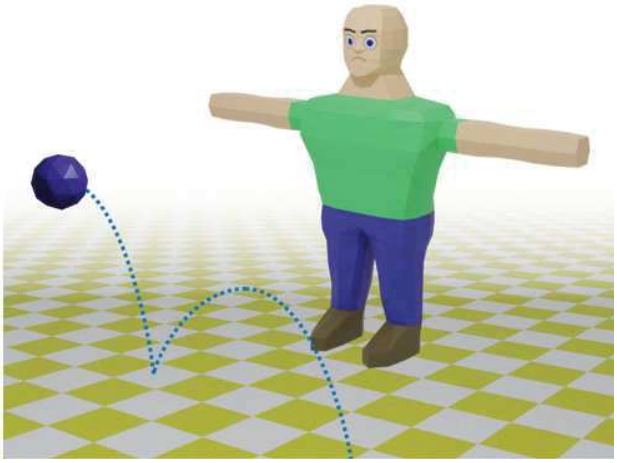
Entwicklung einer Game Engine

Heel Johannes
Kasemann Joel M.
Walch Paul

ProjektbetreuerInnen
DI Rusch Helmut Elmar



Ausgangslage

Computerspiele erfreuen sich heutzutage immer größerer Beliebtheit. Zur Erstellung dieser wird in den meisten Fällen eine sogenannte Game Engine verwendet, ein Grundgerüst mit diversen Funktionen, welches das Ausprogrammieren erleichtert.
Da es zahlreiche Wünsche und Ideen gibt, was eine Game Engine alles können und machen soll, entstand die Idee, eine eigene Engine als Open-Source-Projekt zu entwerfen, bei der alles so programmiert werden kann, wie der/die NutzerIn es will. Stand Beginn der Diplomarbeit gab es kein brauchbares Projekt, das einsehbaren Quellcode hat. Sollte eine Funktion nicht verfügbar sein, soll jede/r NutzerIn nach eigenen Vorlieben die Engine modifizieren können.

Umsetzung

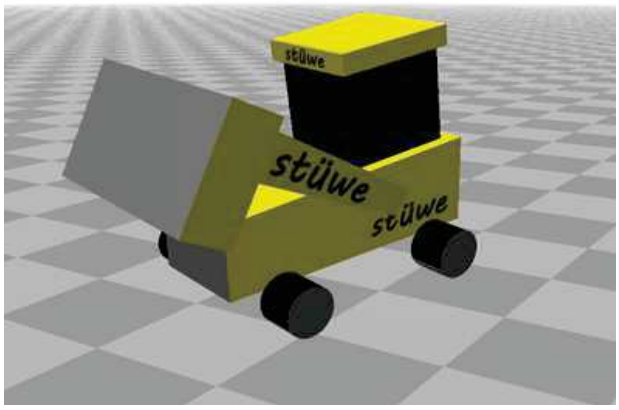
Als Grundlage für die Game Engine wird die Library Helix Toolkit verwendet. Diese ist hauptsächlich für das Importieren und die Anzeige von 3D-Objekten zuständig. Die Niob-Game-Engine ergänzt dazu noch Eingaben sowie eine Physik- und Audio-Engine.
Die Audio-Engine basiert auf einer eigenen Weiterentwicklung von OpenAL, einer frei verfügbaren Audio-Library.
Um physikalische Vorgänge zu simulieren, wurde eine positionsbasierte Physik-Engine implementiert, die auch Kollisionen zwischen unterschiedlichen Objekten unterstützt.

Ergebnis

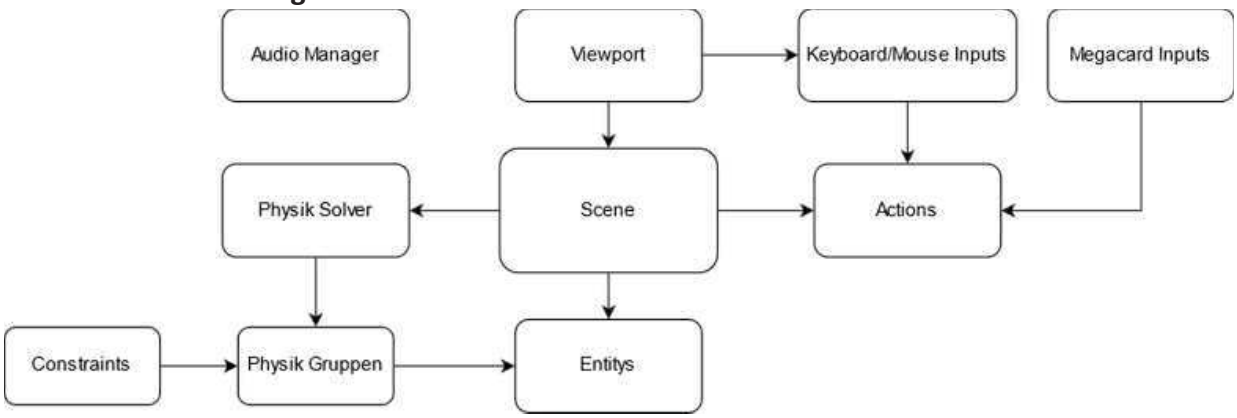
Das Resultat ist eine Game Engine für WPF-Anwendungen, die unter anderem eine MEGACard-Engine-Kommunikation unterstützt, Audiomanagement übernimmt, zahlreiche Physics-Berechnungen durchführt und die Implementation von 3D-Grafiken enorm vereinfacht, sodass schon früher im FSST-Unterricht 3D-Programmierung erklärt werden kann. Außerdem wurde eine Grundlage für zahlreiche weitere Projektmöglichkeiten geschaffen.

Logo

3D-Objekt aus der Demonstration



Ausschnitt aus der Engine-Architektur



Ausschnitt: Code zum Erstellen einer Kette

```
NiobScene scene = new NiobScene();
scene.Entities.Add(new NiobPlane());
scene.Entities.Add(
        new NiobEntity3d("chain.fbx", "") {
            Position = new Vector3(0, 250, 0)
        });

PhysicsGroup group = new PhysicsGroup();
group.GlobalAcceleration = new Vector3(0, -300, 0);
scene.Physics.PhysicsGroups.Add(group);
var constraints = scene.Physics.Constaints;

IPhysicsObject preChain = null;
for (int i = 0; i < 8; i++)
{
    var chain = new NiobEntity3d("chain.fbx", "");
    chain.Position = new Vector3(0, 232 - 18 * i, 0);

    PhysicWrapper pchain = new PhysicWrapper(
        chain, new CuboidBounding(0.2f, 3, 2), 1, false);

    group.Add(pchain);
    scene.Entities.Add(chain);
    if (i==0)
    {
```

Physikalische Simulation von Ketten