SOLAR SYSTEM

Dienesch Florian | Rathbauer Alexander

Table of Contents

No table of contents entries found.

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!

Zeitaufzeichnung

User Story	**Verantwortlicher**	**Zeit [std]**		**Status**			
		Gesch.	Wirkl	Impl	Tests	Doku	Fertig
Einarbeitung	D,R	4	3	X			
UML	R	3		Y			
Galaxy anzeigen	D	1	0.5	X			
Splashscreen erstellen	R	2	1	X			
Splashscreen einbinden	D	1	0.5	X			
Texturen erstellen oder finden	R	0.5	0.5	X			
Texturen auf Objekte legen	R	2	1	X			
Modelle erstellen	R	0.5	0.5	X			
Modelle um sich selbst drehen	R	4	1	X			
Modelle um andere Modelle drehen	R	4		Y			
Modelle in ellipsenbahn drehen lassen	D	4	2	X			
Animation	D	1	0.5	X			
Geschwindigkeit änderbar	R	1	0.5	X			
Licht Quelle gesetzt	D	6	3	X			
Texturen	R	6	3	X			
Steuerbare Kamera	D	3		Y			
Hintergrund	R	1.5	0.5	X			
Userinterface	D	2		Y			
Tests	D	1.5					
Zusammenfassung		46.0	17.5				

