwerper test eerst individueel zijn ontwerp waarbij hij de rol aanneemt van verschillende spelers. Hij probeert hiermee onduidelijkheden of contradicties in de spelregels te elimineren voordat hij anderen het spel laat spelen.

Tijdens het testen met spelers observeert de game designer de handelingen en de bijbehorende ervaringen van de deelnemers. Als het de spelervaring niet beïnvloed kan hij de spelers vragen hardop na te denken tijdens het maken van spel beslissingen. Hiermee kan hij inzicht krijgen in de beleving van de spelers. Na het noteren en verzamelen van zijn bevindingen is het tijd om de informatie te evalueren.

3.2.5 Evalueren

De bevindingen van het playtesten moeten de kwaliteiten en gebreken van het spel belichten. Waarin het spel een oplossing moet zijn voor de ontwerpproblemen. De ontwerper ondervraagd zichzelf, is het doel van het prototype bereikt? Zijn hiermee alle ontwerpdoelen bereikt? Moet het ontwerp of de ontwerpdoelen worden bijgeschaafd om tot een succesvoller resultaat te komen?

Als uit de evaluatie blijkt dat de ontwerpdoelen zijn bereikt of als geen duidelijke aanwijzingen zijn voor verbeteringen is het spel geslaagd. Is dit nog niet geval dan moet de ontwerper terug naar de tekentafel om zijn ontwerp te herzien.

3.2.6 Aanpassen

De ontwerper maakt aanpassingen in zijn ontwerp. Als het onduidelijk is welke aanpassingen het spel kunnen verbeteren kan hij verschillende iteraties prototypen, play testen en evalueren. Als het mogelijk is om een prototype direct aan te passen kan dit tijd besparen.

De ontwerper herhaald deze stappen net zo lang totdat de ontwerpdoelen bereikt zijn.

4 Kaartspellen

Het is lastig om een concrete definitie te geven van een *kaartspel*. Het is geen genre maar meer een medium om gameplay mogelijk te maken, in die zin gelijk aan *computerspellen*, *bordspellen* of *conversatie spellen* (e.g. *20 vragen, twee waarheden en één leugen, etc.)*
Een *kaartspel* is een spel waarbij kaarten het primaire hulpmiddel zijn om gameplay te faciliteren.

4.1 Kenmerken

Het gebruik van kaarten als spelelement kan verschillende redenen hebben.
Welke eigenschappen van kaarten zijn relevant voor het maken van ontwerpbeslissingen? Waarom gebruiken spelontwerpers kaarten? En waar moeten ze rekening mee houden?
De volgende kenmerken zijn zichtbaar in meerdere kaartspellen.

4.1.2 (On)zichtbaarheid van Informatie

Kaarten zijn (over het algemeen) alleen van één kant te onderscheiden. Hiermee kunnen we een belangrijk kenmerk identificeren, een kaartspel is meestal een spel met imperfecte informatie.

Tegenstanders hebben vaak zicht op een andere deelverzameling van kaarten. Waarbij strategisch spel kan ontstaan door het correct observeren van de distributie van kaarten en/of handelingen van andere spelers; pakken ze een kaart? Gooien ze er een weg? Hoe investeren ze hun resources?

4.1.3 Willekeurigheid

Het gebruik van kaarten ondersteund onvoorspelbaarheid, door een stapel kaarten te *schudden,* ofwel door de kaarten in willekeurige volgorde te plaatsen.

4.1.4 Modulariteit

Kaarten zijn losse stukken informatie die uitgewisseld kunnen worden door spelers. Ze kunnen in verschillende configuraties op tafel geplaatst worden waarbij elke variatie een andere betekenis heeft. En deze verdeling van kaarten kan tijdens het spelen veranderen, hierdoor zijn kaarten heel dynamisch.

5 Probleemanalyse

Om een duidelijk beeld te krijgen van de problemen gaan we in gesprek met Meneer Kamp Kamp, de game designer van Codeglue. In een interview noemt meneer Kamp dat Codeglue prototypes maakt op het moment dat zij in het ontwerpproces van hun spellen onbekend terrein betreed en keuzes kunnen niet gebaseerd worden op eerdere kennis, of er zijn weinig of geen voorbeelden te vinden vanuit andere spellen.

Codeglue maakt meestal een digitaal prototype omdat ze uitsluitend digitale producten maken. Helaas is dit niet altijd mogelijk omdat er soms geen programmeurs beschikbaar zijn. Dan probeert de ontwerper het zo goed mogelijk te vertalen naar een fysieke vorm.

De betrokken werknemers van Codeglue maken eerst een overzicht van de ontwerpproblemen die moeten worden opgelost. meneer Kamp is verantwoordelijk voor het vinden van een oplossing. Afhankelijk van de complexiteit van het probleem neemt hij eerst de tijd om op onderzoek uit te gaan. Hij probeert zoveel mogelijk informatie te bemachtigen om een mogelijke oplossing voor het probleem te kunnen vinden.

Hij noemt drie onderdelen van het prototyping proces. De eerste fase is het bedenken en formuleren van een concept. Hierna worden de spelonderdelen gemaakt. Als laatste wordt het ontwerp getest, eerst individueel daarna met spelers. Het proces duurt meestal één week.

meneer Kamp maakt o.a. een prototype om de onderdelen in beweging te zien, nu duurt het twee dagen voordat dit kan. Hij zou vaker prototypes maken als hij deze staat van het proces sneller kan bereiken.

Aanpassingen van een fysiek prototype zijn vaak slordig omdat het veel tijd kost alle kaarten netjes aan te passen. Maar het kost nog meer moeite om de digitale versie aan te passen en deze opnieuw te printen.

5.1 Het Huidige Ontwerpproces

In een interview beschrijft meneer Kamp de stappen van zijn ontwerpproces. Hij is nog niet heel ervaren in het ontwerpen van kaartspellen dus om een beter beeld te krijgen geven we meneer Kamp de opdracht om in twee dagen een prototype te maken van een kaartspel gebaseerd op Spelbenders. Het ontwerpproces wordt vastgelegd met behulp van screen / audio recording software waarbij meneer Kamp hardop zijn handelingen beschrijft. Op het moment dat hij vast lijkt te lopen krijgt hij een aantal vragen om vast te stellen dat hij inderdaad tegen een probleem aan loopt en hoe hij dit probeert op te lossen. Er wordt ook vastgesteld of dit een terugkerend probleem is in zijn ontwerpproces.

We volgen hierbij zijn proces tot aan het playtesten gezien de covid-19 situatie is het lastig om de playtesten met collega's zoals hij beschrijft in het interview.

5.1.2 Doelstellen

meneer Kamp omschrijft kort welke ervaring het spel tot stand moet brengen. Spelbenders heeft snelle rondes waarin spelers verschillende spreuken moeten combineren om hun tegenstander te verslaan. Het kaartspel moet snel te spelen zijn, er moeten interessante keuzes zijn maar de complexiteit mag het tempo niet belemmeren. Daarnaast neemt meneer Kamp het combineren van spreuken als uitgangspunt voor het ontwerp.

5.1.3 Brainstormen

meneer Kamp besteedt ongeveer drie en een half uur aan het vergaren van informatie. Hij is op zoek naar inspiratie door naar video's van andere kaartspellen te kijken. Hij noteert mechanics die passend zijn voor het kaartspel.

Daarna volgt een brainstorming sessie. meneer Kamp maakt notities in een tekstdocument. Hij probeert hierbij niet te lang bij één idee te blijven hangen maar voornamelijk veel verschillende concepten te noteren. Vervolgens evalueert hij de potentie van elk idee. Hij stelt zichzelf de vragen; is dit haalbaar? Kan hetzelfde bereikt worden op een makkelijkere manier? Kan ik dit snel testen?

5.1.4 Aanscherpen van Ideeën

Hij neemt de verschillende ideeën onder de loep, hij beschrijft elk concept in meer details.
Hij noteert mogelijk ontwerpproblemen voorkomen en gaat op zoek naar het idee met de meeste potentie. Hij kiest het spel ontwerp dat het beste bij zijn doelstelling past.

5.1.4 Prototypen

Het Documenteren van Zijn Ontwerp
Het concept met de meeste potentie gaat hij verder uitwerken. Hij omschrijft de doorloop van het spel in een design document. Het doel is om genoeg structuur op te zetten zodat hij het kan testen met behulp van een prototype. Hierbij besteedt hij veel tijd aan het opzetten van een kansberekening formule in excel om de verhouding van de verschillende soorten kaarten te bepalen. Het concept heeft drie type kaarten, “spreuken”, “elementen” en “artifacts”. Spelers hebben “elementen” die nodig zijn om andere kaarten te kunnen spelen. Elke speler moet een aantal van deze kaarten in zijn hand hebben om iets te kunnen doen. Na anderhalf uur heeft hij een formule opgezet om te berekenen hoeveel kaarten van elk type in het spel moeten zitten.

Bedenken van de kaarten
De kaarten die hij voor het prototype gaat maken beschrijft hij in een excel document. Om een “spreuk” te kunnen spelen moet een hoeveelheid “elementen” worden afgelegd. De maximale kost van een spreuk mag niet groter zijn dan een kaart minder dan maximale hoeveelheid kaarten in de hand van een speler. Binnen dit bereik kiest hij arbitraire waardes.

Het uitwerken van zijn ontwerp
Daarna werkt hij de kaarten uit in Photoshop. Hij maakt eerst een template van elk type kaart dat hij kan kopiëren en kan aanpassen met de juiste informatie. Daarna print en snijdt hij de onderdelen om het te kunnen testen.

5.1.5 Playtesten

Hij test het prototype eerst zelfstandig en daarna met collega’s. Het doel van het playtesten is om de onderdelen van het spel in beweging te zien. Hij kan hiermee ontdekken waar het spel vastloopt, of spelers genoeg invloed kunnen uitoefenen op de uitkomst van het spel en of ze het naar hun zin hebben.

5.1.6 Evalueren

Hij noteert bevindingen van het playtesten van elke iteratie. Hij let hierbij op de hoeveelheid keuzes die een speler heeft, of dit interessante keuzes zijn en of ze leiden naar de uitkomst die de speler verwacht. Als een strategie vaak herhaald wordt noteert hij dit ook. Daarna gaat hij opzoek naar een strategie die de dominerende strategie zou kunnen verslaan. Als dit niet het geval is moet het ontwerp worden aangepast.

5.1.7 Aanpassen

Na het testen maakt hij aanpassingen met pen. De aanpassingen zijn gebaseerd op de bevindingen van het testen. Als een specifieke strategie dominant is probeert hij kaarten weg te laten of andere kaarten toe te voegen die deze strategie onderuit kunnen halen. Hij maakt ook aanpassing waarvan hij de balans van het spel express heel erg verschuift. Hij doet dit om een nieuw perspectief te krijgen.

5.1.8 Conclusie

Het ontwerpproces van meneer Kamp komt overeen met de literatuur. Tijdens het brainstormen en aanscherpen van zijn ontwerp kan hij moeilijk zijn ontwerpbeslissingen beargumenteren.
De waardes van de kaarten of de hoeveelheid kaarten in een deck lijkt hij arbitrair te kiezen.

Hij besteed inderdaad veel tijd aan het uitwerken van de kaarten. Hij werkt gestructureerd waardoor het lastig is om later het ontwerp nog aan te passen. Het mag er uit zien als een prototype, want dat is wat het is, maar het moet wel onderhoudbaar zijn. Het lijkt er op dat hier veel winst te behalen valt. Het maken en bijhouden van de kaarten zou niet alleen een stuk sneller moeten kunnen maar ook veel overzichtelijker.

Laten we kijken of hij zich een weg kan vinden in de Card Game Toolkit. In de CGT 1.0 is het nog niet mogelijk om kaarten te maken. Maar we kunnen wel kijken of het opzetten van een formele structuur kan helpen bij het ontwerpen van een spel dat gebruik maakt van traditionele speelkaarten.

6 Game Design Tools

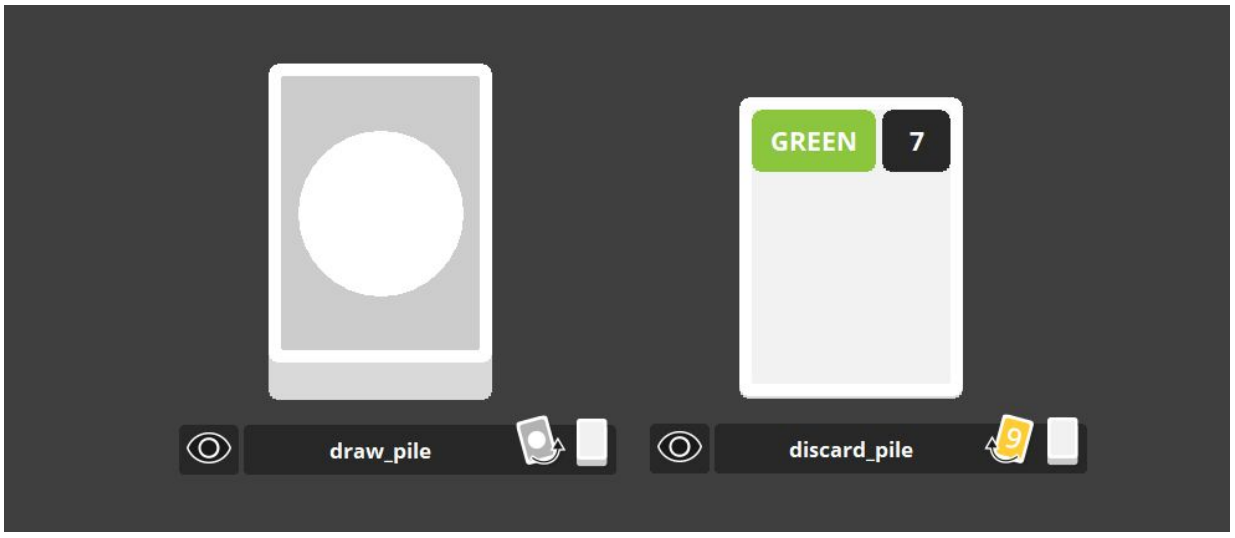
In een meta-onderzoek over de bijdragen en bevindingen van onderzoekers en spelontwerpers in een poging om geformaliseerde game design tools te ontwikkelen beschrijven (Almeida, Silva, 2013) [3][4] een gecomprimeerde lijst van requirements voor deze tools.
Het onderzoek richt zich voornamelijk op de vereisten voor video games, maar een groot deel van de vereisten is relevant voor game design als conceptontwikkeling, ongeacht de uiteindelijke vorm van het ontwerp.

- Het ontwerpproces moet veranderingen toestaan van het spelconcept.
- Design documentatie mag de werklading van de ontwerper niet belasten.
- Design documenten moeten makkelijk te maken zijn, en goed bij te houden zijn.
- De design documentatie moet het mogelijk maken om visuele modellen te maken van een ontwerp (Dormans, 2012).
- Prototype tools moeten een IDE (Integrated Development Environment) aanbieden zodat spelontwerpers hun concept gelijk visueel kunnen modelleren en playtesten.
- Het moet een formele visuele taal aanbieden om spellen te beschrijven als verschillende onderdelen en hun relaties [9].
- Deze taal moet verschillende aspecten van het ontwerp benaderen, op zowel hoge als lage abstractieniveaus [9].
- Deze taal moet gebaseerd zijn op een bewezen bestaande taal vanuit een ander veld, aangepast voor het ontwerpen van spellen.
- Het moet een manier hebben om een collectie van ontwerpkennis te creëren zodat bewezen concepten hergebruikt kunnen worden.
- Het moet begeleiding geven in het opbouwen en gebruiken van deze collectie.
- Het is belangrijk om zowel het perspectief van de ontwerper (producent) als de speler (consument) te overwegen tijdens het ontwerpproces, MDA (LeBlanc et al. 2004). De spelontwerper moet naar zijn ontwerp kunnen kijken vanuit het perspectief van een speler, maar ook vanuit het perspectief van een ontwerper [13].

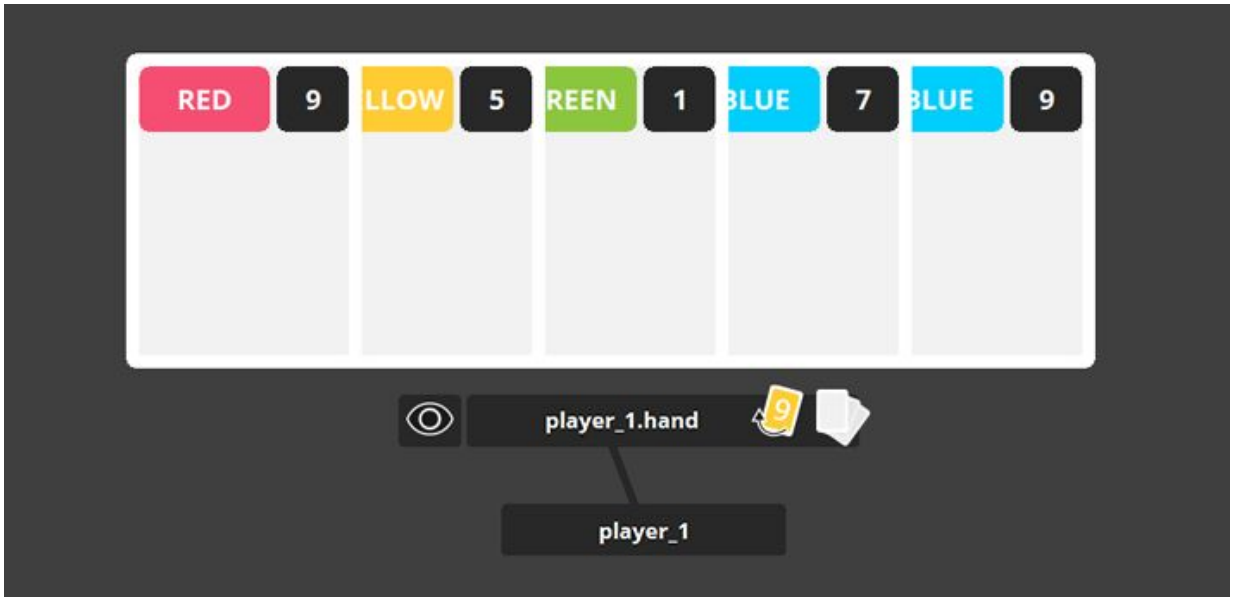
7 Card Game Toolkit 1.0

Het doel van de Card Game Toolkit 1.0 (CGT 1.0) is het formaliseren van kaartspellen door een omgeving te creëren waarin de gebruiker een interactief model kan maken van alle interacties van het spel. Een model bestaat uit *players*, *piles*, *cards* en *actions*.

Players zijn de spelers van het spel. Een *pile* is een betekenisvolle locatie waar een kaart kan bestaan, en een *card* is een collectie van attributen die iets betekenen binnen de context van het spel.

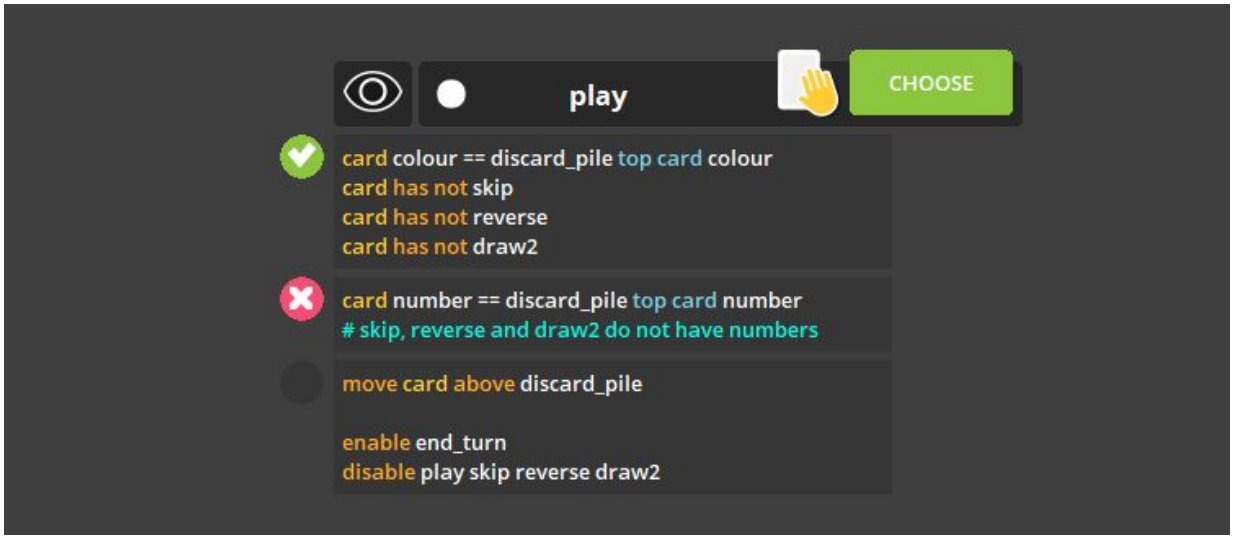


Figuur 1. Een dicht deck met kaarten en een open aflegstapel.



