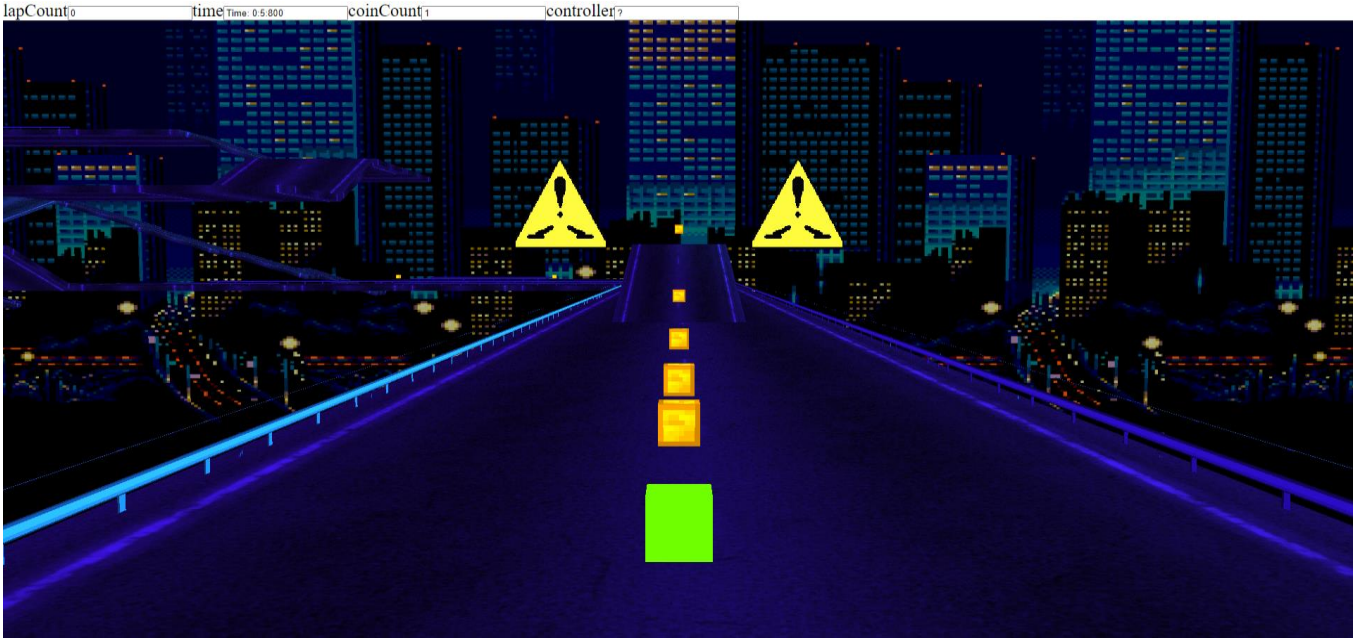
Super Mario Speed Racer – Designdokument

Vasilii Gurev

Wintersemester 2022/23

PRIMA



1. **Was ist 0 und was ist 1?**

0 wird als Startpunkt des Karts definiert. Es ist der Punkt, an welchem das Rennen anfängt und aufhört. 1 wird hier als 1 Meter definiert.

