Bazillionaire

[working title]

LUIK A

BUDGET OVERZICHT

(alle bedragen in €)

Project: Bazillionaire (working title) Aanvrager: CatLab Interactive

Code	Omschrijving	Totaal	Vlaams	Niet-Vlaams
01-00	PREPRODUCTIE	22.500,00	22.500,00	
02-00	SCENARIO & RECHTEN			
03-00	MUZIEK			
05-00	GAMEDESIGNER			
	Totaal Above-the-line	**22.500,00**	**22.500,00**	**0,00**
10-00	PLOEG PRODUCTIE & ADMIN.			
11-00	PLOEG REGIE			
12-00	PLOEG BEELD			
13-00	PLOEG KLANK			
14-00	PLOEG COMPUTER PROGRAMMATIE & CGI			
19-00	PLOEG VERTOLKING			
20-00	SPECIAL EFFECTS			
21-00	MATERIAAL CAMERA			
24-00	MATERIAAL KLANK			
25-00	HUUR STUDIO'S & MAGAZIJNEN			
26-00	MATERIAAL COMPUTER			
27-00	PRODUCTIEBUREAU, POST, TELEFOON			
30-00	FILMS, NEGATIEVEN, FOTO'S			
31-00	GELUIDSDRAGERS			
33-00	OPNAMEKOSTEN			
35-00	TRANSPORT			
36-00	HOTEL & MAALTIJDEN			
	Totaal Productie	**0,00**	**0,00**	**0,00**
50-00	PLOEG POST-PRODUCTIE			
51-00	PLOEG MONTAGE			
52-00	PLOEG KLANK & MIXAGE			
53-00	KLANKSTUDIO'S			
54-00	MONTAGECELLEN			
58-00	HOTEL & TRANSPORT			
	Totaal Post-Productie	**0,00**	**0,00**	**0,00**
70-00	PROMOTIE	2.000,00		2.000,00
72-00	WEBRUIMTE	1.307,50	407,00	900,50
73-00	FINANC. & JURID. KOSTEN	7.000,00	7.000,00	
74-00	TAALVERSIES			
75-00	VERZEKERINGEN			
	Totaal Diversen	**10.307,50**	**7.407,00**	**2.900,50**
	Totaal Below-the-line	**10.307,50**	**7.407,00**	**2.900,50**
	Totaal Above & Below-the-line	**32.807,50**	**29.907,00**	**2.900,50**
90-00	**ONVOORZIENE KOSTEN (max 10% op Below-the-line)**	**1.030,75**	**740,70**	**290,05**
91-00	**OVERHEADS (7,5% op Above + Below-the-line)**	**2.460,56**	**2.243,03**	**217,54**
92-00	**HONORARIUM PRODUCENT (7,5% op A + B-the-line)**	**2.460,56**	**2.243,03**	**217,54**
	ALGEMEEN TOTAAL	**38.759,38**	**35.133,75**	**3.625,63**

Dit formulier is een gereduceerde versie van het standaard budgetformulier van het VAF, dat ook de basis vormt voor de interne gegevensverwerking. Op termijn zal een aangepaste systematiek voor de gamesector op punt gesteld worden. Gelieve naast dit overzicht ook een gedetailleerd budgetoverzicht mee te sturen, in een vorm naar keuze maar wel traceerbaar naar de rubrieken van het overzichtsformulier.

© CatLab Interactive 30/10/2012 – Confidential

Project: Bazillionaire Aanvrager: CatLab Interactive

	€	%
VLAAMS AANDEEL	38.759	100,0%
NIET-VLAAMS AANDEEL	0	0,0%
TOTAAL	**38.759**	**100,0%**

VLAAMS AANDEEL			conf'd	€	%	% TOT
Overheidssteun	VAF	Pre-productiesteun		19.380	50,0%	50,0%
		Productiesteun			0,0%	0,0%
	Andere				0,0%	0,0%
Inbreng producent	Eigen kapitaal			19.380	50,0%	50,0%
	Participatie honorarium producent				0,0%	0,0%
	Participatie faciliteiten producent				0,0%	0,0%
	Participatie overheads				0,0%	0,0%
Coproductie					0,0%	0,0%
					0,0%	0,0%
Participaties					0,0%	0,0%
					0,0%	0,0%
					0,0%	0,0%
Leningen (ter financiering vh projectbudget)					0,0%	0,0%
Risicokapitaal					0,0%	0,0%
Voorverkopen					0,0%	0,0%
					0,0%	0,0%
					0,0%	0,0%
Europese steun					0,0%	0,0%
					0,0%	0,0%
					0,0%	0,0%
Andere (sponsoring…)					0,0%	0,0%
					0,0%	0,0%
Totaal Vlaams aandeel				**38.759**	**100,0%**	**100,0%**

NIET-VLAAMS AANDEEL	Producent:	...		...		...		**Totaal €**	**%**	**%**
	Land:	...	conf'd	...	conf'd	...	conf'd			
Overheidssteun								0	0,0%	0,0%
Inbreng producent								0	0,0%	0,0%
Coproductie (tv, …)								0	0,0%	0,0%
Participaties								0	0,0%	0,0%
Leningen								0	0,0%	0,0%
Risicokapitaal								0	0,0%	0,0%
Voorverkopen								0	0,0%	0,0%
Europese steun								0	0,0%	0,0%
Andere								0	0,0%	0,0%
Totaal Niet-Vlaams aandeel		**0**		**0**		**0**		**0**	**100,0%**	**0,0%**

4. Woord van de producent

“Een nieuwe space game, wie zit daar nu op te wachten?” was de reactie die ik meteen kreeg. “Er zijn al zoveel space games. Wat zal er anders zijn in het jouwe?” “Alles.” antwoordde ik. Het duurde één slapeloze nacht om Jasper te overtuigen.

In 1984 brachten David Braben en Ian Bell Elite[1] uit. Het was niet het eerste space game, maar legde wel de basis voor o.a. E.V.E. Online. De speler kreeg een spaceship, ging de ruimte in, schoot op A.I. gestuurde schepen, raapte bruikbare onderdelen (loot) op, vloog verder, en herhaal. En daarna ging hij naar de “shop” en kocht hij een beter schip.

En natuurlijk werkte het. De speler werd bij elke kill beloond. Elke klik in het zwarte scherm lost een schot, geeft een bonus. De “experience” gaat stadig omhoog. De game theorie ten volle benut. E.V.E. Online[2] ging uiteraard verder dan het basisconcept, maar een beginnende speler blijft vliegtuigjes neerknallen. Eenmaal voorbij het *minen* is het een uitstekend spel, natuurlijk.

Een nieuwe lading space games pakt het helemaal anders aan. Het zijn de strategy space games, waarvan één van de eersten misschien wel Mankind[3] was. Het schieten was niet meer het belangrijkste onderdeel, plots ging het om strategie. Later kwam OGame[4] met hetzelfde concept, maar dan in browser vorm. Text based, omdat meer technologie nog niet voor handen was. Vandaag heeft OGame nog steeds 700'000 spelers.

Het Duitse Dark Orbit[5] haalde het aloude “spaceship schieten” weer helemaal boven. Browsers kregen nu meer mogelijkheden en Flash kwam (met actionscript 3) niet meer “van de hel”. Het was een sprite based 2D game. En het was mooi, en het liep vlot, en het was leuk om vliegtuigjes te schieten. 75 miljoen spelers vonden dat ook.

Maar de technologie staat niet stil en nu, zes jaar later, is het tijd voor wat nieuws. 3D behoort nu ook tot de mogelijkheden op het web en sommige mensen zijn de glorie van Elite nog niet vergeten. Elite, die de basis legde voor de space trading game. Elite, waar het scheepjes schieten niet het belangrijkste gameplay element is.

Met Bazillionaire willen we precies dát bieden. De technologie is er, de markt ook. Met E.V.E. zullen we nooit concurreren, maar misschien zijn sommigen het scheepjes schieten in Dark Orbit nu wel afgeleerd.

In Bazillionaire is het “gevecht” gegeven slechts een klein onderdeel van de gameplay. Het grootste deel van de gameplay wordt opgeslokt door trading, construction, research, invention en zelfs programming. Het spel zelf biedt je enkel de basiselementen, de markt zelf moet door de spelers worden opgebouwd.

