SEW - Projekt

PYTHON SOLAR SYSTEM

Ableitinger, Reiländer, Kölbl

TGM - IT

1 Inhaltsverzeichnis

1 Aufgabenstellung

Wir wollen unser Wissen aus SEW nutzen, um eine kreative Applikation zu erstellen. Die Aufgabenstellung:
Erstelle 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.

Wählt ein geeignetes 3D-Framework für Python (Liste unter https://wiki.python.org/moin/PythonGameLibraries) und implementiert die Applikation unter Verwendung dieses Frameworks.

Abgabe: Die Aufgabe wird uns die nächsten Wochen begleiten und ist wie ein (kleines) Softwareprojekt zu realisieren, weshalb auch eine entsprechende Projektdokumentation notwendig ist. Folgende Inhalte sind in jedem Fall verpflichtend:

- Projektbeschreibung (Anforderungen, Teammitglieder, Rollen, Tools, ...)
- GUI-Skizzen und Bedienkonzept (Schnittstellenentwürfe, Tastaturbelegung, Maussteuerung, ...)
- Evaluierung der Frameworks (zumindest 2) inkl. Beispielcode und Ergebnis (begründete Entscheidung)
- Technische Dokumentation: Architektur der entwickelten Software (Klassen, Design Patterns)
    - Achtung: Bitte überlegt euch eine saubere Architektur!
    - Den gesamten Source Code in 1 Klasse zu packen ist nicht ausreichend!
- Kurze Bedienungsanleitung
- Sauberes Dokument (Titelblatt, Kopf- und Fußzeile, ...)

Hinweise zu OpenGL und glut:

- 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 kann die Library Pillow verwendet werden! Die Community unterstützt Sie bei der Verwendung.

Viel Spaß und viel Erfolg!

2 Frameworkevaluation

Wir haben uns für unser Projekt 2 Framworks genauer angeschaut, anfangs wollten wir uns auch noch PyGame anschauen, doch nachdem wir Panda 3D gefunden haben, haben wir uns entschieden Panda 3D zu verwenden, weshalb es nicht mehrnotwendig war sich genauer mit PyGame zu beschäftigen.

2.1 Pyglet

2.1.1 Allgemein (von Wikipedia)

Pyglet is a library for the Python programming language that provides an object-oriented application programming interface allowing the creation of games and other multimedia applications. Pyglet runs on Microsoft Windows, Mac OS X, and Linux; it is released under BSD Licence.

Offizielle Website: www.pyglet.org

2.1.2 Installation

Die Installation mittels *python setup.py install* hat nicht auf Anhieb funktioniert, deswegen habe ich die benötigten Dateien einfach in das *site-packages* directory kopiert.
Pyglet importieren mit *import pyglet* und das Beispiel reinkopiert. Das Beispiel hat auf Anhieb funktioniert.

Hinweise zur Installation findet man unter https://pyglet.readthedocs.org/en/pyglet-1.2-maintenance/programming_guide/installation.html

2.1.3 Beispiel: Darstellung einer Kugel

Um erstmal 3d Objekte darstellen zu können, muss man pyglet entsprechend konfigurieren. Im folgenden ist ein Beispiel von der benötigten Konfiguration

```
# 3r Projektion einstellen
glEnable(GL_DEPTH_TEST)
# Kamera auf Fenstergröße einstellen
glViewport(0, 0, window.width, window.height)
glPolygonMode(GL_FRONT_AND_BACK, GL_LINE)
glMatrixMode(GL_PROJECTION)
glLoadIdentity()

# Lege Perspektive fest
gluPerspective(65, window.width/window.height, 0.1, 100.0)

# Lade ModelView Matrix
glMatrixMode(GL_MODELVIEW)
glLoadIdentity()

# Verschiebt die Modelle nach vorne, sodass man sie sieht
glTranslatef(0, 0, -5.0)
```

Ist die Konfiguration abgeschlossen, geht das Zeichnen sehr einfach.

```
q = gluNewQuadric()

gluSphere(q, 1, 20, 12)
```

Information über *gluSphere()* findet man in der OpenGL Dokumentation unter https://www.opengl.org/sdk/docs/man2/xhtml/gluSphere.xml

2.1.4 Fazit

Pyglet funktioniert sehr gut, jedoch ist es schwer zu verstehen, da es wie gesagt einiges an Konfiguration bedarf, bevor man überhaupt zeichnen kann.

