Simulation von Boids
nach Craig Reynolds

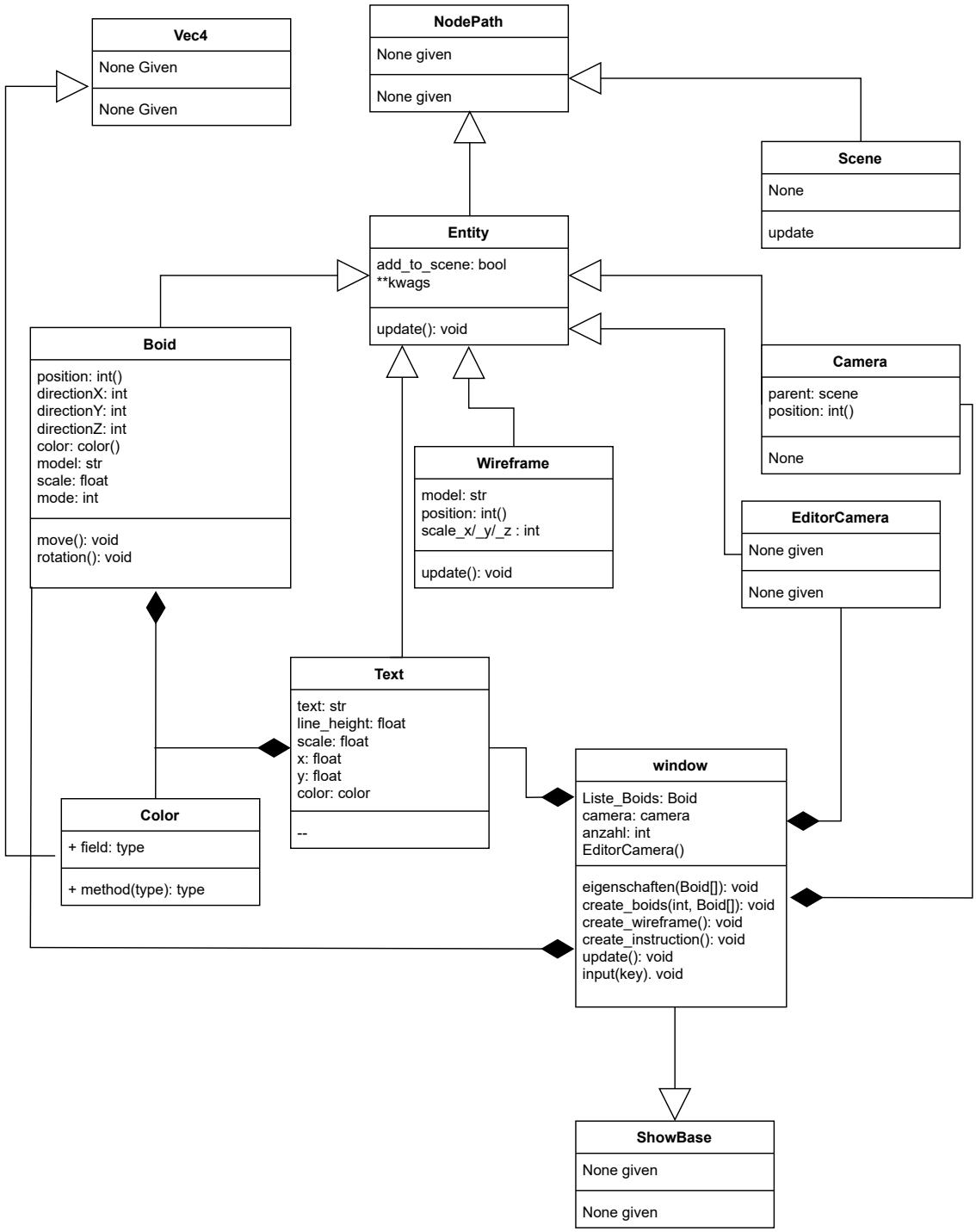
Oliver Fritzler

February 3, 2022

Inhaltsverzeichnis

Contents

1 UML-Diagramm



2 Grundidee

2.1 Umgebung

Die Programmiersprache ***Python*** war vorgegeben, die Bibliothek zur grafischen Darstellung war frei überlassen. Aufgrund der einfachen und verständlichen Syntax, fiel die Auswahl auf die ***ursina engine***, welche auf der ***panda3D engine*** basiert. Ursina ermöglicht es neben vorgegebenen Modellen auch blender-Dateien als Modelle für die einzelnen Objekte zu verwenden. Diese Objekte werden dargestellt und durch spezifische Funktionen (update(), move()) jedes Bild aktualisiert. Dadurch kann man sowohl die Position durch Addition verändern, als auch Eingaben des Benutzers verarbeiten.

2.2 Darstellung des Raumes

Zur Darstellung des Raumes werden 12 Objekte in Form eines Würfels erstellt, jedoch werden diese dann so skaliert, dass sie die Form eines Rechteckes annehmen. Diese werden dann so angeordnet, dass sie die Kanten eines Würfels bilden. Des Weiteren werden die Seitenflächen erstellt. Dabei werden wie davor 6 Objekte in Form eines Würfels erstellt. Nur werden diese jetzt so skaliert, dass sie eine quadratische Fläche bilden. Um diese unsichtbar zu machen wird der Sichtbarkeitswert auf Null gesetzt.