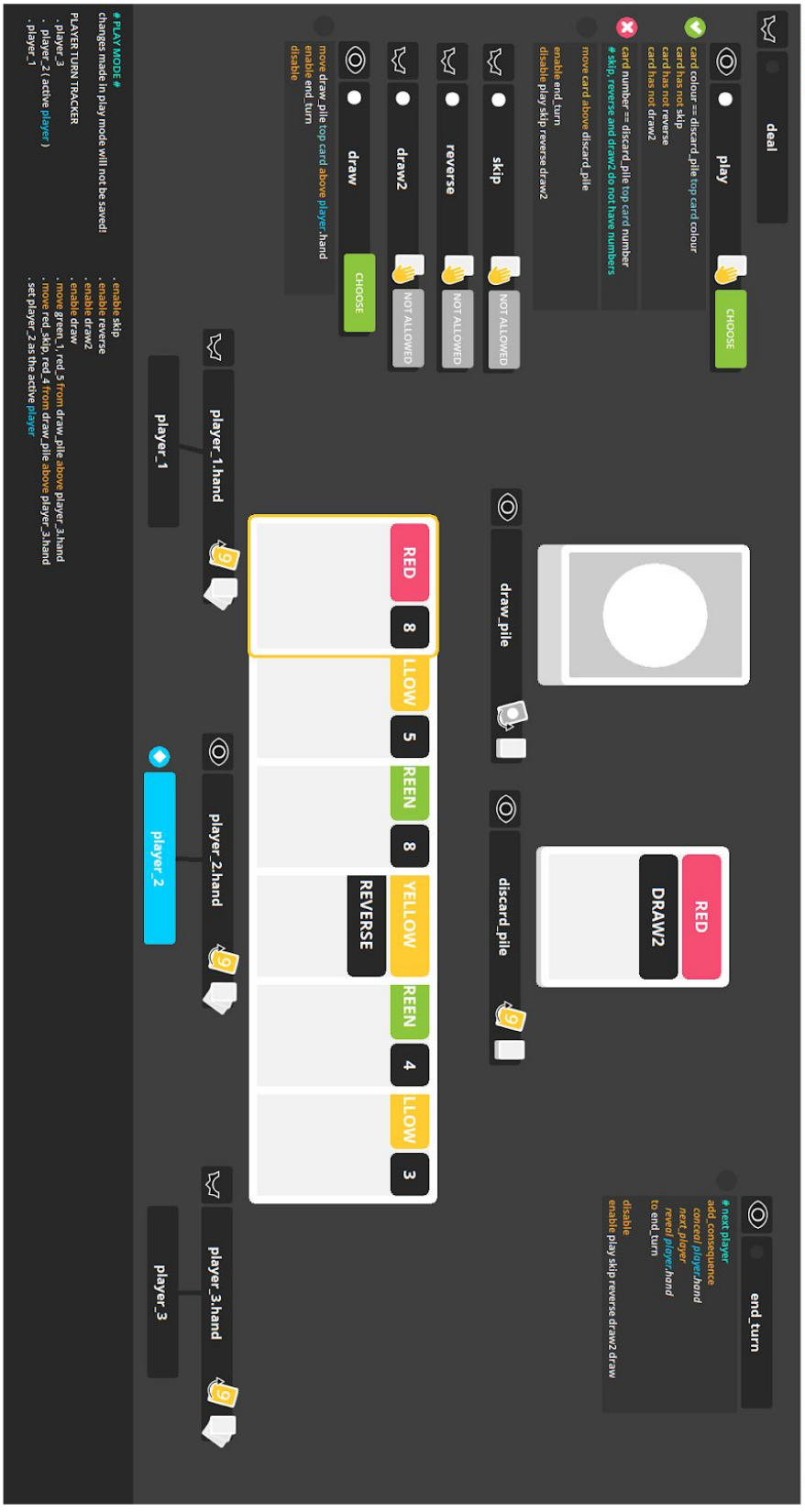
Figuur 2. Een uitgevouwen hand van speler 1.

Acties maken gebruik van CGT Script, een scripttaal om componenten te manipuleren. Een actie kan zijn: een stapel kaarten schudden en verdelen onder de spelers, een kaart verplaatsen vanuit de hand van een speler naar een aflegstapel, of een speler aanwijzen als de actieve speler. Acties kunnen geen componenten toevoegen of verwijderen, ze beïnvloeden alleen de configuratie van de aanwezigen componenten.



Figuur 3. Een actie waarbij de kleur of het nummer van een kaart overeen moet komen met de kaart op de aflegstapel om gespeeld te kunnen worden.

Het model is een abstracte versie van het spel waarbij alle componenten en interacties zichtbaar zijn. De CGT 1.0 heeft een *edit mode*, waarin de gebruiker het spel kan beschrijven en een *play mode* waarin de gebruiker het spel kan spelen door acties te selecteren.



Figuur 4. Een model van UNO. Speler 2 is aan de beurt en heeft een kaart geselecteerd voor de “play” actie.

7.1 Usability Test

Om gericht aan een nieuwe versie van de CGT te kunnen werken begint het onderzoek met een usability test.

7.1.1 Doel

Het doel is het bepalen van de gebruiksvriendelijkheid en de bruikbaarheid van de Card Game Toolkit. De gebruiker zou in staat moeten zijn om binnen twee uur de basisprincipes van de tool te begrijpen en daarmee een model te kunnen maken van het kaartspel UNO. Daarnaast moet de gebruiker herkennen wanneer hij fouten maakt en in staat zijn zichzelf te corrigeren met behulp van de feedback van het programma.

7.1.2 Opzet

De participanten krijgen een executable van de CGT en een document met de opdrachtomschrijving. Ze krijgen maximaal twee uur de tijd om de opdracht te volbrengen, waarvan waarschijnlijk 45 minuten besteed zullen worden aan het leren van de basisconcepten van de CGT waarna de participanten aan de slag gaan met het uitvoeren van de opdracht.

We maken gebruik van een think-aloud protocol waarbij zowel het scherm als de stem van de participanten wordt opgenomen. Participanten zullen tijdens het testen hun gedachten uitspreken zodat de observant inzicht kan krijgen in het denkproces van de gebruiker. Hierbij noteert de observant voornamelijk momenten waar de gebruiker vast loopt.

Het is belangrijk dat de observant tijdens het testen geen informatie geeft aan de participant over het gebruik van de tool. Hij kan de participant wel wijzen op de absentie van functionaliteit als de gebruiker veel tijd besteed aan een probleem buiten de scope van de opdracht.

Bij het testen van software in ontwikkeling is het niet ondenkbaar dat er onverwachte technische problemen optreden. In dit geval mag de observant de gebruiker erop attenderen dat het programma niet naar behoren functioneert om onnodige verwarring te voorkomen. Het gaat hierbij om bugs, niet om fouten van de gebruiker. Bij gebruikers fouten is het juist belangrijk dat de observant niet ingrijpt maar goed observeert hoe de gebruiker hiermee omgaat.

7.1.3 Opdracht

De participanten krijgen twee uur de tijd om een versie van het kaartspel UNO te maken met behulp van de CGT. De opdracht bestaat uit een overzicht van de regels van Uno, een beschrijving van de verschillende componenten ondersteund met afbeeldingen en voorbeeldprojecten en een documentatie van de CGT script, een scripttaal waarmee de gebruiker de spelregels kan definiëren.

De opdracht is ontwikkeld om de *ease of learning*, *efficiency*, *error tolerance* en *engagement* te meten.

Ease of learning
Om de opdracht te volbrengen moet de gebruiker kennis hebben van alle onderdelen van het programma. Op het moment dat participanten bepaalde onderdelen niet hebben kunnen maken kan dit het gevolg zijn van slechte *ease of learning* of slechte documentatie. Om het laatste uit te kunnen is er veel aandacht besteed aan het opstellen van overzichtelijke documentatie met duidelijke voorbeelden.

Efficiency
Om een model van Uno te realiseren zullen de participanten een aantal handelingen vaker moeten toepassen. Als participanten hier elke keer veel tijd aan besteden kan dit er op wijzen dat het uitvoeren van deze taken niet efficiënt is. Tenzij er een andere mogelijkheid is die onbekend is voor de gebruiker, in dat geval kan het een probleem zijn met de *ease of learning*.

Error tolerance
We maken onderscheid tussen twee soorten errors; workflow errors en technische errors.

Een workflow error ontstaat als de gebruiker geconfronteerd wordt met onverwachte resultaten van zijn acties. Een participant kan zijn fout gelijk herkennen en herstellen of hij/zij raakt verward en moet zijn proces onderbreken om het uit te zoeken of er later op terugkomen.

Een technische error ontstaat als een gebruiker componenten onjuist gebruikt of incorrecte syntax gebruikt in het omschrijven van spelregels. Na het ontstaan van deze error noteert de observant niet alleen wat er gebeurd maar ook of de participant inziet dat er iets niet goed gaat. Daarnaast noteert hij of de participant in staat is om met de feedback van het programma zijn/haar fout te herstellen of dat hij/zij het opgeeft.

Engagement
Aan het eind krijgen de participanten vijftien minuten de tijd om te reflecteren op hun proces waarbij de observant zo nodig een aantal opvallende momenten benoemt. Momenten waarbij de participant moeite had met de opdracht of juiste veel voortgang boekte.

Memorability
De participanten hebben niet eerder met het programma gewerkt daarom is deze kwaliteit niet meegenomen in de usability test.

7.1.4 Participanten

Tien participanten, waaronder de game designer van Codeglue, hebben meegewerkt aan de usability test. Er zijn voornamelijk participanten geselecteerd met affiniteit voor game design en/of game development.

Achtergrond	**Hoeveelheid participanten**
Ervaring met game design en programmeren	4
Alleen ervaring met game design	2
Alleen ervaring met programmeren	3
Geen ervaring met game design of programmeren	1

7.1.5 Resultaten

Ease of learning

Alle deelnemers hebben binnen het tijdsbestek een model kunnen maken van de basisregels van Uno, waarvan zeven participanten de opdracht volledig hebben kunnen voltooien, inclusief de implementatie van uitzonderingsregels.

Uit de reflectiegesprekken blijkt dat iedereen de functionaliteit van de componenten makkelijk te begrijpen vond. Maar op het moment dat de complexiteit van hun project groter werd vonden de meeste deelnemers het lastiger om goed het overzicht te bewaren.

Efficiency

Voornamelijk deelnemers met een achtergrond in programmeren hadden moeite met de efficiency van uitzonderingsregels. Ze hadden behoefte aan conditional branching door middel van if/else statements of het definiëren van variabelen. Hierdoor ontstond er bij een aantal lichte frustratie dat er geen efficiëntere oplossing mogelijk was.

Error tolerance

Error	**Instanties (combinatie van 9 participanten)**
Verwarring tussen edit mode / play mode	21
Verwarring tussen een *condition* en een *consequence (in het verkeerde script aanpassingen maken)*	14
Syntax error direct herkennen en herstellen	47
Syntax error niet direct herkennen *(verwarren met een conditie)*	33
Syntax error genegeerd	4
Per ongeluk componenten verwijderen	3

Er werd door bijna alle participanten regelmatig onbedoeld in play mode gewerkt in plaats van edit mode waardoor hun aanpassingen niet werden opgeslagen. Ze hadden dit vaak pas te laat door waardoor ze werk opnieuw moesten doen.

Syntax errors werden nauwelijks opgevangen. Participanten hadden het pas door als ze om een andere reden de log aan het bekijken waren.

Engagement
Alle participanten reageerde enthousiast op het moment dat ze een onderdeel succesvol hadden toegevoegd aan hun project. De participanten hebben de volledige tijdsduur geconcentreerd gewerkt. De reflectiegesprekken bevestigde dat ze het inderdaad leuk vonden om met het programma te werken.

De enige momenten van frustratie ontstonden bij het implementeren van de speciale kaarten. Desondanks heeft er maar één participant het opgegeven om bepaalde taken te volbrengen. Vier van de deelnemers hebben buiten het testen om nog met de CGT geëxperimenteerd.

7.1.6 Discussie

Door de covid-19 situatie was het alleen mogelijk om op afstand te testen met behulp van een videocall waarbij participanten hun scherm moesten delen. Helaas was de internetverbinding tijdens het testen met twee van de participanten niet adequaat om het proces goed te observeren. Deze participanten is gevraagd om de opdracht zelfstandig te maken om hierna een reflectiegesprek te kunnen voeren.

Het organiseren en overzicht bewaren naarmate het project groeide in complexiteit was voor veel deelnemers lastig. Ze probeerden uit te zoomen maar deze functionaliteit is er niet waardoor ze het 'canvas' regelmatig heen en weer moesten slepen. Daarnaast gebeuren alle instructies van een actie in een keer waardoor het af en toe lastig was voor gebruikers om goed te kunnen volgen wat er gebeurde. Het bijhouden van alle acties in een log helpt, maar een korte pauze tussen elke instructie en duidelijke visuele feedback zou dit probleem volledig moeten oplossen.

Tijdens het uitvoeren van acties kunnen andere acties geactiveerd of gedeactiveerd worden. Naarmate de complexiteit van het model toenam vonden participanten het lastig om acties goed te ordenen en het overzicht te bewaren. Er mist een overkoepelende structuur om groepen van acties als geheel te activeren of deactiveren.

Het is onduidelijk wanneer gebruikers in edit mode of in play mode zitten. Een aantal deelnemers maakt actief gebruik van de functionaliteit om in play mode hun ontwerp aan te passen en hiermee snel dingen te testen, dus het editen onmogelijk maken lijkt niet de juiste oplossing. Duidelijke feedback bij het component dat wordt aangepast tijdens play mode zou dit kunnen verhelpen.

De feedback voor een conditie die *false* is lijkt teveel op de feedback van een syntax error. Daarnaast was het niet altijd duidelijk wat er dan precies niet goed ging. Duidelijke feedback bij het script met daarbij contextuele aanwijzingen wat precies het probleem is in plaats van alleen aangeven dat de syntax onjuist is kan zorgen dat dit niet meer voorkomt.

Na afloop van de usability test heeft meneer Kamp op ons verzoek geprobeerd om zijn eerste prototype (van het dag-in-het-leven onderzoek, hoofdstuk 5) te realiseren binnen de CGT 1.0. Het was onze verwachting dat het onmogelijk zou zijn om de formele structuur toe te passen op het ontwerp. Dit bleek inderdaad het geval. Het is onmogelijk om meerdere kaarten tegelijkertijd te vergelijken, er kunnen geen waardes worden bijgehouden en de acties waren te complex en per kaart uniek. Dit zou geen probleem moeten zijn als hij de formele structuur zou kunnen negeren. Waardoor hij zelf controle zou kunnen hebben over de kaarten en waar ze naar toe bewegen.

7.1.6 Conclusie

Alle gebruikers vonden het makkelijk om alle basisregels van UNO te implementeren het model. Daarnaast vonden ze de syntax van de scripttaal makkelijk te leren en zeer leesbaar. De gebruikers met programmeerkennis vonden de taal soms te eenvoudig en hadden behoefte aan complexere functionaliteit, zoals conditional branching.

Daarnaast waren de gebruikers veel tijd kwijt aan het correct opzetten van de structuur van hun spel. Er was bij de meeste deelnemers het gevoel dat het efficiënter en overzichtelijker kon. Het definiëren van de condities en wanneer acties geactiveerd mochten worden werd hierdoor meer de focus dan het ontwerpproces, of het maken van creatieve keuzes. De absentie van het laatste heeft absoluut te maken met restricties van het nabouwen van een bestaand spel, desondanks was het zichtbaar dat creatieve expressie gehindert kan worden door de rigide structuur van een model. De ontwerper zou altijd buiten de gedefinieerde restricties van het spel moeten kunnen experimenteren[3][4]. Hij/zij moet regels kunnen definiëren die hij/zij mag overtreden zolang het systeem dit maar aangeeft.
De gebruikers waren erg tevreden over de visuele feedback op een paar onduidelijke elementen na. Ze vonden het wel lastig dat ze niet konden uitzoomen om een beter overzicht te krijgen.

De formele structuur zou een ervaren ontwerper eventueel kunnen helpen om complexe ontwerpen te kunnen maken. De tool kan de spelregels bijhouden waardoor het de ontwerper meer ruimte geeft om over het ontwerp na te denken.

Meneer Kamp is nog een beginnende ontwerper op het gebied van kaartspellen. Een omgeving waarin hij kan experimenten met de componenten van het spel is belangrijker. Om het makkelijker te maken snel een deck kaarten te creëren en deze vrij te kunnen manipuleren is een andere tool nodig.

Meneer Kamp moet zo snel mogelijk in deze experimentele fase kunnen komen, en als hij daar eenmaal is aangekomen moet hij kaarten snel en gemakkelijk kunnen aanpassen. Zo kan hij het grootste deel van zijn tijd besteden aan het onderzoeken van zijn verschillende aannames. De CGT 2.0 moet dit oplossen.

7.2 Ontwerp Beperkingen

De formele structuur van de CGT maakt de ontwerp ruimte erg beperkt. Dit zou geen probleem moeten zijn als het ontwerp van meneer Kamp veel overlap zou hebben met deze dimensie. Maar de basis concepten van zijn kaartspel zijn zeer lastig te realiseren in de huidige tool.

Kaarten moet unieke effecten kunnen hebben. Combinaties van kaarten moeten geëvalueerd kunnen worden. Verschillende variabelen zoals levenspunten van spelers en grondstofkosten van een kaart moeten bijgehouden kunnen worden.

Een formele structuur opzetten dat elk kaartspel kan beschrijven is een moeilijke opgave. Het is niet onmogelijk maar het is onduidelijk hoeveel onderzoek er nog nodig is om dit te kunnen volbrengen.

We weten dat we een enorme winst kunnen behalen in het productieproces van het prototype en de organisatie van het ontwerp. De volgende versie van de Card Game Toolkit zal zich hierop richten.

7.3 Technische Beperkingen

De CGT is gemaakt in Python met ondersteuning van de Tkinter library. Tijdens het ontwikkelen werd duidelijk dat bepaalde functionaliteit niet of nauwelijks haalbaar is om goed te kunnen implementeren.

Tkinter heeft minimale ondersteuning voor complexe grafische elementen. De library geeft weinig controle over de volgorde waarin elementen weergegeven worden. Om toch bepaalde visuele feedback goed te kunnen realiseren is een significante hoeveelheid tijd besteed aan het creëren van deze functionaliteit. Door een extra abstractie laag toe te voegen was het mogelijk om gemakkelijker complexere visuele elementen toe te voegen en de render volgorde te bepalen. Maar het in- en uitzoomen van de virtuele camera zou veel extra werk kosten waardoor er voor gekozen is het niet te implementeren.

Daarnaast was het lastig om de code en andere resources goed te organiseren naarmate de complexiteit van de applicatie toe nam. Bijna alle functionaliteit van de CGT zit verborgen in right click context menu's waarbij alleen veelgebruikte functionaliteit beschikbaar zijn als knop naast het betreffende component. Dit is in principe een nette manier om de interface overzichtelijk te houden maar het gebruik van context menu’s komt ook vanuit de beperkingen van Python. Tkinter beschikt over een aantal basiselementen maar een complete en voornamelijk dynamische UI is lastig te onderhouden.

8 Card Game Toolkit 2.0

De usability test heeft aangetoond dat het mogelijk is voor nieuwe gebruikers om snel een model te maken van een traditioneel kaartspel. Maar experimenteren met nieuwe gameplay elementen is op dit moment beperkt. De gebruiker kan componenten alleen verplaatsen met behulp van acties. Er waren zelfs een aantal deelnemers, waaronder meneer Kamp, die de behoefte hadden om buiten de structuur van het spel de componenten te verplaatsen. Door bijvoorbeeld een kaart op een stapel leggen.

Het is in de CGT 1.0 niet mogelijk om zelf kaarten te maken. Dit is een belangrijke feature die op dit moment veel tijd in beslag neemt in het ontwerpproces van meneer Kamp.

8.1 Doelstelling

Een van de belangrijke aspecten die nog mist in de eerste versie van de CGT is het creëren en aanpassen van kaarten. De gebruiker moet zelf kaarten kunnen ontwerpen voor zijn spel. Dit is ook een onderdeel van het ontwerpproces van meneer Kamp wat veel tijd in beslag neemt. Daarnaast moet het gemakkelijk zijn om aanpassingen te maken na een playtest. Op dit moment kost het veel extra tijd en moeite om alle kaarten te bewerken.

Daarnaast moet het mogelijk zijn om de kaarten goed te kunnen organiseren. Een van de voordelen van een digitale omgeving is de mogelijkheid om snel te kunnen zoeken naar specifieke kaarten of ze te sorteren op verschillende eigenschappen. De manier van organiseren moet overzichtelijker zijn dan een lijst van kaarten in een excel document.

Als laatste moet er functionaliteit worden toegevoegd aan het formele model zodat de ontwerper meer vrijheid heeft in het manipuleren van de spel componenten. Hij moet kunnen experimenteren met de componenten alsof hij ze op tafel heeft liggen. Zeker in de eerste fase van het ontwerpproces is het belangrijk dat het ontwerp kneedbaar is en niet direct aan een formele structuur moet voldoen.

De ontwerper moet kaarten kunnen maken, ze kunnen organiseren en ze in een digitale omgeving kunnen plaatsen waarin hij ze vrij kan manipuleren.

8.3 Workflow

De workflow van de CGT 2.0 moet er op ingericht zijn dat de gebruiker kaarten kan kan ontwerpen en aanpassen. Hij moet deze kaarten kunnen organiseren en in een virtuele omgeving kunnen plaatsten om met zijn ontwerp te kunnen spelen.

