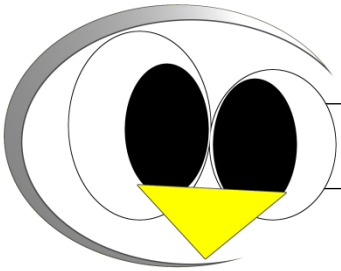
Frank van Smeden

Afstudeerverslag



3D renderer implementatie met OpenGL

Bijlagen

Academie voor ICT & Media

Bredewater 24 Zoetermeer

September 2010

Bijlagen

1. Afstudeerplan

Afstudeerplan

Informatie afstudeerder en gastbedrijf (*structuur niet wijzigen*)

Afstudeerblok:	2010-1.2
Startdatum:	26 april 2010
Einddatum:	24 september 2010
Inleverdatum:	24 september 2010
Studentnummer:	20064039
Achternaam:	dhr. Van Smeden
Voorletters:	F
Roepnaam:	Frank
Adres:	Lumeystraat 3
Postcode:	2722 AM
Woonplaats:	Zoetermeer
Telefoon:	079 342 66 14
Mobiel:	06 11 84 93 67
Opleiding:	Informatica
Locatie:	Zoetermeer
Variant:	voltijd
Naam studieloopbaanbegeleider:	Arno Nederend
Naam begeleider/examinator:	Arno Nederend
Naam expert/examinator:	Tim Cocx
Naam bedrijf:	Academie voor ICT & Media
Afdeling bedrijf:	
Bezoekadres bedrijf:	Bredewater 24
Postcode bezoekadres:	2715 CA
Postbusnummer:	
Postcode postbusnummer:	
Plaats:	Zoetermeer
Telefoon bedrijf:	079 320 87 87
Telefax bedrijf:	
Internetsite bedrijf:	hhs.nl
Achternaam opdrachtgever:	dhr. Broeren
Voorletters opdrachtgever:	*V.E.*
Titulatuur opdrachtgever:	BICT
Functie opdrachtgever:	Docent HHS
Doorkiesnummer opdrachtgever:	
Email opdrachtgever:	V.E.broeren@hhs.nl
Achternaam bedrijfsmentor:	dhr Broeren
Voorletters bedrijfsmentor:	V.E.
Titulatuur bedrijfsmentor:	BICT

Functie bedrijfsmentor: Docent HHS
Doorkiesnummer bedrijfsmentor:
Email bedrijfsmentor: V.E.broeren@hhs.nl

Doorkiesnummer afstudeerder:
Bedrijfsemail afstudeerder:
Functie afstudeerder (deeltijd/duaal):

Opdrachtomschrijving

Titel afstudeeropdracht: *OpenGL implementatie schrijven voor de renderer van Polemos*

1. Bedrijf

Dit project zal uitgevoerd worden op de Academie voor ICT & Media.

De academie voor ICT & Media te Zoetermeer is gestart in 2003 als onderdeel van de Haagse hogeschool. Deze afdeling werd gestart in Zoetermeer.

Op de academie worden op dit moment drie opleidingen gegeven;

- Informatica (I)
- Bedrijfskundige Informatica (BI)
- Information Security Management (ISM, sinds 2007)

Het aantal docenten wat les geeft op de totale academie (deze academiën zijn verdeelt over alle vestigingen van de Haagse Hogeschool) is 164. Daarbij zijn er 2260 studenten aangesloten bij de academie.

Waar de AICTM in Zoetermeer o.a. bekend om is geworden is hun aanbod van 3D-minors. Dit zijn projecten waarin studenten kunnen leren hoe ze een 3D engine kunnen opzetten. Dit is in Nederland uniek.

2. Aanleiding

De AICTM in Zoetermeer biedt al een aantal jaar een aantal minoren aan. Dit zijn projecten die studenten volgen als aanvulling op hun opleiding. De AICTM in Zoetermeer heeft een aantal eigen minoren ontwikkeld waarin studenten kunnen leren hoe een 3D-engine werkt. Hieruit is een project ontstaan dat wordt onderhouden door Dhr. Broeren (docent Haagse Hogeschool). Dit project is het (verder) ontwikkelen van een echte game, op een eigen 3D-engine. Aan dit project wordt regelmatig gewerkt door studenten, die graag als stageopdracht iets met de programmeertaal C++ (taal die gebruikt wordt in dit project), willen doen. De engine waarop de huidige game draait, en waaraan al jaren is door geprogrammeerd, wordt Polemos genoemd.

De volgende stap, die de eigenaar van Polemos graag wilt zien, is het cross-platform maken van de engine. Dit betekent dat de engine draaibaar moet zijn onder Windows, maar ook onder Linux of een ander operatingsystem. Op dit moment is dat niet mogelijk aangezien er alleen ondersteuning is voor DirectX binnen Polemos. DirectX is een verzameling van allerlei stukken code die met hardware kunnen communiceren binnen het Windows operating system. Zo kan Direct3D communiceren met de videokaart binnen een computer, en DirectSound communiceren met de geluidskaart van een computer.

Om crossplatform te worden is het dus nodig om geen DirectX te hoeven gebruiken, maar een aternatief hiervoor. Een veel gebruikt alternatief voor DirectX is OpenGL. OpenGL heeft qua functionaliteit hetzelfde doel als DirectX maar met het verschil dat OpenGL op meerdere platformen draait (Linux / macOS / Windows) en DirectX Windows® specifiek is.

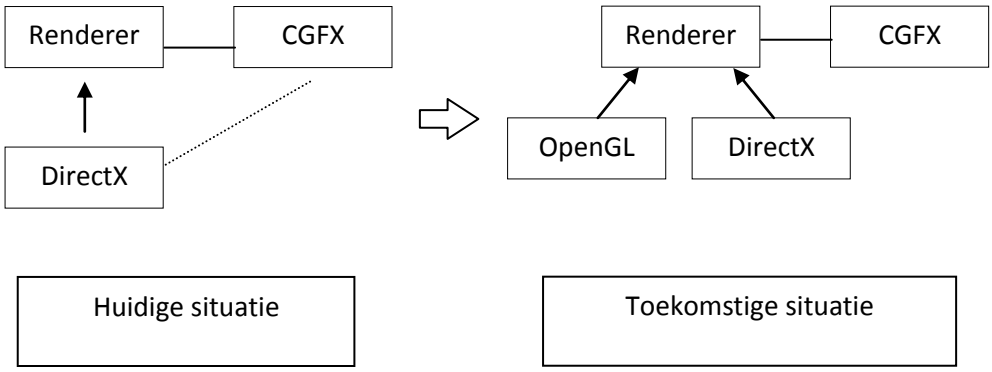
OpenGL is dus een alternatieve verzameling van interfaces ten opzichte van DirectX. Een uitbreiding naar de keuze om allebei deze interfaces te implementeren maakt Polemos dus meer cross-platform. Dit is een vooruitgang omdat een engine die meerdere, verschillende soorten, interfaces ondersteunt, flexibeler is dan een engine die alleen maar DirectX of OpenGL ondersteunt.

3. Probleemstelling

In de huidige staat is Polemos niet cross-platform. Polemos is namelijk geschreven voor het Windows® besturingssysteem en gebruikt dus DirectX. De reden waarom de Academie de 3D engine graag cross platform wilt hebben is ten behoeve van het onderwijs. Om studenten in aanraking te laten komen met OpenGL en om een deur open te zetten naar Linux zijn de hoofdredenen.

4. Doelstelling van de afstudeeropdracht

Het doel van dit project zal zijn om de Polemos engine zo aan te passen dat de renderer naast een DirectX interface, ook een OpenGL interface zal gaan krijgen. De renderer is het onderdeel van de Polemos engine die verantwoordelijk is voor het genereren van een afbeelding op basis van een driedimensionaal model.



Door het aanpassen van de renderer binnen Polemos, zullen er een aantal grote veranderingen binnen de engine plaats moeten gaan vinden. Zo zal de renderer abstract moeten worden gemaakt. Dit betekent dat er binnen de 3D-engine geen verbindingen mogen zijn naar DirectX behalve in de renderer van de engine. Deze renderer zal waarschijnlijk voor een groot deel al abstract zijn. Dit is echter nog nooit goed getest aangezien de enige verzameling van interfaces

die ooit deel heeft uitgemaakt van Polemos, DirectX is. Door het implementeren van een nieuwe interface wordt de engine gelijk goed getest op abstractie.

Zoals te zien is in de afbeelding hierboven, bestaat er in Polemos een verbinding tussen CGFX en DirectX. Dit is qua architectuur niet ideaal en hier zal dus opnieuw naar gekeken moeten worden. CGFX is een manier om shaders (instructies voor de renderer om bepaalde effecten te kunnen berekenen) te gebruiken in een 3D engine. Daarnaast is het de bedoeling dat de renderer dus twee implementaties krijgt (OpenGL en DirectX).

De hoeveelheid tijd die ik bezig zal zijn met het verbeteren van de renderer zal bepalend zijn voor het verloop van het project. Mocht de renderer weinig problemen opleveren en is deze al bij aanlevering van een hoge kwaliteit, dan zal ik meer tijd hebben voor het implementeren van andere soorten interfaces zoals SDL (alternatieve input manager) of X11 (alternatieve window-manager).

Naast het implementeren van een OpenGL renderer zal ik ook een adviesrapport schrijven over de hoeveelheid werk dat het waarschijnlijk zal gaan kosten geheel OpenGL te implementeren in Polemos.

Als gevolg van mijn aanpassingen aan de huidige renderer zal deze moeten worden gestest. Dit zal gebeuren met het softwarepakket CXX. Dit pakket kan unittests uitvoeren op c++ code.

5. Resultaat

Het voornaamste product in dit project is de nieuwe OpenGL renderer en de documentatie van de interface. Verder zal het resultaat voor de engine zijn dat deze flexibeler is geworden en dat er misschien door andere studenten later nog verder gebouwd gaat worden aan een OpenGL implementatie.

6. Uitgangssituatie aan de hand van:

a. Beschikbare noodzakelijke software (*indien van toepassing*)

- Visual Studio 2005
  - Standaard programmeer omgeving
- DirectX 9.0c (augustus 2009)
  - de API om DirectX games te maken
- OpenGL (Versie: zo dicht mogelijk bij directX 9.0c aansluitend)
  - de API om OpenGL games te maken

b. Beschikbare noodzakelijke hardware (*indien van toepassing*)

- PC met Windows en DirectX 9.0c
  - Deze versie van DirectX is waar het huidige Polemos project mee draait.
- Geschikte videokaart met shadermodel 3.0 (minimum)
  - Ik zal een PC in gebruik nemen die shadermodel 3.0 ondersteunt om de nodige shaders die in Polemos zijn verwerkt te kunnen gebruiken. Zodra ik een PC gebruik die geen shader model 3.0 heeft, zal ik dus niet alles op mijn beeldscherm krijgen wat er in de Polemos engine zit.

c. Reeds opgeleverde relevante documenten

- Documentatie van Polemos
  - Zolang er aan deze engine binnen de AICTM aan wordt gewerkt, wordt er ook documentatie voor geschreven. Deze zal voornamelijk bestaan uit documentatie in de broncode.
- Code-convention
  - Polemos heeft regels als het gaat om de syntax van de broncode.
- OpenGL-API
  - OpenGL heeft documentatie over hoe ik dit kan implementeren binnen een engine.
- DirectX-API
  - Polemos gebruikt nu DirectX. Ik zal me dus moeten gaan verdiepen in wat Polemos doet, hiervoor zal ik ook moeten bekijken hoe DirectX werkt.

d. Aanwezige ideeën:

- Abstractie test van huidige Polemos renderer

7. Werkzaamheden aan de hand van:

a. Te hanteren methodieken

- Scrum
  - Deze methode zal ik gebruiken om het project iteratief te voltooien. Deze methode is gedeeltelijk nieuw voor mij, ik zal me hier dus eerst meer in verdiepen voordat ik dit zal gaan toepassen binnen het project.

b. Uit te voeren activiteiten

1. Plan van Aanpak
   - In het begin van het project zal ik beginnen met het schrijven van een plan van aanpak. Hierin zal onder andere een planning komen te staan over het verloop van het project.

2. Opstellen requirements
    - Ik zal een requirements opstellen waarin al mijn werkzaamheden zijn opgenomen.
3. Onderzoek DirectX API
    - Ik zal me inlezen in hoe DirectX werkt. Hiervan zal echter geen product worden gemaakt, maar ik zal deze opgedane kennis moeten gebruiken bij het realiseren van de renderer. Veel van de kennis die ik van DirectX komt uit een minor die ik heb gevolgd aan de AICT.
4. Onderzoek OpenGL
    - Net als bij DirectX zal ik me moeten verdiepen in hoe OpenGL werkt. Ik zal hiervoor een aantal oefeningen (tutorials) uitvoeren die me inzicht zullen geven in hoe rendering plaatsvindt in OpenGL.
5. Scrum methode onderzoeken
    - Ik zal meer kennis moeten opdoen over hoe Scrum werkt en hoe dit in de praktijk moet worden toegepast.
6. Onderzoek verschillen DirectX en OpenGL
    - Om te bekijken wat de verschillen zijn tussen OpenGL en DirectX 9.0c zal ik hier onderzoek naar doen. Dit is van belang om meer kennis te verkrijgen van de huidige situatie van Polemos en ook van belang voor toekomstige studenten die verder willen bouwen aan Polemos in combinatie met OpenGL
7. Documentatie
    - Ik zal tijdens het schrijven van de implementatie van de OpenGL renderer documentatie schrijven binnen de broncode. Daarnaast zal ik me ook houden aan alle 'programmeer-regels' die zijn opgenomen in de code-convention
8. Ontwerp OpenGL renderer + aanpassingen engine
    - Ik zal voordat ik ga programmeren aan een OpenGL renderer, eerst diagrammen produceren die ik zal overleggen met mijn opdrachtgever voordat ik aan de implementatie begin.
9. Aanpassen Polemos renderer abstractie
    - Het realiseren van een abstract renderer voor zover deze nog niet aanwezig is.
10. Programmeren OpenGL renderer
    - Het bouwen van de OpenGL renderer.
11. Testen
    - Ik zal de nieuwe renderer gaan testen op functionaliteit. Dit zal ik doen door middel van een Unittest enzo de kwaliteit van de renderer te testen.
12. Advies hele engine naar Linux poorten.
    - Hoeveel werk het zal gaan kosten om geheel Polemos draaiende te krijgen onder Linux. Of dit rendabel is en hoeveel tijd dit waarschijnlijk zal gaan kosten zal ik gaan onderzoeken.

c. Te gebruiken technieken

- Visual Studio 2005
  - Programmeeromgeving
- UML
  - Techniek om diagrammen te produceren
- CXX test
  - Programma om C++ code te kunnen testen

d. Te vermelden nadrukken

Het project specificeert zich met nadruk op het implementeren van de abstracte renderer van Polemos

8. Risico's en maatregelen

- Risco: Te weinig kennis
  Maatregelen: Hiervoor zal ik me van tevoren al inlezen in DirectX en OpenGL.

  Daarnaast zal ik ook gedurende het project zo veel mogelijk informatie verzamelen.

- Risco: Onderschatting aanpassingen.
  Maatregelen: Het kan zijn dat de aanpassingen meer tijd in zullen nemen dan dat er van tevoren voor is ingeplant. In dat geval zal ik dit overleggen met mijn stagementor en met mijn stagebegeleider wat de maatregelen hierop zullen zijn.

9. Op te leveren (tussen)producten

- OpenGL implementatie van de Polemos Renderer
- Documentatie Polemos
- Rapport verschillen DirectX en OpenGL
- Scrum statistieken (update productbacklog)
- Adviesrapport port naar Linux
- Test rapport van unittest

Benodigde competenties

1.1 Selecteren methoden technieken en tools
1.2 Voorbereiden en opstarten softwareontwikkeltraject
1.4 Uitvoeren analyse door definitie van requirements
3.1 Ontwerpen softwarearchitectuur
3.2 Ontwerpen systeemdeel
3.3 Bouwen applicatie

Naar de beroepstaken van Academie voor ICT & Media Zoetermeer - april 2010

Te demonstreren competenties en wijze waarop

- 1.4 Uitvoeren analyse voor definitie van requirements
  - Plan van aanpak en functioneel ontwerp
- 3.1 Ontwerpen softwarearchitectuur
  - De nieuwe architectuur van Polemos
- 3.2 Ontwerpen systeemdeel
  - Het ontwerp van de nieuwe interface(s)
- 3.3 Bouwen applicatie
  - Het implementeren van een openGL renderer inclusief het testen en aanpasen van de huidige renderer

Persoonlijke verantwoording van keuze opdracht/bedrijf/competenties

Persoonlijk ben ik erg gericht op open-source software in combinatie met linux. Helaas zijn er weinig tot geen stageplaatsen waarin iets gedaan word met open-source software. Later hoorde ik toevallig dat een paar leraren op de academie wel geintereseerd waren in openGL (de grafische en open-source library die gebruikt word onder niet-Windows systemen) in combinatie met de huidige Polemos engine. In mijn opleiding heb ik al eerder een blok 3D-programmeren gevolgd met directX. Het lijkt me daarom erg leuk nu de andere kant van 3D-programmeren te zien en een open-source API te gebruiken.

Nominale duur afstudeerperiode

De stage zal in totaal 17 weken duren.

Activiteitenplanning

afstudeerweek	Activiteiten
1	• Afstemmen opdrachten met opdrachtgever • Inrichting werkomgeving • Installatie Polemos engine • Start Plan van aanpak
2	• Onderzoek directX API

	• Oplevering Plan van Aanpak
3	• Onderzoek verschil DirectX & OpenGL • Opstellen requirements • Begin Scrum-meetings
4	• Testen abstractie renderer • Onderzoek verschil DirectX & OpenGL (+ oplevering) • Oplevering requirements
5	• Begin implementatie renderer
6	• implementatie renderer
7	• implementatie renderer
8	• implementatie renderer
9	• implementatie renderer
10	• implementatie renderer • (implementatie eventuele andere interface)
11	• implementatie renderer • (implementatie eventuele andere interface)
12	• implementatie renderer • (implementatie eventuele andere interface)
13	• Testen OpenGL renderer
14	• Schrijven Afstudeerverslag
15	• Schrijven Afstudeerverslag • Oplevering documentatie renderer • Oplevering adviesrapport poort naar Linux • Oplevering tests abstractie niveau Polemos • Oplevering productbacklog
16	Uitloop
17	Uitloop

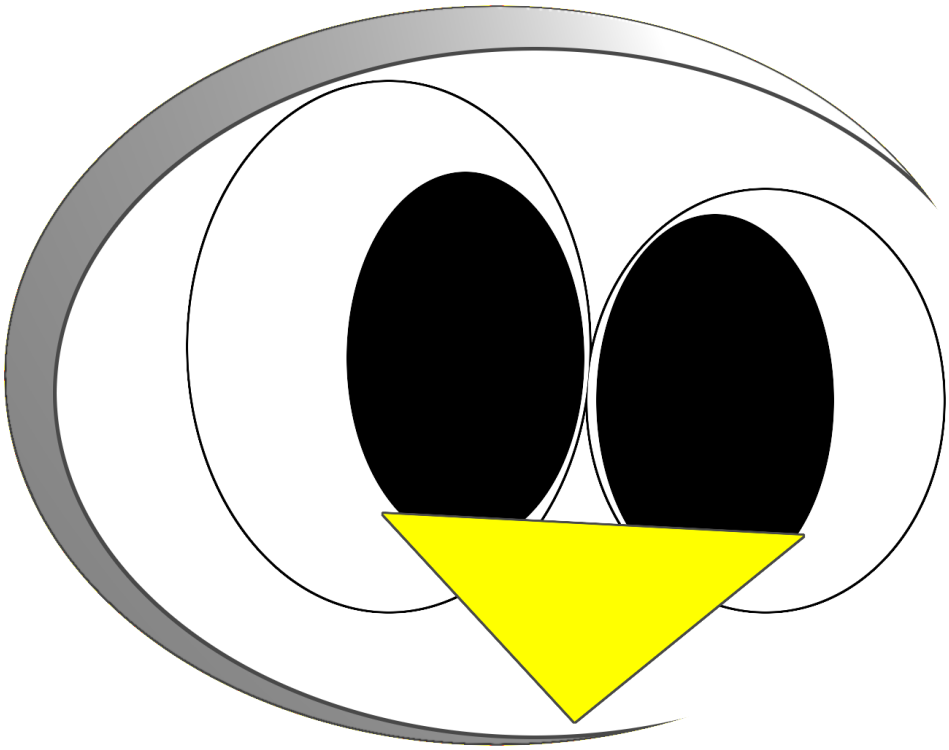
Naam opdrachtgever:
Handtekening voor akkoord:
Datum:

Naam bedrijfsmentor:
Handtekening voor akkoord
Datum:

Naam begeleider/examinator:
Handtekening voor akkoord:
Datum:

Naam expert/examinator:
Handtekening voor akkoord:
Datum:

2. Plan van aanpak



Plan van aanpak

- Frank van Smeden
- Informatica
- 20064039

Academie voor ICT & Media

Bredewater 24 Zoetermeer

Documentbeheer ~ Plan van aanpak

Versie	Datum	Aanpassing
0.1	3 – mei	Hoofdstukken aangemaakt / voorkant geïmporteerd / rapportage toegevoegd
0.2	4 - mei	Inleiding toegevoegd / benodigde middelen toegevoegd / projectorganisatie toegevoegd / randvoorwaarden toegevoegd / risico analyse /
0.3	6 - mei	Planning toegevoegd / Opdrachtomschrijving toegevoegd / inleiding aangepast
0.4	7 - mei	Planning aangepast na overleg opdrachtgever
0.5	19 - mei	Mijlpalen toegevoegd
0.6*	20 - mei	Scrum methodiek toegevoegd

Inhoudsopgave

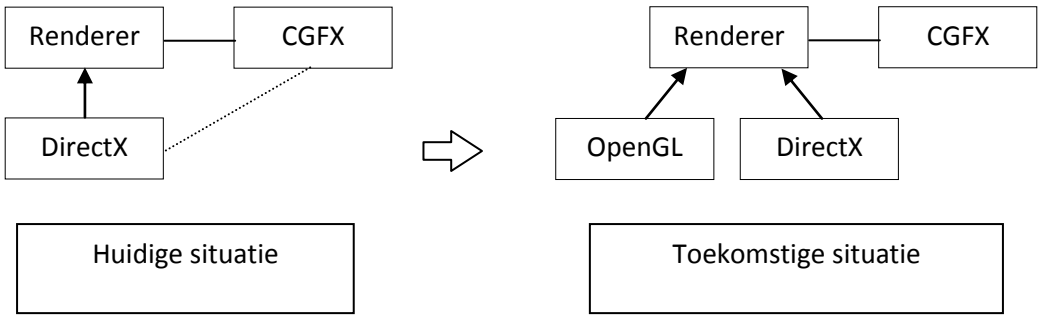
Inleiding

Dit document beschrijft de aanpak van het afstudeerproject van Frank van Smeden aan de Academie van ICT & Media. Er zal te lezen zijn wat mijn project inhoud in de opdrachtomschrijving (hoofdstuk 1). Wat voor middelen ik in mijn project denk te gaan gebruiken staat in hoofdstuk 3. Een globale planning is opgenomen in hoofdstuk 7.

Dit document is geschreven voor de opdrachtgever Dhr. Broeren maar zal ook door mijn stagebegeleiders worden gelezen.

1. Opdrachtomschrijving

Het doel van dit project zal zijn om de Polemos engine (3D engine die wordt gebruikt op de AICTM) zo aan te passen dat de renderer naast een DirectX interface, ook een OpenGL interface zal gaan krijgen.



Zodra de renderer binnen Polemos zal worden aangepast zal dit een effect voor de hele engine hebben. Zo zal eerst de renderer abstract moeten worden gemaakt aangezien de huidige renderer dit niet geheel is.

Naast het implementeren van een OpenGL renderer zal er een advies moeten komen waarin een analyse staat hoeveel werk/kosten het zou vergen om de gehele engine te poorten naar Linux.

2. Rapportage

Gedurende dit project zal ik verschillende producten opleveren. De manier waarop ik dit rapporteer zal per mijlpanel verschillen.

