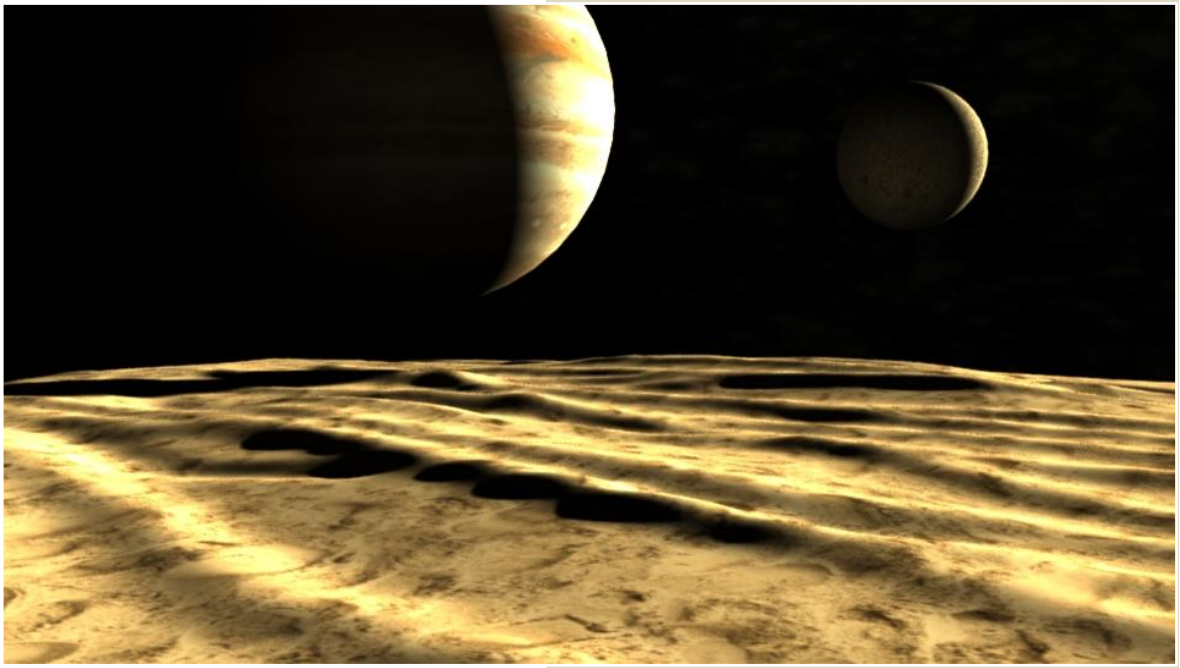
2015

Visit our solar system



Herczeg | Krickl

5BHITM

23.02.2015

Sonnensystem

Inhaltsverzeichnis

Aufgabenstellung

Wir wollen nun unser Wissen aus Medientechnik und SEW nützen um eine etwas kreativere Applikation zu erstellen.

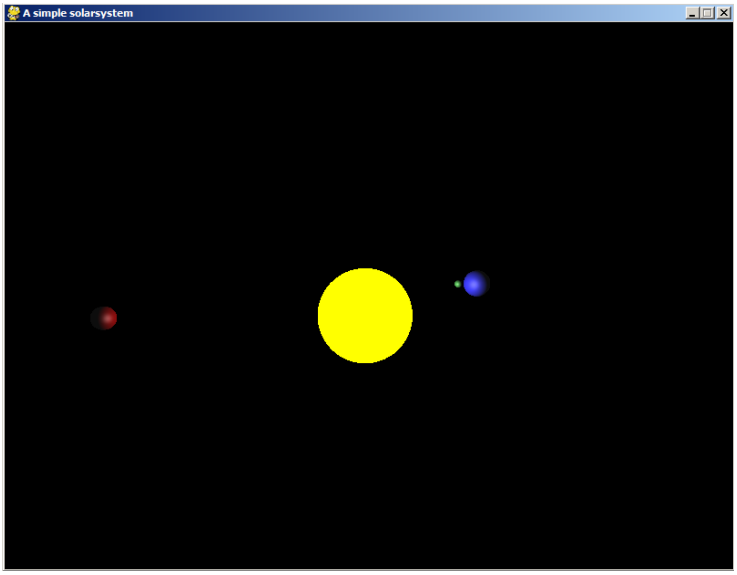
Eine wichtige Library zur Erstellung von Games mit 3D-Grafik ist Pygame. Die 3D-Unterstützung wird mittels PyOpenGL erreicht.