De layout van de Card Editor is vergelijkbaar met image editing software aangezien het productie proces vergelijkbaar is met de workflow van deze software. Daarnaast is meneer Kamp bekend met Photoshop en kan het helpen als we zoveel mogelijk van deze workflow over kunnen nemen.

In tegenstelling tot traditionele image editing software is de tool bewust van de context. Photoshop, Illustrator en Blender zijn ontworpen om zo generiek mogelijk te zijn. De CGT 2.0 kan het process versnellen omdat het bewust is van het eindproduct. De tool faciliteert het ontwerpen en aanpassen van kaarten.

We onderscheiden de kaarten in twee onderdelen: de informatie van de kaart, de eigenschappen die belangrijk zijn voor de context van het spel, en de presentatie, de visuele weergave van deze eigenschappen.

Het idee van de workflow is dat de gebruiker een template kan maken van een kaart en deze toepassen op meerdere kaarten. Het template bepaald welke eigenschappen een kaart heeft en hoe deze worden weergegeven maar de waardes van deze eigenschappen behoren tot de kaart zelf. De kaarten delen hiermee een ontwerp dat per kaart kan worden ingevuld.

Als de informatie van een kaart verandert past de presentatie zich aan om dit te reflecteren. Aanpassing op een template zijn zichtbaar op alle kaarten die gebruik maken van dat template. Het zou hierdoor gemakkelijker moeten zijn om te experimenteren met verschillende waardes of de vormgeving van de kaarten.

Als de gebruiker veel kaarten heeft gemaakt is het handig als hij deze kan onderverdelen in verschillende collecties. Deze collecties helpen de gebruiker om vergelijkbare kaarten te groeperen en deze van elkaar te onderscheiden. Een collectie kan ook gebruikt worden als een deck kaarten. De CGT 2.0 kan per collectie informatie geven over de inhoud van een collectie. Welke eigenschappen zitten er veel in, welke waardes hebben deze eigenschappen. Dit maakt het overzichtelijk voor de ontwerper welke kaarten er in zijn spel zitten en wat de verhouding is van de eigenschappen van deze kaarten.

8.4 Het Functionele Ontwerp

Het ontwerp voor de CGT 2.0 bestaat uit drie onderdelen: de Card Editor, Collections en de Table. Hieronder volgt het initiële ontwerp voor de tweede versie van de Card Game Toolkit.

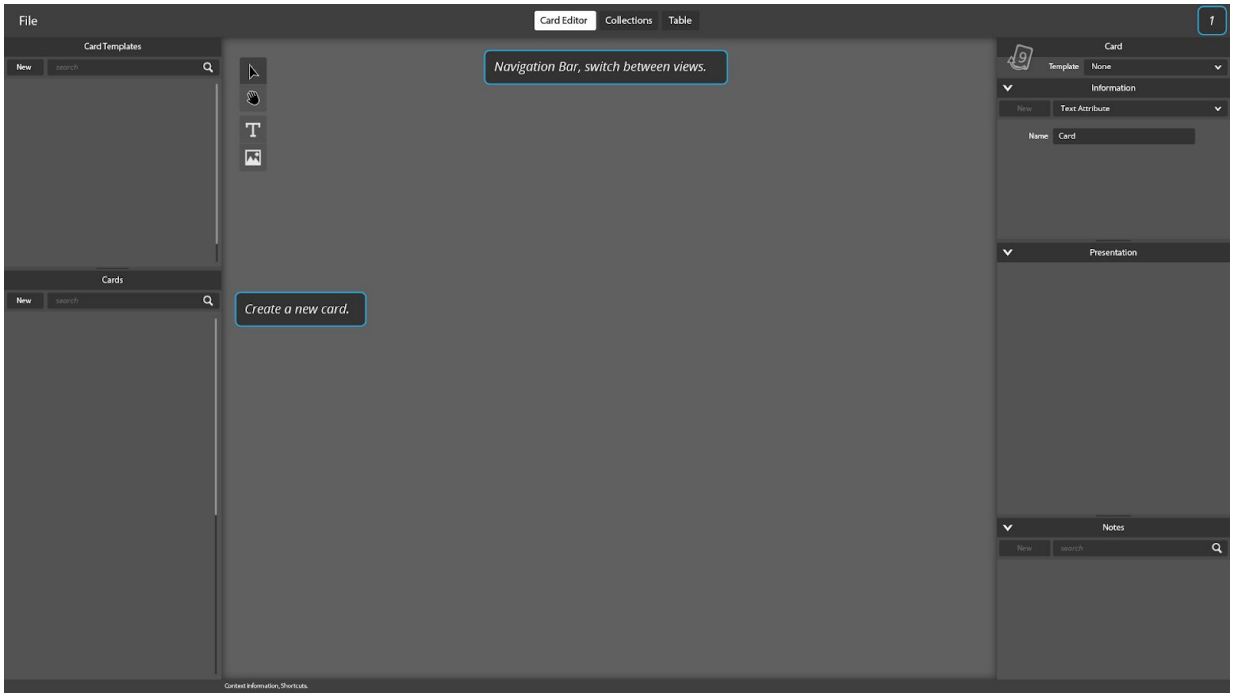
8.4.1 Card Editor

De Card Editor moet de gebruiker helpen om kaarten te maken voor zijn spel. De ontwerper moet gemakkelijk kaarten kunnen ontwerpen en aanpassen.

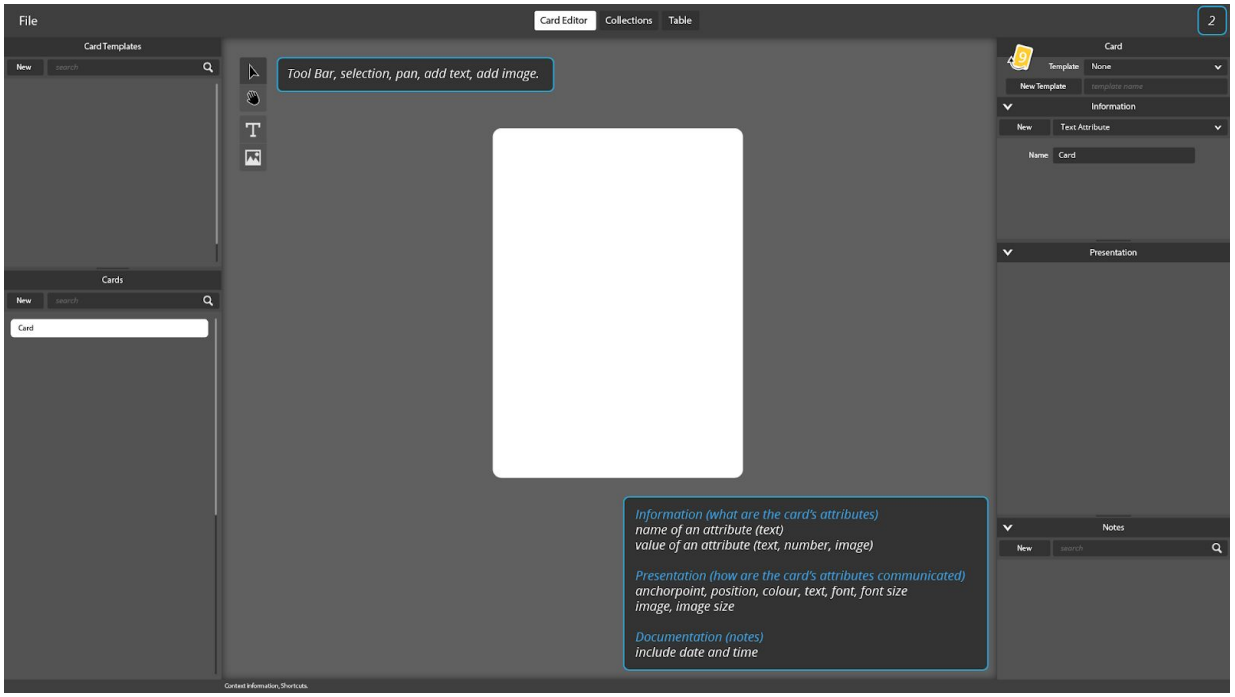
Een kaart bestaat uit twee onderdelen de kaart *informatie* en de kaart *presentatie*. De informatie is een collectie van eigenschappen die een betekenis hebben binnen de context van het spel. De presentatie is de visuele weergave van deze informatie.

De ontwerper kan een kaart maken, een aantal eigenschappen toevoegen en deze eigenschappen gebruiken als onderdeel van grafische elementen. Als de informatie van een kaart verandert zijn deze aanpassing direct zichtbaar op de kaart. Zo kan de ontwerper gemakkelijk de waardes van een kaart aanpassen of experimenteren met de vormgeving zonder dat de eigenschappen van de kaart verloren gaan.

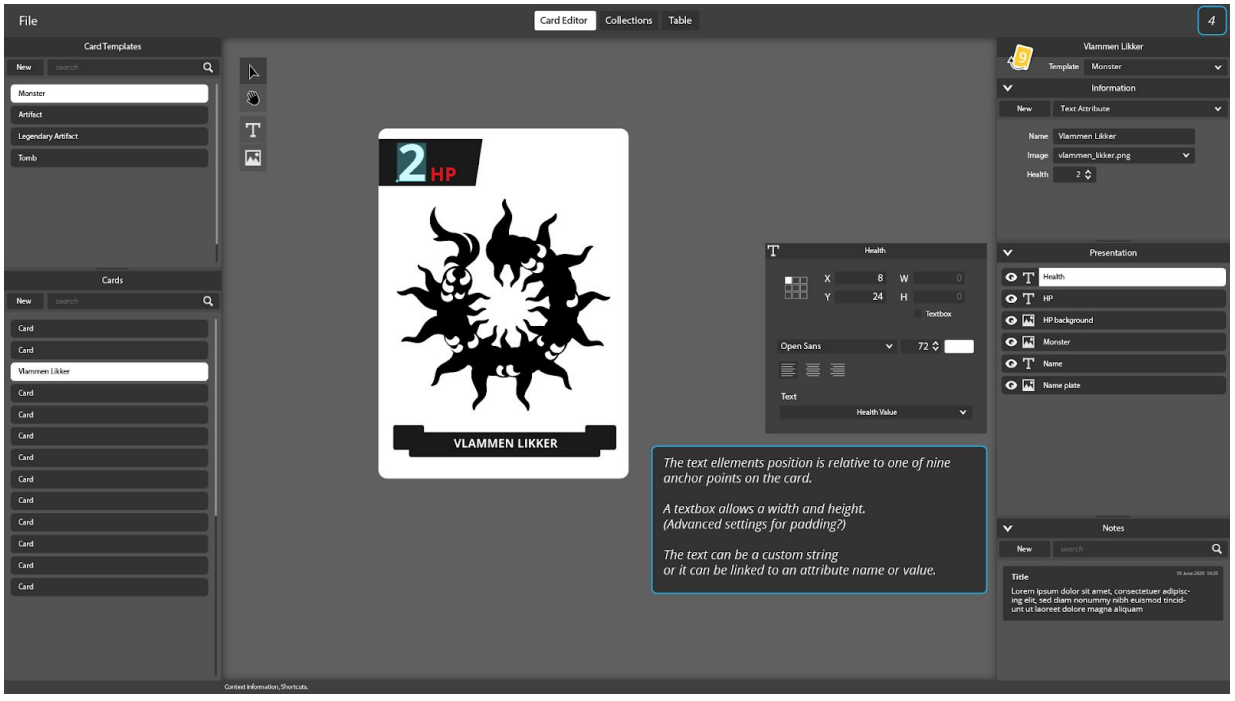
Daarnaast kan de ontwerper een template maken van een kaart. Een template bestaat net als een kaart uit eigenschappen en grafische elementen. Als een kaart gebruik maakt van een template worden alle aanpassingen aan de kaart doorgevoerd op de template, en alle kaarten die gebruik maken van de template. De waardes van eigenschappen zijn uniek per kaart, en worden dus niet overgenomen door een template of andere kaarten.



Mock-up CGT 2.0 (Een leeg project.)



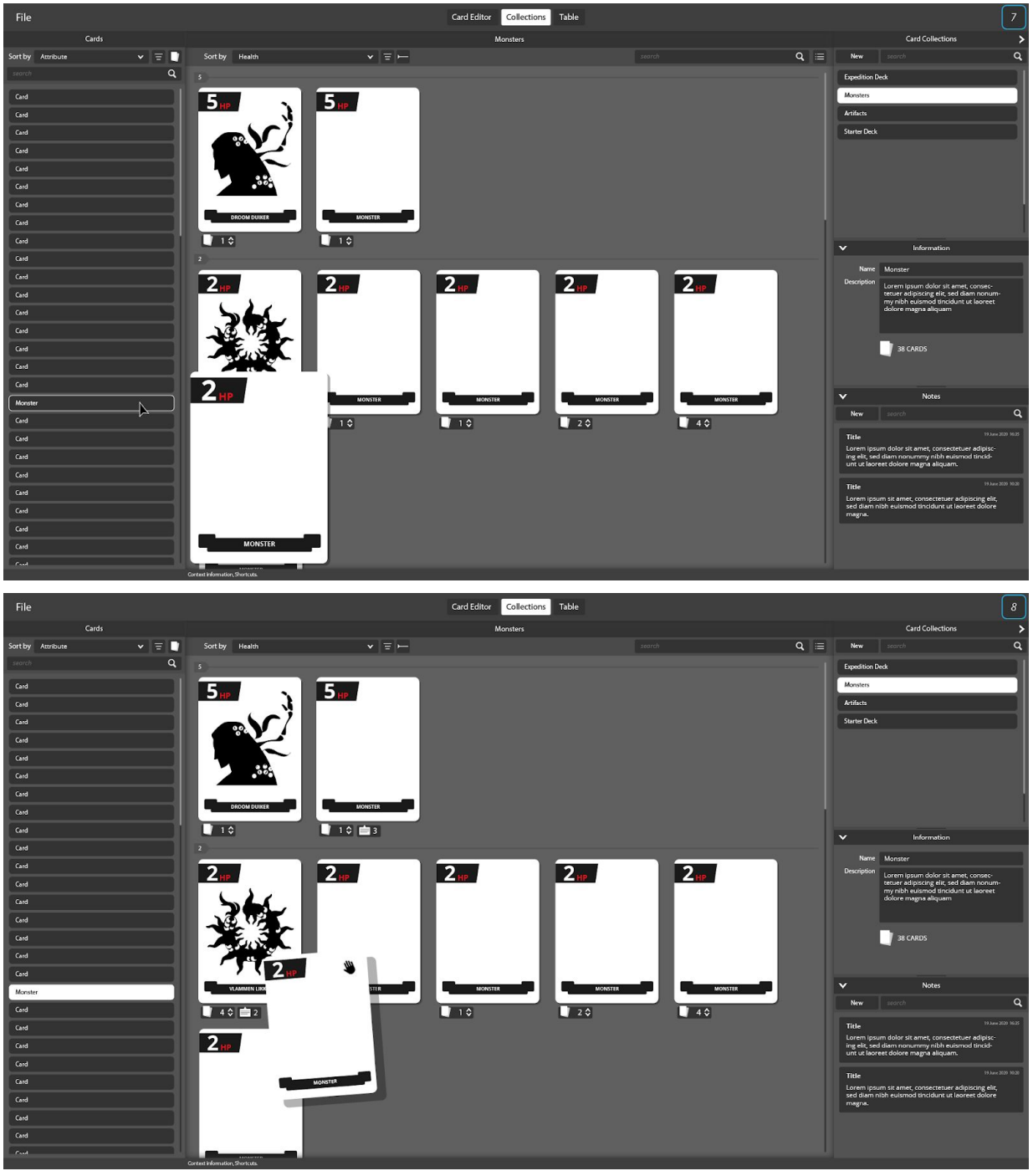
Mock-up CGT 2.0 (Het aanmaken van een nieuw kaart.)



Mock-up CGT 2.0 (Het aanpassen van de verschillende grafische elementen.)

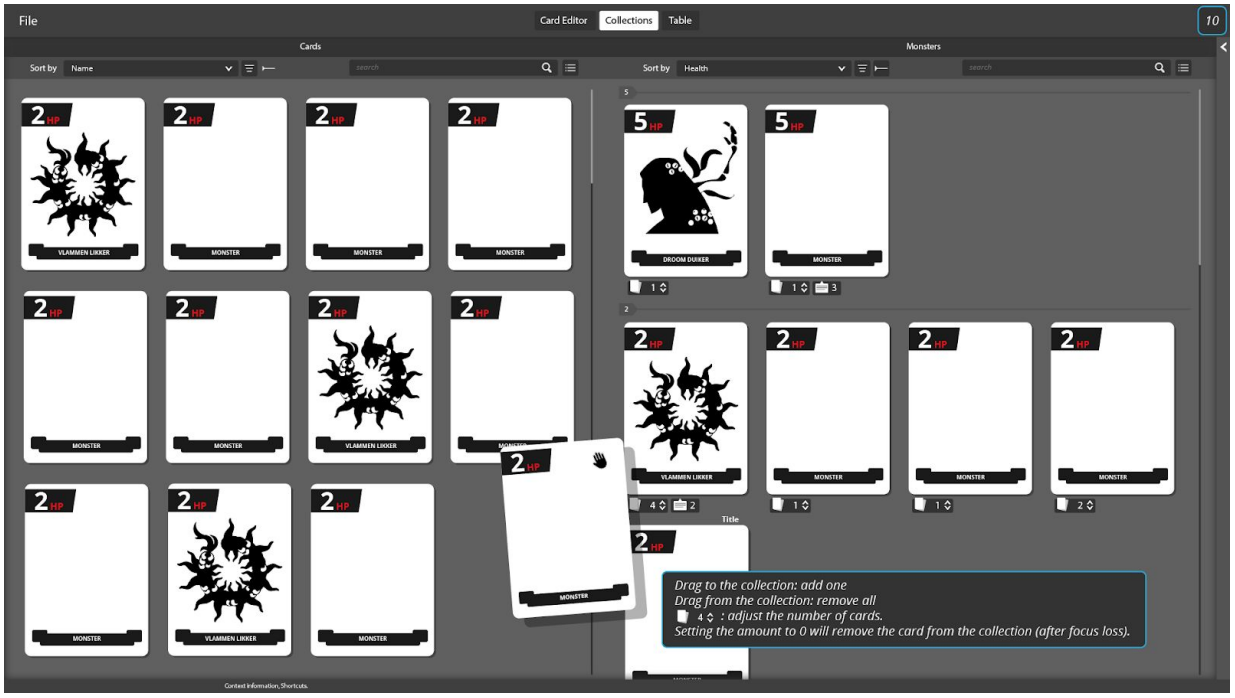
8.4.2 Card Collections

De gebruiker moet de kaarten kunnen groeperen in verschillende collecties. De collecties moeten de ontwerper meer overzicht geven in de verschillende soorten kaarten. Hij heeft de mogelijkheid om collecties te filteren op verschillende eigenschappen, of te zoeken naar specifieke waardes. Een kaart kan in meerdere collecties bestaan, en een collectie kan meer kopieën van dezelfde kaart bevatten. De gebruiker kan op deze manier verschillende decks maken die hij bijvoorbeeld kan exporteren om te printen.

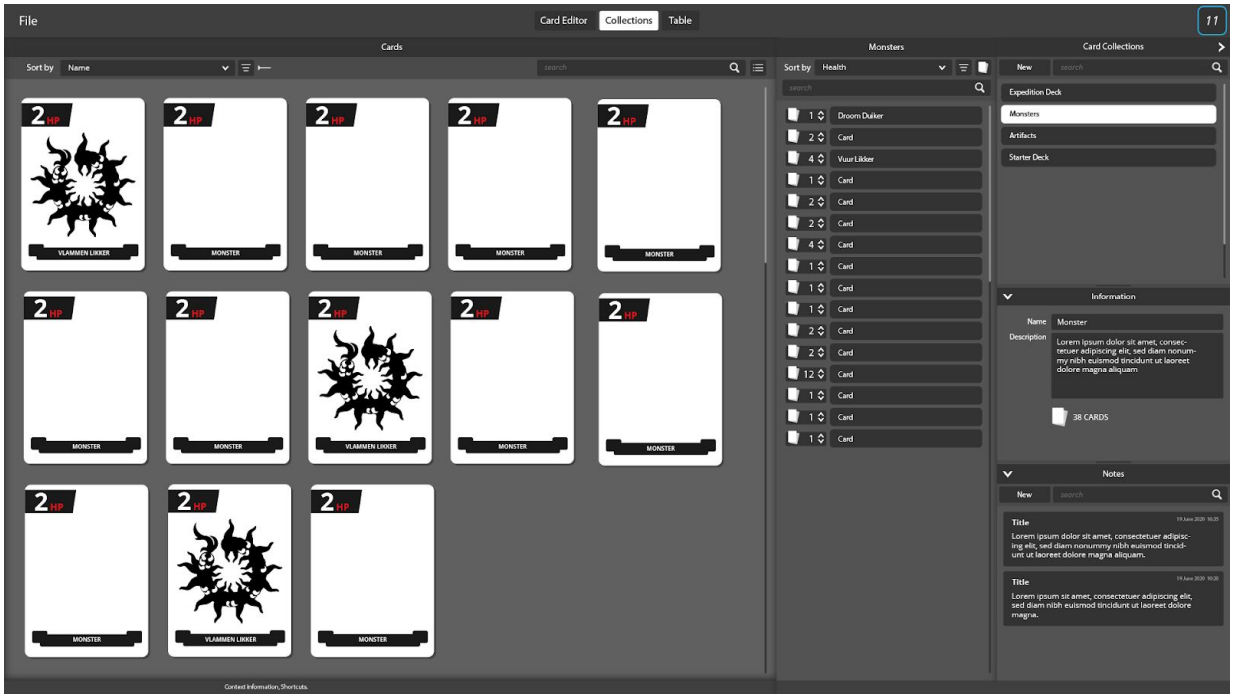


Mock-up CGT 2.0 (Het toevoegen van kaarten aan een collectie.)

