Design-Dokument

AutobahnHunt

Abschlussprojekt des Kurses PRIMA

Referent:	Prof. Dipl.-Ing. Jirka R. Dell'Oro-Fridl
Vorgelegt am:	20.07.2023
Vorgelegt von:	Louis Ronald Kuhnt
	263622
	louis.ronald.kuhnt@hs-furtwangen.de

Inhaltsverzeichnis

1 Konzept

Das Konzept-Kapitel bezieht sich auf die Idee und die ersten konkreten Entwürfe des Abschlussprojekts „AutobahnHunt“.

Das nachfolgende Konzept wurde im Wintersemester 2021/22 schon vorgestellt, aber durch unvorhersehbare Gegebenheiten kam das Projekt nicht über die Konzeptionierungsphase hinaus. Alles nachfolgende wurde auf den neusten Stand von Fudge und den Kurs PRIMA erstellt. So haben sich auch grundlegende Design-Entscheidungen geändert.

1.1 Spielprinzip

Das Spielgenre von AutobahnHunt ist ein „Endless-Runner“, wie zum Beispiel „Traffic Rush Highway Lanes“ oder „Subway-Surfer“. Eigenschaften dieses Spielprinzips sind folgende:

- Dimension: 3D
- Environment: Third-Player-Perspektive
- Game-Mechaniken ist das Ausweichen entgegenkommender oder statischer Elemente und somit Punkte für das Weiterkommen im Level zu generieren.
- Ziel: Highscore.

