SOLAR-SYSTEM

Protokoll

Cindy Lehner und Lukas Zainzinger, 21.12.2015, 5AHITM

Inhalt

Git-Repository: https://github.com/lzainzinger/sunsystem

1. Aufwandsschätzung

Art	Stunde(n)
Recherche	5
Installation	2-3
Programmieren	11
Protokoll	1

Aufwand in Stunden: 20-21

2. Zeitaufzeichnung

In Minuten.

Lehner		Zainzinger	
15.11.2015	60	15.11.2015	60
22.11.2015	75	22.11.2015	75
26.11.2015	90	26.11.2015	110
29.11.2015	40	29.11.2015	50
14.12.2015	80	13.12.2015	130
20.12.2015	150	20.12.2015	150
21.12.2015	150	21.12.2015	210

3. Tatsächlicher Aufwand

In Stunden

Soll-Zustand	Ist-Zustand
20-21	24

4. Aufgabenstellung

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.

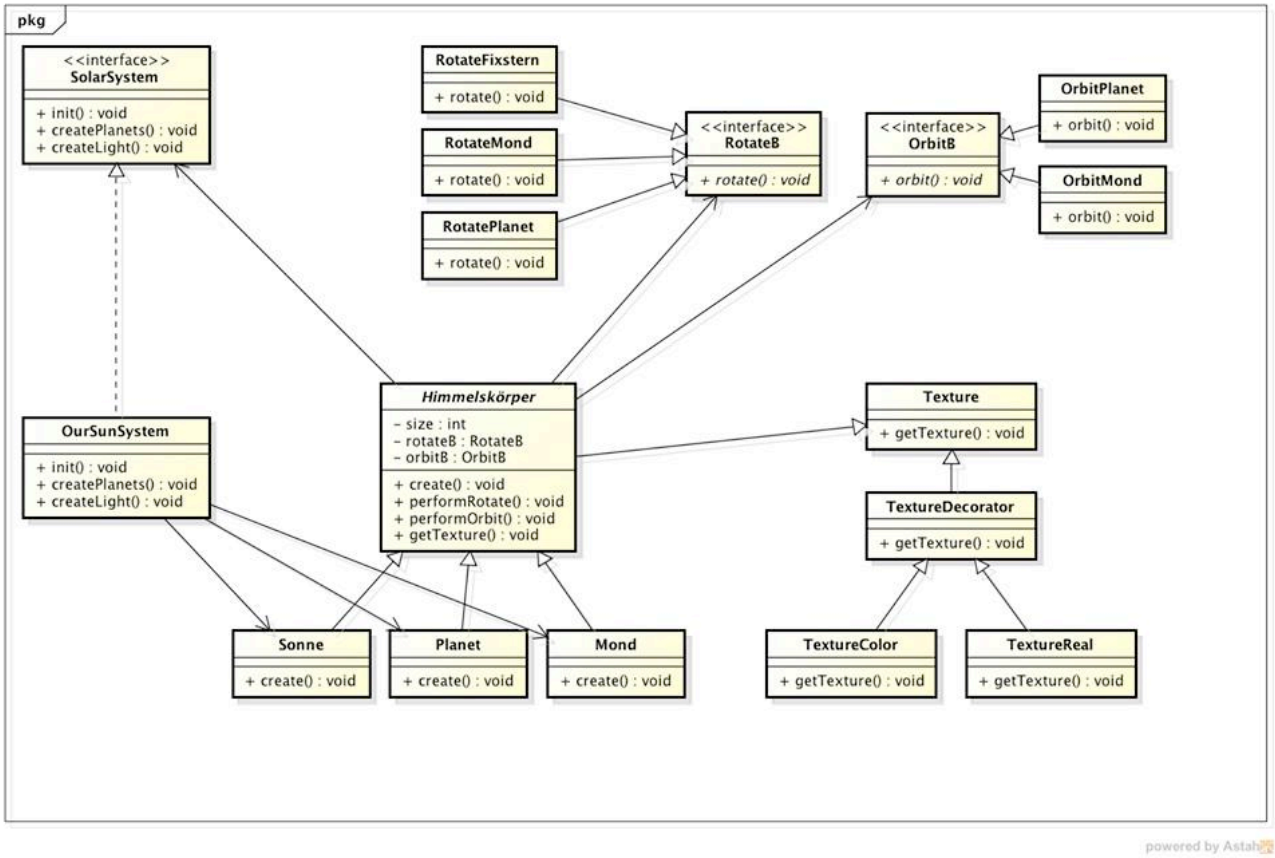
Hinweise:

- 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.

