

Visit our solar system

Daniel Bracher, Martin Suschny

Inhaltsverzeichnis

[1 Aufgabenstellung 1](#_Toc413666411)

[1.1 Zusätzliche Information 1](#_Toc413666412)

[2 Zeitaufzeichnung 2](#_Toc413666413)

[3 Design-Entwurf 3](#_Toc413666414)

[3.1 UML-Diagramm 3](#_Toc413666415)

[3.2 GUI 3](#_Toc413666416)

[4 Arbeitsvorgang 4](#_Toc413666417)

[4.1 Bracher 4](#_Toc413666418)

[4.2 Suschny 4](#_Toc413666419)

[5 Testdokumentation 5](#_Toc413666420)

[Quellen 6](#_Toc413666421)

# 1 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.

## 1.1 Zusätzliche Information

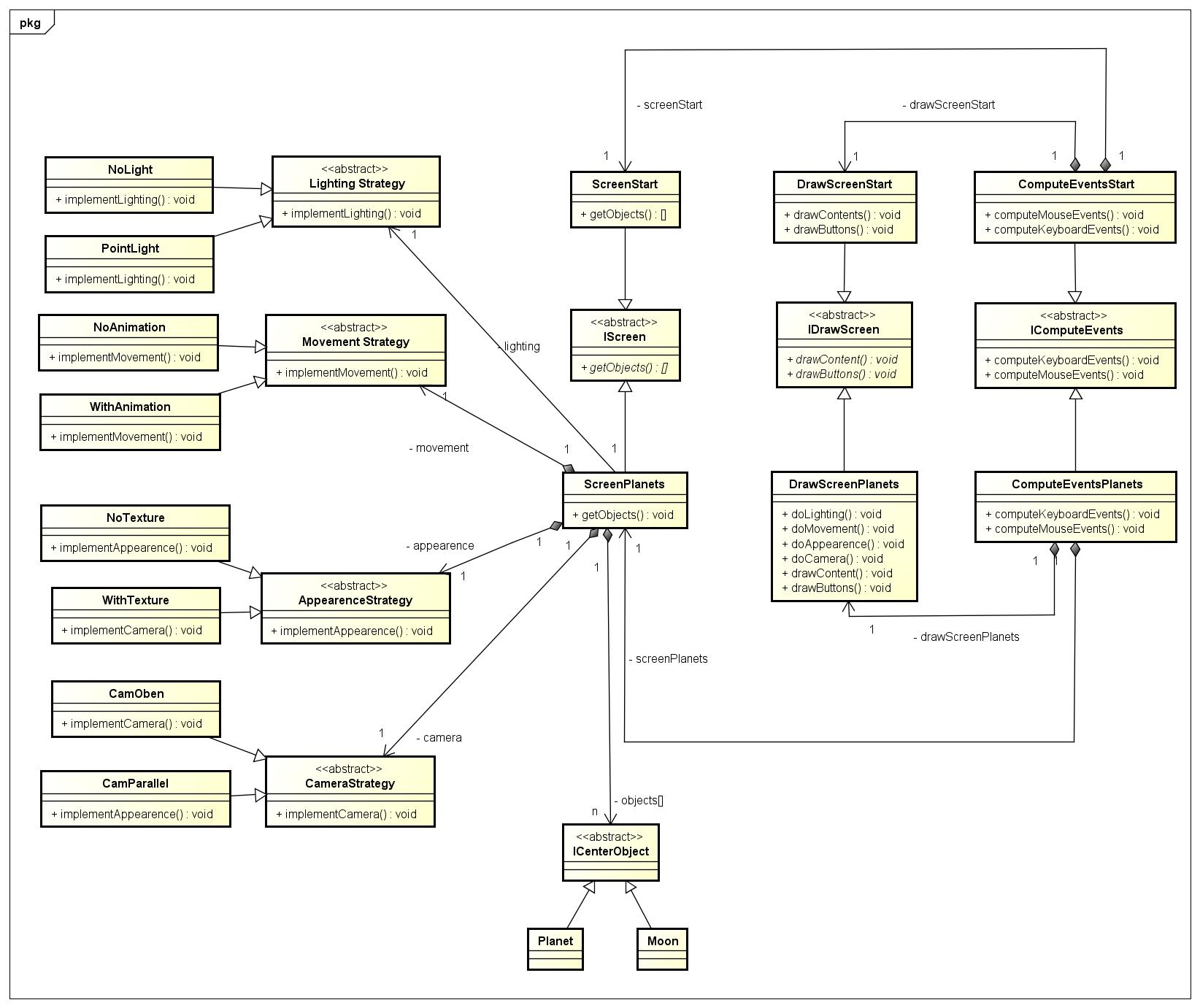
* Ein Objekt kann einfach mittels glutSolidSphere() erstellt werden.
* Die Planten werden mittels Modelkommandos bewegt: glRotate(), glTranslate()
* Die Kameraposition wird mittels gluLookAt() gesetzt
* Bedenken Sie bei der Perspektive, dass entfernte Objekte kleiner - nahe entsprechende größer darzustellen 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.

