VISIT OUR SOLAR SYSTEM …

Marlene Dorfinger, Muhammedmehdi Kanyildiz

INHALT

1. AUFGABENSTELLUNG

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

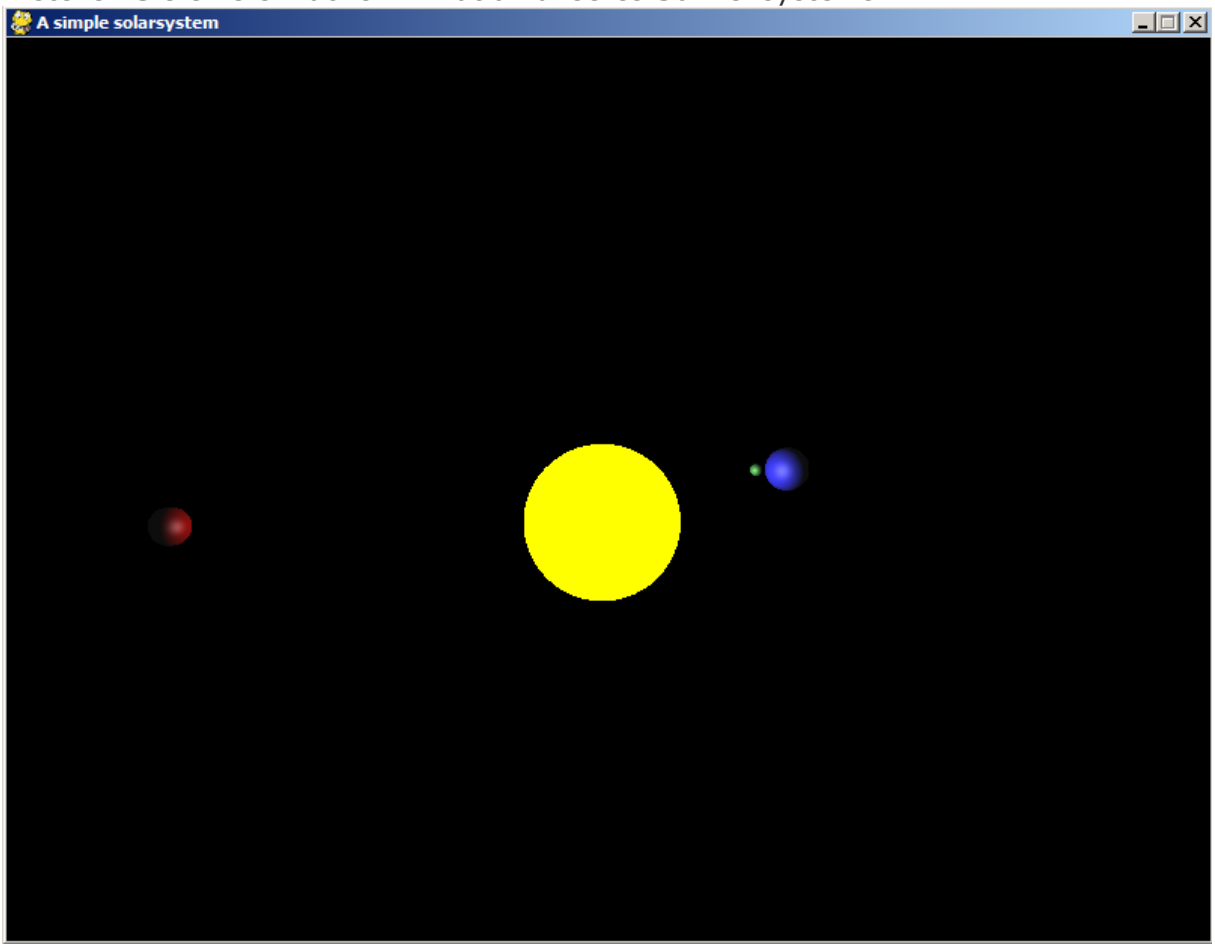
Eine wichtige Library zur Erstellung von Games mit 3D-Grafik ist Pygame. Die 3D-Unterstützung wird mittels PyOpenGL erreicht.

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!