Tutorials:

- Pygame: https://www.youtube.com/watch?v=K5F-aGDIYaM

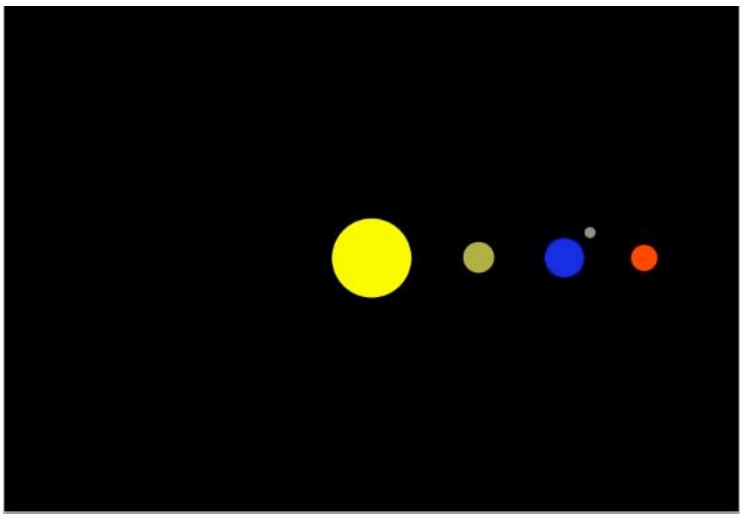
5. UML-Diagramm



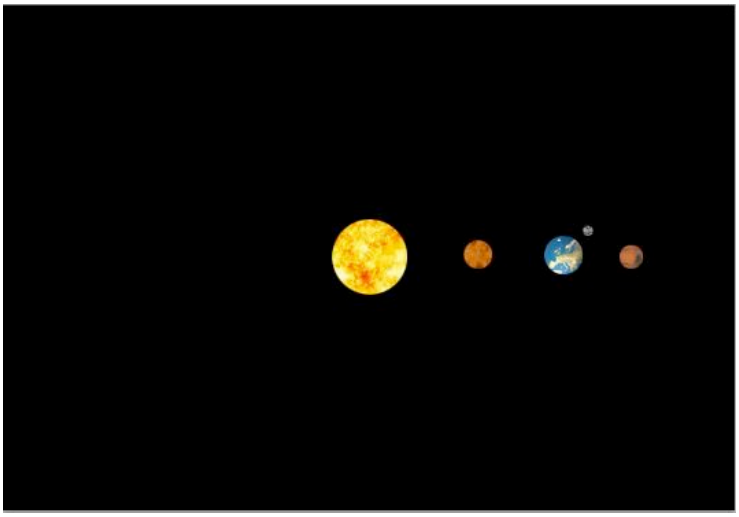
6. Benutzer-Interface

Wir werden unser Solarsystem mit der Sonne, drei Planeten (Venus, Erde und Mars) und einem Mond, der um die Erde kreist, ausstatten. Für Erstbenutzer, beziehungsweise um die verschiedenen Eigenschaften, die unser Programm bietet einzusehen, gibt es ein Hilfe-Menü.

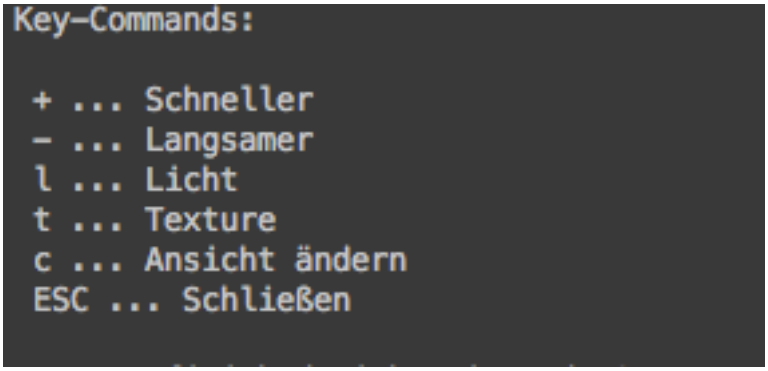
Sonnen-System:



Mit Textur:



Hilfe:



7. Installation

Wir haben unser Projekt mit PyGame umgesetzt. Damit das Ganze funktioniert müssen vorerst ein paar Libraries installiert werden.