1.2 Erstes Konzept

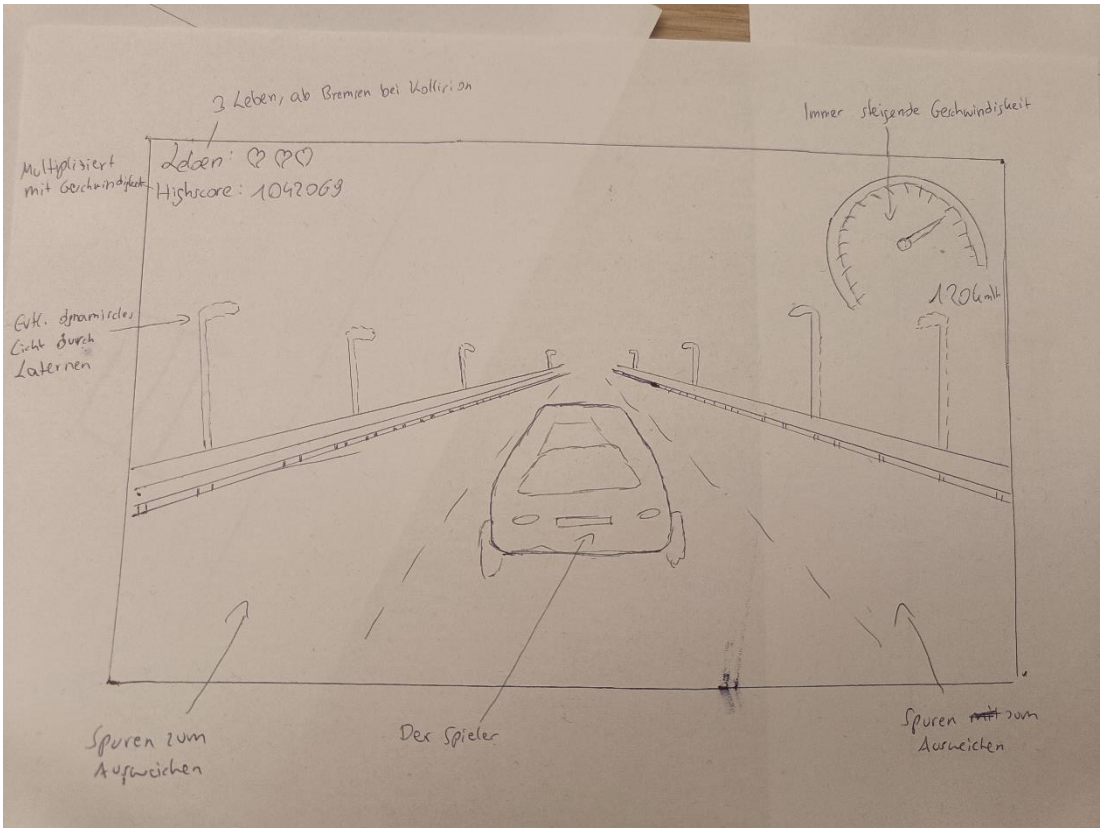

Abbildung 1: Erstes Konzept zum Spiel „AutobahnHunt“

Im ersten Konzept zu sehen ist ein Fahrzeug auf einer Fahrbahn mit zwei Ausweichoptionen zur linken und rechten Seite des Fahrzeuges. Das User-Interface suggeriert das Leben des Spielers, die erreichte Punktzahl und die Geschwindigkeit des Spieler-Fahrzeugs. Die Laternen sollen natürliches Licht auf die Fahrbahn werfen und zur Sicht beitragen.

Damit die Endlosigkeit in dem „Endloss-Runner“ erreicht wird, kommt ein zweites „altes“ Konzept zum Tragen. Auch dieses wurde im Wintersemester erstellt und hat in der aktuellen Umsetzung keine Anwendung mehr, aber auch dies gehört zu einer Konzeptionierungsphase.