Als de betreffende mijlpanel een document betreft zal deze via de mail worden opgeleverd, mits anders door de opdrachtgever wordt verzocht.

Als er fysieke code moet worden opgeleverd, zal dit gaan via XP-dev. Dit is een software development-tool om een project te kunnen monitoren en code te kunnen delen. Deze tool werkt met het softwarepakket SVN en kan worden benaderd:

http://svn2.xp-dev.com/svn/Polemos_OpenGL/

In dit project zal er gebruik gemaakt worden van Scrum. Bij deze methode hoort een gesprek plaats te vinden een keer in de twee weken. In de eerste 4 weken van dit project zal dit meer zijn om wat extra sturing aan het project te geven.

Daarna zullen er zogenaamde sprints worden ingepland. Deze sprints zijn twee weken van werk waarin bepaalde taken uit de productbacklog worden uitgevoerd. De productbacklog is een lijst met de volgende gegevens:

- De uit te voeren taak
- De ingeschatte tijd hoeveel dit waarschijnlijk gaat kosten uitgedrukt in ‘units’ (10 minuten)
- De prioriteit, variërend van 1 tot 10. Er kan echter ook een prioriteit 99 aan worden toegekend, op dat moment moet dit in ieder geval worden uitgevoerd.

3. Benodigde middelen

Software:

- Microsoft Word 2007 - documentatie e.d.
- Visual Studio 2005 - development OpenGL
- Nvidia CGFX - development OpenGL
- DirectX 9.0c SDK (march 2009) - development DirectX renderer
- CXXtest - test omgeving

Hardware:

- PC
  - DirectX 9.0c compatible
  - Shader model 3.0 compatible
- Werkomgeving

4. Project organisatie

Doordat dit een individueel project is zullen alle taken voor dezelfde persoon zijn.

- Programmeur - Frank
- Tester - Frank
- Software architect - Frank

5. Mijlpalen

- Plan van Aanpak
  - Dit document is ook een mijlpaal
- Opstellen requirements voor product backlog (zie hoofdstuk methodiek voor meer informatie)
  - Wat er allemaal aan de engine aangepast moet worden
- Onderzoek verschillen Direct3D en OpenGL
  - Verslag over de verschillen en overeenkomsten tussen Direct3D en OpenGL
- Uitvoeren Unittests
  - Unittests om ervoor te zorgen dat de Direct3D API geen schade ondervind van mijn OpenGL implementatie
- Ontwerp OpenGL renderer + aanpassingen engine
  - Ontwerpen van de nieuwe implementatie
- Aanpassen Polemos renderer abstractie
  - Aanpassingen aan de huidige situatie om OpenGL te kunnen draaien
- De implementatie van de OpenGL renderer
  - De OpenGL renderer in code
- Advies hele engine naar Linux poorten.
  - Advies rapport over Polemos draaiende te krijgen onder Linux

6. Methodiek

Tijdens dit project zal er gebruik gemaakt gaan worden van Agile. Agile is een verzamelnaam van allerlei methodieken die zich richten op software ontwikkeling. Agile heeft eigenlijk als basis de volgende principes:

Liever…	dan…
Individuele personen en communicatie	dan processen en tools.
Werkende software	begrijpelijke documentatie
Klant die meewerkt aan het project	contract onderhandelingen
Reageren op verandering	het volgen van een ontwerp

Elke methodiek binnen Agile heeft een andere doelstelling. De methodiek waarop ik me zal gaan richten is die van Scrum.

Scrum is een iteratief Agile methodiek waarbij de nadruk valt op het managen van een project.

Categorieën en rollen

In Scrum zijn er twee soorten categorieën aan personen binnen een project, Pigs en Chickens. De reden waarom deze twee categorieën zo heten komt door de volgende grap die goed aantoont wat het verschil is tussen deze twee:

"A pig and a chicken are walking down a road. The chicken looks at the pig and says, "Hey, why don't we open a restaurant?" The pig looks back at the chicken and says, "Good idea, what do you want to call it?" The chicken thinks about it and says, "Why don't we call it 'Ham and Eggs'?" "I don't think so," says the pig, "I'd be committed, but you'd only be involved." "

Pigs zijn dus mensen die te maken hebben met het project en er ook meer voor over hebben dan de chickens. Zij zijn ook degene die aan het project werken met een actieve houding. Chickens zijn de mensen die wel degelijk belang hebben bij het project, maar zijn niet zo toegewijd aan het project als de pigs en hebben minder op het spel staan.

Verder worden er nog bepaalde rollen toegekend aan personen die met het project te maken hebben:

- ScrumMaster (pig)
  - Een ScrumMaster is projectlid dat het project in goede banen leidt. Dit betekend praktisch dat deze ervoor zorgt dat het team bezig is en kan zijn met de taken van de huidige sprint. Een ScrumMaster is echter geen leider, binnen scrum bestaat deze niet maar leidt het gehele team.
- Scrum Team (pig)
  - Het Scrum team bestaat uit de daadwerkelijke ontwikkelaars. Deze zijn uiteraard allemaal nauw betrokken bij het project en zijn dus allemaal pig.
- Product owner (pig)

    - Dit is de klant van het project. Voor hem zal het ontwikkelde product gemaakt worden. Scrum streeft ernaar dat deze Product owner nauw samenwerkt met het scrum team. Het kan echter ook zo zijn dat de product owner niet zo betrokken is binnen het project en kan worden genoemd als chicken.
- Stakeholders (chicken)
    - Deze mensen laten het project bestaan door middelen te verschaffen maar zijn in de project ontwikkeling niet nauw betrokken.
- Managers (chicken)
    - Managers zijn alleen betrokken bij het project voor het opzetten van de omgeving voor het Scrum team.

Producten

Scrum werkt met een aantal producten. Deze producten worden continue bijgewerkt. Sprint producten (burndown chart en sprint backlog) worden per sprint bewaard en opgeslagen.

- Product backlog
    - Dit is een sheet met alle ontwikkelingen die nog in het project plaats moeten gaan vinden. Per item worden de een paar gegevens geregistreerd.
        - Het nummer van het Item, ieder item heeft een nummer waarnaar gerefereerd kan worden.
        - Ingeschatte tijd. Hier word geregistreerd hoelang het team nodig denkt te hebben om dit item te ontwikkelen in units. Een unit staat voor tien minuten
        - Omschrijving. Hier staat wat er ontwikkeld moet worden.
        - Prioriteit. De urgentie van het item kan hierin worden opgenomen. Deze kan variëren van 1 tot en met 4
- Sprint backlog
    - Voorafgaande aan een sprint wordt er een sprint backlog opgesteld. Hierin staan de items uit de product backlog die van toepassing zijn voor de aankomende sprint.
- Burndown chart
    - Een burndown chart is een grafiek die de voortgang van de sprint laat zien.

Meetings

Een van de grote verschillen met andere methodieken, zoals RUP of IAD, zijn de meetings. In dit project zal ik gebruik van een aantal soorten meetings. De sprint-meetings zal ik houden samen met mijn opdrachtgever. Deze zullen een keer in de twee weken zijn aangezien één sprint twee weken zal beslaan. Daarnaast zal ik iedere week een meeting hebben met mijn opdrachtgever waarin ik een demonstratie zal geven waarin alles wat werkt getoond zal worden.

Later in dit project zullen er andere studenten bijkomen die ook aan Polemos zullen werken. Vanaf dat moment zullen er ook daily-meetings gehouden worden om de voortgang te bewaken en goede afspraken te kunnen maken. Daily-meetings worden (bijna) iedere dag gehouden met in ieder geval alle pigs.

7. Randvoorwaarden

Om dit project goed te laten verlopen moeten er voldoen worden aan de volgende voorwaarden. Hiervan kan alleen worden afgeweken in overleg met de opdrachtgever.

- Het project moet worden gebouwd in versie 2005 van het softwarepakket Visual Studio. Dit om zoveel mogelijke schade van de installatie op een nieuw systeem te kunnen beperken.
- Het projectlid zal iedere dag aanwezig zijn mits anders is overeengekomen met de opdrachtgever.
- Er zal minstens eens in de twee weken een gesprek met de opdrachtgever zijn. Dit om de voortgang van het project te bewaken

8. Risico analyse

- Risco: Te weinig kennis

  Maatregelen: Hiervoor zal ik me van tevoren al inlezen in DirectX en OpenGL.

  Daarnaast zal ik ook gedurende het project zo veel mogelijk informatie verzamelen door tutorials over OpenGL en DirectX te maken.

- Risco: Onderschatting aanpassingen.

  Maatregelen: Het kan zijn dat de aanpassingen meer tijd in zullen nemen dan dat er van tevoren voor is ingepland. In dat geval zal ik dit overleggen met mijn stagementor en met mijn stagebegeleider wat de maatregelen hierop zullen zijn.

- Risco: Computer crash

  Maatregelen: Het kan gebeuren dat mijn werk-PC het begeeft door een virus of ander probleem. In dat geval zal dit gelijk gerapporteerd moeten worden aan de opdrachtgever. Om dit probleem te beperken zal ik iedere twee weken een backup maken van alle gegevens.

- Risco: source-code management problemen

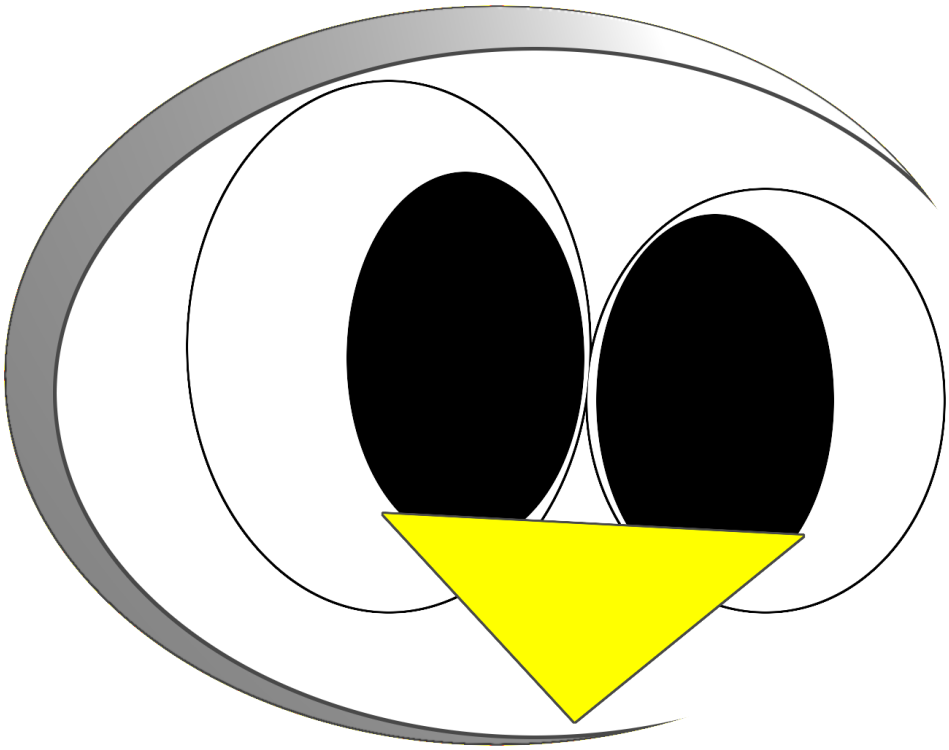
  Maatregelen: Later in mijn afstudeer project zullen ook andere studenten aan het werk gaan met Polemos. Wat erop neerkomt dat we in dezelfde source-code gaan werken. Dit brengt het risico met zich mee dat we in dezelfde bestanden code gaan wijzigen. Dit de engine kapot maken. Om dit te voorkomen zullen we daily -meetings houden en hier afspraken over gaan maken. Daarnaast zal ik unittest schrijven die ervoor moeten zorgen dat er geen corrupte code in de source-code komen.

9. Globale Planning

Hier staat een globale planning zoals opgesteld in week 2. Hiervan zal alleen worden afgeweken met toestemming van de opdrachtgever.

afstudeerweek	Activiteiten
1	• Afstemmen opdrachten met opdrachtgever • Inrichting werkomgeving • Installatie Polemos engine • Start Plan van aanpak
2	• Onderzoek directX API • Oplevering Plan van Aanpak
3	• Onderzoek verschil DirectX & OpenGL • Opstellen requirements (o.a.voor product backlog)
4	• Onderzoek verschil DirectX & OpenGL (+ oplevering) • Verdieping scrum/agile • Tutorials OpenGL • Oplevering requirements
5	• Begin Scrum-meetings • Testen abstractie renderer • Begin implementatie renderer
6	• implementatie renderer
7	• implementatie renderer
8	• implementatie renderer
9	• implementatie renderer
10	• implementatie renderer • (implementatie eventuele andere interface)
11	• implementatie renderer • (implementatie eventuele andere interface)
12	• implementatie renderer • (implementatie eventuele andere interface)
13	• Testen OpenGL renderer
14	• Schrijven Afstudeerverslag
15	• Schrijven Afstudeerverslag • Oplevering documentatie renderer • Oplevering adviesrapport poort naar Linux • Oplevering tests abstractie niveau Polemos • Oplevering productbacklog
16	Uitloop
17	Uitloop

3. Onderzoek Verschillen Direct3D en OpenGL



Onderzoek Direct3D en OpenGL

- Frank van Smeden
- Informatica
- 20064039

Academie voor ICT & Media

Bredewater 24 Zoetermeer

Documentbeheer ~ Onderzoek Direct3D en OpenGL

Versie	Datum	Aanpassing
0.1	11 – mei	Hoofdstukken aangemaakt / voorkant geïmporteerd
0.2	12 - mei	'OpenGL nader bekeken' hoofdstuk gemaakt / 'Direct3D nader bekeken' hoofdstuk gemaakt.
0.3	17 – mei	Direct3D nader bekeken / hoofdstuknummers toegevoegd
0.4	18 – mei	'OpenGL nader bekeken' verbeterd / 'Direct3D nader bekeken' verbeterd / opmaak
0. 5*	19 – mei	Huidige situatie Direct3D
0.6	19 –mei	Voettekst toegevoegd
0.7	20- mei	Framework aangepast / verbeterpunten opdrachtgever verwerken van versie 0.5
0.8	21 -mei	Technische verschillen bijgewerkt / Doelgroep veranderd
0.9*	24 - mei	Typefouten verbeterd
1.0	31 augustus	ProjectMatrix / modelviewProjectionMatrix toegevoegd

* = opgeleverd

Inhoudsopgave

Inleiding

In dit document wordt ingegaan op Direct3D en OpenGL, de API's die videokaart aansturen. Per API wordt uitgelegd hoe de API zich in de loop der tijd heeft ontwikkeld en hoe de API er op dit moment uitziet.

Ook is er per API een overzicht met de belangrijkste versies en datum van release. De grootste verschillen zijn te vinden in hoofdstuk 3.

1. OpenGL nader bekeken

Achtergrond

De geschiedenis van OpenGL begint in het jaar 1982. Een bedrijf dat gespecialiseerd is in het leveren van computerapparatuur brengt een product op de markt dat gebruik maakt van grafische computermogelijkheden. Dit bedrijf is gevestigd in Silicon Valley en bekend onder de naam Silicon Graphics. Dit product waar Silicon Graphics bekend mee is geworden is IRIS GL (de GL staat hierbij voor Graphics library). Dit is een grafische Applicatie interface voor programmeurs (API) om het mogelijk te maken om 2D en 3D beelden te genereren op de IRIS graphical Workstation (werkstations geproduceerd door Silicon Graphics). Deze API was een groot succes en werd flink uitgebreid door de jaren heen. Na een aantal jaren van ontwikkeling, slecht onderhoud en slecht programmeerwerk besloot Silicon Graphics de API opnieuw te schrijven. Daarnaast nam Silicon Graphics nog een belangrijke beslissing. De nieuwe API moest open-source worden. Dit betekend dat de broncode die geprogrammeerd wordt door Silicon Graphics, geheel beschikbaar moet zijn voor iedereen die het wil gebruiken. Dit was een grote verandering, IRIS GL was namelijk betaalde software en alleen Silicon Graphics had de broncode. Bij deze grote veranderingen is ook een nieuwe naam gekozen, namelijk Open GL, Open Graphics Library.

Huidige situatie

Op dit moment wordt OpenGL beheerd door de Khronos Group. De Khronos Group is een organisatie die is samengesteld uit allerlei bedrijven die interesse hebben in de voortgang van open standaarden. Dit omdat ze een bepaalde versie van OpenGL gebruiken in hun software, of omdat hun software communiceert met software die OpenGL gebruikt. Een aantal bedrijven hiervan zijn:

Het logo van de Khronos Group

- AMD/ATI
- Id Software
- Intel Corporation
- Mozilla
- Sun Microsystems
- Apple Inc
- Google
- Nvidia
- Nokia

De Khronos Group is opgericht in 2000, en heeft ongeveer 30 bedrijven die het project sponsoren. Voorzitter van deze groep is Neil Trevett, afkomstig van Nvidia. OpenGL is een van de software pakketten die onder Khronos Group vallen. Verder worden er nog meer beslissingen genomen door deze organisatie over andere multimediastandaarden

Versies

Versie	Release	Aanpassing
OpenGL 1.0	January 1992	• Implementation of IRIS GL
OpenGL 1.1	January 1997	• Textures • Texture-formats
OpenGL 1.2	Maart 1998.	• Packed pixels • Normal rescaling • Clamped/edge texture sampling • Image processing
OpenGL 1.2.1	Oktober 1998	• Multi-texture
OpenGL 1.3	Augustus 2001	• cubemap texture • Multi-texturing • Multi-sampling • Texture unit combine operations
OpenGL 1.4	July 2002	• Hardware shadowing support • Fog coordinates • Automatic mipmap generation • Additional texture modes.
OpenGL 1.5	July 2003	• vertex buffer objects (VBOs) • Occlusion queries • Extended shadowing functions.
OpenGL 2.0	September 2004	• GPU-based assembly language (ARB) • C-style shading language support (GLSL / Slang)
OpenGL 2.1	July 2006	• Pixel buffer objects (PBOs) • sRGB textures • Non-square matrices • GLSL 1.20
OpenGL 3.0	July 2008	• Frame buffer objects • Hardware instancing • vertex array objects (VAOs) • sRGB framebuffers
OpenGL 3.1	Maart 2009	• GLSL 1.40 • Texture Buffer Objects • Uniform Buffer Objects • Signed normalized textures • Instancing • CopyBuffer
OpenGL 3.2	Augustus 2009	• GLSL 1.50 • Geometry Shader support • BGRA vertex ordering • Seamless cube map filtering

		• Fragment depth clamping • Sync and Fence objects
OpenGL 3.3	Maart 2010	• Instanced geometry shaders • Instanced arrays • 64-bit double precision floating point shader operations • Texture state and texture data separation • Shader subroutines • GLSL 3.30
OpenGL 4.0	Maart 2010	• Instanced geometry shaders • Instanced arrays • 64-bit double precision floating point shader operations • Shader Model 4.0 support • drawing of data generated by OpenGL, or external APIs such as OpenCL, without CPU intervention; • Texture state and texture data separation • per-sample fragment shaders and programmable fragment shader input positions for increased rendering quality and anti-aliasing flexibility;

Zoals te zien is, verschillen de aanpassingen van OpenGL 3.3 en 4.0 weinig. Dit komt doordat DirectX 10 geschikte hardware in ieder geval ook OpenGL 3.3 kan draaien. Zo komen deze nieuwe features dus ook vrij voor videokaarten die DirectX 10 ondersteunen. OpenGL 4.0 wordt alleen ondersteund door high-end videokaarten die ook DirextX11 kunnen draaien.

2. Direct3D nader bekeken

Achtergrond

In 1992 startte Servan Keondjian het bedrijf RenderMorphics. Dit bedrijf specialiseerde zich in het gebruik van 3D graphics door middel van het maken van een eigen API (Reality Lab). In 1995 kocht Microsoft RenderMorphics op om een 3D engine te implementeren in Windows 95. Dit resulteerde in een nieuwe API genaamd Direct3D.

In de eerste jaren van ontwikkeling stond Direct3D slecht bekend onder programmeurs, er zaten veel bugs in en de mogelijkheden waren veel minder ten opzichte van OpenGL. Dit is eigenlijk veranderd sinds de komst van Direct3D 7. Toen was de API stabiel genoeg en een stuk sneller, veel meer games gingen over naar Direct3D. Daarna is Direct3d onder leiding van Microsoft qua aanhang gegroeid.

Huidige situatie

Op dit moment is de nieuwste versie van Direct3D versie 11. Deze versie wordt standaard geleverd bij het Windows 7 besturingssysteem. Sinds Direct3D 10 (voor het eerst gelanceerd in Windows VISTA) is Direct3D niet meer backward compatibel. Dit houdt in dat het mogelijk is dat als games gebruik maken van een API voor Direct3D 10, onder Direct3D 10 kunnen crashen omdat de API niet meer overeenkomt. Voor de uitkomst van Direct3D 10 was de backward compatibility nog niet 'gebroken' en konden dus alle games die tot dan toe met Direct3D gemaakt waren gedraaid worden met Direct3D 9.

De reden waarom Microsoft de backward compatibility brak, bij de overgang van Direct3D 9 naar Direct3D 10, was snelheid. Tijdens het ontwikkelen van een API worden steeds nieuwe technieken bedacht om de afhandeling van broncode sneller te laten verlopen. Maar om backward-compatibility niet te verbreken mogen sommige delen van de API niet worden aangepast. Dit omdat anders een stuk software vast loopt omdat de API niet meer klopt met de hoe deze werd aangeroepen in een vorige versie van de API. Het behouden van deze vorm van bereikbaarheid kost echter wel snelheid. Maar om toch backward compatible te kunnen blijven heeft Microsoft Windows 7 uitgerust met twee versies van DirectX. Een die wel compatible is met Direct3D 9 en een de nieuwste DirectX 11

Versies:

Versie	Release	Aanpassing
Direct3D 5.0	Juli 1997	
Direct3D 6.0	Augustus 1998	• Multitexture • Stencil buffers • Geometry pipelines • Optional texture management • S3 texture compression (DXTC)
Direct3D 7.0	September 1999	• .dds format support • Transform hardware acceleration (alleen Nvidia-kaarten)

		• Lighting hardware acceleration (alleen Nvidia-kaarten) • Multitexturing hardware
Direct3D 8.0	November 2000	• vertex shaders • pixel shaders • DirectDraw • texture mapping • bump mapping • fog
Direct3D 9.0	December 2002	• High Level Shader Language • Direct3D 9Ex (alleen VISTA)
Direct3D 10	November 2006	• Shader Model 4.0 • geometry shaders • Fully programmable pipelines • Texture array • Instancing 2.0
Direct3D 10.1	Februari 2008	• Mandatory 32-bit floating point filtering • Mandatory support for 4X anti-aliasing • Shader model 4.1
Direct3D 11	Oktober 2009	• Compute shaders • Tessellation • Multithreaded rendering

3. Verschillen en overeenkomsten

Architecturale verschillen

Open Source

Direct3D is een proprietary API. Dit betekent dat niemand de broncode mag zien of aanpassen. OpenGL verschilt hiervan met Direct3D. OpenGL is een open standaard, waarvan iedereen de broncode mag zien en naar eigen smaak mag aanpassen.

2D / 3D ondersteuning

Direct3D is door Microsoft gespitst op het renderen van 3D beelden. Daarom is er een aparte API voor het renderen van 2D beelden. Deze andere API heet DirectDraw. OpenGL maakt dit onderscheid niet en kan, net zoals 3D beelden, ook 2D beelden renderen.

Platform afhankelijkheid

Hier verschillen Direct3D en OpenGL erg met elkaar. Direct3D valt onder Windows, daarom kan Direct3D alleen werken onder het Windows besturingssysteem (zoals bijvoorbeeld Sega's Dreamcast of de XBOX/ XBOX 360). OpenGL daarentegen is crossplatform. Het draait onder zowel Linux als Windows als MacOS. Hierdoor heeft OpenGL een veel groter draagvlak en word het ook veel vaker gebruikt op mobiele platformen zoals de iPhone en het besturingssysteem van Google (Android)

Doelgroep

De doelgroep van de twee API's verschillen ook met elkaar. Zo is Direct3D veel meer gemaakt voor de gaming-industrie op Windows platformen.

