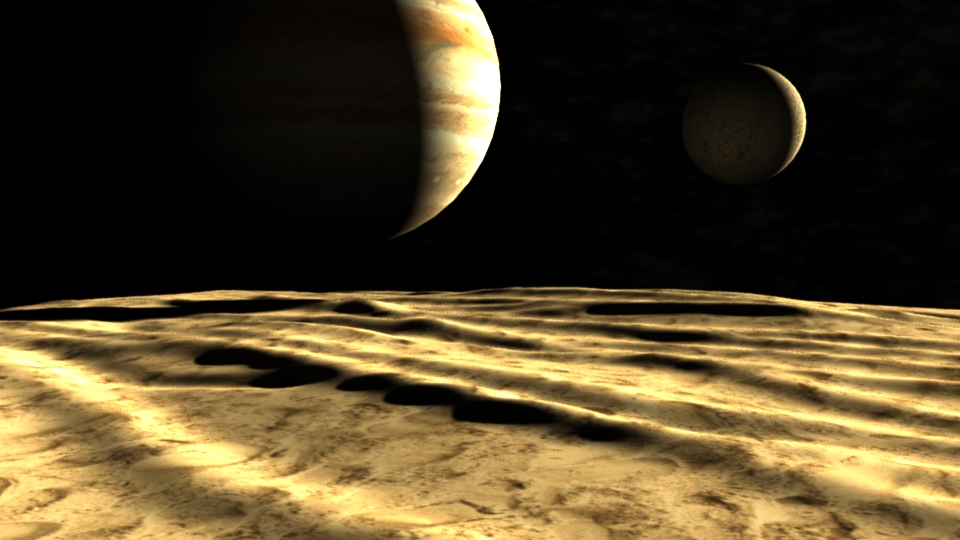
Visit our solar system

**2015**

Herczeg | Krickl

5BHITM

23.02.2015



Sonnensystem

Inhaltsverzeichnis

[Aufgabenstellung 2](#_Toc415398294)

[Zeitaufzeichnung 4](#_Toc415398295)

[Design 6](#_Toc415398296)

[Libraries und Versionen 6](#_Toc415398297)

[UML 8](#_Toc415398298)

[GUI 9](#_Toc415398299)

[Problemstellungen 11](#_Toc415398300)

[GUI 11](#_Toc415398301)

[Texturierung 11](#_Toc415398302)

[Lighting 12](#_Toc415398303)

[Eventhandling 13](#_Toc415398304)

[Animation 13](#_Toc415398305)

[Neue Gestirne hinzufügen 14](#_Toc415398306)

[Relationen der Gestirne 14](#_Toc415398307)

[Tests 15](#_Toc415398308)

[Ausführen und Steuerung 15](#_Toc415398309)

[Quellen 16](#_Toc415398310)

[Im Text verwendet 16](#_Toc415398311)

[Nachschlagen 16](#_Toc415398312)

