Solarsystem



**Softwareentwicklung**

**Philipp Adler, Adin Karic**

Inhaltsverzeichnis

1. Projektbeschreibung 3

1.1. Anforderungen 3

2.1. Teammitglieder 3

2.2. Rollen 3

2.3. Tools 3

3. Evaluierende Frameworks 4

3.1. Pygame 4

3.2. Panda3D 4

4. Design 5

5. Technische Dokumentation 7

5.1. Design Patter 7

6. Bedienungsanleitung 7

# Projektbeschreibung

# Anforderungen

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1. 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 | 80min | Karic |
| Textur Asteroiden |  | 20min |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Events | Datum | Geschätzte Zeit | Zuständig | Erledigt |
| Kamerposition anpassen |  | 60min |  |  |
| Start/Stop der Animation |  | 15min |  |  |
| Textierung Ein/Aus |  | 20min |  |  |
| Schatten von Planeten |  | 120min |  |  |

# 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)

# Rollen

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

# Tools

Als Entwicklungsumgebung wird PyCharm verwendet. Zur Versionskontrolle wird Git benutzt.

# Evaluierende Frameworks

# 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()*

# 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.

# Design

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

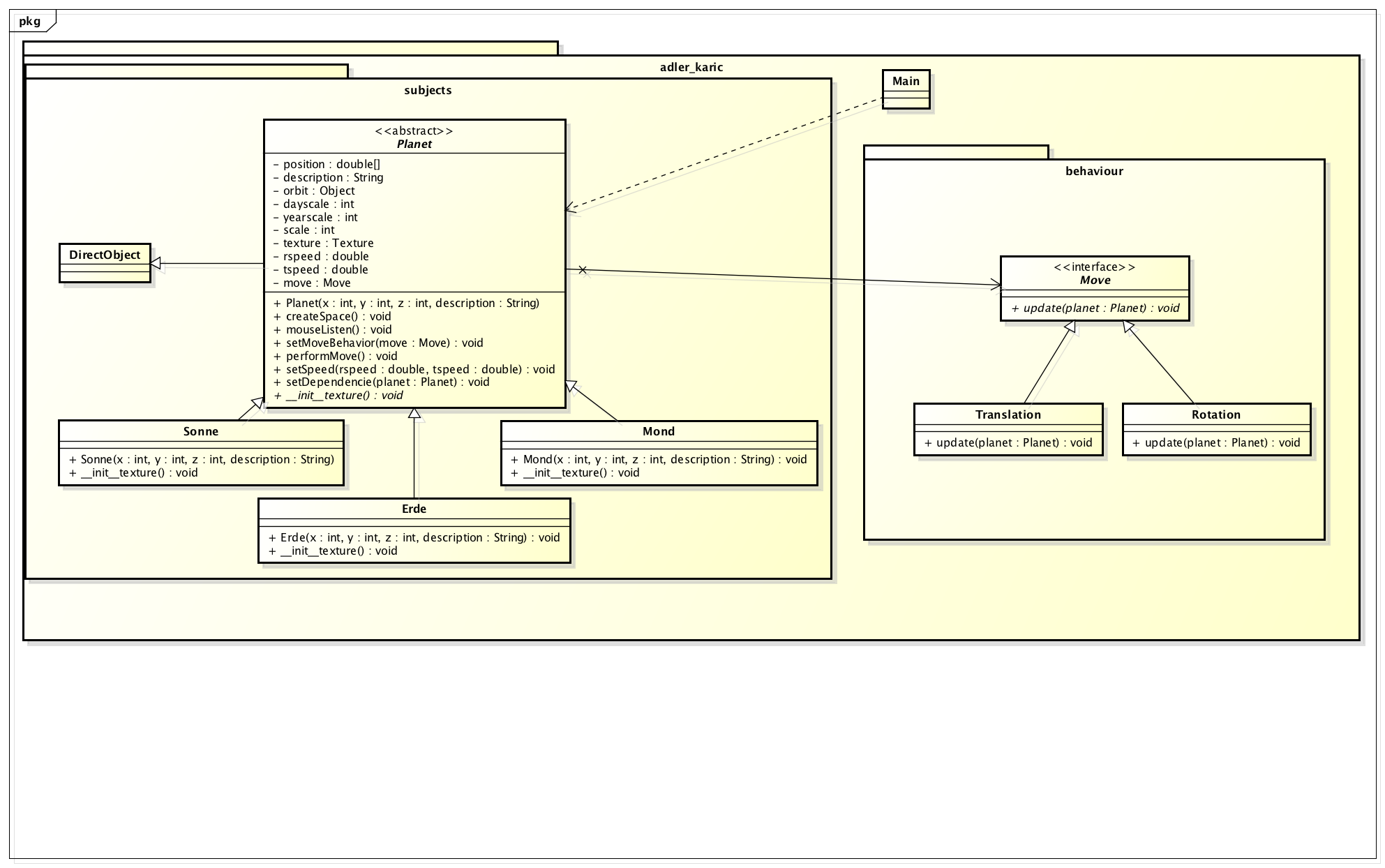
Mond(Kind-Kind)