OpenGL beslaat een veel bredere doelgroep. Vooral bij CAD software wordt OpenGL vaak gebruikt. CAD staat voor Computer-Aided Design. Dit is een naam voor een software pakket dat de gebruiker de mogelijkheid geeft te kunnen ontwerpen en te tekenen, ongedacht welke dimensie (2D of 3D). Vaak wordt CAD in combinatie gebruikt met Computer-aided Manufacturing (CAM). Dit is software die hardware kan aansturen. Dit gebeurd door middel van een door CAD-sofware gemaakt model dat ingeladen wordt. Naast deze vorm van toepassingen van OpenGL wordt OpenGL ook gebruikt bij het ontwikkelen van games. Hieronder staan een paar populaire games die gebruik maken van OpenGL:

- Counter-Strike
- Wolfenstein
- Enemy Territory: Quake Wars
- Unreal Tournament 2004
- Stronghold

Backwards compatibility

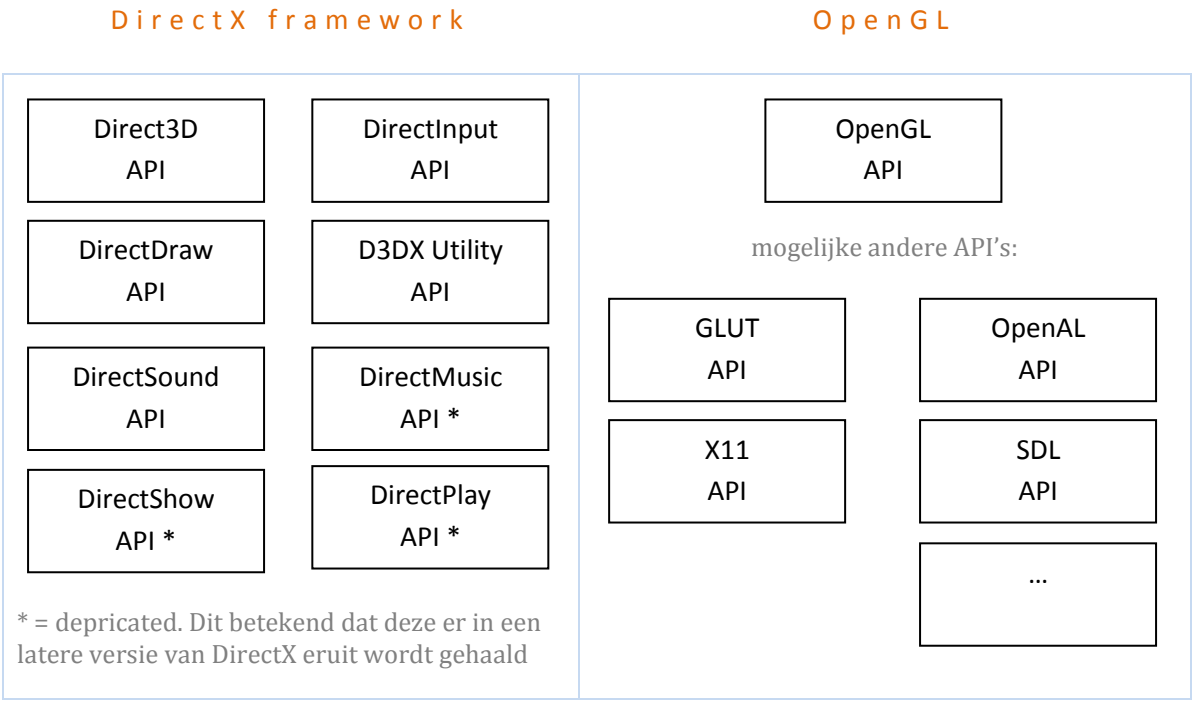
Sinds Direct3D 10 is DirectX niet meer backward compatibel. Voordat Direct3D 10 werd uitgebracht kon elk stuk software dat gebruik maakte van Direct3D gebruik maken van Direc3D 9. Dus ook software die

de API van Direct3D 6 aanriep kon gewoon functioneren met Direct3D 9. Deze vorm van backward-compatibility heeft Microsoft niet meer na DirectX 9. Dit is gedaan om performance redenen.

OpenGL is, tot op de dag van vandaag, wel backward compatibel. Dit gaat ten koste van de snelheid van de API.

Framework

Direct3D maakt gebruik van een framework (DirectX). In dit framework zitten nog andere API's zoals DirectSound en DirectInput. OpenGL heeft geen framework, en er zullen dus altijd meerdere pakketten nog zijn om een engine te creëren die beelden, Music en input ondersteunt.



Dit heeft voordelen en nadelen. Het voordeel van een heel framework is dat alles gemakkelijk met elkaar compatible is, omdat dezelfde makers erachter zitten. En nadeel is echter dat je afhankelijk bent van het framework. Een component aanpassen van de engine kan verstrekkende gevolgen hebben. En zodra er API's worden aangepast of uitgehaald worden kan het voorkomen dat de engine kapot gaat. Bij het niet hebben van een framework bestaat de hele engine uit losse onderdelen die los van elkaar kunnen worden vervangen of worden bijgewerkt. Op deze manier is de engine veel flexibeler bij grote veranderingen.

D3DX , GLUT en openGLUT

D3DX (Direct3D extension), GLUT (OpenGL Utility toolkit) en openGLUT (open OpenGL Utility toolkit) zijn API's die het werk voor de programmeurs makkelijker maken. Deze API's bevatten allerlei functies die de programmeur anders zelf moet schrijven. Deze extra API's zijn dus nooit noodzakelijk voor het creëren van een engine, maar maken het schrijven ervan een stuk gemakkelijker. Een nadeel van het gebruik van een van deze API's is dat deze functies bevatten die handig zijn voor veel delen van de engine. Dit brengt een nadeel met zich mee, namelijk dat de engine dan afhankelijk wordt van deze extra API. Om later dan over te gaan op een andere API kost dan extra tijd. Ook om een engine dan crossplatform te krijgen gaat extra tijd kosten.

Hieronder staan de functionaliteiten van D3DX en GLUT inclusief voorbeelden in code:

Functionaliteiten van D3DX:

- Animatie functies
  - In deze library staan functies voor het creëren van een animatie
    - D3DXCreateKeyframedAnimationSet / D3DXFRAMENumNamedMatrices
- General Purpose Functions
  - Globale functies die in de hele engine van pas kunnen komen
    - D3DXCreateFont / D3DXCreateBuffer / D3DXCreateSprite
- Math Functions
  - Wiskundige functies waaronder de matrices en vectoren
    - D3DXMatrixMultiplyTranspose / D3DXMatrixRotationX
- Mesh Functions
  - Om meshes (verzameling van vertices, edges en surfaces) te kunnen benaderen is er een library met geoptimaliseerde mesh functies.
    - D3DXCreateMesh / D3DXLoadMeshFromX
- UVAtlas Functions
  - Om textures op meshes te 'plakken' kan van deze library gebruik gemaakt worden
    - D3DXUVAtlasCreate / D3DXUVAtlasPartition / D3DXComputeIMTFromSignal
- Precomputed Radiance Transfer (PRT) Functions
  - Geavanceerde dynamische lightning voor bij het renderer van een scene
    - D3DXCreatePRTBuffer / D3DXSavePRTBufferToFile
- Shader Functions
  - Om shaders gemakkelijker te kunnen compileren
    - D3DXCompileShader / D3DXAssembleShader / D3DXGetVertexShaderProfile
- Shape Drawing Functions
  - Het tekenen van standaard vormen of tekst
    - D3DXCreatePolygon / D3DXCreateSphere / D3DXCreatePolygon
- Texture Functions in D3DX 9
  - Het creëren/opslaan/filteren/laden van textures
    - D3DXCreateTextureFromFile / D3DXFillCubeTexture

Functionaliteiten van GLUT

- Basics

    - Basis functies van GLUT. Zoals het initialiseren van een venster en het openen van de GLUTmainLoop
        - glutCreateWindow / glutInitWindowSize / glutMainLoop
- Input
    - Voor het afhandelen van Input-devices, onder andere ook controllers.
        - glutSetKeyRepeat / glutSpecialUpFunc
- Pop-up Menus
    - Standaard functionaliteit om een popup te maken.
        - glutCreateMenu / glutAddMenuEntry
- Fonts
    - Het maken/laden van soorten lettertypen.
        - renderBitmapString / glutBitmapCharacter
- Subwindows
    - Het maken van sub-vensters waarnaar ook gerenderd kan worden
        - glutCreateSubWindow

Een voordeel van GLUT is dat deze cross-platform is. D3DX is dit niet en zal dus alleen werken op een besturingssysteem dat ontwikkeld is door Microsoft. De source-code van deze twee API's is niet vrij beschikbaar. Bij OpenGLUT is dit wel het geval. openGLUT is oorspronkelijk een fork (afsplitsing) van GLUT geweest met het doel deze broncode open-source te maken. OpenGLUT heeft daarom veel van de specificaties van GLUT en lijken deze API's sterk op elkaar. Een nadeel van GLUT en openGLUT is dat deze een stuk minder groot zijn en voor de wat grotere applicaties dus niet toereikend zijn.

Technische verschillen

Matrices

Bij de Polemos 3D engine worden er gebruik gemaakt van drie soorten matrices. De ProjectMatrix, de ViewMatrix en de WorldMatrix. Deze drie matrices komen overeen met de standaard die binnen Direct3D gebruikelijk is. In OpenGL zijn er vier matrices. De ProjectionMatrix, de ModelViewMatrix, de TextureMatrix en de ColorMatrix. Omdat de ColorMatrix en de TextureMatrix binnen Polemos niet van belang zijn, heb ik deze buiten beschouwing gelaten.

De ProjectionMatrix

De ProjectionMatrix word in allebei de API's gebruikt. Beide bestaan deze ook uit een 4X4 array gevuld met floats. Hoe deze floats zijn ingedeeld is echter wel anders. Hieronder staat een Direct3D matrix:

Direct3D

X-as	_M11	_M12	_M13	_M14
Y-as	_M21	_M22	_M23	_M24
Z-as	_M31	_M32	_M33	_M34
Translation	_M41 (x)	_M42 (y)	_M43 (z)	_M44

Deze Direct3D ProjectionMatrix is rowheavy. Dat betekend dat de waarden van links naar rechts moeten worden gelezen. Hierin zit het eerste verschil met de OpenGL ProjectionMatrix. OpenGL matrices zijn altijd Columnheavy. Dit komt erop neer dat er van boven naar beneden moet worden gekeken. Eigenlijk wordt de matrix omgedraaid langs de schaal-as (_M11 / _M22 / _M33 / _M44). Dit bekend dus dat X,Y en Z coördinaten op de plekken _M14 / _M24 en _M34 komen te staan. Het columnheavy van een matrix maken vanaf een rowheavy matrix wordt "*transpose*" genoemd.

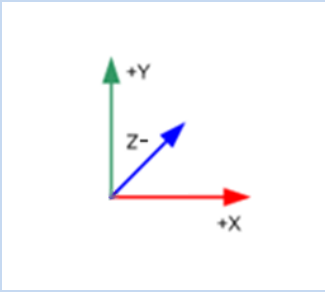
Hiernaast staat een Direct3D ProjectionMatrix die is ge-transpose-ed. Hierin is goed te zien dat de X,Y en de Z as zijn verplaats

OpenGL

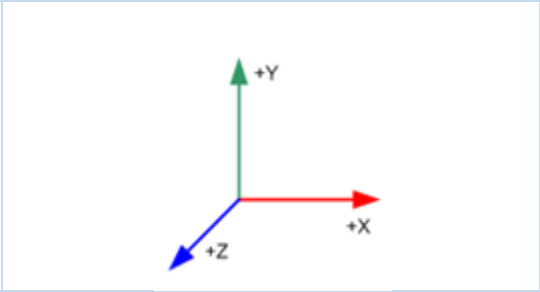
X-As	Y-as	Z-as	Translation
_M11	_M21	_M31	_M41 (x)
_M12	_M22	_M32	_M42 (y)
_M13	_M23	_M33	_M43 (z)
_M14	_M24	_M34	_M44

Naast dit verschil is het tweede verschil van de ProjectionMatrix dat de Z-as moet worden omgedraaid. Dit komt door het feit dat de ene API naar het beeld toe tekent en de andere van het beeld af.

OpenGL



Direct3D



De uiteindelijke ProjectionMatrix van OpenGL komt er dus als volgt uit te zien:

OpenGL

X-As	Y-as	Z-as	Translation
_M11	_M21	- _M31	_M41 (x)
_M12	_M22	- _M32	_M42 (y)
_M13	_M23	- _M33	_M43 (z)
_M14	_M24	_M34	_M44

De ViewMatrix

Zoals al in de inleiding staat beschreven, werkt OpenGL niet met een ViewMatrix. In plaats daarvan is het wel mogelijk om in Polemos de bestaande viewmatrix om te vormen naar een ModelViewMatrix. Om deze vertaalslag te kunnen maken

OpenGL –ModelViewMatrix

X-As	Y-as	Z-as	Translation
_M11	_M21	_M31	_M41 (x)
_M12	_M22	_M32	_M42 (y)
_M13	_M23	_M33	_M43 (z)
_M14	_M24	_M34	_M44

OpenGL – transposed ModelViewMatrix

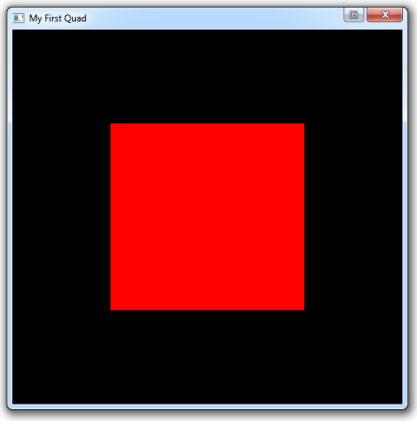
X-As	Y-as	Z-as	Translation
_M11	_M21	_M31	_M41 (x)
_M12	_M22	_M32	_M42 (y)
_M13	_M23	_M33	_M43 (z)
_M14	_M24	_M34	_M44

Direct3D ViewMatrix

X-as	_M11	_M12	- _M13	_M14
Y-as	_M21	_M22	- _M23	_M24
Z-as	- _M31	- _M32	_M33	- _M34
Translation	_M41 (x)	_M42 (y)	_M43 (z)	_M44

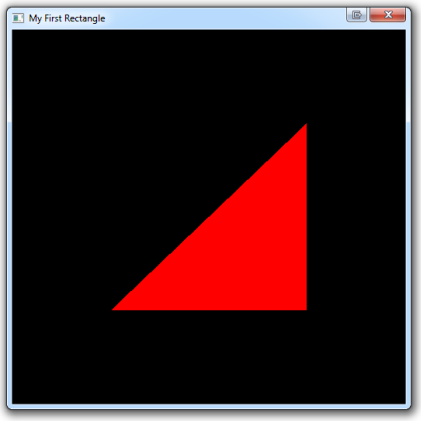
Drawing

Bij het tekenen in de 3D omgeving zijn ook verschillen. Het is namelijk niet mogelijk om, in Direct3D, vierkanten te tekenen. Direct3D kan alleen de vorm van een driehoek tekenen door middel van een TRIANGLELIST.



Een quad gerenderd in OpenGL

<u>Niet</u> mogelijk in Direct3D



Een triangle gerenderd in OpenGL

Ook mogelijk in Direct3D

Door een TRIANGLE list op te geven met 4 vertices (coördinaten) kan een vierkant worden getekend met Direct3D.

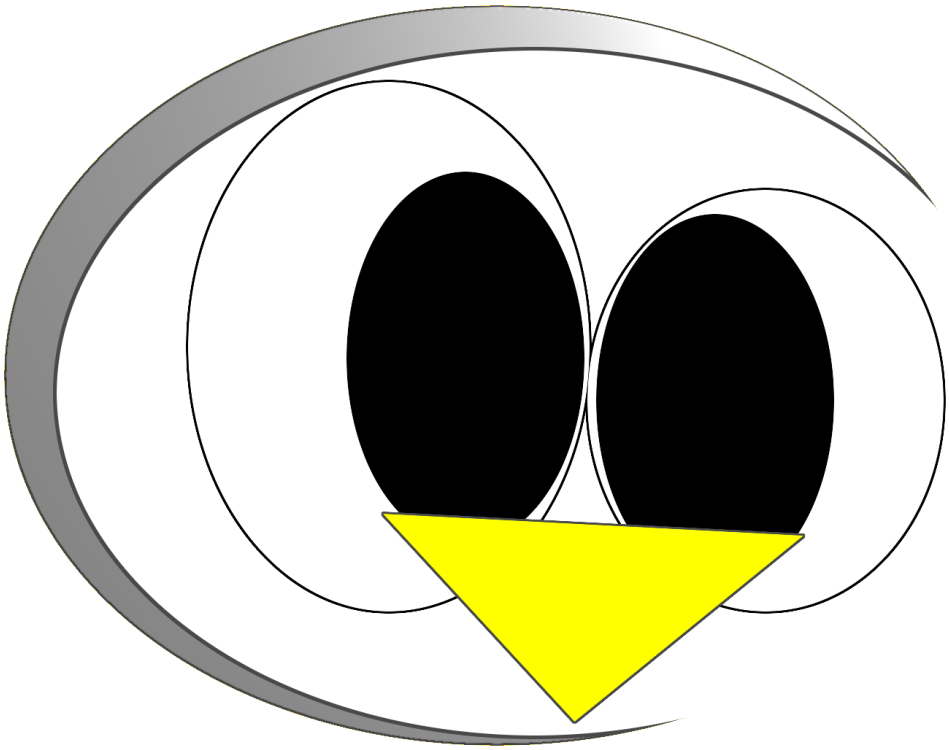
Scene

Om een scene te renderen in Direct3D, moet Direct3D eerst verteld worden dat de engine klaar is om te renderer. Dit wordt gedaan met de functie beginScene();. Zodra er niet meer gerenderd hoeft te worden, wordt de functie endScene(); aangeroepen, en kan er niets meer op het beeld worden gerenderd. OpenGL ken deze functionaliteit niet. De OpenGL renderer kan altijd renderen

Bronnenlijst

- http://www.cpp-home.com/tutorials/327_1.htm
- http://en.wikipedia.org/wiki/Comparison_of_OpenGL_and_Direct3D
- http://linuxnthings.blogspot.com/2010/03/opengl-vs-direct3d.html
- http://www.xmission.com/~legalize/d3d-vs-opengl.html
- http://www.opengl.org/documentation/specs/version1.1/glspec1.1/index.html
- http://social.msdn.microsoft.com

4. Rendererdocumentatie



Renderer documentatie

- Frank van Smeden
- Informatica
- 20064039

Academie voor ICT & Media

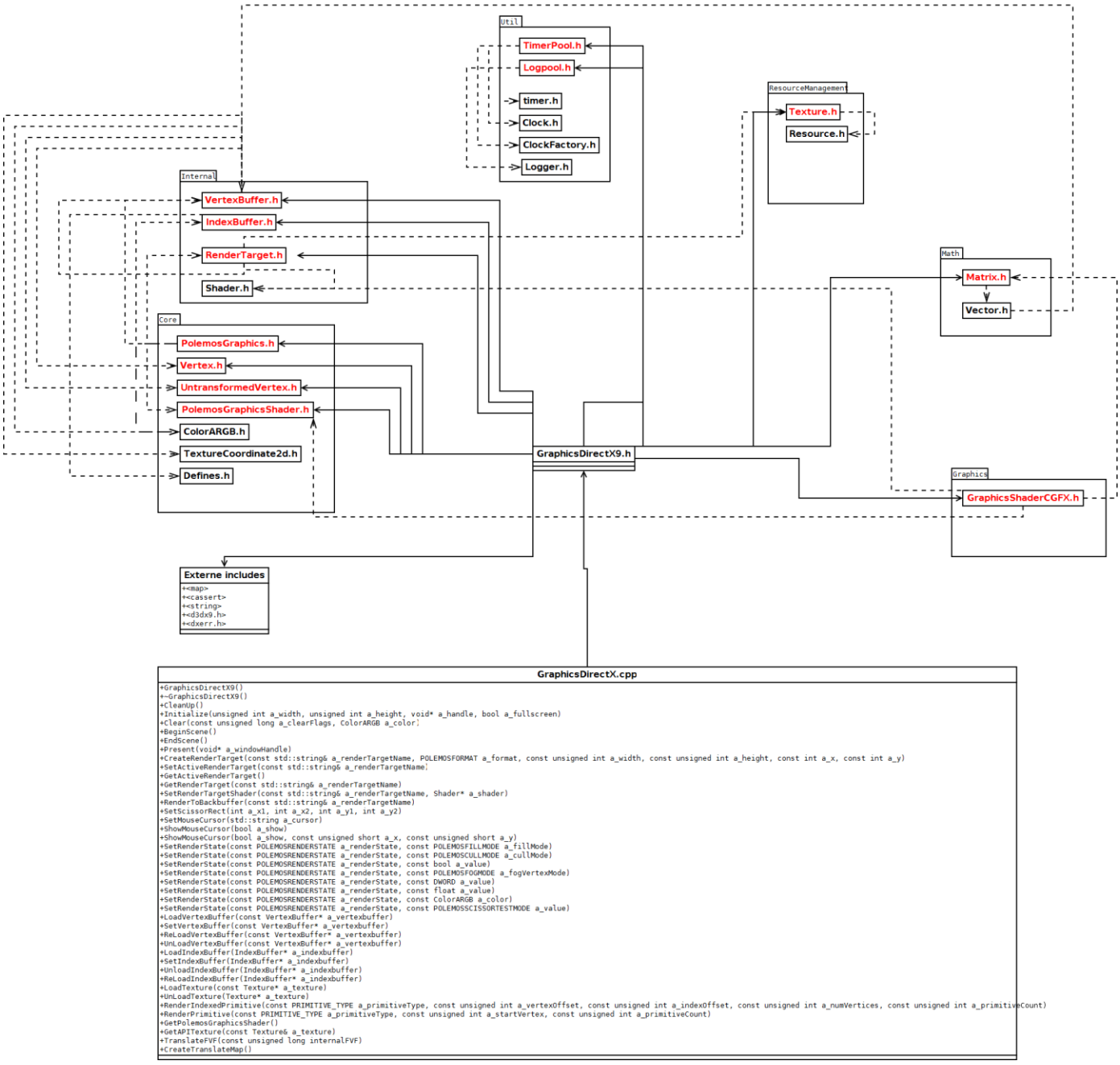
Bredewater 24 Zoetermeer

Inhoud

Situatie renderer

In dit diagram is te zien hoe de Direct3D renderer geïmplementeerd is binnen Polemos. Alle verbindingen die GraphicsDirectX.h heeft zijn aangegeven in het rood. Alle subverbindingen (verbindingen van directe verbindingen met GraphicsDirectX.h) zijn aangegeven in het zwart.

Uiteraard geldt deze structuur ook voor de OpenGL renderer.