2. Zeitaufzeichnung

Aufgabe	Priorität	Erwartete Zeit	Tatsächliche Zeit	Zuständigkeit	Status
OpenGL lernen	Hoch	2:00 h	5:00h	Dorfinger, Kanyildiz	F
Libraries suchen	Hoch	1:00 h	1:00 h	Dorfinger, Kanyildiz	F
Informieren über Sonnensystem	Mittel	1:00 h	0:30h	Dorfinger, Kanyildiz	F
Planeten implementieren (mind. 2)	Hoch	1:00 h	1:00h	Kanyildiz	F
Zentralstern implementieren	Hoch	1:00 h	0:30h	Kanyildiz	F
Mond(e) implementieren	Hoch	1:00 h	0:30h	Kanyildiz	F
Lichtquellen erstellen	Hoch	1:00 h	1:30h	Dorfinger, Kanyildiz	F
Texturen erstellen oder aus Internet suchen	Mittel	2:00 h	1:00 h	Dorfinger	F
Texturen laden	Hoch	0:30 h	0:30 h	Dorfinger	F
Texturen auf Objekte legen	Hoch	1:00 h	1:00 h	Dorfinger	F
Planeten drehen sich um eigene Achse und um Zentrealstern (unterschiedliche Geschw.)	Hoch	1:00 h	2:00h	Kanyildiz	F

Monde drehen sich um sich selbst, um Planeten und um den Zentralstern	Hoch	1:00 h	2:00 h	Kanyildiz	F
Implementieren der Tastensteuerung (Animation und Geschwindigkeit)	Hoch	1:00 h	0:40h	Kanyildiz	F
Einschalten/Ausschalten von Textur und Lichtquelle	Mittel	1:00 h	0:50h	Dorfinger, Kanyildiz	F
Perspektive ändern wen näher/weiter weg	Mittel	1:00 h	0:50h	Kanyildiz	F
3D-Splashscreen	Mittel	2:00 h	2:00 h	Dorfinger	F
Unit-Testing	Hoch	1:00h	2:40h	Kanyildiz	F
Gesamt		20:30h	21:30 h		

Legende Status:

I … Implementierung

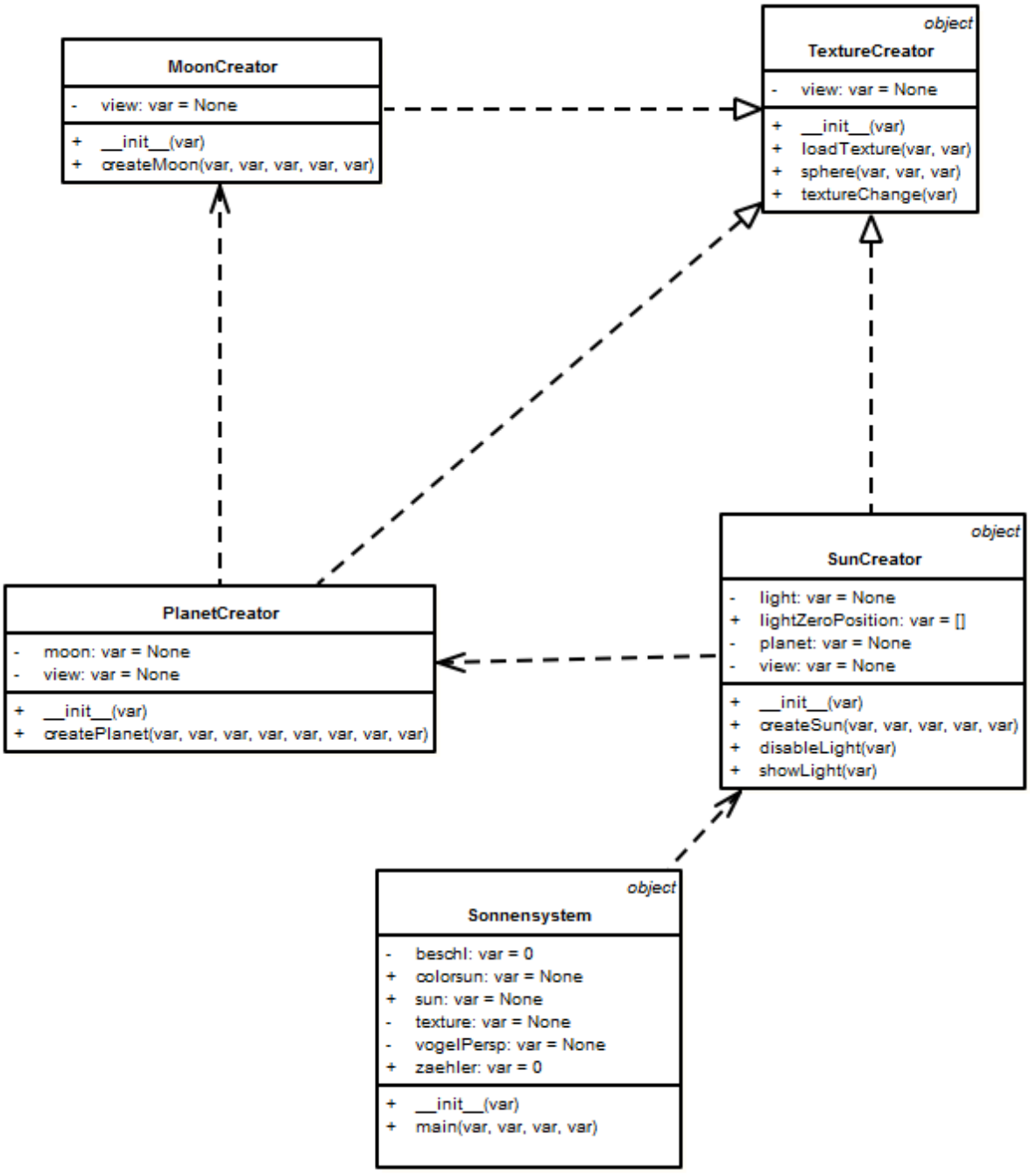
T … Test

K … Dokumentation

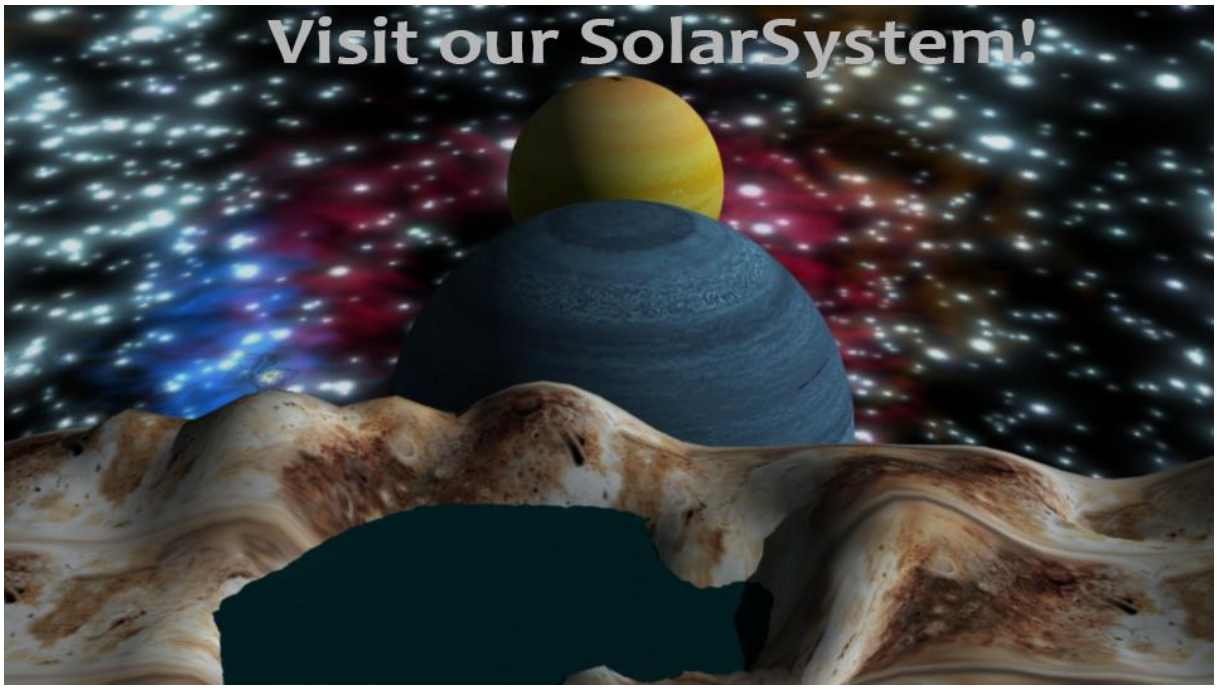
F … Fertig

3. Designüberlegung

3.1 UML



3.2 Splashscreen Prototyp



3.3 Verwendete Libraries

- PyOpenGL
  Verwendet für: Objekte, Lighting, Texturen
- PyGame
  Verwendet für: Benutzersteuerung
- Tkinter
  Verwendet für: Starbildschirm und messagebox
- Pillow
  Verwendet für: Texturen

4. Fortschritt

Geschafft:

- Die Sonne dreht sich nicht
- Es drehen sich mehrere Planeten um die Sonne
- Die Belichtung funktioniert(Die Sonne ist vollkommen beleuchtet während nur die eine Hälfte der Planeten beleuchtet ist)
- Lichter können aktiviert und deaktiviert werden
- Kamerasteuerung funktioniert zu 80%
- Die Texturierung
- Ein/Aus Schalten der texturen
- Unterschiedliche Geschwindigkeit beim Rotieren
- Monde
- Planet und Monde drehen sich um die eigene Achse
- Geschwindigkeit kann gesteuert werden

