UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

FERNANDA DA SILVA GOMES

Estudo de Otimização de Simulação de Cabelo em Tempo Real Baseado em Física em Unity

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Prof. Dr. Dennis Giovani Balreira

Porto Alegre
2024

“And I knew exactly what to do,
but in a much more real sense I had no idea what to do.”
— MICHAEL SCOTT

AGRADECIMENTOS

Gostaria de agradecer à minha família que sempre me proporcionou os meios e incentivou a estudar, da qual o suporte e paciência foi essencial nesta longa jornada que me trouxe até aqui.

Gostaria de agradecer também aos amigos e colegas que me acompanharam nessa saga, trazendo a leveza e a companhia necessárias para seguir em frente.

Agradeço também aos professores que me apresentaram diversos novos conhecimentos e interesses dentro da computação.

E por fim, gostaria de agradecer ao meu orientador, Dennis Giovani Balreira, que foi muito paciente e compreensivo nessa jornada, trazendo a motivação que as vezes me escapava.

RESUMO

Simulações realistas de cabelo em computação gráfica desempenham um papel crucial na criação de personagens em ambientes imersivos, como jogos digitais, filmes e experiências em realidade virtual ou aumentada. No entanto, realizar essas simulações em tempo real apresenta desafios significativos, especialmente considerando as limitações de hardware e as interações com o ambiente virtual, como colisões e gravidade. Este estudo tem como objetivo explorar novas técnicas que busquem um equilíbrio adequado entre desempenho e qualidade na simulação de cabelo em tempo real dentro de ambientes virtuais baseados em física. Guiados por esse objetivo, implementamos um modelo utilizando o motor de jogo Unity e realizamos otimizações utilizando uma variação do método de agrupamento com interpolação. Comparamos os resultados obtidos com o método sem variação e obtivemos uma melhora na qualidade visual da dinâmica dos fios. Espera-se que os resultados obtidos contribuam para a criação de ambientes virtuais mais imersivos e melhorem o desempenho geral da simulação.

Palavras-chave: Modelagem de cabelo. Sistema massa-mola. Simulação baseada em física. Simulação em tempo real.

Study on Optimizing Physics-Based Real-Time Hair Simulation in Unity

ABSTRACT

Realistic hair simulations in computer graphics play a crucial role in creating characters in immersive environments, such as digital games, movies, and virtual or augmented reality experiences. However, achieving real-time performance for these simulations presents significant challenges, especially considering hardware limitations and interactions with the virtual environment, such as collisions and gravity. This study aims to explore new techniques that seek an adequate balance between performance and quality in real-time hair simulation within physics-based virtual environments. Guided by this objective, we implemented a model using the Unity game engine and performed optimizations using a variation of the interpolation-based clustering method. We compared the obtained results with the method without variation and achieved an improvement in the visual quality of the hair dynamics. The results obtained are expected to contribute to the creation of more immersive virtual environments and improve the overall simulation performance.

Keywords: Hair Modeling. Mass-spring system. Physics-based simulation. Real-time simulation.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CPU	Central Processing Unit
DER	Discrete Elastic Rods
FPS	Frame Per Second
GPGPU	General Purpose Graphics Processing Unit
GPU	Graphics Processing Unit
HLSL	High Level Shading Language

SUMÁRIO

1 INTRODUÇÃO

O cabelo é uma característica visual importante dos humanos e desempenha um papel significativo em sua aparência geral (FINK et al., 2016). O estudo da simulação da dinâmica do cabelo é essencial para a criação de cabelos realistas em personagens de jogos, animações, filmes e ferramentas de realidade aumentada. Uma simulação bem feita, além de poder levar uma experiência mais imersiva para o público, pode também ser utilizada como ferramenta em diversas áreas (SGOUROS, 2024).

Além disso, a simulação em tempo real é particularmente importante para ambientes interativos, como em jogos eletrônicos, por exemplo, onde a cada nova interação do usuário, é necessário calcular um novo estado único. Além de melhorar o realismo e imersão, a jogabilidade pode ser impactada positivamente por um número maior de possíveis interações e personalizações. Pode ser interessante, por exemplo, uma mecânica de crescimento de cabelo ao longo de uma campanha de jogo ou o cabelo de um personagem ser afetado por alguma situação específica.

1.1 Motivação

A evolução da simulação de cabelo nas mídias abre um leque de possibilidades para a criação de personagens e estilos visuais, permitindo a representação de diversas visões e estilos, expandindo a criatividade e a representatividade nas mídias.

