

Visit our solar system

Daniel Bracher, Martin Suschny

Inhaltsverzeichnis

1 Aufgabenstellung 1

1.1 Zusätzliche Information 1

2 Zeitaufzeichnung 2

3 Design-Entwurf

­­­3.1 UML-Diagramm 4

3.2 GUI 4

3.3 Verwendete Libraries 4

4 Arbeitsvorgang 8

5 Testdokumentation 11

Quellen 12

# 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]** | | | | **Status** | | | | |
|  |  |  |  | **gesch.(MS)** | **gesch. (DB)** | **tats. (MS)** | **tats. (DB)** | **Design** | **Implem.** | **Test** | **Doku** | **Fertig** |
| **1) Vorbereitung** | |  |  |  |  |  |  |  |  |  |  |  |
|  | Libraries-Recherche | HIGH | DB/MS | 30 | 30 | **90** | **60** |  |  |  |  |  |
|  | Anlernen der Libaries | HIGH | DB/MS | 120 | 120 | **120** | **60** |  |  |  |  |  |
| **2) Designüberlegungen** | |  |  |  |  |  |  |  |  |  |  |  |
|  | SW-Design | HIGH | DB |  | 180 |  | **90** |  |  |  |  |  |
|  | GUI-Design | HIGH | MS | 60 |  | **30** |  |  |  |  |  |  |
|  | UAT Überlegung (UserAcceptanceTest) | HIGH | DB/MS | 120 | 120 |  |  |  |  |  |  |  |
| **3) Implementierung** | |  |  |  |  |  |  |  |  |  |  |  |
| **3.1) OpenGL** | |  |  |  |  |  |  |  |  |  |  |  |
|  | Objekte erstellen | HIGH | DB |  | 150 |  | **60** |  |  |  |  |  |
|  | Objekte mit Texturen belegen | MEDIUM | DB |  | 240 |  | **30** |  |  |  |  |  |
|  | Lichtquelle erstellen | MEDIUM | MS | 180 |  |  |  |  |  |  |  |  |
|  | Schattenberechnung implementieren | LOW | MS | 180 |  |  |  |  |  |  |  |  |
|  | Kamera(s) erstellen | LOW | DB |  | 180 |  | **60** |  |  |  |  |  |
|  | Animation implementieren | HIGH | DB |  | 300 |  | **60** |  |  |  |  |  |
| **3.2) Start-Menü** | |  |  |  |  |  |  |  |  |  |  |  |
|  | Splashscreen erstellen | HIGH | MS | 360 |  | **120** |  |  |  |  |  |  |
| **3.3) Eventkoordinierung** | |  |  |  |  |  |  |  |  |  |  |  |
|  | Kameraposition umschalten | MEDIUM | DB |  | 180 |  | **60** |  |  |  |  |  |
|  | Animation starten/stoppen | MEDIUM | DB |  | 180 |  | **60** |  |  |  |  |  |
|  | Geschwindigkeit beschleunigen/starten | LOW | DB |  | 180 |  | **60** |  |  |  |  |  |
|  | Licht an-/ausschalten | MEDIUM | MS | 180 |  |  |  |  |  |  |  |  |
|  | Spiel starten | HIGH | MS | 180 |  | **60** |  |  |  |  |  |  |
| **5) Dokumentation** | |  |  |  |  |  |  |  |  |  |  |  |
|  | - Sphinx-Doku | MEDIUM | DB/MS | 60 | 60 |  |  |  |  |  |  |  |
|  | - Projektprotokoll | HIGH | DB/MS | 60 | 60 | **60** | **120** |  |  |  |  |  |
|  |  |  |  | 25,5 | 33 | 8 | 12 |  |  |  |  |  |
|  |  |  | **Gesamt (h):** | **58,5** | | **20** | |  |  |  |  |  |
|  |  |  |  | gesch.(MS) | gesch. (DB) | tats. (MS) | tats. (DB) |  |  |  |  |  |

# 3 Design-Entwurf **3.1 UML-Diagramm**



## 3.2 GUI

## 3.3 Verwendete Libraries

#### 3.3.1 PyGame 1.9.2

PyGame ist eine „Cross-Platorm“- Bibliothek, entwickelt um einfach Multimedia-Software, wie Spiele, in Python zu schreiben.