5. Funktionalität

5.1 Pygame installieren

Man geht auf die Seite http://www.lfd.uci.edu/~gohlke/pythonlibs/#pygame und lädt sich die Version Pygame für Python 3.4 runter. Diese installiert man mit:

pip install --no-index --find-links=LocalPathToWheelFile PackageName

5.2 PyOpenGL und Pillow installieren

Mit pip install pyopengl installiert man sich die Library PyOpenGL.
Mit pip install pillow installiert man sich die Library Pillow

5.3 Texturen mit Pillow

Mit der Library Pillow kann man Bilder öffnen und anzeigen lassen.
Man muss die Library einbinden mit `from PIL import Image`.
Mit `im = Image.open("test.jpg")` lädt man die Bilder.
Mit `im.show()` kann man sich die Bilder anzeigen lassen.
Mit `print(im.format,im.size,im.mode)` bekommt man Infos der Bilder.

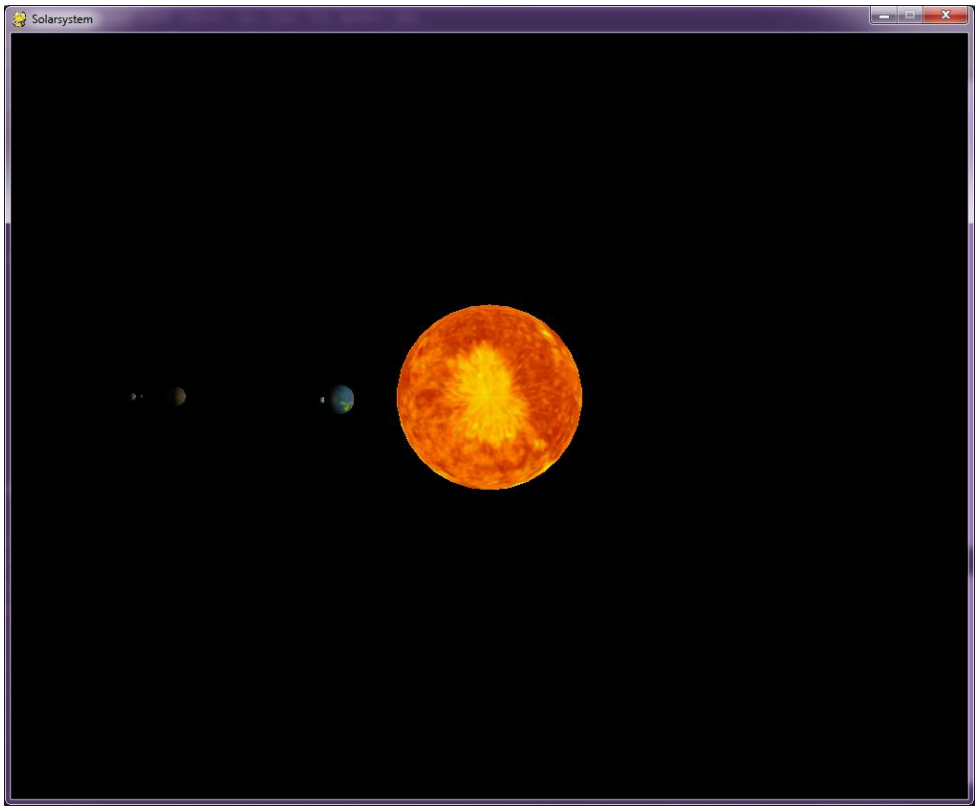
Um ein Bild zu öffnen und es anzeigen zu lassen verwendet man folgenden Code:

```
from PIL import Image

try:
    im = Image.open("test.jpg") #image laden
    print(im.format,im.size,im.mode) #infos ueber das bild
    im.show() #image zeigen

except:
    print ("Unable to load image")
```