Além disso, a simulação de cabelo, combinada com ferramentas de realidade aumentada, pode, por exemplo, revolucionar o treinamento de cabeleireiros. Através da simulação, podendo experimentar combinações de curvatura, densidade e tamanho, pode ser possível, em algumas situações, reduzir a necessidade de materiais físicos, como manequins e cabelo, diminuindo assim os custos e facilitando o aprendizado. É uma tecnologia com potencial de democratizar o acesso a educação para essa profissão.

Simular cabelos de forma realista tem sido um desafio constante na área de computação gráfica, mais ainda quando essa simulação é feita em tempo real. Visto que a tecnologia ainda não consegue lidar com a quantidade de processamento necessária para simular todas as interações entre fios de cabelo em tempo real. Diversos estudos vem sendo feitos ao longo dos anos (ROSENBLUM; CARLSON; III, 1991; ANJYO; USAMI; KURIHARA, 1992; BRIDSON; MARINO; FEDKIW, 2005; BERGOU et al., 2008; CHAI; ZHENG; ZHOU, 2014; DAVIET, 2020; DAVIET, 2023), e cada ano que

passa novas abordagens surgem com cabelos cada vez mais realistas e com processamento mais eficiente (Capítulo 3).

Além disso, novas ferramentas mais acessíveis vem surgindo, tal qual o simulador de cabelos desenvolvido pela Unity, que permite a simulação de altas quantidades de fios de cabelo, com uma ótima performance e muitas possibilidades de configurações. Porém, esse simulador não é muito preciso nas colisões, o que deixa espaço para novas abordagens dentro da própria Unity.

1.2 Objetivos

Este trabalho busca desenvolver parcialmente um simulador de cabelo de qualidade, com foco na dinâmica, utilizando as ferramentas disponíveis em um motor de jogo. O objetivo é investigar e avaliar métodos de otimização para esse simulador que possam potencialmente ser usados com outros modelos semelhantes. Com o propósito de alcançar esse objetivo, este trabalho irá elencar métodos existentes, implementar um ambiente de testes que possibilite experimentos com diferentes métodos e avaliar os resultados obtidos.

1.3 Organização

No capítulo seguinte, abordaremos os conceitos básicos necessários para a construção deste trabalho. Falaremos sobre os conceitos matemáticos que formam a base da computação gráfica, seguindo de explicações específicas sobre computação gráfica, informações relevantes sobre o motor de jogo Unity, e, por fim, os conceitos da física utilizados.

No Capítulo 3, apresentamos os trabalhos relacionados ao estudo de simulações de cabelo baseadas em física e metodologias interessantes existentes.

Já no Capítulo 4, descrevemos os passos realizados na concepção e construção do modelo de simulação utilizado neste trabalho.

E por fim, no Capítulo 5, validamos e discutimos os resultados obtidos, e no Capítulo 6, apresentamos limitações encontradas durante a realização do trabalho, bem como possibilidades de aprimoramento para futuras investigações.

2 CONCEITOS BÁSICOS

Este capítulo apresenta os conceitos e fundamentos utilizados no desenvolvimento deste trabalho. As seções 2.1 e 2.2 utilizam como base os conhecimentos do livro (SHIRLEY; ASHIKHMIN; MARSCHNER, 2009).

2.1 Conceitos Matemáticos

Nessa seção iremos abordar os conceitos matemáticos necessários ao estudo da computação gráfica. A partir dos conceitos mais básicos de geometria, como vetores, pontos, linhas, planos e suas operações, abordaremos transformações geométricas e sistemas de coordenadas.

2.1.1 Vetores

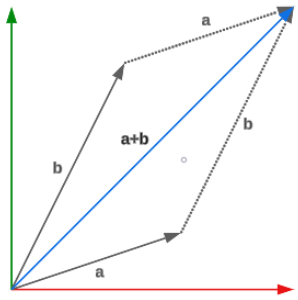
Entidades geométricas com comprimento e sentido, geralmente representadas graficamente por uma seta. Dois vetores são iguais se eles possuem o mesmo comprimento e sentido, mesmo quando pensamos que podem estar localizados em lugares diferentes.