Die Kombination ermöglicht eine einfache und schnelle Entwicklung.

Während pygame sich um Fensteraufbau, Kollisionen und Events kümmert, sind grafische Objekte mittel OpenGL möglich.

Die 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.

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
-

Viel Erfolg!

Lighting

```
        def setupLighting():
                """ Initializing Lighting and Light0

                :return:
                """
                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)
```

Zeitaufzeichnung

Anforderung	Priorität	Verantwortlicher	Zeit (G) [min]	Zeit (T) [min]	Status (D,I,Te,Do,F, A)
Nicht funktional					
Recherche der Libraries und OpenGL	Mittel	K, H	120	120	I
Installation der Libraries	Hoch	K, H	20	30	F
Installation von OpenGL	Mittel	K, H	10	5	F
Texturen suchen und aufbereiten	Mittel	K	60	20	I
Planung					
GUI planen	Mittel	K	100	10	F
GUI erstellen	Hoch	K	100		I
3D-Raum erstellen. Dh.: Die GUI auf eine 3D-Ansicht vorbereiten	Hoch	K	15	50	F
Erstellen der 3D-Objekte	Niedrig	K	5	15	F
Klassendiagramme bzw. Software planen und erstellen	Hoch	H	100	90	F
Implementierung					
Implementierung eines Gestirns Interfaces	Hoch	H	30		I
Implementierung der Translation eines Planeten	Hoch	H	10	40	I
Implementierung der Rotation eines Planeten	Hoch	H	10	20	I
Implementierung einer Mond-Klasse	Hoch	H	30		
Implementierung der Translation des Mondes	Hoch	H	10		

Implementierung der Rotation des Monds	Hoch	H	10		
Fixstern als Punktlicht implementieren	Mittel	K	60	40	I
Schattenberechnung implementieren	Niedrig	K, H	60		I
Kamera implementieren	Hoch	H	20		
Steuerung und Animation					
Benutzersteuerung implementieren	Hoch	K	20	20	F
Kameraposition anpassbar machen	Hoch	K	20	30	F
Animationen implementieren	Hoch	K	10	40	I
Animation stoppbar/startbar machen	Hoch	K	10		
Geschwindigkeit der Animation anpassbar machen	Hoch	H	10		
Licht ein/ausschaltbar machen	Mittel	K	5	20	I
Test und Abnahme					
Prototyp fix-fertig lauffähig machen	Hoch	K, H	15		
Testcases planen	Mittel	K, H	60		
Testcases schreiben	Hoch	K, H	35		
UACs planen	Mittel	K, H	60		
UACs durchführen	Mittel	K, H	30		
Beta-Tests durchführen	Niedrig	K, H	25		
DAU das Programm ausführen lassen	Niedrig	K, H	15		
Abnahme	Hoch	H	30		
Summe Herczeg [min]			542.5 (9h)	270	
Summe Krickl [min]			542.5 (9h)	270	

G … Geschätzt
T … Tatsächlich
D … Design
I … Implementierung
Te … Test
Do … Dokumentation
F … Fertig
A … Abgenommen

H … Herczeg

K … Krickl

Design

Libraries und Versionen

Python 3.4.

Pygame 1.9.

Wird noch nicht verwendet.

Pillow 2.7.1

Library zum Einbinden von Texturen

```
    @staticmethod
    def LoadTexture(pic):
        # Textur
        ix = image.size[0]
        iy = image.size[1]
        image = image.tostring("raw", "RGBX", 0, -1)

        # Textur erstellen
        textures = glGenTextures(1)
        glBindTexture(GL_TEXTURE_2D, textures)  # 2d texture (x and y size)

        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, ix, iy, GL_RGBA,
GL_UNSIGNED_BYTE, image)

        return textures
```

PyOpenGL 3.x.

Library um die GPU des PCs zu verwenden. Leichtes erstellen und einbinden von Standardformen möglich.

