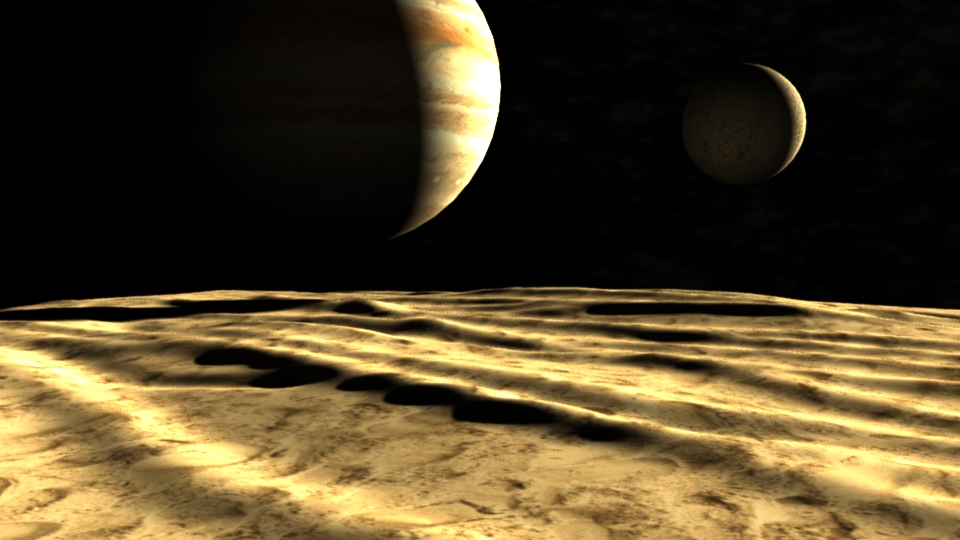
Visit our solar system

**2015**

Herczeg | Krickl

5BHITM

23.02.2015



Sonnensystem

Inhaltsverzeichnis

[Aufgabenstellung 2](#_Toc414258862)

[Zeitaufzeichnung 4](#_Toc414258863)

[Design 6](#_Toc414258864)

[Libraries und Versionen 6](#_Toc414258865)

[UML 6](#_Toc414258866)

[GUI 7](#_Toc414258867)

[Probleme 8](#_Toc414258868)

[Quellen 9](#_Toc414258869)

[Im Text verwendet 9](#_Toc414258870)

[Nachschlagen 9](#_Toc414258871)

# Aufgabenstellung

Wir wollen nun unser Wissen aus Medientechnik und SEW nützen um eine etwas kreativere Applikation zu erstellen.

Eine wichtige Library zur Erstellung von Games mit 3D-Grafik ist Pygame. Die 3D-Unterstützung wird mittels PyOpenGL erreicht.

Die Kombination ermöglicht eine einfache und schnelle Entwicklung.

Während pygame sich um Fensteraufbau, Kollisionen und Events kümmert, sind grafische Objekte mittel OpenGL möglich.

*Die Aufgabenstellung:*

Erstellen Sie eine einfache Animation unseres Sonnensystems:



In einem Team (2) 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
* Kreativität ist gefragt: Weitere Planeten, Asteroiden, Galaxien,...
* Zumindest ein Planet wird mit einer Textur belegt (Erde, Mars,... sind im Netz verfügbar)

Events:

* Mittels Maus kann die Kameraposition angepasst werden: Zumindest eine Überkopf-Sicht und parallel der Planentenbahnen
* 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.
* Schatten: Auch Monde und Planeten werfen Schatten.

Hinweise:

* Ein Objekt kann einfach mittels glutSolidSphere() erstellt werden.
* Die Planten werden mittels Modelkommandos bewegt: glRotate(), glTranslate()
* Die Kameraposition wird mittels gluLookAt() gesetzt
* Bedenken Sie bei der Perspektive, dass entfernte Objekte kleiner - nahe entsprechende größer darzustellen sind.  
  Wichtig ist dabei auch eine möglichst glaubhafte Darstellung. gluPerspective(), glFrustum()
* Für das Einbetten einer Textur wird die Library Pillow benötigt! Die Community unterstützt Sie bei der Verwendung.

 Tutorials:

* Pygame: https://www.youtube.com/watch?v=K5F-aGDIYaM

Viel Erfolg!

Lighting

|  |  |
| --- | --- |
| 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 | def setupLighting():  *""" Initializing Lighting and Light0*    :return:  *"""*  zeros = (0.15, 0.15, 0.15, 0.3)  ones = (1.0, 1.0, 1.0, 0.3)  half = (0.5, 0.5, 0.5, 0.5)    glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_AMBIENT, zeros)  glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_SPECULAR, half)  glMaterialf(GL\_FRONT\_AND\_BACK, GL\_SHININESS, 15)  glLightfv(GL\_LIGHT0, GL\_AMBIENT, zeros)  glLightfv(GL\_LIGHT0, GL\_DIFFUSE, ones)  glLightfv(GL\_LIGHT0, GL\_SPECULAR, half)  glEnable(GL\_LIGHT0)  glEnable(GL\_LIGHTING)  glColorMaterial(GL\_FRONT\_AND\_BACK, GL\_DIFFUSE)    glTexGeni(GL\_S, GL\_TEXTURE\_GEN\_MODE, GL\_SPHERE\_MAP)  glTexGeni(GL\_T, GL\_TEXTURE\_GEN\_MODE, GL\_SPHERE\_MAP)  glEnable(GL\_TEXTURE\_GEN\_S)  glEnable(GL\_TEXTURE\_GEN\_T)    glEnable(GL\_COLOR\_MATERIAL)  glEnable(GL\_NORMALIZE)  glShadeModel(GL\_SMOOTH) |

