Alexander Jaroch

Medieninformatik B.Sc.
8. Semester

PRIMA – Designdokument

Sommersemester 2022

Dozent: Prof Dr. Jirka Dell'Oro-Friedl

Idee: Chess Simulator

Die Idee ist es ein 3D-Schachspiel zu entwerfen. Dabei soll aus zeitlichen Gründen nicht die Spiellogik im Vordergrund stehen. Am Ende soll stattdessen ein Schachsimulator stehen mit

- einem festen Spielbrett
- bewegbaren Schachfiguren
- simulierter Physik
- Animationen

Hierarchie und Koordinatensysteme

In der FUDGE-Engine liegen die Koordinatenachsen wie folgt:

- x-Achse: Breite (von links nach rechts)
- y-Achse: Höhe (von unten nach oben)
- z-Achse: Tiefe (von hinten nach vorne)

Die Höhenachse wird insbesondere bei der Physiksimulation sehr wichtig. Objekte mit einer Rigidbody-Komponente fallen immer von $\hat{y}$ (oben) nach $-\hat{y}$ (unten). Dabei fallen sie immer entgegen der Richtung der Weltkoordinate $\hat{y}$.

Deshalb liegt in der Szene das Schachbrett in der Mitte der Welt. Die Struktur ist dabei wie folgt aufgebaut:

- **Scene**: Hauptszene
    - **Board**: Ursprung für das Spielbrett
        - **Center**: Mittelpunkt des Spielbretts
            - **GrabTool**: Objekt, an welches die Figuren angehängt werden, um diese zu bewegen
            - **Floor**: Boden weit unter dem Schachbrett, um herunterfallende Figuren aufzufangen (Hintergrund)
        - **Pattern**: Container für das Schachbrettmuster
            - **8 x Row**: Reihen, welche abwechselnd helle und dunkle Felder beinhalten
                - **Tile**: Ein einzelnes Feld (hell oder dunkles Material)
            - **Characters**: Container für Reihen- und Spaltenbezeichnung
                - **Top**: Obere Hälfte für einen Spieler
                    - **16 x Character**: Zeichen, welches mittels Sprites texturiert wurde (A-H, 1-8)
                - **Bottom**: Untere Hälfte für anderen Spieler
                    - **16 x Character**: s. o.
        - **GlobalLighting**: Container für Lichter
            - **AmbientSource**: Umgebungslicht (generelles Aufhellen)
            - **PointSource**: Punktlicht für eine natürlichere Beleuchtung

Für die Physikberechnungen liegen auf dem **Board** und auf dem **Floor** jeweils eine Rigidbody-Komponente. Die einzelnen Figuren haben ebenfalls jeweils eine Rigidbody-Komponente, wodurch die Schachfiguren untereinander und mit dem Board kollidieren können.

Schachfiguren

Die Figuren haben jeweils einen eigenen Graphen und wurden aus mehreren Nodes und Meshes zusammengesetzt. Um die Physikberechnungen einfach zu halten, gibt es für jeden Schachfigurengraphen jeweils genau eine Rigidbody-Komponente.

Bei den Schachfiguren hat der FUDGE-Editor den Vorteil, dass ihre Graphen direkt im Editor zusammengesetzt werden und anschaulich mit einer Rigidbody-Komponente versehen werden konnten. Sie erben außerdem von der Node-Klasse, wodurch sie wie eine Node behandelt werden und trotzdem eine eigene Logik beinhalten können:

- **position**: Koordinaten der Schachfigur für die Spiellogik (nicht verwendet)
- **side**: „Farbe" der Figur (hell oder dunkel)
- **type**: Figurtyp (Bauer, Turm, Springer, Läufer, Dame, König)
- **createPiece()**: erzeugen einer Spielfigur aus dem Graph heraus
    - wird dem Spielbrett als Kind angefügt
    - Events und Logik für das Abspielen von Sound bei Kollision

Es werden drei Varianten von Aufprallgeräuschen verwendet. Bei einer Kollision mit dem Brett oder einer anderen Spielfigur wird zufällig einer dieser drei Sounds abgespielt. Die Lautstärke richtet sich an die Stärke des bei der Kollision entstehenden Impulses.

Das Auswählen der zu greifenden Figur sowie das Bewegen der gegriffenen Figur findet über Raycasts statt. Ein Strahl (Ray) wird ausgehend vom Mauszeiger mit Einbeziehung der Kameratransformation ausgesendet und kollidiert mit einer Figur, welche dann durch Anhängen an die GrabTool-Hierarchie „gegriffen" werden kann.

Ein weiterer Raycast kollidiert beim Bewegen der Maus mit dem Spielbrett, wodurch die Zielposition für die gegriffene Figur ermittelt wird. Beim Loslassen der Figur, wird die letzte Geschwindigkeit herangenommen und in eine Krafteinwirkung übersetzt, welche auf die Figur nach dem Loslassen wirkt. Dadurch behält die Figur auch nach dem Loslassen ihre Richtung für eine Zeit lang bei.