CG profiles

Vertex profiles

Profile	Beschrijving	interface	Ondersteund in:	Polemos:
vs_1_1	vertex shader model 1.1	DirectX 8 / DirectX 9	• ATI Radeon 8500 • nVidia GeForce 3	Wordt niet gebruik in Polemos
vs_2_0	vertex shader model 2.0	DirectX 8 / DirectX 9	• ATI Radeon 9600 • nVidia GeForce FX 5 series	• SSDefault.cgfx • Colors.cgfx • Hit.cgfx • Quake.cgfx • selected.cgfx • selectedskelethumanwizard.cgfx • selectedskeletninja.cgfx • skelet.cgfx • skeletninja.cgfx • Skybox.cgfx • Terrain.cgfx
vs_2_x	vertex shader model 2.x	DirectX 8 / DirectX 9	• ATI Radeon X series • nVidia GeForce FX 6 series	Wordt niet gebruik in Polemos
Vs_3_0	vertex shader model 3.0	DirectX 9c	• ATI Radeon HD 2000+ • nVidia GeForce FX 6 series	• Default.cgfx • font.cgfx • gui.cgfx • ParticleDebug.cgfx • ParticleShowFire.cgfx • ParticleShowHeal.cgfx • ParticleShowLevelUp.cgfx • ParticleSimulateFire.cgfx (2x) • ParticleSimulateFountain.cgfx(2x) • ParticleSimulateHeal.cgfx(2x)

				• ParticleSimulateLevelUp.cgfx(2x) • ParticleSpawnFire.cgfx(2x) • ParticleSpawnFountain.cgfx(2x) • ParticleSpawnHeal.cgfx(2x) • ParticleSpawnLevelUp.cgfx(2x) • skelethumanwizard.cgfx • Terrain2.cgfx • water.cgfx
Vs_4_0	vertex shader model 4.0	DirectX 10.0	• nVidia GeForce 8 series • Radeon R600 (HD 2400-HD 2900)	Wordt niet gebruik in Polemos
Arbvp1	OpenGL Assembly Language vertex profile	OpenGL	• ATI Radeon 8500 • nVidia GeForce 3	Wordt niet gebruik in Polemos
Vp20	nVidia-specific OpenGL vertex shader syntax	OpenGL	• ATI Radeon 8500 • nVidia GeForce 3+ ATI Radeon HD 2000+	• SSDefault.cgfx • Colors.cgfx • Hit.cgfx • Quake.cgfx • selected.cgfx • selectedskelethumanwizard.cgfx • selectedskeletninja.cgfx • skelet.cgfx • skeletninja.cgfx • Skybox.cgfx • Terrain.cgfx
Vp30	nVidia-specific OpenGL vertex shader syntax	OpenGL	• nVidia GeForce FX 5 series and higher • ATI Radeon HD 2000+	• Default.cgfx • font.cgfx • gui.cgfx • ParticleDebug.cgfx • ParticleShowHeal.cgfx • ParticleShowLevelUp.cgfx • ParticleSimulateFire.cgfx(2x) • ParticleSimulateFountain.cgfx(2x) • ParticleSimulateHeal.cgfx(2x)

				• ParticleSimulateLevelUp.cgfx(2x) • ParticleSpawnFire.cgfx(2x) • ParticleSpawnFountain.cgfx(2x) • ParticleSpawnHeal.cgfx(2x) • ParticleSpawnLevelUp.cgfx(2x) • skelethumanwizard.cgfx • Terrain2.cgfx • water.cgfx
Vp40	nVidia-specific OpenGL vertex shader syntax	OpenGL	• nVidia GeForce FX 6 series and higher.	• ParticleShowFire.cgfx

Fragment profiles

Profile	Beschrijving	interface	Ondersteund in:	gebruik:
Ps_1_1*	fragement shader model 1.1	DirectX 8 / DirectX 9 / OpenGL	• nVidia GeForce 3 • ATI Radeon 8500	Wordt niet gebruik in Polemos
Ps_1_2*	fragement shader model 1.2	DirectX 8 / DirectX 9 / OpenGL	• nVidia GeForce 3 • ATI Radeon 8500	Wordt niet gebruik in Polemos
Ps_1_3*	fragement shader model 1.3	DirectX 8 / DirectX 9 / OpenGL	• nVidia GeForce 3 • ATI Radeon 8500	Wordt niet gebruik in Polemos
Ps_1_4*	fragement shader model 1.4	DirectX 8 / DirectX 9 / OpenGL	• nVidia GeForce FX 5 series • ATI Radeon 8500	Wordt niet gebruik in Polemos
Ps_2_0	fragement shader model 2.0	DirectX 9	• nVidia GeForce FX 5 series • ATI Radeon 9600	• SSDefault.cgfx • BrightWorldShader.cgfx • Colors.cgfx • fadeout.cgfx • Hit.cgfx • pixellate_old_tv.cgfx • Quake.cgfx • selected.cgfx • selectedskelethumanwizard.cgfx • selectedskeletninja.cgfx • skelet.cgfx • skeletninja.cgfx • Skybox.cgfx • SSDefault.cgfx • Targeted.cgfx • Terrain.cgfx

Ps_2_x	fragement shader model 2.x	DirectX 9	• nVidia GeForce FX 6 series • ATI Radeon X series	Wordt niet gebruik in Polemos
Ps_3_0	fragement shader model 3.0	DirectX 9c	• nVidia GeForce FX6 series • ATI Radeon HD 2000+	• Default.cgfx • font.cgfx • gui.cgfx • ParticleDebug.cgfx • ParticleShowFire.cgfx • ParticleShowHeal.cgfx • ParticleShowLevelUp.cgfx • ParticleSimulateFire.cgfx(2x) • ParticleSimulateFountain.cgfx(2x) • ParticleSimulateHeal.cgfx(2x) • ParticleSimulateLevelUp.cgfx(2x) • ParticleSpawnFire.cgfx(2x) • ParticleSpawnFountain.cgfx(2x) • ParticleSpawnHeal.cgfx(2x) • ParticleSpawnLevelUp.cgfx(2x) • skelethumanwizard.cgfx • Terrain2.cgfx • water.cgfx
Ps_3_x	fragement shader model 3.x	DirectX 9c	• nVidia GeForce FX7 series	Wordt niet gebruik in Polemos
Ps_4_0	fragement shader model 4.0	DirectX 10.0	• nVidia GeForce 8 series • Radeon R600 (HD 2400-HD 2900)	Wordt niet gebruik in Polemos
Arbfp1	OpenGL Assembly Language fragment profile	OpenGL	• ATI Radeon 9600 • nVidia GeForce FX 5 series	Wordt niet gebruik in Polemos
Fp20	nVidia-specific OpenGL fragment syntax	OpenGL	• nVidia GeForce 3 • Ati Radeon 8500	• SSDefault.cgfx • BrightWorldShader.cgfx • Colors.cgfx • fadeout.cgfx

				• Hit.cgfx • pixellate_old_tv.cgfx • Quake.cgfx • selected.cgfx • selectedskelethumanwizard.cgfx • selectedskeletninja.cgfx • skelet.cgfx • skeletninja.cgfx • Skybox.cgfx • SSDefault.cgfx • Targeted.cgfx • Terrain.cgfx
Fp30	nVidia-specific OpenGL fragment syntax	OpenGL	• nVidia GeForce FX 5 series and higher • ATI Radeon HD 2000+	• Default.cgfx • font.cgfx • gui.cgfx • ParticleDebug.cgfx • ParticleShowHeal.cgfx • ParticleShowLevelUp.cgfx • ParticleSimulateFire.cgfx(2x) • ParticleSimulateFountain.cgfx(2x) • ParticleSimulateHeal.cgfx(2x) • ParticleSimulateLevelUp.cgfx(2x) • ParticleSpawnFire.cgfx(2x) • ParticleSpawnFountain.cgfx(2x) • ParticleSpawnHeal.cgfx(2x) • ParticleSpawnLevelUp.cgfx(2x) • skelethumanwizard.cgfx • Terrain2.cgfx • water.cgfx
Fp40	nVidia-specific OpenGL fragment syntax	OpenGL	• GeForce FX 6 series and higher.	• ParticleShowFire.cgfx

Naast een profile bij naam aan te geven, kan er ook gekozen worden om de laatste profile, dat de videokaart aan kan, te gebruiken. Dit is echter niet efficiënt. Om snelheidsredenen is het beter te compileren met een zo laag mogelijk fragment of vertex profile.

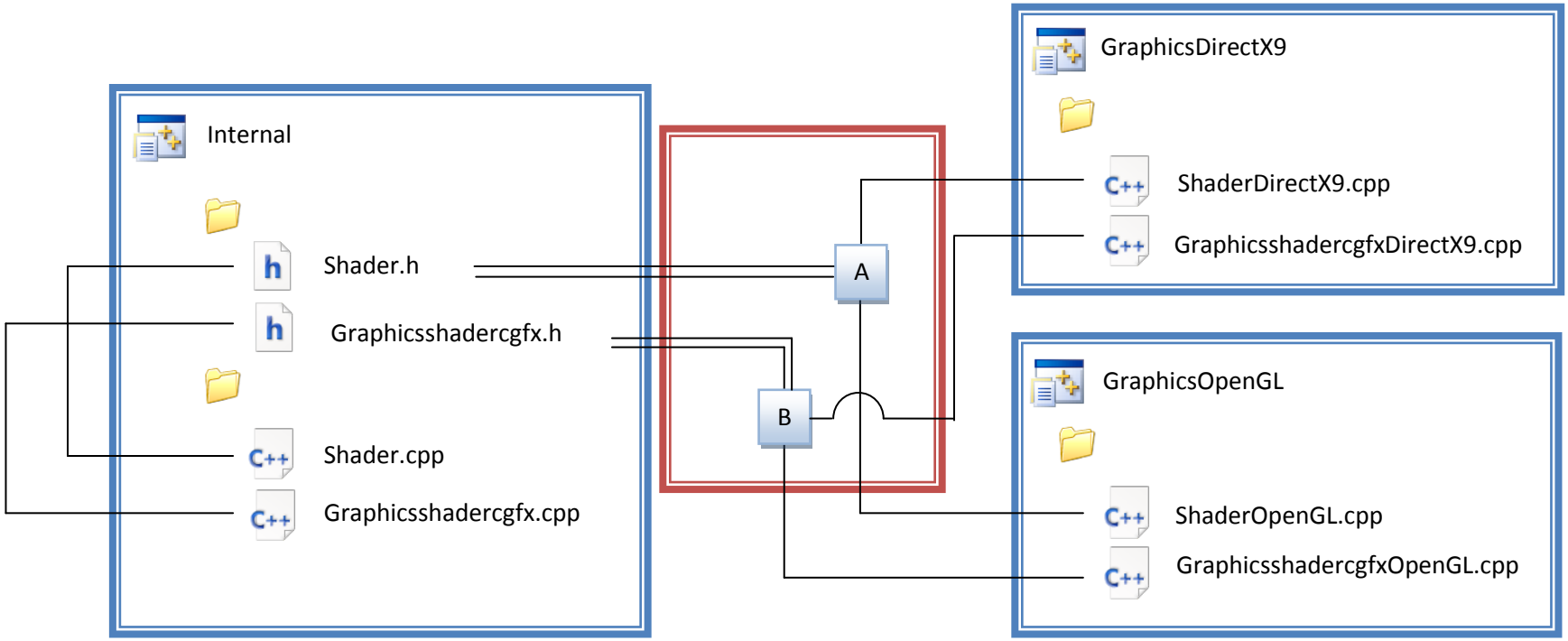
Daarnaast moet tijdens een implementatie van Arbvp1 en Arbfp1 goed gekeken worden of deze profielen wel geschikt zijn. Deze assembly language profiles zijn erg snel en bestaan voor een groot deel uit assembly, maar ondersteunen tot shader model 2.0.

CGFX implementatie

Binnen Polemos wordt gebruik gemaakt van CGFX. CGFX is crossplatform en werkt dus ook in combinatie met Direc3D en OpenGL. Hoewel deze crossplatform is, heeft CGFX wel andere benaderingen als er met een DirectX API wordt gewerkt, dan als er met OpenGL wordt gewerkt. Zo beginnen alle OpenGL specefieke methoden met “cgGL” en alle Direct3D specefieke methoden met “cgD3D”. In Polemos wordt CG geinitialiseerd zodra de renderer wordt geinitialiseerd. Bij het initialiseren van CG wordt ook gelijk een shader geladen (“ssdefault.cgfx”). Ook shaders zijn cross-platform. Er zijn echter wel verschillende profiles nodig voor het compileren (zie hoofdstuk over CG profiles).

In de engine zijn er twee bestanden die gaan over de shaders, namelijk shader (.h en .cpp en shaderCGFX (.h en .cpp). Maar aangezien er soms renderer-specifieke calls worden gedaan zijn die calls opgenomen in ‘ShaderCGFXDirectX.cpp’ en ‘ShaderCGFXOpenGL.cpp’. Deze aparte implementaties worden tijdens het compileren door middel van static-linking aan de niet-renderer specifieke bestanden gekoppeld. Op deze manier hoeft er geen abstractie gemaakt te worden. De reden voor het niet aanmaken van een abstractie is omdat het hier maar over een kleine hoeveelheid code gaat. Abstractie zou de onderhoudbaarheid van de engine negatief beinvloeden.

In het diagram op de volgende pagina wordt weergegeven hoe de verhouding tussen de shaders en de verschillende projecten liggen.


Internal
Shader.h
Graphicsshadercgfx.h
Shader.cpp
Graphicsshadercgfx.cpp
A
B
GraphicsDirectX9
ShaderDirectX9.cpp
GraphicsshadercgfxDirectX9.cpp
GraphicsOpenGL
ShaderOpenGL.cpp
GraphicsshadercgfxOpenGL.cpp

A

Shader.h	Shader.cpp	shaderDirectX9.cpp	ShaderOpenGL.cpp
Beschrijving			
Bevat alle definities van methodes en variabelen van een shader	Alle niet-renderer specifieke implementaties van een shader	Alle Direct3D specifieke implementaties van een shader	Alle OpenGL specifieke implementaties van een shader
Implementaties			
Alle calls van de shader	Alle calls die niet renderer-API specifiek zijn	Alle renderer-api specifieke calls • `SetConstantParams();` specific calls: o `cgGLSetTextureParameter();` o `cgD3D9SetTextureParameter();` • `SetReplacableParams();` specific calls: o `cgGLSetTextureParameter();` o `cgD3D9SetTextureParameter();` • `SetReplacableParamRange();` specific calls: o `cgGLSetTextureParameter();` o `cgD3D9SetTextureParameter();`	

B

Graphicsshadercgfx.h	Graphicsshadercgfx.cpp	GraphicsshadercgfxDirectX9.cpp	GraphicsshadercgfxOpenGL.cpp
Beschrijving			
Bevat alle definities van methodes en variabelen van shadercgfx	Alle niet-renderer specifieke implementaties van shadercgfx	Alle Direct3D specifieke implementaties van shadercgfx	Alle OpenGL specifieke implementaties van shadercgfx
Implementatie			
Alle methoden van de Graphicsshader	Alle calls die niet renderer-API specifiek zijn • `LoadShader();` • `SetWorldMatrix();` • `SetViewMatrix();` • `SetViewMatrix();` • `UpdateTime();` • `AddDefaultParameters();`	Alle renderer-api specifieke calls • `cgErrorCallback();` specific calls: o `cgGetErrorString ();` o `cgD3D9TranslateCGerror ();` • `Initialize();` specific calls: o `cgD3D9SetDevice ();` o `cgD3D9RegisterStates ();` o `cgD3D9SetManageTextureParameters();` o `cgSetParameterSettingMode();` o `cgGLRegisterStates();` o `cgGLSetManageTextureParameters();`	

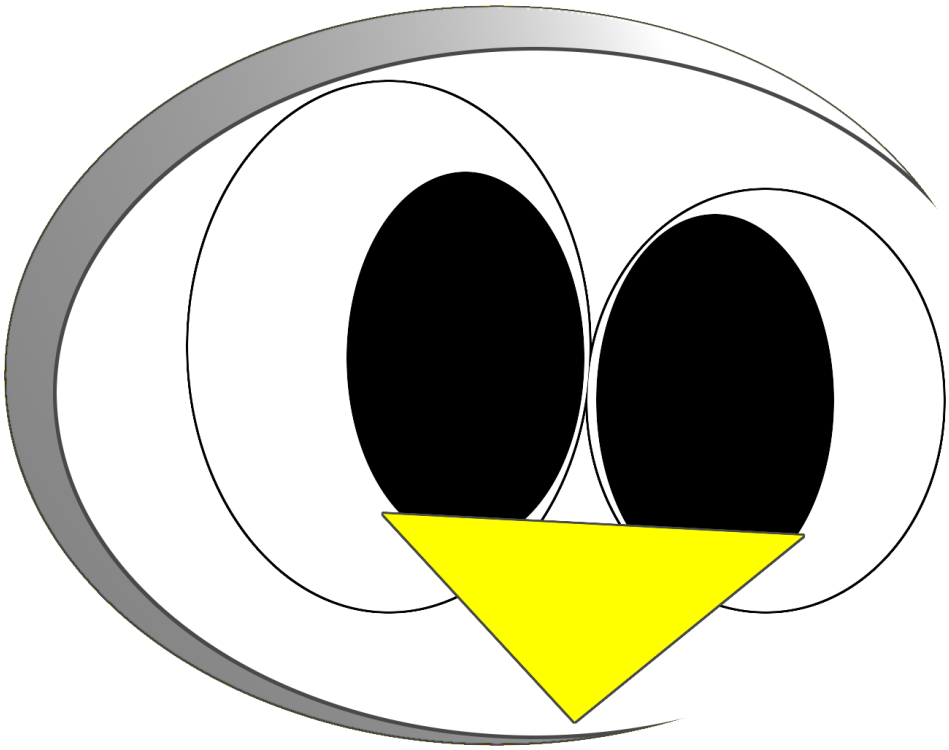
Bestanden

DirectX	OpenGL
Header Files GraphicsDirectX9.h RendererFactory.h	Header Files GraphicsOpenGL.h OpenGLExtensions.h RendererFactory.h
Source Files GraphicsDirectX9.cpp GraphicsShaderCGFXDirectX PolemosGraphicsFactory.cp ShaderDirectX9.cpp	Source Files GraphicsOpenGL.cpp GraphicsShaderCGFXOpenGL.cpp PolemosGraphicsFactory.cpp ShaderOpenGL.cpp

Little things you should know

- Om naar de OpenGL renderer te wissellen in Visual Studio klik op met de rechtermuisknop op het project “Development” > Project dependencies > vink GraphicsDirectX9 uit en vink GraphicsOpenGL aan.
- De OpenGL renderer is gebouwd onder OpenGL versie 3.2
- De defaultshader (“ResourceFiles\Default\Default.OpenGL.cgfx”) kan worden gebruikt bij het testen van OpenGL. Omdat Textures nog niet worden ondersteund laat deze shader een wit vlak zien. Dit kan het testen iets gemakkelijker maken.

5. CXXtest framework aanpassingen



CXXtest framework adjustments

- Frank van Smeden
- Informatica
- 20064039

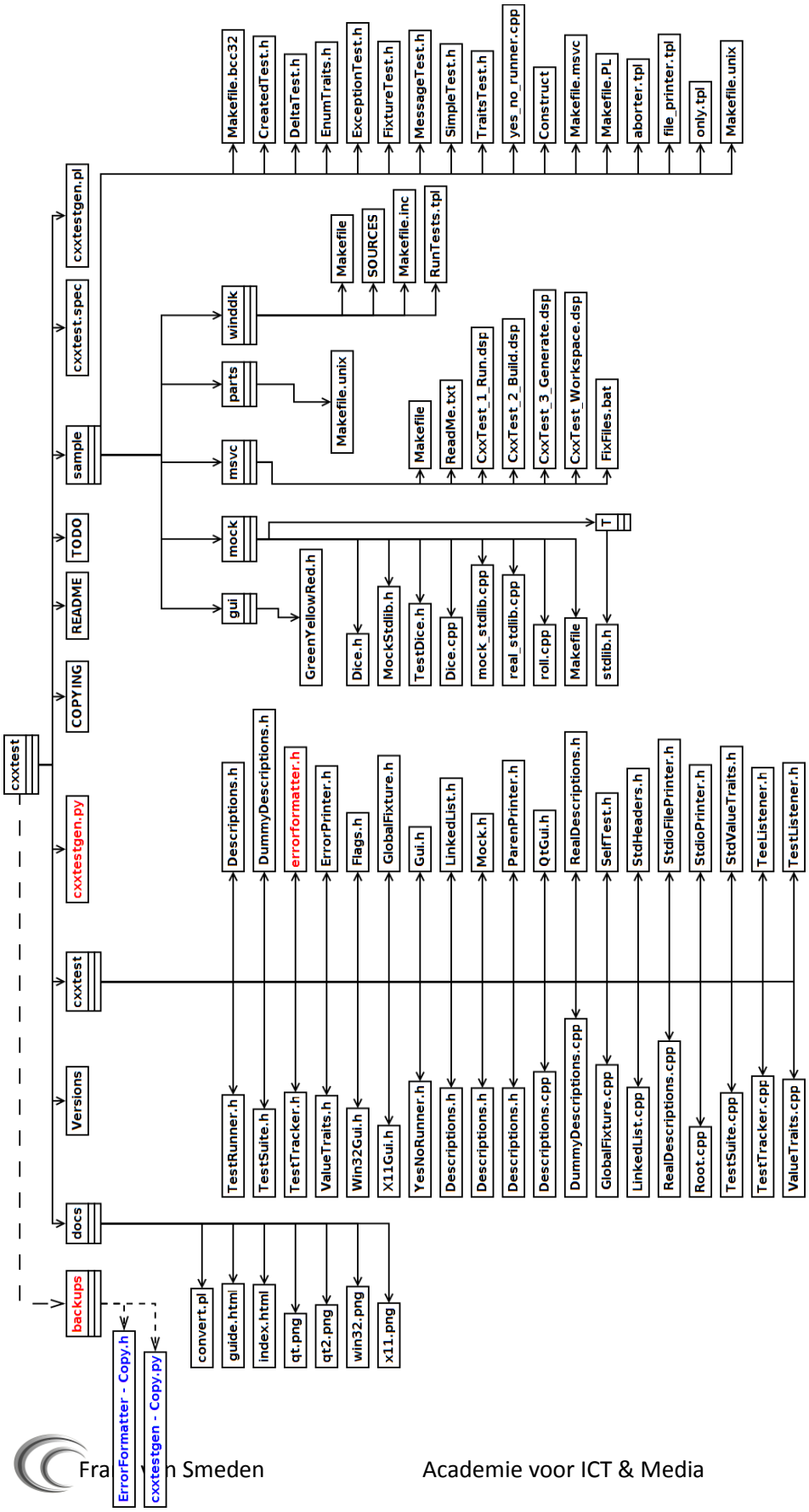
Academie voor ICT & Media

Bredewater 24 Zoetermeer

Documentbeheer ~ Aanpassingen CXXtest - framework

Versie	Datum	Aanpassing
0.1	11 – juni	Hoofdstukken aangemaakt / voorkant geïmporteerd / rapportage toegevoegd
0.2*	14 - juni	Verbeteringen / diagram uitleg /opmaak
0.3	23 - juni	Uitleg toegevoegd

Dit document bevat de documentatie van het testeframework dat is geïmplementeerd binnen de 3D engine Polemos. Dit unittest framework is CXXtest. Verder zijn alle aanpassen op dit framework aangaande de broncode hier beschreven. Indien er een nieuwe versie van CXXtest beschikbaar is, kan deze opnieuw worden aangepast met deze informatie.



Dit diagram legt de mappenstructuur van CXXtest uit

De rode bestanden zijn de bestanden die zijn aangepast om CXXtest aft e stemmen op Polemos. De blauwe bestanden zijn backup betanden die, indien nodig, kunnen worden teruggezet.