# 2 Zeitaufzeichnung

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Userstories | | Priorität | Verantwortliche | Zeit [min] | |
|  |  |  |  | geschätzt | tatsächlich |
| 1) Vorbereitung | |  |  |  |  |
|  | Libraries-Recherche | HIGH | DB/MS | 60 | 60 (DB) |
|  | Anlernen der Libaries | HIGH | DB/MS | 240 | 30 (DB) |
| 2) Designüberlegungen | |  |  |  |  |
|  | SW-Design | HIGH | DB | 180 | 90 (DB) |
|  | GUI-Design | HIGH | MS | 60 |  |
|  | UAT Überlegung (UserAcceptanceTest) | HIGH | DB/MS | 240 |  |
| 3) Implementierung | |  |  |  |  |
| 3.1) OpenGL | |  |  |  |  |
|  | Objekte erstellen | HIGH | DB | 90 | 60 (DB) |
|  | - Zentraler Stern |  |  |  |  |
|  | - 2 Planeten |  |  |  |  |
|  | - 1 Mond |  |  |  |  |
|  | Objekte mit Texturen belegen | MEDIUM | DB | 180 |  |
|  | - Texturen suchen |  |  |  |  |
|  | Lichtquelle erstellen | MEDIUM | MS | 120 |  |
|  | Schattenberechnung implementieren | LOW | DB |  |  |
|  | Kamera(s) erstellen | LOW | MS | 120 |  |
|  | Animation implementieren | HIGH | DB | 240 |  |
|  | - Rotationshierarchie erstellen |  |  |  |  |
| 3.2) GUI | |  |  |  |  |
|  | Splashscreen erstellen | HIGH | MS | 180 |  |
|  | Planeten GUI erstellen | HIGH | DB | 90 | 60 (DB) |
|  | Start GUI erstellen | HIGH | MS | 90 |  |
| 3.3) Eventkoordinierung | |  |  |  |  |
|  | Kameraposition umschalten | MEDIUM | MS | 120 |  |
|  | Animation starten/stoppen | MEDIUM | DB | 120 |  |
|  | Geschwindigkeit beschleunigen/starten | LOW | DB | 120 |  |
|  | Licht an-/ausschalten | MEDIUM | MS | 120 |  |
|  | Spiel starten | HIGH | MS | 120 |  |
| 4) Testing | |  |  |  |  |
|  | - UAT | HIGH | DB/MS | 360 |  |
| 5) Dokumentation | |  |  |  |  |
|  | - Sphinx-Doku | MEDIUM | DB | 120 |  |
|  | - Projektprotokoll | HIGH | MS | 120 |  |
|  |  |  | Daniel |  | 5h |
|  |  |  | Martin |  |  |
|  |  |  | Gesamt: |  |  |

# 3 Design-Entwurf

## 3.1 UML-Diagramm



## 3.2 GUI

# 4 Arbeitsvorgang

## 4.1 Bracher

1.3.2015 -> UML-Diagramm entworfen

3.3.2015 -> UML Diagramm aktualisiert (ICenterObject hinzugefügt)

8.3.2015 -> UML Diagramm aktualisiert (einzelne Relationen hinzugefügt)

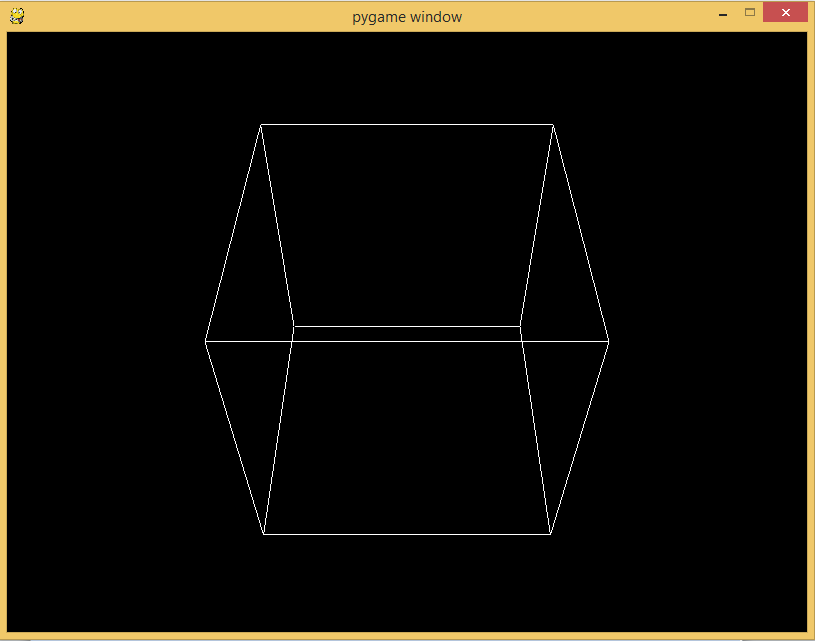
Klassen und Interfaces nach UML-Diagramm erstellt

Mittels PyQt GUI fuer die Planeten erstellt

PyQt-Code in Programm eingefügt

Problem: Controller der Planeten kann nur von einem Interface erben, nicht von zwei (Qt- und Controller-Interface)  
Loesungsansatz1: Problem koennte an gleicher Benennung der Funktionen liegen

9.3.2015-> Implementierung des im Tutorial angefuehrten Codes (GUI + sich drehender Wuerfel) [1]



## 4.2 Suschny

# 5 Testdokumentation

# Quellen

1. Online-Tutorial; <https://www.youtube.com/watch?v=R4n4NyDG2hI>; zuletzt besucht: 9.3.2015