D…Dienesch

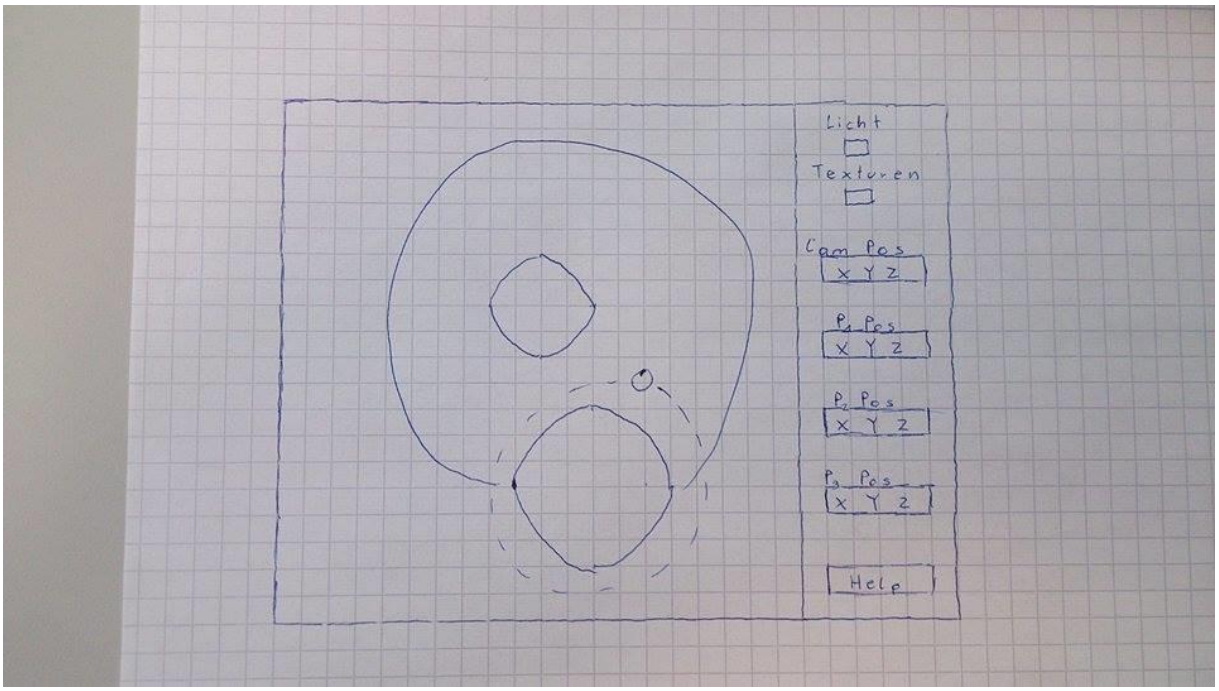
R…Rathbauer

X…Fertig

Y…In Arbeit

Arbeitsschritte

Skizze Layout



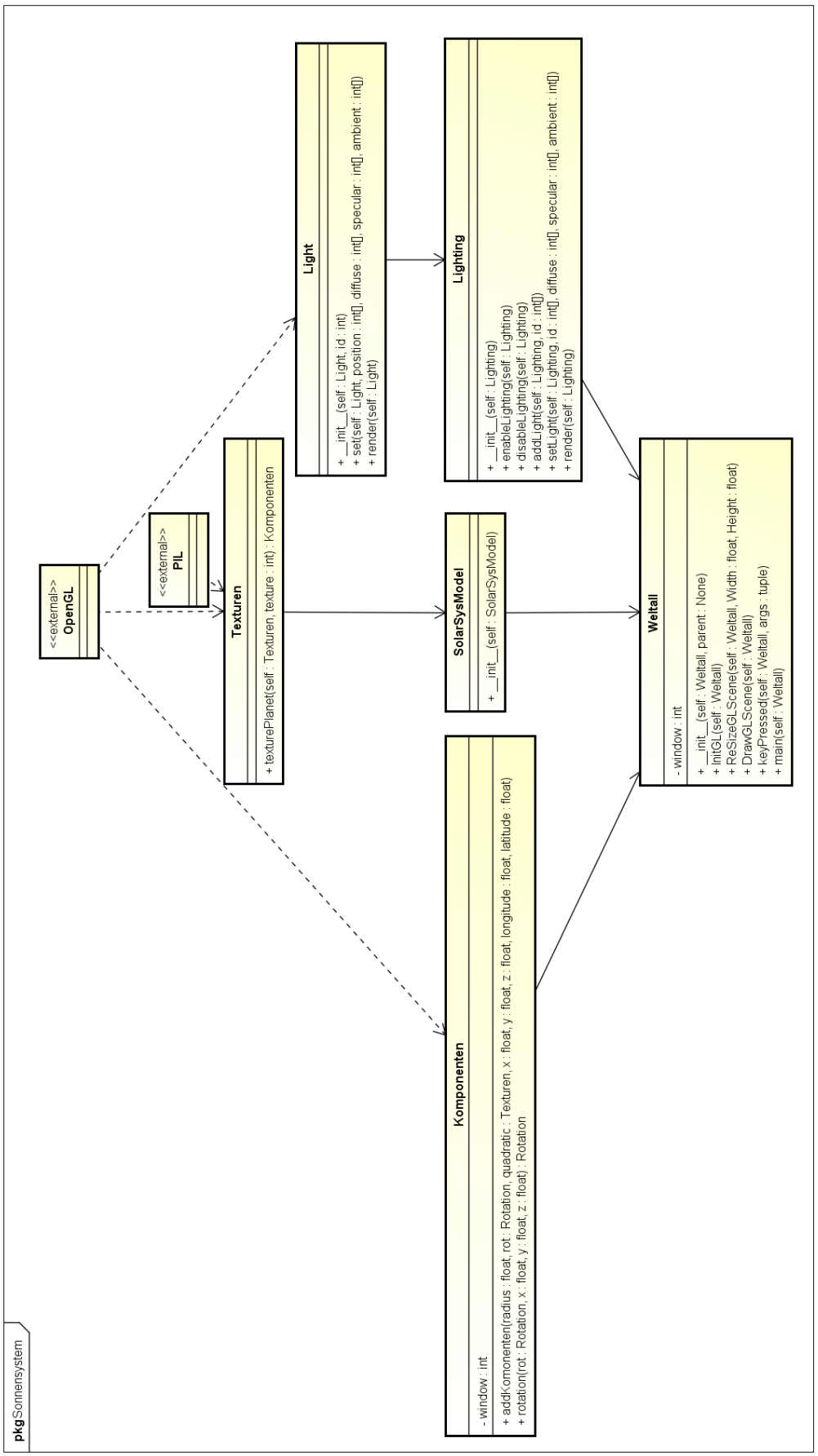
Im Bild können Sie unsere derzeitige Idee von unserem Design finden.

Bevor der Benutzer dieses Fenster sieht, wird davor noch ein paar Sekunden ein Splashscreen eingeblendet.

Auf der rechten Seite des Bildes, kann man Informationen erfahren über den derzeitigen Aufenthaltsortes eines Planeten und man kann zudem auch das Licht oder Texturen ein beziehungsweise ausblenden.

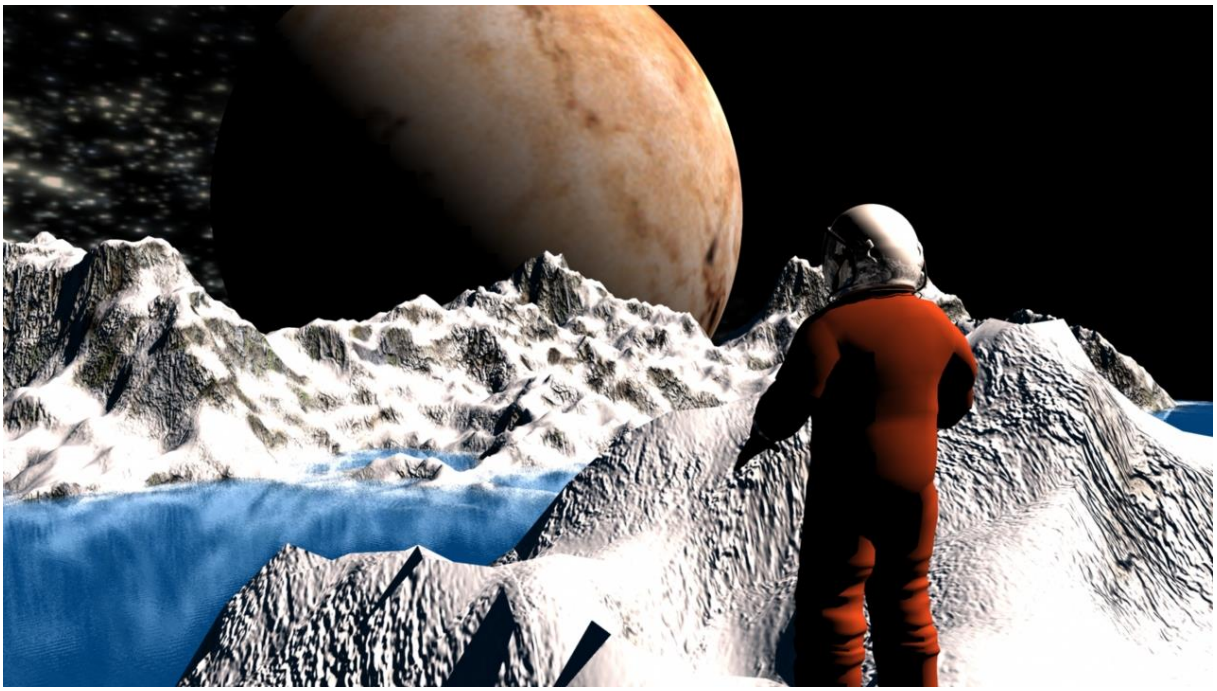
Unten befindet sich ein „Help“ Button. Wird dieser gedrückt, öffnet sich ein weiteres Fenster bei dem die Benutzerinteraktion mit Tastatur oder Maus erklärt wird.

UML – Klassendiagramm



Splashscreen

Hier ist ein Bild von dem von uns erstellten Splashscreen für die Animation des Sonnensystem



Sonnensystem

Informationen über das Sonnensystem

Name	**Durchmesser**	**Fluchtgeschwindigkeit**
Sonne	1 390 000 km	
Merkur	4 900 km	3,70 km/s
Venus	12 100 km	8,87 km/s
Erde	12 800 km	9,77 km/s
Mars	6 800 km	3,69 km/s
Jupiter	143 000 km	23 km/s
Saturn	120 500 km	8,8 km/s
Uranus	51 100 km	8,6 km/s
Neptun	49 500 km	11 km/s

Lichtpunkt setzen

Texturen auf Objekte legen

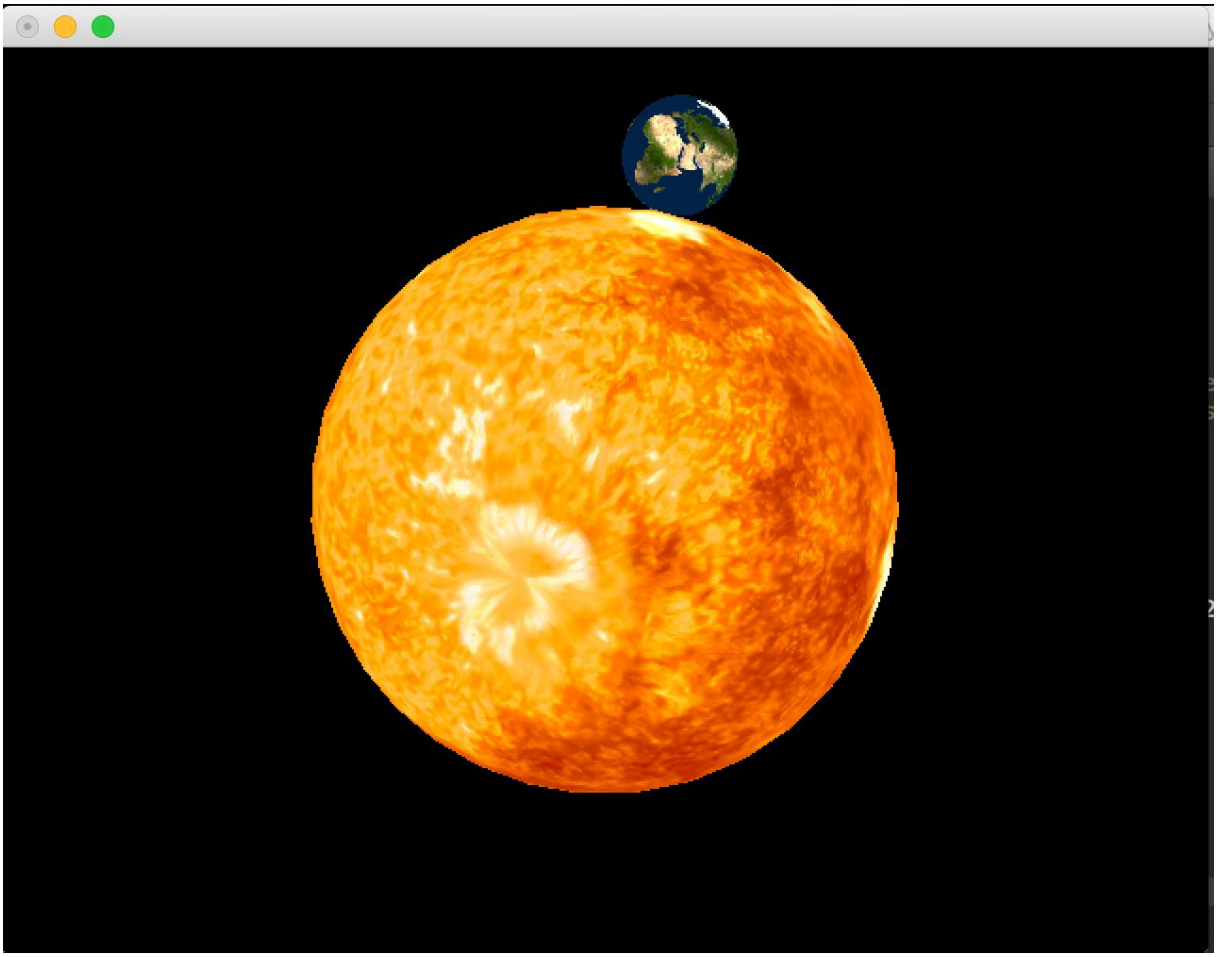
Sich um andere Objekte drehen

Implementierung

V 0.1 am 02.03.2015

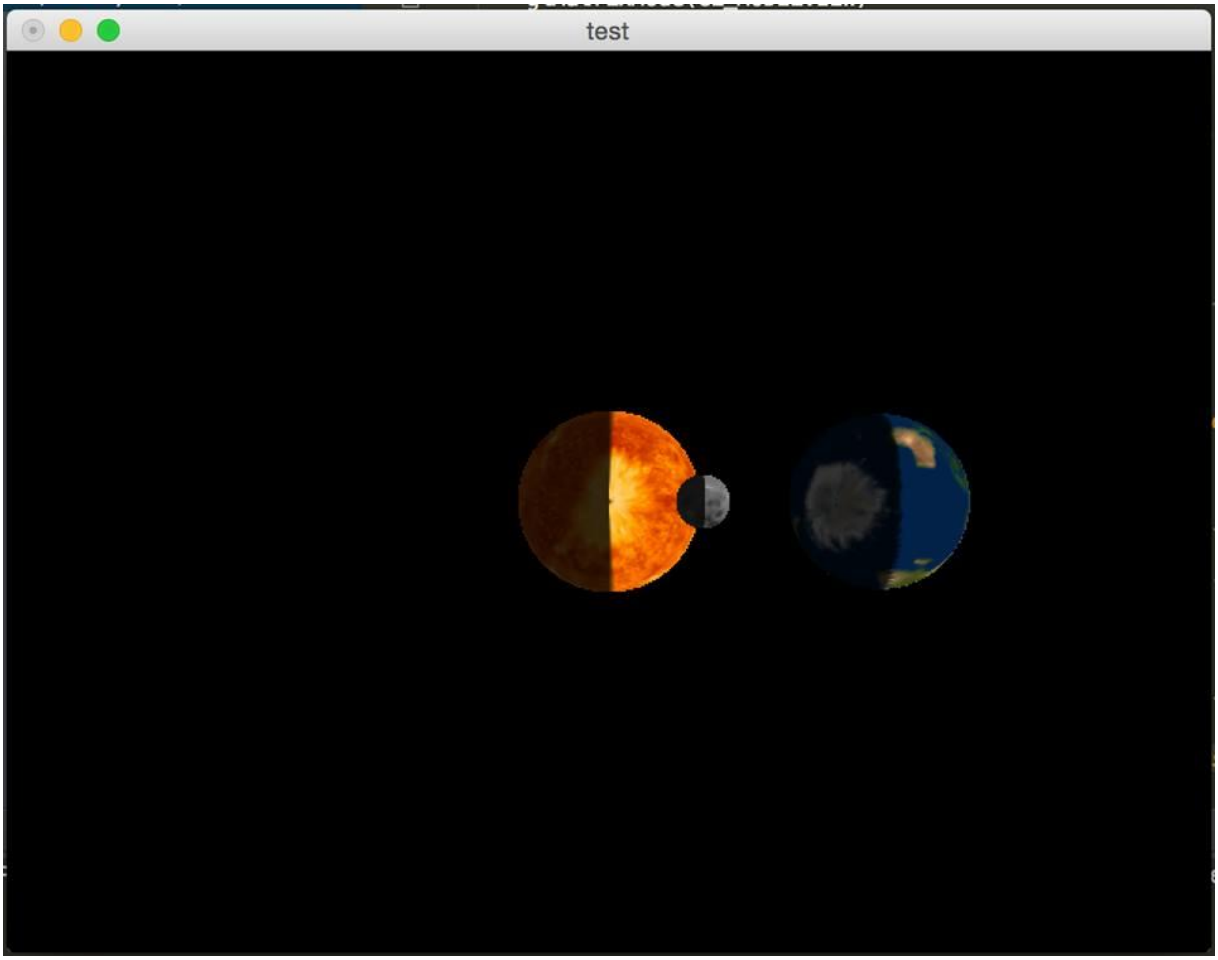
Wie verwenden für die Implementierung pyopengl, pygame sowie pillow.
Die Steuerung des Benutzers wir mittels pygame realisiert, die Erstellung von Objekten, licht usw. mit pyopengl und das importieren für Texturen verwenden wir pillow.
Wobei man sagen kann, dass die Library von pyopengl sehr viel

Damit wir beide den aktuellen Stand haben verwenden wir ein privates Repository auf Github das von Herrn Rathbauer erstellt worden ist.



V0.4 am 09.03.2015

Es wurden Licht und Geschwindigkeit mit Benutzersteuerung implementiert.



V0.5 am 16.03.2015

Implementiert:

- Ein/Ausschalten von Texturen
- Ein/Ausschlaten von Lichtern
- Splashscreen implementiert
- Usersteuerung implementiert
- Fullscreen implementierung

ToDo:
Implementierung von Steuerbare Kammera
Implementierung von Monde
Steuerungserklärung

Probleme

Derzeit sind noch keine großen Probleme aufgetreten, außer dass das Licht nicht so wie gewollt funktioniert hatte.
Umsteigen auf Python 3.4, weil Version 3.3 nicht mit der neuesten Version von PyQt (5) nicht kompatible ist.

Testfälle

Entwicklungsumgebung - Tools

Pycharm

“PyQt is a set of Python v2 and v3 bindings for Digia's Qt application framework and runs on all platforms supported by Qt including Windows, MacOS/X and Linux. PyQt5 supports Qt v5. PyQt4 supports Qt v4 and will build against Qt v5. The bindings are implemented as a set of Python modules and contain over 620 classes.

Digia have announced that support for Qt v4 will cease at the end of 2015. PyQt5 and Qt v5 are strongly recommended for all new development.

PyQt is dual licensed on all supported platforms under the GNU GPL v3 and the Riverbank Commercial License. Unlike Qt, PyQt is not available under the LGPL. You can purchase the commercial version of PyQt here. More information about licensing can be found in the License FAQ.

PyQt does not include a copy of Qt. You must obtain a correctly licensed copy of Qt yourself. However, a binary Windows installers of the GPL version of both PyQt5 and PyQt4 are provided and this includes a copy of the LGPL version of Qt.”[1]

PyQt Designer

“Qt Designer is the Qt tool for designing and building graphical user interfaces. It allows you to design widgets, dialogs or complete main windows using on-screen forms and a simple drag-and-drop interface. It has the ability to preview your designs to ensure they work as you intended, and to allow you to prototype them with your users, before you have to write any code.”[3]

Python OpenGL

“PyOpenGL is the most common cross platform Python binding to OpenGL and related APIs. The binding is created using the standard ctypes library, and is provided under an extremely liberal BSD-style Open-Source license.”[4]

Code Snippets

Pillow

Wird zum laden von Texturen verwendet.

Code Snippets

```
image = open("./texture_moon.png")

ix = image.size[0]
iy = image.size[1]
image = image.convert("RGBA").tostring("raw", "RGBA")

textures = glGenTextures(2)
glBindTexture(GL_TEXTURE_2D, int(textures[0]))  # 2d texture (x and y size)

glBindTexture(GL_TEXTURE_2D, int(textures[0]))
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)
```

```
planet = gluNewQuadric()
gluQuadricNormals(planet, GLU_SMOOTH)  # Create Smooth Normals (NEW)
gluQuadricTexture(planet, GL_TRUE)  # Create Texture Coords (NEW)

return Planet
```

Quellen

[1] Pycharm : https://www.jetbrains.com/pycharm/ gesehen am: 23.02.2015

[2] PyQT Designer: http://pyqt.sourceforge.net/Docs/PyQt4/designer.html gesehen am: 23.02.2015

[3] PythonOpenGL: http://pyopengl.sourceforge.net/ gesehen am: 23.02.2015