ErrorFormatter.h

~ void leaveTest(const TestDescription &)

Standard implementation

```
void leaveTest( const TestDescription & )
{
     if ( !tracker().testFailed() )
     {
        ((*_o) << ".").flush();
        _dotting = true;
     }
}
```

Must become

```
void leaveTest( const TestDescription & )
{
     if ( !tracker().testFailed() ) {
        //((*_o) << "...").flush();
        _dotting = true;
     }
}
```

ErrorFormatter.h

~ void enterWorld(const WorldDescription & /*desc*/)

Standard implementation

```
void enterWorld( const WorldDescription & /*desc*/ )
{
    (*_o) << "Running " << totalTests;
    _o->flush();
    _dotting = true;
    _reported = false;
}
```

Must become

```
void enterWorld( const WorldDescription & /*desc*/ )
{
    (*_o) << "==========================================================================\n";
    (*_o) << "Running " << totalTests << "\n\nMessages and warnings...\n" ;

    _o->flush();
    _dotting = true;
    _reported = false;
}
```

ErrorFormatter.h

~ void leaveWorld(const WorldDescription &desc)

Standard implementation

```
void leaveWorld( const WorldDescription &desc )
{
    if ( !tracker().failedTests() )
    {
        (*_o) << "OK!" << endl;
        return;
    }
    newLine();
    (*_o) << "Failed " << tracker().failedTests() << " of " << totalTests << endl;
    unsigned numPassed = desc.numTotalTests() - tracker().failedTests();
    (*_o) << "Success rate: " << (numPassed * 100 / desc.numTotalTests()) << "%" << endl;
}
```

Must become

```
void leaveWorld( const WorldDescription &desc )
{
   (*_o) << "\n==========================================================================\n";
   if ( !tracker().failedTests() )
   {
        (*_o) << "Conclusion:\t\t\t\tVictory, all tests passed! congrats!" << endl;
   }
   else
   {
         (*_o) << "Conclusion:\t\t\t\tToo bad, Some tests failed" << endl;
   }
   newLine();
   (*_o) << "Number of failed\t\t\t" << tracker().failedTests() << " of " << totalTests << endl;
   (*_o) << "Number of warnings:\t\t\t" << tracker().warnings() << endl;

   unsigned numPassed = desc.numTotalTests() - tracker().failedTests();
   (*_o) << "Success rate: \t\t\t\t" << (numPassed * 100 / desc.numTotalTests()) << "%" << endl;
}
```

ErrorFormatter.h

~ OutputStream &stop(const char *file, unsigned line)

Standard implementation

```
OutputStream &stop( const char *file, unsigned line )
{
    newLine();
    reportTest();
    return (*_o) << file << _preLine << line << _postLine << ": ";
}
```

Must become

```
OutputStream &stop( const char *file, unsigned line )
{
    newLine();
    reportTest();

    string a = file;
```

```
    string b= "Undefind";

    b = a.substr( a.rfind("\\") , a.size() );
    return (*_o) << "\t" << b.c_str() << " " <<_preLine << "Line:" << line << _postLine << " " <<
": ";
}
```

ErrorFormatter.h

~ void reportTest(void)

Standard implementation

```
void reportTest( void )
{
    if( _reported )
        return;
        (*_o) << "In " << tracker().suite().suiteName() << "::" << tracker().test().testName() <<
":" << endl;
        _reported = true;
}
```

Must become

```
void reportTest( void )
{
     if( _reported )
        return;

        (*_o) << "->Report from: \n\t" << tracker().suite().suiteName() << "::" <<
tracker().test().testName() << ":" << endl;
        _reported = true;
}
```

ErrorFormatter.h

~ void reportTest(void)

Standard implementation

```
void reportTest( void )
{
     if( _reported )
        return;

     (*_o) << "In " << tracker().suite().suiteName() << "::" << tracker().test().testName() <<
":" << endl;
      _reported = true;
}
```

Must become

```
void reportTest( void )
{
     if( _reported )
        return;

     (*_o) << "->Report from: \n\t" << tracker().suite().suiteName() << "::" <<
```

```
tracker().test().testName() << ":" << endl;
      _reported = true;
}
```

Cxxtestgen.py

~ def writeMain(output):

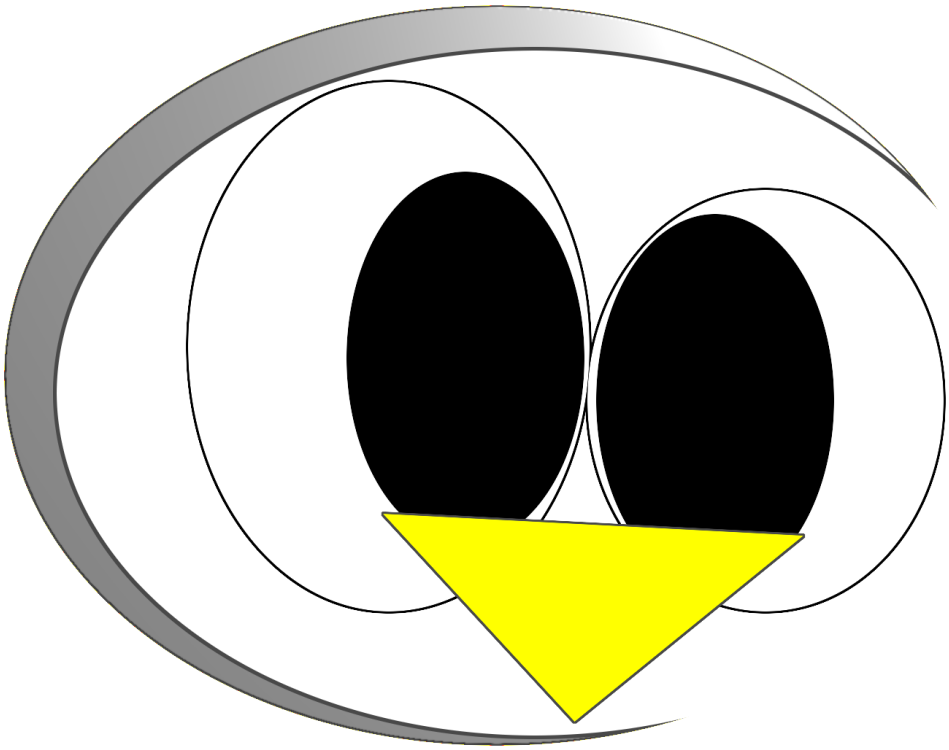
Standard implementation

```
def writeMain( output ):
    '''Write the main() function for the test runner'''
    if gui:
        output.write( 'int main( int argc, char *argv[] ) {\n' )
        if noStaticInit:
            output.write( ' CxxTest::initialize();\n' )
        output.write( ' return CxxTest::GuiTuiRunner<CxxTest::%s, CxxTest::%s>( argc, argv
).run();\n' % (gui, runner) )
        output.write( '}\n' )
    elif runner:
        output.write( 'int main() {\n' )
        if noStaticInit:
            output.write( ' CxxTest::initialize();\n' )
        output.write( ' return CxxTest::%s().run();\n' % runner )
        output.write( '}\n' )
```

Must become

```
def writeMain( output ):
    '''Write the main() function for the test runner'''
    if gui:
        output.write( 'int main( int argc, char *argv[] ) {\n' )
        if noStaticInit:
            output.write( ' CxxTest::initialize();\n' )
        output.write( ' return CxxTest::GuiTuiRunner<CxxTest::%s, CxxTest::%s>( argc, argv
).run();\n' % (gui, runner) )
        output.write( '}\n' )
    elif runner:
        output.write( 'int main() {\n' )
        if noStaticInit:
            output.write( ' CxxTest::initialize();\n' )
        output.write( ' int result = CxxTest::%s().run();\n ' % runner )
        output.write( ' int c; \n' )
        output.write( ' std::cin >> c; \n' )
        output.write( '  return result;\n' )
        output.write( '}\n' )
```

6. Adviesrapport



Adviesrapport platformonafhankelijkheid Polemos

- Frank van Smeden
- Informatica
- 20064039

Academie voor ICT & Media

Bredewater 24 Zoetermeer

Inhoudsopgave

Inleiding

In dit rapport zal worden beschreven wat de voor- en nadelen zijn voor het crossplatform maken van de Polemos engine. Ook zal ik de aanpassingen beschreven waarbij rekening moet worden gehouden bij een eventuele implementatie van andere libraries.

Ook zal ik in dit rapport een conclusie vormen waarin een aantal mogelijke richtingen uiteen worden gezet over de toekomst van Polemos.

Onderzoek

Is het rendabel om Polemos crossplatform te gaan maken? Om op deze vraag een antwoord te geven heb ik de ISO standaard (ISO 9001) als richtlijn gebruikt. Door middel van deze richtlijnen kan er worden nagegaan waarin Polemos vooruit of achteruit zou gaan met een overgang naar een crossplatform implementatie.

- Functionality

  De functionaliteit van Polemos zal niet veranderen, de implementatie van crossplatform libraries zal dezelfde functionaliteit verspreiden over meerdere platformen. Wel is het zo dat bij de implementatie van andere libraries misschien andere functionaliteiten mogelijk zijn dan bij de huidige libraries. Dit is echter afhankelijk van welke library zal worden geïmplementeerd.
- Usability

  De gebruiksvriendelijkheid van Polemos zal niet aangepast worden.
- Maintainability

  De onderhoudbaarheid van Polemos kan erg achteruit gaan als er wordt gekozen om meerdere libraries tegelijkertijd te onderhouden. Als bijvoorbeeld de Direct3D renderer naast de OpenGL renderer onderhouden moet worden, zal alle nieuwe functionaliteit voor deze renderers twee keer moeten worden geïmplementeerd. Namelijk binnen OpenGL en binnen Direct3D. Als er gekozen word voor 1 renderer API zal dit zeker zorgen voor een betere onderhoudbaarheid. Nadeel hiervan is echter dat het wel veel werk zal zijn om Polemos geheel crossplatform te maken
- Reliability
  - De betrouwbaarheid van Polemos zal niet aangepast worden.
- Efficiency
  - De efficiëntie van Polemos zal niet aangepast worden.
- Portability
  - De Portability van Polemos zal erg worden verhoogd met een crossplatform implementatie. Ook zal hierbij een stap richting de consoles worden gemaakt. Als er namelijk meer crossplatform libraries worden gebruikt, is het gemakkelijker om later over te stappen naar bijvoorbeeld een Wii of een Playstation omdat veel libraries dan goed abstract zijn geïmplementeerd.

Naast de ISO standaard is het van belang te kijken naar waar Polemos voor gebruikt wordt. Polemos is in principe een leerproces voor 3^{e} en 4^{e} jaars informatica studenten. Het bestaansrecht voor Polemos is dus niet de kwaliteit van de engine en het gebruik hiervan, maar de leeromgeving door middel van het ontwikkelen van een 3D engine.

Daarnaast zou, door het gebruik van andere en nieuwe libraries, Vincent Broerer (leraar 3D engine en eigenaar van Polemos) eerst kennis op moeten doen over de nieuwe libraries alvorens hij hier les in kan

geven. Hier tegenover staat dat studenten wel is aanraking komen met andere soorten libraries en waardoor hun kennis dus wordt vergroot op het gebied van 3D engine development.

Aanpassingen

Wat op dit moment Polemos weerhoud om op meerdere besturingsystemen gedraaid te kunnen worden, zijn de externe libraries die worden gebruikt. Deze libraries zorgen in Polemos voor het afhandelen van de Input (DirectInput), het aanmaken en configureren van een window (Windows SDK) en het renderen van output (Direct3D). Deze naast deze drie niet-crossplatform libraries zijn er ook nog een aantal libraries die worden gebruikt en wel al crossplatform zijn namelijk CG, LUA en OpenGL.

Om ervoor te zorgen dat de drie niet-crossplatforme libraries uit Polemos kunnen worden gehaald zonder functionaliteit te verliezen zullen er alternatieve moeten worden geïmplementeerd. De volgende project maken gebruik van een niet-crossplatforme library:

- GraphicsDirectX9 –project (Direct3D)
    - Op dit moment zijn er twee renderers aanwezig in Polemos. Een hiervan is geschreven in Direct3D en staat in het “GraphicsDirectX9”-project. De andere renderer is geschreven in OpenGL waarvan op dit moment alleen basisfunctionaliteit geïmplementeerd is. Een alternatief voor deze library is dus al aanwezig, maar niet geheel operationeel.

- Input (Windows SDK / *DirectInput)*
    - *DirectInput wordt op dit moment gebruikt voor de Input. Deze werkt uiteraard alleen op het Windows besturingsysteem.*
- *Math (Windows SDK)*
    - *Dit Project gebruikt de Windows SDK het de berekening van Planes en quad-tree’s. (zie plane.h en quad-tree.h.*
- *Window (Windows SDK)*
    - *Als alternatief voor het aanmaken van Windows kan X11 worden gebruikt. Deze werkt niet standaard op het Windows besturingssysteem maar kan wel worden gedraaid met een paar kleine aanpassingen.*

Naast de externe libraries zijn er nog een aantal aspecten waarmee rekening mee moet worden gehouden in het geval van een crossplatform implementatie:

- Path-abstractie moeten worden in gebouwd
    - Linux en andere besturingsystemen maken anders gebruik van paths dan Windows. Hier zal dus een abstractie in moeten komen om andere soorten paths-formats te kunnen gebruiken
- Little and big endian

- Bij de opslag en lezen van data,bepaald de architectuur van de CPU hoe deze data word geïnterpreteerd. Vaak wordt hier een standaard format voor gebruikt, maar sommige besturingsystemen op een andere architecturen gebruiken een ander format. Op dit moment gebruikt Polemos een little-endian format.

  Little-endian besturingsystemen:

  - Linux (x86, x64, Alpha en Itanium)
  - Mac OS (86, x64)
  - Solaris (x86, x64, PowerPC)
  - Tru64 (Alpha)
  - OpenVMS (VAX, Alpha and Itanium)
  - Windows (x86, x64 en Itanium)

  Big-endian besturingssystemen:

  - AIX (POWER)
  - AmigaOS (PowerPC and 680x0)
  - HP-UX (Itanium and PA-RISC)
  - Linux (IPS, SPARC, PA-RISC, POWER, PowerPC en 680x0)
  - Mac OS (PowerPC and 680x0)
  - Solaris (SPARC)

  Als er naar een little-endian format moet worden overgestapt naar big-endian zullen er problemen ontstaan met het inladen van textures en het inladen van de scene-file.

Huidige situatie library-gebruik per project

Hieronder staat per Polemos project weergegeven of deze externe libraries gebruikt en of deze crossplatform zijn of niet.

Animation -project		In huidige vorm crossplatform:	Ja
Maakt gebruik van externe libraries:		*geen*	
Header files:		Source files:	
h	AnimationLoader.h	C++	AnimationLoader.cpp
h	animationset.h		
Beschrijving			
De anitmaties van Polemos zijn op dit moment crossplatform			

Audio -project	In huidige vorm crossplatform: Ja
Maakt gebruik van externe libraries:	• *FMOD*

Header files:	Source files:
AudioManager.h	AudioManager.cpp
ResourceFMODModule.h	ResourceFMODModule.cpp
ResourceFMODSample.h	ResourceFMODSample.cpp
ResourceFMODStream.h	ResourceFMODStream.cpp
ResourceLoaderFMODModule.h	ResourceLoaderFMODModule.cpp
ResourceLoaderFMODSample.h	ResourceLoaderFMODSample.cpp
ResourceLoaderFMODStream.h	ResourceLoaderFMODStream.cpp

Beschrijving

Het audio-project maakt gebruik van FMOD. Deze library is van zichzelf crossplatform

Core -project	In huidige vorm crossplatform: Ja
Maakt gebruik van externe libraries:	*geen*

Header files:	Source files:
ColorARGB.h	Kernel.cpp
Defines.h	PolemosAudioManager.cpp
Kernel.h	PolemosGraphics.cpp (excluded from build)
PolemosAudioManager.h	PolemosGraphicsShader.cpp
PolemosGraphics.h	PolemosWindowManager.cpp
PolemosGraphicsFactory.h	PolemosInputManager.cpp
PolemosGraphicsShader.h	
PolemosWindowManager.h	
PolemosWindowManagerFactory.h	
TextureCoordinate2d.h	
Vertex.h	
PolemosActionListener.h	
PolemosInputDefs.h	
PolemosInputManager.h	
PolemosInputManagerFactory.h	
PolemosKeyListener.h	

Beschrijving
Het core-project mag geen externe libraries bevatten aangezien dit project alleen abstracties van implementaties mag benaderen.

Development -project		In huidige vorm crossplatform:	Ja
Maakt gebruik van externe libraries:		*geen*	
Header files:		Source files:	
h	actionlistenertest.h	C++	actionlistenertest.cpp (excluded from build)
h	Allocator.h	C++	AudioTest.cpp (excluded from build)
h	ListenerTest.h	C++	entitycamerafollow.cpp (excluded from build)
		C++	GuitTest.cpp (excluded from build)
		C++	InputManagerTest.cpp (excluded from build)
		C++	ListenerTest.cpp (excluded from build)
		C++	Main.cpp (excluded from build)
		C++	ModelProfiling.cpp (excluded from build)
		C++	ParticleSystemTest.cpp
		C++	renderertestdirect3d.cpp (excluded from build)
		C++	ScriptingLinkerTest.cpp (excluded from build)
		C++	ScriptingManagerTest.cpp (excluded from build)
		C++	testBasic.cpp (excluded from build)
		C++	TestWindow.cpp (excluded from build)

Beschrijving
Het development project bevat eigenlijk alleen de main functie waar de engine wordt gestart. Deze zal dus altijd onafhankelijk zijn van externe API's

GraphicsDirectX9 -project	In huidige vorm crossplatform: NEE
Maakt gebruik van externe libraries:	• *<D3dx9.h>* • *<Dxerr.h>* • *<cg/cg.h>* • *Cg/cgD3D9.h>*
Header files:	Source files:

	GraphicsDirectX9.h		GraphicsDirectX9.cpp
	GraphicsShaderCGFXDirectX9.h (excluded from build)		GraphicsShaderCGFXDirectX.cpp
	RendererFactory.h		PolemosGraphicsFactory.cpp
			ShaderDirectX9.cpp

Beschrijving

De Direct3D renderer is niet crossplatform. Direct3D kan ook nooit crossplatform gemaakt worden omdat deze exclusief is geschreven voor het Windows besturingssysteem.

GraphicsOpenGL -project	In huidige vorm crossplatform: NEE
Maakt gebruik van externe libraries:	• *<gl/gl.h>* • *<gl/glu.h>* • *<cg/cggl.h>* • *<cg/cg.h>* • *<Windows.h>*

Header files:		Source files:	
	GraphicsOpenGL.h		GraphicsOpenGL.cpp
	GraphicsShaderCGFXOpenGL.h (excluded from build)		GraphicsShaderCGFXOpenGL.cpp
	OpenGLExtensions.h		PolemosGraphicsFactory.cpp
	RendererFactory.h		ShaderOpenGL.cpp

Beschrijving

Alhoewel OpenGL op zichzelf een crossplatform API is. Wordt er in deze renderer ook nog gebruik gemaakt van Windows.h om OpenGL te initialiseren.

Gui -project	In huidige vorm crossplatform: JA
Maakt gebruik van externe libraries:	*geen*

Header files:		Source files:	
	EventHandler.h		EventHandler.cpp
	GuiManager.h		GuiManager.cpp
	Mouse.h		Mouse.cpp
	Quad.h		Quad.cpp
	Character.h		Character.cpp
	Font.h		Font.cpp

h	FontBitmap.h	C++	FontBitmap.cpp
h	FontManager.h	C++	FontManager.cpp
h	GuiLoader.h	C++	GuiLoader.cpp
h	TagReader.h	C++	TagReader.cpp
h	TagReaderButton.h	C++	TagReaderButton.cpp
h	TagReaderCheckBox.h	C++	TagReaderCheckBox.cpp
h	TagReaderDropDownBox.h	C++	TagReaderDropDownBox.cpp
h	TagReaderGrid.h	C++	TagReaderGrid.cpp
h	TagReaderImage.h	C++	TagReaderImage.cpp
h	TagReaderListBox.h	C++	TagReaderListBox.cpp
h	TagReaderProgressBar.h	C++	TagReaderProgressBar.cpp
h	TagReaderRadioButton.h	C++	TagReaderRadioButton.cpp
h	TagReaderScrollBar.h	C++	TagReaderScrollBar.cpp
h	TagReaderSlider.h	C++	TagReaderSlider.cpp
h	TagReaderTabControl.h	C++	TagReaderTabControl.cpp
h	TagReaderTextBox.h	C++	TagReaderTextBox.cpp
h	TagReaderWindow.h	C++	TagReaderWindow.cpp
h	Style.h	C++	Style.cpp
h	StyleManager.h	C++	StyleManager.cpp
h	Text.h	C++	Text.cpp
h	TextManager.h	C++	TextManager.cpp
h	Button.h	C++	Button.cpp
h	ButtonBase.h	C++	ButtonBase.cpp
h	CheckBox.h	C++	CheckBox.cpp
h	Container.h	C++	Container.cpp
h	DropableContainer.h	C++	DropableContainer.cpp
h	DropDownBox.h	C++	DropDownBox.cpp
h	Grid.h	C++	Grid.cpp
h	Image.h	C++	Image.cpp
h	ListableItem.h	C++	ListableItem.cpp
h	ListBox.h	C++	ListBox.cpp
h	ProgressBar.h	C++	ProgressBar.cpp
h	RadioButton.h	C++	RadioButton.cpp
h	RadioButtonGroup.h	C++	RadioButtonGroup.cpp
h	Slider.h	C++	Slider.cpp
h	TabControl.h	C++	TabControl.cpp
h	TextBox.h	C++	TextBox.cpp
h	Widget.h	C++	Widget.cpp
h	BorderDecorator.h	C++	BorderDecorator.cpp
h	Decorator.h	C++	Decorator.cpp
h	ScrollBarDecorator.h	C++	ScrollBarDecorator.cpp
h	TitleBarDecorator.h	C++	TitleBarDecorator.cpp

Beschrijving
GUI is in zijn huidige staat crossplatform.

Input -project	In huidige vorm crossplatform: NEE
Maakt gebruik van externe libraries:	• *<Windows.h>* • *<Xinput.h>* • *<Dinput.h>*

Header files:		Source files:	
h	Controller.h	C++	Controller.cpp
h	Keyboard.h	C++	Keyboard.cpp
h	Mouse.h	C++	Mouse.cpp
h	InputManager.h	C++	InputManager.cpp
h		C++	PolemosInputManagerFactory.cpp

Beschrijving
Het input-project maakt gebruik van DirectInput en de windows SDK. Een alternatief voor deze library kan SDL zijn. Het nadeel van het gebruik van SDL voor input is echter wel, dat SDL alleen input kan afvangen van een SDL-window. Er zal dan ook gebruik moeten worden gemaakt van een SDL window.

Internal -project	In huidige vorm crossplatform: JA
Maakt gebruik van externe libraries:	• *<CG>*

Header files:		Source files:	
h	IndexBuffer.h	C++	IndexBuffer.cpp
h	Joint.h	C++	Joint.cpp
h	Material.h	C++	Material.cpp
h	Mesh.h	C++	Mesh.cpp
h	RenderTarget.h	C++	VertexBuffer.cpp
h	VertexBuffer.h	C++	GraphicsShaderCGFX.cpp
h	GraphicsShaderCGFX.h	C++	Shader.cpp
h	Shader.h		

Beschrijving
De shader maakt op dit moment gebruik van CG. CG is op zichzelf crossplatform maar wordt wel anders geïnitialiseerd per renderer-API (zie document:"Polemos Renderer Documentatie")

Math -project	In huidige vorm crossplatform: NEE
Maakt gebruik van externe libraries:	• *<Math.h>* • *<cmath>*

Header files:		Source files:	
h	BoundingBox.h	C++	BoundingBox.cpp
h	Frustum.h	C++	Frustum.cpp
h	Matrix.h	C++	Matrix.cpp
h	Plane.h	C++	Plane.cpp
h	Quaternion.h	C++	Quaternion.cpp
h	Vector.h	C++	Vector.cpp

Beschrijving

Het Math-project bevat de windows SDK. Om deze crossplatform te maken zal voor plane.h en quad-tree een alternatief moeten worden gevonden.

ParticleSystem -project	In huidige vorm crossplatform: JA
Maakt gebruik van externe libraries:	*geen*

Header files:		Source files:	
h	Emitter.h	C++	Emitter.cpp
h	ParticleSystem.h	C++	ParticleSystem.cpp

Beschrijving

Het particleSystem is crossplatform.

ResourceManagement -project	In huidige vorm crossplatform: Ja
Maakt gebruik van externe libraries:	*geen*

Header files:		Source files:	
h	Resource.h	C++	Resource.cpp

h	ResourceLoader.h	C++	ResourceLoader.cpp
h	ResourceManager.h	C++	ResourceManager.cpp
h	Heightmap.h	C++	Heightmap.cpp
h	Model.h	C++	Model.cpp
h	ShaderFile.h	C++	ShaderFile.cpp
h	Texture.h	C++	Texture.cpp
h	ResourceLoaderHeightmap.h	C++	ResourceLoaderHeightmap.cpp
h	ResourceLoaderModelMS3D.h	C++	ResourceLoaderModelMS3D.cpp
h	ResourceLoaderShaderCGFX.h	C++	ResourceLoaderShaderCGFX.cpp
h	ResourceLoaderTexture.h	C++	ResourceLoaderTexture.cpp

Beschrijving

Het resourcemanagement –project is op dit moment crossplatform. Wat er echter wel aan verder geprogrameerd zal moeten worden is path-abstractie. Ook in de resources zelf staan hardcoded verwijzingen naar andere resources.