Python

PyGame

Pillow

PyOpenGL

8. Code

7.1 Planeten

Wir haben drei Planeten: die Venus, die Erde und den Mars. Planeten werden mit folgenden Zeilen erstellt.

```
def sphereErde():
    glColor3f(0.0, 0.0, 1.0)
    glutSolidSphere(0.35, 30, 30)
```

Mit glColor3f wird die Farbe festgesetzt. glutWireSphere bestimmt den Radius und gibt dann an, wieviele Wires gezeichnet werde. Da es solid ist sieht man diese aber natürlich nicht.

```
# Venus
glRotatef(360.0 * venusDayOfYear / 225, 0.0, 1.0, 0.0)
glPushMatrix()
glTranslatef(2.0, 0.0, 0.0)
glTranslatef(1.0, 0.0, 0.0)
glRotatef(360.0 * dayOfYear / 243.0, 0.0, 1.0, 0.0)
sphereVenus()
glPopMatrix()

# Erde
glRotatef(360.0 * dayOfYear / 365.0, 0.0, 1.0, 0.0)
glPushMatrix()
glTranslatef(5.0, 0.0, 0.0)
glRotatef(360.0 * hourOfDay / 24.0, 0.0, 1.0, 0.0)
sphereErde()

# Mond
glRotatef(360.0 * 12.0 * dayOfYear / 365.0, 0.0, 1.0, 0.0)
glPushMatrix()
glTranslatef(0.5, 0.0, 0.0)
sphereMond()
glPopMatrix()
glPopMatrix()

# Mars
glRotatef(360.0 * marsDayOfYear / 545, 0.0, 1.0, 0.0)
glPushMatrix()
glTranslatef(7.0, 0.0, 0.0)
glRotatef(360 * dayOfYear / 24.0, 0.0, 1.0, 0.0)
sphereMars()
glPopMatrix()
```

In der Rotate Funktion berechnen wir uns die Umlaufbahn um die Sonne und um die eigne Achse. Der Mond dreht sich um die Erde und ist deshalb innerhalb der Matrix der Erde.

7.2 Fixstern

Unser Fixstern, als das Zentrum um das sich die Planeten drehen, ist die Sonne.

```
def sphereSonne():
    glColor3f(1.0, 1.0, 0.0)
    glutSolidSphere(1.0, 50, 50)
```

7.3 Mond

Mond sind die einzigen Sphären, die sich nicht um die Sonne, sondern einen Planeten drehen. Daher befindet er sich in der Matrix der Erde.

```
# Erde
glRotatef(360.0 * dayOfYear / 365.0, 0.0, 1.0, 0.0)
glPushMatrix()
glTranslatef(5.0, 0.0, 0.0)
glRotatef(360.0 * hourOfDay / 24.0, 0.0, 1.0, 0.0)
sphereErde()

if texture:
    setupTexture(textureMond)

# Mond
glRotatef(360.0 * 12.0 * dayOfYear / 365.0, 0.0, 1.0, 0.0)
glPushMatrix()
glTranslatef(0.5, 0.0, 0.0)
sphereMond()
glPopMatrix()
glPopMatrix()
```

7.4 Licht

Mit der Taste „L“ wird das Licht ein- und ausgeschalten.

```
def setupLighting():

    zeros = (0.15, 0.15, 0.15, 0.3)
    ones = (1.0, 1.0, 1.0, 0.3)
    half = (0.5, 0.5, 0.5, 0.5)

    glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, zeros)
    glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, half)
    glMaterialf(GL_FRONT_AND_BACK, GL_SHININESS, 15)
    glLightfv(GL_LIGHT0, GL_AMBIENT, zeros)
    glLightfv(GL_LIGHT0, GL_DIFFUSE, ones)
    glLightfv(GL_LIGHT0, GL_SPECULAR, half)
    glEnable(GL_LIGHT0)
    glEnable(GL_LIGHTING)
    glColorMaterial(GL_FRONT_AND_BACK, GL_DIFFUSE)

    glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, GL_SPHERE_MAP)
    glTexGeni(GL_T, GL_TEXTURE_GEN_MODE, GL_SPHERE_MAP)
    glEnable(GL_TEXTURE_GEN_S)
    glEnable(GL_TEXTURE_GEN_T)

    glEnable(GL_COLOR_MATERIAL)
    glEnable(GL_NORMALIZE)
    glShadeModel(GL_SMOOTH)
```

Anschließend wird nur noch programmiert, dass bei dem Tastaturbefehl, die Funktion ausgeführt wird.