Er kunnen twee collecties tegelijkertijd opstaan waarbij de gebruiker kaarten heen en weer kan slepen. Een collectie kan weergegeven worden als lijst van namen of thumbnails.



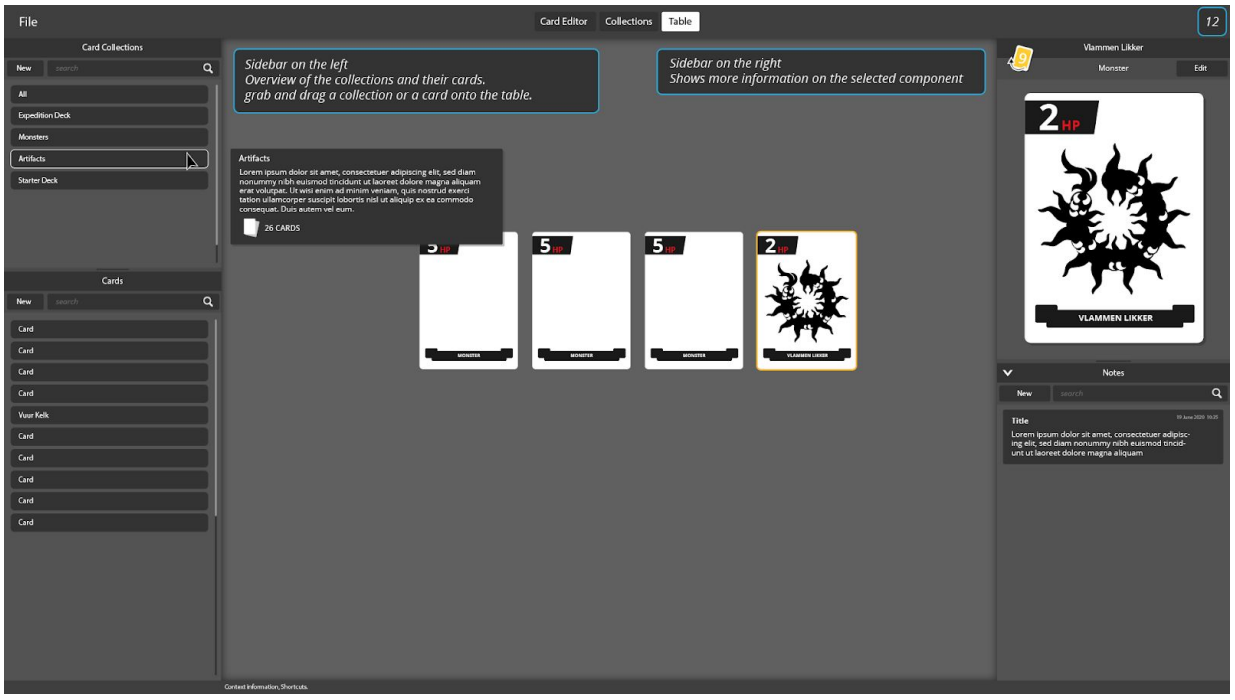
Mock-up CGT 2.0 (Het toevoegen/verwijderen van kaarten.)



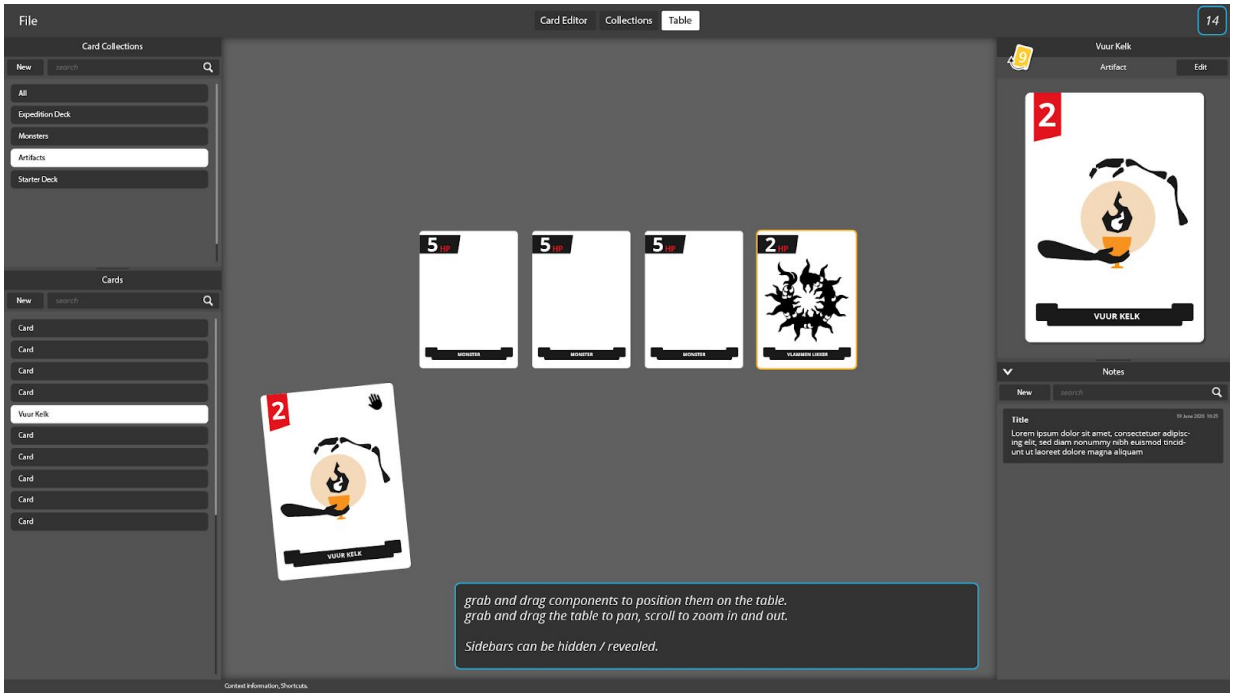
Mock-up CGT 2.0 (Verschillende views.)

8.4.3 Table

De Table is vergelijkbaar met de eerste versie van de CGT, maar de gebruiker heeft de mogelijkheid om componenten direct te manipuleren. Naast het definiëren van acties kan de gebruiker componenten handmatig kaarten verplaatsen, op een stapel leggen, omdraaien, etc. De ontwerper moet kunnen experimenteren alsof hij de kaarten voor zich op tafel heeft liggen.



Mock-up CGT 2.0 (Tafel overview.)



Mock-up CGT 2.0 Het toevoegen van een kaarten aan de tafel)

8.5 Card Editor Uitgewerkt

Het ontwerp van de CGT 2.0 is vrij complex en de scope bleek te groot om alle onderdelen volledig te realiseren. Om te bepalen wat de juiste scope zou moeten zijn, zijn de verschillende onderdelen tegen elkaar afgewogen. Hoeveel tijd kost het om elk onderdeel te realiseren en wat levert het op. Meneer Kamp heeft aangegeven veel tijd kwijt te zijn aan het maken en bijhouden van kaarten voor zijn spel. Met een Card Editor zouden we relatief veel winst moeten kunnen behalen. We hebben daarom gekozen om in ieder geval de Card Editor uitwerken.

8.5.1 Project Menu

New Project: Maak een nieuw project.
Open Project: Open een bestaand project (selecteer de project folder).
Save Project: Sla het project op.

8.5.2 Card List

New: Maak een nieuwe kaart.
De naam van de kaart kan in de inspector worden aangepast.

8.5.3 Inspector

De inspector laat de eigenschappen van de geselecteerde kaart zien. De gebruiker kan de naam van de kaart aanpassen, een template selecteren, en de informatie en presentatie aanpassen.

8.5.4 Card Template

New Template: Maakt een nieuwe template van de geselecteerde kaart.

Als een nieuw template gemaakt wordt van de huidige kaart neemt het template de informatie en presentatie van de kaart over, **behalve de waarde van de informatie**. De waarde zijn voor elke kaart uniek.

Als een bestaande template aan een kaart wordt toegevoegd, **behoud de kaart alleen informatie die ook op de template staat!** De kaart neemt de presentatie over van de template.

bijvoorbeeld, een template “Treasure” heeft de eigenschappen “Goud” en “Omgeving”. Als de gebruiker de “Treasure” template selecteert voor een kaart met de eigenschappen “Goud = 7” en “Type = Ring” behoudt de kaart de “Goud” eigenschap, krijgt het een “Omgeving” eigenschap zonder waarde en verliest het de eigenschap “Type”.

Als de kaart geen gebruik meer maakt van een template neemt hij de informatie en presentatie van de het template over.

8.5.5 Information

New: Voeg een nieuwe eigenschap toe, aan de kaart en aan de template als de kaart daar gebruik van maakt.

Eigenschappen hebben een naam en een waarde. De waarde van een eigenschap kan een tekst (“Text Attribute”) of een nummer (“Number Attribute”) zijn.

Text Attribute: De waarde is een stuk text. Gebruik het voor namen, types of beschrijvingen van kaarten.

Number Attribute: De waarde is een nummer. Gebruik het voor numerieke eigenschappen van een kaart, bijvoorbeeld “Kost”, “Schade” of “Punten”.

8.5.6 Presentation

New: Voeg een nieuw grafisch element toe, aan de kaart en aan de template als de kaart daar gebruik van maakt.

Een grafisch element kan een tekst (“Text Element”) zijn of een afbeelding (“Image Element”).

Text Element: Een text kan een *custom text* zijn of de waarde van een attribute over nemen. Een custom text kan ook waardes van attributen bevatten, plaats de naam van de eigenschap tussen blokhaken “[attribute_name]” en het wordt vervangen door de juiste waarde.

Tekst tussen “[ ]” wordt vervangen door de waarde van de genoemde eigenschap. Als de eigenschap niet bestaat wordt de tag verwijderd. De tag [cardname] kan gebruikt worden om te verwijzen naar de naam van de kaart.

Image Element: Kies een afbeelding en een schaal. Het is nog niet mogelijk om zelf afbeeldingen toe te voegen.

8.6 Het Technische Ontwerp

8.6.1 Godot Game Engine

Tijdens het ontwikkelen van de CGT 1.0 bleek dat Python niet de juiste omgeving is voor het renderen en manipuleren van complexe grafische elementen.

De CGT en mogelijke verbeterinigen kunnen alleen gerealiseerd worden met behulp van flexibele tools die complexe UI elementen mogelijk maken. De open source game engine Godot lijkt een uitstekende kandidaat. Godot beschikt over uitgebreide features met betrekking op UI. In tegenstelling tot beschermde software hebben we ook toegang tot de source code in het geval dat we aanpassingen willen maken aan de engine. Dit zorgt er ook voor dat de CGT onderhoudbaar is omdat alle code toegankelijk is.

Het gebruik van een game engine geeft niet alleen de mogelijkheid om snel visuele feedback toe te voegen maar ook om beter te organiseren. Als het project moet kunnen groeien en complexere structuren moet kunnen visualiseren is Python niet de juiste oplossing.

Godot is een gratis open source game engine. Godot beschikt over visuele tools voor het ontwerpen en uitwerken van een complexe UI. Het maakt het zeer gemakkelijk om de virtuele camera in en uit te laten zoomen, en zwaardere grafische elementen te gebruiken.

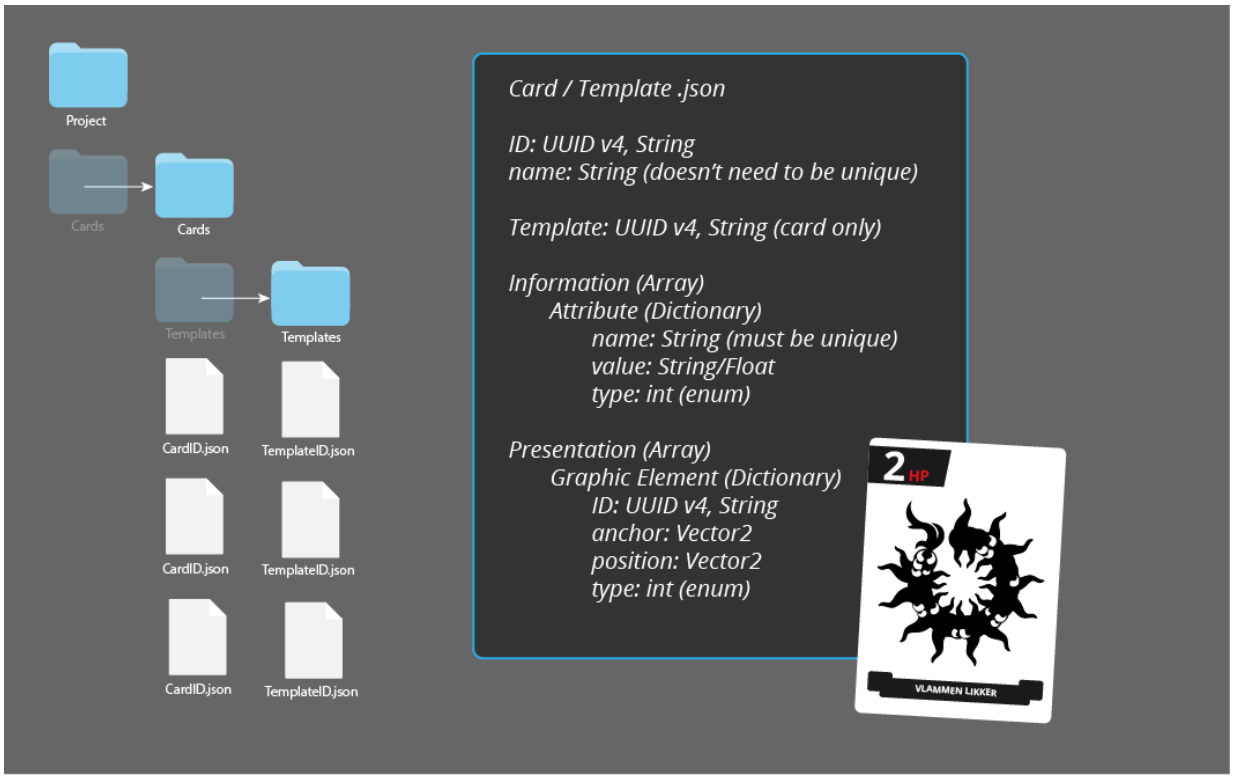
Godot maakt gebruik van verschillende programmeertalen waaronder C++, C# maar ook GDScript, een taal die nauw geïntegreerd is in de development omgeving. Dit maakt het mogelijk om snel te itereren en bugs gemakkelijk te identificeren. GDScript is vergelijkbaar met Python dus de overgang naar Godot was gauw gemaakt.

Daarnaast maakt Godot gebruik van de observer pattern voor communicatie tussen verschillende componenten. Een design pattern die we ook in de eerste versie succesvol hebben toegepast.

8.6.2 Het Data Model

Een CGT project folder bestaat uit een “Cards” folder met daarin een “Templates” Folder. Zowel de kaarten als de templates hebben een UUID (Universally Unique Identifier), een uniek 128-bit nummer. Deze ID’s worden gebruikt als key in een dictionary die alle kaarten bijhoud, ze maken het gemakkelijk om de relatie tussen verschillende componenten op te slaan en ze zijn een uitstekende bestandsnaam.

We slaan de kaarten en templates op als individuele json bestanden. Dit maakt de data niet alleen heel leesbaar maar ook makkelijk te gebruiken door andere onderdelen van de CGT of andere programma’s.



Data Model (Project folder, card en template data.)

Na het openen van een project laad de applicatie de data van de verschillende json files als Godot resource files. Het voordeel hiervan is dat het makkelijk is om data te manipuleren en signalen te sturen vanuit de resource wanneer deze veranderd. Hierdoor kunnen andere componenten luisteren naar veranderingen in de data en daarop hun gedrag aanpassen.

De applicatie bewaard de kaarten en templates in een dictionary waarbij de ID van de resource de key is, dit maakt de toegang tot een kaart O(1) op het moment dat een script de ID van een kaart weet.

8.6.3 Undo/Redo

Voor het ontwikkelen en ontwerpen van een goede UX en UI is het verstandig om naar vergelijkbare programma’s te kijken. De interface en functionaliteit is gebaseerd op applicaties zoals Photoshop, Illustrator, Blender en andere image editing software. Hierdoor zou het gemakkelijk en natuurlijk moeten aanvoelen voor iemand met ervaring met één van deze programma's om zich door de interface van Card Editor te navigeren.

Ook in de functionaliteit verwacht de gebruiker dat hij bepaalde handelingen kan verrichten, zoals bijvoorbeeld het herstellen van fouten met behulp van een undo / redo feature. Het is als gebruiker zeer frustrerend als je per ongeluk een verkeerde handeling uitvoert die niet meer omkeerbaar is. Dit valt ook onder een van de tien heuristieken van Nielsen “user control and freedom”.

Command Pattern
De command pattern is perfect om een structuur te implementeren zodat de gebruiker zijn acties terug kan draaien.

In de CGT 2.0 is elke actie uitgedrukt in een command object. Een command bestaat uit data, een execute functie, een undo en redo functie.

```
1  class_name Command
2  extends Reference
3
4  var description: String
5
6
7  func _init():
8      # store the original state
9      # and the parameters for the execute function
10     pass
11
12
13 func execute():
14     # execute the command, using the provided parameters
15     pass
16
17
18 func undo():
19     # return to the original state
20     pass
21
22
23 func redo():
24     # execute the command (again)
25     execute()
26
```

Command.gd

De redo functie is in de meeste gevallen een alias voor de execute functie. Maar elk object dat overerft van de command class heeft de mogelijkheid de redo functie te overschrijven. De *CreateCard* command maakt een kaart aan. Na het uitvoeren van de command verschijnt de beschrijving in de statusbar van de applicatie, zo weet de gebruiker welke acties hij onderneemt en wat hij dus kan terugdraaien.

```
class_name CreateCard
extends Command

var card_id


func _init(card_id = null):
    if not card_id:
        # generate a new ID
        self.card_id = UUID.v4()
    else:
        self.card_id = card_id

    description = 'Create Card (ID: %s)' % self.card_id


func execute():
    var card = CardData.new(card_id)
    card.name = "Card"

    # add the card to the project library
    Global.card_library[card_id] = card

    Global.emit_signal('card_created', card_id)


func undo():
    # remove the card from the project library
    Global.card_library.erase(card_id)

    Global.emit_signal('card_destroyed', card_id)

```

CreatCard.gd

Commander

Naast de commands is er een commander: een singleton script met als functie het uitvoeren en herstellen van commands. De commander heeft twee stacks, een undo stack en een redo stack. Nadat een actie is uitgevoerd plaatst de commander deze bovenop de undo stack. Als de gebruiker een actie hersteld pakt de commander het bovenste command van de undo stack, roept de undo functie aan en plaats deze bovenop de redo stack. Als de gebruiker de actie toch wilt uitvoeren pakt de commander het eerste command van de redo stack en voert de redo functie uit. De commander wist de redo stack als hij een nieuw command uitvoert, zodat de geschiedenis en de toekomst op elkaar aansluiten.

```
8  # undo/redo
9  var undo_stack = []
10 var redo_stack = []
11 var max_undo_commands = 100
12
13
14 func _unhandled_input(event):
15     # check redo first because it's the shifted version of undo's shortcut
16     if event.is_action_pressed('redo'):
17         redo()
18     elif event.is_action_pressed('undo'):
19         undo()
20
21
22 # execute and keep track of commands
23 func execute(command: Command):
24     # clear the command history
25     redo_stack.clear()
26     # execute the command and add it to the undo stack
27     command.execute()
28     undo_stack.push_back(command)
29
30     if undo_stack.size() > max_undo_commands:
31         undo_stack.pop_front()
32
33     emit_signal('command_executed', command.description)
```

Commander.gd (execute wist de toekomst en verplaatst de command naar de undo stack.)

Het is belangrijk om rekening te houden dat de undo de staat van het systeem perfect herstelt en dat de redo functie consistent dezelfde verandering toepast, willekeurigheid moet opgeslagen worden in het command object.

```gdscript
# undo the last executed command
func undo():
    # check if there is an undo available
    if undo_stack.size() > 0:
        # undo the last executed command and add it to the redo stack
        var command = undo_stack.pop_back()
        command.undo()
        redo_stack.push_back(command)

        emit_signal('command_undone', command.description)


# redo the last undone command
func redo():
    # check if there is a redo available
    if redo_stack.size() > 0:
        # redo the last undone command and add it to the undo stack
        var command = redo_stack.pop_back()
        command.redo()
        undo_stack.push_back(command)

        if undo_stack.size() > max_undo_commands:
            undo_stack.pop_front()

        emit_signal('command_executed', command.description)
```

Commander.gd (commands worden van de undo naar redo verplaatst en vice versa)

8.6.4 Observer Pattern

De CGT 1.0 maakt gebruik van een observer pattern om de functionaliteit zo min mogelijk dependencies te laten creëren. Deze strategie hebben we wederom toegepast in de CGT 2.0 met behulp van Godot's *signals*. Naast de mogelijkheid om zelf signalen te creëren hebben objecten in Godot een aantal ingebouwden signalen die verstuurd worden na aanleiding van specifieke events. Een object kan zich verbinden met een signaal, zodat het kan reageren op veranderingen van de zender.