ScriptGrabTool

Das Greifen und Loslassen (hinzufügen des Kindes am entsprechenden Knotenpunkt) sowie das Abspielen der Greifanimation geschieht über das Custom-Script für das GrabTool. Dieses liefert folgende Methoden:

- **connect()**:
  - hängt das über den Raycast ausgewählte Objekt (Kind der Schachfigur) an die Hierarchie des GrabTools
  - macht das Objekt kinematisch
  - bereitet Animation für das Anheben und ggf. Aufrichten der Spielfigur auf
  - spielt die Animation einmalig durch Anhängen der Animator-Komponente an das GrabTool ab
    - nach der Animation steht die Figur aufrecht und wird mit Ausreichendem Abstand zum Brett hochgehalten
- **disconnect()**:
  - hängt das zuvor an die Hierarchie des GrabTool angehangene Objekt wieder an das Ursprungsobjekt (Schachfigur)
  - gibt die Transformation des Objekts an das Elternobjekt (Schachfigur) weiter
  - macht das Objekt dynamisch
  - nimmt die letzte Geschwindigkeit und übersetzt diese in eine Krafteinwirkung

Spiellogik

Die Singleton-Klasse für die Spiellogik hat die Attribute und Methoden:

- **pieces**: Figuren im Spiel
- **time**: Vergangene Spielzeit
- **side**: Spielende Seite
- **adder** und **remover** für Figuren
- **reset**: Zurücksetzen des Spielzustands
- **init**: Initialisieren der Spiellogik und der Spielfiguren

Über einen Mutator werden die Attribute **time** und **side** an das VUI übergeben und dort zeitgetreu dargestellt.

Userinterface

Die Klasse erweitert die Klasse Mutable, wodurch ihre öffentlichen Attribute mithilfe eines UI-Controllers an ein virtuelles Userinterface weitergegeben und durchgehend aktualisiert werden können. Außerdem wird hier das Userinterface um interaktive Eingabemöglichkeiten erweitert:

- Button, um die spielende Seite wechseln (Kamera rotiert um festen Punkt auf die andere Seite)
- Eingabefelder für die Farben der Spielfiguren auf beiden Seiten
- Rücksetzbutton, um den Spielzustand und die Spielfiguren zurückzusetzen

Verwaltung von Ressourcen

Mit einer Konfigurationsdatei werden die verschiedenen Resourcen, welche im Code verwendet werden, verwaltet. Die Datei erleichtert in Kombination mit der Klasse ResourceManager den Zugriff, indem es mit kurzen Keys auf die unhandlichen Ressourcen-Ids verweist:

- Ressourcen werden zu Beginn geladen
- Kurze, eindeutige Keys werden auf Ressourcen(-IDs) gemappt
- Auf Ressourcen wird über die Methoden des ResourceManagers und dem Key zugegriffen

```
{
    "type": "graph",
    "name": "Pawn",
    "resource": "Graph|2022-07-18T08:58:02.952Z|87490"
},
```

- Bsp.: Pawn → Graph|2022-07-18T08:58:05.952|87490
- Die Ressource kann geändert werden
    - Die Figur wird mit der neuen Figur ersetzt sich beim erneuten Laden

Kriterienkatalog

Nr	Kriterium	Erklärung
1	Units and Positions	Das Schachbrett liegt im Weltkoordinatensystem so, dass die Physik richtig wirken kann. Es liegt auf der x-z-Ebene und die Figuren stehen senkrecht dazu, also parallel zur Weltkoordinatenachse y. Die simulierte Schwerkraft wird entgegen der y-Achse. Die lokalen Koordinaten der Figuren und des Bretts selbst stimmen mit richtungsmäßig weitestgehend mit den Weltkoordinaten überein.
2	Hierarchy	Es gibt beim Schach-Simulator eine Gesamtszene mit dem Spielbrett, einen Graphen für das GrabTool, mit welchem die Figuren bewegt werden können, und jeweils einen Graphen für jeden Figurentyp (Bauer, Turm, Springer, Läufer, Dame, König). Die Hauptszene beinhaltet nur das Brett und das GrabTool. Die Figuren werden beim Initialisieren der GameLogic auf dem Brett platziert. Durch eine gute Hierarchie können zusammengehörende Elemente zusammenbleiben und z.B. problemlos angepasst oder wiederverwendet werden.
3	Editor	Wie oben schon erwähnt werden die Figuren durch eine Funktion im Code auf dem Spielbrett platziert. So können, neben Reduzierung von Fleißarbeit durch manuelles Platzieren, zudem die Figuren mit Zusatzfunktionen ausstatten werden. Da die Initialisierung in eine eigene Funktion ausgelagert ist, können diese jederzeit beliebig oft neu aufgestellt werden.
4	Scriptcomponents	Eine Skriptkomponente wird für die Steuerung des Aufhebe-Mechanismus verwendet. In ihr kann der gesamte Code für diese Funktion ausgelagert werden, wodurch die Game Loop sauberer bleibt. Außerdem kann die Skriptkomponente theoretisch wiederverwendet werden. Z.B. für einen zweiten Greifer (GrabTool), welcher dann unabhängig vom ersten auch Objekte greifen und loslassen kann (Greifen = Anhängen von Objekt an die Greifer-Komponente, welche wiederum mit der Maus bewegt wird, mehr dazu in B).
5	Extend	Wie schon in 3 angedeutet, kann von FudgeCore-Klassen geerbt werden. Die Klasse für die Schachfiguren erbt hier von der Node-Klasse von FudgeCore. Dadurch können die Figuren, wie ganz normale Nodes gehandhabt werden und trotzdem erweiterte Logik haben. So könnten die Figuren z.B. jeweils ihre möglichen Züge speichern und dadurch eine Validierung des Zuges erfolgen.

