Python-Projekt Grafikprogrammierung 11/I

Um Ihnen einen kleinen Überblick über unser Projekt zu geben:
Wir haben ein Programm geschrieben, welches drei verschiedene Körper (Keplerstern, Ikosaeder, Dodekaeder) in 3D darstellt, die sich mit verstellbarer Mausempfindlichkeit oder Buttons um bestimmte Winkel (Grad/Bogenmaß) drehen lassen. Man kann zwischen zwei Zeichenmodi (nur Kanten oder Kanten und Flächen) wählen, mit RGB und hex-Farbcodes die Farben von Kanten und Flächen anpassen, und einen RGB-Farbverlauf mit anpassbarer Geschwindigkeit erzeugen. Im Programm selbst sind einige Hilfestellungen zu finden, und die für das Auge schädliche Farbkombination schwarz-gelb wird sehr streng herausgefiltert, um das bestmögliche Nutzererlebnis zu ermöglichen. Hier finden Sie ein kurzes Vorschaubild zu unserem Programm:

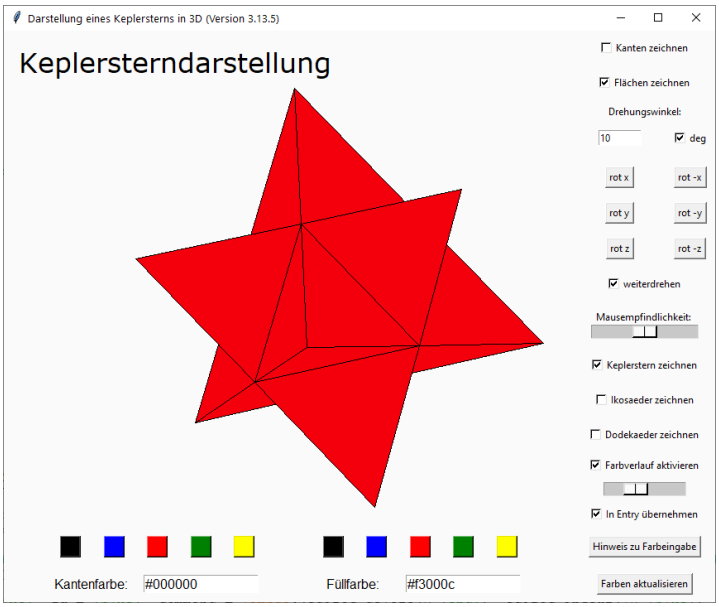

Abbildung 1: Vorschaubild des Programms

Dieser Ordner enthält allerdings lediglich einen Bruchteil unserer Produkte. Da unser Projekt sehr umfangreich geworden ist, empfehlen wir Ihnen, sich unser GitHub repository einmal anzuschauen:

https://github.com/tscholz26/python-project

Dort finden Sie auch eine weitere readme-Datei, die Ihnen die Ordnerstruktur dort erklärt. Um unser Programm übersichtlich und verständlich für andere zu gestalten, haben wir das OpenSource-Dokumentationstool „Doxygen“ verwendet. Wenn man Codeblocks in den Quelltext einfügt, die bestimmte Methoden erklären, kann man mit Doxygen automatisch eine vollständige Dokumentation für das gesamte Projekt erstellen. Diese ist in GitHub zu finden, wir haben sie Ihnen aber auch in diesem Ordner als .html-file zur Verfügung gestellt. Auf der nächsten Seite finden sie ein Vorschaubild, wie dieses Dokument aussieht. Die Dokumentation wirkt natürlich noch etwas leer, da Doxygen sonst eher für professionellere Programme verwendet wird.
Wir wünschen Ihnen viel Spaß mit unserem Ergebnis

- Luisa Lindner, Fabien Streuber, Tristan Scholz

Python Projekt Grafik

In diesem Dokument finden Sie eine Dokumentation zum Projekt der Gruppe 4, bestehend aus Luisa Lindner, Fabien Streuber und Tristan Scholz, die die einzelnen Methoden und eine Übersicht über unsere Klasse ermöglichen soll. Unter Classes -> Class List -> Object3d sollte dies am übersichtlichsten sein.