- *Vetor Zero*: Vetor de comprimento igual a zero;
- *Vetor Unitário*: Vetor de comprimento igual a um;
- *Vetores Ortogonais*: Dois vetores são ortogonais se o seu produto escalar for zero, o que significa que o ângulo entre eles é de 90 graus ou um deles é zero;
- *Base*: Dois vetores linearmente independentes (não-zero e não-paralelos) formam uma base;
- *Base Ortonormal*: Uma base é ortonormal se os vetores forem unitários e ortogonais entre si;
- *Soma*: A soma de dois vetores é encontrada posicionando o fim de um vetor com o início de outro em qualquer ordem (Figura 2.1);
- *Subtração*: É uma soma onde o segundo vetor tem sua direção invertida (Figura 2.2);
- *Produto Escalar*: Valor escalar relacionado ao comprimento dos vetores e o ângulo entre eles, um uso comum é computar o cosseno desse ângulo. Pode também ser

usado para encontrar a projeção de um vetor no outro (Figura 2.3);

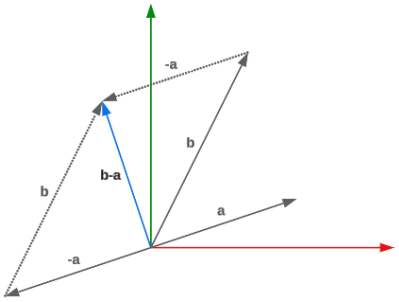
- *Produto Cruzado*: Geralmente é usado apenas para vetores tridimensionais. O produto cruzado de dois vetores retorna o vetor perpendicular aos dois vetores usados de argumentos da operação, o comprimento desse vetor resultante é relacionado ao seno do ângulo (Figura 2.4);

Figura 2.1 – Soma dos vetores a e b.



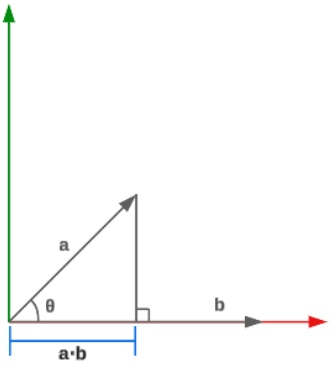
Fonte: O Autor

Figura 2.2 – Subtração dos vetores b e a.



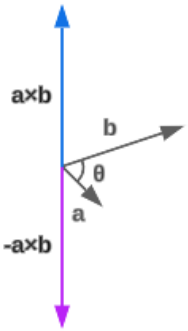
Fonte: O Autor

Figura 2.3 – Projeção do vetor a em b usando produto escalar.



Fonte: O Autor

Figura 2.4 – Produto cruzado de a e b.



Fonte: O Autor

2.1.2 Pontos

Representam um deslocamento no espaço a partir da origem em um sistema de coordenadas.

2.1.3 Linhas

Conjunto contínuo de pontos colineares definidos por uma função.

2.1.4 Planos

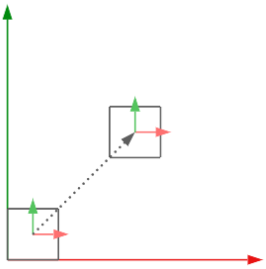
Conjunto contínuo de pontos definido por um ponto e vetor normal onde o vetor resultante de quaisquer dois pontos é ortogonal a normal.

2.1.5 Transformações Geométricas

As transformações geométricas são realizadas em relação a um sistema de coordenadas e podem ser representadas por matrizes.

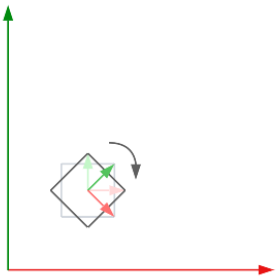
- Translação: Consiste em mover um objeto do a ao ponto b (Figura 2.5);
- Rotação: Consiste em rotacionar um objeto ao redor de um eixo específico (Figura 2.6);
- Escala: Consiste em alterar o tamanho de um objeto, podendo ser de forma uniforme (mesma alteração em todas as dimensões) ou não uniforme (alterações diferentes em cada dimensão) (Figura 2.7);

Figura 2.5 – Translação



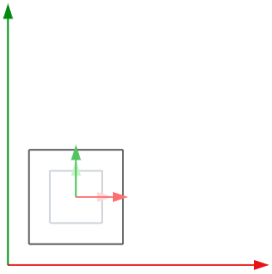
Fonte: O Autor

Figura 2.6 – Rotação



Fonte: O Autor

Figura 2.7 – Escala



Fonte: O Autor

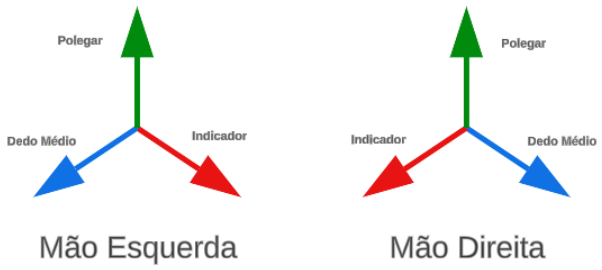
2.1.6 Sistemas de Coordenadas

Um sistema de coordenadas é composto por um ponto de origem e vetores base, geralmente ortonormais, no qual cada vetor define uma dimensão. Normalmente, uma cena virtual usa diversos sistemas de coordenadas, um deles, por exemplo, é o sistema de coordenadas local, que é único para cada objeto da cena, e é a partir de onde são definidos todos os vértices do objeto. Outro sistema de coordenadas é o global, onde os objetos são posicionados, rotacionados e escalados na cena a partir de transformações geométricas