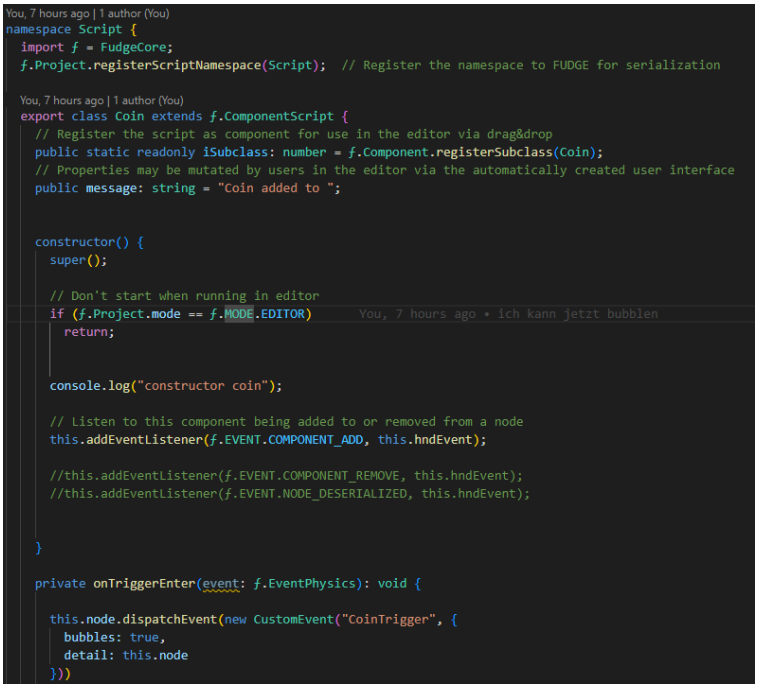
2. **Graph-Setup & Editor**

- **Main**
  - **World**
    a) **Road**
    b) **Blocks**
    c) **…**
  - **Kart**
    a) **Cam**
  - **Items**
    a) **Coins**
  - **Sounds**
    a) **Collectsound**
    b) **Backgroundmusic**

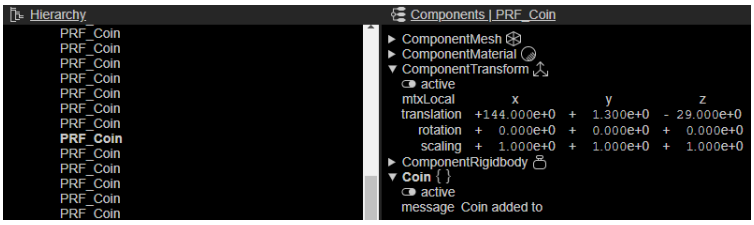
Grob aufgeteilt wird die Hierarchie mit 4 übergeordneten Kategorien. Diese sind World, Kart, Items & Sounds. Unter World werden alle Elemente geordnet, welche hauptsächlich im Viewport erstellt wurden und zuständig für die Spielewelt sind. Dies sind beispielsweise zusammenhängende vorgebaute Straßenstücke oder Umgebungsdekorationen. Die Übergruppe: "Kart" enthält eine Kamera-Komponente, welche über den Code an das ebenfalls mit Code generierte eigentliche Kart gehängt wird. Die Items enthalten die einsammelbaren Münzen, welche im Editor platziert, jedoch durch den Code u.a. mit Custom Script-Components bedient werden. Schließlich enthält die Node Sounds die im Spiel vorhandenen Sounds wie Musik und das Aufsammeln der Coins.

3. Script Components

Die Script Components wurden für die Coins benutzt, um Events wie das Aufsammeln einer Münze zu feuern und auch, um den Sound dabei abzuspielen und den Münzzähler zu verändern.



```
You, 7 hours ago | 1 author (You)
namespace Script {
  import ƒ = FudgeCore;
  ƒ.Project.registerScriptNamespace(Script);  // Register the namespace to FUDGE for serialization

  You, 7 hours ago | 1 author (You)
  export class Coin extends ƒ.ComponentScript {
    // Register the script as component for use in the editor via drag&drop
    public static readonly iSubclass: number = ƒ.Component.registerSubclass(Coin);
    // Properties may be mutated by users in the editor via the automatically created user interface
    public message: string = "Coin added to ";


    constructor() {
      super();

      // Don't start when running in editor
      if (ƒ.Project.mode == ƒ.MODE.EDITOR)      You, 7 hours ago • ich kann jetzt bubblen
        return;


      console.log("constructor coin");

      // Listen to this component being added to or removed from a node
      this.addEventListener(ƒ.EVENT.COMPONENT_ADD, this.hndEvent);

      //this.addEventListener(ƒ.EVENT.COMPONENT_REMOVE, this.hndEvent);
      //this.addEventListener(ƒ.EVENT.NODE_DESERIALIZED, this.hndEvent);


    }

    private onTriggerEnter(event: ƒ.EventPhysics): void {

      this.node.dispatchEvent(new CustomEvent("CoinTrigger", {
        bubbles: true,
        detail: this.node
      }))
```



4. Extend

Die Klasse Kart extendet ƒ.Node, die Klasse Coin extendet ƒ.ComponentScript. Statistics extendet ƒ.Mutable.

Das Extenden der Klassen war sehr hilfreich beim Verwenden der Methoden und Eigenschaften von Node. Das Extenden vom Component Script hat das Implementieren spezieller Funktionen für meinen Anwendungszweck deutlich erleichtert.