We gaan zelfs een stap verder door spelers de mogelijkheid te geven zelf handelszaken te laten programmeren. Casino's, banken, interim kantoren, verzekeringskantoren, ship repair construction sites, … De mogelijkheden worden enkel beperkt door de fantasie van de spelers.

En alles in een virtuele omgeving die volgens wereldse economische regels opereert. Zodat we ook eens leren hoe die rijken nu eigenlijk rijk blijven.

Buy low. Sell high.

---

1 Elite. Release in 1984. http://en.wikipedia.org/wiki/Elite_(video_game)
2 E.V.E. Online. Release in 2003. http://www.eveonline.com/
3 Mankind, eerste release in 1996. Gepubliceerd door het later failliet gegane Franse Cryo Networks. http://www.mankind.net/
4 OGame, gestart in 2002. Wordt gespeeld in de browser en is voornamelijk text based, met wat achtergrondafbeeldingen. Uitgebracht door Gameforge. http://www.ogame.org/
5 Dark Orbit. Release in 2006. Massive multiplayer game in Flash.

5. Game design

Bazillionaire wordt een *persistent massive multiplayer browser game*. Dit betekent:

- Het spel wordt gespeeld in de webbrowser, waardoor het erg toegankelijk is voor iedereen met een internetverbinding.
- Het spel berust volledig op samenwerking tussen spelers. Groepen van spelers werken ofwel samen, of kunnen andere groepen saboteren om zo hun positie in het spel te verbeteren.
- Het spel is persistent, wat betekent dat het spel blijft veranderen zelfs terwijl de spelers niet spelen. Spelers kunnen door middel van allerlei tools hun bedrijf zo inrichten dat het blijft draaien ook als ze op dat moment niet aan het spelen zijn.

Het spel speelt zich af in een niet nader benoemde toekomst. De uitvinding van de FTL[6]-motor heeft de mensheid toegang gegeven tot het hele universum. Het mensdom begint andere planeten te koloniseren en ontdekt snel dat het niet alleen is. Een hele diversiteit aan ruimtewezens heeft een stevige voorsprong op de interstellaire goederenhandel.

De spelmechanismes in Bazillionaire zijn bewust relatief eenvoudig gehouden om de leercurve zo laag mogelijk te houden. Eenmaal een speler zich verder ontwikkelt, moeten de mechanismes echter open genoeg zijn om ingewikkelde constructies, samenwerkingsakkoorden, oorlogen, … mogelijk te maken. Door met een eenvoudige, maar open basis te beginnen, hopen we dit mogelijk te maken.

Voor de visuele representatie van het spel kijken we naar WebGL. Deze technologie staat nog in zijn kinderschoenen, maar wordt ondertussen wel ondersteund door de meerderheid van de webbrowsers[7]. WebGL laat ons toe indrukwekkende 3D modellen te tonen en de speler het gevoel te geven dat ze echt in de ruimte vliegen, in tegenstelling tot de text-based of 2D games die momenteel de browser-game markt overspoelen.

Als alternatief op WebGL willen we echter ook een lage kwaliteit canvas-rendering methode aanbieden die dan gebruikt kan worden op o.a. mobile devices. Mobile devices zijn echter niet onze primaire markt, maar het lijkt ons toch belangrijk toegang tot de game via mobile devices mogelijk te maken. (Dit kan eventueel ook via een Unity client => zie aanzet tot technisch design document.)

Dankzij websockets[8] (of fallback methodes) zijn browsers nu ook in staat *echte* real time communicatie tussen spelers mogelijk te maken. Hiermee kunnen we de vlucht van de schepen in real time weergeven, oorlogen uitvechten, …

Ik wil met Bazillionaire zo snel mogelijk gebruik maken van de nieuwe technologieën die moderne browsers bieden en zo de markt van de browser game naar een nieuw niveau trekken. Dankzij het zijproject Gambic kan het spel ook meteen gepubliceerd worden als een Facebook app en kan ik het spel ook via de Firefox- en Chrome shop aanbieden.

Tot slot wil ik even benadrukken dat de naam Bazillionaire een *working title* is. Hoewel we in eerste instantie een knipoog naar LavaMind's Gazillionaire[9] wouden geven – een humoristisch marktsimulatiespel uit 1995 – berust er een trademark op de naam Gazillionaire.

6 Faster-than-light motor. http://nl.wikipedia.org/wiki/Sneller_dan_het_licht

7 WebGL wordt momenteel ondersteund in: Mozilla Firefox 4.0 en later, Google Chrome 9 en later, Safari 5,1 en later (standaard uitgeschakeld), Opera 11 en later (standaard uitgeschakeld). Voor internet explorer zijn er plugins voor handen. http://en.wikipedia.org/wiki/WebGL

8 Google's Socket.IO biedt een ijzersterke implementatie van het websocket protocol. http://socket.io/

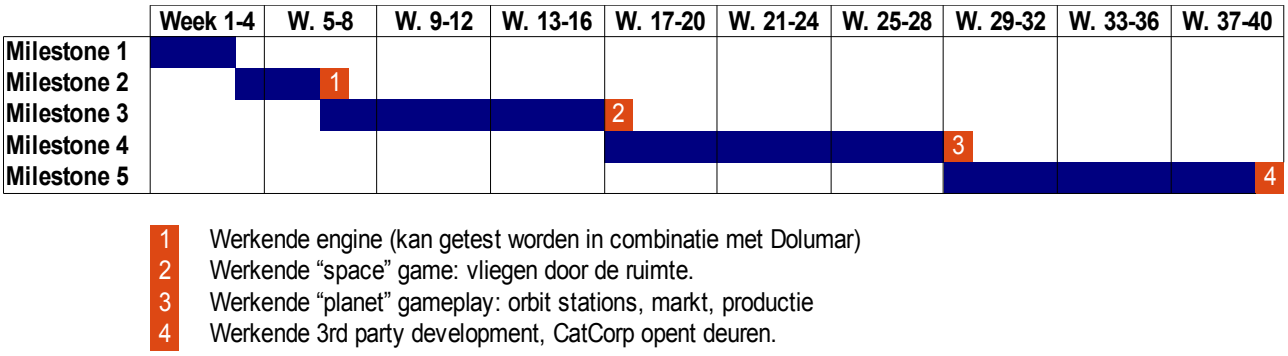
9 http://www.gazillionaire.com/index.php

6. Werkprocedure

De milestones die hier opgenomen zijn, zijn behoorlijk technisch. Neem daarom ook een kijkje in het deel “technish design document”.

We willen het prototype in maximaal **9 maanden** klaar hebben. Het prototype zal echter meteen ook de basis van het eindproduct zijn, waarbij enkel de grafische laag belangrijke updates zal moeten krijgen. Het prototype moet een dermate kwaliteit hebben dat het reeds gebruikt kan worden voor gameplay testing op grote schaal.

Voor de ontwikkeling van het prototype hebben we de volgende milestones gedefinieerd:



Afbeelding 1: Visuele representatie milestones.

Milestones

1 Ontwikkeling Nodejs communciatieserver [3 weken]

Een deel van de nodige programmatie kan hergebruikt worden uit ons vorige project Dolumar (speler registratie, communicatiesystemen tussen spelers, ranking, integratie OpenID voor identificatie, integratie PUSH notificaties, …) Dolumar is echter volledig geschreven in PHP.
Gezien PHP niet goed overweg kan met websockets, en die in dit project erg belangrijk zijn, hebben we besloten om een server-side javascript laag tussen de PHP software en de client te zetten. Deze server zal alle real-time communicatie op zich nemen, maar voor de complexere (trage) systemen, blijven we PHP gebruiken.

2 Ontwikkeling client [3 weken]

Dit bevat een prototype van de 3D interface, geprogrammeerd in javascript. Communiceert met de NodeJS communicatieserver en biedt de PHP server de mogelijkheden om informatie op het scherm te tonen.
Net zoals bij Dolumar kiezen we voor een *thin client*, dat betekent dat er in de client zelf erg weinig logica te vinden is. De communicatieserver levert de informatie in XML / XHTML formaat aan, de client toont deze en biedt de spelers alle mogelijke opties om op deze informatie te reageren.
Het prototype zal gebruik maken van online beschikbare open source 3D modellen en graphics die dan in productiefase vervangen worden door hun finale tegenhangers.