6	Sound	Es werden drei Sounds verwendet. Das sind drei Varianten von Aufprallgeräuschen, zufällig bei der Kollision mit dem Brett oder einer anderen Spielfigur abgespielt werden. Dabei richtet sich die Lautstärke an die Stärke des bei der Kollision entstehenden Impulses. Die Logik dazu befindet sich in der Klasse der Schachfigur, welche von Node erbt.
7	VUI	Es gibt ein simples VUI, welches unveränderlich die Zeit und die aktuell spielende Seite anzeigt. Diese Werte werden mithilfe von FudgeUserInterface und Mutables aktualisiert. Des Weiteren wurde das VUI um interaktive Komponenten, wie der Möglichkeit zum Wechseln der spielenden Seite, zur Einstellung der Farben der Spielfiguren beider Seiten sowie einem Reset-Button erweitert
8	Event-System	Das Event-System wird an mehreren Stellen verwendet. Zum einen werden mithilfe von Kollisions-Events die Sounds für den Aufprall abgespielt. Des Weiteren werden Pointer-Events für das Greifen (pointer down, pointer up) und Bewegen (pointer move) der Figuren verwendet. Außerdem werden allgemeine Events wie das Anfügen einer Komponente in Skriptkomonenten verwendet und vorausgesetzt.
9	External Data	Mit einer Konfigurationsdatei werden die verschiedenen Ressourcen, welche im Code verwendet werden, verwaltet. Mit der JSON-Datei sollte der Verwendung verschiedener Ressourcen-IDs im Code entgegengewirkt werden. In der Datei werden einfache Schlüssel auf die unhandlichen Ressourcen -IDs gemappt, wodurch deren Handhabung stark erleichtert wird. Auf diese Weise wurden im Verlauf des Projekts schon mehrmals Ressourcen schnell und einfach durch andere Ressourcen ausgetauscht.
A	Light	Es werden zwei Lichtquellen verwendet: Ein Point-Light (vgl. Sonne: stark, "wirft Schatten") und ein Ambient-Light (vgl. Himmel: diffus, erhellt Umgebung). Durch die Verwendung dieser beiden Quellen sollten natürlich wirkende Lichtverhältnisse geschaffen werden.
B	Physics	Jede Figur, und das Spielbrett haben einen Rigidbody, wodurch sie miteinander kollidieren und interagieren können. Zunächst sollten Joints für das Greifen der Figuren verwendet werden. Allerdings hat das mit dem Bewegen des Joints nicht ideal funktioniert. Die Figur bleibt nicht am Joint und schleift am Brett entlang. Das Auswählen der zu greifenden Figur sowie das Bewegen der gegriffenen Figur findet über Raycasts statt. Ein Strahl wird ausgehend vom Mauszeiger mit der Richtung der Kamera ausgesendet und "kollidiert" mit einer Figur. Diese Figur kann dann "gegriffen" werden. Ein weiterer Raycast kollidiert beim Bewegen der Maus (pointer move event) mit dem Spielbrett, wodurch die neue Position für die gegriffene Figur bestimmt wird. Beim Loslassen der Figur, wird die letzte Geschwindigkeit genommen, um eine Kraft auf die Figur zu wirken, wodurch sie "geworfen" werden kann.

C	Net	nicht verwendet
D	State Machines	nicht verwendet
E	Animation	Das Greifen wird mithilfe der Animationskomponente animiert. Die Figur wird angehoben, damit sie nicht am Brett entlang schleift und ggf. wenn umgekippt und verschoben in die Ursprungsposition (lokal 0) zurückgebracht. Dazu wird für den Start der Animation der aktuelle Zustand der Transformationskomponente genommen und für das Ende der Animation die lokale Verschiebung und Drehung auf 0. Die Animationsklasse aus diesen "Keyframes" eine Animation, welche über das Anhängen der Animationskomponente abgespielt werden kann. Diese wird beim Greifen angehängt und somit abgespielt (einmalig = PLAYONCE). Für den Hintergrund wurde ein animierter "Sternenhimmel" (eher Boden) mithilfe von Sprites erzeugt. Dieser ist zufällig und somit bei jedem Laden anders.