Scene -project		In huidige vorm crossplatform:	JA
Maakt gebruik van externe libraries:		• *<math.h>*	
Header files:		Source files:	
h	AnimatedEntityModel.h	C++	AnimatedEntityModel.cpp
h	Entity.h	C++	Entity.cpp
h	EntityCamera.h	C++	EntityCamera.cpp
h	EntityCameraFollow.h	C++	EntityCameraFollow.cpp
h	EntityContainer.h	C++	EntityContainer.cpp
h	EntityModel.h	C++	EntityModel.cpp
h	EntitySound.h	C++	EntitySound.cpp
h	Patch.h	C++	Patch.cpp
h	PolemosEntityFactory.h	C++	PolemosEntityFactory.cpp
h	PolemosSceneManagerFactory.h	C++	PolemosSceneManagerFactory.cpp
h	Quad-Tree.h	C++	Quad-Tree.cpp
h	Quad-TreeNode.h	C++	Quad-TreeNode.cpp
h	Scene.h	C++	Scene.cpp
h	SceneManager.h	C++	SceneManager.cpp
h	Sky.h	C++	Sky.cpp
h	ChunkManager.h	C++	ChunkManager.cpp
h	SceneLoader.h	C++	SceneLoader.cpp
h	ChunkManagerAnimatedEntityModel.h	C++	ChunkManagerAnimatedEntityModel.cpp
h	ChunkManagerEntity.h	C++	ChunkManagerEntity.cpp
h	ChunkManagerEntityContainer.h	C++	ChunkManagerEntityContainer.cpp

h	ChunkManagerEntityEmitter.h	C++	ChunkManagerEntityEmitter.cpp
h	ChunkManagerEntityModel.h	C++	ChunkManagerEntityModel.cpp
h	ChunkManagerEntitySound.h	C++	ChunkManagerEntitySound.cpp

Beschrijving

De Scene is niet crossplatform door het gebruik maken van math.h uit de windows SDK.

Scripting-project	In huidige vorm crossplatform: JA
Maakt gebruik van externe libraries:	• *<lua.h>* • *<lualib.h>* • *<lauxlib.h>* • *<luabind/luabind.hpp>*

Header files:		Source files:	
h	LuaVM.h	C++	LuaVM.cpp
h	Script.h	C++	Script.cpp
h	ScriptingManager.h	C++	ScriptingManager.cpp

Beschrijving

Het scripting-project van Polemos bevat de externe library Lua. Lua is een crossplatform API en hoeft dus niet te worden vervangen.

Unittest -project	In huidige vorm crossplatform: JA
Maakt gebruik van externe libraries:	*geen*

Header files:		Source files:	
h	TestRendererDX9.h	C++	TestRendererDX9.cpp
h	TestRendererOGL.h (excluded from build)	C++	TestRendererOGL.cpp (excluded from build)

Beschrijving

De unittests zijn geshreven in puur c++. Om CXXtest te kunnen gebruiken (het testframework) is wel een python interpreter nodig.

Window -project	In huidige vorm crossplatform: NEE

Maakt gebruik van externe libraries:		• *<windows.h>*	
Header files:		Source files:	
h	Window.h	C++	PolemosWindowManagerFactory.cpp
h	WindowManager.h	C++	Window.cpp
		C++	WindowManager.cpp

Beschrijving

Om een window te maken wordt het Window-project van Polemos gebruikt. Deze maakt gebruik van de Windows SDK. Een alternatief voor dit project zou kunnen worden geïmplementeerd met X11.

Crossplatform libraries

Hieronder staat een lijst met mogelijke nieuwe libraries voor Polemos.

OpenGL
OpenGL is op dit moment deels geimplementeerd binnen Polemos.

SDL
Simple DirectMedia Layer is een API die het mogelijk maakt input van Keyboard, muis en joystick af te vangen. Daarnaast kan er ook gewerkt worden met audio en kan SDL een window genereren.

OpenAL
Een alternatieve Audio API. De implementatie van deze library is in theorie niet van toepassing voor Polemos. Polemos gebruikt op dit moment FMOD, een library die ook crossplatform is.

OIS Object Oriented Input System
Een crossplatform Input systeem geschreven in C++ voor keyboard muis en joystick.

X11
Een Window library die gebruikt kan worden onder Windows, MAC OS en Linux. Het voordeel van deze API boven die van bijvoorbeeld SDL is dat deze meerdere windows tegelijk kan maken.

CG
Een crossplatform shader language. Deze library is al geimplementeerd binnen Polemos.

Allegro
Een Library voor graphics, sound, Input (Keyboard muis en joystick) en timers

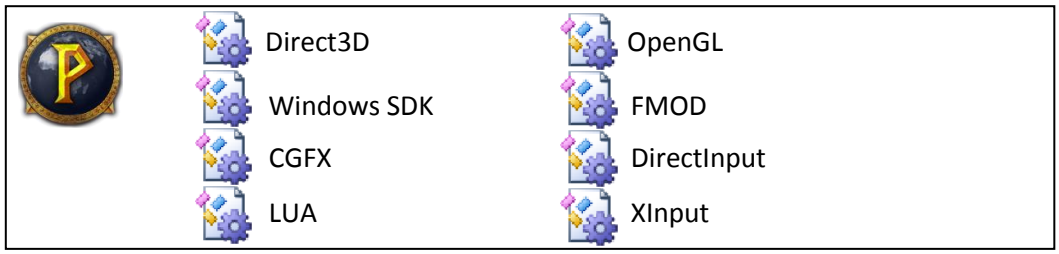
Mogelijkheden

Mogelijkheid 1

Polemos gaat niet crossplatform.

De tijd die het kost om Polemos crossplatform te maken door implementatie van nieuwe libraries is een te tijdrovend proces. Daarnaast is het doel van Polemos om studenten kennis te laten maken met de ontwikkeling van 3D software, het crossplatform maken van Polemos is daarom van ondergeschikt belang.

Een bijkomstigheid van deze keus, is de vier huidige crossplatforme libraries (OpenGL, LUA, CGFX FMOD) hun voordeel om crossplatform te zijn verliezen. Een engine is immers alleen crossplatform als alle libraries crossplatform implementaties hebben. Als er dus toch niet voor een crossplatform implementatie wordt gegaan, kan er beter dieper worden ingegaan op alle APIs van DirectX, deze zullen immers goed met elkaar te combineren zijn.

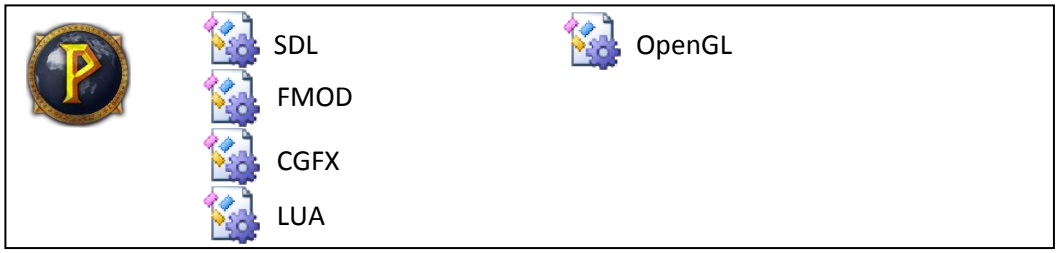


Voordelen	Nadelen
• Er kan meer worden gespitst op DirectX API's	• Polemos kan alleen worden gedraaid op het Windows besturingssysteem
• De onderhoudbaarheid van Polemos blijft gelijk	

Mogelijkheid 2

Polemos gaat crossplatform, Direct3D, DirectInput, XInput en de windows SDK worden eruit gehaald en SDL wordt geïmplementeerd.

SDL bestaat uit meerdere libraries. Deze libraries kunnen Direct3D, DirectInput, Xinput en de Windows SDK vervangen. Op deze manier worden er minder externe libraries in Polemos gebruikt en kunnen de rest van de crossplatform implementaties blijven.

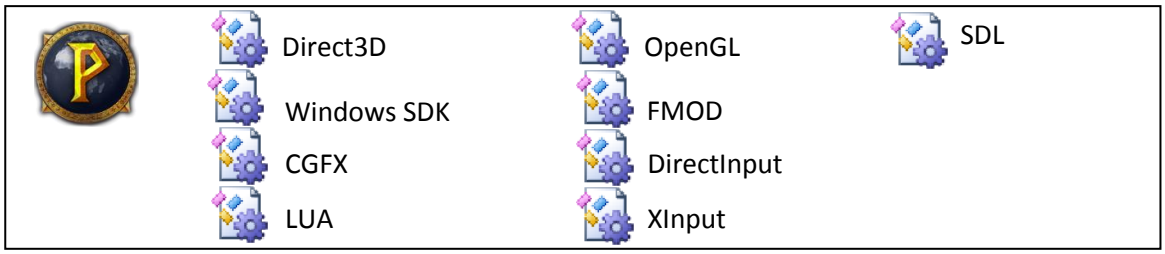


Voordelen	Nadelen
• Polemos kan op meerdere besturingssystemen draaien	• Vincent Broeren moet eerst kennis opdoen van SDL
• Studenten maken kennis met andere libraries	• Geen multiple window support.

Mogelijkheid 3

Polemos gaat crossplatform, Direct3D, DirectInput en de windows SDK worden onderhouden naast andere implementaties die Polemos crossplatform maken.

Deze optie is het meest tijd intensief. Om deze optie te onderhouden is erg lastig. Er zal dan altijd synchroon moeten worden ontwikkeld op meerdere besturingssystemen om er zeker van te zijn dat de engine crossplatform blijft.



Voordelen	Nadelen
• Verschillende	• Moeilijk te onderhouden
• Draait op verschillende besturingsystemen	
• Goede abstractie tussen verschillende implementatie	

Conclusie

Persoonlijk vind ik de volgende mogelijkheid de beste: (mogelijkheid 2 uit hoofdstuk “mogelijkheden”)

Aangezien er op de AICT geen les meer wordt gegeven over het gebruik van besturingssystemen, leren studenten niet crossplatform programmeren. Het 3D blok kan hierin een belangrijke rol spelen in het crossplatform leren programmeren. Als Polemos dus crossplatform zou worden kan dit een groot gat vullen wat nu binnen de AICT speelt. Daarnaast hoeft er maar één extra library te worden geïmplementeerd, namelijk SDL. Een nadeel van de library is dat er maar een windows tegelijkertijd kan worden getoond.

Ook voor de studenten is het een goede stap om kennis op te doen van verschillende libraries in plaats van de DirectX API’s en om crossplatform te leren programmeren.

Daarnaast is SDL een stabiele library die al jaren worden gebruik in professionele games (World of Goo / Unreal tournament / Tribes).

Door een keuze voor crossplatform te gaan, krijgen de huidige implementaties van CGFX, FMOD, OpenGL en LUA ook meer effect. De nadelen van het niet kunnen gebruiken van meerdere windows is een kleine opoffering aangezien deze functionaliteit eigenlijk nooit bij het maken van spellen wordt gebruikt.

Voordelen	Nadelen
• Polemos kan op meerdere besturingssystemen draaien	• Vincent Broeren moet eerst kennis opdoen van SDL (aanpassingen van colleges)
• Studenten maken kennis met andere libraries	• Geen multiple window support.
• Studenten leren crossplatform programmeren	

7. Sprint-backlogs

Sprint-backlog 1

Units voor sprint: 270 (42 units/dag)

volgende sprint: 7 juni

Ingeplande units: 205 losse units: 65

#	Prio	Afh	Units	Requirement
1.	3	-	20	Testen renderer (unittests) – `GraphicsDirectX9::Initialize`
2.	3	-	25	Testen renderer (unittests) - `GraphicsDirectX9::Clear`
3.	3	-	10	Testen renderer (unittests) – `GraphicsDirectX9::BeginScene`
4.	3	-	10	Testen renderer (unittests) – `GraphicsDirectX9::endScene`
5.	3	-	10	Testen renderer (unittests) – `GraphicsDirectX9::present`
6.	3	-	30	Testen renderer (unittests) – `GraphicsDirectX9::createRenderTarget`
7.	3	-	30	Testen renderer (unittests) – `GraphicsDirectX9::SetActiveRenderTarget`
8.	1	-	1	PARAM indexed prim + vertexbuff omgedraaid (offset)
9.	2	#11	10	Diagram huidige implementatie
10.	2	-	20	Diagram structuur OpenGL implementatie
11.	3	-	160	Documentatie OpenGL renderer
12.	1	-	30	Projectsettings (solution Polemos) systeemvariabelen laten gebruiken voor de SDK en CG links (voorkomt de Linking errors bij het installeren van de Polemos Engine)
13.	1	-	20	Gebruik maken van één resource folder die in de root van het project staat om duplicatie van resources te voorkomen
14.	1	#17-23	80	Per window gespecificeerde renderer (OpenGL & Direct3D initialisatie)
15.	2	-	80	Polemos porten naar Visual Studio 2010
16.	4	#1 -7	6	Implementatie OpenGL – OpenGL include
17.	4	#1 -7	200	Implementatie OpenGL – OpenGL initialisatie (GL_initgraphics)
18.	4	#1 -7	150	Implementatie OpenGL – OpenGL Clear (glClear)
19.	4	#1 -7	50	Implementatie OpenGL – OpenGL beginScene (…)
20.	4	#1 -7	50	Implementatie OpenGL – OpenGL endScene (…)
21.	4	#1 -7	150	Implementatie OpenGL – OpenGL present (popmatrix / swapBuffers / bindbuffer)
22.	4	#1 -7	200	Implementatie OpenGL – OpenGL setStreamSource (glVertexPointer/glAttribPointer)
23.	2	-	12	Resource folder flexible maken (nu hard coded op working dir) _probing
24.	4	-	40	Inlezen / runnen / oefenen testframework

Sprint-backlog 2

Units voor sprint:	364 (42 units/dag)	7 juni – 18 juni
open dag 11 juni:	12 units	
Sprint administratie	20	
verslag	24	

Ingeplande units:	360	losse units:	4

#	Prio	Afh	Units	Requirement
1.	3	-	30	Fullscreen window kunnen deleten
2.	4	-	80	Aanmaken OpenGL renderer project files
3.	4	-	20	OpenGL headers includen
4.	4	-	200	`Opengl::Initialize`
5.	3	-	40	`Opengl::Clear`
6.	3	-	60	`Opengl::Present`
7.	2	-	50	`Opengl::CcreateRenderTarget`
8.	2	-	50	`Opengl::SetActiveRenderTarget`
9.	2	-	40	`Opengl::BeginScene`
10.	2	-	40	`Opengl::EndScene`
11.	2	-	50	`Opengl::setTexture`
12.	2	-	80	`Opengl::DrawPrimitive`
13.	1	-	200	`Opengl::setStreamSource`
14.	3	-	30	`documentatie aanpassingen CXXtestframework`

Sprint-backlog 3

totaal aantal units voor sprint: 420 (42 units/dag) 28 juni – 9 juli

afscheid Ton 6 units

6 Juli AICT dicht 42

Planbare units: 366 unitbuffer: 183 (±50%)

#	Prio	Afh	Units	Requirement
1.	1	-	5	Unused shaders opruimen
2.	1	-	10	Shaders laten compileren op laagst mogelijk vertex profile + documentatie
3.	1	-	10	Shaders laten compileren op laagst mogelijk fragment profile + documentatie
4.	3	-	20	"Renderer documentatie document"
5.	3	-	200	Abstractie CG implementatie. Geheel abstract zonder CG includes binnen de renderer
6.	2	-	30	Translation map.

Sprint 8 backlog:

#	Items from product backlog	units
#332	OpenGL::CreateRendertarget	50
#313	OpenGL::clear	50
#310	OpenGL::present	100
#331	OpenGL::setActiveRendertarget	50
#344	Rendererdocumentatie	20
#345	Translationmaps	30
	Totaal: (366)	300

Sprint 8 backlog buffer

#	Items from product backlog	units
#312	OpenGL::BeginScene	50
#311	OpenGL::EndScene	50
#334	OpenGL::DrawPrimitive	200
	Totaal: (183)	300

Sprint-backlog 4

totaal aantal units voor sprint: 1470 (42 units/dag) 12 juli – 27 augustus

Afstudeerverslag	20 units
Niet aanwezig	20 units
Vakantie	1050

Planbare units: 380 unitbuffer: 190 (±50%)

#	Prio	Afh	Units	Requirement
1.	2	-	30	CXXtests schrijven OpenGL renderer
2.	1	-	25	Renderer document 'testen' + verbeteren
3.	1	-	35	Product-backlog prioriteiten opnieuw definiëren
4.	2	-	20	Verschillen OpenGL en DirectX bijwerken

Sprint 9 backlog:

#	Items from product backlog	units
334	OpenGL::Drawprimitive	80
312	OpenGL::BeginScene	50
311	OpenGL::EndScene	50
309	OpenGL::SetStreamSource	200
	Totaal: (380)	380

Sprint 9 backlog buffer

#	Items from product backlog	units
354	CXXtest koppellen aan OpenGLrenderer door middel van static-linking	5
353	renderer documentatie 'testen' en verbeteren	20
350	Textureloading onderzoek (sdl / OpenIl /...)	115
335	OpenGL::"setTexture"	50
	Totaal: (190)	190

Sprint-backlog 5

totaal aantal units voor sprint: 420 (42 units/dag) 30 augustus – 10 september
Afstudeerverslag 200 units

Planbare units: 220 unitbuffer: 200 (±50%)

#	Prio	Afh	Units	Requirement
1.	4	-	100	Adviesrapport Polemos crossplatform
2.	3	-	80	Aanvullen verschillen OpenGL & Direct3D
3.	2	-	3	OpenGL renderer unittesten
4.	3	-	37	Aanvullen OpenGL renderer documentatie

Sprint 9 backlog:

#	Items from product backlog	units
1	Adviesrapport Polemos crossplatform	100
2	Aanvullen verschillen OpenGL & Direct3D (matrices)	80
3	OpenGL renderer unittesten	3
4	Aanvullen OpenGL renderer documentatie	31
5	Code clean (hacks/tests eruit)	6
	Totaal: (220)	**220**

Sprint 9 backlog buffer

#	Items from product backlog	units
1	Projection/view/World matrices transleren/transposen	40
	Totaal: (110)	**40**

8. Product-backlog

Prio	Item #	Name	Description:	Units	Sprint	Done?	Comments
2	358	Projection/view/World matrices transleren/transposen voor OpenGL		40			
4	357	OpenGL renderer unittesten		3	10	y	
4	356	Code opschonen van OpenGL project		6	10	y	
4	355	Advies rapport Polemos crossplatform		10	10	y	
2	354	CXXtest koppellen aan OpenGLrenderer door middel van static-linking		5			
4	353	renderer documentatie 'testen' en verbeteren		25	10	y	
1	352	Product-backlog prioriteiten opnieuw definieren		35			
2	351	Verschillen DirectX en OpenGL bijwerken		20			
3	350	Textureloading onderzoek (sdl / OpenIl /...)	niet meer nodig, is opgenomen binnen adviesrapport crossplatform (#355)	25			
1	349	Unused shaders opruimen		5			
1	348	Shaders laten compileren op laagst mogelijk vertex profile + documentatie		10			
1	347	Shaders laten compileren op laagst mogelijk fragment profile + documentatie		10			
1	346	Abstractie CG implementatie. Geheel abstract zonder CG includes binnen de renderer	Zou het mooiste zijn om de renderer en shaders geheel van elkaar los te krijgen.	200			
4	345	SetTranslationMaps laten werken met DirectX + OpenGL		30	8	y	
3	344	Documentatie Renderer (OpenGL + DirectX)		20	8	y	
1	343	ResourceManager aanpassen. Moet alleen bestandsnaam krijgen en zelf zoeken binnen “ResourceFiles/” op extensie		120			
1	342	Externe resources aanpassen zodat deze linken naar bestandsnamen in plaats van een relatief pad + bestandsnaam		40			
1	341	Alle calls binnen de engine naar resourceManager gebruik laten maken van nieuwe ‘call’ style resourceManager (item #342)		40			
1	340	GUI-project (parsers) gebruik laten maken van de resourceManager		25			
1	339	ResourceFiles controleren dat geen enkele file dezelfde naam heeft		2			
3	338	Fullscreen window kunnen deleten		30	7	y	
4	337	Aanmaken OpenGL renderer project files		80	7	y	
3	336	Documentatie aanpassingen CXXtest framework		30	7	y	
2	335	OpenGL::”setTexture”		50			
2	334	OpenGL::” drawprimitive”		80	9	0.5	
2	332	OpenGL::”createRenderTarget”		50	8	y	
2	331	OpenGL::”SetActiveRenderTarget”		50	8	y	
3	330	Testen renderer (unittests) – GraphicsDirectX9::Initialize		20	6	Y	

3	329	Testen renderer (unittests) - GraphicsDirectX9::Clear		25	6	Y	
3	328	Testen renderer (unittests) – GraphicsDirectX9::BeginScene		10	6	Y	
3	327	Testen renderer (unittests) – GraphicsDirectX9::endScene		10	6	Y	
3	326	Testen renderer (unittests) – GraphicsDirectX9::present		10	6	Y	
3	325	Testen renderer (unittests) – GraphicsDirectX9::createRenderTarget		30	6	Y	
3	324	Testen renderer (unittests) – GraphicsDirectX9::SetActiveRenderTarget		30	6	Y	
1	323	PARAM indexed prim + vertexbuff omgedraaid (offset)		1			
2	322	Diagram huidige implementatie		10	6	Y	
2	321	Diagram structuur OpenGL implementatie		20	6	n	
3	320	Documentatie OpenGL renderer		160			
1	319	Projectsettings (solution Polemos) systeemvariabelen laten gebruiken voor de SDK en CG links (voorkomt de Linking errors bij het installeren van de Polemos Engine)		30			
1	318	Gebruik maken van één resource folder die in de root van het project staat om duplicatie van resources te voorkomen		60		Y	
1	317	Per window gespecificeerde renderer (OpenGL & Direct3D initialisatie)		80			
2	316	Polemos porten naar Visual Studio 2010		80			
4	315	Implementatie OpenGL – OpenGL include		6	7	y	
4	314	Implementatie OpenGL – OpenGL initialisatie (GL_initgraphics)		200	7	y	
4	313	Implementatie OpenGL – OpenGL Clear (glClear)		50	8	y	
4	312	Implementatie OpenGL – OpenGL beginScene (…)		50	9	y	
4	311	Implementatie OpenGL – OpenGL endScene (…)		50	9	y	
4	310	Implementatie OpenGL – OpenGL present		100	8	y	
4	309	Implementatie OpenGL – OpenGL setStreamSource (glVertexPointer/glAttribPointer)		200	9	n	
2	308	Resource folder flexible maken	(nu hard coded op working dir) _probing	12		y	
4	307	Inlezen / runnen / oefenen testframework		40	6	n	
4	306	Particlesystem documenteren		40	5	Y	
4	305	Game documenteren		40	5	Y	
1	304	pointscaling uit de renderer halen	werkt niet omdat het fixed function is	10			
3	303	particle system entity toevoegen aan sceneeditor	de editor moet particle systems kunnen toevoegen aan de scene	80	5	Y	
2	302	waterval toevoegen aan game		40			
2	301	waterval maken		40			
4	300	particle systems toevoegen aan game		40	5	Y	