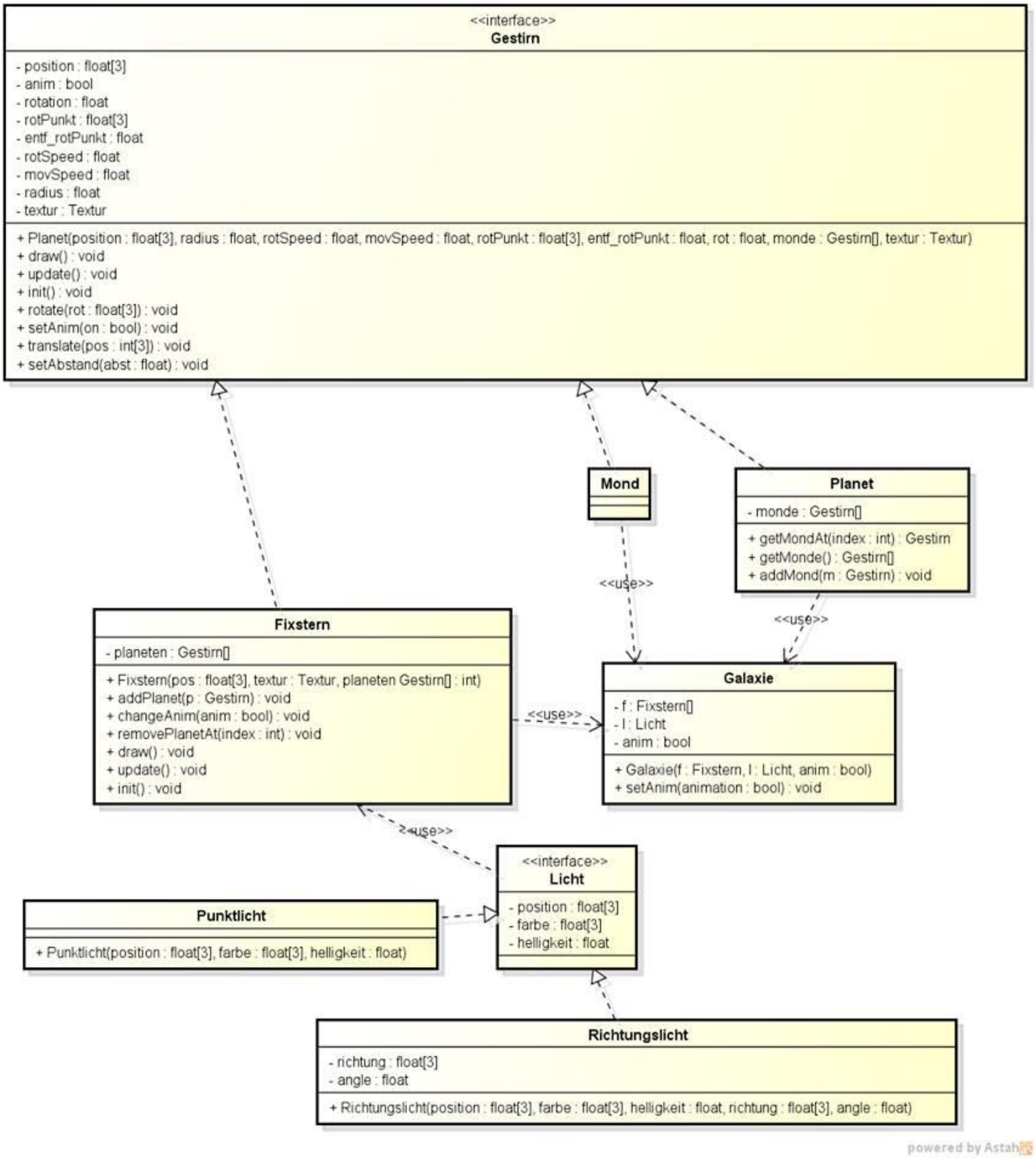
```
def InitGL(self, Width, Height):
                glEnable(GL_TEXTURE_2D)
        glClearColor(0.0, 0.0, 0.0, 0.0)    # Hintergrundfarbe
        glClearDepth(1.0)                   # Loeschen des Depth Buffers
```

```
        glDepthFunc(GL_LESS)                    # The Type Of Depth Test To Do
        glEnable(GL_DEPTH_TEST)                 # Enables Depth Testing
        glShadeModel(GL_SMOOTH)                 # Enables Smooth Color Shading
        glMatrixMode(GL_PROJECTION)             # Reset The Projection Matrix
        glLoadIdentity()

        # camera
        gluPerspective(45.0, float(Width) / float(Height), 0.1, 100.0)
        glMatrixMode(GL_MODELVIEW)

    """
    Wenn die groesse vom Fenster geaendert wird
    """
    def ReSizeGLScene(self, Width, Height):
        glViewport(0, 0, Width, Height
        glMatrixMode(GL_PROJECTION)
        glLoadIdentity()
        # Perspektive
        gluPerspective(50.0, float(Width) / float(Height), 0.1, 100.0)
        glMatrixMode(GL_MODELVIEW)


    """
    szene zeichenn
    """
    def DrawGLScene(self):

        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
        # Planet P1
        self.rot_pl2 = self.gestirn.rotation(self.rot_pl2, 0, 0.04, 0)
        self.gestirn.DrawGLScene_P(0.5, self.rot_pl2, self.light,0.8,0,-10)

        glutSwapBuffers()  # zeichnen

    def main(sc):
        #solarsystem
        glutInit(sys.argv)

        glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH)
        glutInitWindowSize(1000, 600)
        glutInitWindowPosition(50, 50)
        glutCreateWindow(b'Solarsystem')
        glutDisplayFunc(sc.DrawGLScene)
        glutIdleFunc(sc.DrawGLScene)
        glutReshapeFunc(sc.ReSizeGLScene)
        sc.InitGL(640, 480)
        glutMainLoop()


s = universe()
main(s)
```