Het gebruik van signalen werkt zeer goed samen met het eerder beschreven command pattern. Een command maakt bijvoorbeeld een nieuwe kaart en verzend daarna een signaal dat er een kaart is aangemaakt. Verschillende elementen van de UI zijn verbonden met het 'card_created' signaal en kunnen zich updaten om de nieuwe staat te reflecteren.

Het is belangrijk om te rekening te houden met het feit dat de volgorde waarin objecten het signaal ontvangen niet altijd hetzelfde zijn.

8.7 Usability Test

8.7.1 Doel

Het doel is het bepalen van de bruikbaarheid en de efficiency van de Card Game Toolkit Card Editor. Meneer Kamp beschrijft zijn ontwerp voornamelijk in google docs en maakt de componenten met behulp van Photoshop. Hij zou in staat moeten zijn om kaarten van zijn Spellbenders kaartspel prototype in een kortere tijd binnen de Card Editor te realiseren. Daarnaast zou hij deze kaarten een stuk gemakkelijker moeten kunnen aanpassen, waarbij hij niet voor elke kaart individueel veranderingen moet doorvoeren.

8.7.2 Opzet

Om de kwaliteit van de Card Editor goed te kunnen testen gebruiken vragen we meneer Kamp een deel van zijn ontwerpproces te herhalen zoals hij dat heeft toegepast in het maken van zijn prototype tijdens het dag-in-het-leven onderzoek (beschreven in hoofdstuk 5).

Net als de eerste usability test maken we gebruik van een think-aloud protocol waarbij zowel het scherm als de stem van de participanten wordt opgenomen. De participant zal tijdens het testen zijn gedachten uitspreken zodat de observant inzicht kan krijgen in het denkproces van de gebruiker.

Meneer Kamp krijgt een korte introductie opdracht om zich bekend te maken met de functionaliteit van de Card Editor. Hij is gewend om met Photoshop te werken, om een goede vergelijking te maken tussen de verschillende workflows is het belangrijk dat hij de functionaliteit van de Card Editor begrijpt. Bij de eerste opdracht observeren we of de functionaliteit van de Card Editor duidelijk is voor iemand die de tool voor het voor de eerste keer gebruikt.

Voordat meneer Kamp aan de volgende opdracht kan beginnen is het belangrijk dat hij de workflow van van de Card Editor begrijpt. De observant zal hem vragen de functionaliteit van de tool te beschrijven. Daarna wijst de observant hem op onderdelen die hij eventueel gemist heeft.

Daarna krijgt meneer Kamp de tijd om alle kaarten van zijn eerdere ontwerp te maken in de Card Editor. Het uiteindelijke product en de tijd die het hem kost om dit te maken kunnen we hierna vergelijken met het maken van de kaarten met behulp van Photoshop.

Hierna zal de observant hem vragen een aantal aanpassing terug te halen die hij in vorige prototypes heeft toegepast. Waarna meneer Kamp een aantal veranderingen moet op schrijven alsof hij net een playtest heeft gedaan. Het is onbelangrijk of het daadwerkelijk goede aanpassing zijn zolang het in lijn ligt met het soort aanpassingen die hij zou maken na het testen van zijn spel. Na deze playtest simulatie zal hij deze aanpassingen doorvoeren in zowel zijn eerste versie van het prototype als de versie die hij zojuist in de Card Editor heeft gemaakt.

Nu kunnen we de efficiency en het uiteindelijke resultaat met elkaar vergelijken.

8.7.3 Resulaten

Alle tijden zijn afgerond op halve minuten zodat het beter leesbaar is.

Taak	Card Editor	Photoshop
Het maken van één Spell kaart	40min	-
Het maken van 14 Spell kaart	23½min	-
Het maken van één Resource kaart	6min	-
Het maken van 14 Resource kaart	8½min	-
Het maken van één Item kaart	3½min	-
Het maken van 4 Item kaart	3½min	-
Totaal	1h 25min	2h 27min

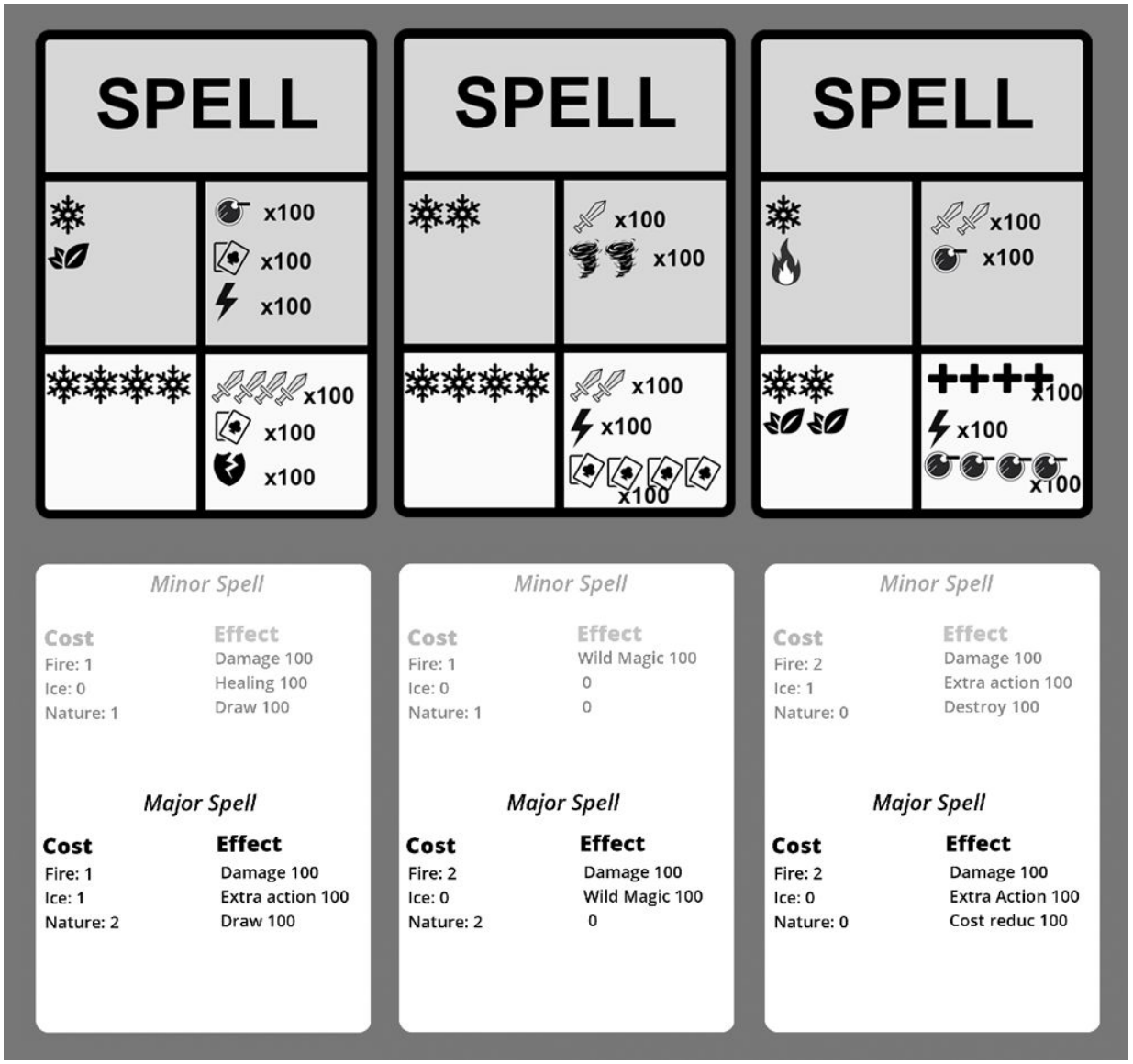
Tijdsduur van het maken van de kaarten (bij de eerste test is alleen de totale tijd gemeten).

Vanuit de gesimuleerde playtest heeft meneer Kamp vastgesteld dat een aantal kaarten niet sterk genoeg waren, en hun waardes aangepast moeten worden. Daarnaast vonden spelers de impact van de kaarten niet bij het thema passen, en ze vonden de twee effecten van de kaarten lastig in een oogopslag te onderscheiden.

Voor het aanpassen van de kaarten heeft hij de waardes van 10 kaarten verhoogt om ze sterker te maken. Daarna heeft hij alle waardes vermenigvuldigt met 100 zodat kaarten sterker lijken. Als laatst heeft hij het sterke effect van de Spell kaarten opvallender gemaakt ten opzichte van het zwakke effect.

Taak	Card Editor	Photoshop
Waarde aanpassen van 15 kaarten	3½min	4min
Alle waardes aanpassen van 15 kaart	3min	6min
Stijl van één Spell kaart aanpassen	½min	3min
Stijl van 14 Spell kaart aanpassen	geen extra tijd (½min)	4½min
Totaal	7min	17½min

Tijdsduur van het aanpassingen van de kaarten na de gesimuleerde playtest.



Bovenaan de kaarten van het eerste ontwerp in Photoshop, onderaan de kaarten gemaakt in de Card Editor.

Na afloop heeft meneer Kamp een aantal vragen beantwoord over zijn ervaring met de Card Editor.
Hij vond de interface duidelijk en de workflow makkelijk te leren. Hij vond het wel duidelijk dat als hij een template wou aanpassen, hij een van de kaarten dat gebruik moest selecteren. Hij miste de mogelijkheid om een template zonder data aan te kunnen passen.

Hij miste functionaliteit om de naam van grafische elementen aan te passen of eigenschappen van kaarten een andere naam te geven. En de mogelijkheid om de lijst van deze elementen te ordenen.

Als er meer functionaliteit zou worden gegeven, bijvoorbeeld om zelf afbeeldingen toe te voegen en de anchors van grafische elementen te veranderen zou hij het een zeer nuttige tool vinden. Meneer Kamp zou het ook fijn vinden als hij gelijk zijn kaarten kan testen in dezelfde omgeving.

8.7.4 Discussie

Het gebruik van de Card Editor heeft de productietijd van de kaarten flink gereduceerd: met één uur, dat is 40% sneller dan de tijd die nodig was om de kaarten in Photoshop te maken. De meeste tijdswinst lijkt zoals verwacht te komen door gebruik te maken van templates. Omdat vergelijkbare kaarten dezelfde vormgeving delen is het na het ontwerpen alleen een questie van data invullen. Dit zou in de toekomst eventueel geautomatiseerd kunnen worden.

Daarnaast is het opvallend dat voornamelijk het aanpassen van de kaarten een ruim twee keer zo snel te verwerkelijken is in de Card Editor. Zeker als het gaat om aanpassingen in de presentatie van de informatie. Waarbij meneer Kamp in Photoshop elke kaart individueel moet aanpassen hoeft hij in de Card Editor maar één kaart te veranderen.

Er is nog wel winst te behalen in het aanpassen van de waardes. Het is nu niet mogelijk om meerdere kaarten te selecteren en alle eigenschappen in een keer aanpassen. Daarnaast zou het kunnen helpen om veel gebruikte waarde te kunnen definiëren zodat hij gewoon een keuze kan maken vanuit een lijst in plaats van het elke keer uit te typen.

Na afloop heeft meneer Kamp een aantal vragen beantwoord over het gebruik van de tool. Hij ervaarde de tool snel en sterk in het maken van kaarten. De templates besparen enorm veel tijd en laten je makkelijk iteraties maken op data. Hij kan zich goed voorstellen dat als er nog meer opties zijn om grafische elementen te manipuleren het een uitstekende tool is.

Hij mist op dit moment een manier om de kaarten direct te kunnen gebruiken in een virtuele omgeving of een optie om de kaarten te kunnen exporteren als pdf.

Als het gaat om de usability van de tool mist hij een manier om de volgorde van de eigenschappen en grafische elementen te veranderen en deze zelf een naam te geven.
De parameters van de grafische element vind hij nu nog beperkt ten opzichte van wat hij gewend is. Functionaliteit om te bepalen hoe bijvoorbeeld een text elementen schaalt, zou hij graag in de tool willen zien.

Hij is over het algemeen erg enthousiast over de werking van de tool.

8.7.5 Conclusie

De CGT 2.0 is erg sterk in het aanpassen van de presentatie van meerdere kaarten. Hierdoor was het voor meneer Kamp mogelijk om de kaarten van zijn prototype in 40% minder tijd te maken met behulp van de Card Editor ten opzichte van zijn oude workflow.

Hij mist nog wel functionaliteit die hij gewend is in PPhotoshop zoals het bepalen van het anker van een element, en hoe een langere text schaalt. Dit zou in een volgende versie toegevoegd kunnen worden.

Helaas is de tijdswinst in het aanpassen van de data niet heel veel sneller dan zijn oude workflow. Een volgend ontwerp zou hier meer aandacht op kunnen focussen. Het zou mogelijk moeten zijn om de waardes van meerder kaarten tegelijkertijd aan te passen. Hoe dit precies moet werken is nog onduidelijk.

9 Conclusies

Het moet mogelijk zijn voor de ontwerper op te experimenteren met zijn ontwerp, hij moet kunnen spelen met de parameters van zijn ontwerp. Zoals bijvoorbeeld de hoeveelheid kaarten die een speler mag pakken of spelen. Welke waardes er op de kaarten moet staan of hoe de doorloop moet zijn van het spel.

Een kaartspel heeft veel verschillende variabelen die het ontwerp definiëren dit maakt het enorm lastig om een formele structuur te creëren waarin elk kaartspel gedefinieerd kan worden. In de CGT 1.0 hebben we geprobeerd om het ontwerp van een kaartspel te abstraheren naar een generieke vorm. De CGT 1.0 maakt het mogelijk een klein stukje van de ontwerp ruimte te beschrijven.

Na de CGT 1.0 te hebben getest op de case van Codeglue bleek een te grote stap in het ontwerpen van een design tool kaartspellen. De formele structuur beperkte de spelontwerpen in het uitwerken van zijn concept.

Voor het ontwerpen van de CGT 2.0 hebben we geprobeerd winst te behalen in het produceren van prototype met als doel om de ontwerper meer tijd te geven die hij kan besteden in de ontwerp- en testfase in plaats van de productiefase.

Door een workflow aanbieden waarbij de ontwerper templates van kaarten kon maken, waar meer kaarten gebruik van kunnen maken is het meneer Kamp gelukt om 40% sneller zijn kaarten uit te werken en het was nog makkelijker om deze kaarten daarna aan te passen.
De CGT 2.0 zorgt er voor dat de tijd die nodig is om een bepaalde hoeveelheid kaarten aan te passen gelijk is een de hoeveelheid tijd die nodig is om één kaart te veranderen.

Meneer Kamp mist nog wel een aantal features die hij gewend is in Photoshop. Hij zou graag een afgebakend tekstveld willen maken en/of kunnen bepalen hoe de tekst van een template schaalt als deze groter is voor bepaalde kaarten tegenover andere kaarten die hetzelfde template gebruiken. Hij wil de grafische elementen een naam kunnen geven om zijn project beter te kunnen organiseren.

De CGT 2.0 lijkt de juiste richting op te gaan. Het verder uitwerken van het ontwerp ziet er veelbelovend uit. Het formele model van de CGT 1.0 is helaas nog niet overtuigend genoeg om direct te implementeren in de volgende versie van de tool.

10 Aanbevelingen

Allereerst kunnen we aanraden om het gemaakte ontwerp verder uit te werken. Een aantal features die meneer Kamp miste in de card editor zijn meegenomen in het ontwerpproces maar door een tekort aan tijd nog niet geïmplementeerd. De MOSCOW lijst in de bijlage en het ontwerp op GitHub geven een overzicht van de geplande features.

Een aantal van de belangrijke features:

- Het maken van collecties van kaarten
- Het filteren op en zoeken naar specifieke eigenschappen van kaarten
- Het hernoemen van attributen en grafische elementen
- Het toevoegen van user created resources zoals afbeeldingen en fonts
- Een aantal export opties om een fysieke versie van het ontwerp te playtest
- Het toevoegen van documentatie aan kaarten en templates
- Het zichtbaar maken van vorige iteraties van een kaart

Daarnaast lijkt het nuttig om het ontwerp uit te breiden zodat het mogelijk is om de eigenschappen van meerdere kaarten tegelijkertijd te manipuleren. Dit zou niet alleen efficiënter zijn maar ook minder gevoelig voor fouten. Het zou ook interessant kunnen zijn om de kaart data automatisch te genereren van een excel document. Hiermee zouden eerder gemaakte ontwerpen in de tool opgenomen kunnen worden.

Hierdoor zou de productiefase vloeiend moeten verlopen. Daarna kan het interessant zijn om te kijken naar de andere fases van het ontwerpproces. Hierbij kan het formele model van de CGT 1.0 inspiratie bieden voor een uitbreiding op de CGT 2.0.

Een CGT 3.0 zou zich kunnen richten op het genereren van data voor verschillende kaarten, of analyze tools kunnen aanbieden om collecties van kaarten te kunnen evalueren.

Bronnenlijst

[1] Bell, C. & Goadrich, M. (2016). *Automated Playtesting with RECYCLEd CARDSTOCK*, *Game & Puzzle Design* (vol. 2), pp. 71-83.

[2] Van Rozen, R. (2015). *A Pattern-Based Game Mechanics Design Assistant.*

[3] Almeida, M.S.O., & Corrêa da Silva, F.S. (2013). *Requirements for game design tools, A Systematic Survey.*

[4] Almeida, M.S.O., & Corrêa da Silva, F.S. (2013). *A Systematic Review of Game Design Methods and Tools.*

[5] Osborn, J.C., Grow, A., and Mateas, M. (2013). *Modular Computational Critics for Games.*

[6] Neil, K. (2012). *Game design tools: Time to evaluate*

[7] Mahlmann, T., Togelius, J. and Yannakakis, G. N. (2012) *Evolving Card Sets Towards Balancing Dominion.*

[8] Thielscher, M. (2011). *The General Game Playing Description Language is Universal. AAAI Publications, Twenty-Second International Joint Conference on Artificial Intelligence.*

[9] Thielscher, M. (2010). *A General Game Description Language for Incomplete Information Games*, AAAI (vol. 10), pp. 994–999.

[10] Font, J.M., Mahlmann, T., Manrique, D. and Togelius, J. (2013). *A Card Game Description Language*, *Applications of Evolutionary Computation*, pp. 254–263.

[11] Nelson, M. J., & Mateas, M. (2009). *A Requirements Analysis for Videogame Design Support Tools. In Proceedings of the 4th International Conference on Foundations of Digital Games*, pp. 137-144.

[12] Salen, K., & Zimmerman, E. (2004). *Rules of Play.* Cambridge, Massachusetts; London, England: The MIT Press.

[13] Hunicke, R., LeBlanc, M. and Zubek, R. (2004). *MDA: A Formal Approach to Game Design and Game Research.*

[14] Grünvogel S. (2004). *Formal Models and Game Design.*

[15] Järvinen, A. (2003). *Making and breaking games: a typology of rules. Level Up: Digital Games Research Conference,* pp. 68-79.

[16] Nielsen Normal Group
https://www.nngroup.com/articles/usability-101-introduction-to-usability/

Bijlage

Card Game Toolkit source en executable
https://github.com/Okatt/CardGameToolkit

Ontwerp CGT 2.0
https://github.com/Okatt/CardGameToolkit/blob/master/ontwerp%20CGT%202.0.pdf

Probleemanalyse Prototyping Process
Achteraan het afstudeerrapport

Dag In het Leven Onderzoek Log
Achteraan het afstudeerrapport

Card Game Toolkit Usability Test
Achteraan het afstudeerrapport

Card Game Toolkit 2.0 Usability Test
Achteraan het afstudeerrapport

Card Game Toolkit 2.0 MOSCOW
Achteraan het afstudeerrapport

Probleemanalyse Prototyping Process

Probleem

Het maken van een (paper) prototype kan een ontwerper helpen om vroegtijdig de succesvolle dan wel spelbrekende elementen van hun ontwerp te identificeren. Codeglue erkent het nut van het maken van (paper) prototypes, desondanks maken ze er weinig gebruik van.

Onderzoeksvraag

Waarom gebruikt Codeglue het maken van paper prototypes nauwelijks tijdens het ontwerpproces?

> *Hypothese: De waarde van de resultaten staat niet in verhouding met de tijdsinverstering.*
>
> *Er is geen gestructureerd proces om snel prototypes te maken. Niemand documenteert voorafgaand aan het proces welke kennis het prototype moet opleveren. De ontwerper noteert zijn ideeën in een text document, hij bedenkt een concept voornamelijk vanuit intuïtie en creëert daarna de benodigde componenten om het te testen.*
>
> *De spelontwerper gebruikt alleen Notepad en Photoshop om het prototyping proces te ondersteunen. Hij beschrijft zijn concept in Notepad. Daarna werkt hij grotendeels in Photoshop. Photoshop is een programma voor het creëren en bewerken van foto en illustraties, en geeft weinig inzicht in de functionaliteit van de spelelementen. Terugkoppeling van een fysieke versie naar Photoshop kost veel tijd.*
>
> *Ontwerpbeslissingen zijn voornamelijk gebaseerd op intuïtie van de ontwerper. Hij kan niet vroeg in het proces steunen op data die zijn ideeën bevestigen of ontkrachten.*
>
> *De spelontwerper print en snijdt de kaarten. Hij simuleert het spelen van het spel door de rollen van alle spelers aan te nemen. Hij krijgt hierdoor inzicht in de interactie van de belangrijkste spelregels en -elementen. De ontwerper maakt hierna de grootste aanpassingen aan het prototype voordat hij gaat playtesten met spelers. Na twee dagen krijgt de ontwerper pas inzicht in de interacties van zijn concept.*
>
> *De spelontwerper legt zijn observaties vast in een text document. Het document bepaald de volgende stappen in het ontwerpproces.*

Het Ontwikkelen van Prototypes voor Codeglue

Semi-Structured Interview met meneer Kamp, Game Designer bij Codeglue.