aplicadas ao sistema de coordenadas local de cada objeto. Uma mudança de sistema de coordenadas nada mais é do que uma transformação geométrica.

Em sistemas de coordenadas 3D, podemos definir o terceiro eixo de duas formas, utilizando a regra da mão esquerda ou mão direita. Essas regras definem para qual lado do plano 2D o terceiro eixo está apontado e qual a orientação das rotações (Figura 2.8).

Figura 2.8 – Regra da mão direita e regra da mão esquerda.



Fonte: O Autor

2.2 Computação Gráfica

2.2.1 Modelagem e Animação

É a construção da representação de um objeto, de forma que o pipeline gráfico consiga processar e transformar em uma imagem na tela. Diferentes aplicações vão necessitar de diferentes representações de um mesmo objeto. Uma representação muito comum da geometria de objetos é a representação por malhas de triângulos com vértices compartilhados. Dentro da construção de um modelo podemos definir atributos além da geometria do objeto, como cor, textura, material e aplicar configurações usadas para animação como *rigging* e *skinning*. *Rigging* sendo a criação de um sistema de juntas que permitem que o objeto seja animado e *skinning* envolve conectar a malha de superfície a essas juntas, de forma que a malha deforme conforme o movimento das juntas.

2.2.2 Rasterização e Pipeline Gráfico

O processo de encontrar todos os *pixels* em uma imagem que são ocupados por uma primitiva geométrica (e.g. um triângulo) é chamado de rasterização. Esse método de renderização é bem eficiente e frequentemente usado em aplicações que precisam renderizar cenas em tempo real. A sequência de operações necessárias para transformar objetos de uma cena virtual 3D em *pixels* de uma imagem 2D, é conhecida como o pipeline gráfico. O pipeline gráfico pode ser descrito da seguinte forma:

- *Entrada*: A aplicação providencia a descrição da cena virtual e as representações dos objetos;
- *Processamento Geométrico*: Antes da primitiva poder ser rasterizada, os vértices precisam estar em coordenadas de tela. Cor ou quaisquer atributos que precisam ser interpolados têm de ser conhecidos. Preparar esses dados é função da etapa de processamento geométrico. Nesse estado os vértices de entrada são transformados pelas matrizes de *modeling*, *viewing* e *projection*, mapeando pontos no seu sistema de coordenadas original para o sistema de coordenadas da tela. Ao mesmo tempo, cores, normais e coordenadas de texturas são transformadas conforme necessário;
- *Rasterização*: Para cada primitiva, o rasterizador enumera os *pixels* que são cobertos pela primitiva e interpola os valores de quaisquer atributos conhecidos. O resultado são fragmentos com uma série de informações que definem um *pixel*;
- *Processamento de Fragmentos*: Após a rasterização, a partir dos fragmentos com seus atributos, mais operações são feitas para computar a cor e profundidade de cada fragmento, como avaliação de modelo de iluminação, texturas, definições de materiais, etc;
- *Saída*: As informações processadas de cada fragmento são armazenadas no *framebuffer*;

2.3 Unity

A Unity é um motor de jogo proprietário multiplataforma bem conhecido na indústria de jogos, amplamente utilizado nos mais diversos jogos e nas mais diversas escalas (TECHNOLOGIES, 2024a). Usamos como base de conhecimento desta seção o manual de usuário da Unity (TECHNOLOGIES, 2024b).

2.3.1 Sistema de Coordenadas

Este motor de jogo utiliza o sistema de coordenadas da mão esquerda e um sistema hierárquico de objetos, onde cada objeto tem seu próprio sistema de coordenadas local. Neste caso, o objeto é posicionado, rotacionado e escalado em relação ao seu objeto pai e qualquer uma dessas transformações são também refletidas nos objetos aninhados.

2.3.2 Pipeline de Execução

O pipeline de execução de *scripts* (TECHNOLOGIES, 2024g) têm três métodos importantes para este trabalho, sendo esses os métodos: *Start*, *FixedUpdate* e *Update* (existem muitas outras etapas, mas não tão relevantes para este protótipo).