# 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 | 130 | F |
| Installation der Libraries | Hoch | K, H | 30 | 40 | F |
| Installation von OpenGL | Mittel | K, H | 10 | 5 | F |
| Texturen suchen und aufbereiten | Mittel | K | 80 | 40 | I |
| **Planung** |  |  |  |  |  |
| GUI planen | Mittel | K | 120 | 130 | F |
| GUI erstellen | Hoch | K | 140 | 150 | F |
| 3D-Raum erstellen. Dh.: Die GUI auf eine 3D-Ansicht vorbereiten | Hoch | K | 40 | 60 | F |
| Erstellen der 3D-Objekte | Niedrig | K | 20 | 30 | F |
| Klassendiagramme bzw. Software planen und erstellen | Hoch | H | 120 | 100 | F |
| **Implementierung** |  |  |  |  |  |
| Implementierung eines Gestirns Interfaces | Hoch | H | 30 | 20 | F |
| Implementierung der Translation eines Planeten | Hoch | H | 20 | 40 | F |
| Implementierung der Rotation eines Planeten | Hoch | H | 10 | 20 | F |
| Implementierung einer Mond-Klasse | Hoch | H | 30 | 50 | F |
| Implementierung der Translation des Mondes | Hoch | H | 10 | 20 | F |
| Implementierung der Rotation des Monds | Hoch | H | 30 | 35 | F |
| Fixstern als Punktlicht implementieren | Mittel | K | 60 | 70 | F |
| Schattenberechnung implementieren | Niedrig | K, H | 60 | 80 | F |
| Kamera implementieren | Hoch | H | 20 | 30 | F |
| **Steuerung und Animation** |  |  |  |  |  |
| Benutzersteuerung implementieren | Hoch | K | 20 | 35 | F |
| Kameraposition anpassbar machen | Hoch | K | 20 | 30 | F |
| Animationen implementieren | Hoch | K, H | 10 | 40 | F |
| Animation stoppbar/startbar machen | Hoch | H | 10 | 30 | F |
| Geschwindigkeit der Animation anpassbar machen | Hoch | H | 10 | 20 | F |
| Licht ein/ausschaltbar machen | Mittel | K | 20 | 35 | F |
| **Test und Abnahme** |  |  |  |  |  |
| Prototyp fix-fertig lauffähig machen | Hoch | K, H | 15 | 15 | F |
| Testcases planen | Mittel | K, H | 60 | 30 | F |
| Testcases schreiben | Hoch | H | 35 | 20 | F |
| 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]** |  |  | 545 | 620 |  |
| **Summe Krickl [min]** |  |  | 545 | 620 |  |

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.**

Wird noch nicht verwendet.

**Pillow 2.7.1**

Library zum Einbinden von Texturen

@staticmethod

**def** LoadTexture**(**pic**):**

# Textur

ix **=** image**.**size**[**0**]**

iy **=** image**.**size**[**1**]**

image **=** image**.**tostring**(**"raw"**,** "RGBX"**,** 0**,** **-**1**)**

# Textur erstellen

textures **=** glGenTextures**(**1**)**

glBindTexture**(**GL\_TEXTURE\_2D**,** textures**)** # 2d texture (x and y size)

glTexParameteri**(**GL\_TEXTURE\_2D**,** GL\_TEXTURE\_MAG\_FILTER**,** GL\_LINEAR**)**

glTexParameteri**(**GL\_TEXTURE\_2D**,** GL\_TEXTURE\_MIN\_FILTER**,** GL\_LINEAR\_MIPMAP\_NEAREST**)**

gluBuild2DMipmaps**(**GL\_TEXTURE\_2D**,** 3**,** ix**,** iy**,** GL\_RGBA**,** GL\_UNSIGNED\_BYTE**,** image**)**

**return** textures

**PyOpenGL 3.x.**

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

**def** InitGL**(**self**,** Width**,** Height**):**

glEnable**(**GL\_TEXTURE\_2D**)**

glClearColor**(**0.0**,** 0.0**,** 0.0**,** 0.0**)** # Hintergrundfarbe

glClearDepth**(**1.0**)** # Loeschen des Depth Buffers

glDepthFunc**(**GL\_LESS**)** # The Type Of Depth Test To Do

glEnable**(**GL\_DEPTH\_TEST**)** # Enables Depth Testing

glShadeModel**(**GL\_SMOOTH**)** # Enables Smooth Color Shading

glMatrixMode**(**GL\_PROJECTION**)** # Reset The Projection Matrix

glLoadIdentity**()**

# camera

gluPerspective**(**45.0**,** float**(**Width**)** **/** float**(**Height**),** 0.1**,** 100.0**)**

glMatrixMode**(**GL\_MODELVIEW**)**

"""

Wenn die groesse vom Fenster geaendert wird

"""

**def** ReSizeGLScene**(**self**,** Width**,** Height**):**

glViewport**(**0**,** 0**,** Width**,** Height

glMatrixMode**(**GL\_PROJECTION**)**

glLoadIdentity**()**

# Perspektive

gluPerspective**(**50.0**,** float**(**Width**)** **/** float**(**Height**),** 0.1**,** 100.0**)**

glMatrixMode**(**GL\_MODELVIEW**)**

"""

szene zeichenn

"""