Maakt Codeglue gebruik van het ontwikkelen van (paper) prototypes?

Hypothese:
Ja.

Ja.

Waarom worden er (fysieke) prototypes gemaakt?

Als Codeglue in het ontwerpproces van hun spellen onbekend terrein betreed en keuzes kunnen niet gebaseerd worden op eerdere kennis, of er zijn weinig of geen voorbeelden te vinden vanuit andere spellen.

Hypothese:
Om gameplay aspecten van onze games te testen. De ontwerper heeft weinig mogelijkheden om aanpassingen te maken binnen de game. Het geeft de spelontwerper een manier om zijn ideeën te testen voordat deze geïmplementeerd kunnen worden.

Wanneer (binnen de workflow van CCodeglue) worden er prototypes gemaakt?

Op het moment dat Codeglue niet weet of een concept werkt maken ze een prototype. Het maken van een fysiek prototype heeft de voorkeur omdat het minder tijd kost dan het ontwikkelen van een digitaal product. Als een idee niet blijkt te werken verlies je bij het maken van fysieke prototypes minder tijd.

Wanneer de complexiteit groot is wordt er voor een digitale versie gekozen. Helaas is dit niet altijd mogelijk omdat er geen programmeurs beschikbaar zijn. Dan probeert de ontwerper het zo goed mogelijk te vertalen naar een fysieke vorm.

Hypothese:
Als er nieuwe ideeën getest moeten worden. Als het niet door de ontwerper in het spel kan worden aangepast maakt hij een fysiek prototype.

Hoe vaak is dit het geval?

Het verschilt enorm, het is onvoorspelbaar. Het inkomen van Codeglue is grotendeels afhankelijk van externe opdrachten, en deze hebben daarom een hogere prioriteit dan de interne projecten.
Het werk voor externe bedrijven is voornamelijk gericht op uitwerking. De spelontwerper heeft hierdoor meer tijd beschikbaar voor Codeglue's eigen spellen.

Hoe vaak worden prototypes gemaakt?

Spellbender is eigenlijk een prototype, we testen onze ideeën direct in het spel.

Spelbenders is misschien in een prototype achtige fase maar het dient ook als de basis voor een product. Hoe vaak worden er echte prototypes gemaakt?

In de beginfase van Spellbenders zijn er een aantal prototypes gemaakt door een andere spelontwerper. meneer Kamp werkt iets meer dan een jaar aan Spellbenders en heeft hiervoor één keer een prototype gemaakt. Er wordt geen prototype gemaakt als het makkelijk in het spel zelf getest kan worden.

Hypothese:
Een of twee keer per jaar.

Wie maakt het prototype?

meneer Kamp, soms in samenwerking met een programmeur.

Hypothese:
meneer Kamp, de enige full-time spelontwerper.

Hoe worden prototypes van Codeglue ontwikkeld?

Codeglue maakt eerst een overzicht van de ontwerpproblemen die moeten worden opgelost.

meneer Kamp omschrijft zijn ontwerp in google docs met draw.io, een plugin voor google docs om grafieken te maken. Als hij een structuur voor het prototype heeft gemaakt gaat hij verder in Photoshop om concrete kaarten te spelonderdelen te maken.

In Photoshop leg ik alle kaarten naast elkaar en kijk welke interacties er mogelijk zijn.

Maar als het veel kaarten zijn hoe hou je het overzicht?

Je kan inderdaad niet zoeken of makkelijk concrete data krijgen.

Daarna print en snijdt hij de onderdelen om het te kunnen testen. Eerst zelfstandig daarna met collega’s. Tijdens en na het testen maakt hij aanpassingen met pen.

Hypothese:
De ontwerper noteert zijn ideeën in een text document, hij bedenkt een concept voornamelijk vanuit intuïtie en creëert daarna de benodigde componenten om het te testen.

De ontwerper gebruikt alleen Notepad en Photoshop om het prototyping proces te ondersteunen. Hij beschrijft zijn concept in Notepad. Daarna werkt hij grotendeels in Photoshop.

De ontwerper print en snijdt de kaarten. Hij simuleert het spelen van het spel door de rollen van alle spelers aan te nemen. Hij krijgt hierdoor inzicht in de interactie van de belangrijkste spelregels en -elementen, en maakt de laatste grote aanpassingen voordat hij andere spelers het laat spelen.

Hoeveel tijd wordt er gepland om een prototype te maken en testen? (waar hangt dit van af?)

De ontwerper geeft aan wanneer het nodig is, en hoe lang hij hier ongeveer voor nodig heeft. Voor een fysiek prototype meestal een week. Voor een digitaal prototype hangt het er voornamelijk van af of er programmeurs beschikbaar zijn.

Hypothese:
één tot twee weken. meneer Kamp moet ook werken aan andere projecten.

Voor Spellbender, hoeveel tijd is ere voor prototyping?

Ongeveer twee weken op een jaar.

Hoeveel tijd wordt er besteed per onderdeel (bedenken, maken, testen)?

De verhouding tussen het bedenken en uitwerken van een concept is ongeveer 75/25. Het testen van een ontwerp neemt meer tijd in beslag omdat het afhankelijk is van de beschikbaarheid van andere werknemers.

Hypothese:
één dag conceptontwikkeling, één tot twee dagen maken en printen. één week testen.

Zou Codeglue meer / minder tijd willen besteden aan prototyping?

We zouden wel meer tijd hiervoor willen hebben. Er is afwisselend meer en minder tijd beschikbaar om aan eigen projecten te werken. Voor externe opdrachten is er vaak geen prototype nodig.

We streven naar een 50/50 tussen interne en externe projecten, maar dat is financieel nog niet haalbaar.

Hypothese:
Ja maar er is maar beperkte tijd beschikbaar. meneer Kamp krijgt wel vrijheid om zijn tijd zelf in te delen.

Hoe worden prototypes van Codeglue getest?

De ontwerper test eerst zelf de globale regels van het spel waarna het wordt blootgesteld aan collega's. Hij noteert zijn observaties worden en maakt aanpassingen van waardes en specifieke regels na elke playtest om grip te krijgen op de impact van individuele spelelementen.

Hypothese:
De ontwerper simuleert het spelen van het spel door de rollen van alle spelers aan te nemen. Daarna observeert de ontwerper hoe verschillende collega's het spel spelen.

Wie test het prototype?

meneer Kamp.

Hypothese:
meneer Kamp, de enige full-time spelontwerper.

Wat zou Codeglue graag willen behalen met het testen van hun prototypes?

Codeglue wil hun ontwerpkeuzes kunnen onderbouwen. Hiervoor worden een aantal vragen vastgesteld in een design document. Ze willen data vergaren om antwoord te kunnen geven op deze vragen.

Hypothese:
Kennis krijgen van de balans tussen spelelementen van hun digitale game. Een model maken van hoe de digitale game kan worden verbeterd. Ondervinden of spelelementen leuk zijn.

Het doel van het prototype wordt in een tekstdocument beschreven.

Wat wordt er behaald met het testen van hun prototype?

Alle observaties van het playtesten en wat dit betekend voor het project worden beschreven in een design document.

Hypothese:
Observeringen over welke strategieën ondergeschikt of dominant zijn, wat er voor nodig is om een betere balans te creëren.

De resultaten van het prototype worden in een tekstdocument toegelicht.

Wat gebeurd er met de verkregen kennis? Hoe worden de resultaten toegepast?

Het bepaald welke onderdelen de programmeurs moeten implementeren om de nieuwe elementen van het ontwerp te kunnen verwerkelijken.

Hypothese:
Om nieuwe features / aanpassingen te plannen voor de digitale games.
Het dient meer als een richtlijn voor systemen die nodig zijn of strategieën die ondersteund zouden moeten worden, dan concrete aanpassingen van parameters van het spel.

Worden er iteraties gemaakt van een prototype?

Ja, de ontwerper maakt kleine aanpassingen na elke playtest. Het laatste prototype is ongeveer 30 keer gespeeld waarna er vaak aanpassingen zijn gemaakt.

Hypothese:
Alleen direct op de kaarten na elke playtest. Zo'n 20 kleine aanpassingen.

Wanneer tijdens het process worden de meeste aanassingen gemaakt?

Op het moment dat er getest wordt met andere spelers. Ze doen dingen die meneer Kamp niet had verwacht.

Hypothese:
Na de eerste keer individueel playtesten voordat het in handen komt van de spelers.

Hoeveel iteraties worden er gemaakt? (binnen welke tijdsperiode)

Het laatste prototype is ongeveer 30 keer gespeeld waarna er vaak aanpassingen zijn gemaakt.

Hypothese:
Zo'n 15 kleine aanpassingen.

Welke aanpassingen worden gemaakt per iteratie?

De kern blijft hetzelfde. Maar individuele elementen worden met pen aangepast.

Hoe worden grote veranderingen geïmplementeerd?

Je houd je ook een beetje in omdat je alle kaarten moet aanpassen, en het kost veel tijd om alles weer terug koppelen en opnieuw uit te printen. meneer Kamp doet geen grote veranderingen als dat hij misschien zou willen.

Soms moet het wel, maar vaak is het niet waard om het terug te koppelen.

Hypothese:
Alleen kleine aanpassingen omdat het lang duurt het hele model aan te passen.

Waren er tijdens het testen van het prototype voor Spellbenders momenten waarop je graag een hele andere richting in had willen slaan?

Nee, ik kreeg wel nieuwe ideeën maar die waren niet relevant voor het huidige prototype.

Ben je gewoon een hele goede ontwerper en voldoet het eerste prototype gelijk om de juiste data te verkrijgen?

Er was best wat onderzoek gedaan, dus er was een redelijke aanname dat het ontwerp zou werken.

Je noemt onderzoek, waar was die aanname op gebaseerd?

Op bestaande systemen van vergelijkbare spellen.

Wat als er geen voorbeelden zijn van andere spellen?

Dan moeten we er meer tijd voor inplannen om meer iteraties te kunnen maken.

Waar denk je dat de knelpunten liggen in het ontwikkelen van prototypen?

Het terugkoppelen kost veel tijd en wordt daardoor niet altijd gedaan. Daarnaast ben je afhankelijk van de beschikbaarheid van collega's, voor het ontwikkelen van een digitaal prototype of het testen van een fysiek prototype.

De stap van een idee naar een speelbaar prototype kost veel tijd, daarom wordt er niet vaak gebruik gemaakt van prototyping. Zeker bij Spellbenders. Er is al een speelbare versie, dus is het verleidelijk om direct een rudimentaire versie van een idee te implementeren.

Het is een grote translatie tussen een idee en een spel. Je maakt een prototype omdat je de elementen in beweging wilt zien.

Terugkoppeling is moeizaam. Alle bestanden moeten worden aangepast en zijn niet gekoppeld. Als er één omgeving zou zijn waar het ontwerp in leeft is dat niet nodig.

Dag In het Leven Onderzoek Log

meneer Kamp heeft 3 - 4 uur voorafgaand aan de opdracht research gedaan naar andere games (gameplay video's en spelregels lezen) om inspiratie op te doen voor mechanics.

doelstellen, time: time: 5min

- meneer Kamp stelt een doel, wat maakt spellbenders uniek?
- Welke van deze aspecten moeten zichtbaar worden in het spel?

brainstormen, time: time: 53 min

- meneer Kamp omschrijft in een paar zinnen een idee
- Hij genereert zo veel mogelijk ideeën, hij blijft niet langer dan 5 minuten bij een idee hangen.

ontwerpen aanscherpen en vastleggen, time: 113 min

- Hij pakt de ideeën die bij zijn doel passen en werkt ze verder uit
- Hij identificeert mogelijke problemen
- Hij kiest een idee uit om uit te werken.

opzetten, time: 87 min

- meneer Kamp maakt een excel sheet met de spel variabelen (hoeveelheid kaarten totaal, hoeveel kaarten per speler hand, hoeveelheid spelers, percentages van elk type kaart)
- Hij schrijft functies om variabelen te koppelen zodat hij de veranderingen voor het spel kan zien als hij schuift met de waardes.
- Hij moet regelmatig kansberekening theorie opzoeken, en uitzoeken hoe hij dat naar excel vertaald.
- Hij wil de samenstelling van een hand kunnen zien. Hij maakt een tabel om te simuleren hoe een gemiddelde hand er uit ziet met X kaarten.

spel variabelen uitproberen, time: 58 min

- meneer Kamp wilt vaststellen hoeveel hij van elk type kaart er moet zijn, en hoeveel kaarten iedere spelere krijgt. Zodat hij met het maken van de kaarten een houvast heeft.
- meneer Kamp wil dat als spelers een hand kaarten pakken dat ze genoeg opties hebben. Zijn concept heeft drie type kaarten, een grondstofkaart, een eenmalige effect dat grondstoffen kost om te spelen, en zeldzame kaarten met een blijvend effect.
- Hij begint met een hand van zes kaarten, hij stelt vast wat een optimale hand verdeling is (de meeste grondstof kaarten, twee effectkaarten zodat de speler keuze heeft, en niet meer dan één blijvend effect.)

- Hij veranderd de verhouding van kaarten, en simuleert hoe de hand er dan uit komt te zien.
- Hlj kiest voor drie type grondstoffen, die overeenkomen met de elementen van spelbenders.
- meneer Kamp simulated i.p.v. berekenen wat een gemiddelde hand is. Hij zou het wel willen berekenen maar hij vind het lastig en het hoeft ook niet super precies. Hij wil een basis hebben om de waardes van de kaarten te kunnen kiezen en hoeveel er in moeten zitten.

documenteren van welke verhouding er moet zijn, en waarom, time: 12 min

- meneer Kamp schrijft zijn beslissingen op gebaseerd op de data van zijn kansberekening.

bedenken van kaart effecten, time: 38 min

- meneer Kamp noteert effecten om het doel te kunnen bereiken, levenspunten van de tegenstanders moeten naar 0. Daarnaast baseert hij de effecten op spreuken in spelbenders.
- Hij sorteert de lijst op de basis effecten en de complexere effecten. Hij beschrijft het doel van elke type kaart (grondstof, effect, sterk effect)

maken van de kaarten, time: 147 min

- Hij zoekt een template op internet om zijn kaarten op te baseren
- Hij experimenteert met de layout van een kaart en houd daarbij rekening met de uiterste waarde van zijn ontwerp.
- Hij maakt een spell kaart die hij kan kopiëren en plakken
- Hij vult alle kaarten in
- Hij herhaald dit proces voor elk type kaart
- Zijn groepen en layers hebben geen naam en onoverzichtelijk

Card Game Toolkit Usability Test

Contents

Introduction

The Card Game Toolkit (CGT) is a software tool that allows you to build and play a card game by manipulating a set of components with a simple scripting language. The CGT is designed to create card games that can be played using traditional playing cards.

Games are created by combining four components; **players**, **piles**, **cards** and **actions**.

A player represents a player in the game, a pile is a meaningful location that a card can occupy, and a card is a collection of attributes that mean something within the context of the game. Actions describe the rules of the game.

All components can be controlled by actions. There might be an action that shuffles a pile and deals cards, one that discards a card from a player's hand and another that moves the active player token to the next player. Actions do not add or remove components, they will only change their configuration.

The UI of the card game toolkit (CGT) is split into two segments. The *table* which occupies most of the screen space, and a game log at the bottom of the screen.

The CGT has two modes in which it operates, *edit* and *play*. When you open the program it will always be in *edit* mode. Use *edit* to set up the game, then switch to *play* to test your design. While in *play* mode, actions can be executed, changing the state of the game.

Play does not prevent you from editing your design but changes made in this mode will not be saved when returning to *edit* mode. This allows you to quickly return to the initial game state.

The game can quickly become a complex web of interactions, it is therefore recommended to switch often between the *edit* and *play* to test your design and save your progress before making substantial alterations.

Scroll Wheel (Click and Drag) allows you to navigate the table.

Right Mouse (Click) on the table reveals a context menu. Use it to create components, save or load projects or clear the table.

Spacebar toggles between the *edit* and *play* mode (changes made in *play* mode will not be saved).

1.0 Assignment

You will be creating a version of UNO using the Card Game Toolkit. This page includes an overview of UNO's components and its rules.

1.1 UNO

UNO is a card game for 2 - 10 players (create UNO for three players).

1.1.1 Components

20 blue cards (two copies of 0 to 9)
20 green cards (two copies of 0 to 9)
20 red cards (two copies of 0 to 9)
20 yellow cards (two copies of 0 to 9)
8 *skip* cards (two of each colour)
8 *reverse* cards (two of each colour)
8 *draw2* cards (two of each colour)

1.1.2 Goal

Each player begins with a full hand of seven cards, be the first player without any cards in your hand to win the game.

1.1.3 Game Setup

All cards are shuffled into a draw pile that is placed face down on the table. Every player draws 7 cards, then the top card of the draw pile is placed face up onto a discard pile. A random player may begin their turn.

1.1.4 Player Turn

During a player's turn, they may play a card onto the discard pile and/or draw a card from the draw pile. You can only play a card onto the discard pile if it matches the top card; they must have the same number and/or colour.
You may end your turn after you play a card or gain a card from the draw pile. Then the next player takes their turn. Continue clockwise, taking turns until one player is out of cards (the order can be reversed with the *reverse* card).

1.1.5 Special Cards

There are three special cards. Cards that match an ability can be played on top of each other, similar to the regular cards (e.g. you may play a *skip* onto another *skip* card).

skip	The next player skips their turn.
reverse	The player turn order is reversed.
draw2	The next player must draw two cards and skip their turn.

1.1.6 Game over

The first player to play all their cards wins the game. (The CGT has no game over state. Whenever a player has no cards left, they should be able to click on the “win” action that moves all cards back to the draw pile.)

1.2 Let’s Get Started! Right?

Well.. not just yet. Please read the chapter on each component, and at least skim through the rest of the document before you begin building your game. The next chapter will ask you to look at some examples provided with the tool, you can load projects from the *table’s* context menu.

Before you begin creating UNO, read the final page of this document. It includes useful TIPS!

Good Luck!

2.0 Components

Whenever you add a new component to the *table*, immediately give them an appropriate name. This will not only help you organize your project, it will also make it easier for you to reference them in one of the scripts. Using descriptive names right away will save you a lot of time in the future.

The component's name is used as its identifier and must be unique. Piles, however, can have the same name as long as they are connected to a different player.

The CGT comes with a few example projects to help you learn the core concepts. **You can open the example projects from the *table's* context menu** (right-click on any empty space). After loading a project, switch to *play* mode by pressing the spacebar. The first example gives a quick overview of all components, it should help you better understand the other examples.

Alright, let's **checkout** *example_0* (after loading use spacebar to switch to play mode! you can use the scroll wheel to grab and drag the *table*.)

Left Mouse (Click) on a component to select it.

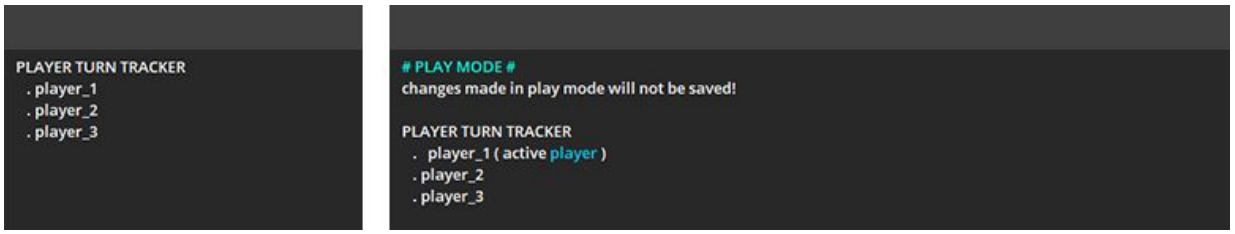
Left Mouse (Click and Drag) to move the component to another location.

Right Mouse (Click) on a component to open a context menu.