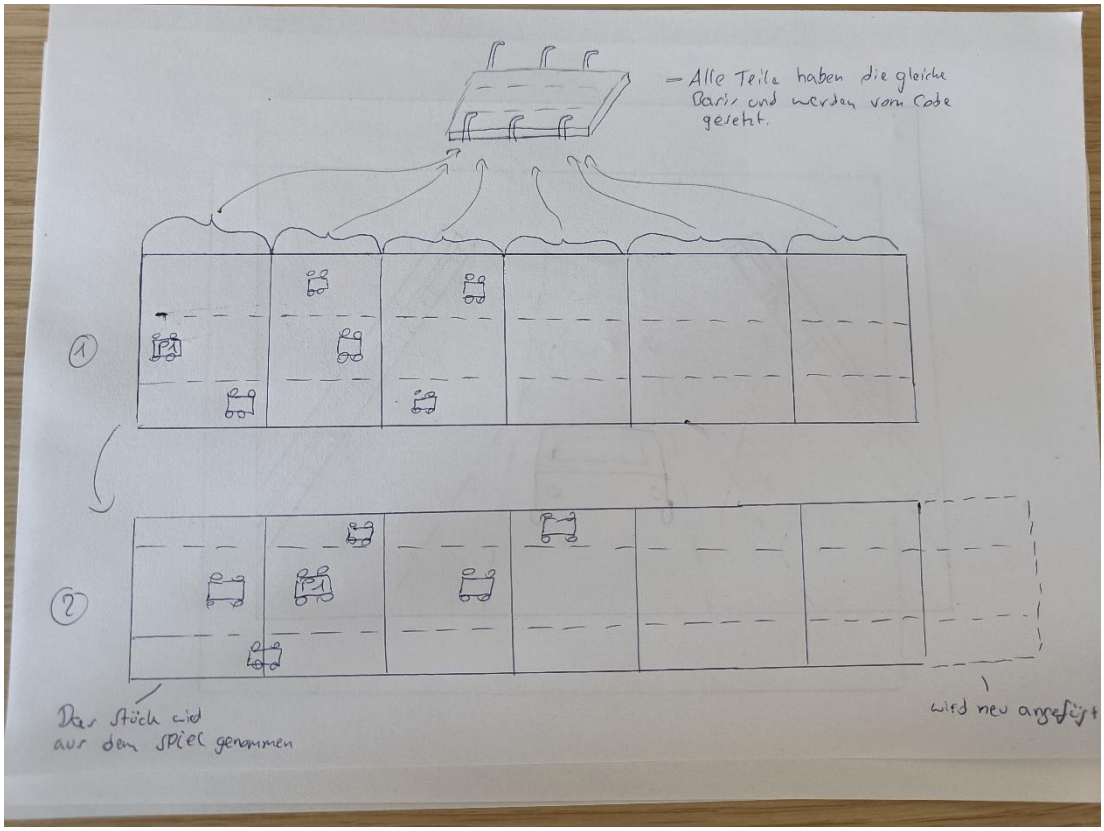

Abbildung 2: Konzept für die "Endlosigkeit"

Im ersten Konzept die Illusion einer Endlosigkeit zu erschaffen, war es geplant die Straßenstücke nacheinander aufzubauen und nach dem Spieler-Fahrzeug wieder „abzubauen“. Es war somit geplant eine Template-Straße zu erstellen und diese wieder zu kopieren und aneinander zu ketten.

Im neusten Konzept, welches im fertiggestellten Spiel zum Einsatz kommt, wird ein anderes System verwendet. Dieses hat die Chance nicht allzu fehleranfällig zu sein und auch potenziell performanter.

1.3 Zweites Konzept

Im zweiten verbesserten Konzept wird das alte Spielprinzip auf die neuen Möglichkeiten von Fudge angepasst.