Positives

Wenn man man herausgefunden hat, wie man mit Pyglet umgehen muss und wie alles genau funktioniert, funktioniert alles reibungslos und ohne jegliche Probleme

Negatives

Anfangs hat man Probleme herauszufinden, wo man nachschauen muss denn ein funktionierendes Example für eine einfache Kugel, habe ich auf Anhieb nicht gefunden.

2.2 Panda 3D

2.2.1 Allgemein (von Wikipedia)

Panda3D is a game engine that includes graphics, audio, I/O, collision detection, and other abilities relevant to the creation of 3D games.

Panda3D is open source and is, as of May 28, 2008, free software under the revised BSD license. Releases prior to that date are not considered free software due to certain errors in the design of the old Panda3D license. Despite this, those older releases of Panda3D can also be used for both free and commercial game development at no financial cost.

Offizielle Website: www.panda3d.org

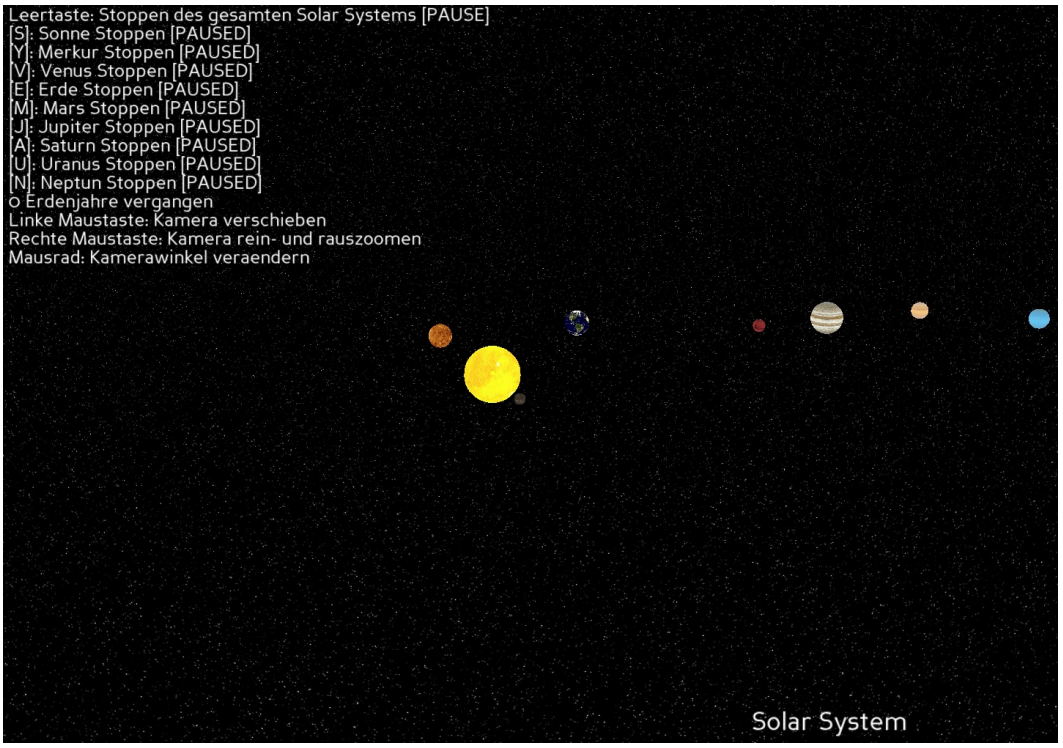
2.2.2 Installation

Die Installation von Panda 3D funktioniert über einen Setup Wizard, ein sehr gutes Tutorial welchem wir einfach gefolgt sind gibt es hier:
https://www.panda3d.org/manual/index.php/Installing_Panda3D_in_Windows

Nach der Installation können die Python files auch einfach in ein PyCharm Projekt importiert werden.

2.2.3 Beispiel

Panda 3D hat viele sehr gute Beispiele, was auch der Grund war, warum wir uns für dieses Framework entschieden haben. Es gibt auch bereits ein Solar System Beipsiel, welches unserer Aufgabenstellung sehr Ähnlich ist, sowie einige zur Steuerung der Kamera.



2.2.4 Fazit

Durch die vielen guten Beispiele, die einem den Einstieg ins Framework sehr einfache machen, haben wir uns dafür entschieden Panda 3D für unser Projekt zu verwenden.