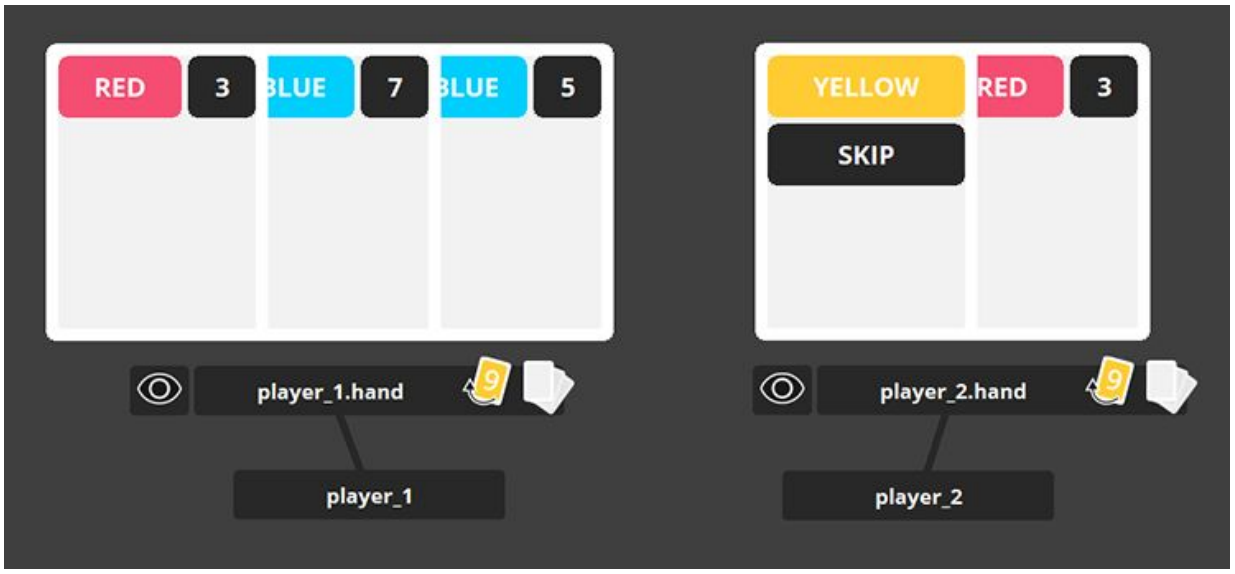
Delete deleted the selected component (it will not ask for confirmation! be careful.)

2.1 Players

Use a player component to represent a player in the game. Player components appear in the player turn tracker at the bottom of the screen. The turn tracker shows the active player (you need to set the active player using a script, see 3.2.1 Player Consequences) and the order in which players take their turns.



Right Mouse Click on the player component will open their context menu. You can rename the player, delete them or connect them to a pile. The option **Add Pile** allows you to click anywhere on the *table* to create a new pile, or select an existing pile to attach it to the player.



Piles can have the same name as long as they are connected to a different player. Notice that their identifiers have changed to “player_1.hand” and “player_2.hand” respectively. The dot indicates that the components are connected. Now, why is this useful?

There can only be one active player at a time. The CGT script has a reserved word for the active player, simply **player**. If you would like to do something with the active player’s “hand”. You can use “player.hand”, “player” will be replaced by the active player’s name.

Alright, let’s **checkout** *example_01* (after loading use spacebar to switch to play mode! you can use the scroll wheel to grab and drag the *table*.)

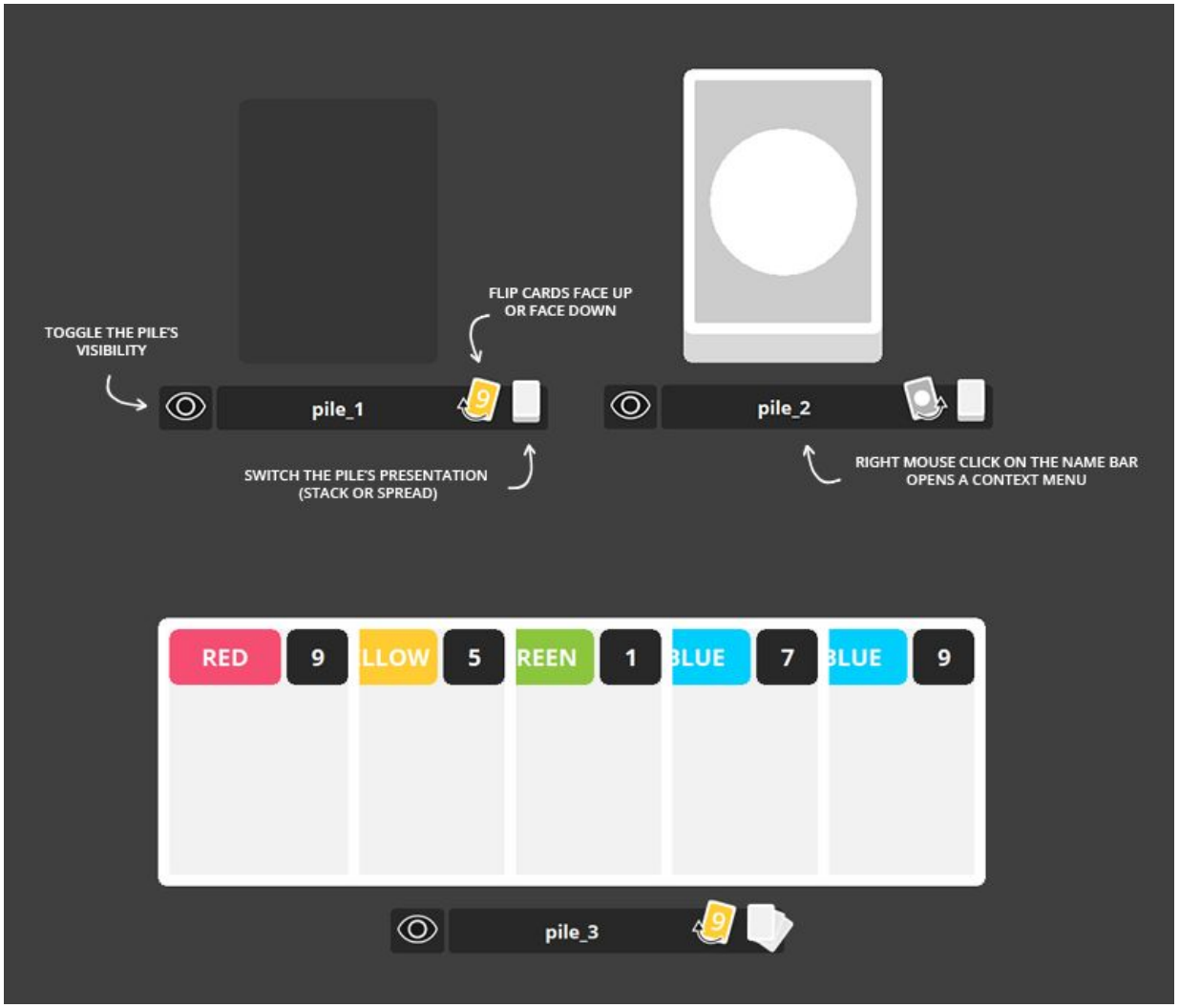
2.2 Piles

Use a pile component to represent a collection of cards (e.g. a deck of cards or a player's hand). You should create a pile for each meaningful location a card can exist in. Its context menu includes a few options for adding cards.

A pile, and all of its cards, can be face up or face down. Cards that are facedown cannot be evaluated by a *condition* (see actions), because their information is hidden.

A pile can be presented as a stack or a spread of cards. Only the top card of a stack can be selected, whereas a spread allows any card to be selected.

You can change these by using one of the designated buttons.



2.3 Cards

Use a card component to represent a card in the game. A card is a collection of attributes that mean something within the context of the game. CGT script has syntax to compare these attributes, allowing you to establish and enforce the game's rules.

The cards you will be working with can have a **number** (0-9), **colour** (red, yellow, green or blue) **skip**, **reverse** or **draw2** (the last three are attributes without a value).

Cards can only exist in a pile. You can reference a section of a pile (e.g. the top 6 cards, all cards or just the bottom one) and evaluate their attributes. Cards that are face down can not be evaluated, because you cannot see them.

Checkout *example_02*.

You can also reference the selected card, by using the reserved word "card". In the next example, notice how the colour of "card" is yellow when used in this context.

Checkout *example_03*

Then take a look at the following two examples.

Checkout *example_04* and *example_05*

The last two examples are quite similar, but there is one important distinction. In *example_04* you can select the red card that is not in the players hand. Without specifying that the selected card needs to be part of a specific pile, any face-up card can be selected. *example_05* makes sure the player can only select a card from their hand by checking if "player_1.hand has card".

2.4 Actions

As you have seen in the examples, you can use actions to create the flow of the game. Actions determine what can happen at any moment during the game.

Actions are the most complex components of the Card Game Toolkit, but they are also the most powerful. Actions use a scripting language (CGT script) to manipulate any component of the game.

An action can have **consequences** attached to it. A *consequence* is a set of instructions that describe what happens when the action is executed (e.g. shuffle a pile, move a card to another location or change the active player). The syntax for these statements can be found in the next chapter about CGT script.

In most games, there are activities that are only performed once (e.g. during the setup of the game) or they might only be available during specific phases.

One way to control the state of the game is by *enabling or disabling* actions. This can be done manually to set-up the initial game state, or during *play*, as a result of performing a specific action.
Only actions that are *enabled* can be executed.

Checkout example_06

Another way to control when actions can be performed is by using **conditions**. A *condition* is a script with one or more expressions that can be true or false. If all expressions evaluate to true, the action can be performed. An action can be executed if at least one condition is met.

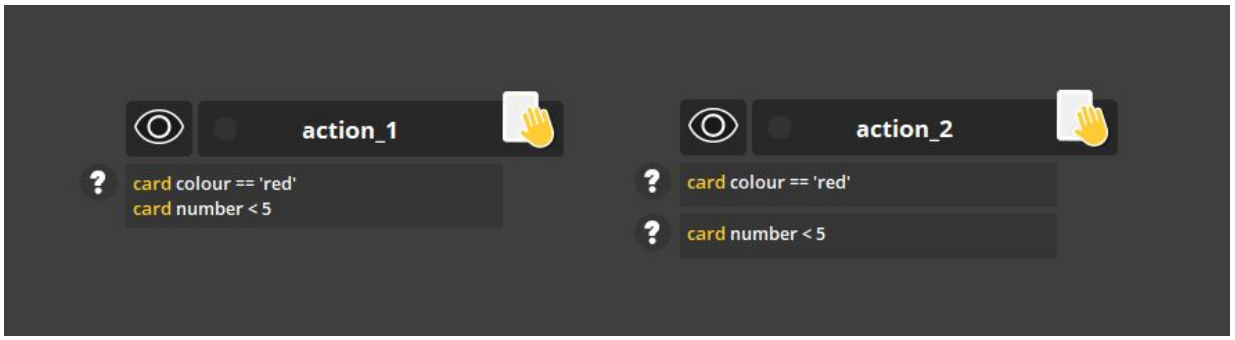
Checkout example_07

During *play,* only actions that are enabled will have a button to execute them. Actions are **disabled** by default. You can manually enable them using the button in front of their name. Use the “enable” and “disable” instructions to move through the phases of the game, and the conditions to check the variable parts of the rules (e.g. after “deal” enable “play_card” but have a condition on “play_card” to specify which cards can be used for that action.)

You can add conditions and consequences from the *action* context menu, and delete them using their own context menus.

An action can have multiple conditions, one condition or none at all. The order of the conditions does not matter, as long as one of its conditions is true, the action can be executed. If an action requires multiple expressions to be true, they need to be contained in one condition.

In the example below “action_1” can only be used if the selected card’s colour is red and its number is less than five. However, “action_2” can be used if either the selected card’s colour is red **or** its number is less than five.



Checkout *example_8*

You should use enable/disable to control the phases of the game. e.g. an action “shuffle” enables another action “deal” that gives each player a hand of cards and chooses a player to start their turn. “deal” enables the actions a player can use during their turn “play”, “draw”, “destroy_card” or “end_turn”.

You should use conditions to determine if these actions can actually be used. “play” might require a card with a number higher than 5, “draw” can only happen if the player has less than 3 cards in their hand or “end_turn” can only happen once “play” or “destroy_card” has been performed and disabled.

The last example is a simple one player game. A deck of cards is shuffled, you are dealt 5 cards. You can only play cards onto the discard pile if they have a higher number, unless the last card you have played is a red card, then the number must be lower.

Checkout example_09

3.0 CGT Script

CGT script is a language designed for describing *conditions* and *consequences* that determine when an action can be performed and what happens when it is executed.

Each expression or instruction in a CGT script needs to be on a separate line. Lines that start with a hashtag are comments and are ignored by the program.

```
# incorrect
shuffle draw_pile move draw_pile top 5 cards above another_pile

# correct
shuffle draw_pile
move draw_pile top 5 cards above another_pile
```

CGT script has a few reserved words that have special meaning and cannot be used as component names, these words will be highlighted in another colour. There are two keywords that require a bit more attention.

The keywords **player** and **card.** *player* references the active player and *card* references the currently selected card. A script containing one of these words can only be performed if there is an active player or a card selected.

```
# you can reference the active player
player

# do something with a pile connected to the active player
player.hand
player.deck

# move the top card from the “draw_pile” to the active player’s “hand” pile
move draw_pile top card player.hand

# you can reference the selected card
card

# check if a card has an attribute
card has colour
card has number

# compare an attribute of the selected card
card colour == ‘red’
card number == 10

# move the selected card to the “trash_pile”
move card above trash_pile
```

3.1 Conditions

A condition is a script that evaluates to true or false (e.g. check if a card has a specific attribute, compare one or more cards or check if a pile is empty).

3.1.1 Comparison Operators

In a condition you can use comparison operators (==, !=. <, <=, >, >=) to evaluate the relationship between two things. The following expressions are all true.

```
# true if A is equal B ( == )
10 == 10
‘red’ == ‘red’

# true if A is not equal B ( != )
4 != 8
‘red’ != ‘blue’

# true if A is greater than B ( > )
6 > 3

# true if A is greater than or equal to B ( >= )
20 >= 13

#  true if A is smaller than B ( < )
6 < 60

#  true if A is smaller than or equal to B  ( <= )
1 <= 1
```

3.1.2 has (not)

Use **has** to check if a card has a specific attribute or if a pile contains a card that does.

```
# true if card has an attribute named “rainbow”
card has rainbow

# true if card has an attribute named “magic_shield”
card has magic_shield

# true if the supply_pile has at least one card with an attribute named “magic_shield”
supply_pile has magic_shield

# invert the result by adding ‘not’
card has not rainbow
```

```
# true if the active player does not have a draw2 card in their hand
player.hand has not draw2
```

3.1.3 is (not)

Some components have properties that can be true or false (e.g. an action can be enabled / disabled or a pile can be face up or face down). Use **is** to check if these properties are true.

```
# true if the “play” action is enabled
play is enabled

# similar to ‘has’ you can use ‘not’
play is not enabled

# or
play is disabled

# a few more examples
buy_pile is face_up
draw_pile is empty
player.hand is not empty
```

3.1.4 Comparing Piles

Piles have a single attribute named *count* that represents the number of cards they have.

```
# draw_pile cannot be empty
draw_pile count != 0

# draw_pile must have more than 10 cards
draw_pile count > 10
```

```
# check if a pile is face up
pile_name is empty
pile_name is not empty

#e.g. the active player's "hand" pile must be be empty
player.hand is empty

# check if a pile is face up
pile_name is face_up
pile_name is not face_up
pile_name is face_down
pile_name is not face_down

# check if a pile is a stack or spread
pile_name is stack
pile_name is not stack
pile_name is spread
pile_name is not spread
```

```
# check if the pile has a card with a specific attribute
pile_name has attribute_name

# e.g.
supply_pile has magic_shield

# at least one card must have skip
player.hand has skip

# no card can have draw2
player.hand has not draw2

# the selected card is the active player's "hand" pile
player.hand has card
```

3.1.5 Comparing Cards

You can compare the attributes of the selected card with fixed values or that of other cards. If a card does not have an attribute with the specified name, the expression will always be false. **A card must be face up to inspect their attributes.**

The words **top**, **bottom**, **all** and **card(s)** are used to describe a selection of cards from a pile. Notice that the word "card" appears in blue when it is used in this context.

```
# do something with the top card of a pile (both statements do the same)
pile_name top card
pile_name top 1 cards

# or with the bottom card
pile_name bottom card
pile_name bottom 1 cards

# do something with the top 3 cards, bottom 6 cards or all cards of a pile
pile_name top 3 cards
pile_name bottom 6 cards
pile_name all cards
```

```
# the selected card must have a 'colour' attribute with the value 'red'
card colour == 'red'

# the selected card must have a 'number' attribute with a value greater than 5
card number > 5

# the selected card's number must be larger than the number on the top card of the discard
pile
# both cards must have a "number" attribute otherwise the expression will be false
card number > discard_pile top card number

# the selected card's number must be smaller than all the numbers on cards in the
discard_pile
# all cards must have a "number" attribute otherwise the expression will be false
card number < discard_pile all cards number
```

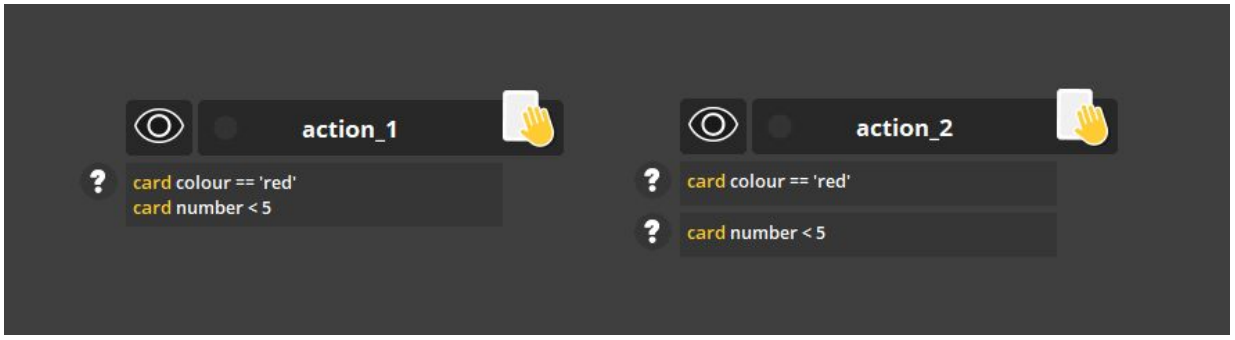
```
# check if the selected card has a specific attribute
card has attribute_name

# e.g.
card has skip
discard_pile top card has not reverse
```

3.1.6 Comparing Actions

```
# check if an action is enabled/disabled
pile_name is enabled
pile_name is not enabled
pile_name is disabled
pile_name is not disabled
```

3.1.7 Multiple Conditions

All expressions in a single condition must be true for the condition to be true. In the example below “action_1” can only be used if the selected card’s colour is red and its number is less than five. However, “action_2” can be used if either the selected card’s colour is red **or** its number is less than five.



3.2 Consequences

A consequence is a script that manipulates the game state (e.g. it can move cards around, set the current player or enable/disable actions).

3.2.1 Player Consequences

Set the active player and manage the player turn order.

```
# set a specific player as the active player
set_player player_name

# set a random player as the active player
random_player

# makes the next player in the player turn tracker the active player
# or it sets the first player in the turn tracker as the active player
next_player

# reorders the player turn tracker
reverse_player_order
```

3.2.2 Pile Consequences

Shuffle piles and change their presentation.

```
# shuffle one or more piles (separate their names with whitespace)
shuffle pile_name
shuffle pile_1  pile_2  pile_3

# change presentation of one or more piles
stack pile_name
spread pile_1  pile_2  pile_3

# flip one or more piles face up or face down
face_up pile_name
face_down pile_1  pile_2  pile_3

# minimize piles
conceal pile_name
reveal pile_1  pile_2  pile_3
```

3.2.3 Card Consequences

Move cards from one pile to another.

```
# move   (pile name)   (a selection of cards)   (above or under)   (destination pile name)

# move the top card of the discard_pile on top of the discard_pile.
move draw_pile top card above discard_pile

# move the bottom card of the draw_pile on top of the discard_pile.
move draw_pile bottom card above discard_pile

# e.g
move draw_pile top 2 cards under supply_pile
move draw_pile bottom 5 cards above player.hand
move discard_pile all cards under draw_pile


move   card   (above or under)   (destination pile name)

# move the selected card on top of or underneath a pile.
move card above destination_pile_name
move card under destination_pile_name
```

```
# move the selected card on top of a pile named “trash_pile”
move card above trash_pile

# or used together, move the selected card on top of the active players ‘hand’ pile.
move card above player.hand

# move the selected card above a pile named ’hand’ that is attached to a player named
‘player_1’
move card above player_1.hand
```

3.2.4 Action Consequences

```
# shuffle one or more piles (separate their names with whitespace)
enable action_name
disable action_1 action_2 action_3

# minimize
conceal action_name
reveal action_name
```

4.0 Setting Expectations

The current version of the CGT does not allow you to create or edit cards. Instead, it provides a set of cards found in the card game UNO. A card can have a **number** (0-9), **colour** (red, yellow, green or blue), **skip**, **reverse** or **draw2** attribute (the last three do not have a value, they function as a label).

There is no way to define a game-over state. I suggest you create an action named “win” and define its conditions so a player can only choose the “win” action when the rules would allow them to end the game.

CGT uses a scripting language and is not compiled before execution. A script will notify you when they do not understand your instructions. But some errors will only be noticed during execution. I haven’t had any errors that terminate the program, but you never know with software in development. I suggest you save your progress often.

5.0 Tips

TIP. Begin by creating a version of UNO without the special cards.

TIP. Create separate “play” actions for each type of special card.

TIP. Create a clear starting point by *disabling* all but one action. Use this action to set up the random elements of the game (e.g. shuffle a pile and deal cards). Make sure this action disables itself and enables other actions so the game can continue.

TIP. I am well aware the *table* can get crowded rather quickly. This is definitely something that will be addressed in a future version, for now, utilize the eye icon to minimize clutter.