4	299	selected shader mooi maken		2	4	Y	
3	298	performance van spawnen van particles verbeteren	Spawnen is nu acceptable snel maar is momenteel de bottleneck van het particlesystem	40			
4	297	Resourcemanagement documenteren		40	5	Y	
2	296	Engine niet meer initen via script	Er zijn delen die in c++ geinit moeten worden en dan is het beter om alles bij elkaar te houden ipv het ene in c++ en het andere in lua	12			
2	295	particle system documenteren	comments en handleiding	0	5	Y	dubbele taak met #306
2	294	mutiple render targets inbouwen	MRT zorgt ervoor dat er minder renderstates e.d. geset dienen te worden.	30			
1	293	gui moet vanuit runtime widgets kunnen toevoegen / verwijderen	op het moment moeten alle widgets in de pml vermeld worden, dit is onhandig voor bijvoorbeeld het aanleren van nieuwe skills of het toevoegen van items aan een inventory/lootwindow	80			
2	292	onderzoek doen naar interfaces in cgfx	deze zijn handig om bijvoorbeeld bij particle system te gebruiken	40			
3	291	rendertargets ombouwen zodat ze niet meer gekeyed worden op string	het kan net als bijvoorbeeld Texture gemaakt worden	30			
4	290	particle system gebruik laten maken van nieuwe shader systeem		60	5	Y	
8	289	Saven van scene debuggen en afmaken en in scene editor bouwen		80	5	Y	
4	288	het inventory window moet leeg beginnen	er zijn nu een aantal items standaard ingevuld in het inventory scherm terwijl de speler deze helemaal niet bezit	6			
8	287	na #286 als er iets toegevoegd wordt aan de speler's inventory moet het inventory window geupdated worden.		12			
8	286	als er op een item in het lootwindow geklikt wordt moet het item toegevoegd worden aan de speler's inventory.	weten nog niet hoe widget callbacks werken (of ze er uberhaupt zijn), zou via script kunnen maar daar is geen noodzaak voor	40			
8	285	na #284 als lootwindow wordt weergegeven moet hij geinit worden met de juiste items	omdat alle items in het lootwindow moeten staan, moeten de items die niet in de lootbox zitten worden verborgen	18			
8	284	als lootbox geopend wordt moet het lootwindow worden weergegeven		6			
4	283	lootwindow widget toevoegen aan pml file	hierin moet al ruimte zijn voor alle mogelijke items die gedropt kunnen worden	18			

5	282	texture unload bug fixen	als de applicatie(entitycamerafollowtest) af sluit crasht de applicatie, missende texture?	18			
3	281	AI besturing van enemy moet opgeruimd worden nadat hij niet meer nodig is	bijvoorbeeld nadat de lootbox despawned	24			
4	280	geen loadvertexbuffer meer doen in setvertexbuffer	Dit breekt waarschijnlijk de gui dus die moet ook aangepast worden.	80			
3	279	zorgen dat de speler dood kan gaan	de speler kan nu wel 0 hp berijken maar dan moet hij nog dood gaan ook, wellicht respawnen?	200			Dubbele taak.. zie #217
8	278	crash met selectie van dode/dood gaande enemies opzoeken en eruit halen	De game crashed soms met een onbekende combinatie van selecteren van dode of dood gaande vijanden.	12			
2.5	267	timers moeten events kunnen vuren		18			
3	266	Range op attacks		6	4	Y	
2	266	rendererdirectx rekening laten houden met pure-device	Hij moet bij de get methods of gebruik maken van state blocks of hij moet een error geven als hij op pure-device staat	??			
3	265	Model op de heightmap zetten ipv op de hoogte van de camera in de scene editor		18			
	264						
1	263	Treeview voor resources selecteren in scene editor		40			
1	262	Model preview in scene editor	Als je een model selecteerd om in de scene te plaatsen moet je een preview krijgen	200			
7	261	Bounding boxes van entity meeverplaatsen	Als in de editor een entity verplaatst wordt moet de bounding box meeverplaatsen.	40			
3	260	warnings van scipting pdb files weghalen		40			
4	259	Matrix *= operaor overloaden		6	4	Y	
4	258	Particle system uitbreiden zodat hij meer kan dan alleen een fontein	vuurbal + healing + levelup	160	4	Y	
9001	257	Renderstate manager voor cgfx maken	Renderstates in cgfx zijn zo traag dat nvidia adviseert dit zelf te managen voor performance	120			
3	256	Particlesystem schrijven		160	3	Y	
3	255	Particlesystem design		40	3	Y	
3	254	Rendertarget fixen zodat er meer als één mogelijk is.		30	3	Y	
3	253	Point sprite support in engine		20	3	Y	
3	252	Floating point texture/rendertarget support in engine		10	3	Y	
3	251	Scene Editor entity beweeg pijltjes laten	Zodat je als een een entity	80			

		werken	aanklikt pijltjes krijgt op elke as waarmee je de entity kan verschuiven				
8	250	game werkend maken met het nieuwe shader systeem		40	4	Y	
8	249	shader interface opschonen	De interface van de polemosgraphicsshader moet opgeschoond worden zodat de methodes die niet meer gebruikt worden ook niet meer aanwezig zijn	40			
8	248	het overige gedeelte van de engine gebruik laten maken van het shader systeem	samen met task #249	80	4	Y	
8	247	een test werkend maken met het nieuwe shader systeem	het gaat dan om entitymodel, sky, button, guiloader, container, progressbar en font. Deze worden allemaal gebruikt in de test	40	3	Y	
8	246	shaders parameters toe laten voegen op load time en niet meer mappen op naam	de parameters die gebruikt worden door de shaders moeten loadtime opgezocht worden, runtime kan dan de index van de parameter in de lijst gebruikt worden om deze parameter aan te passen	40	3	Y	
3	244	resource manager of resource loaders moeten checken of de file bestaat. Alleen in Debug	de ResourceLoaderShaderCgFX bijvoorbeeld controleert niet of de shader bestaat dus krijg je gewoon een shaderfile terug waarvan geen shader gemaakt kan worden	20			
1	243	Onderzoek doen naar hoe veel werk het is om streaming uit de engine te halen.		40			
2	242	GetShader methode in EntityModel maken	zodat we in de game weten welke shader we op moeten ruimen, normal of selected shader	3			
2	241	het spawnpoint dat in gameobjectcontrollerentity bijgehouden wordt moet relatief worden aan het spawnpoint	dit komt omdat bij streaming alle locaties worden geupdated maar niet de eerder opgevraagde locatie die als spawnpoint gebruikt wordt.	40	5	Y	
0	240	companion pet maken die de speler volgt	is awesome als je een puppy of een lolcat hebt die je volgt	40			
1	239	experience bar de experience van de player weer laten geven	na #237	6			
1	238	weergave van target's health maken dmv gui		18			
1	237	experience bar toevoegen aan de gui	Er moet een weergave van de speler's experience komen dus daarvoor moet eerst een widget toegevoegd worden	0			

2	236	level van enemies die gespawnd worden aanpasbaar maken	spawnpoint spawnt alleen maar level 1 monsters, dit moet instelbaar worden	3			
2	235	Windowgrootte aan Gui shader meegeven		20			
B	234	Window grootte niet overal als magic numbers gebruiken	Deze data zou in op een config class opgehaald kunnen worden of gewoon bij windowmanager	20			
1	233	De speler's inventory moet weergegeven worden dmv de gui	De items die door de speler verzameld zijn moeten weergegeven worden in het spel	20		Y	
2	232	De speler moet een inventory krijgen	De speler moet items kunnen verzamelen dus er moet een verzameling aan items zijn die de speler al verzameld heeft	18			
8	229	management dat over shaders heen zit er uit halen	de GraphicsShaderCGFX managed de shaders, de shaders moeten dit zelf doen	40	3	Y	
2	227	Als een monster gedood is moet hij een kistje droppen in de kistjes die door monsters gedropt worden moet een item zitten als er op het kistje geklikt wordt moet de open animatie afgespeeld worden	De game moet uitgebreid woorden met lootbox objecten, zo'n lootbox moet in de wereld weergegeven worden nadat een monster gedood is	58	4	Y	
2	226	kernel initialize gebruiken	op het moment wordt alleen de audiomanager in de initialize geinitializeerd, kijken naar wat er nog meer hier in moet, guimanager / scripting manager bijvoorbeeld?	12	4	Y	
2	225	vuur spreuk dmg laten doen healing spreuk moet hp geven	als de vuur spreuk geactiveerd wordt moet hij schade doen als de healing spreuk geactiveerd wordt moet hij hp terug geven	18			
2	224	global cooldown inbouwen	ervoor zorgen dat de global cooldown synchroon loopt met de animaties en ervoor zorgen dat als de global cooldown nog niet voorbij is de speler niet kan aanvallen	24	4	Y	
2	223	als je een potion gebruikt moet je hp terug krijgen		4			
1	222	kernel::cleanup just needs to destroy the instance, not all variables	opruimen hoort in de destructor van de instance te gebeuren, niet in de cleanup zelf	12			
1	221	gui vuur knopje ervoor laten zorgen dat een vuur spreuk gecast wordt. gui healing knopje moet healing spell casten	het knopje voor de fireblast die in de gui staat moet de vuur spreuk activeren het knopje voor de healing die in de gui staat moet de healing spreuk activeren	4			

2	220	de speler een verzameling aan spells geven	er moet een verzameling aan spells komen waarvan de speler de fireblast en de healing spreuk kent	80			
1	219	gui aanvals knopje ervoor laten zorgen dat de speler aan valt	het melee aanvals knopje op de gui moet een melee aanval activeren	2			
2	218	speler´s hp aan de gui koppelen	de progressbar die staat voor de speler´s hp moet ingevuld worden met de speler´s hp	20	4	Y	
2	217	de speler moet dood kunnen gaan	op het moment verliest de speler wel hp maar gaat hij nog niet dood als hij 0 hp heeft. Dus doods animatie en dan respawnen met full health.	4			
0	216	callback maken vanuit scene naar de game als entities verwijderd worden	streaming verwijdert entities, als de game nog pointers naar deze entities heeft worden deze ongeldig. Met een callback kan de game eerst zijn pointers vrij geven waardoor er geen gebruik gemaakt wordt van ongeldige pointers	80			
1	215	terrain collision entitycamerafollow	zodat je niet onder het terrein kan komen met de camera.	20			
3	214	wow models meshes mergen	als het kan (1 texture/meerdere meshes?) model's meshes mergen. Dit zorgt er voor dat het tekenen van models lichter wordt	20			
2	213	wow model rip guide updaten	toevoegen van: sounds, gui textures, wmo's	40			
2	212	terrein in scenefile implementeren	paden naar splat/detail/light textures moeten in de scenefile komen	20		Y	
4	211	Alles wat gemaakt is op dvd branden		20			
1	210	dobbelsteen in de engine maken		2			
3	209	Scripts aan guiwidgets hangen	Zodat knopjes ook een effect kunnen hebben	120			
2	208	Interface boven gui refactoren	PolemosGuiManager	80			
2	207	pickray fixen	pickray houd geen rekening met een draaiende camera of draaiende modellen	160			
2	206	Pickray naar util verplaatsen	de pickray klasse staat nu wel in het util project maar nogsteeds in de math folder	3			
3	205	Bepalen welke widgets we nodig hebben voor de game		20	2	Y	
3	204	Gui goed laten omgaan met resources	in en uitladen	40	2	Y	
3	203	Gui in kernel inbouwen		2	2	Y	

2	202	Hardware cursor debuggen		40		Y	
3	201	Gui aanpassen zodat we de nodige widgets hebben voor de game	na taak 205	40	2	Y	
4	200	Renderen van GUI debuggen		40	2	Y	
2	200	rendertarget queue maken	er is nu geen ondersteuning om meerdere rendertargets te gebruiken, de scene set een rendertarget, hierin kan er dan geen andere entiteit zijn die een rendertarget set omdat dan het systeem in de renderer niet meer werkt.	40			
4	199	Het mysterie van waarom GUI geen widgets heeft na guimanager->load(); oplossen		80	2	Y	
2	198	Per pixellighting voor models		18			
3	197	Respawn systeem maken	Zodat entities en player kan respawnen na dat ze dood zijn gegaan	40	2	Y	
3	196	Gebouw modellen rippen		40	2	Y	
2	195	Water in meer maken	Translucent plane met bewegende textures	80	2	Y	
2	194	Auto walk		6			
0	193	Click to walk	Diablo style input. Als je ergens klikt dat je daar naartoe beweegt. Dit betekend dat we moeten raycasten op terrein.	100			
4	192	NPC enemies laten aanvallen als ze binnen attack range zijn.		6	2	Y	
4	191	NPC enemies naar je toe laten kopen om je aan te vallen als je binnen range bent.		18	2	Y	
3	190	NPC's random laten rondlopen.		18	2	Y	
2	189	Terrein collision aanpassen zodat het werkt met zwaartekracht uit taak 189	Normals genereren zodat de player niet op bergen kan springen	40			
2	188	Zwaartekracht in game bouwen (velocity, acceleration, springen)		20			
8	187	Game solution die werkt op alle pc's maken.		6	2	Y	
3	186	Handleiding voor wow models rippen schrijven		12	2	Y	
8	185	Huidige staat van het game project nakijken		12	2	Y	
3	184	Een levelup of ander character development systeem ontwerpen		60	2	Y	
3	183	Ontwerpen welke stats characters krijgen en wat die gaan doen.		60	2	Y	
4	182	Shaders laden en uitladen fixen		200			
2	181	Nieuwe resourcemanagement commenten	resourcemanagement van taak 114	60			
2	180	Nieuwe resourcemanagement testen	resourcemanagement van taak 114	40			
2	179	Gui met lua configuratie files laten werken		80			
4	178	Gui met nieuwe input laten werken	nu niet gehacked	40			

2	177	Sounds verzamelen		40	2	Y	
2	176	Muziek verzamelen		40	2	Y	
4	175	Input bug fixen	Input krijgt soms key ups of downs niet. Dit komt door lage fps. De buffer wordt niet snel genoeg verwerkt. Dus deze taak is opgelost zodra er meer fps is	80			
2	174	Alle heightmaps preloaden	Alle heightmaps loaded zijn net 1 mb in het geheugen maar door ze te preloaden zal streaming een stuk soepeler gaan.	20			
3	173	Bounding box collision op models	Boundingbox collsion moet opnieuw geschreven worden en er moet physics aan gebonden worden.	80	2	Y	
3	172	Zorgen dat entities niet op steile bergen kunnen lopen	Een stukje voor de entities kijken op hoogte en bij een te grote hoogte verschil niet verder laten lopen.	12	3	Y	
3	171	Mist voor de scene verbeteren		12	2	Y	
1	170	Game handleiding maken		80			
3	169	Spells maken met scripts	Spells die besproken zijn.	40			
3	168	Items maken met scripts	Voor nu alleen healing potions	20			
2	167	NPC's toevoegen aan scene	Voor nu alleen npc's laten spawnen door een hardcoded spawnpoint. Met de scene editor kunnen later decoraties toegevoegd worden	12			
2	166	Inventory maken	Op een manier dat de speler iets kan met de items in de inventory. Dus iets linken aan een GUI.	20			
2	165	Communicatie van questgever maken in game	Moet eigenlijk met gui maar die werkt waarschijnlijk nog niet helemaal.	40	2		
2	164	Sounds aan acties binden in game	attack maakt geluid + gui click	12	3	Y	
4	163	Muziek in game bouwen		12	3	Y	
4	162	Oude quest systeem onderzoeken		40	2		
3	161	Animaties aan acties binden in game		12	2	Y	
3	160	Weergave van target maken in game	Rode outline of iets dergelijks	12	2	Y	
4	159	Raycasting in game bouwen	Zodat je iets kan targetten	12	2	Y	
1	158	Bilinear filter voor patch maken	Terrein vertoont trapjes. Een filter bij het aanmaken van de vertices kan dit verhelpen.	40			
6	157	Refactor shader implementatie	Er wordt niet goed omgegaan met parameters, shader en shaderfile.	400			is al gedaan?

2	156	Scene editor api docs in resource package toevoegen na dat resource management is herschreven.		30			
2	155	er moet niet geprobeerd worden om patches die niet bestaan in te laden		12	2	Y	
3	154	16 bit terrain bestandsformaat implementeren	PNG bijvoorbeeld. Hierdoor wordt het terrein een stuk vloeiender	80			
3	153	ShaderParams gebruiken in patch	Meer gebruik maken van shaderparameters, bijvoorbeeld bij terrain die allemaal dezelfde base textures hebben.	200			
1	152	materials goed gebruiken	meshes bevatten een material die voor die mesh's vertexbuffer gebruikt moet worden, dit gebeurt nog niet, op het moment heeft bv de EntityModel de shader en die wordt gebruikt om alle meshes te tekenen	200			
3	151	oude sound gebeuren uit de engine halen	het oude geluids project en de daarbij horende interfaces in core moeten verplaatst worden naar ongebruikte code zodat er ruimte is voor het nieuwe audio project	12	2	Y	
2	150	beter onderscheid maken tussen geanimeerde en niet geanimeerde models	alle models hebben nu bijvoorbeeld een array aan joint matrices, zelfs als er 0 joints zijn	40	2		
2	149	PreviousPosition uit EntityModel halen	Dit was gemaakt voor wannabe collision, of collision moet er helemaal in of helemaal niet, nu is het een onnodig aanhangsel	6	2	Y	
4	148	methodes van entities in core niet meer dubbel interfacen	bijvoorbeeld methodes die uit PolemosEntity komen niet nog een keer in PolemosEntityModel zetten. Is nu tijdelijk gefixt om minder units te gebruiken in task #13. MOET dus of alsnog doorgevoerd worden of gedeleted.	2 of 6			
1	147	in animatie van models, per mesh joints setten?		18			
3	146	grootte van matrices die meegestuurd worden als joints door animatie optimaliseren	er zit data in de matrix die we niet nodig hebben, checksum 0, scale?	40			
2	145	voorbeeldtest maken voor standaard implementatie audio	Er moet een test in het development project komen voor het gebruik van audio.	96			
3	144	documentatie standaard implementatie audio	De standaard implementatie van audio moet gedocumenteerd	96	3	Y	

			worden, het gaat dan om comments				
8	143	Game eisen vastleggen in document.	Eisen staan momenteel op papier, er dient een formele document an gemaakt te worden om	20	1	Y	
8	142	Ontwerp maken van Game	Een formele ontwerp is misschien verreist om het ovverdraagbaar en uitbreidbaar te maken.	200	1		
4	141	GUI fixen	Voor het game hebben we een GUI nodig of iets wat erop lijkt.		2	Y	
3	140	Extra models voor game extracten/vinden.	Voor het game zijn er meer geanimeerde models nodig als wat we nu hebben.	160			
4	138	Uitwerking basis model audio in code	Het basis ontwerp van audio moet geimplementeerd worden	144	3	y	
1	137	Lua dependencies in correcte mappen stoppen	Momenteel worden alle files van lua in de lib folder gestopt, er dient misschien gekeken te worden naar waar ze wel horen.	18			Deze kan waarschijnlijk niet vanwege interne dependencies in lua
B	136	Terrain artefacten in releasemode bug bugfixen	In releasemode worden er extra driehoeken getekend die er niet horen.	80			Is waarschijnlijk een probleem van initialisatie. Op release mode worden variabelen niet standaard op 0 geinitaliseerd
4	135	Dependency op MSCRT8D.DDL eruit halen.	In release mode blijft er een dependency op de debug dll.	80			
3	134	CPU profilen	Bottleneck eruit halen.	200			Er is ge profiled en performance is een stuk beter. CGFX state manager zou wel noeg moeten helpen
1	133	Irritante print in release mode fixen.	Engine print shader loaded berichten te vaak uit. Probleem van loaden opl. niet de print eruit	18			
2	132	Grote gebruik VRAM nalopen.	Engine VRAM gebruik loopt op tot 300Mb	80			
3	131	Tweede pass uit sky halen.	Sky heeft een verouderde "bugfix" die een extra pass vereiste	6			
4	130	Nieuwe Resourcemanager package aanmaken.	voor overstap	3	1	Y	
4	129	Oude loaders converteren naar nieuwe ontwerp		40	1	Y	
3	128	memory allocatie door string opsporen en verminderen	const & + string zelf	80	1	Y	
4	126	Uitwerking basis ontwerp audio		40	1	Y	
8	125	Scene editor bestaande functionaliteit laten werken met nieuwe engine libs	Scene laden, entities plaatsen, verplaatsen,	160	3	Y	

			scalen, roteren en naam aanpassen.				
2	124	Scene editor API documenteren		80			
2	123	Scene editor CLI koppeling API documenteren		80	1	Y	
2	122	Scene editor documenteren		80			
3	121	Scene editor CLI koppeling documenteren		80	1	Y	
2	120	Scene editor naar code conventie brengen		80	5	Y	
3	119	Scene editor CLI koppeling naar code conventie brengen		80	1	Y	
2	118	project files opschonen	powerpc en project settings	200			
4	117	allocatie fix graphicsshadercgfx	per matrix die geset wordt worden 16 floats aangemaakt en opgeruimd wellicht matrix aanpassen zodat de eerste float waarde gebruikt kan worden voor het address van het begin, er moet gelet worden op row/column major	2	1	Y	
4	116	allocatie fix camera	entitycamera build in de game loop een nieuwe viewmatrix, hierbij maakt hij steeds een nieuwe matrix aan beter zou zijn om alleen opnieuw te builden als de camera bewogen is en dan niet een nieuwe matrix aan te maken maar de waardes van de oude matrix overschrijven	6	1	Y	
4	115	allocatie fix quadtree	AddEntitiesToRenderlist kopieert zijn entity vector alvorens hier doorheen te lopen en toe te voegen. de enst van dit probleem verergert met het aantal entities dat in deze node zit	1	1	Y	
4	114	Nieuwe ontwerp resource management implementeren.		120	1	Y	
4	113	Ontwerp Resource management.	Ontwerp maken die de gekozen oplossingen implmenteerd.	18	1	Y	
4	112	Onderzoek mogelijke oplossingen Resource management.	Worden meerdere oplossingen gezocht voor de problemen die gevonden zijn in 111.	18	1	Y	
4	111	Onderzoek problemen resource management.	Onderzoeken wat voor problemen er zijn met de huidige implementatie van resource management.	12	1	Y	
2	110	Undo /Redo in scene editor bouwen	Command pattern is een goede manier om dit te implementeren	120			
3	109	Lijnen uit het terrein halen	De texture atlas van de pixelshader in taak 89	160			