**Code-Snippet:**

|  |
| --- |
| import pygame, os  from pygame.locals import \*  from math import sin  main\_dir = os.path.split(os.path.abspath(\_\_file\_\_))[0]  def main():  #initialize and setup screen  pygame.init()  screen = pygame.display.set\_mode((640, 480), HWSURFACE|DOUBLEBUF) |

#### 3.3.2 PyOpenGL 3.1

PyOpenGL ist die häufigste Cross-Plattform-Python-Anbindung an OpenGL und verwandte APIs.

**Code-Snippet:**

|  |
| --- |
| from OpenGL.GLUT import \*  from OpenGL.GLU import \*  from OpenGL.GL import \*  import sys  name = 'ball\_glut'  def main():  glutInit(sys.argv)  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH)  glutInitWindowSize(400,400)  glutCreateWindow(name)  glClearColor(0.,0.,0.,1.)  glShadeModel(GL\_SMOOTH)  glEnable(GL\_CULL\_FACE)  glEnable(GL\_DEPTH\_TEST)  glEnable(GL\_LIGHTING)  lightZeroPosition = [10.,4.,10.,1.]  lightZeroColor = [0.8,1.0,0.8,1.0] #green tinged  glLightfv(GL\_LIGHT0, GL\_POSITION, lightZeroPosition)  glLightfv(GL\_LIGHT0, GL\_DIFFUSE, lightZeroColor)  glLightf(GL\_LIGHT0, GL\_CONSTANT\_ATTENUATION, 0.1)  glLightf(GL\_LIGHT0, GL\_LINEAR\_ATTENUATION, 0.05)  glEnable(GL\_LIGHT0)  glutDisplayFunc(display)  glMatrixMode(GL\_PROJECTION)  gluPerspective(40.,1.,1.,40.)  glMatrixMode(GL\_MODELVIEW)  gluLookAt(0,0,10,  0,0,0,  0,1,0)  glPushMatrix()  glutMainLoop()  return  def display():  glClear(GL\_COLOR\_BUFFER\_BIT|GL\_DEPTH\_BUFFER\_BIT)  glPushMatrix()  color = [1.0,0.,0.,1.]  glMaterialfv(GL\_FRONT,GL\_DIFFUSE,color)  glutSolidSphere(2,20,20)  glPopMatrix()  glutSwapBuffers()  return  if \_\_name\_\_ == '\_\_main\_\_': main() |

# 4 Arbeitsvorgang

1.3.2015 -> UML-Diagramm entworfen

3.3.2015 -> UML Diagramm aktualisiert (ICenterObject hinzugefügt)

8.3.2015 -> UML Diagramm aktualisiert (einzelne Relationen hinzugefügt)

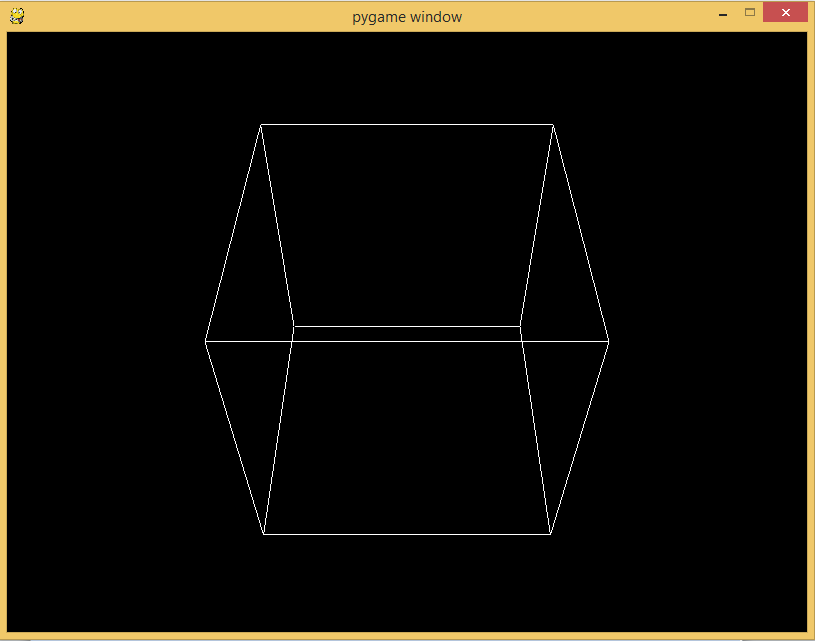
Klassen und Interfaces nach UML-Diagramm erstellt