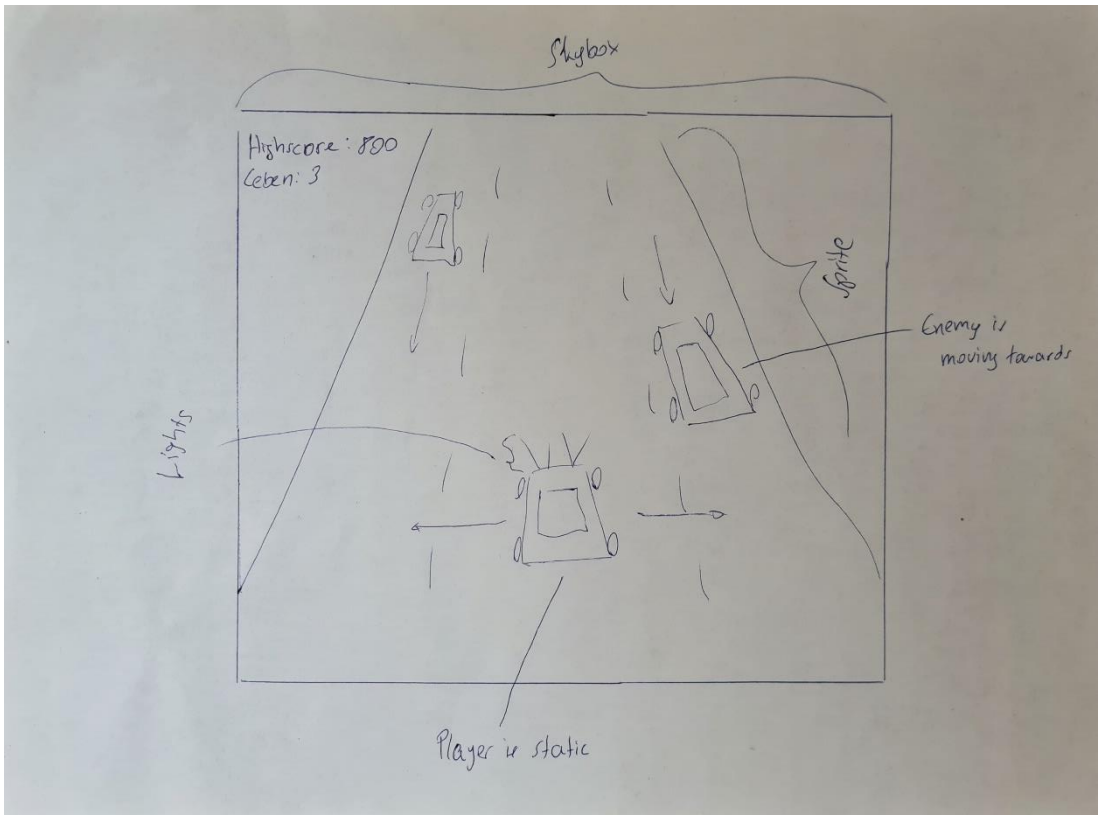

Abbildung 3: Umgesetzte Konzept

Im aktuellen Konzept wird das Problem der Endlosigkeit in Form von Sprites gelöst. Die Fahrbahn wird aus 12 Frames aufgebaut und simuliert die vorbeiziehende Straße. Das Sprite wurde selbst erstellt, sowie alles aus dem späteren Spiel, mit Ausnahme der Skybox.

Weiterhin ergeht aus dem Konzept, dass das Spieler-Fahrzeug nicht fährt, sondern die Gegner-Fahrzeuge auf den Spieler zu steuern. Beim Ausweichen des Spielers, außerhalb der Sicht der Kamera und somit des Spielers, werden die Fahrzeuge gelöscht.

2 Umsetzung

In diesem Kapitel werden die Funktionen des Spieles erläutert mit Fokussierung der Abschlussarbeitskategorien.

2.1 Units and Position

Unter „Units and Position“ wird die Ausrichtung des Spieles auf dem bekannten Koordinatensystem (X,Y,Z) erfragt, sowie der Vergleich zwischen Einheiten im Spiel und auf die reale Welt abgebildet.

Das Spiel „AutobahnHunt“ sitz auf dem Koordinatensystem in dem die X-Achse die horizontale ist. Auf dieser Achse bewegt sich hauptsächlich das Spieler-Fahrzeug. Die Y-Achse ist für die vertikale Beweglichkeit verantwortlich, in diesem Spiel wird diese aktiv im Spiel nicht verwendet. Die Z-Achse ist für die tiefe zuständig. Auf dieser Achse bewegen sich die gegnerischen Fahrzeuge. Das Spieler-Fahrzeug befindet sich auf dem Null-Punkt des Koordinatensystems.

Die Umwandlung von Units in die reale Einheit von Metern, ist in diesem Spiel im Faktor x0.25. Somit ist eine Unit im Spiel ungefähr 0.25 Meter. In der nachfolgenden Abbildung werden die Objektgrößen in Meter aufgezeigt.

