JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

SolarSystem

Systemtechnik Labor
5BHIT 2015/16, Gruppe A

Hagen Aad Fock & Stefan Polydor

Version 0.1

Betreuer:
Note:

Begonnen am 3.11.2015
Beendet am 30.11.2015

Inhaltsverzeichnis

Einführung

Diese Aufgabe soll …

Ziele

Hier werden die zu erwerbenden Kompetenzen und deren Deskriptoren beschrieben. Diese werden von den unterweisenden Lehrkräften vorgestellt.

Dies kann natürlich auch durch eine Aufzählung erfolgen.

Voraussetzungen

Welche Informationen sind notwendig um die Laborübung reibungslos durchführen zu können? Hier werden alle Requirements der Lehrkraft detailliert beschrieben und mit Quellen untermauert.

Hier zum Beispiel die Architektur der Common Object-Request-Broker Architecture:

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 DONE
- Zumindest 2 Planeten, die sich um die eigene Achse und in elliptischen Bahnen um den Zentralstern drehen DONE
- Ein Planet hat zumindest einen Mond, der sich zusätzlich um seinen Planeten bewegt DONE
- Kreativität ist gefragt: Weitere Planeten, Asteroiden, Galaxien,... TODO
- Zumindest ein Planet wird mit einer Textur belegt (Erde, Mars,... sind im Netz verfügbar) DONE

Events:

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

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

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!

Projektbeschreibung

Teammitglieder/Rollen

Mitgliedsname	Rolle
Hagen Fock	Developer
Stefan Polydor	Developer

Tools

Tool	Toolname
IDE	PyCharm - v. 5
Framework	Panda3D - v. 1.8.1 & 1.9
Versionierungstool	Github
Programmiersprache	Python – 2.7

GUI-Skizzen und Bedienkonzept

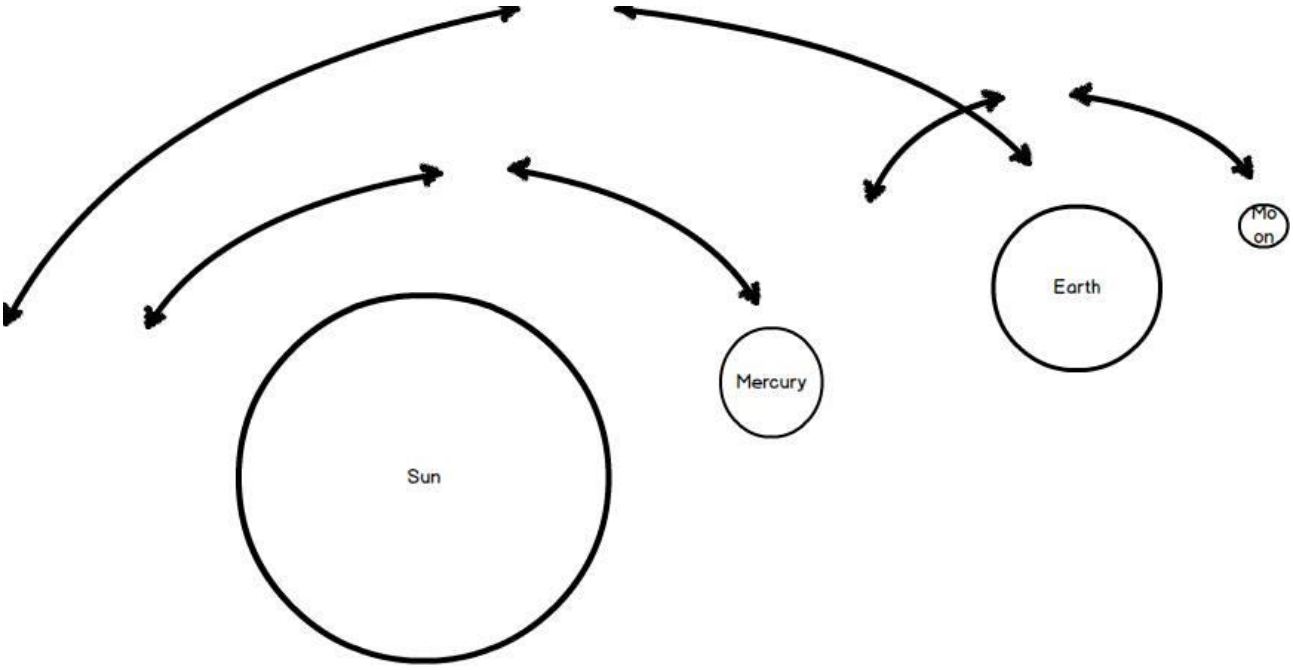

Abbildung 1 Balsamiq

Bedienkonzept

- 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.

Auswahl der Technologie

Wir hatten uns drei Technologien angeschaut:

- Pyglet [1]
- Panda3D [2]
- Pygame [3]

Wir haben uns nach ausprobieren aller Frameworks für Panda3D entschieden. Panda 3D hat 24 Samples. Darunter auch ein Sample über ein Solarsystem mit Texturen für die Planeten. Innerhalb der 24 Samples findet man alle Funktionen und Implementierung die man benötigt um die Aufgabenstellung umzusetzen.

