Solarsystem

Softwareentwicklung

Philipp Adler, Adin Karic

Inhaltsverzeichnis

1. Projektbeschreibung

1.1. Anforderungen

2. Anforderungen	Datum	Geschätzte Zeit	Benötigte Zeit	Zuständig
Zentraler Stern (Sonne)	22.11.15	10min	30min	Adler
2 Planeten	22.11.15	20min	30min	Karic
Planettextur	22.11.15	10min	30min	Karic
Planet um eigene Achse	22.11.15	30min	60min	Adler
Elliptische Bahn um Zentralstern	22.11.15	50min	90min	Adler
Weitere Planeten	22.11.15	40min	60min	Karic

Events	Datum	Geschätzte Zeit	Benötigte Zeit	Zuständig
Kamerposition anpassen	29.11.2015	60min	40min	Adler
Start/Stop der Animation	27.11.2015	15min	80min	Adler
Textierung Ein/Aus	29.11.2015	20min	50min	Karic
Punktlichtquelle + Schatten	29.11.2015	120min	90min	Karic

2.1. Teammitglieder

Zur Realisierung der Software ist natürlich ein ausreichend qualifiziertes und motiviertes Team nötig. Dieses Team besteht aus Philipp Adler und Adin Karic. Beide derzeit Schüler am TGM mit Fachrichtung Informationstechnologie. Beide Teammitglieder nehmen im Team dieselben gleichberechtigten Rollen ein (Entwickler/Projektmanager)

2.2. Rollen

Beide Teammitglieder nehmen im Team dieselben gleichberechtigten Rollen ein (Entwickler/Projektmanager)

2.3. Tools

Prinzipiell wird in Python programmiert. Genutzt wird das Panda3D-Framework. Entwickelt wird in der Entwicklungsumgebung PyCharm. Zur Versionskontrolle wird Git verwendet (git@github.com:padler-tgm/SOLARSYSTEM-ADLER-KARIC.git) Für die Teamkommunikation wurde meistens Skype verwendet.

3. Evaluierte Frameworks

3.1. Pygame

(http://www.pygame.org/) Pygame stellt eine Sammlung von Modulen zur Verfügung, die es dem Programmierer ermöglichen ein Python basiertes Spiel zu realisieren. Da Pygame auf Python basiert, ist es Plattform unabhängig, sofern dort ein Interpreter installiert ist. Für das Ausführen muss Pygame importiert werden. Nähere Informationen sieht man in dem folgenden Codeabschnitt:

```
import pygame
def main():
        pygame.init()//Pygame-Module initialisieren
        screen = pygame.display.set_mode((800, 600))//Bildschirmgröße
        pygame.display.set_caption("SEW")//Titel
        pygame.mouse.set_visible(1)//sichtbarer Mousezeiger
        pygame.key.set_repeat(1, 1)//Tastdrücke
        clock = pygame.time.Clock()//Fragmente begrenzen
        running = True
        while running://Spielbetrieb
        clock.tick(30)//30 F/s
        screen.fill((0, 0, 0))//schwarzer Hintergrund
        for event in pygame.event.get()://Events abfragen
            if event.type == pygame.QUIT://Spiel beenden
            running = False
        pygame.display.flip()//Inhalt anzeigen
main()
```

3.2. Panda3D

(https://www.panda3d.org) Panda3d ist eine ausgereifte Spieleengine. Durch verwendung der Scriptsprache Python ist es möglich, komplexe Aufgaben schnell zu lösen um Prototypen herzustellen. Die Prototypen können problemlos erweitert werden und zu komplexen Systemen zusammengestellt werden. Hier sehen wir uns ebenfalls einen Codeabschnitt an.

```
import direct.directbase.DirectStart  #Initialize Panda and create a window
from panda3d.core import *     #Contains most of Panda's modules
from direct.gui.DirectGui import *    #Imports Gui objects we use for putting
import sys
class World
        def __init__(self):
        self.title = OnscreenText(
        text="Panda3D: Tutorial 1 - Solar System",
        style=1, fg=(1,1,1,1), pos=(0.8,-0.95), scale = .07)
        base.setBackgroundColor(0, 0, 0)//Backroundcolor black
        base.disableMouse()//disable Mouse
        camera.setPos ( 0, 0, 45 )//Kameraposition
        camera.setHpr ( 0, -90, 0 )//Kameraorientierung

w = World()
run()
```

	PyGame	Panda3D
Installation	8/10 Nach einiger Recherche erfolgreich	10/10 Gut beschrieben & ohne Probleme
Komplexität/ Handhabung	8/10 Man verliert am Anfang ein wenig den Überblick	8/10 Nach einigem Einlesen akzeptabel
Dokumentation	6/10 Viele Tutorials, Doku gibt's auch (wenn auch nicht so gut)	8/10 Sehr ausführliches Manual vorhanden
(Lizenz-)kosten	10/10 Freie Nutzung	10/10 Freie Nutzung
Community	7/10 Ist vorhanden, aber nicht so gut wie bei Panda3D	9/10 Forum, IRC-Channel und Blog sind vorhanden
Prototyp (Example)	10/10 Ufo-Spiel als Example Funktionierte einwandfrei	10/10 SolarSystem als Example Funktionierte einwandfrei
Gesamtpunktzahl	**49/60**	**55/60**

Aufgrund der obigen Bewertung der beiden Frameworks (mit den gewählten Kriterien) ist die Benutzung von Panda3D für unser Vorhaben zu präferieren.

4. Designüberlegungen

Planet(Elternklasse)

- Koordinaten(x,y,z)
- Name
- Rotationsspeed
- Translationsspeed
- Textur, Farbe
- Größe dafür benutzen wir eine Skalierungsvariable
- Move Objekt für die Rotation und Translation

Planeten(Kind)

- Koordinaten(x,y,z)
- Name
- Textur, Farbe
- Größe

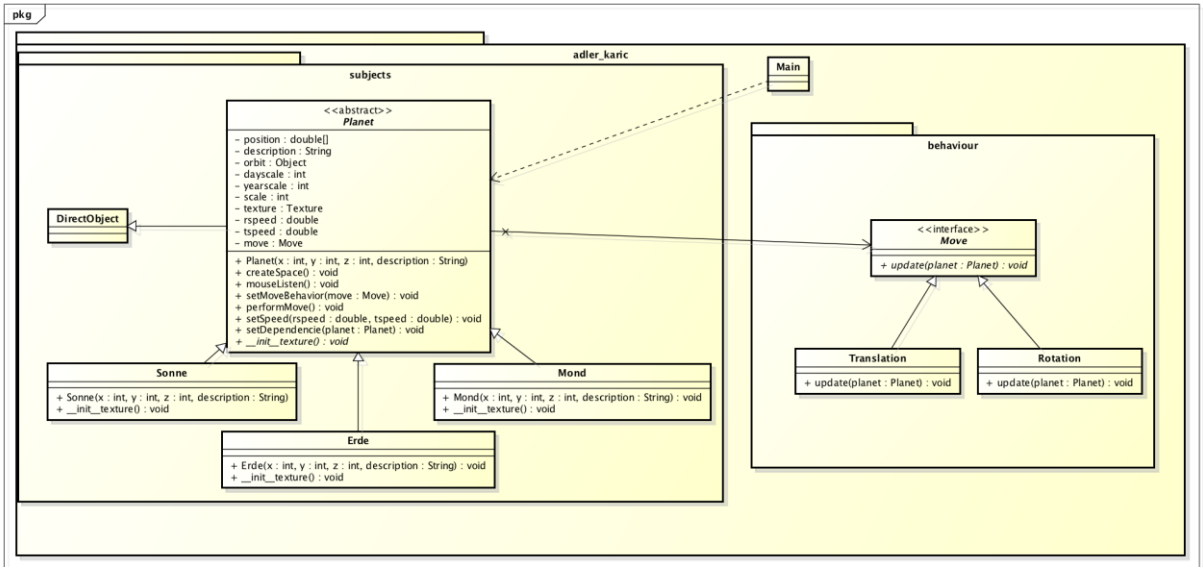
Mond(Kind-Kind)

- Koordinaten(x,y,z)
- Name
- Textur, Farbe
- Größe
- Abhängigkeit

Move(Algorithmen)

- Rotation des Planeten(Rotationsspeed)
- Translation(Laufbahn) um abhängigen Planeten(Translationsspeed)

Skizzen:



5. Technische Dokumentation

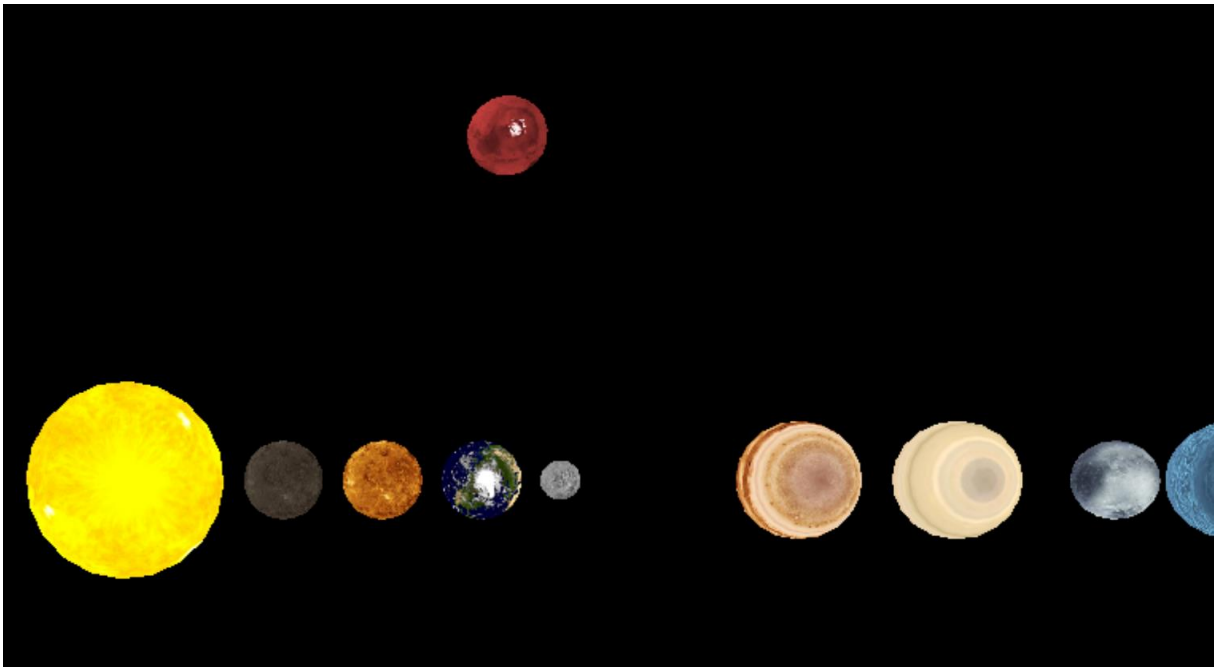
5.1. Design Pattern

Zur Implementierung der Software wird die Anwendung der folgenden Design-Patterns angedacht:

- **Strategy**
  Da sich die Planeten mit verschiedener Geschwindigkeit, sowie verschiedenen Rotationseigenschaften fortbewegen, lagern wir das Verhalten unserer Himmelskörper aus und wenden dadurch das Strategy-Pattern an.

- **Decorator** (erst ansatzweise vorhanden)
  Durch das Decorator-Pattern sollen Planeten sich sowohl fortbewegen als auch rotieren können. Wir decorieren die Planeten, also mit diesen beiden Fähigkeiten.

- **Factory** (erst ansatzweise vorhanden)
  Zur Erzeugung von Planeten wird die Implementierung von Planeten-Factories angedacht.



6. Bedienungsanleitung

Der erste Schritt ist der Start des Programms. Nun finden Sie sich im Weltraum wieder. Sie können sich im Weltraum durch die Verwendung Ihrer Maus fortbewegen und drehen.

Außerdem stehen folgende Funktionen zur Verfügung:

Aktion	Auslöser
Anzeigen der Hilfe	h
Programm beenden	esc
Textur an/aus	t
Punktlichtquelle an/aus	Ctrl+Linksklick
Geschw. der Planeten rauf/runter	Mausrad rauf/runter
Start/Stopp der Animation	space

7. Probleme

Dieses Kapitel dient zur Beschreibung der aufgetretenen und bis zum jetzigen Stand (30.11.2015) noch nicht gelösten Problemen.

Das erste Problem bezieht sich auf die Kamerasicht im Programm. Uns ist es noch nicht gelungen um die Z-Achse zu rotieren und somit ist die Fortbewegung im "Universum" ein wenig eingeschränkt.

Das zweite Problem ist die Positonierung der Punktlichtquelle. Logisch gesehen wäre die richtige Position der Punktlichtquelle die Mitte der Sonne. Wenn man das so macht wird das Licht auch richtig auf die Planeten geworfen und der Schatten etc. stimmen auch. Das Problem hierbei ist, dass die Sonne vollkommen dunkel erscheint wenn man die Punktlichtquelle in ihrer Mitte positioniert. Man könnte nun natürlich mehrere Punktlichtquellen neben die Sonne setzen um sie "künstlich" zu beleuchten. Dabei ist aber das Problem, dass die anderen Planeten falsch belichtet werden.

Ein weiteres Problem trat bei uns bei der Regelung der Geschwindigkeit (Translation wie auch Rotation) der Planeten auf. Dabei konnte man zunächst ganz normal die Geschwindigkeit durch Scrollen rauf und runter setzen. Wenn man jedoch eine bestimmte Geschwindigkeit überschritt begann das Programm zu "laggen".

Das letzte von uns noch nicht behobene Problem ist dass der Mond sich nicht korrekt um die Erde bewegt.