- *Start*: Executa na etapa de inicialização que ocorre apenas uma vez quando um objeto é instanciado, geralmente usado para inicializar propriedades e buscar componentes e objetos na hierarquia;
- *FixedUpdate*: Executa no início de cada ciclo de atualização do motor de física, onde, geralmente, atualizações relacionadas aos componentes físicos são feitas, como por exemplo, aplicação de força ou movimento a um *rigidbody*. Tem a frequência fixa definida nas configurações do projeto, e pode acontecer mais ou menos de uma vez por intervalo de *frame*, dependendo da frequência de *frames* por segundo;
- *Update*: Executa uma vez a cada *frame* e é onde é executada a maior parte da lógica de jogo e as atualizações de objetos não afetados pelo motor de física;

2.3.3 Motor de Física

Um motor de física é um *software* de computador que oferece uma simulação aproximada de sistemas físicos, e é utilizada na simulação de ambientes virtuais. O motor de física (TECHNOLOGIES, 2024c) utilizado pela Unity é uma integração do Nvidia PhysX (NVIDIA, 2021).

2.3.4 Compute Shaders

Compute shaders são *shaders* que rodam na GPU, fora do pipeline de renderização. Eles podem ser usados para executar algoritmos de GPGPU altamente paralelizáveis, ou acelerar partes da renderização de jogos. Esses *shaders* são escritos em linguagem HLSL (TECHNOLOGIES, 2024e).

2.3.5 Parallel Jobs

Parallel jobs são tarefas independentes agendadas para rodar em paralelo nos *cores* disponíveis da CPU (TECHNOLOGIES, 2024h).

2.4 Física

Para definir alguns conceitos básicos necessários no entendimento deste trabalho, essa seção utiliza os conhecimentos do livro (SALES; MAIA, 2018) como base.

- *Velocidade*: Vetor de deslocamento em função do tempo;
- *Aceleração*: Vetor de variação da velocidade de um objeto em relação ao tempo;
- *Força*: Forças são interações entre corpos que podem provocar variações em sua velocidade;
- *Força Gravitacional*: Força fundamental de atração que age entre todos os objetos, proporcional às suas massas e inversamente proporcional ao quadrado da distância que separa seus centros de gravidade;
- *Constante Gravitacional*: Aceleração aplicada a objetos em queda livre na superfície da terra, usada para compor a Força Peso de objetos;
- *Fricção/Atrito*: Força que resiste ao movimento relativo de superfícies sólidas, camadas de fluido e elementos materiais que deslizam uns contra os outros. Um exemplo de atrito é a resistência do ar;
- *Força Elástica*: Quando um corpo é deformado, em um alongamento ou contração, a força contrária ao sentido da força que o deformou é denominada força elástica. Sistemas massa-mola-amortecedor se utilizam desse cálculo de força.

3 TRABALHOS RELACIONADOS

Simulações de cabelo realistas têm sido um campo de pesquisa importante há décadas (WARD et al., 2007), como mostrado pelos trabalhos de (ROSENBLUM; CARLSON; III, 1991) e (ANJYO; USAMI; KURIHARA, 1992). Diversas metodologias continuam sendo implementadas com o intuito de simular e renderizar cabelos tanto de forma dinâmica e interativa em tempo real, quanto de forma não interativa. Modelos dinâmicos para animar fios independentes têm sido estudados extensivamente, sendo esse tipo de modelo o com resultados mais fiéis a realidade, simulando os mais complexos comportamentos de cada fio (BERTAILS et al., 2006; BERGOU et al., 2008; SELLE; LENTINE; FEDKIW, 2008; CASATI; BERTAILS-DESCOUBES, 2013; DAVIET, 2023). Entretanto, estas técnicas são mais caras computacionalmente, o que dificulta o uso em simulações em tempo real em situações com restrições de *hardware*.

Cada modelo de simulação de cabelo tem um equilíbrio específico na relação entre realismo e eficiência, baseado no objetivo e caso de uso de cada um deles. Modelos usados em simulações em tempo real tem como objetivo a obtenção de efeitos visuais plausíveis dentro de um ambiente. Dessa forma, possuem limitação de tempo de processamento e necessitam de um certo nível de eficiência, enquanto que modelos sem restrições de tempo de processamento podem levar o tempo necessário para gerar simulações mais realistas.