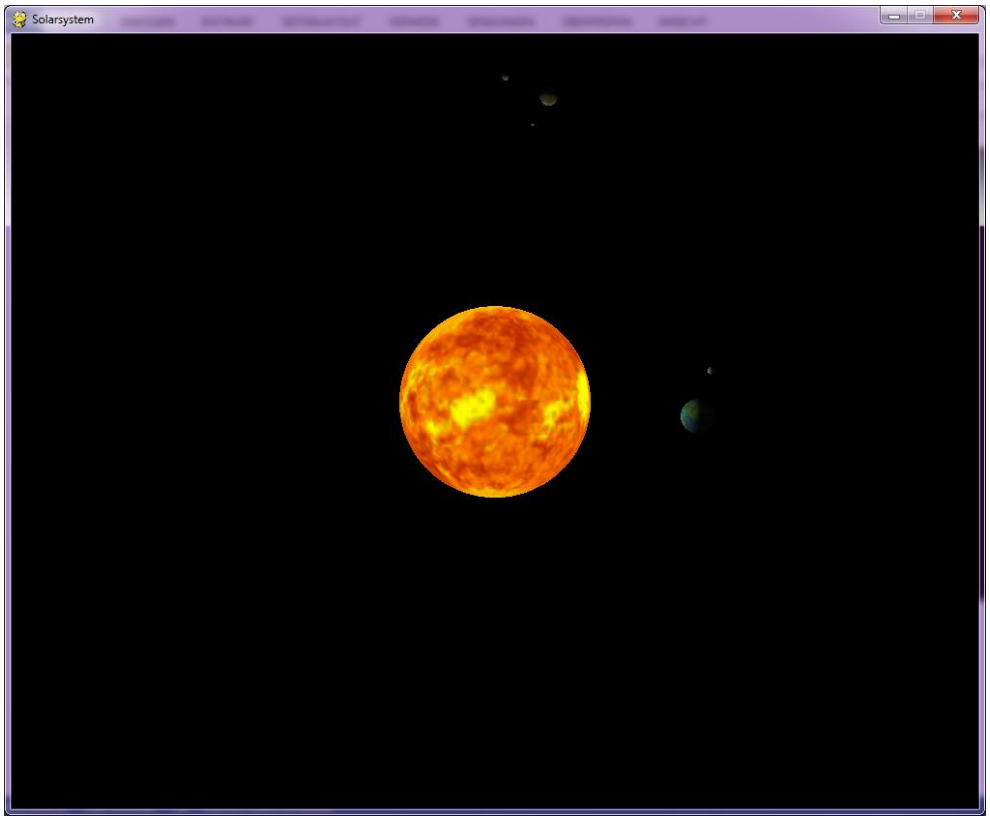
5.4 Sonnensysteminfos

Name	Größe (Durchmesser)	Geschwindigkeit um Sonne (km/s)
Sonne	1.390.000 km	-
Merkur	4.900 km	48
Venus	12.100 km	35
Erde	12.800 km	29.8
Mars	6.800 km	24
Jupiter	143.000 km	14
Saturn	120.500 km	9.6
Uranus	51.100 km	6.8
Neptun	49.500 km	5.4

