Design document

# Racing game simulator

* Race game simulatie
* Parser: XML (Auto specificaties)
* Parser: image parser (Track-schets naar een logistische track)
* Neural network & reinforcement learning

# Algemene uitleg:

Ons project simuleert een race game, waarbij we een neural network trainen om auto’s zo snel mogelijk over een racebaan te kunnen laten rijden. We werken met een soort puntensysteem: Als de auto’s een goede tijd halen krijgen ze punten en anders verliezen ze punten. Hierdoor passen de auto’s zich aan om betere tijden te halen. Als de auto’s tegen een van de muren van de racetrack botsen, dan stoppen ze en verliezen ze punten.

# Game:

Simpele race game simulatie

* Entity system
  + Auto’s
  + Solid walls
* Physics met fixed timestep & velvet velocity integration (met constante acceleratie)
* SAT collisions
* Input map voor elke auto
  + Uitgebreide input (draaien, acceleration, remmen, ...)
* Autos bevatten zicht en collision sensors
* Autos kunnen hun data outputten (voor het neural network)

Easter egg: Tokyo drift muziek

# XML-Parser:

Specificaties van de soort auto uit een xml bestand halen.

* Versnelling (m/s^2)
* Max snelheid
* Massa (kg)
* Remkracht (m/s^2)
* Breedte (m)
* Lengte (m)
* Kleur (#HEX)

# Image-Parser:

Mogelijkheid om te tekenen in de game. De afbeelding wordt opgeslagen en geparsed.

Elke pixel wordt bekeken en adhv de kleur wordt het type entity bepaald.

* Zwart: weg
* Rood: startpunt

# Neural network & reinforcement learning:

* Ai die een track aflegt om te leren

Krijgt negatieve of positieve punten gebaseerd op:

* Tijd om het track te voltooien
* Hoeveelste plaats hij is aangekomen
* Of hij is aangekomen (na de eerste + tijd)
* Hoe Vaak de ai tegen de kant is gebotst

We genereren een populatie van auto’s met elk hun eigen neuraal netwerk. Elke generatie bepalen we de fitness van elke auto (progress op de racetrack, lap time, snelheid) en bepalen we de top presterende auto’s.

Via een genetic algorithm (uniform cross-breeding) en random mutation bepalen we mbv de top presterende auto’s de children van de volgende generatie.

**(FEEDFORWARD NEURAL NETWORK)** De architectuur van het neural network bestaat uit 5 neuronen als input (5 raycasts die de afstand tussen de auto en de muren bepalen (misschien nog huidige speed en richting erbij)), hidden layers en 4 neuronen als output (versnelling, remkracht, links sturen, rechts sturen).

* **Fitness functie**: (NEAT?)

**DOEL**: score toekenen aan de neurale networken binnen een generatie

**WERKING:**

Dit wordt gedaan door een score toe te kennen aan het neurale netwerk op basis van de seconden die nodig waren om te crashen of een ronde af te leggen, en vervolgens een straf van een aantal seconden toe te voegen voor elke afstand kort voor de finishlijn

* **Architecture feedforward neural network:** (perceptron?) (genoom van elke auto)

**DOEL:** auto besturen (output layer) mbv inputs ivm car physics en car control (input layer)

**ARCHITECTURE:**

- input layer (#neuronen afhankelijk van aantal sensors, car physics/control)

- hidden layers (#neuronen experimenteel bepalen)

- output layer (1 neuron: waarde tussen [0,1] die aangeeft hoe zwaar de auto moet turnen (simple nu)

* **Genetic algorithm:** (mutatie en crossover)

**DOEL:** een nieuwe generatie netwerken bouwen die gebaseerd is op de beste netwerken van de vorige generatie

**WERKING:**

Mutatie wordt uitgevoerd door een willekeurige verandering aan te brengen in de bias/weights waarden om het netwerk te wijzigen en een nieuw netwerk te creëren. Dit wordt gecombineerd met crossover, een methode om twee netwerken te nemen en een derde te genereren door waarden van elk netwerk te combineren. (specifieke cross-over: single-point cross-over?)