Visit our solar system

Daniel Bracher, Martin Suschny

Inhaltsverzeichnis

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.

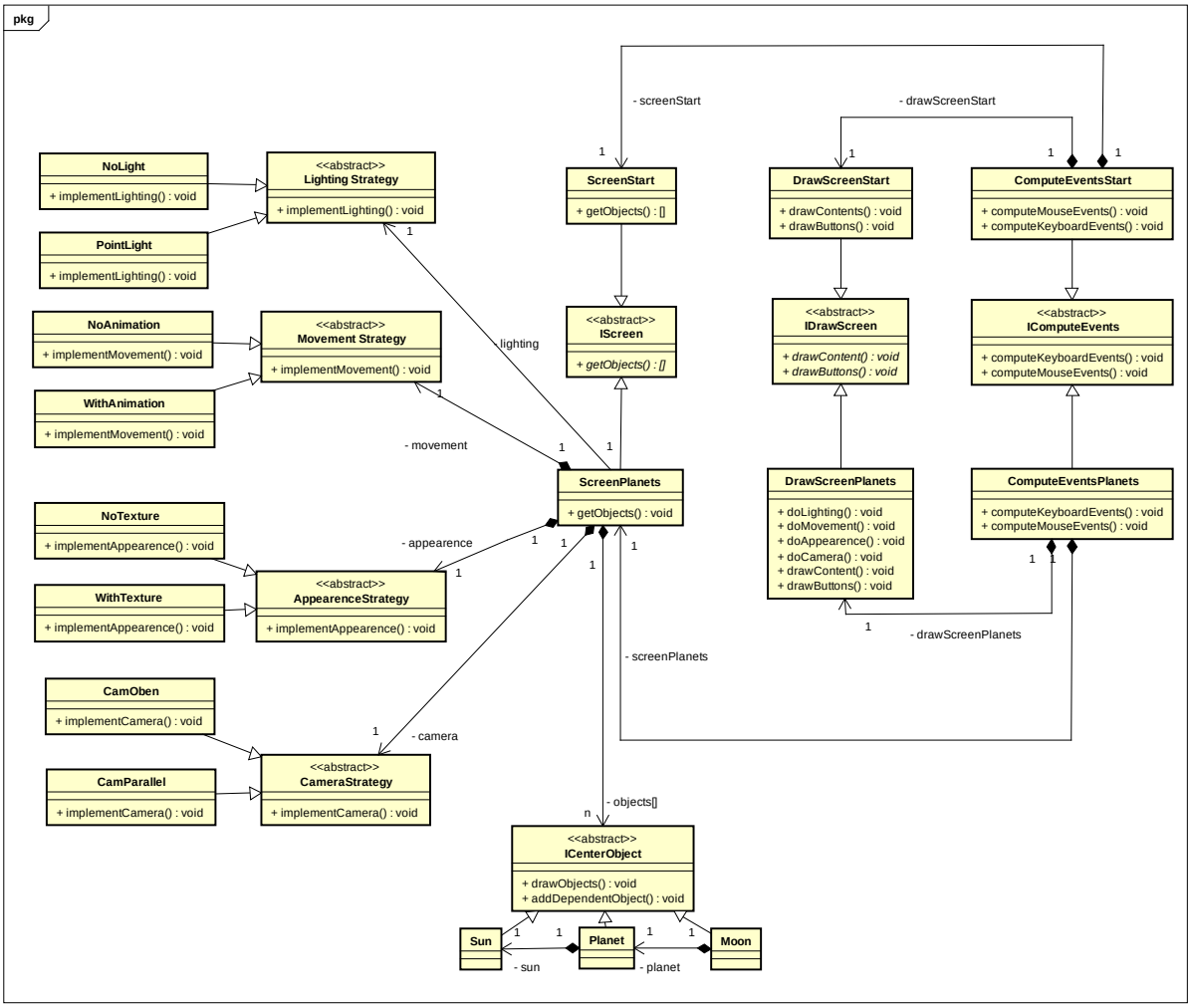
2 Zeitaufzeichnung

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**	**120**					**x**
Anlernen der Libaries	HIGH	DB/MS	120	120	**120**	**90**					**x**
2) Designüberlegungen											
SW-Design	HIGH	DB		180		**90**					**x**
GUI-Design	HIGH	MS	60		**30**						**x**
UAT Überlegung (UserAcceptanceTest)	HIGH	DB/MS	120	120	**120**						**x**
3) Implementierung											
3.1) OpenGL											
Objekte erstellen	HIGH	DB		150		**60**					**x**
Objekte mit Texturen belegen	MEDIUM	DB		240		**180**					**x**
Lichtquelle erstellen	MEDIUM	MS	180					**x**			
Schattenberechnung implementieren	LOW	MS	180				**x**				