Positives

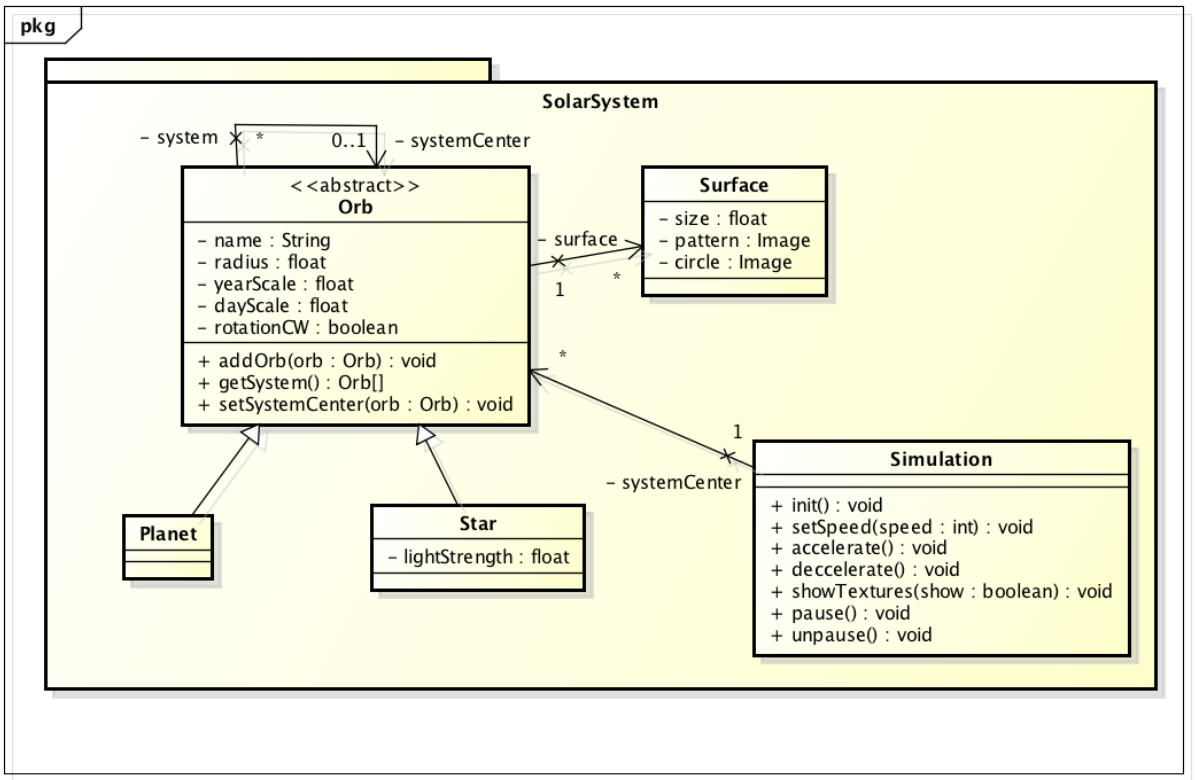
Das Framework ist vergleichsweise einfach zu verwenden und durch die zahlreichen Beispiele wird der Einstieg sehr einfach.

Negatives

Im Vergleich zu Pyglet sind viele Funktionalitäten nicht verfügbar, dies sind aber meistens Funktionen, welche für unser Projekt nicht notwendig sind.

3 Design

3.1 UML



Die abstrakte Klasse Orb stellt bei uns einen Himmelskörper (Stern, Planet, Mond) dar. Diese hat 2 konkrete Implementierungen: Star und Planet, wobei Star einen Stern (Sonne) darstellt, eine Lichtquelle ist und Planet einen Planeten oder Mond darstellt, welcher keine Lichtquelle ist.

Simulation stellt dabei die zentrale Klasse dar, wo alle Darstellungsrelevanten Werte angepasst werden können.

3.2 Verwendete Designpatterns

3.2.1 Strategy Pattern

Für die Oberfläche (Surface) verwenden wir ein Strategy Pattern, damit wir während der Laufzeit die Oberflächentextur eines Orbs ändern oder entfernen können.

3.2.2 Decorator/Composite Pattern

Für die Darstellung eines Sonnen/Planeten Systems verwenden wir ein kombiniertes Decorator/Composite Pattern, jeder Orb kann einem anderen Orb zugewiesen werden, wobei bei voneinder wissen, aber nach außen hin nur als ein Orb erscheinen.