# Zeitaufzeichnung

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Anforderung** | **Priorität** | **Verantwortlicher** | **Zeit (G) [min]** | **Zeit (T)**  **[min]** | **Status (D,I,Te,Do,F, A)** |
| **Nicht funktional** |  |  |  |  |  |
| Recherche der Libraries und OpenGL | Mittel | K, H | 120 | 120 |  |
| Installation der Libraries | Hoch | K, H | 20 | 30 | F |
| Installation von OpenGL | Mittel | K, H | 10 | 5 | F |
| Texturen suchen und aufbereiten | Mittel | K | 60 | 20 | I |
| **Planung** |  |  |  |  |  |
| GUI planen | Mittel | K | 100 | 10 | F |
| GUI erstellen | Hoch | K | 100 |  |  |
| 3D-Raum erstellen. Dh.: Die GUI auf eine 3D-Ansicht vorbereiten | Hoch | K | 15 | 50 | I |
| Erstellen der 3D-Objekte | Niedrig | K | 5 | 15 | I |
| Klassendiagramme bzw. Software planen und erstellen | Hoch | H | 100 | 90 | I |
| **Implementierung** |  |  |  |  |  |
| Implementierung einer Planeten-Klasse | Hoch | H | 30 |  |  |
| Implementierung der Translation eines Planeten | Hoch | H | 10 |  |  |
| Implementierung der Rotation eines Planeten | Hoch | H | 10 |  |  |
| Implementierung einer Mond-Klasse | Hoch | H | 30 |  |  |
| Implementierung der Translation des Mondes | Hoch | H | 10 |  |  |
| Implementierung der Rotation des Mondes | Hoch | H | 10 |  |  |
| Punktlicht implementieren | Mittel | K, H | 60 |  |  |
| Schattenberechnung implementieren | Niedrig | K, H | 60 |  |  |
| Kamera implementieren | Hoch | H | 20 |  |  |
| **Steuerung und Animation** |  |  |  |  |  |
| Benutzersteuerung implementieren | Hoch | K | 20 |  |  |
| Kameraposition anpassbar machen | Hoch | K | 20 |  |  |
| Animationen implementieren | Hoch | K | 10 |  |  |
| Animation stoppbar/startbar machen | Hoch | K | 10 |  |  |
| Geschwindigkeit der Animation anpassbar machen | Hoch | H | 10 |  |  |
| Licht ein/ausschaltbar machen | Mittel | H | 5 |  |  |
| **Test und Abnahme** |  |  |  |  |  |
| Prototyp fix-fertig lauffähig machen | Hoch | K, H | 15 |  |  |
| Testcases planen | Mittel | K, H | 60 |  |  |
| Testcases schreiben | Hoch | K, H | 35 |  |  |
| UACs planen | Mittel | K, H | 60 |  |  |
| UACs durchführen | Mittel | K, H | 30 |  |  |
| Beta-Tests durchführen | Niedrig | K, H | 25 |  |  |
| DAU das Programm ausführen lassen | Niedrig | K, H | 15 |  |  |
| Abnahme | Hoch | H | 30 |  |  |
| **Summe Herczeg [min]** |  |  | 542.5 (9h) | 170 |  |
| **Summe Krickl [min]** |  |  | 542.5 (9h) | 170 |  |

G … Geschätzt

T … Tatsächlich

D … Design

I … Implementierung

Te … Test

Do … Dokumentation

F … Fertig

A … Abgenommen

H … Herczeg  
K … Krickl

# Design

## Libraries und Versionen

**Python 3.4.**

**Pygame 1.9.**

**Pillow 2.7.1**

Library zum Einbinden von Texturen

**PyOpenGL 3.x.**

Library um die GPU des PCs zu verwenden. Leichtes erstellen und einbinden von Standardformen möglich

**Freeglut 2.8.1 32bit**

Wird alternativ zum OpenGL Utility Toolkit verwendet.