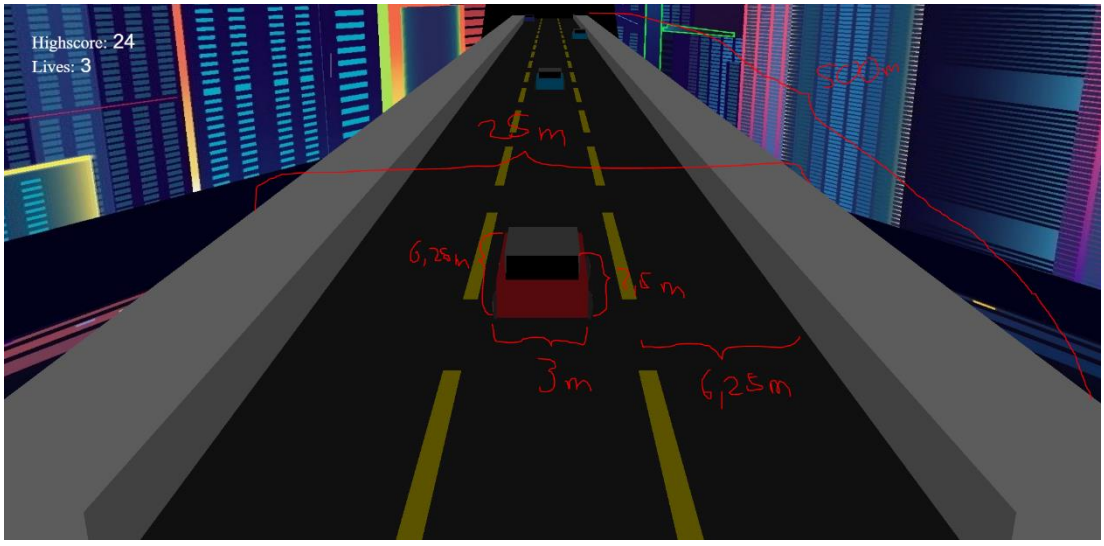

Abbildung 4: Spiel mit Units in Meter

2.2 Hierarchy

Die Hierarchy der erstellten Objekte ist in zusammenhängende Gruppen unterteilt. Das ist sinnvoll, um Objekte von anderen abzugrenzen und somit Übersichtlichkeit zu schaffen.

Des Weiteren ist eine Besonderheit das im Hauptgraph „Game“, noch ein weiterer Graph untergebracht ist. Der Graph „EnemyCar“ wird benötigt, um im Code das komplette Objekt als Vorlage für weitere Instanziierung zu verwenden.

Abbildung 5: Hierarchy im Editor

2.3 Editor

Im Abschlussprojekt wurde viel im Editor erstellt. Durch die 3D-Eigenschafft des Spieles, musste die Objekte im Editor zusammengebaut werden. Somit wurden das Spieler-Fahrzeug, Gegner-Fahrzeug, die Straße, der Gehweg, die Skybox, die Physikalischen Objekte und das Licht im Editor erstellt und hinzugefügt. Das einzige, außer der neue Instanziierungen der Gegner-Fahrzeuge, wurde im Code die Kamera gelöst. Da diese immer wieder Probleme im Editor hervorrufen.

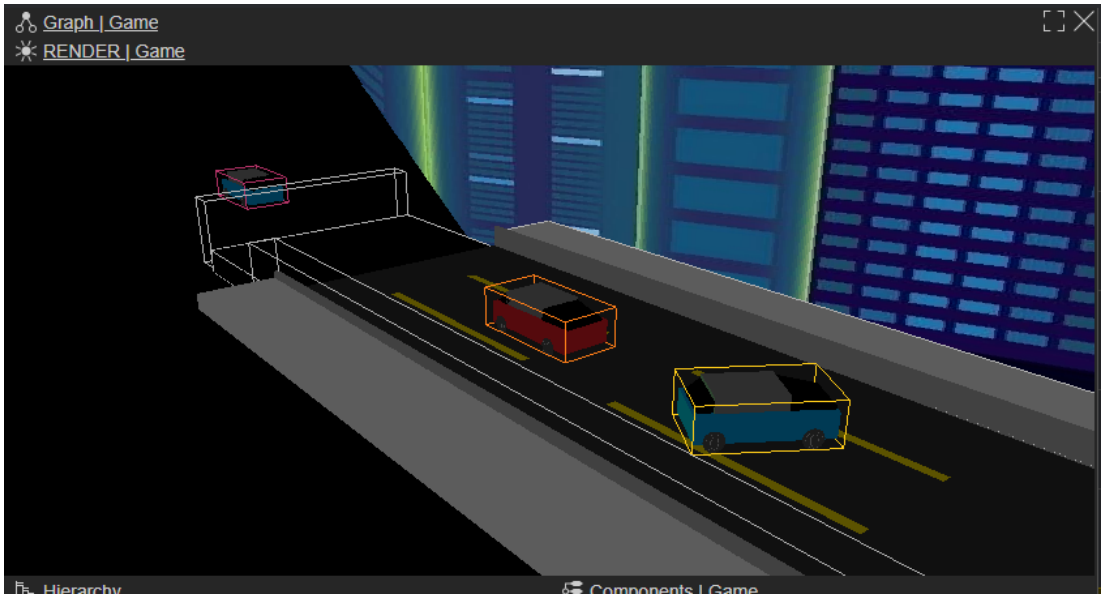

Abbildung 6: Editor Graph

2.4 Scriptcomponents

Es wurden insgesamt drei individuelle Scriptcomponents erstellt. „RandomEnemySpawnScript“ ist für die zufällige Positionierung der Gegnerischen Fahrzeuge. Dieses Script wird nach dem Aufbau des Fahrzeuges an dieses angefügt und danach sofort ausgeführt. Die Positionen unterscheiden sich in einer von den drei möglichen Fahrbahnen.