3 Basis space-gameplay [10 weken]

Ontwikkeling van de basis gameplay elementen, opgesplitst in:

1. Willekeurige opbouw van het universum. “the map”
2. Mogelijkheid om met een schip tussen planeten te vliegen.
3. Aanvallen: spelers kunnen elkaar aanvallen en producten stelen terwijl ze al vechtend door de ruimte razen.
4. Markt-mechanisme op planeten: spelers kunnen producten kopen en verkopen. Hier moet ook een “eenvoudig” prototype van de marktsimulatie software ingebouwd worden zodat grondstoffen aan een beredeneerde prijs gekocht en verkocht worden.
5. Aanpassen van het schip: nieuwe motor, nieuwe boordcomputer, ...

4 Basis planet-gameplay [12 weken]

De speler krijgt de mogelijkheden fabrieken te bouwen op een planeet. In deze fabrieken kunnen ze producten produceren doormiddel van een recipe technology tree.
Bevat:

1. Aankoop bouwgrond op een planeet.
2. Onderzoek in tech-tree
3. Visuele weergave van bouwgrond & mogelijkheid tot bouwen machines. (2D isometrische interface gebaseerd op de Dolumar engine.)
4. Simulatiesysteem productiviteit van elke fabriek.
5. Contracten-systeem: een open module waarbij spelers contracten tussen elkaar kunnen opstellen (= handelen, jobs, verzekering, …) Wordt later ook als basis gebruikt voor de 3rd party development tools.
6. Visuele programmatie.

5 Basis CatCorp & 3rd party development [11 weken]

Spelers kunnen in een later stadium van het spel zelf handelszaken opzetten op planeten. Hierbij kunnen spelers ofwel zelf applicaties hosten, ofwel gebruik maken van de beschikbare modules die we ontwikkelen.
Bovendien is er standaard een *monopolist* in het spel (“CatCorp”). Dit is een virtuele speler die andere spelers op weg helpt door de standaardfunctionaliteit aan te bieden (zoals verzekeringen, verkoop van schepen, verkoop van machines, …) Deze virtuele speler heeft toegang tot dezelfde tools als andere spelers en op termijn is het dus mogelijk te concurreren met de monopolist.
Bevat:

1. Ontwikkeling 3rd party API access via iframe met javascript & webservice API's. Vergelijkbaar met facebook apps.
2. Ontwikkeling van CatCorp services die gebruik maken van bovenstaande API's.
3. (Testomgeving voor 3rd party developers – niet nodig voor prototype, maar misschien nodig als testomgeving voor 5.2)

Interface, art & models

Tijdens het bouwen van het prototype wordt er voornamelijk gebruik gemaakt van open source graphics. Hierdoor verlagen we de kost van het prototype drastisch. Zodra het prototype klaar is, kunnen we met het prototype in de hand op zoek naar een designer.

Er zijn echter voldoende graphical assets beschikbaar die – zelfs in een commerciële omgeving – vrij gebruikt kunnen worden (mits vermelding van de artiest in de credits).

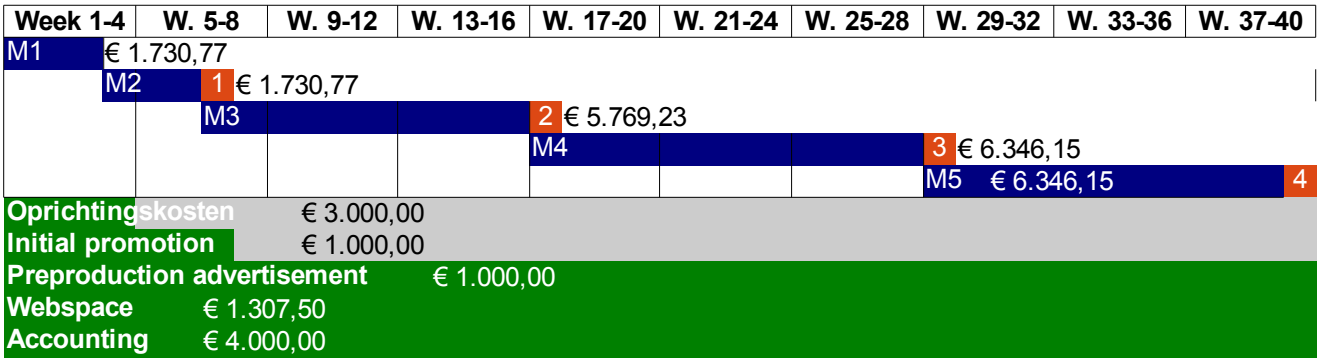
Financiering

Het project wordt gefinancierd met eigen middelen en steun van het VAF/Gamefonds. In een later stadium en met het prototype klaar willen we een *artist* in het project betrekken die dan de eigenlijke look & feel van de game zal bepalen.

In wezen vormt de eigenlijke productiekost dus de grootste pot. Daarnaast zijn er fondsen vrijgehouden voor pre-release advertisement (in de vorm van een developer blog en een kleine hoeveelheid advertisement om interesse te wekken via social media), webhosting, boekhouding, etc.

Financiering gebeurt 50% vanuit eigen middelen en 50% met steun van het VAF/Gamefonds. Indien steun van het gamefonds uit blijft, ligt het project nog steeds op tafel, maar dan moeten we alles wat kleinschaliger aanpakken.

	€ 38.759,38	€ 19.379,69		€ 19.379,69	
Uitgaven:	**Raming totaal**	**Eigen kapitaal**		**VAF/Gamefonds**	
Oprichtingskosten	€ 3.000,00	€ 3.000,00	100%	€ 0,00	0%
Initial promotion	€ 1.000,00	€ 0,00	0%	€ 1.000,00	100%
Preproduction advertisement	€ 1.000,00	€ 0,00	0%	€ 1.000,00	100%
Webspace	€ 1.307,50	€ 653,75	50%	€ 653,75	50%
Accounting	€ 4.000,00	€ 2.000,00	50%	€ 2.000,00	50%
Production:					
Milestone 1	€ 1.730,77	€ 865,38	50%	€ 865,38	50%
Milestone 2	€ 1.730,77	€ 865,38	50%	€ 865,38	50%
Milestone 3	€ 5.769,23	€ 2.884,62	50%	€ 2.884,62	50%
Milestone 4	€ 6.923,08	€ 3.461,54	50%	€ 3.461,54	50%
Milestone 5	€ 6.346,15	€ 3.173,08	50%	€ 3.173,08	50%
Overhead, onvoorziene kosten, …	€ 5.951,88	€ 2.475,94	42%	€ 3.475,94	58%



Risicofactoren

Hoewel CatLab Interactive de mogelijkheden zoveel mogelijk onderzocht heeft, houden we rekening met de volgende risicofactoren.

1 Technische limitaties

Risico

Hoewel we al enkele performantie testen gedaan hebben in de WebGL omgeving, is het mogelijk dat we naarmate het project vordert in problemen geraken met onze library. Three-js, de library die we voor het project willen gebruiken, heeft zijn kunnen bewezen in vele voorbeelden[10] en enkele games, maar nog niet in grote, commerciële projecten.

Het is mogelijk dat we daarom performantieproblemen in de client side kunnen krijgen.

Oplossing

Gezien we een thin client gebruiken, is het relatief eenvoudig om een nieuwe client te schrijven. Eén van de opties daarin is een client in Unity, die dan later ook gebruikt kan worden voor mobile devices.

In eerste instantie wilt CatLab Interactive echter bij javascript blijven omdat Unity nog geen linux support biedt en omdat voor Unity een plugin geïnstalleerd moet worden.

2 Gameplay niet interessant

Risico

Hoewel wij het spel zelf leuk vinden, is het steeds mogelijk dat we de enigen zijn.

Oplossing

Daarom willen we zo snel mogelijk speltests houden. Dankzij Dolumar heeft CatLab Interactive een kleine basis hardcore browser game spelers opgebouwd, die we zo snel mogelijk willen inschakelen om het spel voor ons te testen.

Vanaf de 14de week in de ontwikkeling van het prototype willen we daarom al contact opnemen met de geïnteresseerden om “gametest events” te houden. Daarin tonen we dan, gedurende een paar uur en voor een selecte groep spelers, wat de mogelijkheden in het spel zijn.

3 Overschrijding van geraamde tijd

Risico

De geschatte tijd voor de milestones wordt overschreden.

