Q4 Kursarbeit Informatik

Yannick Hein

Table of Contents

Table of Figures

Index of Tables

1. Basics

1.1 Fahrzeug, Fahrrad und PKW

Ich werde den Quellcode oft nicht in einer Tabelle in das Dokument einfügen, weil die Indentierung nicht übernommen wird und der Code deswegen schlecht lesbar ist.

Quellcode siehe no1.py

1.2 Vererbung

Bei der init-Methode der Fahrrad und Pkw Klassen muss als erstes die init der Elternklasse (also Fahrzeug) mit super().__init__() aufgerufen werden.
super() ist ein shortcut um die Elternklasse anzusprechen.

Bei der str Methode habe ich der Vollständigkeit halber mit super().__str__() den String der Elternklasse zu dem String der Fahrrad und Pkw Klassen hinzugefügt.

1.3 Fahrzeugvermietung

Quellcode siehe no1.py

1.4 UML Diagramm von Fahrzeugvermietung

Table 1: UML-Diagramm Fahrzeugvermietung

Fahrzeugvermietung
myFahrzeug: Fahrzeug myFahrrad: Fahrrad myPkw: PKW
printAllVehicles(): void

2. Parser

Quellcode siehe MyFileParser.py

Table 2: Ausgabe des Parsers

<Section: DEFAULT> <Section: user> name: Herbert lastname: Smith email: herb.smith@poodle.com password: x45655554xoxoxo   <Section: DEFAULT> <Section: user> name: Herbert lastname: Smith email: neuHerbert@uwu.com password: x45655554xoxoxo

Table 3: user.ini vor und nach Ausführung des Programms

```
[user]
name = Herbert
lastname = Smith
email = herb.smith@poodle.com
password = x45655554xoxoxo
```

```
[user]
name = Herbert
lastname = Smith
email = neuHerbert@uwu.com
password = x45655554xoxoxo
```

2.1 Erweiterung des Parsers

Quellcode siehe MyFileParser.py

Ich habe den Parser um die Methoden readValuesFromDatabase() und writeCursorToDatabase() erweitert und die Init geändert.

Table 4: Ausgabe des überarbeiteten Parsers

```
<Section: DEFAULT>
<Section: user>
name: Herbert
lastname: Smith
email: neuHerbert@uwu.com
password: x45655554xoxoxo


<Section: DEFAULT>
<Section: user>
name: Herbert
lastname: Smith
email: neuHerbert@uwu.com
password: x45655554xoxoxo

ID: 0
name: Herbert
lastname: Smith
email: neuHerbert@uwu.com
password: x45655554xoxoxo
```

3. Panda3d

3.1 Kollision

3.1.1 Erklärung

Jedes Objekt wird durch einen Kreis ungefähr repräsentiert. Eine Kollsion passiert, wenn die Distanz zwischen zwei Objekten kleiner ist als der Summe ihrer Radii (Radiussen? XD).

Im Spiel haben wir die Kollisionen von Asteroid – Schiff und Asteroid – Bullet.