5.5 Prototyp



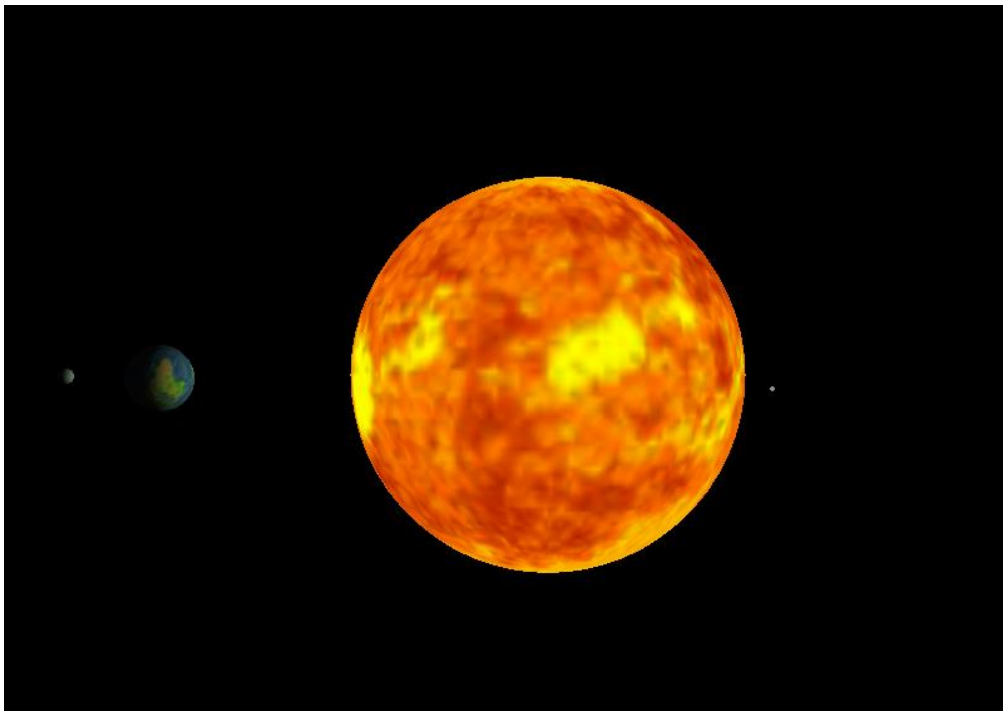
Ansicht von der Seite



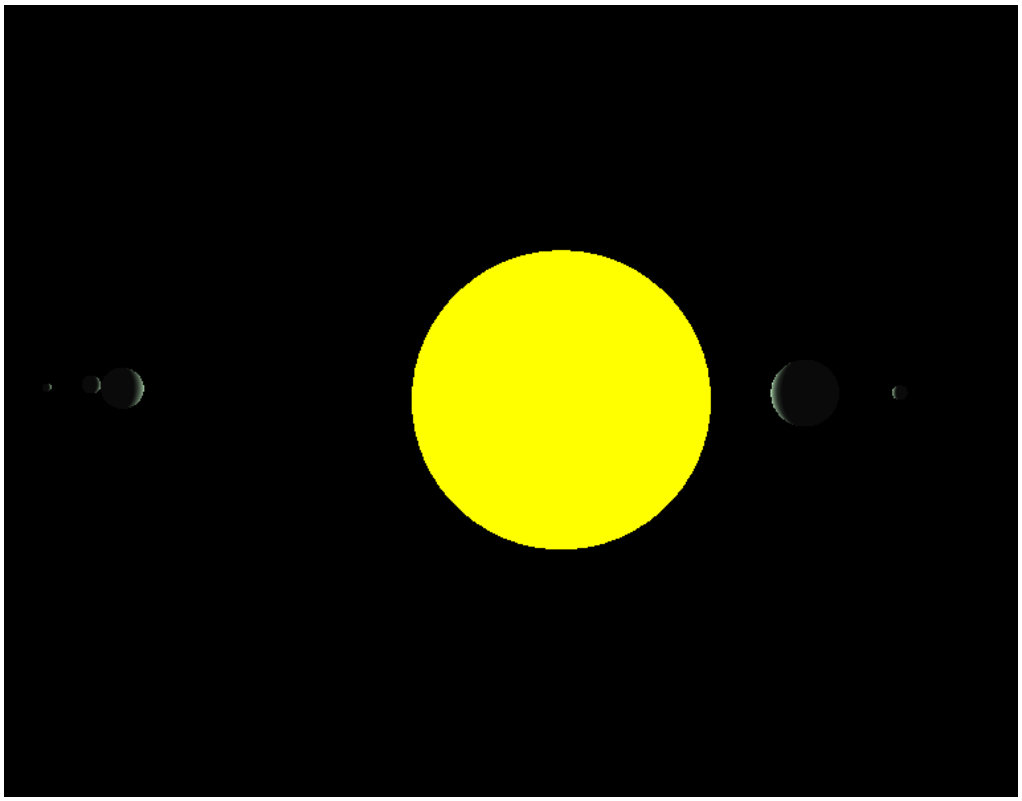
Ansicht von oben

Die Geschwindigkeit kann mit den Pfeiltasten Links und Rechts reguliert werden. Des Weiteren ist das ein(Taste o) und aus(Taste f) Schalten der Lichter möglich.

Die Kamera position kann mit der mittleren Maustaste geöndert werden (Rann- und Raus-Zoomen)



Außerdem werden die Texturen beim Betätigen der Taste T ein und ausgeschaltet.



Auch bei ausgeschalteter Textur kann man das Licht ein und ausschalten.

5.6 Splashscreen

Der Splashscreen wird mit tkinter implementiert.
Foldende Seite hat mir geholfen den Splashscreen zu mahen.

http://code.activestate.com/recipes/576936-tkinter-splash-screen/

5.7 Texturen auf Objekte legen

Zuerst muss man Texturen Einschalten, damit die Texturen angezeigt werden.

```
glEnable(GL_TEXTURE_2D)
```

dann muss man die Texturen auf ein Objekt legen mit

```
glBindTexture(GL_TEXTURE_2D, textures)
```

5.8 Texturen ein/aus schalten

Mit einer changeTexture-Methode wird die Textur ein bzw ausgeschalten. Diese Methode wird mit einem bool am laufen gehalten. Wenn der bool True ist, dann wird die Textur eingeschalten mit `glEnable(GL_TEXTURE_2D)` und die Texturen werden erneuert, außerdem wird mod auf False gesetzt. Wenn der bool falsch ist, wird die Textur ausgeschalet mit `glDisable(GL_TEXTURE_2D)`, außerdem wird mod auf True gesetzt.

Nun muss nur noch ein Standartwert gesetzt werden und danach das ganze aufgerufen werden.

```
if event.key == pygame.K_t:
                        self.__texture.textureChange()
```

5.9 MESSAGEBOX

Die Massagebox ist dazu da, dem User die Tastensteuerung zu erklären. Ich habe im Internet recherchiert wie man diese Messagebox mit tkinter umsetzen kann, kam zu der Lösung „tkintermessagebox“ zu verwenden. Allerdings gibt es das in den neueren Pythonversionen nicht. Durch ein bisschen ausprobieren bin ich dann zu der Lösung gekommen „`from tkinter import messagebox`“ zu importen und dann die Box mit `messagebox.showinfo(„Titel“, „Nachricht“)` aufzurufen. Mit \n kann man in die nächste Zeile springen.

```
messagebox.showinfo("Steuerung", "W ... Ansicht ändern \n T ... Textur [..] ")
```

5.11 LICHT POSITION GLEICH SONNEN POSITION

Um das Licht in dieselbe Position der Sonne zu bringen muss die Belichtungsmethode in demselben pop und push-matrix aufgerufen werden wie die Sonne. Damit die Belichtung nicht von der Sonne blockiert wird ist es dringend notwendig das Licht nach dem Aufruf der Sphere-Methode zu erstellen.

```
glPushMatrix()
self.disableLight()

self.__view.txtsonne = self.__view.loadTexture("./textures/sonne.jpg")
glRotate(90, 1, 0, 0)#um die texuren zu fixen müssen wir unser sphere drehen
self.__view.sphere(sizeSonne, self.__view.txtsonne)
glRotate(-90, 1, 0, 0)
self.showLight()
glPopMatrix()
```

5.12 DIE LICHTQUELLE ERSTELLEN

Um auf die Methoden zum Kreieren der Lichtquelle zugreifen zu können, muss einfach nur OpenGL importiert werden

```
from OpenGL.GL import *
```

Um ein Lichtquelle zu erstellen zu können müssen einige Operationen wie z.B ***Gl_Lighting, Gl_CULL_FACE*** und ***GL_DEPTH_TEST.***Dies geschieht durch die Methoden **gEnable, glDisable**, welche eine server-seitige Fähigkeit aktivieren bzw. deaktivieren.

GL_LIGHTING
Wenn aktiviert, werden die aktuellen Beleuchtungsparameter genutzt um die Vertex- oder Indexfarbe zu berechnen.

GL_DEPTH_TEST
Wenn aktiviert werden Tiefenvergleiche getätigt und der Tiefenpuffer aktualisiert.

GL_CULL_FACE
Wenn aktiviert, werden Polygone entsprechend ihrer Zeichnungsrichtung ausgeschlossen.

Die Lichtposition wird wie folgt bestimmt

```
lightZeroColor = [0.8,1.0,0.8,1.0]
```

Als nächstes werden die einzelnen Größen initialisiert, und dem Licht mit dem Namen GL_LIGHT0 zugeordnet.

```
glLightfv(GL_LIGHT0, GL_POSITION, self.lightZeroPosition)
glLightfv(GL_LIGHT0, GL_DIFFUSE, lightZeroColor)
glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, 0.1)
glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, 0.05)
```

Um OpenGL zu sagen, dass es Lichtberechnungen durchführen soll, muss die Funktion glEnable() mit dem Parameter GL_LIGHTNING verwendet werden:

```
glEnable(GL_LIGHTING)
```

Dann wird noch die Lichtquelle verfügbar gemacht werden:

```
glEnable(GL_LIGHT0)
```

5.13 Licht ein/aus Schalten

Damit die Lichtquelle ausgeschaltet werden kann ist die Methode glDisable() sehr wichtig. Um die Lichtbrechung zu deaktivieren muss man der Methode den Parameter GL_LIGHTING übergeben

```
glDisable(GL_LIGHTING)
```

Um die Lichtquelle zu deaktivieren wird FL_LIGHT0 als Parameter übergeben

```
glDisable(GL_LIGHT0)
```

5.14 Das Erstellen eines Objektes

Das Erstellen eines Objektes ist in OpenGl sehr einfach solange man die vorgefertigten Methoden benutzt.

Die Methode gluSphere Ermöglicht es uns eine Kugel zu zeichnen Dabei sind folgende Parameter zu beachten.

quad

Defniert das Quadratics Objekt

radius

Definiert den Radius des Kreises

slices

Die Unterteilungen um die Z-Achse

stacks

Die Unterteilungen in die Richtung der Z-Achse

6. Fehleranalysen

6.1 Texturen werden nicht angezeigt

Problem:
Texturen laden zwar aber sie werden auf den Planeten nicht angezeigt.