5. Sound

Ein Rennspiel soll schnelllebig, spannungsgeladen und unterhaltsam sein. Genauso sollten die Sounds gewählt werden, denn diese sind für die Immersion von größter Bedeutung. Ohne es zu kompliziert und zu überladen machen zu wollen, beschränkte ich mich auf 2 verschiedene Sounds um auch dem Umfang eines „Retro“- Spiels gerecht zu werden. Man sollte einen Belohnungssound hören, sobald man eine Münze aufgesammelt hat. Außerdem hat man während des gesamten Rennens gegen die Zeit eine aufregende Hintergrundmusik.

6. VUI

Das VUI habe ich für meinen Zweck mit 2 Resultaten erstellt. Ich brauchte zunächst einen Live-Tweaker sowie eine übersichtliche VUI für den Spieler im Spiel. Zunächst benutzte ich die VUI um Werte wie die aktuelle Geschwindigkeit des Karts, den aktuellen Richtungsvektor und ähnliches auszulesen, sodass ich Fehler beheben oder auch vermeiden konnte. Das VUI sollte jedoch am Ende dafür genutzt werden, dem User die passenden und relevanten Informationen aufzuzeigen. Das VUI zeigt die Nummer der aktuellen Runde, die Anzahl gesammelter Münzen, sowie die aktuelle Zeit an.

7. Event-System

Der Graph „Root“ hört auf das Event CoinTrigger. Es wird auf die spezifische Script Component der Coins gehört, denn es feuert sobald man gegen eine Münze fährt.

```
function initCoins() {

  root.addEventListener("CoinTrigger", onCoinEnter);
  coinsContainer = items.getChildrenByName("Coins")[0];
}

function onCoinEnter(event: CustomEvent) {

  statistics.coinCount = Number.parseInt(statistics.coinCount) + 1 + "";

  soundCollect.play(true);

  coinsContainer.removeChild(event.detail);
}
```

Innerhalb dieses Events erfolgt das Reagieren auf den Trigger, das hochzählen des Münz-Zählers sowie das Entfernen der Münze, denn man darf diese nicht nochmals aufsammeln. Ebenfalls hört der Graph auf ein Event, welches ausgelöst wird, sobald man eine Runde gefahren ist und eine Trigger-Barrier passiert hat. Dieses Event hat ebenfalls das Resultat, dass der Round-Counter (Lap) hochgezählt wird.

Custom-Events sind durchaus hilfreich, da man auf diese Weise nicht nur eine sinnvollere Lösung, sondern auch eine sauberere Arbeitsweise erreicht.

8. External Data

Die Daten, welche man direkt einsehen und ggf. verändern kann sind in der externalData.json zu finden: "maxSpeed" - Die maximale Geschwindigkeit des Karts,"acceleration" - Die Beschleunigung des Karts,"maxSteerAngle" - Der Maximale Drehwinkelwert des Karts,"steeringSpeed" - Die Drehgeschwindigkeit des Karts,"mass" - Die Masse des Karts,"friction" - Die Reibung.

9. Light

In einem gewöhnlichen, alten Markio Kart gibt es keine wirklich realistische Lichtsetzung. Diese dient jedoch dazu, dass Umfeld optisch angenehm zu gestalten. Darum entschied ich mich für eine Zwischenlösung aus Ambient- + Sunlight. Diese beiden Lichter werden jedoch äußerst dunkel abgebildet, da die Rennstrecke bei spätem Abend/Nacht gespielt wird. So harmonisieren die Objekte in der Welt mit dunkler, grauer Stadttextur mit dem dunklen Blau/Lila des Lichtes. Dadurch wird die erzielte Immersion erzeugt.

10. Physics

Die Physik war der mitunter schwerste Teil der Umsetzung. Alles musste so berechnet und gesetzt werden, dass man eine realistische Näherung aber auch gleichzeitig ein immersives Fahrgefühl erhält und empfindet.

Das Kart erfährt Einwirkung von Kräften bei Beschleunigung, Drehung, Bremsung, Fall und Collision. (forces + collisions + torques) Alle Objekte, bis auf Dekorationen wie das Hintergrundbild oder die Warnsymbole, besitzen Ridgi- Bodys um mit dem Kart interagieren zu können.

11. Textures: