Projektbeschreibung für das p5.js-Projekt von Mario

Projektübersicht

Das Projekt besteht aus mehreren HTML-Dateien, die verschiedene kreative Codings und Simulationen implementieren. Jede Datei enthält ein eigenständiges Beispiel, das die Verwendung von p5.js zur Erstellung visueller Effekte und interaktiver Animationen zeigt.

Struktur des Projekts

1. **1.1.html - 1.3.html**: Grundlegende Beispiele für p5.js
    - **1.1.html**: Zeichnen von Linien mit unterschiedlichen Farben und Dicken.
    - **1.2.html**: Erzeugung eines Musters aus Punkten.
    - **1.3.html**: Zeichnen von verschachtelten Rechtecken mit Zufallsversatz.
2. **2.1.html - 2.6.html**: Bilder und Bildmanipulationen
    - **2.1.html**: Invertieren der Farben eines Bildes.
    - **2.2.html**: Skalieren eines Bildes basierend auf dem Abstand des Mauszeigers.
    - **2.3.html**: Zufällige Versätze der Pixel eines Bildes.
    - **2.4.html**: Anwendung von Dithering auf ein Bild.
    - **2.5.html**: Sortieren der Bildpixel nach Helligkeit.
    - **2.6.html**: Kombinieren von zwei Bildern mit unterschiedlichen Blendmodi.
3. **3.1.html - 3.3.html**: Komplexere Animationen und Simulationen
    - **3.1.html**: Zufällige Bewegung von Punkten auf der Leinwand.
    - **3.2.html**: Zeichnen von Kurven basierend auf der Mausbewegung.
    - **3.3.html**: Schwarmverhalten von Punkten.
4. **4.1.html - 4.3.html**: L-Systeme
    - **4.1.html**: Ein L-System zur Erzeugung von Pflanzenstrukturen.
    - **4.2.html**: Ein einfaches Fraktal-L-System.
    - **4.3.html**: Ein L-System zur Erzeugung von fraktalen Bäumen.
5. **5.1.html**: Markov-Ketten
    - Ein Textgenerator basierend auf Markov-Ketten, der zufälligen Text erzeugt.
6. **6.1.html - 6.3.html**: Game of Life
    - **6.1.html**: Standard-Game of Life Simulation.
    - **6.2.html**: Game of Life mit farbigen Zellen, die die dominante Farbe ihrer Umgebung annehmen.
    - **6.3.html**: Erweiterte Version von 6.2 mit detaillierterer Darstellung der dominanten Farben.
7. **7.1.html**: Rennsimulation
    - Eine Simulation, bei der drei verschiedene Algorithmen auf einer zufällig generierten Karte mit Hindernissen gegeneinander antreten. Die Algorithmen sollen so gestaltet sein, dass das Rennen spannend und ausgeglichen bleibt. Ein Algorithmus wird die beste Lösung anzeigen und die 2 Anderen suchen den Weg zum Ziel.

Beschreibung der drei Algorithmen in der Rennsimulation

1. **Algorithmus 1, Rot (Random Movement with Backtracking)**:
    - Der Agent bewegt sich zufällig in eine der vier möglichen Richtungen (oben, unten, links, rechts).
    - Vermeidet Hindernisse und versucht, nicht auf bereits besuchte Positionen zurückzukehren.
2. **Algorithmus 2, Grün (Greedy Best-First Search)**:
    - Der Agent sucht den kürzesten Weg zum Ziel basierend auf der Manhattan-Distanz.
    - Nutzt eine Prioritätswarteschlange, um die Position mit dem geringsten Heuristikwert auszuwählen.
    - Aktualisiert die beste Route kontinuierlich, um das Ziel schnellstmöglich zu erreichen.
3. **Algorithmus 3, Blau (Depth-First Search)**:
    - Der Agent durchläuft die Karte, indem er tief in einen Pfad vordringt, bevor er zu vorherigen Verzweigungen zurückkehrt.
    - Nutzt einen Stapel (Stack), um die zu besuchenden Positionen zu verfolgen.
    - Der Agent wählt zufällige Richtungen aus, solange er Hindernissen ausweicht.