Threads und State: Wimmelnde Roboter

In dieser Übung wollen wir ein kleines Spiel bauen. Ähnlich wie bei John Conway's "Game of life" gibt es in diesem keine echten Mitspieler. Stattdessen simulieren wir Spieler indem wir in separaten Threads zufällige Kommandos abfeuern, die den Zustand des Spiels verändern.

Hierfür existiert bereits ein vorbereitetes Projekt:

https://github.com/9elements/crosscard-tag4

1. Position modellieren

Als erstes macht es Sinn einen Typen für die Position eines Spielers auf dem Spielfeld anzulegen. Da der Buffer unseres virtuellen Displays eindimensional ist, ist es zudem ratsam eine Methode anzulegen, die die zweidimensionale Position in einen eindimensionalen Index umwandelt.

2. State modellieren

In diesem Schritt soll der State des Spiels modeliert werden. Der State muss eine Menge von Spielern halten und in der Lage sein diese eindeutig zu identifizieren. Ein Spieler besteht aus einer Position und einer Farbe (`u32`).

3. Messages

Unser virtuelles Display im Hauptthread soll über einen channel Nachrichten empfangen. Diese Nachrichten sollen die verschiedenen Spieleraktionen abbilden:

- Nach oben laufen
- Nach unten laufen
- Nach links laufen
- Nach rechts laufen

Diese Nachrichten sollten den State des Spiels und somit auch die Anzeige verändern. Dazu muss jede Nachricht den Absender eindeutig identifizieren.

4. Zufällige Aktionen

Jetzt muss eine Methode erstellt werden, die zufällig eine `GameAction` generiert. Es wäre gut, wenn die generierung nicht jedes Mal einen neuen `ThreadRng` oder ähnliches erstellen würde und man stattdessen der Methode diesen oder ähnliche übergeben könnte.

5. Spielerthreads spawnen

Um einen Spieler zu erstellen sollte ein Thread gestartet werden, der zufällig Nachrichten für das Spiel generiert und über den oben erstellten channel an den Hauptthread schickt. Zusätzlich sollte es möglich sein, einen Spieler zu "töten" bzw. ihm eine Nachricht zu schicken, die ihm mitteilt, dass er sich terminieren soll.

Wie Farbe und initiale Position des Spielers entstehen könnt ihr selbst entscheiden. Diese Werte können zum einen bei der Erstellung eines Spielers übergeben oder zufällig generiert werden.

6. Anzeige

Nach dieser ganzen Arbeit müssen wir noch eine Hand voll Spieler erschaffen, im Game State speichern und im display-loop anzeigen.

Die Farben und initialen Positionen der Spieler können zufällig mit der Funktion `rand::random()` generiert werden.

7. Kollision

Wenn zwei Spieler auf der selben Position sind, soll dies zu einer "Kollision" führen. In diesem Fall sollen beide Spieler aus dem Spiel ausscheiden.

8. Ende

Das Spiel soll enden, wenn nur noch ein Spieler übrig ist. Das Fenster sollte dann gänzlich in der Farbe des Gewinners erscheinen.