CT Projekt: Raycasting Game Engine II (Huskyrock)

Christian Korn

14.01.2022 - 04.06.2022

Inhaltsverzeichnis

1 Ziele **2**
1.1 Muss-Ziele 2
1.2 Kann-Ziele 2

2 Verwendete Technologien **3**
2.1 Rust 3
2.2 Entity component system 3

3 Quellen **4**

1 Ziele

1.1 Muss-Ziele

Hundefels

Das neue Projekt muss die Kernfunktionalität des Vorgängerprojektes enthalten:

- Anzeigen eines 2D Levels in 2,5D per Raycasting
- Bewegungsfreiheit im Level
- Anzeige von Entities im Level
- Kollisionserkennung

Neues

Natürlich soll das neue Projekt auch eigene Features und Technologien enthalten, die es vom Alten unterscheiden:

- Wand-Texturen
- Sprites für Entities
- KI für Entities

Das neue Projekt wird in der Programmiersprache Rust geschrieben, anstatt eines objektorientierten Designs wird ein Entity component system benutzt.

1.2 Kann-Ziele

Hundefels

Einige Features des Vorgängerprojektes werden als Kann-Ziele klassifiziert, da sie nicht für die Funktionalität des Projektes absolut notwendig sind.

- Laden von Leveln aus externen Dateien (wäre durch neue features komplizierter)
- Command-Line Argumente (Funktionalität größtenteils überflüssig oder besser als Konfigurationsdatei zu lösen)

Andere

- Visuelle Effekte (view-bobbing, etc.)
- Schießen
- Kartenansicht

2 Verwendete Technologien

2.1 Rust

Rust ist eine Multiparadigmen-Systemprogrammiersprache mit einem Fokus auf Performance und Sicherheit. Dies wird erreicht, indem anstatt eines Garbage Collectors oder händischem Memory managements ein sogenannter Borrow Checker verwendet wird, der sogenannte Lifetimes und Scope von Variablen überprüft und Regeln für Referenzen durchsetzt.

2.2 Entity component system

Entity component system (kurz ECS) ist ein Softwarearchitekturmuster, welches vor allem in der Spieleentwicklung genutzt wird.
Im Gegensatz zu objektorientierten Programmen werden im ECS Daten und Funktionalität nicht gruppiert, sondern getrennt in sogenannte "Components" (Daten) und "Systems" (Funktionalität). Zentral im ECS sind die "Entities": Objekte, die Components enthalten, auf welche Systems angewandt werden.
Systems können spezifizieren, welche Components für sie notwendig sind und werden nur auf die Entities angewandt, die alle benötigten Components enthalten.
Zum Beispiel wird in diesem Projekt mit einem Component definiert, ob eine Entity der Spieler ist, und deshalb eine Kamera enthalten sollte, mit einem anderen ob die Entity mit einer KI ausgestattet ist. Entities ohne diese beiden Components sind stationär oder bewegen sich gleichförmig ohne steuerung (abhängig davon, ob sie die Bewegungs-Component enthält).

3 Quellen

- Viel Code wurde direkt vom Vorgängerprojekt zu Rust übersetzt und wiederverwendet.
  `https://github.com/MacAphon/hundefels2d`
- Steve Klabnik & Carol Nichols: "The Rust Programming Language"
  `https://doc.rust-lang.org/stable/book/`
- "Rust by Example"
  `https://doc.rust-lang.org/rust-by-example/`
- "Rust Coockbook"
  `https://rust-lang-nursery.github.io/rust-cookbook/`
- "Learn Game Development in Rust"
  `https://sunjay.dev/learn-game-dev`
  `https://github.com/sunjay/rust-simple-game-dev-tutorial/`
- 3DSage: "Make Your Own Raycaster Part 1"
  `https://youtu.be/gYRrGTC7GtA`
  Quellcode verfügbar unter
  `https://github.com/3DSage/OpenGL-Raycaster\_v1`