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

Move(Algorithmen)

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

Skizzen:



# Technische Dokumentation

# Design Patter

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

* **Strategy**

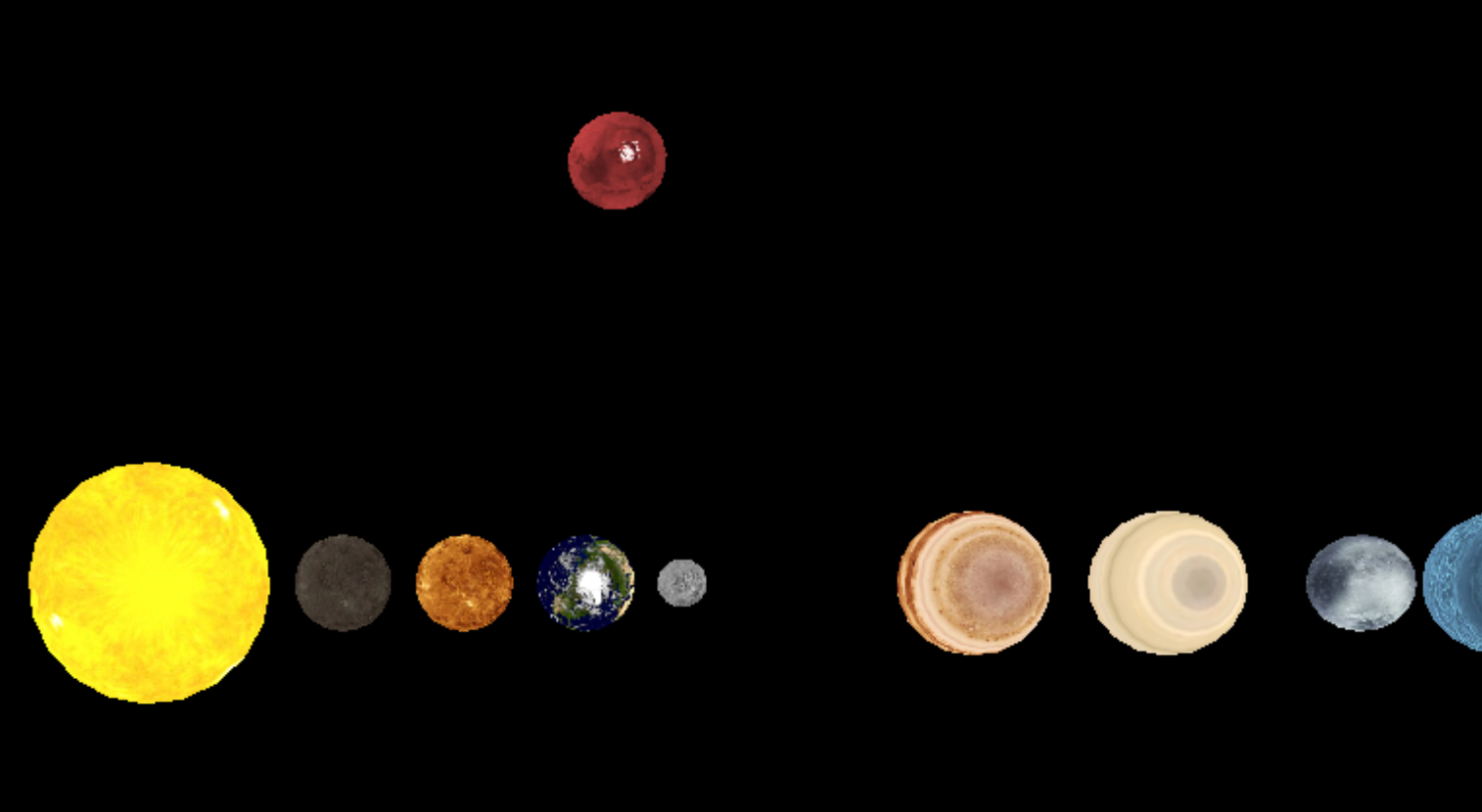
Da sich die Planeten mit verschiedener Geschwindigkeit, sowie verschiedenen Rotationseigenschaften fortbewegen, lagern wir das Verhalten unserer Himmelskörper aus und wenden dadurch das Strategy-Pattern an.

* **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.



# Bedienungsanleitung