**def** DrawGLScene**(**self**):**

glClear**(**GL\_COLOR\_BUFFER\_BIT **|** GL\_DEPTH\_BUFFER\_BIT**)**

# Planet P1

self**.**rot\_pl2 **=** self**.**gestirn**.**rotation**(**self**.**rot\_pl2**,** 0**,** 0.04**,** 0**)** self**.**gestirn**.**DrawGLScene\_P**(**0.5**,** self**.**rot\_pl2**,** self**.**light**,**0.8**,**0**,-**10**)**

glutSwapBuffers**()** # zeichnen

**def** main**(**sc**):**

#solarsystem

glutInit**(**sys**.**argv**)**

glutInitDisplayMode**(**GLUT\_RGBA **|** GLUT\_DOUBLE **|** GLUT\_DEPTH**)**

glutInitWindowSize**(**1000**,** 600**)**

glutInitWindowPosition**(**50**,** 50**)**

glutCreateWindow**(**b'Solarsystem'**)**

glutDisplayFunc**(**sc**.**DrawGLScene**)**

glutIdleFunc**(**sc**.**DrawGLScene**)**

glutReshapeFunc**(**sc**.**ReSizeGLScene**)**

sc**.**InitGL**(**640**,** 480**)**

glutMainLoop**()**

s **=** universe**()**

main**(**s**)**

**Freeglut 2.8.1 32bit**

Wird alternativ zum OpenGL Utility Toolkit in Kombination mit PyOpenGL verwendet. Beispielcode von PyOpenGL lässt sich nur mit Freeglut verwenden.