Podemos separar uma simulação de cabelo em três tarefas distintas (HADAP; MAGNENAT-THALMANN, 2001):

- *Modelagem Geométrica*: Definição da geometria do fio.
- *Dinâmica*: Como é simulado o comportamento desse modelo.
- *Rendering*: Define a apresentação visual desse modelo, onde técnicas são utilizadas para gerar a composição de cores do cabelo.

Dentro da etapa de dinâmica, temos a dinâmica individual do fio e a dinâmica de interação com outros objetos. A dinâmica individual controla o comportamento do fio de forma isolada, como a flexão, torção, elasticidade e ondulação. Diversos modelos foram estudados ao longo dos anos, como por exemplo o modelo massa-mola (ROSENBLUM; CARLSON; III, 1991), onde os fios são definidos como um conjunto de partículas conectadas por molas, e que possui diversas variações de implementação.

Já a dinâmica da interação dos fios entre si e com o ambiente pode ser abordada sob a perspectiva de fios individuais ou tratando um conjunto de fios como um contínuo

(HADAP; MAGNENAT-THALMANN, 2001).

O modelo de cabelo como um contínuo considera o cabelo como uma entidade única, com comportamento semelhante a de um fluido, ignorando os detalhes do comportamento individual de cada fio. Esse modelo existe a partir da ideia de que, de forma geral, o comportamento de um conjunto de fios de cabelo é coerente e pode ser calculado como um único comportamento (HADAP; MAGNENAT-THALMANN, 2001). Apesar de dar bons resultados de modo mais geral e ser uma abordagem bem eficiente, a perda de detalhes, como as interações entre fios, causa uma piora na qualidade da simulação, visto que o atrito entre fios de um cabelo influencia no movimento e na tendencia de manter um formato específico (JIANG et al., 2020). Este trabalho também peca em situações de colisões e movimentações mais complexas. Alguns trabalhos usando essa abordagem podem ser encontrados em (HADAP; MAGNENAT-THALMANN, 2001; BANDO; CHEN; NISHITA, 2003; PETROVIC; HENNE; ANDERSON, 2005; MCADAMS et al., 2009; MÜLLER; KIM; CHENTANEZ, 2012; JIANG; GAST; TERAN, 2017).

Já na perspectiva de fios individuais, cada fio tem suas interações computadas de forma individual. Este é um tipo de modelo mais caro computacionalmente, visto que um cabelo tem uma grande quantidade de fios que estão em contato uns com os outros. Porém, em contrapartida, gera resultados mais fiéis à realidade. Várias implementações sugerem abordagens diferentes para esse aspecto da simulação (KAUFMAN et al., 2014; BERTAILS-DESCOUBES et al., 2011), como por exemplo, a utilização de forças e impulsos que são aplicadas ou removidas em situações específicas (JIMENEZ; LUCIANI, 1993; CHANG; JIN; YU, 2002; BRIDSON; MARINO; FEDKIW, 2005; SELLE; LENTINE; FEDKIW, 2008; IBEN et al., 2013).

Modelos que consideram fios individualmente geralmente possuem um processamento mais lento. Portanto, uma abordagem para esse problema são os modelos simplificados. Um desses modelos simplificados mais comuns é o modelo de cabelo em mechas (KOH; HUANG, 2000; KOH; HUANG, 2001; GUANG; ZHIYONG, 2002; LIANG; HUANG, 2003; TAŞKIRAN; GÜDÜKBAY, 2005; SUGISAKI et al., 2005), que define uma mecha como um conjunto de fios simulados em conjunto, como se fosse um único fio. Esse modelo diminui a quantidade de processamento consideravelmente e, portanto, é bastante eficiente. Porém, como é uma simplificação, ele acaba perdendo muitos detalhes e diminui o nível de fidelidade. Variações desse modelo estão presentes nos modelos de interpolação (CHAI; ZHENG; ZHOU, 2014), onde os fios são interpolados considerando um conjunto de fios guia.

Como podemos concluir, simular cabelo em tempo real com alto nível de detalhamento não é uma tarefa fácil. Os métodos mais eficientes simplificam tanto a ponto de diminuir a qualidade da simulação, e os métodos mais realistas têm um processamento muito lento para serem usados em tempo real. Porém, atualmente, tem surgido modelos de simulação com alto nível de fidelidade e processamento rápido o suficiente para serem usados em tempo real. Um exemplo é o modelo proposto em (JIANG et al., 2020), com uma abordagem que captura de forma eficiente a mecânica de inextensibilidade, flexão e torção do fio, apresentando ao mesmo tempo a aderência/repulsão e efeitos de colisão detalhados em tempo real.

Figura 3.1 – Alguns resultados das simulações do modelo (JIANG et al., 2020).