Card Game Toolkit 2.0 Usability Test

Introductie

De Card Game Toolkit (CGT) is een samenstelling van tools voor het ontwerpen van kaartspellen.

Een onderdeel van de CGT is de Card Editor, een tool voor het gemakkelijk ontwerpen en aanpassen van kaarten.

Een kaart bestaat uit twee onderdelen de kaart *informatie* en de kaart *presentatie*. De informatie is een collectie van eigenschappen die een betekenis hebben binnen de context van het spel. De presentatie is de visuele weergave van deze informatie.

De ontwerper kan een kaart maken, een aantal eigenschappen toevoegen en deze eigenschappen gebruiken als onderdeel van grafische elementen. Als de informatie van een kaart verandert zijn deze aanpassing direct zichtbaar op de kaart. Zo kan de ontwerper gemakkelijk de waardes van een kaart aanpassen of experimenteren met de vormgeving zonder dat de eigenschappen van de kaart verloren gaan.

Daarnaast kan de ontwerper een template maken van een kaart. Een template bestaat net als een kaart uit eigenschappen en grafische elementen. Als een kaart gebruik maakt van een template worden alle aanpassingen aan de kaart doorgevoerd op de template, en alle kaarten die gebruik maken van de template. De waardes van eigenschappen zijn uniek per kaart, en worden dus niet overgenomen door een template of andere kaarten.

1.0 CGT Card Editor Overview

Bekijk tijdens of na het lezen de CGT screenshots als een korte doorloop van de interface.

1.1 Project Menu

New Project: Maak een nieuw project.
Open Project: Open een bestaand project (selecteer de project folder).
Save Project: Sla het project op.

1.2 Card List

New: Maak een nieuwe kaart.

De naam van de kaart kan in de inspector worden aangepast.

1.3 Inspector

De inspector laat de eigenschappen van de geselecteerde kaart zien. De gebruiker kan de naam van de kaart aanpassen, een template selecteren, en de informatie en presentatie aanpassen.

1.3.1 Card Template

New Template: Maakt een nieuwe template van de geselecteerde kaart.

Als een nieuw template gemaakt wordt van de huidige kaart neemt het template de informatie en presentatie van de kaart over, **behalve de waarde van de informatie**. De waarde zijn voor elke kaart uniek.

Als een bestaande template aan een kaart wordt toegevoegd, **behoud de kaart alleen informatie die ook op de template staat!** De kaart neemt de presentatie over van de template.

bijvoorbeeld, een template “Treasure” heeft de eigenschappen “Goud” en “Omgeving”. Als de gebruiker de “Treasure” template selecteert voor een kaart met de eigenschappen “Goud = 7” en “Type = Ring” behoudt de kaart de “Goud” eigenschap, krijgt het een “Omgeving” eigenschap zonder waarde en verliest het de eigenschap “Type”.

Als de kaart geen gebruik meer maakt van een template neemt hij de informatie en presentatie van de het template over.

1.3.2 Information

New: Voeg een nieuwe eigenschap toe, aan de kaart en aan de template als de kaart daar gebruik van maakt.

Eigenschappen hebben een naam en een waarde. de waarde van een eigenschap kan een tekst (“Text Attribute”) of een nummer (“Number Attribute”) zijn.

Text Attribute: De waarde is een stuk text. Gebruik het voor namen, types of beschrijvingen van kaarten.

Number Attribute: De waarde is een nummer. Gebruik het voor numerieke eigenschappen van een kaart, bijvoorbeeld “Kost”, “Schade” of “Punten”.

1.3.3 Presentation

New: Voeg een nieuw grafisch element toe, aan de kaart en aan de template als de kaart daar gebruik van maakt.

Een grafisch element kan een tekst (“Text Element”) zijn of een afbeelding (“Image Element”).

Text Element: Een text kan een *custom text* zijn of de waarde van een attribute over nemen. Een custom text kan ook waardes van attributen bevatten, plaats de naam van de eigenschap tussen blokhaken “[attribute_name]” en het wordt vervangen door de juiste waarde.

Tekst tussen “[ ]” wordt vervangen door de waarde van de genoemde eigenschap. Als de eigenschap niet bestaat wordt de tag verwijderd. de tag [cardname] kan gebruikt worden om te verwijzen naar de naam van de kaart.

Image Element: Kies een afbeelding en een schaal. Het is nog niet mogelijk om zelf afbeeldingen toe te voegen.

2.0 Opdrachten

2.1 Think Aloud Protocol

Denk hardop tijdens het maken van de opdrachten. Beschrijf voordat je iets gaat doen wat je wilt bereiken en benoem de stappen die je onderneemt. Als je ergens tegenaan loopt probeer het zelf op te lossen. De observeerder kan tijdens het proces om opheldering vragen over je denk process.

2.2 Better Save Than Sorry

De CGT is software in ontwikkeling en nog niet uitgebreid getest. Sla tussendoor je werk op voordat het verloren kan gaan, dit kan via het project menu of met de shortcut (Ctrl + S).

2.3 Handige Shortcuts

New Project: Ctrl + N
Open Project: Ctrl + O
Save Project: Ctrl + S

Undo: Ctrl + Z
Redo: Ctrl + Shift + Z

2.1 Opdracht 1

Het doel van de eerste opdracht is om de features van de Card Editor te verkennen. De kaart hieronder komt uit het kaartspel, Magic the Gathering. De kaart heeft een naam (1), cost (2), creature type (3) en een power (4) en toughness value (5).

2.1.1 Open een nieuw project en maak deze kaart in de tool.

2.1.2 Maak een template van deze kaart en gebruik deze om nog 5 creature kaarten te maken.

2.1.3 Maak de volgende aanpassing aan de template door een van de kaarten te veranderen.

- verplaats de power / toughness naar links onder
- verander de kleur van de naam van de kaart naar iets opvallends
- voeg een nieuwe eigenschap toe “ability text”, en geef elke kaart een effect. Bijv. “All Golems gain +1/+0.” of “When this card enters play, deal 5 damage.”

2.2 Opdracht 2

2.2.1 Open een nieuw project en maak de kaarten het eerste ontwerp van de spellbenders card game.

2.2.2 Noteer een aantal aanpassingen die je zou willen doen alsof je net een playtest hebt gedaan (max 15 min).

- Een van de aanpassingen moet te maken hebben met de balans van een aantal kaarten. De waardes moeten worden aangepast.
- Een van de aanpassingen moet te maken hebben met de presentatie van de kaarten, een onderdeel van de layout was niet duidelijk voor de spelers.
- Denk na over veranderingen die je bij een ander prototype hebt moeten doen, en beschrijf een vergelijkbare verandering die je bij de spellbenders card game wilt doorvoeren.

2.2.3 Maak de aanpassingen in het eerste ontwerp, zowel in het excel als psd document.

2.2.4 Maak dezelfde aanpassingen in het CGT project.

3.0 Vragenlijst

Ingevuld door meneer Kamp, Game Designer bij Codeglue.

1. Hoeveel kaartspellen heb je (ongeveer) ontworpen?
Ik heb een stuk of 5 Card games ontworpen.

2. Heb je hiervoor software tools gebruikt? zo ja, welke?
Voor al deze card games heb ik Photoshop gebruikt om de kaarten te maken. Daarnaast gebruik ik een combinatie van een tekstprogramma (google Docs) en een spreadsheet programma (Google sheets).

3. Voor welke onderdelen van het ontwerpproces gebruik je deze tools?
Photoshop is voornamelijk om de kaarten visueel vorm te geven zodat ik ze kan printen en kan playtesten. Google Doc is om mijn initiële gedachten uit te typen, regels op papier te zetten etc. In Google sheets maak ik een overzicht van alle data en doe ik wat initiële balancing op basis van die overzichten.

4. Heb je andere tools overwogen? waarom kies je voor deze tools?
Prototypes maken gaat voor mij heel erg om het idee uit mijn hoofd op papier te krijgen en te gaan testen. Dat wil ik zo snel mogelijk doen, op een manier waar ik vertrouwd mee ben. Dit zorgt ervoor dat ik kies voor tools waar ik bekend mee ben.

5. Welke aspecten van deze tools maken ze geschikt voor het ontwerpen van kaartspellen?
Photoshop geeft een extreem niveau van controle over de visuals aan de gebruiker, waardoor de kaarten precies zo vorm gegeven kunnen worden als dat ik wil. Google docs is een prima tekstverwerker waar ik ideëen kan uittypen en eventueel comments kan plaatsen voor extra kanttekeningen. Google sheets is sterk in het organiseren van data en daarop wat basis berekeningen laten lopen zodat je een basis idee kan krijgen van de balance van het spel.

6. Welke aspecten maken ze minder geschikt voor het ontwerpen van kaartspellen?
Photoshop maakt elke kaart uniek, waardoor aanpassingen aan een kaart doen veel extra werk kost. Daarnaast kan het allemaal heel overzichtelijk worden, zeker als je veel unieke kaarten hebt.
Google Doc kan ik niet echt iets verzinnen wat het ongeschikt maakt, het is heel sterk in wat het doet qua tekstverwerking maar de vervolgstappen moeten in een ander programma gebeuren.
Google Sheet kan alleen maar wat basisberekeningen runnen, en volledige simulaties zijn haast niet te doen, waardoor de kaarten fysiek gemaakt moeten worden zodat de game getest kan worden.

7. Hoeveel uur per week gebruik je deze tools?
Mijn week is heel gevarieerd, met veel verschillende projecten in verschillende stadia van ontwikkeling. Al met al denk ik dat ik ongeveer tussen de 10 en 20 uur per week in een van die tools werk.

8. Vind je de interface van de Card Editor duidelijk? zo nee, wat vind je onduidelijk?
Ik vond eigenlijk alles wel duidelijk op wat kleine usability dingetjes na. Sommige elementen schiepen een bepaalde verwachting van wat je ermee kon, wat uiteindelijk niet kon, maar dat was meer verwachting vanuit mij dan onduidelijkheid vanuit de tool.

Het enigste wat een klein beetje onduidelijk was, was de card template aanpassen. Dit gebeurde zodra je een kaart aanpast die gebruik maakt van een template, maar het voelt niet alsof je dan een template aan het aanpassen bent, maar de unieke kaart.

9. Vind je de workflow van de Card Editor gemakkelijk te leren?
Ja, de workflow was heel gemakkelijk op te pakken, de PDF had een korte uitleg en daarna kon ik aan de slag.

10. Vind je de Card Editor prettig om te gebruiken?
Ja, de editor was snel en sterk in het maken van kaarten. De templates besparen enorm veel tijd en laten je heel makkelijk itereren op data. Ik kan me dan ook goed voorstellen dat als je eenmaal iets meer grafische opties erin hebt dat je enorm makkelijk grafisch kan itereren.

12. Welke aspecten van de Card Editor maakt het geschikt voor het ontwerpen van kaartspellen?
Het creëren van templates en het aanmaken en refereren naar variables maken de tool heel geschikt. Je zet een keer de basis op van een kaart en kan dan vervolgens heel veel kaarten daar vanuit creëren, met veel informatie die automatisch ingevuld wordt door de variabelen.

13. Welke aspecten maken de Card Editor minder geschikt voor het ontwerpen van kaartspellen?
Dat zit hem voor mij dan voornamelijk in ontbrekende elementen. Alles wat er wel is is zeer geschikt om kaartspellen mee te bouwen. De grootste missende factor is dat er op het moment van testen geen manier leek om de kaarten te gebruiken in een spelvorm (printen of digitaal) om daadwerkelijk het kaartspel te testen.

14. Zou je de Card Editor gebruiken om de kaarten van je spel te ontwerpen? en bij te houden?
Ja, ik denk dat ik mijn kaarten in dat programma zou bouwen, niet alleen om ze snel visueel op te kunnen zetten, maar ook om er snel op te kunnen itereren.

15. Zo nee, wanneer zou je de Card Editor wel gebruiken?

16. Zijn er features die je mist?
Oh boy. Er zijn heel veel dingen die ik zou willen, maar dat is enthousiasme van mijn kant omdat ik zie hoeveel potentie de tool heeft. De main dingen die ik denk dat de tool nog echt nodig heeft zijn:

- Een export functie om de kaarten te kunnen printen, waarbij je kan aangeven hoeveel van elke kaart je wilt printen. Dit aangeven van hoe vaak elke kaart bestaat is misschien ook wel iets wat je ergens anders wilt kunnen aanpassen omdat je anders alle data van

aantallen op een andere plek moet opslaan tot het moment dat je wilt gaan printen/exporteren.
- letsjes meer organisatie kunnen aanbrengen aan de velden op de kaart, voornamelijk het kunnen renamen zodat het duidelijk is welk veld wat is.
- Het kunnen reorganiseren van de volgorde van de aangemaakte variabelen, zodat tijdens het process dingen die bij elkaar horen bij elkaar kunnen blijven.
- lets meer functionaliteit/controle bij bijvoorbeeld een tekstelement zodat je de afmeting waarbinnen de tekst moet vallen kan instellen. Maar ook grafische elementen misschien niet uniform schalen en de alpha kunnen aanpassen.

17. Heb je nog andere opmerkingen?
Het was erg fijn om het hele process mee te mogen maken en als een van de testpersonen te mogen zijn binnen dat process. Ik denk dat de tool erg veel potentie heeft en nu al erg bruikbaar is. Super gedaan!

Card Game Toolkit 2.0 MOSCOW

Must have

Project Organisatie

Doel: Het organiseren van een project

- De gebruiker kan een nieuw project aanmaken.
- De gebruiker kan een project opslaan / laden.

Card Editor

Doel: Het beschrijven van kaarten.

- De gebruiker kan kaarten toevoegen en verwijderen van de *card library*.
- De gebruiker kan een *attribute* aan een kaart toevoegen / van een kaart verwijderen. Een *attribute* is een element dat betekenis heeft binnen de context van het spel, het heeft een naam en één of meerder waardes.
    - Een *attribute* kan een nummer als waarde hebben.
    - Een *attribute* kan een naam als waarde hebben.
    - Een *attribute* kan een kaartbeschrijving (stuk tekst) als waarde hebben.
- De gebruiker kan de locatie (op de kaart) van een *attribute* aanpassen.
- De gebruiker kan een overzicht zien van alle kaarten.
- De gebruiker kan een template maken van een kaart, kaarten die gebruik maken van een template gebruiken dezelfde layout van attributen maar met andere waardes.
- De gebruiker kan notities aan kaarten toevoegen die alleen voor de ontwerper zichtbaar zijn, en niet onderdeel uitmaken van het spel.

Card Collections

Doel: Het beschrijven van decks en het organiseren van kaarten.

- De gebruiker kan een collectie toevoegen of verwijderen.
    - De gebruiker kan kaarten aan de collectie toevoegen / van de collectie verwijderen.
    - De gebruiker kan aangeven hoeveel van elke kaart in de collectie moet.
- De gebruiker kan notities toevoegen om de collectie te beschrijven.

Game Sandbox

Doel: Het visualiseren van de spel componenten, het experimenteren met de spel componenten.

- De gebruiker kan kaarten toevoegen aan het spel / verwijderen van het spel.
    - De gebruiker kan van een of meerdere kaarten selecteren van de *card library* en per kaart aangeven hoeveel hij er wil toevoegen.
    - De gebruiker kan een complete collectie toevoegen.
- De gebruiker kan kaarten verplaatsen.
- De gebruiker kan kaartlocaties (zones) toevoegen. De locatie heeft een betekenis binnen de context van het spel en bepaald de presentatie van de kaarten.
    - Een kaartlocatie kan een stapel zijn.
        - de gebruiker kan de bovenste/onderste kaart pakken.
        - de gebruiker kan een kaart bovenaan/onderaan neerleggen.
    - Een kaartlocatie kan een spread zijn.
        - de gebruiker kan elke kaart selecteren / verplaatsen.
        - de gebuiker kan de volgorde aanpassen.
    - Een kaartlocatie kan open of gesloten zijn (face-up/face-down).
    - De gebruiker kan een naam toevoegen aan een kaartlocatie.
- De gebruiker kan kaarten aan locaties toevoegen of kaarten van locaties verwijden.
    - Afhankelijk van de locatie kan de gebruiker bepalen waar nieuwe kaarten worden toegevoegd of verwijderd.
- De gebruiker kan een speler toevoegen / verwijderen.
    - De gebruiker kan aangeven welke spelers welke locaties mogen zien (gesloten kaarten blijven gesloten maar de speler mag spieken).
    - De gebruiker kan voor elke speler aangeven welke locaties ze mogen beïnvloeden en wat ze mogen doen.
- De gebruiker kan wisselen tussen de game designer view (kan altijd spieken) en de view van elke speler (in een lijst klikken of met hotkeys switchen).
- De gebruiker kan in en uitzoomen.
- De gebruiker kan zijn ontwerp opslaan en laden.
    - De gebruiker heeft een lijst van alle iteraties.
    - De gebruiker kan notities toevoegen aan elke iteratie.

Should have

Card Editor

- De waarde van een *attribute* kan gekoppeld worden aan een symbool. e.g. het type van een kaart wordt weergegeven met een symbool.
- De gebruiker kan meerdere kaarten selecteren en hiermee in één keer alle waardes aanpassen.
- De gebruiker kan kaarten genereren aan de hand van een aantal parameters.
    - e.g. maak kaarten van elke mogelijke samenstelling van een of meer *attributes.*

Card Collections

- De gebruiker kan kaarten sorteren op *attributes.*
- De gebruiker kan kaarten zoeken op *attribute* of text in een notitie.

Game Sandbox

- De gebruiker kan aangeven welke spelers welke locaties mogen beïnvloeden (Welke kaarten mogen ze verplaatsen).
- De gebruiker kan notities toevoegen aan / verwijderen van het “canvas”
    - De gebruiker kan de tekst aanpassen.
    - De gebruiker kan de notitie verplaatsen.
- De gebruiker kan tijdelijke waardes bijhouden door middel van tokens.
    - Hij kan tokens toevoegen aan / verwijderen van kaarten.
- De gebruiker kan de waardes van geselecteerde componenten zien in een inspector.

Playtest

Evalueren

- Tijdens het spelen worden alle interacties bijgehouden in een log.
- All logs worden opgeslagen.
- De gebruiker kan logs verwijderen.

Could have

Sandbox

- De gebruiker kan een groep componenten markeren. e.g. een kleur code / tag.

Playtesting

- De gebruiker kan spellers simuleren met behulp van een simpele AI
- De gebruiker het gedrag van de AI instellen om verschillende strategieën te simuleren.

Game Structure

Doel: (onderdelen van) het spel formaliseren, het is een laag over de sandbox mode.
Het moet de ontwerper cognitief ontladen door bij te houden welke spelregels er op elk moment van toepassing zijn.

- De gebruiker kan een game structure creëren (een soord state machine)
- De gebruiker kan fases definiëren (toevoegen / verwijderen)
- De gebruiker kan aan elke fase acties toevoegen om component manipulatie te automatiseren.
    - e.g. het schudden van een stapel kaarten, het delen van kaarten.
    - e.g. het zetten van de actieve speler.
    - e.g. het wisselen tussen fases.
- De gebruiker kan per fase noteren welke spelregels er gelden.
- De gebruiker kan per fase noteren wanneer deze overgaat naar een andere fase.
- De gebruiker kan per actie aangeven wanneer hij gedaan mag worden (er wordt niets gecontroleerd door de tool, het is alleen een notitie)
- De gebruiker kan zijn spel spelen.
    - Hij kan handmatig door alle processes heen klikken.
    - Hij kan alle regels automatisch uitvoeren totdat er een speler actie verwacht wordt of als een regel niet geautomatiseerd is.
        - Er zit een korte pauze tussen elke regel.
        - Er is minimale visuele feedback wat er gebeurd (kaarten die verplaatst worden of waardes die veranderen).
- De spelregels hebben een formele structuur waardoor de gebruiker hints kan krijgen wat wel en niet mag.
    - e.g. een specifieke actie mag alleen als de speler drie “goud kaarten” in zijn hand heeft. de actie kan nog steeds gedaan worden maar het wordt aangegeven als valsspelen.
- De gebruiker kan regels toevoegen aan een fase.
    - als er meerdere regels tegelijkertijd mogelijk zijn moet de gebruiker een keuze maken welke toegepast wordt (vertakkingen).
    - of in welke volgorde ze uitgevoerd moeten worden.
- De gebruiker kan *tokens* toevoegen / verwijderen om tijdelijke waardes bij te houden.
    - e.g. score van spelers
    - tokens kunnen globaal bestaan als waarde.
    - tokens kunnen op kaarten, zones of players geplaatst worden.

    - spelregels kunnen verwijzen naar tokens op een kaart.

Evalueren

Doel: inzicht krijgen in de kwaliteiten van een ontwerp

- De gebruiker kan informatie opvragen van een collectie kaarten.
    - De gebruiker kan informatie krijgen over de verdeling van kaarten als hij een aantal kaarten krijgt van een stapel.
        - e.g de gebruiker wil weten hoeveel kaarten een speler van een stapel moet pakken om 75% kans te hebben dat er specifieke kaarten tussen zitten.
        - e.g. de gebruiker wil weten wat de gemiddelde verdeling is van attributen als een speler 5s kaarten pakt.
    - De gebruiker kan van een collectie of selectie kaarten de verdeling van *attributes* en hun waardes zien.
- De gebruiker kan commentaar plaatsen bij elke log, in de vorm van een text beschrijving.