**Relatório Simulador de partículas**

**Elian Silva**

**Isaak Henrique**

**Iago Alves**

**Introdução**

O multithreading é uma técnica fundamental no desenvolvimento de jogos modernos, permitindo que várias tarefas sejam executadas simultaneamente para melhorar o desempenho e a fluidez da execução. Em aplicações como simulações de partículas, o processamento paralelo pode reduzir significativamente o tempo de cálculo e melhorar a responsividade do sistema. Este relatório explora a implementação de multithreading em uma simulação de partículas, destacando as ferramentas utilizadas, a abordagem adotada e os benefícios observados.

**Desenvolvimento**

A implementação foi realizada em Java, utilizando a biblioteca Swing para renderização gráfica e manipulando threads diretamente para distribuir a carga de trabalho. Cada partícula é representada por um objeto da classe Particle, contendo informações sobre sua posição e cor. A simulação ocorre em uma instância da classe ParticleSimulation, que atualiza e redesenha as partículas periodicamente.

Para implementar o multithreading, as partículas foram divididas entre diferentes threads com base no número de processadores disponíveis. Cada thread processa um subconjunto de partículas, atualizando suas posições de maneira paralela. O uso de Thread.join() garante que as threads concluam suas execuções antes da próxima atualização da tela.

**Resultados**

A comparação entre a abordagem single-thread e a multi-thread foi realizada medindo o tempo de execução de cada ciclo de atualização. Os resultados indicaram uma redução significativa no tempo de processamento quando o multithreading foi empregado, especialmente à medida que o número de partículas aumentava. Para sistemas com múltiplos núcleos, o ganho de desempenho foi mais evidente, demonstrando a eficácia do paralelismo na simulação.

**Conclusão**

A implementação do multithreading na simulação de partículas proporcionou melhorias expressivas em desempenho, reduzindo o tempo de execução e aumentando a fluidez da renderização. No entanto, desafios como sincronização e divisão eficiente da carga de trabalho para cada thread foram encontrados, destacando a importância de uma abordagem bem planejada. O estudo reforça a relevância do multithreading no desenvolvimento de jogos e simulações interativas.