Oplossing

CatLab Interactive ziet de milestones niet als een richtlijn, maar als een contract. Indien we een milestone niet halen, zullen we genoodzaakt zijn – in de mate van het mogelijke – functionaliteiten te schrappen en zoveel mogelijk naar de basis te werken.

We volgen de SCRUM methodes in ontwikkeling met een release cycle van ongeveer 3 weken. In ontwikkeling beginnen we bij de basis, maar zorgen we ervoor dat we elke drie weken een werkende versie kunnen releasen. Elke iteratie voegt nieuwe functionaliteiten toe tot alle functionaliteiten van de milestone ingebouwd zijn. Hierdoor verlagen we het risico.

10 Three.js proof of concepts: http://mrdoob.github.com/three.js/

7. Concept

Het begin

De vrije markt economie heeft zich, met de komst van de eerste commerciële ruimtetuigen, verruimd tot het hele heelal. In het begin van het spel wordt de speler gecontacteerd door een sale representative van CatCorp VC, die hem een een geldbedrag aanbiedt in ruil voor een participatie van 10% in het nieuw op te richten bedrijf. Deze participatie kan ten alle tijde afgekocht worden, mits een minimum return van 100%.

De speler gebruikt de investering om een ruimteschip te kopen. Het ruimteschip heeft zekere cargo-grootte en een aantal module slots die gebruikt kunnen worden om het schip sterker te maken (zie "Ruimteschepen").

De speler kan met het ruimteschip tussen de verschillende planeten vliegen en daar handel drijven. De speler vliegt bijvoorbeeld naar Mars om daar Broodroosters te kopen, vliegt met een volle cargo naar Venus om die daar tegen winst te verkopen, koopt daar Uranium en vliegt daarmee naar Pluto omdat daar een Uranium-tekort is, …

De speler moet zijn schip echter uitrusten met Drones, dit zijn robots of wezens die het schip beschermen in geval van een aanval. Hij kan het schip natuurlijk ook gebruiken om andere spelers aan te vallen en te proberen hun grondstoffen te stelen. De schepen maken elkaar niet af, daarvoor zijn de vernieuwde starwars-achtige schilden te sterk; er wordt enkel gevochten om de buit: de cargo.

Naast de markt is het ook mogelijk om contracten af te sluiten in de verschillende handelszaken die op elke planeet te vinden zijn. Deze handelzaken zijn ofwel eigendom van de monopolist CatCorp, ofwel zijn ze eigendom van andere spelers. Deze contracten kunnen zijn: jobs (breng grondstof A uit mijn silo op planeet B naar planeet C), verzekeringen (= terugbetaling van gestolen producten), verhandelen van grondstoffen / schepen / fabrieken, …

Daarnaast heeft de speler ook de mogelijkheid om bouwgrond te kopen om fabrieken op te bouwen. Dit is echter enkel mogelijk nadat de speler al wat kapitaal heeft opgebouwd als loopjongen. Bouwgrond is duur en machines zijn nog duurder. (Zie "Fabrieken".)

Space

Waarom het ruimte-thema? Er zijn verschillende redenen tot het besluit een spel in de ruimte te ontwerpen.

- Het "transport" thema komt erg goed tot zijn recht in een ruimte-omgeving en is bovendien relatief eenvoudig te implementeren. De route tussen twee planeten ligt – in onze vereenvoudigde versie – simpelweg op één lijn.
- Hoewel WebGL redelijk wat mogelijkheden biedt om indrukwekkende 3D omgevingen weer te geven, is het budget voor 3D development beperkt. Een ruimtegame laat relatief eenvoudig tamelijk indrukwekkende visualisaties toe zonder te moeten rommelen met *collision detection* of ingewikkelde plaatsbepalende algoritmen.
- Spelers hebben een goede voeling met plaatsbepaling in de ruimte.

Ruimteschepen

1 Onderdelen

Elk ruimteschip bestaat uit een basis (“hull”) en een aantal onderdelen die vervangen kunnen worden. Ook heeft elk schip een aantal “drone bays” waar de (oorlogs)drones in geplaatst kunnen worden.

De onderdelen:

1. “Hull” (bepaalt hoeveel cargo er in het ruimteschip kan)
2. Motor (bepaalt de snelheid van het ruimteship; snelheid is ook afhankelijk van de cargo)
3. Boordcomputer (bepaalt het aantal commandolijnen dat opgeslagen kan worden)

2 Commando's

Vliegen met een ruimteschip bestaat uit commando's die in een wachtrij geplaatst worden. Het aantal commando's dat je aan een ruimteschip kan geven zijn afhankelijk van de gebruikte boordcomputer. Een eenvoudige boordcomputer kan maar één commando onthouden: “Vlieg naar Mars”. Een meer geavanceerde boordcomputer kan meerdere commando's onthouden of zelfs in een “lusconstructie” gaan waardoor dit ruimteschip de hele tijd tussen twee planeten vliegt (zelfs als de speler offline is.)

Elke speler kan meerdere ruimteschepen hebben en kan op bovenstaande manier met alle ruimteschepen tegelijk vliegen.

Een mogelijk commando is echter ook “ondersteun ruimteschip X”. In dit geval zal het ruimteschip steeds in de buurt van X blijven vliegen en – als X aangevallen wordt – de verdediging op zich nemen.

Alle commando's voor de schepen worden via een visuele programmeeromgeving ingegeven. (Zie “visueel programmeren.”)

Drones

1 Over drones

Drones zijn de basis voor het aanvalssysteem. Een drone kan vergeleken worden met een pokémon: een drone heeft een aantal statistieken die bepalen hoe hard hij is in aanval en hoeveel in verdediging. Tevens wordt een drone sterker naarmate hij ervaring krijgt in oorlogsvoering (= XP of experience).

Er zijn drie (in het prototype) types drones die elk een bepaalde defensie hebben tegenover de andere types. Zo zal een Element (magisch creature) sterker zijn dan een Robot (technisch type), die dan weer sterker zal zijn dan een Technomech; maar die zijn dan weer goed tegen Elements. Doormiddel van deze driehoeksrelatie kan een zwakkere drone toch winnen tegen een sterkere drone.

In de toekomst kunnen deze drones dan ook nog eens uitgerust worden met verschillende wapens, maar dat is buiten de scope van het prototype.

2 Aanvallen

Piraterij is een misdrijf, maar het is ook een stevige bron van inkomsten voor spelers met een grote

vloot fighters. Piraten-spelers vliegen door de ruimte op zoek naar een zwakke target. Spelers kunnen hun schepen de opdracht geven een doelwit aan te vallen door in hun radar op het schip te klikken en het “aanval” commando te geven.

Het schip zal daarop naar de tegenstander vliegen (afhankelijk van de snelheid van de aanvaller en de verdediger duurt dit kort of lang – of komt de aanvaller helemaal niet toe). De verdediger krijgt op dat moment al een melding dat het schip aangevallen wordt.

De verdediger heeft op het moment van deze melding 10 minuten de tijd (minimale tijd – afhankelijk van de afstand dat de aanvaller moet vliegen kan dit langer zijn) om een verdedigingsformatie in te stellen voor zijn drones. In werkelijkheid houdt dit in dat de verdediger voor elke aanvallende drone een verdedigende drone kiest.

Daarna vindt de eigenlijke aanval plaats: de statistieken van de aanvallende drones wordt tegen de statistieken van de verdedigende drones gelegd en afhankelijk daarvan wordt bepaald wie wint en wie de grondstoffen mee naar huis neemt.

3 Ondersteunende schepen

Als schip A ondersteuning geniet van schip B, dan zal in het geval van een aanval op schip A, schip B automatisch de aanvaller X aanvallen. Hierdoor moet schip X dus eerst de aanval van B verwerken, voordat hij verder kan gaan met de aanval op A.

Dit systeem kan ook verder geschaald worden. Bekijken we de volgende situatie:

- Schip A wordt ondersteund door schip B.
- Schip X wordt ondersteund door schip Y.

Bekijken we een eventuele aanval:

- X valt A aan.
- B ziet de aanval en reageert: B valt X aan.
- Y ziet de aanval op X en reageert: Y valt B aan.
- Resultaat: X valt nog steeds A aan, Y wordt aangevallen door B.