Freeglut 2.8.1 32bit

Wird alternativ zum OpenGL Utility Toolkit in Kombination mit PyOpenGL verwendet. Beispielcode von PyOpenGL lässt sich nur mit Freeglut verwenden.

UML


<<interface>>
Gestirn
- position : float[3]
- anim : bool
- rotation : float
- rotPunkt : float[3]
- entf_rotPunkt : float
- rotSpeed : float
- movSpeed : float
- radius : float
- textur : Textur
+ Planet(position : float[3], radius : float, rotSpeed : float, movSpeed : float, rotPunkt : float[3], entf_rotPunkt : float, rot : float, monde : Gestirn[], textur : Textur)
+ draw() : void
+ update() : void
+ init() : void
+ rotate(rot : float[3]) : void
+ setAnim(on : bool) : void
+ translate(pos : int[3]) : void
+ setAbstand(abst : float) : void
Mond
Planet
- monde : Gestirn[]
+ getMondAt(index : int) : Gestirn
+ getMonde() : Gestirn[]
+ addMond(m : Gestirn) : void
<<use>>
<<use>>
Fixstern
- planeten : Gestirn[]
+ Fixstern(pos : float[3], textur : Textur, planeten Gestirn[] : int)
+ addPlanet(p : Gestirn) : void
+ changeAnim(anim : bool) : void
+ removePlanetAt(index : int) : void
+ draw() : void
+ update() : void
+ init() : void
<<use>>
Galaxie
- f : Fixstern[]
- l : Licht
- anim : bool
+ Galaxie(f : Fixstern, l : Licht, anim : bool)
+ setAnim(animation : bool) : void
<<use>>
<<interface>>
Licht
- position : float[3]
- farbe : float[3]
- helligkeit : float
Punktlicht
+ Punktlicht(position : float[3], farbe : float[3], helligkeit : float)
Richtungslicht
- richtung : float[3]
- angle : float
+ Richtungslicht(position : float[3], farbe : float[3], helligkeit : float, richtung : float[3], angle : float)
powered by Astah

Model
Man hat eine Galaxie die einen Fixstern besitzt, den man Planeten hinzufügen kann. Monde sind Planeten die man einem Planet hinzufügen kann.