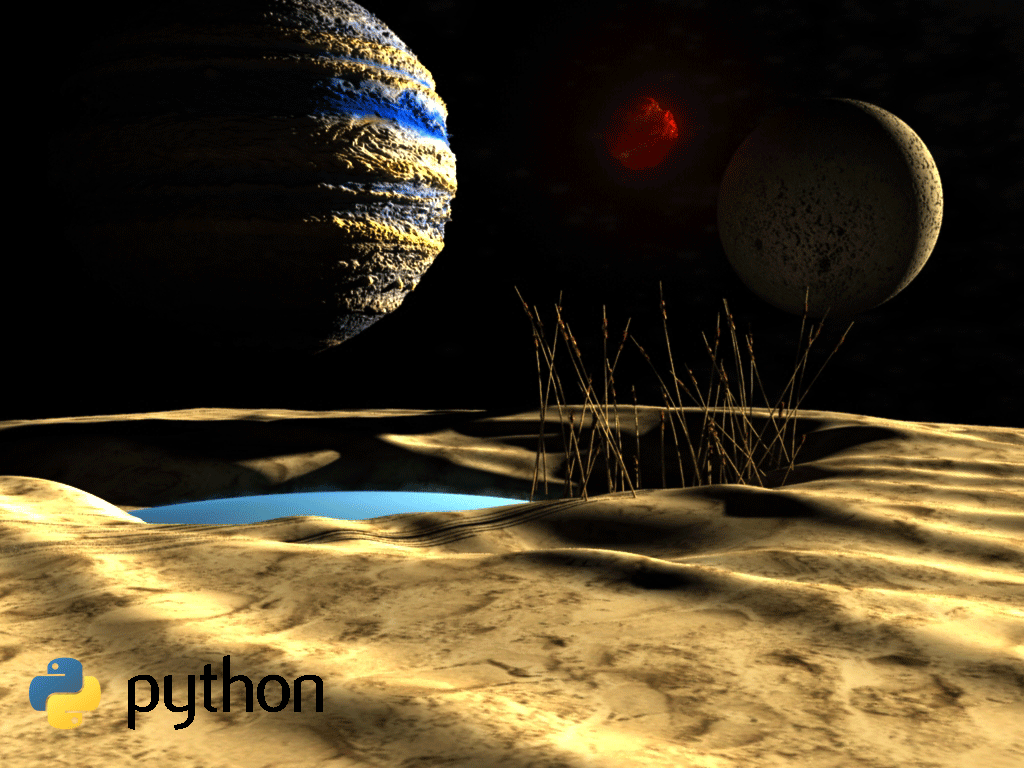
## UML

Eine Galaxie kann einen oder mehrere Fixsterne besitzen, um den Fixstern können sich Planeten drehen und um die Planeten können sich Monde bewegen. Da man ein Gestirn (Himmelkörper) verallgemeinern kann, da Monde, Planeten und Fixsterne sehr ähnliche Eigenschaften haben, gibt es ein Interface ‚Gestirn‘, das diese Abstaktion umsetzt. Alle Himmelskörper implementieren dieses Interface und fügen eigene spezifische Methoden hinzu. Da das Licht vom Fixstern ausgeht, verwendet dieser auch das ‚Licht‘-Interface, dass Punkt- und Richtungslicht implementiert hat.

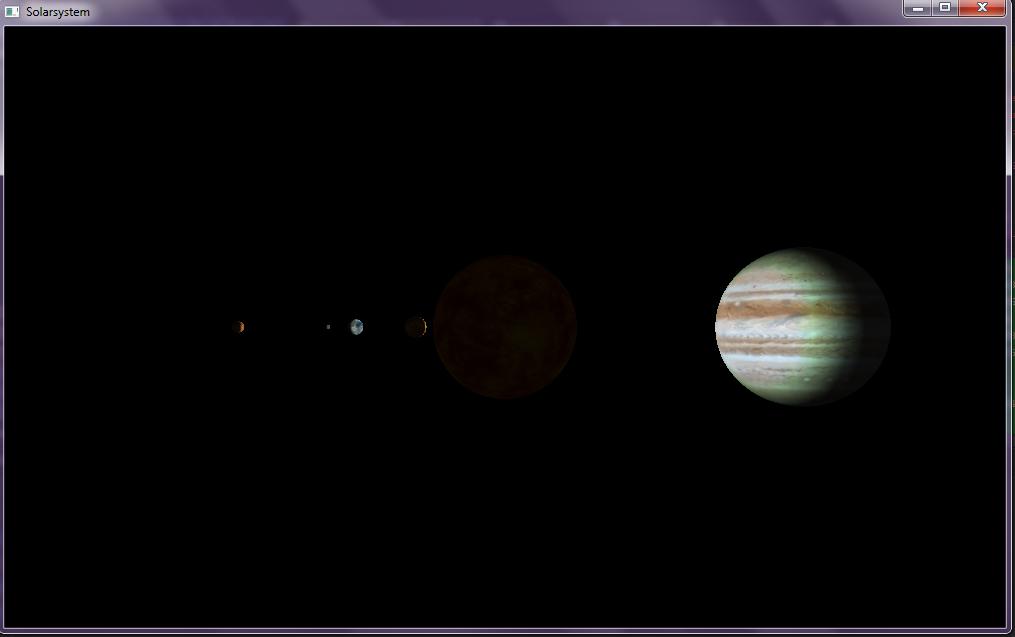
## GUI

Splashscreen [1]

Wird ein paar Sekunden lang angezeigt, bevor die Animation startet.



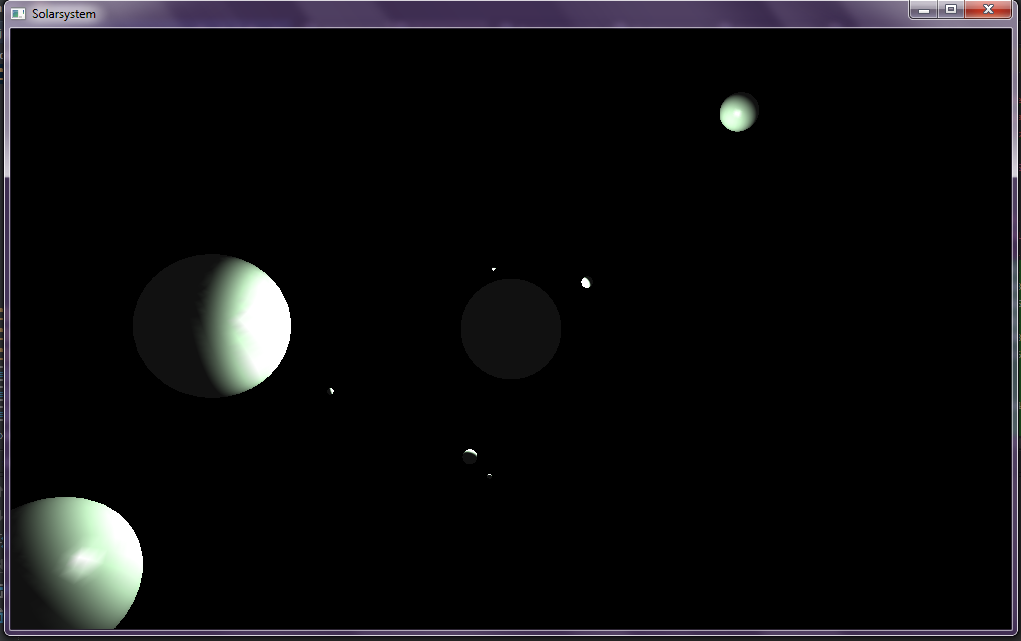
Perspektive von vorne. Taste ‚c‘ zum Perspektive ändern.