Fonte: (JIANG et al., 2020)

Outro modelo recente é o (DAVIET, 2023), que apesar de ter o foco maior em

realismo, propõe um simulador de *Discrete Elastic Rods* (DER) (BERGOU et al., 2008) utilizando o modelo de contato *Coulomb friction* (DAVIET; BERTAILS-DESCOUBES; BOISSIEUX, 2011; DAVIET, 2020) paralelizado em GPU que é capaz de simular milhares de hastes elásticas em tempo real e com alta resolução (Figura 3.2).

Figura 3.2 – Alguns resultados das simulações do modelo (DAVIET, 2023).

Fonte: (DAVIET, 2023)

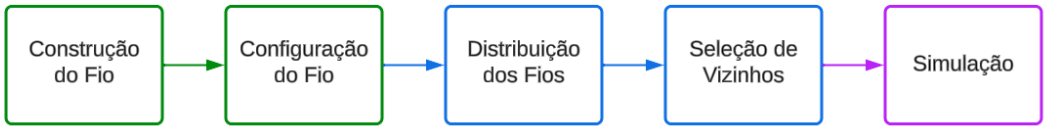
3.1 Aprendizados

A partir da nossa análise da literatura, até onde nos foi possível investigar, identificamos que trabalhos mais recentes na área de simulação de cabelo tendem a ser mais complexos e não costumam utilizar técnicas de agrupamento em mechas em conjunto com outras técnicas mais avançadas e recentes de simulação e otimização. O foco principal desses trabalhos reside na simulação individual de cada fio de cabelo, buscando gerar resultados mais realistas. Com base nisso, detectamos que uma boa abordagem para ser trabalhada seria uma evolução do modelo de agrupamentos que pudesse ser utilizada em conjunto com simulações de fios individuais.

4 METODOLOGIA

Este capítulo apresenta a metodologia usada na simulação de fios de cabelo, começando pelas etapas de construção e configuração dos fios, que são estáticas, seguindo pelas etapas que ocorrem no início da execução e, por fim, a etapa de simulação que ocorre a cada *frame* da execução. As etapas podem ser visualizadas na Figura 4.1.

Figura 4.1 – Fluxograma da metodologia do trabalho. Em verde as etapas durante a implementação do projeto, em azul as etapas que ocorrem no início da execução e em roxo a etapa que ocorre em tempo real.



Fonte: O Autor

4.1 Proposta

Neste protótipo, implementamos um método de simulação inspirado pelo método de agrupamento em mechas, onde a simulação é calculada para o grupo de fios como um todo, porém, adicionamos uma pequena variação na movimentação. Nessa implementação existem os fios guia, que são simulados pelo motor de física e são afetados pelas forças do ambiente, e os fios seguidores, que não são afetados pela física e têm seus posicionamentos e rotações calculados com base nos fios guia. Esse cálculo é baseado em diversos parâmetros (descritos na Seção 4.6.2) que geram variações únicas para cada segmento de fio.

4.2 Estrutura do Cabelo

Inicialmente, o fio seria gerado proceduralmente a partir de segmentos em formato de cilindro, o que traria maior flexibilidade na hora de escolher o comprimento de cada fio. Porém, isso implicaria ou em uma malha de renderização para cada segmento, o que causaria quebras visuais em um fio, ou em uma maior complexidade na hora de renderizar

esse fio. Como esse não era o foco deste trabalho, optamos por modelar o fio inteiro como um único objeto 3D de tamanho e subdivisões fixas. Modelamos cada fio como um objeto 3D no Blender (BLENDER, 2024), sendo esse uma malha semelhante a um cilindro, constituído em uma cadeia de 23 segmentos, como mostra a Figura 4.2.

Essa estrutura específica foi escolhida considerando que estaríamos modelando um cabelo liso com aproximadamente 2,5 vezes o tamanho do diâmetro da base esférica. Outras texturas de cabelo podem se beneficiar de outros tipos de estrutura.

Figura 4.2 – Geometria do modelo de um fio de cabelo em segmentos.

Fonte: O Autor

4.3 Configuração

Esta seção aborda as configurações e componentes necessários para gerar uma simulação dinâmica estável e realista.

4.3.1 Projeto

Para o desenvolvimento deste protótipo, utilizamos o motor de jogo Unity (TECHNOLOGIES, 2024a) na versão 2022.3.2f1. Todos os *scripts* executados na CPU são escritos em C e os *scripts* executados na GPU são escritos em HLSL.