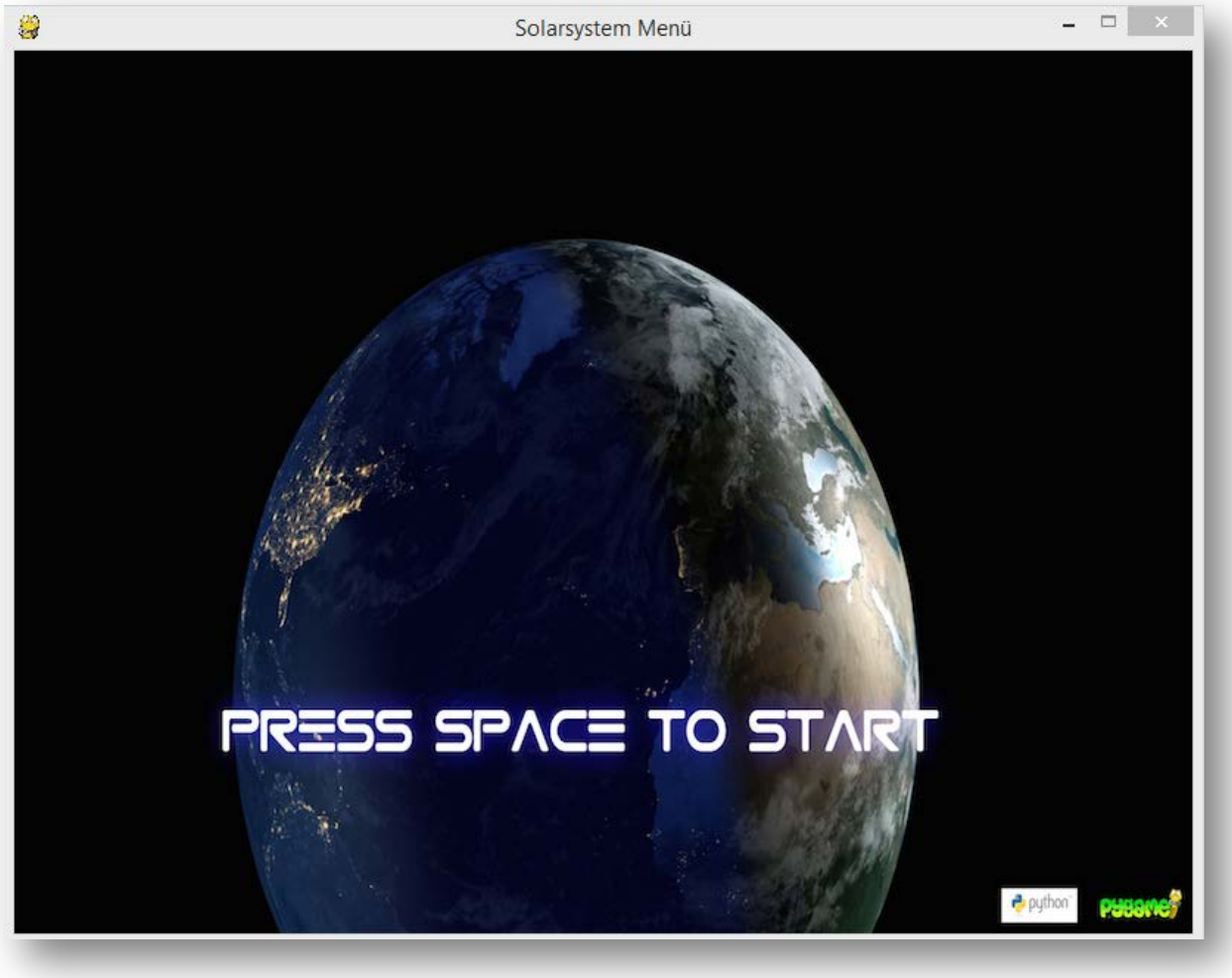
	Kamera(s) erstellen	LOW	DB		180		**60**					**x**
	Animation implementieren	HIGH	DB		300		**60**					**x**
3.2) Start-Menü												
	Splashscreen erstellen	HIGH	MS	360		**300**						**x**
3.3) Eventkoordinierung												
	Kameraposition umschalten	MEDIUM	DB		180		**60**					**x**
	Animation starten/stoppen	MEDIUM	DB		180		**60**					**x**
	Geschwindigkeit beschleunigen/starten	LOW	DB		180		**60**					**x**
	Licht an-/ausschalten	MEDIUM	MS	180					**x**			
	Spiel starten	HIGH	MS	180		**120**						**x**
