Portfolio Prüfung

30. November 2024

Inhaltsverzeichnis

1	Mul	ti-Thre	aded-	Mond	olith														2
	1.1	Aufba	u																2
		Erklär																	
		1.2.1	Firef	ly															2
		1.2.2	Firef	lyGri	d .	 													3
		1.2.3	Main																3
	1.3	Konze	pt				•				•				•				4
2	Vert	teiltes S	Systen	1															5

1 Multi-Threaded-Monolith

1.1 Aufbau

Die Aufgabe bestand darin, eine Simulation von synchronisierten Glühwürmchen zu implementieren, die gemäß dem Kuramoto-Modell interagieren. Die Simulation nutzt eine Torus-Anordnung, in der jedes Glühwürmchen als Thread realisiert ist. Es gibt drei zentrale Klassen:

- Firefly: Stellt ein Glühwürmchen dar, das seine Phase durch Interaktion mit Nachbarn gemäß dem Kuramoto-Modell aktualisiert.
- **FireflyGrid**: Organisiert die Glühwürmchen in einer Torus-Anordnung und definiert Nachbarschaften.
- Main: Verantwortlich für die Benutzeroberfläche, die Glühwürmchen als Bilder visualisiert. (jeweils 1 jpeg für status blinken und status nicht blinken).

1.2 Erklärung der Klassen

1.2.1 Firefly

Die Klasse Firefly implementiert das Verhalten eines einzelnen Glühwürmchens.

- \bullet Jedes Glühwürmchen besitzt eine Position (x, y), eine Eigenfrequenz und eine Phase, die seine Helligkeit bestimmt.
- Es wird ein Thread erstellt, der kontinuierlich die Phase des Glühwürmchens aktualisiert.
- Die Methode updatePhase() berücksichtigt die Phasen der Nachbarn und passt die Phase des aktuellen Glühwürmchens gemäß dem Kuramoto-Modell an:

Neue Phase = Eigenfrequenz + $K \cdot$ Einfluss der Nachbarn

• Die Methode getBrightness() gibt die relative Helligkeit des Glühwürmchens zurück, basierend auf seiner Phase.



(a) Helles Glühwürmchen in der aktiven Phase.



(b) Dunkles Glühwürmchen in der inaktiven Phase.

Abbildung 1: Darstellung eines Glühwürmchens in zwei verschiedenen Phasen.

1.2.2 FireflyGrid

Die Klasse FireflyGrid erstellt ein Gitter von Glühwürmchen und definiert ihre Nachbarschaften.

- Jedes Glühwürmchen wird in einer Torus-Struktur angeordnet, d.h., die Ränder des Gitters sind miteinander verbunden.
- Die Methode initializeGrid() erzeugt die Glühwürmchen mit voreingestellten Phasen.
- Die Methode setNeighbors() definiert die Nachbarn jedes Glühwürmchens, wobei die Modulo-Arithmetik die Torus-Struktur gewährleistet.

1.2.3 Main

Die Main-Klasse ist für die Benutzeroberfläche und die Darstellung der Glühwürmchen zuständig.

- Mithilfe von Swing wird ein Fenster erstellt, das die Glühwürmchen in einem Gitter visualisiert.
- Die Methode paintComponent(Graphics g) zeichnet jedes Glühwürmchen als farbiges Rechteck, wobei das Bild von der Helligkeit abhängt.
- Threads werden gestartet, um die Phasen der Glühwürmchen parallel zu aktualisieren.

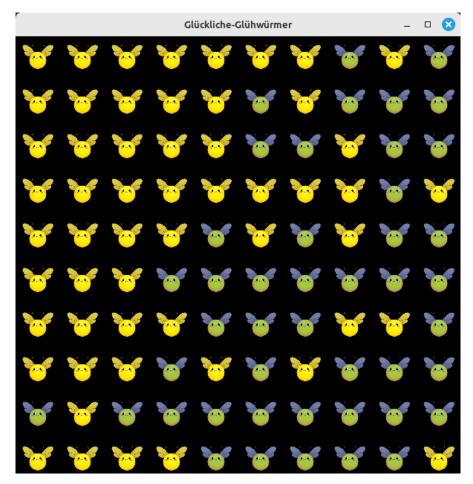


Abbildung 2: Torus-Anordnung der Glühwürmchen (10x10-Gitter, zyklische Verbindung).

1.3 Konzept

Die Eigenfrequenz (ω) eines Glühwürmchens beschreibt seinen natürlichen Rhythmus, in dem es zwischen hellen und dunklen Phasen wechselt. Die Phase (ϕ) gibt den aktuellen Zustand innerhalb dieses Zyklus an und bestimmt, wann das Glühwürmchen aufleuchtet. Durch die Interaktion mit Nachbarn, beeinflusst durch die Kopplungsstärke K, passt ein Glühwürmchen seine Phase an, sodass es sich nach und nach synchronisiert. Dabei nähert sich die Phase der Glühwürmchen einander an, bis sie gleichzeitig aufleuchten. Zum Start des Programm hat jedes Glühwürmchen eine Zufällige Phase, die sich mit der Zeit angleichen.

Simon Szulik MtrNr.: 1474315	SA4E	30. November 2024
s4siszul@uni-trier.de	\ddot{U} bungsblatt 01	50. November 2024

2 Verteiltes System