Fabrieken, grondstoffen en producten

1 Basis fabrieken

Rond elke planeet zijn er “orbit stations”. Die kunnen ofwel door CatCorp opgericht zijn, of door spelers zelf gebouwd worden (héél erg duur!) In deze orbit stations kunnen spelers “productiegrond” kopen waar ze een klein fabriekje kunnen opstellen.

Het gebruik van orbit stations in plaats van effectief landen op planeten geeft ons de mogelijkheid op een eenvoudigere manier een mooie grafische impressie van het docking proces te tonen. Ook kunnen we op die manier een schaarste van productiegrond simuleren.

Er zijn twee soorten machines:

1. **Grondstoffen ontginnen**
   Afhankelijk van op welke planeet een speler zich bevindt, is er een grote hoeveelheid grondstoffen te vinden. Op Mars is er bijvoorbeeld veel Uranium te winnen, dus is het een goed idee daar een uraniumontginningsmachine te bouwen.

2. **Grondstoffen produceren**
   Deze machines vormen een gegeven *input* om in een *output*. De *input* en *output* worden gedefinieerd door middel van een recept dat via research gevonden kan worden.
   Voorbeelden van recepten zijn: 3 ijzererts => 1 ijzer. Maar ook: 10'000 vijzen, 200 koper, 35'000 metaal => 1 Fighter X1 ruimteschip.

2 Silo's

Zowel input als output wordt opgeslagen in silo's, dus de speler moet ook voldoende silo's bouwen om de fabriek draaiende te houden. Door middel van contracten of teamplay kan een fabriek echter ook draaiende gehouden worden met een *just-in-time* filosofie, maar daarvoor is een hele hoop coördinatie van de spelers nodig.

3 Tech tree

In regel zijn alle producten in Bazillionaire produceerbaar en verhandelbaar. De monopolist CatCorp biedt van alle producten een basisversie (ruimteschepen, drones, …), maar om echt hogerop te geraken zullen spelers producten van andere spelers nodig hebben. Die producten zullen ze dan via een tech-tree moeten unlocken. Hierbij zal het onmogelijk zijn voor een speler om alle producten in de hoogste graad te kunnen *unlocken*, spelers zullen dus gedwongen worden zich te specialiseren op één bepaald gebied en hun andere producten bij een andere speler te halen.

Een voorbeeld van de levensloop van een machine:

1. Speler A doet research in de branch “ruimteschepen” en ontwikkelt de blueprint voor een nieuwe fighter. Deze blueprint kan enkel gebruikt worden om een machine te bouwen die de nieuwe fighter bouwt, maar daarvoor moet speler A de technologie “machinebouw ruimteschepen – niveau 8” hebben. Dat heeft hij niet, dus contacteert hij Speler B.
2. Speler B heeft “machinebouw ruimteschepen – niveau 8” al eerder ontwikkeld. Dat stelt hem in staat om de machine te bouwen. Speler A en speler B sluiten een contract waarin staat dat speler B de toelating heeft de machine 3x te bouwen, éénmaal voor speler A en 2 maal voor “eigen gebruik” (of om later te verkopen).
3. Speler B laat zijn machines draaien en produceert de gevraagde machine. Hij stuurt deze naar speler A (met een hele vloot aan defensieschepen – een machine is veel geld waard en de piraten staan altijd klaar). Daarna produceert hij dezelfde machine nog twee keer. Het contract zorgt ervoor dat de optie om die machine te bouwen daarna verdwijnt.
4. Speler C heeft veel geld verdiend met schepen aan te vallen en wil nu ook eens fabriek bouwen. Speler B heeft ondertussen de machine op de markt gezet met een vraagprijs van 57'000'000 eurodollar. Speler C koopt de machine, laat die ophalen door één van z'n runners en begint met de productie van fighters. Ondertussen biedt speler A dezelfde fighter echter ook al aan op de markt, dus nu moeten ze elkaar beconcurreren.

Door middel van regelmatige updates zal de tech. tree virtueel oneindig gehouden worden: zodra een speler het einde van de tech. tree dreigt te halen, voegen we een reeks nieuwe technologieën toe.

Zijn inbegrepen in de tech. tree:

1. Alle ruimteschepen (opgedeeld in verschillende categorieën) en hun onderdelen (motor, boordcomputer, ...)
2. Alle drones (opgedeeld in 3 categorieën)
3. Alle machines om de bovenstaande producten te produceren (opgedeeld in het gelijke aantal

categorieën)

4. Brandstof voor de ruimteschepen (betere brandstof = een snellere motor)
5. “Handelwaren” die niet bruikbaar zijn voor de spelers, maar op de interstellaire markt wel geld kunnen opbrengen. (Broodroosters, koffiezet-apparaten, keukenrobots, …)

In het prototype gaan we uit van een eenvoudige tech. tree. De technologie zal echter klaar zitten om “oneindig” uitgebreid te gaan.

4 Het productieproces definiëren

Doormiddel van de visuele programmeeromgeving die we ook in het programmeren van de ruimteschepen willen gebruiken, kan de speler het productieproces van zijn fabrieken definiëren. Zie “visueel programmeren”.

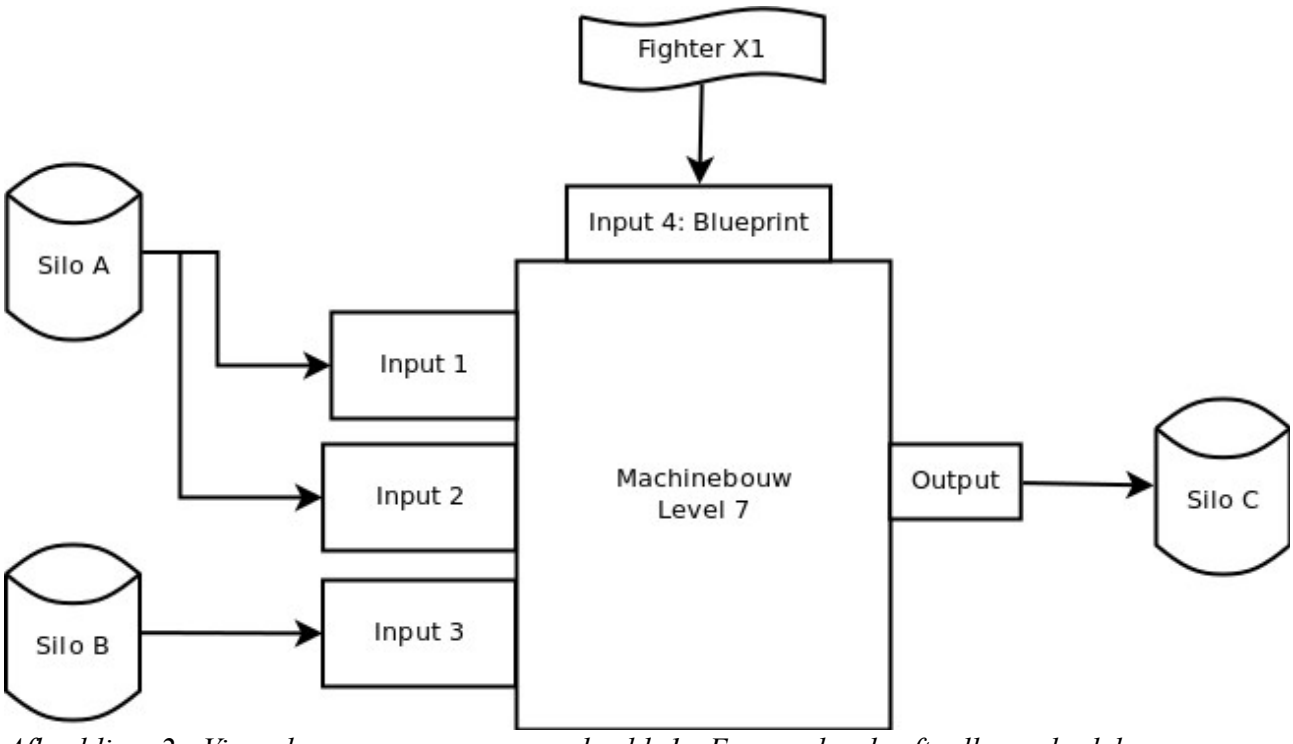
Visueel programmeren

Doormiddel van een node-based programmeeromgeving willen we de speler een visueel hulpmiddel aanbieden waarmee ze hun fabrieken en ruimteschepen kunnen programmeren.