7.5 Texturen

Die Funktion getImage() holt sich die Bilder für die Planeten vom Dateipfad und lädt sie. Mit setUpTexture werden die Texturen auf die Planeten gelegt und wenn das richtige KeyEvent gedrückt wird angezeigt.

```
def getImage(pic):

    dateipfad = "images/" + pic + ".jpg"
    print(dateipfad)

    try:
        image = Image.open(dateipfad)

        x, y = image.size

        image = image.convert("RGBA").tobytes("raw", "RGBA")
        textur = glGenTextures(1)
        glBindTexture(GL_TEXTURE_2D, textur)
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR)
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_NEAREST)
        gluBuild2DMipmaps(GL_TEXTURE_2D, 3, x, y, GL_RGBA, GL_UNSIGNED_BYTE, image)

        return textur
    except:
        raise FileNotFoundError("Textur konnte nicht geladen werden")


def setupTexture(imgID):
    glEnable(GL_TEXTURE_2D)
    glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST)
    glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST)
    glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL)

    glBindTexture(GL_TEXTURE_2D, imgID)
```

```
# Texturen laden
textureSun = getImage('sun')
textureMars = getImage('mars')
textureVenus = getImage('venus')
textureErde = getImage('erde')
textureMond = getImage('mond')
```

7.6 Kamera Perspektive

Die Kameraperspektive wird durch den Befehl „glLookAt()“ geändert. In unserem Projekt gibt es eine Draufsicht und eine Ansicht von der Seite.

```
gluLookAt(0, looky, lookz,
          0, 0, 0,
          0, 0, 1)
```

Die Werte werden mithilfe einer IF-Anweisung verändert:

```
elif k[K_c]:
    if looky == 8:
        looky = 0.1
        lookz = 7
    else:
        looky = 8
        lookz = 0
```

7.7 Key-Events

Um die verschiedenen Events auszuführen, haben wir uns jeweils einen Buchstaben auf der Tastatur ausgesucht.

```
# Event-Handling
for event in pygame.event.get():
    k = pygame.key.get_pressed()
    # Quit-Handling
    if event.type == pygame.QUIT:
        pygame.quit()
        quit()
    if k[K_p]:
        if paused:
            paused = False
        else:
            paused = True
    elif k[K_PLUS]:
        if animateIncrement < 150:
            animateIncrement = animateIncrement + 10
    elif k[K_MINUS]:
        if animateIncrement > 10:
            animateIncrement = animateIncrement - 10
    elif k[K_t]:
        if texture:
            texture = False
            glDisable(GL_TEXTURE_2D)
        else:
            texture = True
            glEnable(GL_TEXTURE_2D)
            glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, GL_SPHERE_MAP)
            glTexGeni(GL_T, GL_TEXTURE_GEN_MODE, GL_SPHERE_MAP)
            glEnable(GL_TEXTURE_GEN_S)
            glEnable(GL_TEXTURE_GEN_T)
```

```
elif k[K_l]:
    if light:
        light = False
        glDisable(GL_LIGHTING)
        glDisable(GL_LIGHT0)
    else:
        light = True
        setupLighting()
elif k[K_ESCAPE]:
    pygame.quit()
    quit()
elif k[K_c]:
    if looky == 8:
        looky = 0.1
        lookz = 7
    else:
        looky = 8
        lookz = 0
```

Mit „L“ wird das Licht ein- und ausgeschalten. Mit „P“ wird das Bild gestoppt und auf Pause gesetzt. Mit „+“ bzw. „-„ werden die Planeten schneller bzw. langsamer gemacht. Mit „T“ wird die Textur ein- oder ausgeschalten und mit

7.8 Hilfe-Menü

Nachdem wir Probleme bei unserem erst gedachten Design hatten, haben wir uns dazu entschieden am Anfang beim Starten des Programmes alle Key-Events in der Konsole auszugeben.

```
# Beschreibung
print("Key-Commands: \n  \n + ... Schneller \n - ... Langsamer \n l ... Licht \n t ... Texture \n c ... Ansicht ändern \n ESC ... Schließen")
```

```
Key-Commands:

 + ... Schneller
 - ... Langsamer
 l ... Licht
 t ... Texture
 c ... Ansicht ändern
 ESC ... Schließen
```

9. Quellen

PyGame Tutorials:
https://www.youtube.com/watch?v=K5F-aGDIYaM

Tutorial für das Erstellen von Buttons:
https://pythonprogramming.net/pygame-buttons-part-1-button-rectangle/

Die PyGame Dokumentation generell:
https://www.pygame.org/docs/

Die API von OpenGL im Allgmeinen:
https://www.opengl.org/documentation/

http://www.stackoverflow.com/

https://pythonprogramming.net/