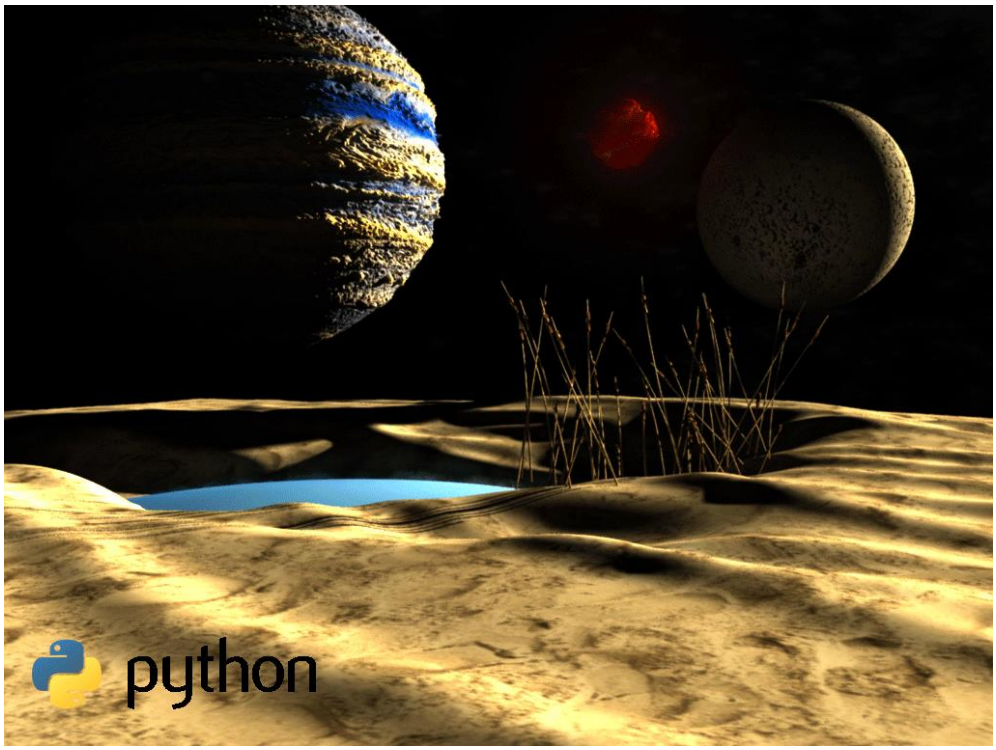
Control
Übernimmt die Steuerung bei Events (Mausklicks, Buttons,..).

Sie „verwaltet“ die Galaxie indem diverse Planeten mit Methoden hier in die View gesetzt und animiert werden.

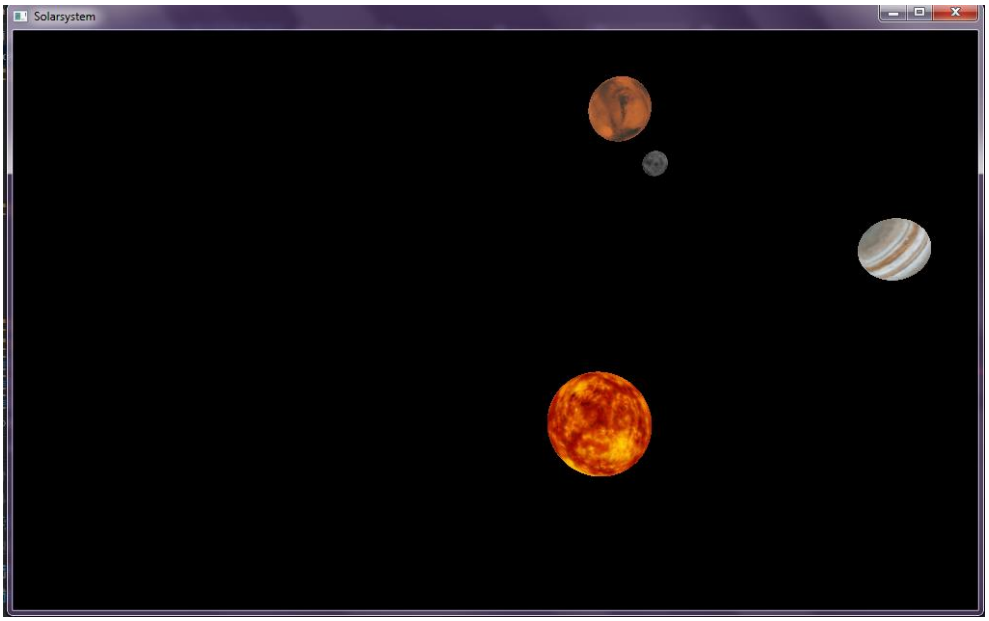
View
Planeten, Schatten, Texturen und Buttons werden alle angezeigt.