Ons node-based systeem bestaat uit:

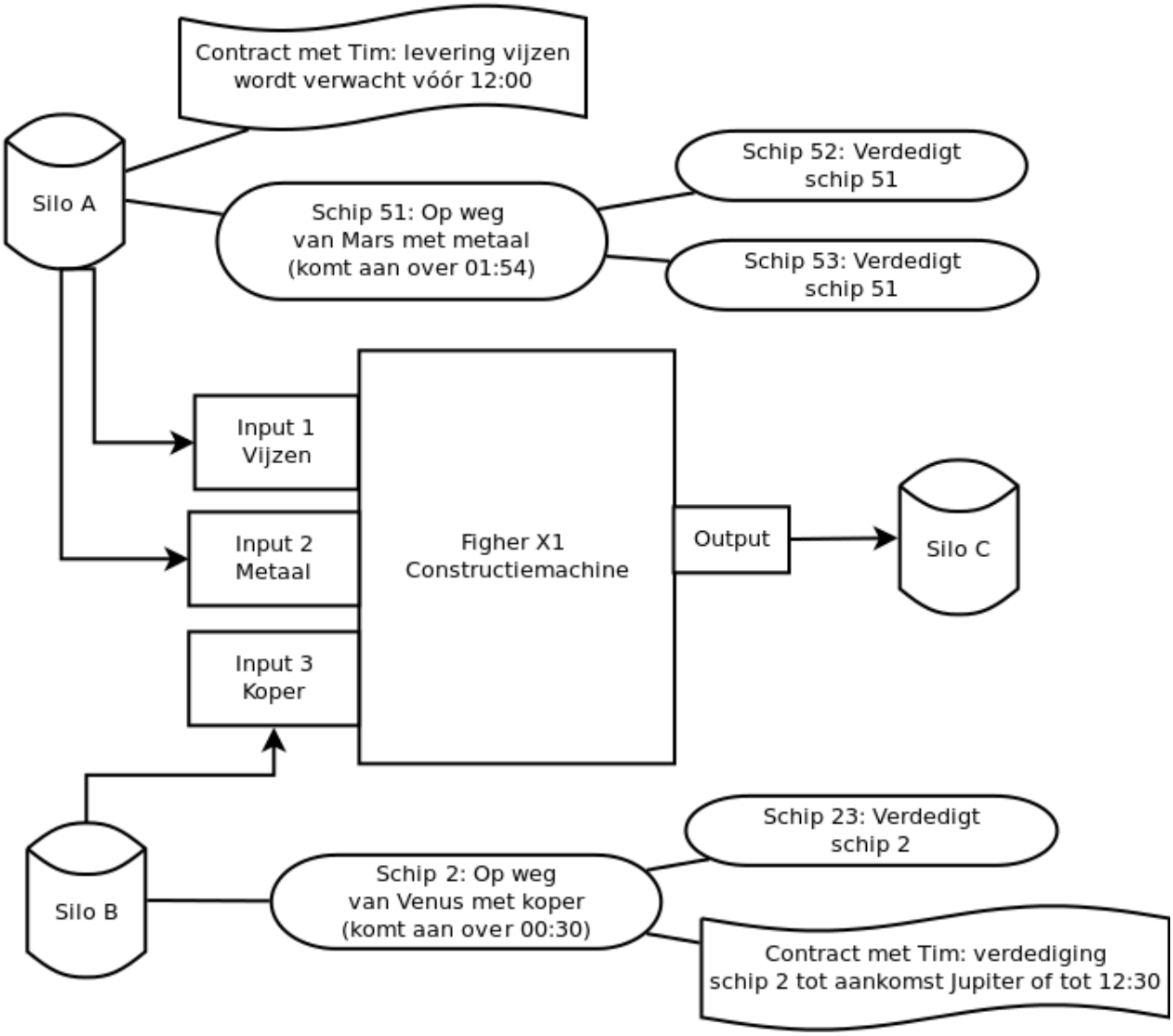
1. Nodes: Dit zijn de eigenlijke elementen die in het spel gebruikt kunnen worden. Nodes zijn bijvoorbeeld ruimteschepen, silo's, machines, blueprints; maar ook logische nodes (om bijvoorbeeld de productie te limiteren), contracten met andere spelers, …
2. Connecties: De speler verbindt de nodes met elkaar doormiddel van connecties. Afhankelijk van de nodes hebben die connecties dan een zekere betekenis.

Onderstaande voorbeelden geven een aantal geavanceerde situaties weer. Grafisch moet het er voor de speler natuurlijk een stuk aantrekkelijker uit zien.



Afbeelding 2: Visueel programmeren, voorbeeld 1: Een speler heeft alle onderdelen voor een FIgher X1 machine in zijn silo's opgeslagen. Input 1, 2 en 3 nemen de grondstoffen gebaseerd op wat er nodig is in de blueprint. De output wordt in Silo C gestoken.

Fabriek op Jupiter:



Afbeelding 3: Visueel programmeren, voorbeeld 2: Via een "just in time" constructie worden de grondstoffen geleverd tijdens de productie. Als de grondstoffen in de silo op zijn, valt de productie stil. Daarom is het belangrijk contracten te sluiten en schepen opdracht te geven de silo's te bevoorraden.

Het multiplayer-aspect

Spelers kunnen ofwel participaties in elkaars bedrijven kopen (en daar dan dividenden uit krijgen), ofwel een nieuw bedrijf oprichten waarin een aantal spelers dan een participatie “koopt”. De transfer van participaties gebeurt steeds via wederzijdse overeenkomst doormiddel van een contract.

Hierdoor kunnen verschillende spelers samen een bedrijf runnen en de taken verdelen.

3rd party development

Zoals eerder beschreven worden er API's opengesteld om de spelers programmatische toegang te

geven tot contracten. Hierdoor wordt het mogelijk voor spelers om zelf functionaliteit aan het spel toe te voegen.

Op elke planeet kan een speler “handelszaken” openen die geschreven- en gehost worden door spelers. Andere spelers kunnen dan deze handelszaken bezoeken en hun functionaliteit gebruiken.

De monopolist CatCorp, die in het spel geïntroduceerd wordt om de initiële leegte op de markt te vullen, zal gebruik maken van deze API's. Op die manier speelt de monopolist volgens exact dezelfde regels als alle andere spelers.

We verwachten niet dat elke speler zelf applicaties kan programmeren, maar spelers kunnen wel applicaties delen tussen elkaar. De programmatuur zal zodanig opgesteld zijn dat het volstaat een URL op te geven bij het opstarten van de handelszaak. De applicatie zal dan, afhankelijk van de speler, de planeet en de handelszaak, de juiste informatie inladen. Spelers die hun applicatie niet willen delen met andere spelers, zijn natuurlijk vrij om dat niet te doen.

8. Keuze van het platform

De keuze “browser based” is voor veel game developers niet evident. De technologie legt beperkingen op, voornamelijk op het vlak van performantie. CatLab Interactive heeft echter een grote kennis opgebouwd rond de bouw van javascript based games (Dolumar) en de bouw van web applicaties (o.a. voor Dashlane SAS).

Er zijn echter ook voordelen aan browser games. Zo is de drempel om het spel te beginnen spelen erg laag en kunnen potentiële spelers met enkele muisklikken overtuigd worden om het spel te proberen. Ook lijkt de game sector zich steeds meer te wenden tot het free-to-play model[11], waar zowat alle browser-games zich aan houden.

De keuze voor het html5 platform ligt voor ons voor de hand. CatLab Interactive heeft immer door de jaren heen een solide kennis van deze technologie opgebouwd. De toekomst van het gamen blijft volgens mij in het web liggen en WebGL brengt die toekomst een grote stap dichter bij.

9. Beoogde doelgroep

In eerste instantie willen we ons richten op casual gamers die de tienerjaren voorbij zijn. De leercurve van het spel moet in het begin laag zijn: rondvliegen en producten kopen en verkopen. De speler leert naar mate zijn voortgang de meer geavanceerde onderdelen van het spel kennen.

Gezien de “ernst” van het spel willen we voornamelijk jong-volwassenen en volwassenen aantrekken.