Lösung:
glEnable(GL_TEXTURE_2D)

6.2 MESSAGEBOX

Problem:
Die Messagebox öffnet sich mit der Nachricht, man muss sie aber schließen, damit sich das Programm öffnet. Messagebox verschwindet dann.

Provisorische Lösung:
Die Steuerung wird auch in der Konsole ausgegeben.

6.3 Ruckeln der Applikation

Problem:
Die Applikation hat geruckelt (Planeten, Drehung, Steuerung).

Lösung:
Die Texturen kleiner von der Größe her machen.

6.4 TEXTUR ID

Beim Testing wurde ein Fehler in der texture Klasse vorgefunden. Die Methode ***glBindTexture*** bekommt als Parameter einen String mitgegeben anstatt einer Textur ID. Doch dieser „Fehler“ taucht nur beim Testing auf und nicht beim Ausführen des Programms.

Lösung:

Die Zahl 1 als ID übergeben. So werden alle Testfälle erfolgreich abgeschlossen

6.5 UNTERSCHIEDLICHE GESCHWINDIGKEITEN DER PLANETEN

Problem:

Alle Planeten drehen sich mit derselben Geschwindigkeit

Lösung:

Um eine unterschiedliche Rotation der Planeten zu ermöglichen müssen wir folgendes beachten
Zuerst Drehen wir den jeweiligen Planeten um einen bestimmten Grad. Dieser Grad wird mit einem Zähler multipliziert. Dieser Zähler wird nach jedem Durchlauf der while-Schleife ein Wert hochgezählt. Anschließend wird der Planet um ein Wert verschoben. Dieser Vorgang wiederholt sich mithilfe der while-Schleife immer und immer wieder.

```
#rotation um die Sonne
glRotatef(speed*zaehler, 0, 1, 0)   #zuerst rotieren
glTranslatef(-abstand, 0, 0)          #und dann um die gewünschte distanz
verschieben
```

6.7 FEHLSCHLÄGE BEIM TESTING

Problem:

Beim Testing schlug das testen jedes Mal fehl, obwohl die richtige Ausgabe erwartet wurde

Lösung:

Um dieses Problem zu umgehen muss man vor dem Aufruf der zu testenden Methode ein ***lambda:*** hinzugefügt werden

```
def testPlanetAbstandString(self):
        self.assertRaises(TypeError, lambda: p1.main(arg1,arg2,arg3)
```

QUELLEN

1. Playlist zu Pygame (Python Game Development) von thenewboston, https://www.youtube.com/watch?v=K5F-aGDIYaM&list=PL6gx4Cwl9DGAjkwJocj7vlc_mFU-4wXJq gesehen 23.02.2015

2. Unofficial Window Binaries für Python Extension Packages, http://www.lfd.uci.edu/~gohlke/pythonlibs/#pygame gesehen 23.02.2015

3. How do I install Python libraries?, http://stackoverflow.com/questions/21222114/how-do-i-install-python-libraries gesehen: 23.02.2015

4. Reference, http://pillow.readthedocs.org/reference/index.html gesehen 01.03.2015

5. Umlaufgeschwindigkeiten von Planeten, http://www.astrologie.de/forum/astrologie-allgemein-f1/umlaufgeschwindigkeit-der-planeten-t1933.html gesehen 02.03.2015

6. Größenvergleich der Planeten unseres Sonnensystems, http://www.astronomie.de/astronomie-fuer-kinder/interessantes-fuer-lehrer-eltern/in-der-schule/groessenvergleich-der-planeten/ gesehen 02.03.2015

7. Tkinter Splash Screen (Python Recipe), http://code.activestate.com/recipes/576936-tkinter-splash-screen/ gesehen 24.03.2015

8. OpenGl, https://www.opengl.org/sdk/docs/man2/xhtml/gluSphere.xml , gesehen 29.03.2015

9. OpenGL, https://www.opengl.org/sdk/docs/man2/xhtml/gluLookAt.xml, gesehen 29.03.2015

10. Unofficial Window Binaries für Python Extension Packages, http://www.lfd.uci.edu/~gohlke/pythonlibs/#pyopengl, gesehen 29.03.2015

11. OpenGL with PyOpenGL tutorial Python and PyGame p.1 - Making a rotating Cube Example, https://www.youtube.com/watch?v=R4n4NyDG2hI, gesehen 29.03.2015