Main Page | Namespaces | Classes | Search

latest | Object3d

Public Member Functions | List of all members

latest.Object3d Class Reference

Inheritance diagram for latest.Object3d:

Public Member Functions

def __init__ (self, name=str, xyz=[], xy=[], sl=float, ek=int, a=int)
def name (self)
def initkep (self)
def initiko (self)
def initdod (self)
def conv32 (self)
def getcolor (self, string)
def initgradient (self)
def initfuncgradient (self)
def gradientgetcolor (self)
def zsort (self)
def resetspeeds (self)
def getdeg (self)
def setspeeds (self, char, speed)
def infiniterep (self)
def rotall (self)
def rotx (self, sgndeg)
def roty (self, sgndeg)
def rotz (self, sgndeg)
def avoidbvb (self)
def drawedges (self)
def drawfaces (self)
def zeichnen (self)

Detailed Description

```
Diese Klasse beschreibt unsere Körper, mit Name, 3D- und 2D-Koordinaten, Seitenlänge sl, Streckungsfaktor a und der Eckenzahl der Seitenflächen ek.
```

Member Function Documentation

avoidbvb()

def latest.Object3d.avoidbvb (self)

```
Diese Methode prüft, ob der Nutzer bei der Farbauswahl aufgrund geistiger
Umnachtung die katastrophale Farbkombination schwarz-gelb gewählt hat, und
stellt eine angenehmere Farbkombination ein, da es sonst zu seelischen Schmerzen
aufgrund der schwarz-gelben Farbe kommen könnte und die Versicherungen der
Codeschreiber diese Gefahr nicht abdecken (nichtmal Fabis private Kasse).
```

conv32()

def latest.Object3d.conv32 (self)

```
Diese Methode wandelt die dreidimensionalen Koordinaten aller Punkte in
2D-Koordinaten um, die gezeichnet werden können. An den 3D-Koordinaten werden
keine Änderungen vorgenommen, da 2D- und 3D-Koordinaten in verschiedenen Arrays
enthalten sind.

returns:
    xy: list
```

drawedges()

def latest.Object3d.drawedges (self)

```
Diese Methode leert zuerst vollständig die Zeichenfläche. Danach erzeugt sie mit
der conv32() Methode die 2D-Koordinaten des "Körpers" und kombiniert nun immer 2
Punkte, solange bis alle Kombinationen probiert sind. Wenn der Abstand der zwei
Punkte der Seitenlänge obj.sl (mit einer Toleranz von 5) entspricht, werden mit der
getcolor() Methode die gewünschte Linienfarbe errechnet und die Punkte verbunden.
```

drawfaces()

def latest.Object3d.drawfaces (self)

```
Diese Methode leert zuerst vollständig die Zeichenfläche. Die 3D-Koordinaten
werden mit zsort() sortiert. Danach  werden systematisch Punkte kombiniert, und
überprüft ob der Abstand der Punkte der Seitenlänge obj.sl entspricht (Toleranz).
Wenn man so Dreiecke/Fünfecke findet, die zum Körper gehören, werden die Punkte
die dazu gehören dem Array facelist hinzugefügt. Bei Dreiecken wird geprüft, ob die
Fläche bereits enthalten ist, um Flächen nicht doppelt zu zeichnen. Bei Fünfecken
bräuchte dies Verfahren mehr Rechenleistung als das doppelt zeichnen. Danach werden
Flächenmittelpunkte berechnet und nach höchsten z-Koordinaten sortiert, um die
verdeckten Flächen zum Schluss zu malen, und es werden dann die gefundenen Flächen
in facelist gezeichnet.
```

getcolor()

def latest.Object3d.getcolor (self,
string
)

```
Diese Methode fragt je nachdem ob der String fc (face color) oder oc (Outline
Color) enthält den Inhalt des zugehörigen Entrys ab. Wenn dort ein rgb-Wert steht,
wird dieser mit der rgbtohex() Methode in hex umgewandelt. Zum Schluss wird die
Zeichenfarbe für das jeweilige Element (face/outline) zurückgegeben.

args:
    string: string, der 'fc' oder 'oc' enthält

returns:
    color: string, der Farbe enthält
```

Abbildung 2: Vorschaubild der Dokumentation, erzeugt von Doxygen