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]



10.3.2015-> Dynamisches Zeichnen aller in der Model-Klasse erstellten Objekte

11.3.2015 -> BIG RELEASE  
  
Mehrstufiges Abfragen der Events implementiert

def computeKeyboardEvents(self, event):

"""

Verarbeitet die Keyboard-Events

:return: Nichts

"""

#if event.key == ­­­­­­.K\_LEFT:

def computeMouseEvents(self, event):

"""

Verarbeitet die Mouse-Events

:return: Nichts

"""

if event.type == pygame.QUIT: #Falls das Fenster geschlossen werden soll

pygame.quit()

quit()

elif event.type == pygame.MOUSEBUTTONDOWN: #Falls etwas mit der Maus gedrückt wurde

if event.button == 1: #Falls linke Maustaste gedrückt wurde

self.screenContent.changeMovement() #Movement-Strategie aendern

elif event.button == 3: #Falls rechts Maustaste gedrückt wurde

self.screenContent.changeLighting()

elif event.button == 4: #Falls Maus-Rad nach vorne gescrollt wird

glTranslatef(0.0,0.0,1)

elif event.button == 5: #Falls Maus-Rad nach hinten gescrollt wird

glTranslatef(0.0,0.0,-1)

Aendern der Movement-Strategie und Lighting-Strategie während der Laufzeit implementiert

def changeMovement(self):

#Falls Movement derzeit NoAnimation-Strategie implementiert

if isinstance(self. movement, NoAnimation):

#Auf WithAnimation-Strategie aendern

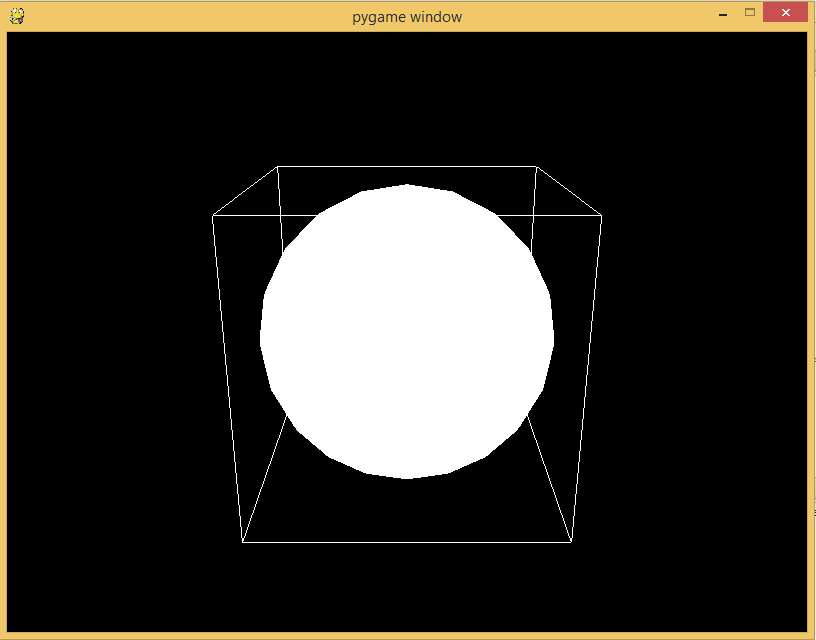
self.movement = WithAnimation()

#Falls WithAnimation-Strategie implementiert

elif isinstance(self.movement, WithAnimation):

#Auf NoAnimation-Strategie aendern

self.movement = NoAnimation()  
Sonne als Kugel implementiert



# 5 Testdokumentation

# Quellen

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