

Dokumentation

Venus and Mars

Teammitglieder: Thomas Taschner, Michael Weinberger

Version 1.0

09.12.2015

Status: [RELEASE]

Git-Pfad: /doc/ Dokument: dokumentation.tex	
---	--

Inhaltsverzeichnis

1 Changelog	3
2 Übersicht	4
3 Projektbeschreibung	4
3.1 Anforderungen	4
3.2 Teammitglieder	5
4 Evaluierung der Frameworks	5
4.1 Pygame	5
4.2 Panda3D	5
4.3 Fazit	5
5 Stand vom 24. November 2015	6
6 Stand vom 30. November 2015	7
7 Stand vom Abgabetag 09.12.2015, Rückblick	8
8 Quellen	9

1 Changelog

Version	Datum	Status	Bearbeiter	Kommentar
0.1	14.10.2015	Erstellt	Thomas Taschner	Erstellt und Inhalte hinzugefügt
0.2	23.11.2015	Bearbeitet	Michael Weinberger	Dokumentenstruktur geändert, Evaluierung
0.3	24.11.2015	Bearbeitet	Michael Weinberger	Evaluierung, Zwischenstand
0.4	24.11.2015	Bearbeitet	Thomas Taschner	Fehlerkorrekturen
0.5	30.11.2015	Bearbeitet	Michael Weinberger	Aktueller Zwischenstand
1.0	09.12.2015	Bearbeitet	Michael Weinberger	Abschlusszustand dokumentiert, Rückblick

2 Übersicht

Erstelle eine einfache Animation unseres Sonnensystems!

3 Projektbeschreibung

3.1 Anforderungen

In einem Team (2 Personen) sind folgende Anforderungen zu erfüllen.

- Ein zentraler Stern
- Zumindest 2 Planeten, die sich um die eigene Achse und in elliptischen Bahnen um den Zentralstern drehen
- Ein Planet hat zumindest einen Mond, der sich zusätzlich um seinen Planeten bewegt
- Weitere Planeten, Asteroiden, Galaxien,...
- Zumindest ein Planet wird mit einer Textur belegt

Events:

- Mittels Maus kann die Kameraposition angepasst werden: Zumindest eine Überkopfsicht und parallel der Planetenbahnen
- Da es sich um eine Animation handelt, kann diese auch gestoppt werden.
- Mittels Tasten kann die Geschwindigkeit gedrosselt und beschleunigt werden.
- Mittels Mausklick kann eine Punktlichtquelle und die Textierung ein- und ausgeschaltet werden.
- Auch Monde und Planeten werfen Schatten.

Hinweise zu OpenGL und glut:

- Ein Objekt kann einfach mittels glutSolidSphere() erstellt werden.
- Die Planeten werden mittels Modelkommandos bewegt: glRotate(), glTranslate().
- Die Kameraposition wird mittels gluLookAt() gesetzt.
- Entfernte Objekte sollen kleiner und nahe Objekte größer.
- Wichtig ist dabei auch eine möglichst glaubhafte Darstellung.
gluPerspective(), glFrustum()
- Für das Einbetten einer Textur kann die Library Pillow verwendet werden!

3.2 Teammitglieder

Name	Rolle
Thomas Taschner, Michael Weinberger	Entwickler, Dokumentation

4 Evaluierung der Frameworks

Im Rahmen der Evaluierung haben wir uns auf zwei Python-3D-Frameworks konzentriert, Pygame und Panda3D.

4.1 Pygame

Pygame ist ein plattformübergreifendes Set an Python-Modulen zur Spieleprogrammierung. Mit Millionen Downloads hat das Tool auch einen annehmbaren Verbreitungsgrad sowie eine aktive Community. Die derzeitige Version ist ausgelegt auf Python 2.7, was nicht der neuesten Version entspricht, ein inoffizieller Release verschafft Abhilfe. Wir bekamen auch den Tipp, dass die 64 Bit-Version auf manchen Rechnern schlicht nicht läuft. Außerdem fehlen wichtige, benötigte Pakete, die über pip nachinstalliert werden müssen. Das Tool an sich ist ok, jedoch ist die Usability klar verbesserungswürdig.

4.2 Panda3D

Panda3D ist eine kostenlose Spiele-Engine, entwickelt von Disney, der Carnegie Mellon University und einigen freiwilligen Entwicklern. Von der technischen Seite her sind viele Features vorhanden. Mithilfe der Runtime (etwa 2 MB) ist das Tool auch weitgehend plattformunabhängig, der SDK wiegt etwa 200 MB. Die gute Dokumentation, die weite Verbreitung und die vielen Beispielcodes in Foren sprechen klar für Panda3D. Die vorgefertigten Beispiele (Solar System bereits existent) sind gut aufgebaut, kommentiert und einfach erweiterbar.