Ansicht von oben. Taste ‚c‘ um die Perspektive zu ändern.



Taste ‚t‘ um die Texturen auszuschalten.



Über ‚l‘ lässt sich die Belichtung an und ausschalten.



Taste ‚p‘ um die Animation zu pausieren und wieder anzuschalten. Über die Tasten w/s verändert sich die Geschwindigkeit der Animation und über die Tasten +/- kann man den Zoom verändern.

# Problemstellungen­­

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

## GUI

Die Animation des Universums wird über PyOpenGL und Freeglut angezeigt. Der Splashscreen aber wurde mithilfe von Tkinter realisiert. Mithilfe von Tkinter wird ein Bild (.gif) großflächig am Bildschirm angezeigt [1].

Nach diesem Verfahren öffnet Tkinter allerdings ein Standardfenster, das man durch folgenden Methodenaufruf schließen kann, da wir es sonst nicht gebraucht haben.

tkRoot**.**withdraw**()**

Der Aufruf dieser Methode geschieht nach dem Aufruf der enter und exit Methode der Splashscreen Klasse. Danach werden alle Komponenten von PyOpenGL gestartet die das Universum anzeigen.

## Texturierung

Alle Texturen wurden auf eine Breite von maximal 700px skaliert, da eine höhere Auflösung erstens die Animation sichtlich verlangsamt und zweitens man mit freiem Auge auch keinen Qualitätsunterschied erkennt. Für die Gestirne wurde das Bildformat ‚jpg‘ verwendet weil es PyOpenGL gut verarbeiten kann. Allerdings wurde für den Splashscreen das Format ‚gif‘ gewählt, da es schneller geladen und somit angezeigt werden kann, aber auch weil der Code des Splashscreens nicht in Verbindung mit PyOpenGL steht sondern im Tkinter und sich deshalb keine besonders gute und schnelle Bildverarbeitung angeboten hat.

Prinzipiell wurden die Texturen der Gestirne über eine eigene Texturen Klasse geladen. Die Textur wird über den Dateipfad angegeben.

**class** **Texturen():**

@staticmethod

**def** LoadTexture**(**pic**):**

image **=** open**(**pic**)**

# Textur

ix **=** image**.**size**[**0**]**

iy **=** image**.**size**[**1**]**

image **=** image**.**tostring**(**"raw"**,** "RGBX"**,** 0**,** **-**1**)**

# Textur erstellen

textures **=** glGenTextures**(**1**)** #textur ID

glBindTexture**(**GL\_TEXTURE\_2D**,** textures**)** # 2d texture (x and y size)

# Planet P1

glTexParameteri**(**GL\_TEXTURE\_2D**,** GL\_TEXTURE\_MAG\_FILTER**,** GL\_LINEAR**)**

glTexParameteri**(**GL\_TEXTURE\_2D**,** GL\_TEXTURE\_MIN\_FILTER**,** GL\_LINEAR\_MIPMAP\_NEAREST**)**

gluBuild2DMipmaps**(**GL\_TEXTURE\_2D**,** 3**,** ix**,** iy**,** GL\_RGBA**,** GL\_UNSIGNED\_BYTE**,** image**)**

**return** textures

Beim Erstellen eines Gestirns wird die Textur als Parameter angegeben und verwendet.