2.3 Allgemeines zu den Boids

Boids sind Körper, die sich Bewegen. Zur Veranschaulichung der Richtung sollten die Boids einen Körper mit einer Spitze haben, wie zum Beispiel ein Kegel. Da ein Kegel sehr viele Seiten hat, die alle jedes Bild berechnet werden müssen und dadurch die Performance senken, sollte man auf einen einfacheren Körper mit Spitze zurückgreifen, wie eine Pyramide. Diese Körper bewegen sind jedoch nicht chaotisch, da sie sich an drei Regeln handeln, die später erklärt werden.

Verhalten an Wänden Das Verhalten der Boids an den Wänden kann man auf verschiedene Weise gestalten. In dieser Simulation wurden drei davon versucht umzusetzen.

1. **Warp**
   Bei dieser Umsetzung werden die Boids bei Kontakt mit einer Wand auf die gegenüberliegende Seite teleportiert.

2. **Wände vermeiden**
   Dabei drehen die Boids vor Kontakt mit einer Wand ab.

3. **Pong**
   Hier verhalten sich die Boids, wie der Ball im Spiel **"PONG"**, d.h. sie bewegen sich auf die Wand in einem bestimmten Winkel zu und verlassen diese Wand dann in dem gleichen Winkel, nach dem *Einfallswinkel-gleich-Ausfallswinkel-Prinzip*

2.4 Regeln

Boids sind Objekte die sich in einem Raum bewegen. Dabei verfolgen sie drei Grundregeln:

- **Seperation**
  Diese Regel besagt, dass jeder einzelne Boid versucht, keinen anderen Boid zu treffen.

- **Alignement**
  Diese Regel besagt, dass jeder Boid versucht, in die selbe Richtung wie ein anderer sich zu bewegen. Dadurch entsteht ein sogenannter "Flock" also ein Schwarm von Boids.

- **Coheseion**
  Diese Regel besagt, dass die Boids in die Mitte des Schwarms steuern.

2.5 Eingabe

Der Benutzer soll durch Bewegen der Maus die Kamera rotieren können und durch Tastendruck die Kamera bewegen können. Außerdem soll es ihm ermöglicht werden die Anzahl der Boids und deren Verhalten zu ändern.

3 Umsetzung

3.1 Allgemein

Aufrgund der Umsetzung der Eigesnchaften ist es nicht möglich die Darstellungsebene und die Informationsebene in verschiedene Dateien aufzuteilen. Es würde ein Import-Fehler auftreten, da die Liste aus der Darstellungsebene in die Informationsebene importiert werden müsste.

3.2 Fenster

Mit dem Aufrufen der Klasse ***Ursina*** wird ein Fenster erschaffen, welches am Ende ausgeführt wird, mit all den benötigen Grafiken. Diesem Fenster kann man einen Titel vergeben, welche dann als Task angezeigt wird. Diese Klasse hat wie jede andere Klasse in Ursina die Funktion ***input()*** mit deren Hilfe man die Eingabe des Benutzers verarbeiten kann, die in dieser Umsetzung hier nachzulesen sind. Um dem Benutzer die Nutzung der Simulation zu vereinfachen, erstellt man in der Funktion ***createInstruction()***, mit der Klasse ***Text*** eine Beschreibung und skaliert und positioniert sie so, dass sie nicht im Vordergrund liegt.[Code]

3.3 Raum

In der Funtkion ***createWireframe()*** werden 12 Objekte der Klasse ***Entity*** in Form eines Würfel erstellt, die so verformt werden, dass sie zu Quadern werden und so positioniert werden, dass sie die Kanten eines größeren Würfels bilden. Darauf werden wie zuvor 6 Würfel erstellt, jedoch werden diese so skaliert, dass sie zu quadratischen Flächen werden, welche nach der Positionierung die Seitenfläche des Würfels bilden. Da diese Seiten nicht nötig zu sehen sind, kann der 4. Faktor der Farbe auf 0 gesetzt werden oder die Variable α, die den Sichtbarkeitswert darstellt, auf 0 gesetzt wird. Bei dieser Umsetzung wird letzteres benutzt.[Code]

3.4 Boids

Die Darstellung der Boids erfolgt durch die Implementierung der gleichnamigen Klasse ***Boid***. Diese Klasse erbt von der Klasse ***Entity***, welche laut den Entwicklern von Ursina die "god class" ist. Den Boids wird eine selbsterstellte .blender-Datei als Model gegeben. Dieses Model gleicht einer Pyramide. Die Farbe dieser Pyramide wird mithilfe des *random_color()* Befehls der color Klasse festgelegt, welche innerhalb self.color gespeichert wird. Des

Weiteren werden den Boids Positionsvariablen, die sowohl in einem Tuple unter self.position als auch als einzelne Werte in self.x, self.y und self.z abgespeichert wird. Um die Bewegung zu ermöglichen wird dem Boid

3.5 Eingabe

4 wichtigste Funktionen

5 Abbildungsverzeichnis

6 Quellennachweis