4.3 Fazit

PyGame bietet zwar einen guten Einstieg per Video-Tutorial (in 100 Teilen, sehr zeitaufwändig), benötigt aber vom Endbenutzer eine Vielzahl an Vorbereitungen, damit das Programm ausführbar gemacht wird.

Panda3D ist unserer Ansicht nach die beste Variante die Aufgabe umzusetzen, da bereits ein entsprechendes Beispiel vorhanden ist. Die Runtime kann der Abgabe beigefügt werden inklusive Verknüpfung, so entsteht kein Aufwand für den Benutzer, da lediglich ein Doppelklick das Programm startet.

Innerhalb von wenigen Minuten ist ein zufriedenstellendes Ergebnis sehbar, die Einarbeitung und Erweiterung des Codes geht leicht von der Hand. Codebeispiele aus dem Forum/der Dokumentation können mit nur minimalen Änderungen weiterverwendet werden.

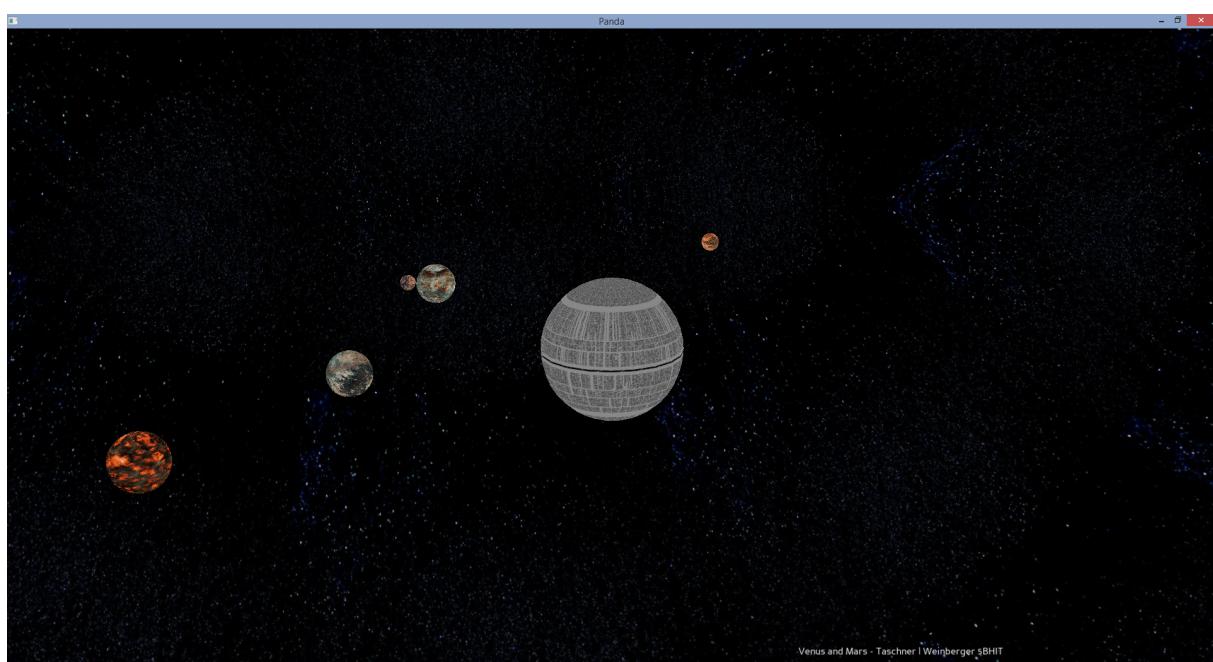
5 Stand vom 24. November 2015

Das UML ist aufgrund von Panda3D und der abgenommenen Planung derzeit obsolet. Das Example wurde angepasst, wir haben uns bereits einige Funktionen angeschaut.

Die Texturen wurden ausgetauscht mit höher aufgelösten, und ein Todesstern ist jetzt das Zentrum der Galaxis. Die Planetenbahnen wurden angepasst, auch der Hintergrund wurde verbessert. Als Testfall haben wir einen neuen Planeten hinzugefügt, inklusive Textur, Umlaufbahn, Größe und Rotationsverhalten. Im Hintergrund spielt der Imperial March (80% Lautstärke, linker Audiokanal), und eine Weltraum-Kampfszene aus Star Wars als Ambiance-Musik (100% Lautstärke, rechter Audiokanal)

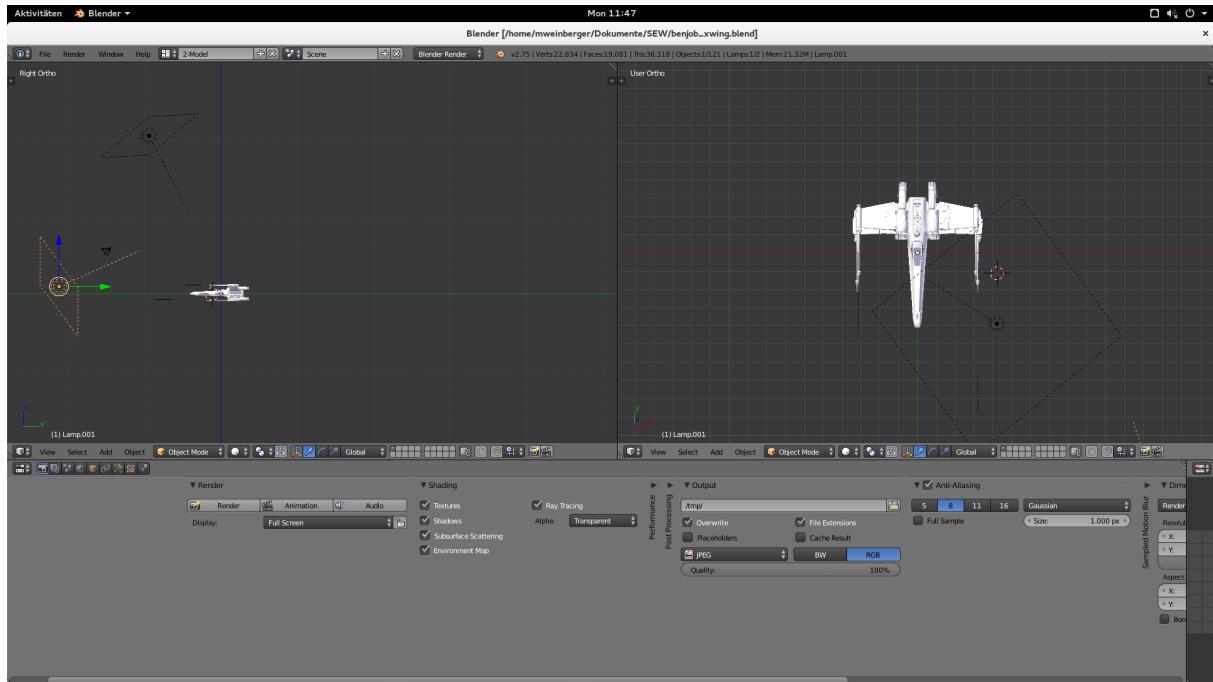
Nächste wichtige Schritte:

- Den Code verbessern, ihn mithilfe von Designpatterns effizienter machen und aufteilen
- Kantenglättung zum Laufen bringen
- Bessere GUI, Navigation per Menü möglich (Overlay, Buttons?)
- Mehr Planeten, Asteroiden, ...
- Schatten bzw. Beleuchtung generell einbinden
- Hilfenster zwecks besserer Usability
- Hintergrund verbessern bzw. Beschränkung beim Zoomen einführen
- Nice to have: Per Klick oder Tastendruck Fokus auf bestimmten Planeten
- Nice to have: Mehrere Galaxien (Strategy Pattern?)
- Nice to have: Bessere Maussteuerung
- Nice to have: Raumschiffe
- Nice to have: Geführte Tour durchs Solarsystem, feste Routen von außen nach innen, "Ego-Perspektive"



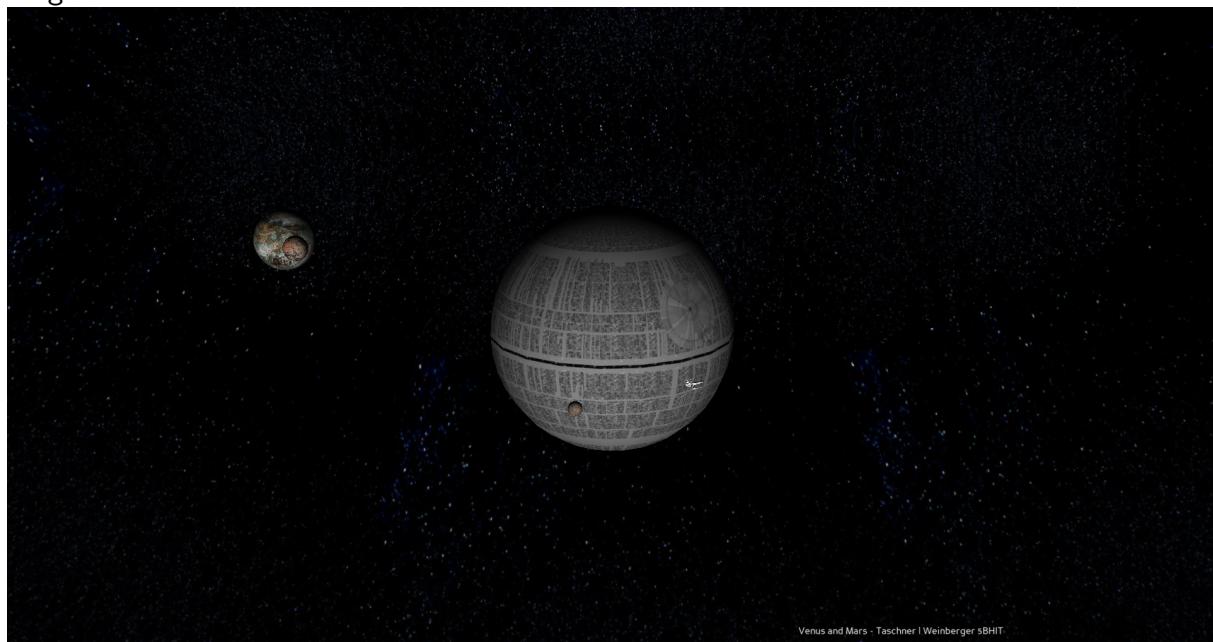
6 Stand vom 30. November 2015

Im Gegensatz zur vorigen Woche hat sich vieles verändert. Da der Todesstern keine Lichtquelle ist, haben wir uns für ein DirectionalLight von der Seite entschieden. Die Kantenglättung ist noch nicht implementiert, doch aufgrund des Schattenwurfs sind diese zumindest weniger sichtbar. Mithilfe von Blender [1], einer freien 3D-Grafiksoftware, wurde von einer Modelldatenbank ein X-Wing aus Star Wars importiert. Die Software enthält Funktionen, um dreidimensionale Körper zu modellieren, sie zu texturieren, zu animieren und zu rendern.



Der X-Wing (noch(?) ohne Textur) fliegt mit einer schnelleren Umlaufzeit als die Planeten um den Todesstern, entsprechend größenskaliert.

Die Anforderungen von voriger Woche bleiben natürlich bestehen, und werden in weiterer Folge umgesetzt.



7 Stand vom Abgabetag 09.12.2015, Rückblick

Ein Großteil der verbleibenden Arbeitszeit wurde für das Codedesign aufgewandt, mehr dazu in der *Technischen Dokumentation*. Neben der großen Veränderung auf Codeebene wurde zuletzt noch ein Riesenplanet etwas außerhalb des gleich zu Beginn sichtbaren Planetensystems eingefügt.

Der erste Punkt von 24. November, die Codeverbesserung, wurde erfolgreich abgeschlossen. Die Kantenglättung wurde obsolet, da die Treppchenbildung dank Schatten nicht mehr sichtbar war. Die Entwicklung einer umfangreicherem GUI wurde zwecks Ressourcenoptimierung ausgesetzt, dafür haben es mehr Planeten ins fertige Programm geschafft. Da der Todesstern keine Lichtquelle ist, wurde eine Beleuchtung von der Seite aus implementiert. Bei der Problematik des Hintergrunds (Ende des Bereichs sichtbar bei zu großem Zoom) wurde auch eine Lösung gefunden: Die Galaxie-Sphere um den Faktor 10 größer machen, nun ist beim Zoom diese harte Grenze um einiges schwieriger zu erreichen.

Damit das Programm ausgeführt werden kann, muss die Panda3D-Runtime installiert werden.

Über eine Verknüpfung mit der Runtime, unter Windows etwa *D:/Programme/Panda3D-1.8.1/python/python.exe -E VenusAndMars.py*, kann das Programm dann gestartet werden.

8 Quellen

[1]: <https://www.blender.org/>, Blender Foundation, zuletzt abgerufen am 30.11.2015