## Lighting

Das Hauptlicht (Punktlicht) geht vom Fixstern aus/von den Fixsternen. Aufgrund dessen werden die Schatten dynamisch berechnet und somit werden die Planeten umso näher sie an der Kamera sind, dunkler. Das Licht lässt sich mit dem Tastaturbefehl ‚l‘ ein- und ausschalten.

In PyOpenGL wurde das folgendermaßen gelöst:

**def** init**(**self**):**

glLightfv**(**GL\_LIGHT0**,** GL\_AMBIENT**,** **(**0.25**,** 0.25**,** 0.25**,** 1.0**))**# AmbientL

glLightfv**(**GL\_LIGHT0**,** GL\_DIFFUSE**,** **(**1.0**,** 1.0**,** 1.0**,** 1.0**))** # Diffuse L

glLightfv**(**GL\_LIGHT0**,** GL\_POSITION**,** **(**0.0**,** 0.0**,** 2.0**,** 1.0**))** # PositionL

glMaterialfv**(**GL\_FRONT\_AND\_BACK**,** GL\_AMBIENT**,** zeros**)**

glMaterialfv**(**GL\_FRONT\_AND\_BACK**,** GL\_SPECULAR**,** half**)**

glMaterialf**(**GL\_FRONT\_AND\_BACK**,** GL\_SHININESS**,** 100**)**

glEnable**(**GL\_LIGHT0**)**

glEnable**(**GL\_LIGHTING**)**

glColorMaterial**(**GL\_FRONT\_AND\_BACK**,** GL\_DIFFUSE**)**

glEnable**(**GL\_COLOR\_MATERIAL**)**

glEnable**(**GL\_NORMALIZE**)**

glShadeModel**(**GL\_SMOOTH**)**

Durch Objektattribut in der Klasse Light wurde es ermöglicht das Licht ein- und auszuschalten in der draw Methode:

**def** draw**(**self**):**

**if** self**.**enabled**:**

glShadeModel**(**GL\_SMOOTH**)**

glEnable**(**GL\_CULL\_FACE**)**

glEnable**(**GL\_DEPTH\_TEST**)**

glEnable**(**GL\_LIGHTING**)**

glDepthFunc**(**GL\_LESS**)**

lightZeroPosition **=** self**.**position

lightZeroColor **=** self**.**color

glLightfv**(**GL\_LIGHT0**,** GL\_POSITION**,** lightZeroPosition**)**

glLightfv**(**GL\_LIGHT0**,** GL\_DIFFUSE**,** lightZeroColor**)**

glLightf**(**GL\_LIGHT0**,** GL\_CONSTANT\_ATTENUATION**,** 0.01**)**

glLightf**(**GL\_LIGHT0**,** GL\_LINEAR\_ATTENUATION**,** 0.01**)**

glEnable**(**GL\_LIGHTING**)**

glEnable**(**GL\_LIGHT0**)**

**else:**

glDisable**(**GL\_LIGHTING**)**

glDisable**(**GL\_LIGHT0**)**

## Eventhandling

Das Eventhandling wurde von der Glut Library über die keyPressed Methode übernommen. [2]

**def** keyPressed**(**self**,** **\***args**):**

**if** args**[**0**]** **==** b'c'**:**

self**.**changePos**()**

**if** args**[**0**]** **==** b't'**:**

self**.**changeTextures**()**

**if** args**[**0**]** **==** b'l'**:**

self**.**disable\_light**()**

Hier wird der Tastaturbefehl entgegen genommen und ausgwertet. Den Tastatur-Input kann man mit b (für byte, gibt den byte Code des Nachfolgenden Strings zurück) ‚t‘ angegeben werden (hier wird auf den Buchstaben t geprüft). Wenn der Vergleich erfolgreich war wird die entsprechende Methode aufgerufen.

Die Steuerung über die Tastatur musste lediglich aktiviert werden über folgenden Methodenaufruf in der initGL Methode:

glutKeyboardFunc**(**self**.**keyPressed**)**

Hier gibt man an auf welche Methode bei einer gedrückten Taste referenziert werden soll. Da hier nur ein ‚call‘ angegeben wird, sind keine Klammern notwendig.