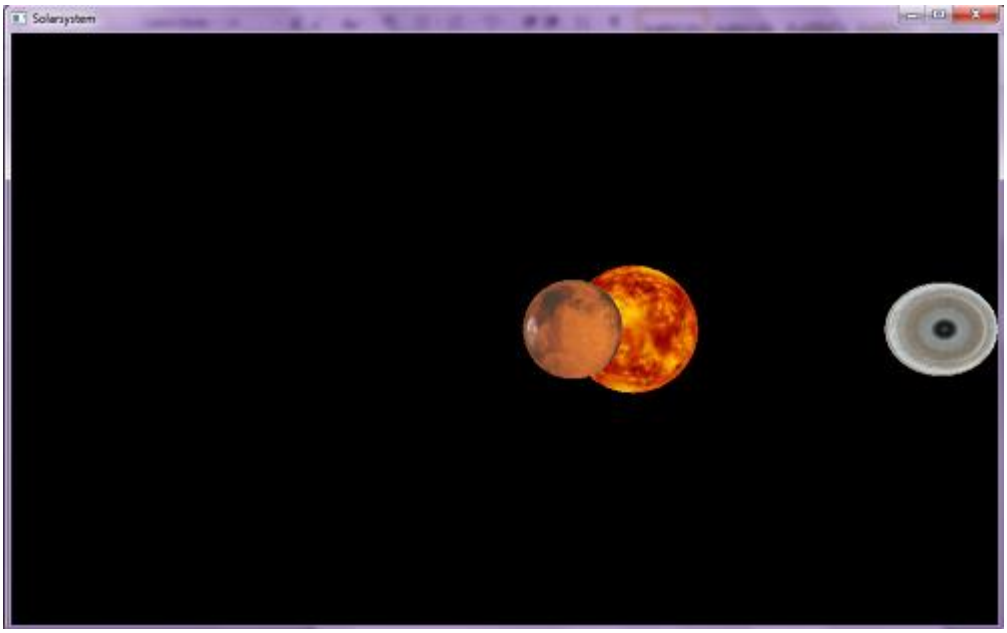
GUI

Splashscreen

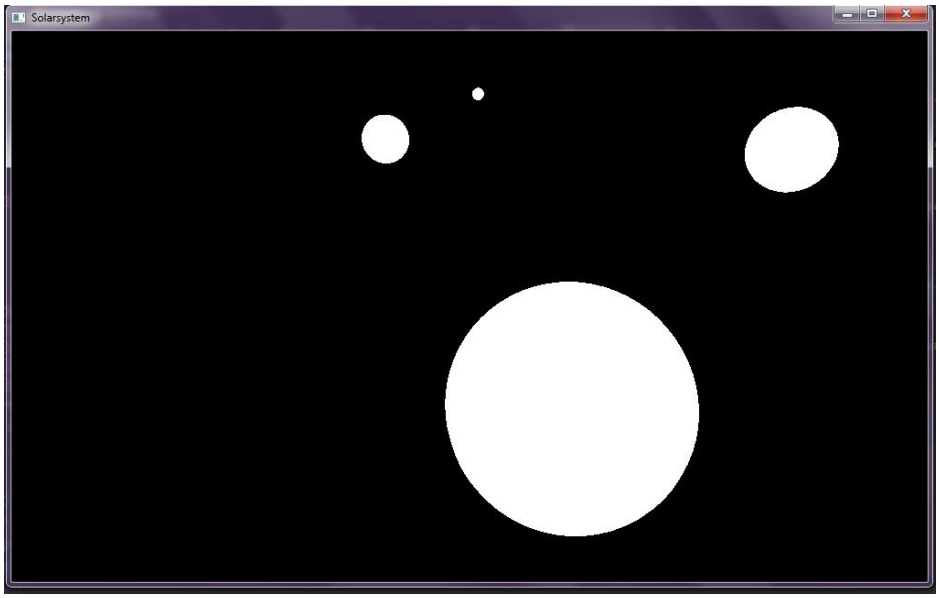


Perspektive seitlich von vorne. Taste ‚c‘ zum Perspektive ändern. Taste ‚p‘ unten rechts um die Animation zu pausieren und wieder anzuschalten.





Taste ‚t' um die Texturen auszuschalten.



Ansicht von oben. Taste ‚c' um die Perspektive zu ändern. Über ‚l' lässt sich die Belichtung an und ausschalten. Über die Pfeiltasten verändert sich die Geschwindigkeit der Animation.

Probleme

Installation von Pillow 2.7.1 auf Python 3.3 war nicht möglich, deshalb Umstieg auf Python 3.4, wo es problemlos möglich war.

Quellen

Kompletter Code siehe Github
https://github.com/akrickl-tgm/solar01.git

Im Text verwendet

Nachschlagen

PyOpenGL 3.x
http://pyopengl.sourceforge.net/
24.02.2015

Pillow 2.6.1
https://pypi.python.org/pypi/Pillow/2.6.1
24.02.2015

PyGame Download
http://www.pygame.org/download.shtml
24.02.2015

PyOpenGL Tutorial mcfeltch
http://bazaar.launchpad.net/~mcfletch/pyopengl-demo/trunk/view/head:/PyOpenGL-Demo/proesch/simple/simpleInteraction.py
03.03.2015