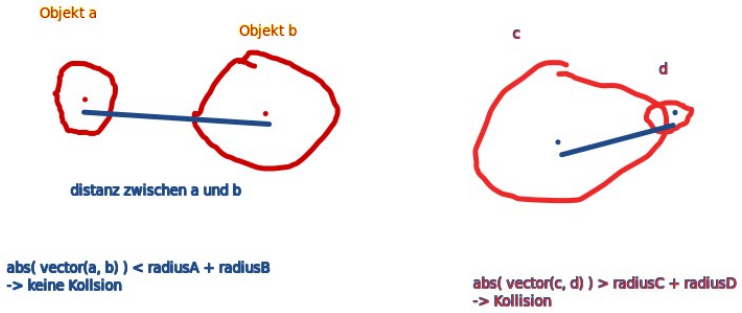

Figure 1: Skizze Kollision

3.1.2 und 3.1.3 Implementierung von und Vorgehen bei Kollision

Table 5: Beispiel: Bullet-Asteroid

```
def checkBulletAsteroidCollision(self):
for bullet in self.bulletList:

for i in range(len(self.asteroids)-1, -1, -1):
asteroid = self.asteroids[i]

# if the bullet and asteroid distance is less than the sum of their radii
if ((bullet.getPos() - asteroid.getPos()).lengthSquared() < (((bullet.getScale().getX() +
asteroid.getScale().getX()) * .6) ** 2)):
# make it that the bullet gets removed in the next gameloop and handle the hit
self.setExpires(bullet, 0)
self.asteroidHit(i)
```

Alle Kollisionen werden einmal pro GameLoop abgefragt.

Bei der Bullet-Asteroid Kollision gibt es zwei for-loops, um bei allen Objekten die Kollision zu prüfen. obj.getPos() - obj2.getPos() gibt uns einen Vektor und mit .lenghtSquared rechnen wir dessen Quadrat der Länge aus.

obj.getScale().getX() gibt uns die Größe des Objektes und wir behandeln das wie den Durchmesser, also müssen wir die beiden Durchmesser mal 0.5 rechnen für die Summe der Radii.

Weil die Länge quadriert ist müssen wir die Radii auch quadrieren und dann können wir beide vergleichen.

Mit setExpires(obj) wird das Objekt im nächsten Gameloop entfernt.

Table 6: wenn ein Asteroid getroffen wurde

```
def asteroidHit(self, index):

self.playMusic(self.PATH + "/sounds/stone.ogg")

# if the asteroid is too small remove the asteroid and exit the function
if(self.asteroids[index].getScale() < self.AST_MIN_SCALE):
self.thisPlayer.score += 1111
self.asteroids[index].removeNode()
del self.asteroids[index]

self.thisPlayer.score += 333
# if there are no asteroids left
if len(self.asteroids) == 0:
self.thisPlayer.score += 666
self.levelUp()
self.playerNode.setText(str(self.thisPlayer))
return
oldPos = LPoint3(self.asteroids[index].getX(), self.asteroids[index].getZ())
oldVel = self.getVelocity(self.asteroids[index])
oldScale = self.asteroids[index].getScale().getX()

# delete the old asteroid after its properties have been stored
self.asteroids[index].removeNode()
del self.asteroids[index]

self.playerNode.setText(str(self.thisPlayer))

# make one Asteroid with the same velocity and one with negative velocity
for i in (-1, 1):
self.newAsteroid(pos=oldPos, vector=oldVel * i, scale=oldScale *
self.AST_SCALE_SMALLER)
```

Zuerst wird Musik gespielt und z.B. 333 Punkte vergeben.

Wenn der Asteroid zu klein ist, wird dieser aus der Liste entfernt und es könnten 1111 Punkte extra vergeben werden.

Wenn dann kein Asteroid mehr da ist, werden viele Punkte vergeben und levelUp() wird aufgerufen.

Wenn der Asteroid groß genug ist, werden zwei kleinere Asteroid geschaffen, wovon der eine die selbe Geschwindigkeit und der andere die entgegengesetzte Geschwindigkeit des ursprünglichen Asteroids hat. Die neuen Asteroids sind um den Faktor AST_SCALE_SMALLER kleiner.

Bei der Schiff – Asteroid Kollision ist das Prinzip gleich, bei einem Treffer passiert nur etwas anderes.

3.2 MusikPanda

3.2.1 und 3.2.2 Abspielen mit Space

Quellcode siehe MusikPanda.py

3.2.3 Funktionstest

Es funktioniert alles wie vorgesehen. Der Sound wird bei Drücken der Space-Taste einmal abgespielt. Man müsste den Dateipfad mit der os Bibliothek noch anpassen.

Die Variable lastPressed in dem Tipp wird nicht benötigt, wenn man das Abspielen der Musik so macht, wie bei 3.2.2 beschrieben.