5) Dokumentation												
	- Sphinx-Doku	MEDIUM	DB/MS	60	60		**120**					
	- Projektprotokoll	HIGH	DB/MS	60	60	**180**	**120**					
				25,5	33	16	18					
			Gesamt (h):	**58,5**		**34**						
				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()
```

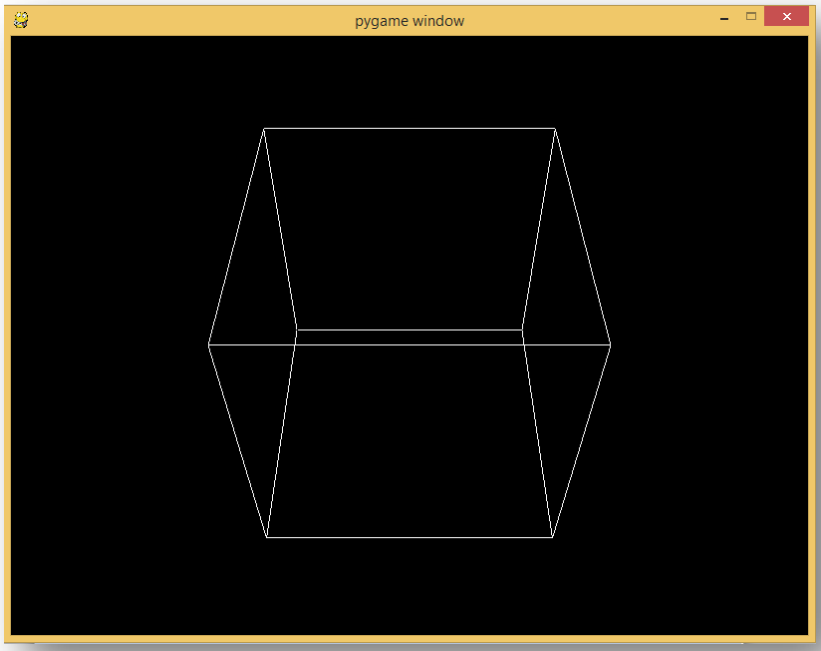
4 Arbeitsvorgang

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)



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

11.3.2015 -> BIG RELEASE

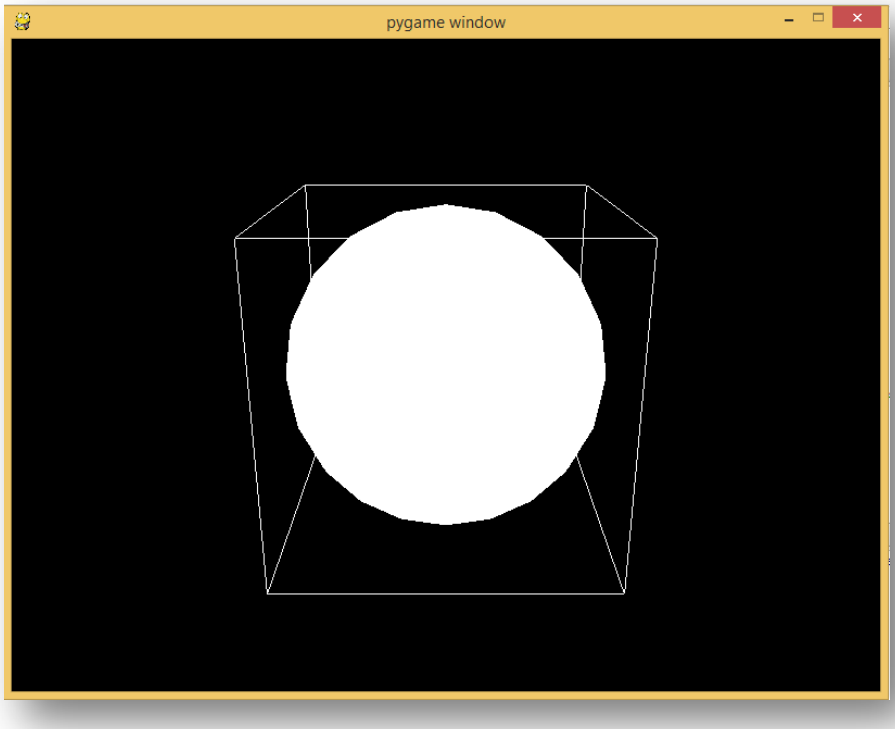
Mehrstufiges Abfragen der Events implementiert

```
def computeKeyboardEvents(self, event):
    """
    Verarbeitet die Keyboard-Events

    :return: Nichts
    """
    #if event.key == .K_LEFT:


def computeMouseEvents(self, event):
    """
```

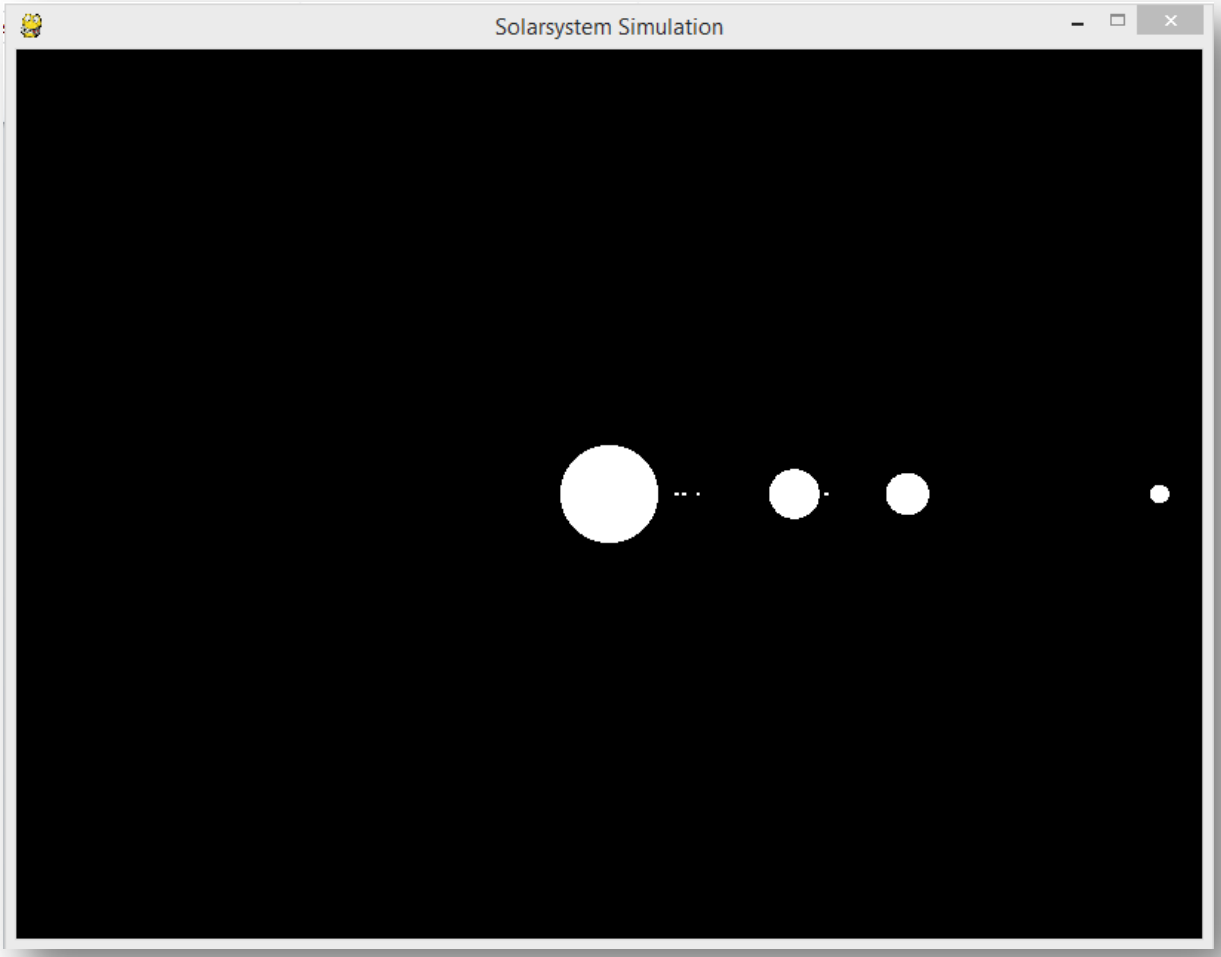
```
    Verarbeitet die Mouse-Events

    :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

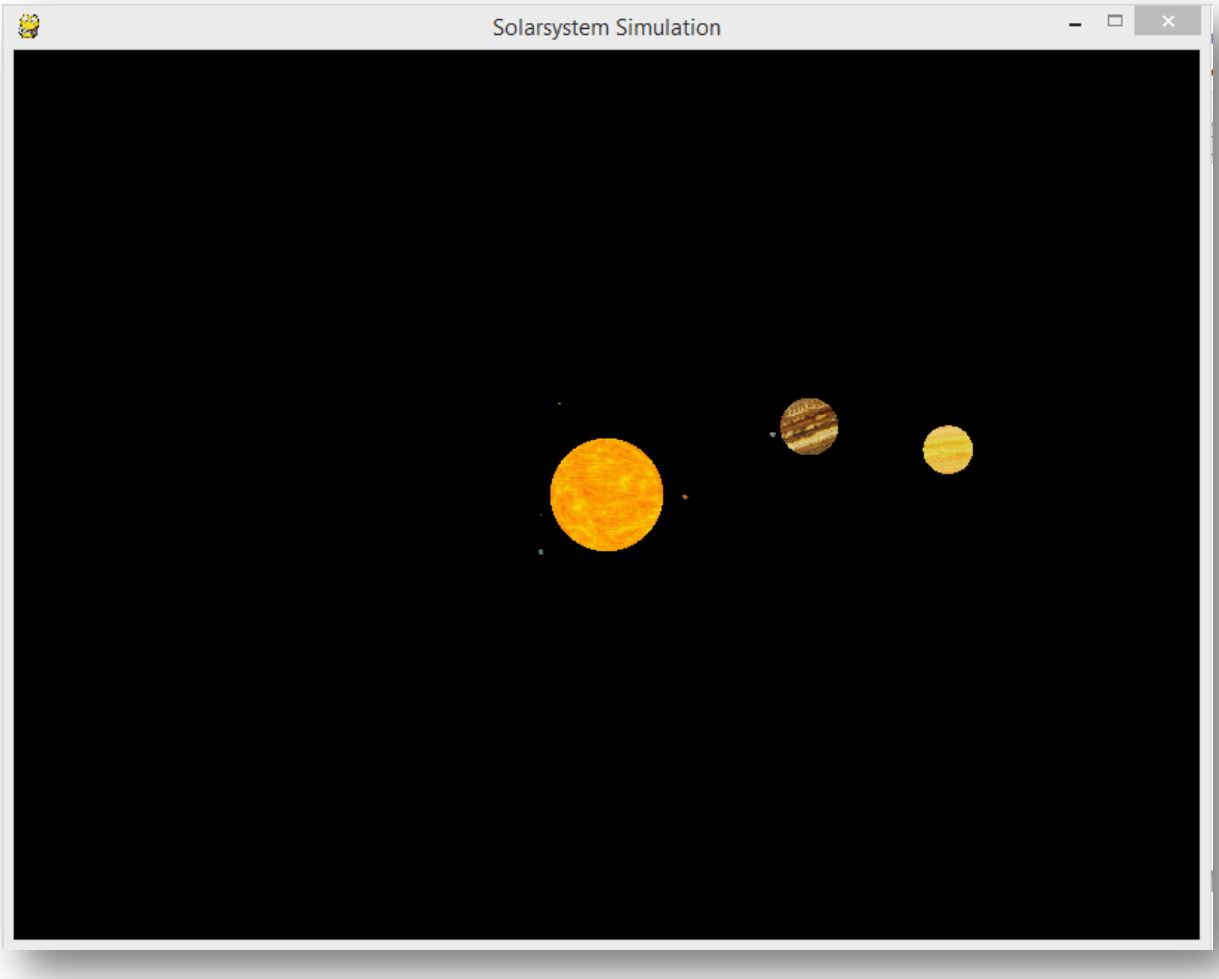
```
def changeMovement(self):
    #Falls Movement derzeit NoAnimation-Strategie implementiert
    if isinstance(self. movement, NoAnimation):
     #Auf WithAnimation-Strategie aendern
     self.movement = WithAnimation()
    #Falls WithAnimation-Strategie implementiert
    elif isinstance(self.movement, WithAnimation):
     #Auf NoAnimation-Strategie aendern
     self.movement = NoAnimation()
    Sonne als Kugel implementiert
```



20.3.2015 -> Objekte bewegen sich
21.3.2015 -> Planeten drehen sich um Sonne
22.3.2015 -> Monde drehen sich um Planeten



27.3.2015 -> Alle Objekte haben Texturen
28.3.2015 -> Beschleunigung/Bremsung möglich
Kamerapositionen lassen sich ändern
29.3.2015 -> Texturen lassen sich an- / ausschalten


Solarsystem Simulation

5 Testdokumentation

5.1 User Acceptance Tests

Test Fall 1

Test Fall Nr.: 1

Test Priorität (Niedrig/Mittel/Hoch): Hoch

Modul Name: StartScreen.py

Test Titel: Starten der Applikation

Beschreibung: Die Applikation soll über das Modul StartScreen.py gestartet werden können.

Test erstellt von: Martin Suschny

Test erstellt am: 17.03.2015

Test durchgeführt von: Martin Suschny

Test durchgeführt am: 24.03.2015

Voraussetzung:
Alle notwendigen Interpreter und Bibliotheken müssen auf dem System installiert sein.

Schritt	Test Schritte	Test Daten	Erwartetes Ergebnis	Tatsächliches Ergebnis	Status (x / ✓)	Anmerkungen
1	Ausführen der StartScreen.py Datei	-	Ein Fenster, welches das Hauptmenü beinhaltet muss auf dem Bildschirm angezeigt werden.	Ein Fenster, welches das Hauptmenü beinhaltet wird auf dem Bildschirm angezeigt werden.	✓	

Nachbedinungen:
Die Applikation muss ein Fenster mit Hauptmenü anzeigen.

Test Fall 2

Test Fall Nr.: 2

Test Priorität (Niedrig/Mittel/Hoch): Hoch

Modul Name: StartScreen.py

Test Titel: Anzeigen des Splashscreens

Beschreibung: Die Applikation soll im Hauptmenü einen Splashscreen anzeigen.

Test erstellt von: Martin Suschny

Test erstellt am: 17.03.2015

Test durchgeführt von: Martin Suschny

Test durchgeführt am: 28.03.2015

Voraussetzung:

Alle notwendigen Interpreter und Bibliotheken müssen auf dem System installiert sein. Die Applikation muss starten können.

Schritt	Test Schritte	Test Daten	Erwartetes Ergebnis	Tatsächliches Ergebnis	Status (x / ✓)	Anmerkungen
1	Ausführen der StartScreen .py Datei	Splashscreen	Ein Hauptmenü mit entsprechendem Splashcreen	Das Hauptmenü wird mit dem Splashscreen angeziegt	✓	

Nachbedinungen:

Die Applikation muss sich im Hauptmenü befinden und den Splashscreen anzeigen.

Test Fall 3

Test Fall Nr.: 3

Test Priorität (Niedrig/Mittel/Hoch): Hoch

Modul Name: StartScreen.py

Test Titel: Anzeigen der Simulation

Beschreibung: Die Simulation soll mittels Leertaste gestartet werden können.

Test erstellt von: Martin Suschny

Test erstellt am: 17.03.2015

Test durchgeführt von: Martin Suschny

Test durchgeführt am: 25.03.2015

Voraussetzung:

Die Applikation muss sich im Hauptmenü befinden.

Schritt	Test Schritte	Test Daten	Erwartetes Ergebnis	Tatsächliches Ergebnis	Status (x / ✓)	Anmerkungen
1	Drücken der Leertaste	-	Wechsel von Hauptmenü in Simulation	Fenster der Simulation wurde gestartet	✓	

Nachbedinungen:

Die Applikation muss sich in der Simulation befinden und diese anzeigen.

Test Fall 4

Test Fall Nr.: 4

Test Priorität (Niedrig/Mittel/Hoch): Hoch

Modul Name: StartScreen.py

Test Titel: Schließen der Anwendung1

Beschreibung: Die Anwendung muss aus dem Hauptmenü mit den Betriebssystem üblichen Methoden geschlossen werden können.

Test erstellt von: Martin Suschny

Test erstellt am: 17.03.2015

Test durchgeführt von: Martin Suschny

Test durchgeführt am: 25.03.2015

Voraussetzung:
Die Applikation muss sich im Hauptmenü befinden.

Schritt	Test Schritte	Test Daten	Erwartetes Ergebnis	Tatsächliches Ergebnis	Status (x / ✓)	Anmerkungen
1	Schließen durch Betriebssystem übliche Methoden	-	Geschlossener Prozess	Geschlossener Prozess	✓	

Nachbedinungen:
Die Applikation muss geschlossen sein.

Test Fall 5

Test Fall Nr.: 5

Test Priorität (Niedrig/Mittel/Hoch): Hoch

Modul Name: PlantenScreen.py

Test Titel: Besitzen eines zentralen Sternes

Beschreibung: Die Simulation muss einen Zentralen Stern besitzen um den sich die Planten drehen.

Test erstellt von: Martin Suschny

Test erstellt am: 17.03.2015

Test durchgeführt von: Martin Suschny

Test durchgeführt am: 27.03.2015

Voraussetzung:

Die Applikation muss sich in der Simulation befinden.

Schritt	Test Schritte	Test Daten	Erwartetes Ergebnis	Tatsächliches Ergebnis	Status (x / ✓)	Anmerkungen
1	Überprüfen ob die Simulation einen zentralen Stern besitzt	-	Ein Zentraler Stern	Ein Zentraler Stern	✓	

Nachbedinungen:

Die Simulation muss einen Zentralen Stern anzeigen.

Test Fall 6

Test Fall Nr.: 6

Test Priorität (Niedrig/Mittel/Hoch): Hoch

Modul Name: PlantenScreen.py

Test Titel: Besitzen von Planeten

Beschreibung: Die Simulation muss Planten besitzen.

Test erstellt von: Martin Suschny

Test erstellt am: 17.03.2015

Test durchgeführt von: Martin Suschny

Test durchgeführt am: 27.03.2015

Voraussetzung:

Die Applikation muss sich in der Simulation befinden.

Schritt	Test Schritte	Test Daten	Erwartetes Ergebnis	Tatsächliches Ergebnis	Status (x / ✓)	Anmerkungen
1	Überprüfen ob die Simulation Planten besitzt	-	Planten, welche angezeigt werden	Planten, welche angezeigt werden	✓	Die Simulation besitzt 9 Planeten

Nachbedinungen:

Die Simulation muss mindestens 2 Planeten anzeigen.

Test Fall 7

Test Fall Nr.: 7

Test Priorität (Niedrig/Mittel/Hoch): Hoch

Modul Name: PlantenScreen.py

Test Titel: Besitzen von Monden

Beschreibung: Die Simulation muss Monde besitzen.

Test erstellt von: Martin Suschny

Test erstellt am: 17.03.2015

Test durchgeführt von: Martin Suschny

Test durchgeführt am: 29.3.2015

Voraussetzung:

Die Applikation muss sich in der Simulation befinden.

Schritt	Test Schritte	Test Daten	Erwartetes Ergebnis	Tatsächliches Ergebnis	Status (x / ✓)	Anmerkungen
1	Überprüfen ob die Simulation Monde besitzt	-	Ein Planet angezeigt mit einem Mond	Erde und Jupiter haben einen Mond	✓	

Nachbedinungen:

Mindestens ein Planet muss einen und oder mehre Monde haben.

Test Fall 8

Test Fall Nr.: 8

Test Priorität (Niedrig/Mittel/Hoch): Mittel

Modul Name: PlantenScreen.py

Test Titel: Wechseln der Perspektive

Beschreibung: Die Perspektive der Simulation muss gewechselt werden können.

Test erstellt von: Martin Suschny

Test erstellt am: 17.03.2015

Test durchgeführt von: Martin Suschny

Test durchgeführt am: 27.03.2015

Voraussetzung:
Die Applikation muss sich in der Simulation befinden.

Schritt	Test Schritte	Test Daten	Erwartetes Ergebnis	Tatsächliches Ergebnis	Status (x / ✓)	Anmerkungen
1	Drücken der Taste „c“	-	Wechsel der Perspektive	Wechsel der Perspektive	✓	Die Simulation kann in 3 verschiedene Perspektiven wechseln(Parallel, von Oben und eine weitere, leicht schräge)
2	Drücken der Taste „c“	-	Wechsel der Perspektive	Wechsel der Perspektive	✓	
3	Drücken der Taste „c“	-	Wechsel der Perspektive	Wechsel der Perspektive	✓	

Nachbedinungen:
Die Simulation muss die Perspektive gewechselt haben und wieder in die ursprüngliche Perspektive wechseln können.

Test Fall 9

Test Fall Nr.: 9

Test Priorität (Niedrig/Mittel/Hoch): Mittel

Modul Name: PlantenScreen.py

Test Titel: Starten/Stopen der Animation

Beschreibung: Die Animation der Simulation muss mittels Maus gestartet und gestoppt werden können.

Test erstellt von: Martin Suschny

Test erstellt am: 17.03.2015

Test durchgeführt von: Martin Suschny

Test durchgeführt am: 27.03.2015

Voraussetzung:

Die Applikation muss sich in der Simulation befinden.

Schritt	Test Schritte	Test Daten	Erwartetes Ergebnis	Tatsächliches Ergebnis	Status (x / ✓)	Anmerkungen
1	Drücken der linken Maustaste	-	Starten der Animation	Starten der Animation	✓	
2	Nochmaliges Drücken der Maustaste	-	Stoppen der Animation	Stoppen der Animation	✓	

Nachbedinungen:

Die Simulation muss die Animation gestartet haben und wieder stoppen können.

Test Fall 10

Test Fall Nr.: 10

Test Priorität (Niedrig/Mittel/Hoch): Mittel

Modul Name: PlantenScreen.py

Test Titel: Bewegung der Planeten

Beschreibung: In der Animationen müssen sich Planeten um ihre eigene Achse und um die Bahnen des Zentralsterns bewegen.

Test erstellt von: Martin Suschny

Test erstellt am: 17.03.2015

Test durchgeführt von: Martin Suschny

Test durchgeführt am: 27.03.2015

Voraussetzung:
Die Applikation muss sich in der Simulation befinden. Die Animation muss gestartet sein.

Schritt	Test Schritte	Test Daten	Erwartetes Ergebnis	Tatsächliches Ergebnis	Status (x / ✓)	Anmerkungen
1	Überprüfen ob sich Planeten um ihre eigne Achse drehen	-	Drehung der Planten um die eigene Achse	Drehung der Planten um die eigene Achse	✓	
2	Überprüfen ob sich Planten um den Zentralstern drehen	-	Drehung der Planeten um den Zentralstern	Drehung der Planeten um den Zentralstern	✓	

Nachbedinungen:
Die Simulation muss die Animation gestartet haben und die oben angeben Drehungen durchführen.

Test Fall 11

Test Fall Nr.: 11

Test Priorität (Niedrig/Mittel/Hoch): Mittel

Modul Name: PlantenScreen.py

Test Titel: Bewegung der Monde

Test erstellt von: Martin Suschny

Test erstellt am: 17.03.2015

Test durchgeführt von: Martin Suschny

Test durchgeführt am: 29.03.2015

Beschreibung: In der Animationen müssen sich Monde um die Achse deren Planten drehen.

Voraussetzung:
Die Applikation muss sich in der Simulation befinden. Die Animation muss gestartet sein.

Schritt	Test Schritte	Test Daten	Erwartetes Ergebnis	Tatsächliches Ergebnis	Status (x / ✓)	Anmerkungen
1	Überprüfen ob sich Monde um die Achse ihres Planten drehen	-	Drehung um den eignen Planten	Drehung um den eignen Planten	✓	

Nachbedinungen:
Die Simulation muss die Animation gestartet haben und die oben angeben Drehungen durchführen.

Test Fall 12

Test Fall Nr.: 12

Test erstellt von: Martin Suschny

Test Priorität (Niedrig/Mittel/Hoch): Niedrig

Test erstellt am: 17.03.2015

Modul Name: PlantenScreen.py

Test durchgeführt von: Martin Suschny

Test Titel: Steuern der Animationsgeschwindugkeit

Test durchgeführt am: 27.03.2015

Beschreibung: Die Animationsgeschwindigkeit der Simulation muss mittels Pfeiltasten gesteuert werden können.

Voraussetzung:

Die Applikation muss sich in der Simulation befinden. Die Animation muss gestartet sein.

Schritt	Test Schritte	Test Daten	Erwartetes Ergebnis	Tatsächliches Ergebnis	Status (x / ✓)	Anmerkungen
1	Mehrmaliges drücken der rechten Pfeiltaste	-	Animation wird schneller	Animation wird schneller	✓	
2	Mehrmaliges drücken der linken Pfeiltaste	-	Animation wird langsamer	Animation wird langsamer	✓	

Nachbedinungen:

Die Simulation muss die Animation gestartet haben und sowohl vorwärts als auch rückwärts ihre Animationsgeschwindigkeit verändert haben.