Ergebnisse

Pattern

Factory Pattern

Chain-of-responsibility pattern

Ein zentraler Stern

Als zentralen Stern haben wir klarerweise die Sonne gewählt. Wir realisieren unsere Lösung für „Ein zentraler Stern“, indem wir `render` zum Mittelpunkt unseres Systems machen und alle Planeten an den Mittelpunkt anbringen. Man kann das wie eine CD sehen. Wenn der Mittelpunkt sich dreht, dreht sich alles mit.

Beispiel: Sonne

```
# Hier wird die Form fuer die Sonne geladen
# In diesem Fall ist eine planet_sphere
self.sun = loader.loadModel("../../models/planet_sphere")
# Hier wird die Sonne ins Zentrum des SolarSystems platziert
self.sun.reparentTo(render)
# Hier wird der Sonne die gelbe Sonnen Textur geladen
self.sun_tex = loader.loadTexture("../../models/sun_1k_tex.jpg")
# Hier wird die Textur gesetzt
self.sun.setTexture(self.sun_tex, 1)
# Hier wird die Groesse des Himmelskoerper gesetzt
self.sun.setScale(2 * self.sizescale)
```

Beispiel: Erde

```
#Hier wird die Erde an die Sonne/render (den Mittelpunkt) angehaengt
self.orbit_root_earth = render.attachNewNode('orbit_root_earth')
```

Ein Planet hat zumindest einen Mond, der sich zusätzlich um seinen Planeten bewegt

Bei dieser Aufgabe haben wir anstatt der Sonne die Erde zu dem Mittelpunkt für den Mond festgelegt.

```
#Hier wird die Erde an die Sonne/render (den Mittelpunkt) angehaengt
self.orbit_root_earth = render.attachNewNode('orbit_root_earth')
```

```
# Hier wird der Mond an die Erde gehaengt
self.orbit_root_moon = (
    self.orbit_root_earth.attachNewNode('orbit_root_moon'))
```

Zumindest 2 Planeten, die sich um die eigene Achse und in elliptischen Bahnen um den Zentralstern drehen

Siehe Codeausschnitt

```
def loadEarth(self):
    #Hier wird die Erde an die Sonne/render (den Mittelpunkt) angehaengt
    self.orbit_root_earth = render.attachNewNode('orbit_root_earth')
    # Load earth
    self.earth = loader.loadModel("../../models/planet_sphere")
    self.earth_tex = loader.loadTexture("../../models/earth_1k_tex.jpg")
    self.earth.setTexture(self.earth_tex, 1)
    self.earth.reparentTo(self.orbit_root_earth)
    self.earth.setScale(self.sizescale)
    self.earth.setPos(self.orbitscale, 0, 0)
# end loadEarth

def rotateEarth(self):
    # earth
    self.orbit_period_earth = self.orbit_root_earth.hprInterval(
        self.yearscale, (360, 0, 0))
    self.day_period_earth = self.earth.hprInterval(
        self.dayscale, (360, 0, 0))

def loadMars(self):
    self.orbit_root_mars = render.attachNewNode("orbit_root_mars")

    # Load Mars
    self.mars = loader.loadModel("../../models/planet_sphere")
    self.mars_tex = loader.loadTexture("../../models/mars_1k_tex.jpg")
    self.mars.setTexture(self.mars_tex, 1)
    self.mars.reparentTo(self.orbit_root_mars)
    self.mars.setPos(1.52 * self.orbitscale, 0, 0)
    self.mars.setScale(0.515 * self.sizescale)

def rotateMars(self):
    self.orbit_period_mars = self.orbit_root_mars.hprInterval(
        (1.881 * self.yearscale), (360, 0, 0))
    self.day_period_mars = self.mars.hprInterval(
        (1.03 * self.dayscale), (360, 0, 0))
```

Zumindest ein Planet wird mit einer Textur belegt (Erde, Mars,… sind im Netz verfügbar)

Siehe Codeausschnitt

```
def loadMars(self):
    self.orbit_root_mars = render.attachNewNode("orbit_root_mars")

    # Load Mars
    self.mars = loader.loadModel("../../models/planet_sphere")
    self.mars_tex = loader.loadTexture("../../models/mars_1k_tex.jpg")
    self.mars.setTexture(self.mars_tex, 1)
    self.mars.reparentTo(self.orbit_root_mars)
    self.mars.setPos(1.52 * self.orbitscale, 0, 0)
    self.mars.setScale(0.515 * self.sizescale)
```

Mittels Maus kann die Kameraposition angepasst werden: Zumindest eine Überkopf-Sicht und parallel der Planentenbahnen

In Panda3D wenn man mit ShowBase arbeitet ist prinzipiell schon eine „Kamera“ für die Maus implementiert. Damit meine ich, dass wenn man sie nicht explizit ausstellt (*base.disableMouse()*) kann man sich mit der Maus in der erstellten Umgebung umsehen.

- Rechte Maustaste ➔ Rein und raus zoomen
- Linke Maustaste ➔ sich in die gewünschte Richtung verschieben
- Maus-Rad ➔ Sich auf dem Punkt wo man sich befindet umschauen

Quellen

- [1] Pyglet
  - Online: https://bitbucket.org/pyglet/pyglet/wiki/Home
    - Zuletzt besucht am 24.11.2015
- [2] Panda3D
  - Online: https://www.panda3d.org/
    - Zuletzt besucht am 24.11.2015
- [3] Pygame
  - Online: http://pygame.org/hifi.html
    - Zuletzt besucht am 24.11.2015