## Animation

Ein Gestirn jedes Universums hat nicht nur eine Rotation. Die Gestirne drehen sich um den Fixstern, die Monde drehen sich um den jeweiligen Planeten und alle Gestirne drehen sich auch um sich selbst.

Prinzipiell wird beim Erstellen eines Gestrins übergeben ob sich dieses Bewegen soll, den Rest berechnet die jeweilige Gestrin-Klasse automatisch mit den Methoden setAnimation, animateAllChildrenSlower und animateAllChildrenFaster. Beispiel dafür

**def** animateAllChildrenFaster**(**self**,** factor**,** factorMov**):**

self**.**rotSpeed **+=** factor

self**.**movSpeed **+=** factorMov

**for** i **in** range**(**0**,** len**(**self**.**monde**)):**

self**.**monde**[**i**].**animateFaster**(**factor**,** factorMov**)**

## Neue Gestirne hinzufügen

Der Code ist daraus ausgelegt schnell und einfach neue Gestirne hinzuzufügen (Planeten, Monde).

Um eine neue Sonne anzulegen müssen folgende Befehle ausgeführt werden.

# Parameter(Fixstern): (position, rotSpeed, textur, planeten, anim, licht, radius, divisions)

self**.**sonne **=** Fixstern**([**0**,** 0**,** **-**80**],** 0.2**,** self**.**sonnenTextur**,** **None,** **True,** self**.**light**,** 3**,** 64**)**

Wenn man eine Sonne hat, kann man dieser Planeten hinzufügen um sie um die Sonne rotieren zu lassen.

# Parameter(Planet): (position, anim, rotation, rotSpeed, rotPoint, movSpeed, radius, textur, divisions, monde)

self**.**merkur **=** Planet**([-**5**,** 0**,** **-**80**],** **True,** **[**90**,** 0**,** 0**],** 0.05**,** self**.**sonne**.**position**,** 0.0001**,** 1.25**,** self**.**merkurTextur**,** 32**,** **None)**

self**.**sonne**.**addPlanet**(**self**.**merkur**)**

Um einem Planeten einen Mond hinzuzufügen muss man folgenden Code ausführen (Planet muss wie im vorherigen Schritt beschrieben davor angelegt werden).

# Parameter(Mond): (anim, rotation, rotSpeed, parent, entf\_rotPoint, movSpeed, radius, textur, divisions)

self**.**mond **=** Mond**(True,** **[-**90**,** 0**,** **-**80**],** 0**,** self**.**erde**,** 5**,** **-**0.0005**,** 1**,** self**.**mondTextur**,** 24**)**

self**.**erde**.**addMond**(**self**.**mond**)**

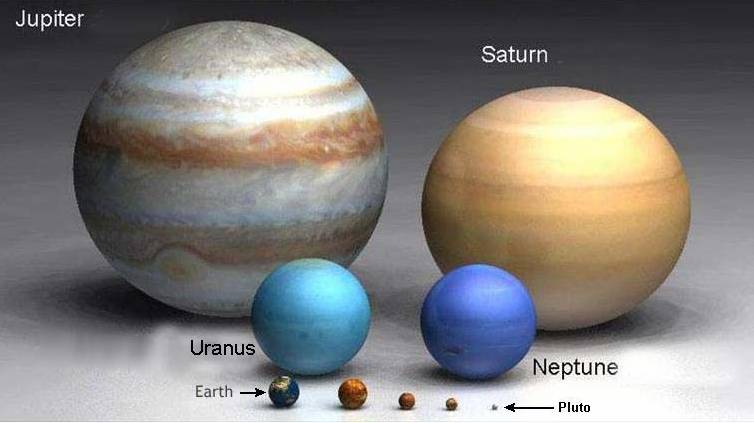
## Relationen der Gestirne

Da die Sonne in unserem Sonnensystem im Gegensatz zu den anderen Planeten unwahrscheinlich groß ist wurde diese nicht in ihrem echten Verhältnis dargestellt (Fixstern).