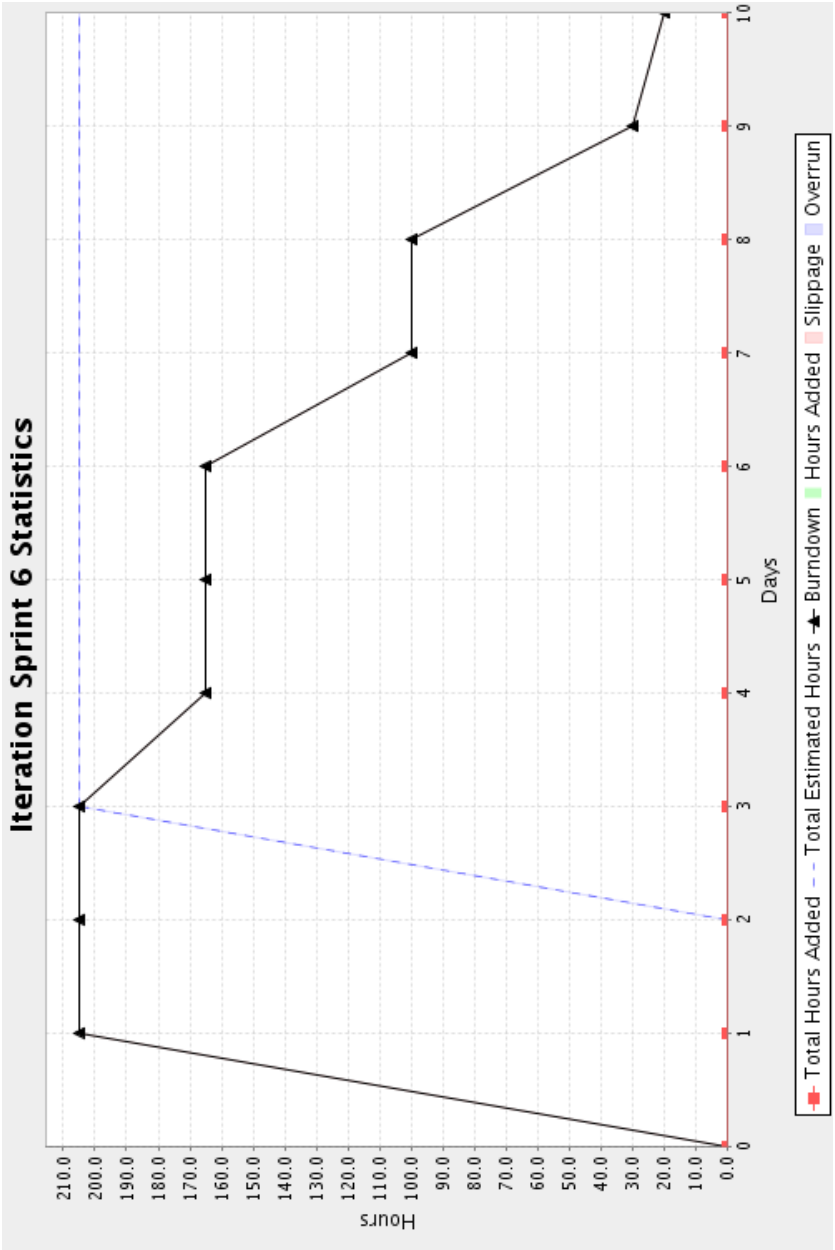
			produceert lijnen met linear filters				
3	108	Hardcoded paden naar terrain textures in Scene::LoadPatch() opschonen	Nieuwe paden staan nog niet in de scenefile. Als de nieuwe scenfile wordt gemaakt kan deze taak gedaan worden.	12			
9001	107	Shader systeem ombouwen zodat FindParameter niet zo zwaar is.	Moet misschien wachten op nieuwe resource systeem.	240	3	Y	
1	106	Lenght methode in vector moet length heten		1	2	Y	
1	105	ToIndentity methode in matrix, moet ToIdentity heten		1	2	Y	
3	104	Particlesystem mogelijke uitbreidingen onderzoek.			0	Y	
3	103	Particlesystem optimalisatie onderzoek			0	Y	
3	102	Particlesystem mogelijke effecten onderzoek			0	Y	
3	101	Particlesystem renderen onderzoek			0	Y	
3	100	Particlesystem particle representatie onderzoek			0	Y	
3	99	Particlesystem emitter type onderzoek			0	Y	
3	98	Particlesystem ontwerp onderzoek			0	Y	
3	97	Patch met nieuwe pixelshader laten werken	Nieuwe texture splatting volgens task 89		0	Y	
3	96	Onderzoek naar sound: -welke API's zijn er? -standaarden voor de implementatie? -is het bestaande project nog (deels) bruikbaar? -hoe kan het goed geimplantereerd worden in de engine?			0	Y	
8	95	Scene editor naar de nieuwe scene file format laten schrijven.	Scene serialiseren en dan schrijven naar de scene file.	200	4	Y	
2	94	Interfaces van scene editor gelijk maken aan nieuwe polemos interfaces.			0	Y	
4	93	Onderzoek naar wat aangepast moet worden in de scene file.		40	3	Y	
8	92	Onderzoek hoe streamende scene kan worden geserialiseerd.		80	4	Y	
2	91	Onderzoek naar werking van oude scene editor.			0	Y	
3	90	Heightmaploader uit grote files laten lezen.	Leest nu uit kleinde files per patch. Wordt een aantal grote files waaruit een deel wordt gelezen per patch		0	Y	De patch leest uit een deel van de heightmap die door de loader wordt geladen. Loader leest niet een deeltje uit maar alles.
3	89	Nieuwe terrain pixelshader schrijven.	Texture splatting met 2 splat textures, 8 base textures in 2 files (4 blokjes per file), 8 detail textures in 2 files (1 per color channel)		0	Y	
2	88	Animatie update niet in render.		18			

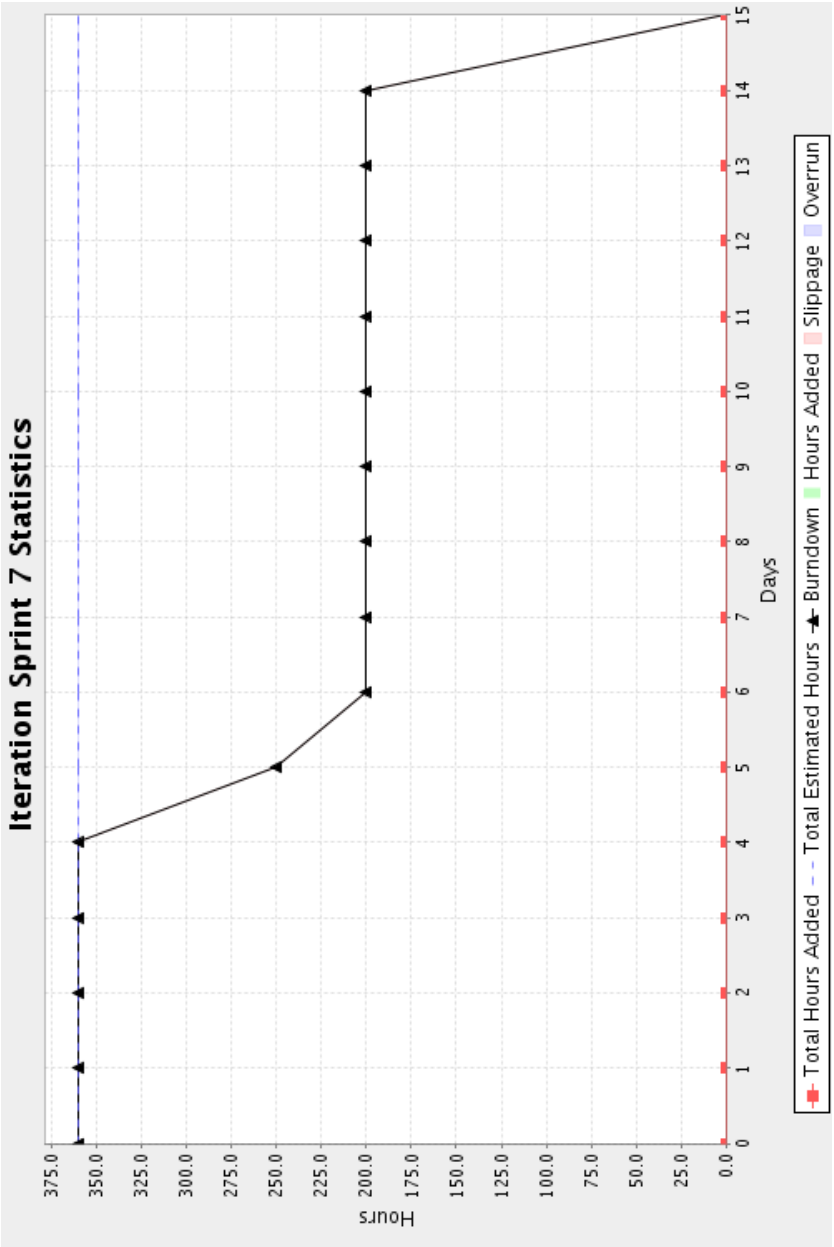
1	82	entitycontainer moet null kunnen returnen bij getentity, wel warning geven		12			
1	81	Int type in chunks moet een char worden		12			
3	80	Event systeem eruit halen			0	Y	
2	79	Animatie project nakijken		40			
1	78	Ongebruikte files weghalen		6			
1	77	Namen projecten fixen		20			
1	76	Namespaces toevoegen		20			
2	75	Scenes project refactoren			0	Y	
1	74	Shaders lostrekken van renderer			0	Y	
2	73	Onnodige lagen in renderer fixen			0	Y	
2	72	Window project refactoren			0	Y	
1	71	Kleinere scene om te testen		40			
2	70	Scenetransitie moet weg			0	Y	
3	69	Indoor Scene weggooien			0	Y	
1	68	rendererdirectx9::loadindexbuffer heeft code duplicatie in zichzelf		18			
1	67	matrix moet conform directx worden, gaat om translate, rotate, scale		20			
2	66	code duplicatie in switch en recurseprocess in quadtreenode			0	Y	
3	65	load methodes inbouwen			0	Y	
2	64	geen gebruik maken van pointers waar niet nodig(quad-tree)		18			
2	63	quadtree directions in types/defines zetten en gebruiken in scene/quadtree			0	Y	
1	62	RendererOpenGL map verwijderen, niet deleten		6			
1	61	vertex, colorargb, texturecoords in types/defines stoppen		12			
1	60	quaternion verwijderen, niet deleten		6			
1	59	polemosscenefactory verwijderen, niet deleten		0	???		die hebben we niet meer?
2	58	polemosoutdoorscene over laten erven van polemosscene, en renamen		20			
2	57	PolemosGraphics reloadshader toevoegen		20			
2	56	oplossing bedenken van hoe 1 matrix in een entity, ipv abs/relative		18			
1	55	yaw, pitch, roll moet rotate Y, X, Z worden			0	Y	
2	54	SetTranslation moet setposition worden			0	Y	
2	53	OverridedefaultShader moet SetShader gaan heten			0	Y	
1	52	PolemosEntityFactory uit project verwijderen, niet deleten			0	Y	
2	51	per implementatie van entity type returnen, geen private var			0	Y	
2	50	PolemosEntity commented constructors deleten			0	Y	
1	49	PolemosEngine .h/.cpp verwijderen, niet deleten			0	Y	
1	48	Defines.h en Types.h samenvoegen in			0	Y	

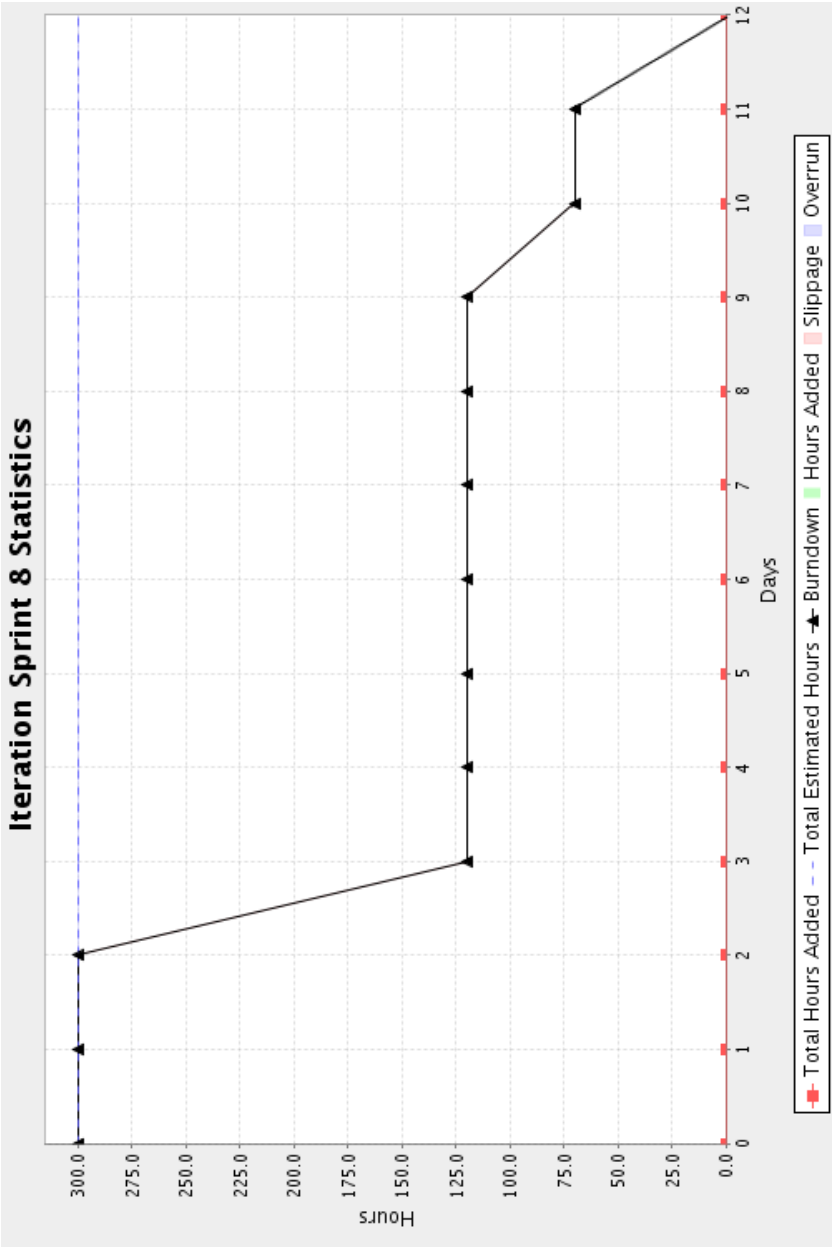
		Defines.h					
1	47	Matrix operatoren toevoegen, uitbreiden met bv .ToString		40			
2	46	Vector operatoren toevoegen			0	Y	
1	44	Addpatch in quadtree dient setpatch the heten		1			Add patch is toch logischer omdat je niet een patch toevoegt aan een specifieke node maar de patch boven in de tree naar binnen gooit en de quadtree zelf de juiste node kiest aan de hand van de locatie. Je voegt dus een patch toe aan de quadtree dus AddPatch
3	43	UnloadScene gaat er vanuit dat de scene bestaat			0	Y	
2	42	Functienamen moeten een volledige beschrijving worden		20			
3	41	Voorkomen van entities met dubbele namen			0	Y	
4	40	Scene update laten gebruiken			0	Y	
2	39	Streaming verbeteren			0	Y	
2	38	Dynamische models tegen frustum checken			0	Y	
1	37	Polemosentitycontainer structuur verbeteren		3600			
3	36	Castray verbeteren			0	Y	
2	35	Quadtree kan camera/frustum aanpassen			0	Y	
2	34	Quadtree kent camera			0	Y	
3	33	Gebruik quilts OF leafs			0	Y	
1	32	Attribuutnamen in scene verbeteren			0	Y	
2	31	Oude chunksysteem behalve shaders eruit halen			0	Y	
1	30	Niet onnodig resources met new aanmaken		20			
1	29	Chunkreaders mogen hun types niet gehardcoded hebben			0	Y	
2	28	Chunkreaders moeten entitylijst niet leeg kunnen gooien	er kan een apparte lege vector worden meegegeven die gekopieerd wordt naar de totale lijst.	20			
1	27	Minimale includes		40			
1	26	Chunkreader entitycontainer fixen	de weirdness die er nu in zit eruit halen	80			
2	25	Leaks eruit halen bij het falen van inladen van een scene			0	Y	
1	24	Skybox loader maken		40			Sky kan zichzelf laden zolang hij het pad naar een model krijgt. Dit pad kan in de

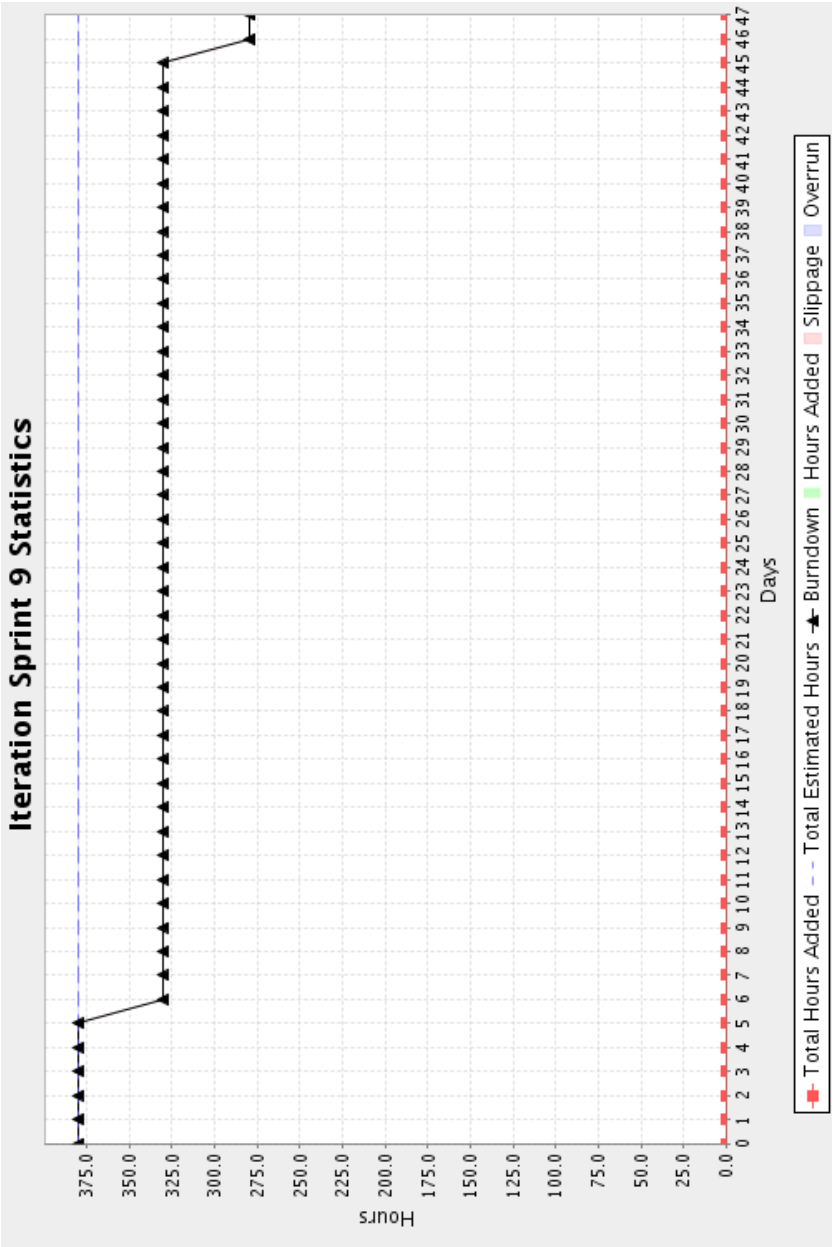
							fileheader can de scenefile staan. Deze header wordt al uitgelezen.
1	23	Versiechecks in sceneloading		20			
3	22	quiltoffsets safe deleten			0	Y	
1	21	Paden uit scenefile in een struct zetten			0	Y	
2	20	Skybox texture coordinates fixen			0	Y	
1	19	Pragma comments eruit halen			0	Y	
1	18	Logger inbouwen		40			
1	17	Incorrect logbericht bij ontbrekende resources			0	Y	
2	16	Getentities in quadtreegeeft een referentie naar een lokale variabele		20			
3	15	Patch buren updaten als die buur verwijderd wordt			0	Y	
2	14	Entity container's entity's positie moet relatief worden aan de container		40			
2	13	Const correctness	bestaande const correctnes checken	20	1	Y	Taak kan continue aan gewerkt worden.
3	12	Hardcoded variabelen in scene fixen	Quadtree dimensies en patch file en offset berekeningen gebruiken hardcoded variabelen.	160			
2	11	Methodes getregionmatrix en getworldmatrix renamen			0	Y	
1	10	Clearcolor en hwnd gebruiken			0	Y	
3	9	Streaming quilt naar patch omreken bug eruit halen			0	Y	
3	8	Cameracontroller fixen			???		camera controller gebruiken we niet meer. We gebruiken een overerving van camera.
2	7	Development project moet schoon gemaakt worden			0	Y	
4	6	Boundingbox fixen			0	Y	
4	5	Quake uit patch halen			0	Y	
4	4	Memory leaks eruit halen			???		is al gedaan? behalve een paar bytes in input die ondertussen vervangen is.
2	3	Warnings eruit halen			0	Y	
1	2	Global enums eruit halen		20			
1	1	Sky refactoren			0	Y	

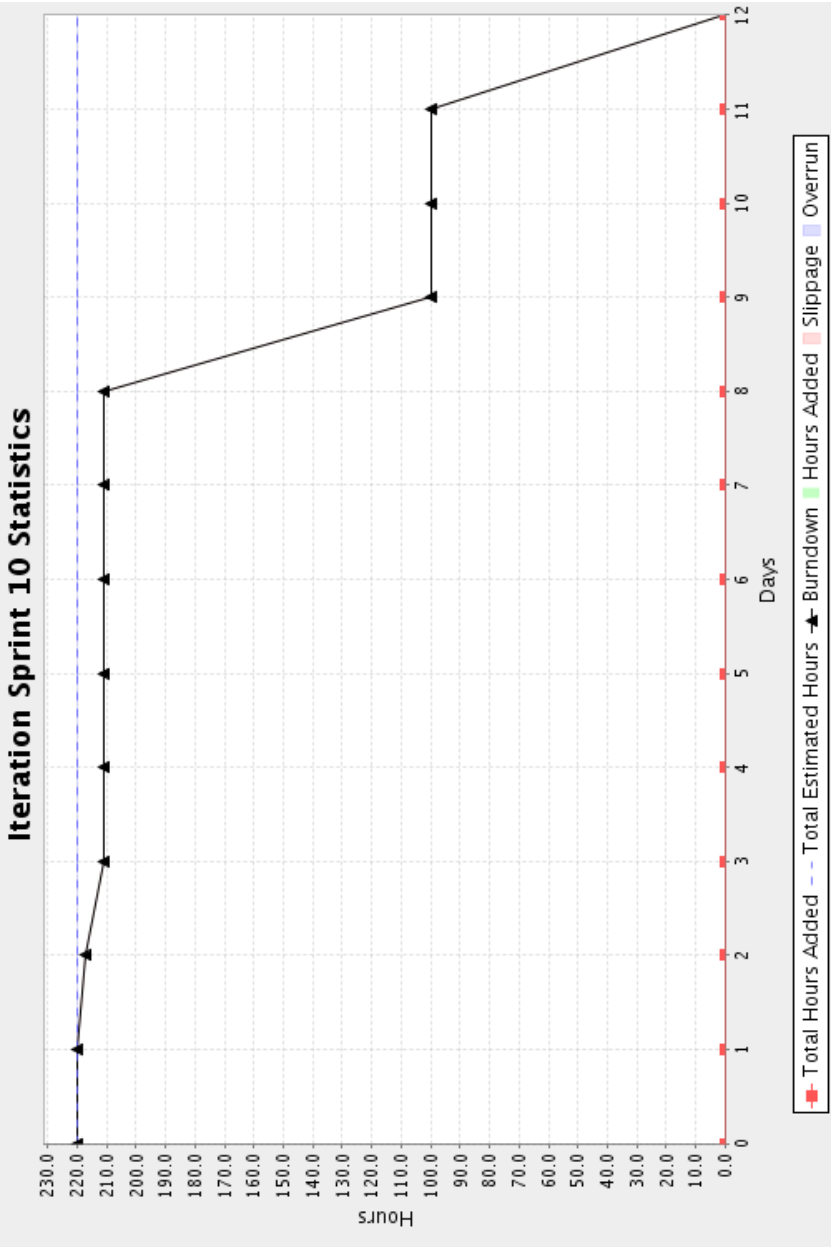
9. Burndown charts


Iteration Sprint 6 Statistics
Hours
210.0
200.0
190.0
180.0
170.0
160.0
150.0
140.0
130.0
120.0
110.0
100.0
90.0
80.0
70.0
60.0
50.0
40.0
30.0
20.0
10.0
0.0
0
1
2
3
4
5
6
7
8
9
10
Days
Total Hours Added
Total Estimated Hours
Burndown
Hours Added
Slippage
Overrun


Iteration Sprint 7 Statistics
Hours
375.0
350.0
325.0
300.0
275.0
250.0
225.0
200.0
175.0
150.0
125.0
100.0
75.0
50.0
25.0
0.0
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Days
Total Hours Added
Total Estimated Hours
Burndown
Hours Added
Slippage
Overrun


Iteration Sprint 8 Statistics
Hours
300.0
275.0
250.0
225.0
200.0
175.0
150.0
125.0
100.0
75.0
50.0
25.0
0.0
0
1
2
3
4
5
6
7
8
9
10
11
12
Days
Total Hours Added
Total Estimated Hours
Burndown
Hours Added
Slippage
Overrun


Iteration Sprint 9 Statistics
Hours
375.0
350.0
325.0
300.0
275.0
250.0
225.0
200.0
175.0
150.0
125.0
100.0
75.0
50.0
25.0
0.0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
Days
Total Hours Added
Total Estimated Hours
Burndown
Hours Added
Slippage
Overrun


Iteration Sprint 10 Statistics
Hours
230.0
220.0
210.0
200.0
190.0
180.0
170.0
160.0
150.0
140.0
130.0
120.0
110.0
100.0
90.0
80.0
70.0
60.0
50.0
40.0
30.0
20.0
10.0
0.0
0
1
2
3
4
5
6
7
8
9
10
11
12
Days
Total Hours Added
Total Estimated Hours
Burndown
Hours Added
Slippage
Overrun