„EnemyCollisionDetect“ und „PlayerCollisionDetect“ sind für die Registrierung der Kollisionen der Objekte mit anderen physikalischen Objekten verantwortlich. Wenn hierbei eine Berührung registriert wird, werden individuelle Events abgefeuert. Im Falle des Gegners wird dieser nach Berührung gelöscht und im Falle des Spielers ein Leben abgezogen.

2.5 Extend

Es werden f.Node von FudgeCore verwendet, um auf die Eigenschaften des eigenen Nodes im Objekt zugreifen zu können. Des Weiteren wurde eine weitere Abstraktionsebene „AllEntity“ hinzugefügt. Diese beinhaltet das Leben des Spielers und des Gegners und bedient sich auf bei f.Node.

2.6 Sound

Geräusche werden in diesem Abschlussprojekt verwendet, um den Spieler auf dem Audio-Weg den aktuellen Status des Spiels zu kommunizieren.

Der Start des Spiels wird durch einen startenden Motor suggeriert. Der normale Ablauf wird durch ein gleichmäßiges Fahr-Geräusch gezeigt und bei einem Crash kann dieser Zustand durch einen Unfall-Sound festgestellt werden.

Dabei werden die Sounds jeweils auf dem Objekt-Node abgespielt, welcher gerade am meisten Sinn ergibt.

2.7 VUI

In der Benutzeroberfläche, während der Laufzeit des Spiels, kann der Spieler seine aktuelle Punktzahl und die verbleibenden Leben herausfinden.

Bevor das Spiel startet, wird der Spieler durch das Spiel gebeten, die Leertaste für das starten des Spiels zu triggern.

Wenn alle Leben aufgebraucht wurden, wird ein rudimentärer Anzeige erstellt und angezeigt, welche den Highscore und eine Status-Nachricht beinhalten (zu sehen in Abbildung 7).

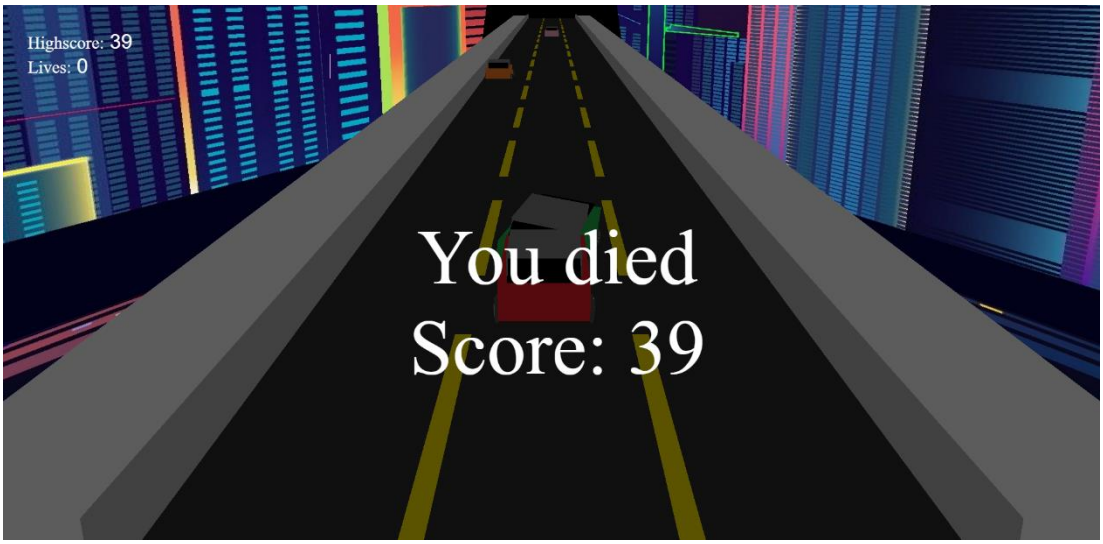

Abbildung 7: Death-Screen

2.8 Event-System

Es wurden Custom-Events verwendet, um die Kollisionen an den Graphen durchzugeben und Gegnerische-Fahrzeuge bei Berührung löschen zu las-

sen. Auch der Main wird durch ein Event mitgeteilt, ob der Spieler noch genug leben besitzt oder doch der „Death-Screen“ angezeigt werden muss.

2.9 External Data

Es werden dem Spieler Optionen überlassen, das Spiel nach eigenen Vorlieben zu gestalten und somit zu personalisieren. Folgende Einstellungsmöglichkeiten werden geboten:

- „enemy_speed“ beinhaltet die Geschwindigkeit der gegnerischen Fahrzeuge, mit welcher diese sich auf der Fahrbahn bewegen. In diesem Fall auf den Spieler hinzu.
- Mit den Einstellungen von „acceleration_left“ und „acceleration_right“ ist es möglich das eigene Fahrzeug auf die eigene Spielweise anzupassen. Es wird die Geschwindigkeit der Lenkung angepasst, in dem Fall die Schnelligkeit des Spurwechsels.
- „player_lives“ gibt die verfügbaren Leben des Spielers an.
- Mit „spawn_interval“ hat der Spieler die Möglichkeit den Spawn zu beeinflussen und somit das Spiel schwerer oder leichter zu gestalten. Eine niedrigere Nummer bedeutet schnellere Spawns der Gegner und eine hohe einen langsameren Spawn.

2.10 Light

Es wird Licht verwendet, um die Scheinwerfer des Spielerfahrzeugs zu emulieren, dabei wurde der SPOT-Type verwendet. Die gegnerischen Fahrzeuge werden vom Scheinwerfer-Licht des Spielers angestrahlt und reflektieren es, somit kann der Spieler die gegnerischen Fahrzeuge besser erkennen und potenziell ausweichen. Ansonsten wird das Ambient-Light für die Umgebung verwendet, so gesagt für die "Lichtverschmutzung" der Außenwelt.