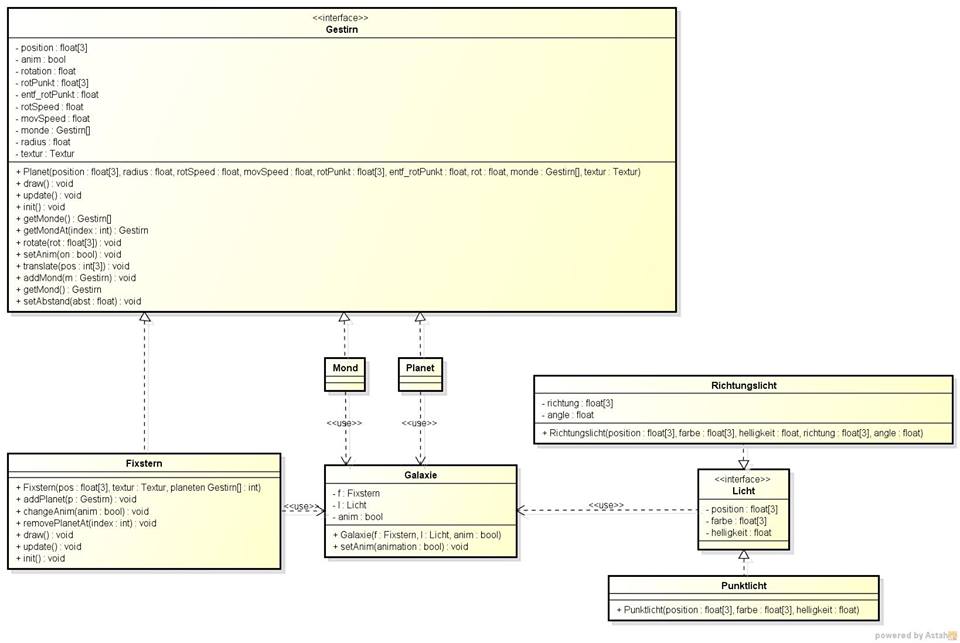
## UML

*Model*Man hat eine Galaxie die einen Fixstern besitzt, den man Planeten hinzufügen kann. Monde sind Planeten die man einem Planet hinzufügen kann.

*Control*Übernimmt die Steuerung bei Events (Mausklicks, Buttons,..).

Sie „verwaltet“ die Galaxie indem diverse Planeten mit Methoden hier in die View gesetzt und animiert werden.

*View*Planeten, Schatten, Texturen und Buttons werden alle angezeigt.



## GUI

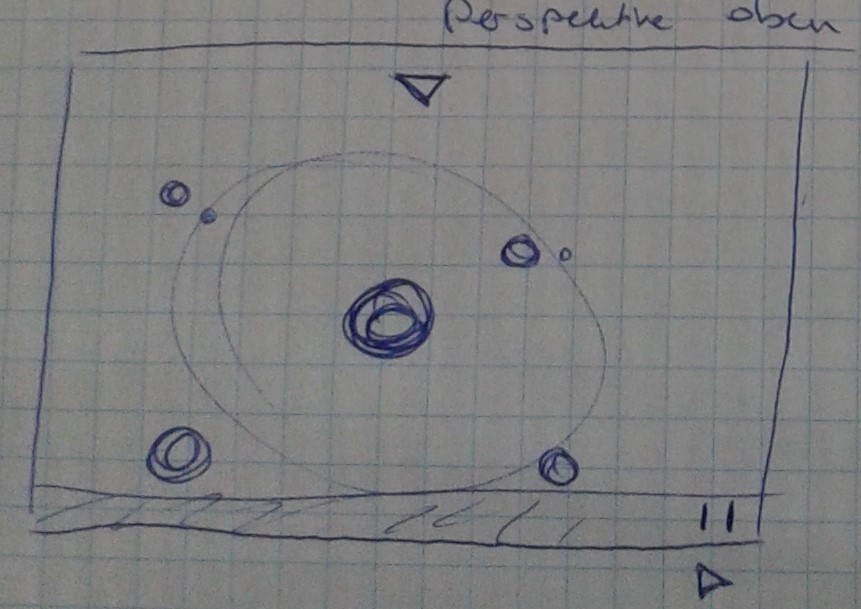
Startbildschirm mit Start Button und Hintergrundbild.



Perspektive seitlich von vorne. Pfeilbutton oben zum Perspektive ändern. Pausebutton unten rechts um die Animation zu pausieren (springt auf den Play Button um sobald die Animation pausiert ist und umgekehrt).



Ansicht von oben. Pfeil oben um die Perspektive zu ändern. Über Mausklicks lassen sich Textur und Belichtung an und ausschalten. Über die Pfeiltasten verändert sich die Geschwindigkeit der Animation.



# Probleme

Installation von Pillow 2.7.1 auf Python 3.3 war nicht möglich, deshalb Umstieg auf Python 3.4, wo es problemlos möglich war.

# Quellen

Kompletter Code siehe Github  
*https://github.com/akrickl-tgm/solar01.git*

## Im Text verwendet

## Nachschlagen

**PyOpenGL 3.x**  
http://pyopengl.sourceforge.net/  
24.02.2015

**Pillow 2.6.1**  
https://pypi.python.org/pypi/Pillow/2.6.1  
24.02.2015

**PyGame Download**http://www.pygame.org/download.shtml  
24.02.2015

**PyOpenGL Tutorial mcfeltch**  
http://bazaar.launchpad.net/~mcfletch/pyopengl-demo/trunk/view/head:/PyOpenGL-Demo/proesch/simple/simpleInteraction.py  
03.03.2015