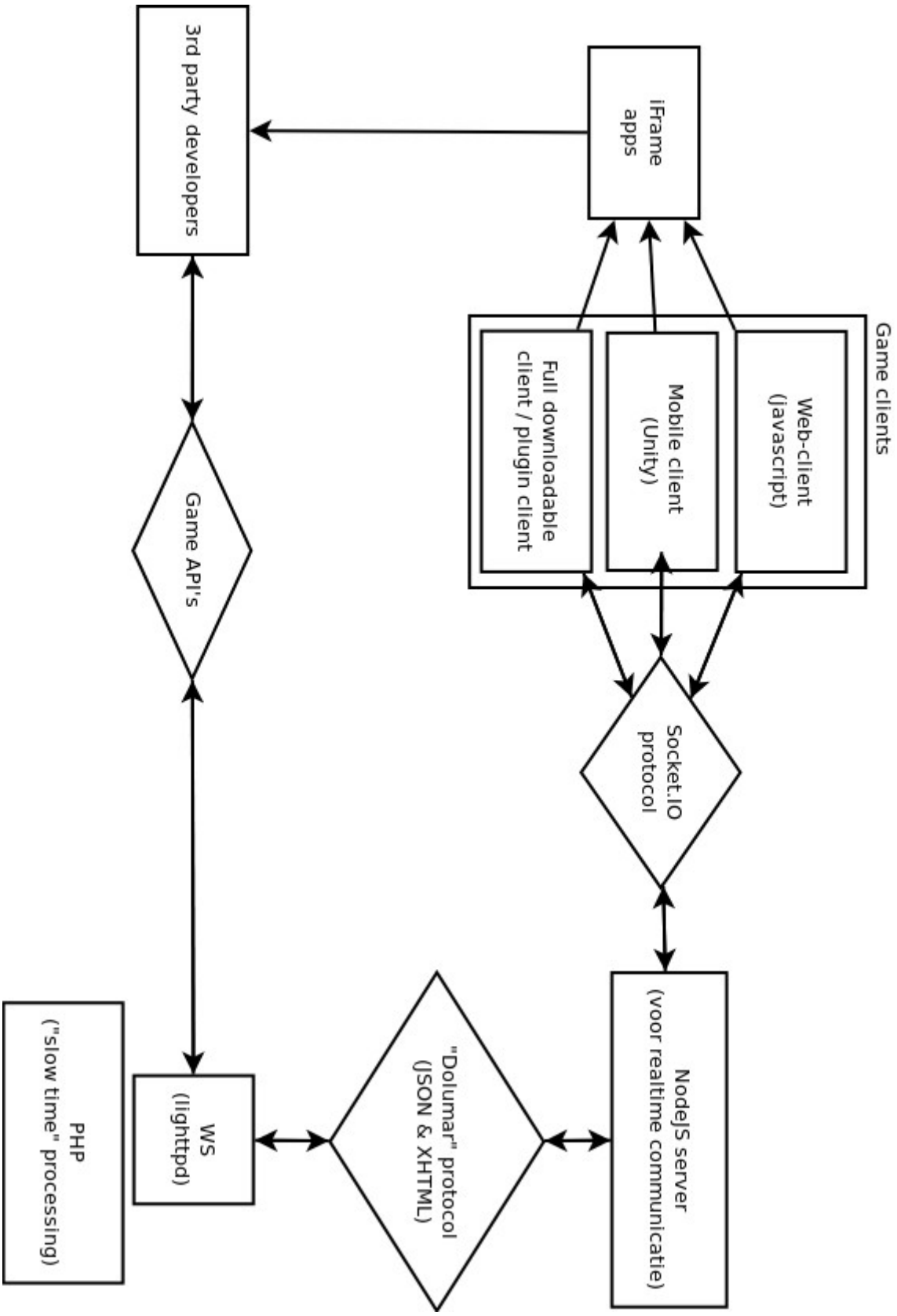
Op termijn willen we echter ook contact opnemen met onderwijsinstellingen met het voorstel hen een eigen “ronde” aan te bieden waar alle leerlingen van een klas of zelfs de hele school aan kunnen deelnemen. Deze spelers zouden dan in een leeg universum starten en enkel met elkaar concurreren.

Het is de bedoeling dat Bazillionaire een realistische marktsituatie simuleert en we geloven dat het spel hierdoor ook “leerrijk” kan zijn. Deze piste moet echter nog verder uitgediept worden.

11 The maturing “massively popular” free to play MMO market. http://www.wappworks.com/2011/12/08/the-maturing-massively-popular-free-to-play-mmo-market/

10. Aanzet tot Technical Design document

Diagram architectuur



Keuze architectuur

De keuze voor PHP in een game lijkt misschien omstreden, maar CatLab heeft met Dolumar reeds bewezen dat het zeker niet onmogelijk is. Games als Travian, Tribal Wars, OGame en dergelijke hebben de sterkte van PHP zeker bewezen.

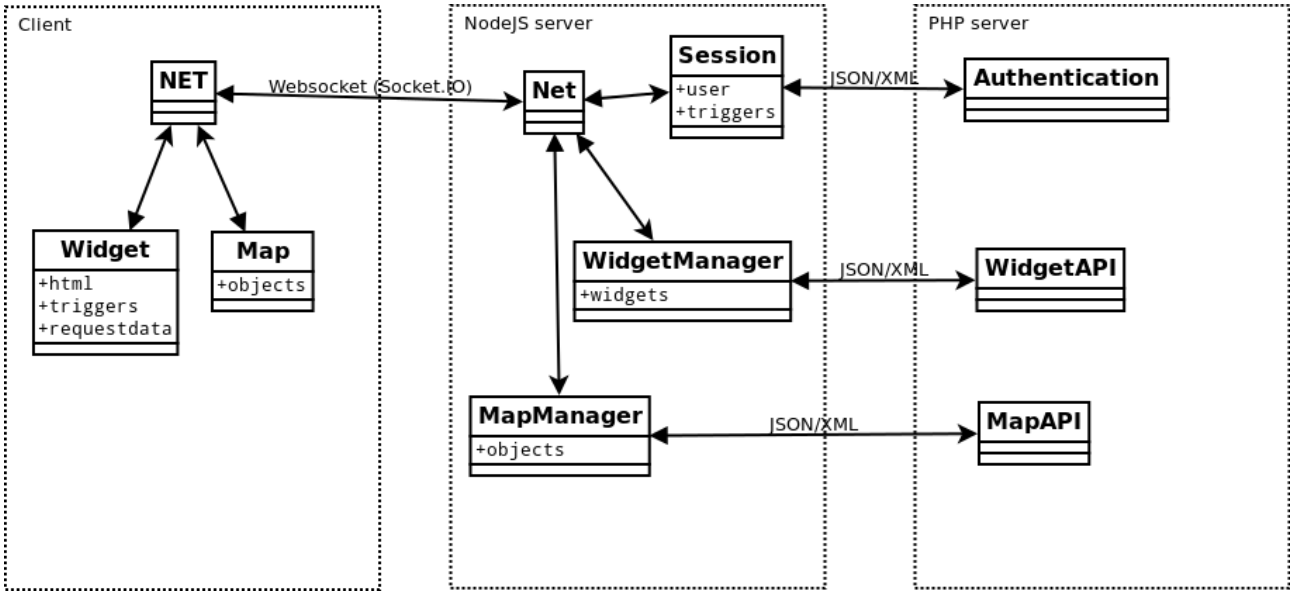
Voor het real-time aspect van het spel zijn we echter op zoek moeten gaan naar een nieuwe technologie. Daarom is er besloten een NodeJS server te plaatsen tussen de client en de server. Door middel van een observer structuur zorgt de NodeJS server ervoor dat de informatie bij alle spelers steeds up to date is.

In Bazillionaire nemen we een groot deel van de architectuur van Dolumar over. De GUI modules in het spel (“widgets”) worden op de server voorbereid en doorgestuurd naar de thin clients. Het enige wat de client moet doen is de HTML die als attribuut meegestuurd wordt, op het scherm tonen en de input van deze widgets terug doorsturen naar de server.

Dit laat ons toe het grootste deel van de game development op de server uit te voeren. Dankzij de thin client is het relatief eenvoudig nieuwe clients uit te brengen (Android, iOS, en – waarom niet – xbox360). En als het spel nieuwe features krijgt, moet daarvoor de client niet aangepast worden.