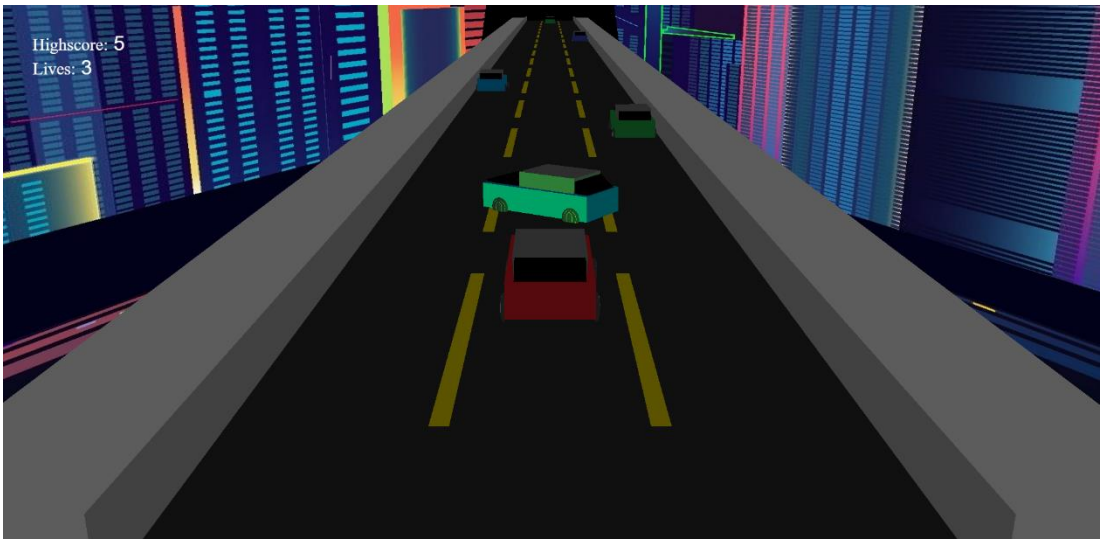

Abbildung 8: Gegner wird durch den Scheinwerfer des Spielers angestrahlt

Leider war es nicht möglich die Straße, also den Sprite, anzuleuchten. Hierbei habe ich mir Hilfe geholt, aber keine gut Umsetzbare Antwort gefunden. So müsste der Asphalt, also auch der Sprite, in verschiedene Objekte aufgeteilt werden, damit das Licht diese anstrahlen kann. Der Aufwand konnte zeitlich nicht mehr aufgebracht werden.

2.11 Physics

Es wird Physik in Form von RigidBodies verwendet. Jedes Objekt besitzt einen eigenen RigidBody. Dabei sind der Boden und die Wände statisch, damit diese als Abgrenzungen des Spieles dienen. Das Spieler-Fahrzeug ist als Kinetisches spielbares Objekt gesetzt. Die Gegnerischen-Fahrzeuge besitzen eine Dynamischen Body, damit diese auch von Unfällen beeinflusst werden können.

Hinzu werden die Gegner mithilfe von Force auf Ihrer Bahn bewegt. Wenn die Gegner den letzten Checkpoint, statischer RigidBody, berühren werden die Instanzen aus dem Spiel entfernt.

2.12 Animation

Es werden in diesem Abschlussprojekt Animationen in Form von Sprites verwendet. Die Straße wird mithilfe der eigens erstellten Sprite animiert und so auch die Endlosigkeit simuliert.

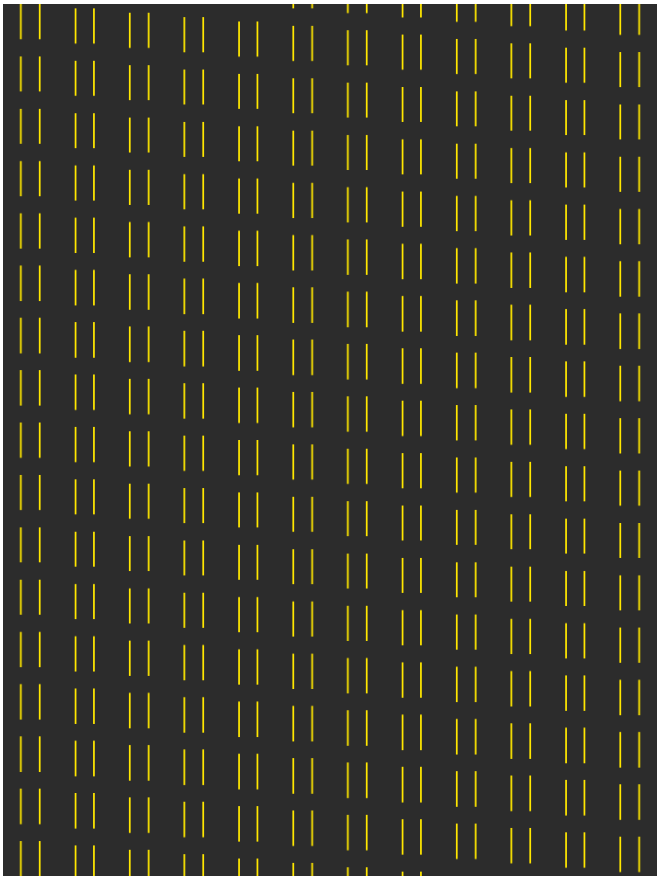

Abbildung 9: Erstelltes Sprite