Test Fall 13

Test Fall Nr.: 13

Test erstellt von: Martin Suschny

Test Priorität (Niedrig/Mittel/Hoch): Niedrig

Test erstellt am: 17.03.2015

Modul Name: PlantenScreen.py

Test durchgeführt von: Martin Suschny

Test Titel: Zoomen der Ansicht

Test durchgeführt am: 27.03.2015

Beschreibung: Die Nähe/Ferne der Perspektive muss mittels Mausrad gesteuert werden können.

Voraussetzung:
Die Applikation muss sich in der Simulation befinden.

Schritt	Test Schritte	Test Daten	Erwartetes Ergebnis	Tatsächliches Ergebnis	Status (x / ✓)	Anmerkungen
1	Bewegen des Mausrads nach vorne	-	Zoom nach Außen	Zoom nach Außen	✓	Nicht unendlich möglich
2	Bewegen des Mausrads nach hinten	-	Zoom nach Innen	Zoom nach Innen	✓	Nicht unendlich möglich

Nachbedinungen:
Die Simulation muss die Perspektive ran gezoomt und fern gezoomt haben.

Test Fall 14

Test Fall Nr.: 14

Test erstellt von: Martin Suschny

Test Priorität (Niedrig/Mittel/Hoch): Mittel

Test erstellt am: 17.03.2015

Modul Name: PlantenScreen.py

Test durchgeführt von: Martin Suschny

Test Titel: Besitzen von Texturen

Test durchgeführt am: 29.03.2015

Beschreibung: Planten, der Zentralstern und Monde müssen mit einer Textur behaftet sein.

Voraussetzung:

Die Applikation muss sich in der Simulation befinden.

Schritt	Test Schritte	Test Daten	Erwartetes Ergebnis	Tatsächliches Ergebnis	Status (x / ✓)	Anmerkungen
1	Überprüfen ob die Objekte Texturen besitzen	-	Alle Objekte sind texturiert	Alle Objekte sind texturiert	✓	

Nachbedinungen:

Alle Objekte in der Simulation müssen texturiert sein.

Test Fall 15

Test Fall Nr.: 15

Test erstellt von: Martin Suschny

Test Priorität (Niedrig/Mittel/Hoch): Mittel

Test erstellt am: 17.03.2015

Modul Name: PlantenScreen.py

Test durchgeführt von: Martin Suschny

Test Titel: Aktivierung der Beleuchtung

Test durchgeführt am: 29.03.2015

Beschreibung: Mittels rechter Maustaste soll die Simulation beleuchtet werden.

Voraussetzung:

Die Applikation muss sich in der Simulation befinden.

Schritt	Test Schritte	Test Daten	Erwartetes Ergebnis	Tatsächliches Ergebnis	Status (x / ✓)	Anmerkungen
1	Drücken der rechten Maustaste	-	Beleuchtung ist aktiv	Keine Beleuchtung	**x**	
2	Erneutes Drücken der rechten Maustaste	-	Beleuchtung ist wieder deaktiviert		**x**	

Nachbedinungen:

Die Simulation muss beleuchtet sein und die Beleuchtung muss auch wieder deaktiviert werden können.

Test Fall 16

Test Fall Nr.: 16

Test Priorität (Niedrig/Mittel/Hoch): Niedrig

Modul Name: PlantenScreen.py

Test Titel: Schatten werfen

Beschreibung: Planten und Monde müssen Schatten werfen können.

Test erstellt von: Martin Suschny

Test erstellt am: 17.03.2015

Test durchgeführt von: Martin Suschny

Test durchgeführt am: 29.03.2015

Voraussetzung:
Die Applikation muss sich in der Simulation befinden und diese muss beleuchtet sein.

Schritt	Test Schritte	Test Daten	Erwartetes Ergebnis	Tatsächliches Ergebnis	Status (x / ✓)	Anmerkungen
1	Überprüfen ob Schatten geworfen werden	-	Schatten von allen Planten und Monden werden geworfen	Keine Schatten	**x**	

Nachbedinungen:
Die Simulation muss beleuchtet sein und alle Objekte abgesehen vom Zentralstern müssen Schatten werfen.

Test Fall 17

Test Fall Nr.: 17

Test Priorität (Niedrig/Mittel/Hoch): Hoch

Modul Name:Planeten Screen.py

Test Titel: Schließen der Anwendung2

Beschreibung: Die Anwendung muss aus der Simulation mit den Betriebssystem üblichen Methoden geschlossen werden können.

Test erstellt von: Martin Suschny

Test erstellt am: 17.03.2015

Test durchgeführt von: Martin Suschny

Test durchgeführt am: 25.03.2015

Voraussetzung:

Die Applikation muss sich in der Simulation befinden.

Schritt	Test Schritte	Test Daten	Erwartetes Ergebnis	Tatsächliches Ergebnis	Status (x / ✓)	Anmerkungen
1	Schließen durch Betriebssystem übliche Methoden	-	Geschlossener Prozess	Geschlossener Prozess	✓	

Nachbedinungen:

Die Applikation muss geschlossen sein.

Quellen

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

[2] Pygame Documentation; http://www.pygame.org/docs/; zuletzt besucht: 27.3.2015