Dankzij deze widget structuur is het ook eenvoudiger mogelijk ons platform open te stellen voor andere developers. Het volstaat hier immers de 3rd party applicatie in te laden in een iframe (zie facebook applicaties). Browsers zijn standaard voldoende uitgerust om veiligheid in deze 3rd party apps te verzekeren door toegang tot de gamecode zelf te ontzien.

Onderaan vindt u een eenvoudige representatie van het klassendiagram. De NodeJS server wordt in dit diagram enkel gebruikt als “manager” die de aanvragen naar de eigenlijke game server (php) coördineert.



Schaalbaarheid

In dit ontwerp is meteen ook rekening gehouden met schaalbaarheid. We verwachten dat de bottleneck zich vooral bij de websockets zal voordoen. Daarom wordt de functionaliteit van de nodejs servers gelimiteerd en willen we de nodejs servers volledig zelfstandig laten werken. Zo zal het in de toekomst mogelijk zijn meerdere nodejs servers te laten draaien om zo de workload voldoende op te vangen.

Als, in een nog later stadium, ook de php server onder een grote werklading komt te staan, volstaat het ook daar horizontaal te schalen. Gezien de php servers “request based” draaien, maakt het voor de nodejs servers niet uit welke php server hun verzoek verwerken.

Eenmaal het aantal php servers groeit, blijft er nog één bottleneck over: de databank die alle informatie van het spel opslaat en die aangesproken wordt door de php servers. Maar daar biedt de grote schaalbaarheid van MySQL of een ander databanksysteem voldoende mogelijkheden.

Open source

Tijdens de ontwikkeling maken we gebruik van volgende open source pakketten:

- Voor de 3D interface maken we gebruik van Three.js, een library die zowel softwarematische- als hardwarematische visualisatie van 3D modellen biedt. Three.js heeft zijn kunnen bewezen in onder andere TriggerRally, een browser based racing game. http://mrdoob.github.com/three.js/
- Voor realtime communicatie tussen clients gebruiken we Socket.IO, ontwikkeld door Google. Socket.IO biedt voor oudere browsers een websocket implementatie in Flash. http://socket.io/
- Als framework voor de interface elementen in het spel gebruiken we jQuery en jQuery UI. Deze zijn beide voorbereid op gebruik in mobiele browsers en zijn sterk aanpasbaar waardoor we in een later stadium de *look & feel* van het spel kunnen bepalen.
- Aan de server zijde gebruiken we:
    - NodeJS voor realtime communicatie. http://nodejs.org/
    - Lighttpd als tussenlaag tussen PHP en NodeJS. (Performanter dan Apache) http://www.lighttpd.net/
    - PHP 5 voor implemenatie van de logische gameplay. http://php.net/
    - MySQL als databaselaag. http://www.mysql.com/

11. Art design document

In dit prototype willen we ons volledig op de technische kant van de zaak storten. Eenmaal het spel technisch in orde is – en een selecte groep beta testers hun mening hebben gegeven – zullen we de open source graphics vervangen door een eigen stijl.

Als referentie van naar wat we willen werken zijn hier echter enkele van de spelen waar we als kleine game developer erg veel respect voor hebben.

Een eerste test van hoe het spel er uit kan zien vindt u hieronder (of op http://tests.bazillionaire.eu/threejs-1/ – live rendering met WebGL).

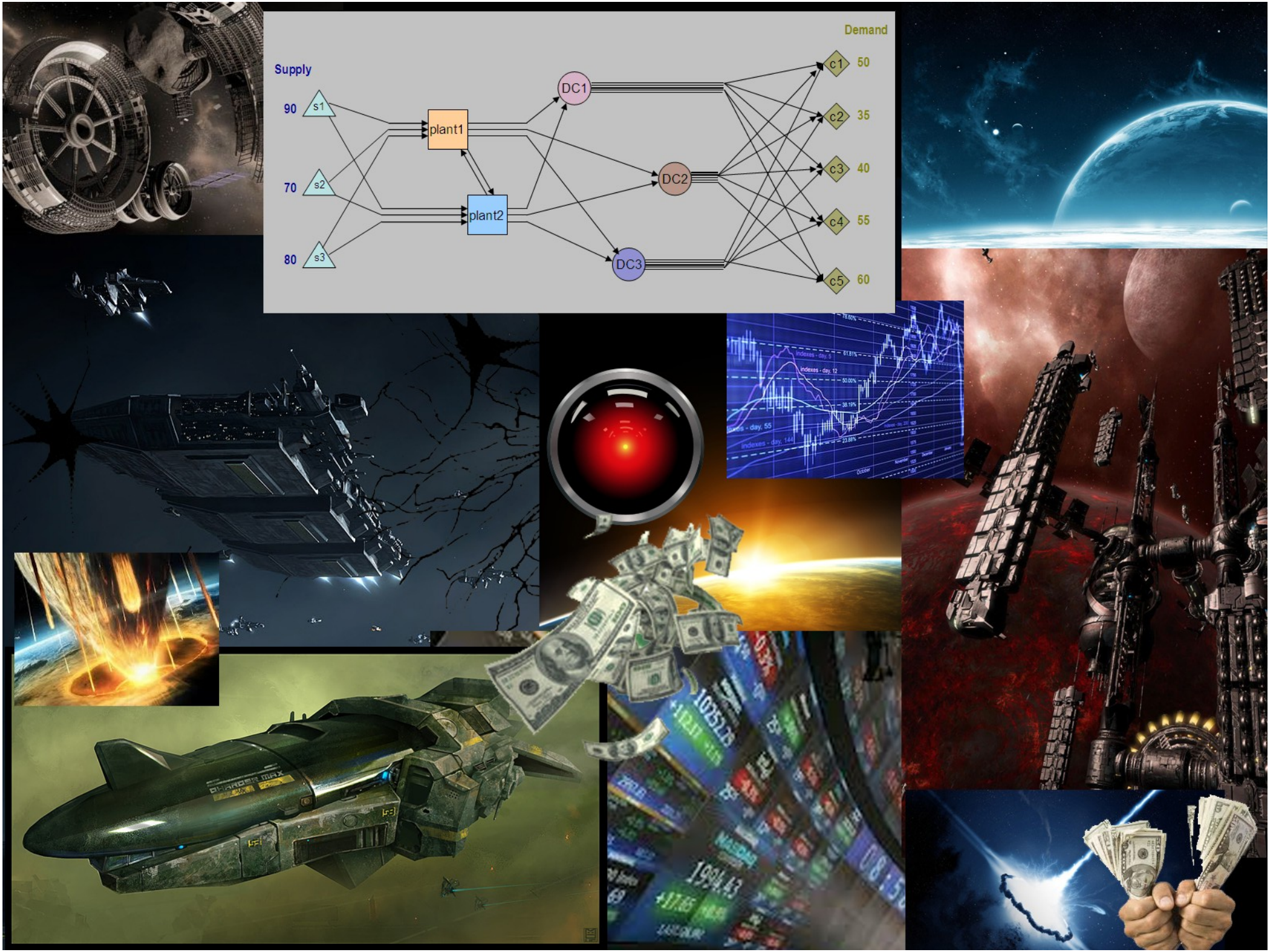
Afbeelding 4: Bazillionaire - engine test (open source graphics)

Afbeelding 5: Eve Online

Afbeelding 6: Freelancer

Afbeelding 7: Gazillionaire

Afbeelding 8: OGame


Supply
Demand
90
70
80
s1
s2
s3
plant1
plant2
DC1
DC2
DC3
c1
c2
c3
c4
c5
50
35
40
55
60

12. Personagebeschrijving

Doordat er niet echt een protagonist of zelfs een verhaallijn is, is een personagebeschrijving niet meteen belangrijk. De personages waarmee de speler te maken krijgt zullen hoofdzakelijk werknemers van CatCorp zijn (winkeluitbaters, verzekeringsadviseurs, belastingcontroleurs, …)

Een "guide" in de rechter hoek van het scherm (zoals we in Dolumar gedaan hebben) zal in "Japanse RPG-stijl" de speler wegwijs maken doorheen het spel en tips geven.

Wel willen we de *encounters* die de spelers zullen ontmoeten in zekere mate humoristisch maken. Zo zullen er in de conversaties referenties te vinden zijn naar oudere games en gebeurtenissen – waarbij we ook de internet memes aan beurt willen laten komen.

Maar een echte biografische beschrijving van de karakters is niet meteen van toepassing.