Die Relationen der Planeten zueinander wurden bestmöglich an die echten Maßstäbe angepasst. [3]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [*Sonne*](http://www.astronomie.de/astronomie-datenbank/astronomisches-lexikon/?tx_contagged%5Bsource%5D=default&tx_contagged%5Buid%5D=820&cHash=111ff06a1489f789c408b7f998c7845e) | ca. 1.390.000 Kilometer |  | [*Jupiter*](http://www.astronomie.de/astronomie-datenbank/astronomisches-lexikon/?tx_contagged%5Bsource%5D=default&tx_contagged%5Buid%5D=374&cHash=fb49b0b7023bd9a0550c161a0eb7a7ae) | ca. 143.000 Kilometer |
| [*Merkur*](http://www.astronomie.de/astronomie-datenbank/astronomisches-lexikon/?tx_contagged%5Bsource%5D=default&tx_contagged%5Buid%5D=376&cHash=ed2d6030d1b4c3026258354304e8d701) | ca. 4.900 Kilometer |  | [*Saturn*](http://www.astronomie.de/astronomie-datenbank/astronomisches-lexikon/?tx_contagged%5Bsource%5D=default&tx_contagged%5Buid%5D=378&cHash=15a811113b7b98499b08c512b3f39092) | ca. 120.500 Kilometer |
| [*Venus*](http://www.astronomie.de/astronomie-datenbank/astronomisches-lexikon/?tx_contagged%5Bsource%5D=default&tx_contagged%5Buid%5D=380&cHash=45b8dbe8d3e8f9f22c1f3c18897d70a1) | ca. 12.100 Kilometer |  | [*Uranus*](http://www.astronomie.de/astronomie-datenbank/astronomisches-lexikon/?tx_contagged%5Bsource%5D=default&tx_contagged%5Buid%5D=379&cHash=31ea794754611df47ce4825eebc91e65) | ca. 51.100 Kilometer |
| *Erde* | ca. 12.800 Kilometer |  | [*Neptun*](http://www.astronomie.de/astronomie-datenbank/astronomisches-lexikon/?tx_contagged%5Bsource%5D=default&tx_contagged%5Buid%5D=377&cHash=7fd14c6321e9cde91185f1d48e158955) | ca. 49.500 Kilometer |
| [*Mars*](http://www.astronomie.de/astronomie-datenbank/astronomisches-lexikon/?tx_contagged%5Bsource%5D=default&tx_contagged%5Buid%5D=375&cHash=07bd8bd0d90574d1568f38656fa55b87) | ca. 6.800 Kilometer |  |  |  |

Die Verhältnisse als Bild zur besseren Vorstellung:



Es werden allerdings nicht alle Planeten in unserem Sonnensystem zu sehen sein.

# Tests

Die wichtigsten Tests waren, ob das Programm richtig reagiert wenn man beim Anlegen eines bestimmten Gestirns falsche Werte übergibt.

# Ausführen und Steuerung

Um das Programm zu starten muss man im Package astrid die Datei start.py ausführen.

Die Steuerung funktioniert wie folgt.

|  |  |
| --- | --- |
| Taste(n) | Aktion |
| l | Licht an/aus |
| t | Texturen an/aus |
| p | Animation an/aus |
| c | Kameraposition ändern & zurücksetzen |
| w/s | Animationen schneller/langsamer |
| +/- | Mehr/Weniger Zoom |

# Quellen

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

## Im Text verwendet

[1] **Splashscreen with Python**  
http://code.activestate.com/recipes/576936-tkinter-splash-screen/  
02.03.2015

[2] **Tastatursteuerung**  
<https://www.opengl.org/documentation/specs/glut/spec3/node49.html>  
<http://stackoverflow.com/questions/8272463/pyopengl-glut-input>  
16.03.2015

[3] **Größenverhältnisse der Planeten**  
http://www.astronomie.de/astronomie-fuer-kinder/interessantes-fuer-lehrer-eltern/in-der-schule/groessenvergleich-der-planeten/  
26.03.2015

## 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

**Rotation**  
http://nehe.gamedev.net/tutorial/arcball\_rotation/19003/  
04.03.2015