O projeto foi construído a partir de um *template* 3D básico, com algumas alterações nas configurações do motor de física (Figura 4.3). Como temos uma cadeia longa de objetos com juntas, para melhorar a estabilidade da simulação, optamos por aumentar o número de interações do *solver* de 6 para 20 e a velocidade dessas interações de 1 para 10.

4.3.2 Fio de Cabelo Guia

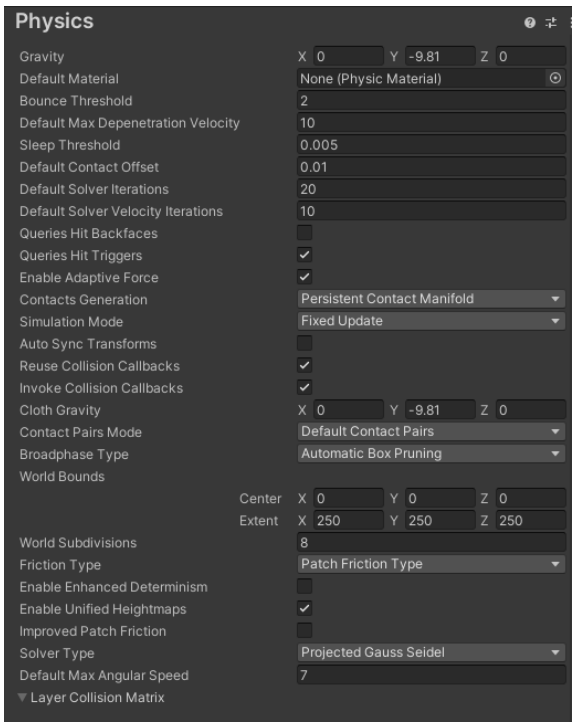
Para a simulação do comportamento dos fios, utilizamos o próprio motor de física da Unity. Cada fio é um objeto cilíndrico dividido em diversos segmentos de tamanho variável, organizados em uma hierarquia (Figura 4.4) onde cada segmento tem os componentes descritos nas próximas subseções.

Rigidbody

Componente que controla a movimentação de um objeto pela simulação física. Um objeto que contenha um *rigidbody* passa a ter a sua movimentação controlada pelo motor físico e afetada pela gravidade e outras forças aplicadas (TECHNOLOGIES, 2024j). Configuramos o *rigidbody* de cada segmento como na Figura 4.5.

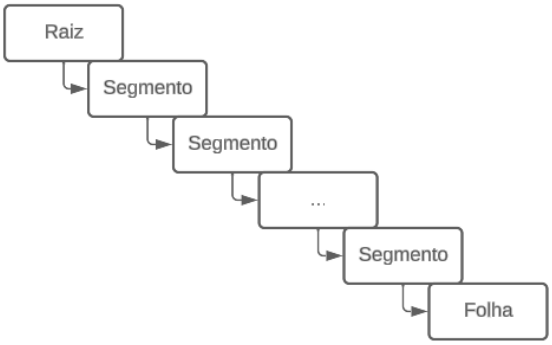
Utilizamos o modo de detecção de colisão discreto para este protótipo, pois é aquele que possui melhor performance. Além disso, para o caso de uso deste trabalho, não houve a necessidade de uma detecção mais precisa. Porém, simulações com muitas colisões em alta velocidade podem necessitar de um modo de detecção de colisão mais preciso. Outra escolha importante está relacionada a não utilizar interpolação para a suavização do movimento. Apesar da interpolação causar uma pequena melhora visual, o

Figura 4.3 – Configurações do projeto.



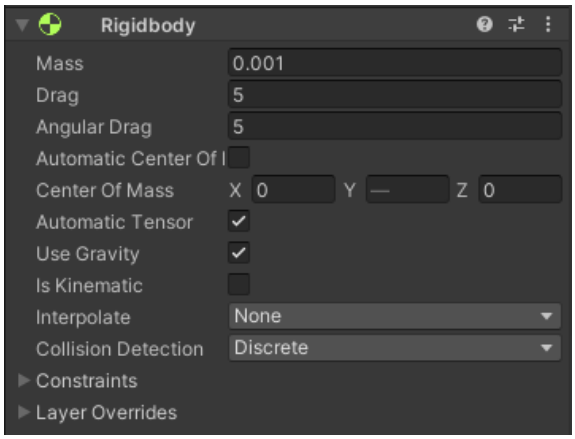
Fonte: O Autor

Figura 4.4 – Hierarquia dos segmentos de um fio de cabelo.



Fonte: O Autor

Figura 4.5 – Configurações do *rigidbody*.



Fonte: O Autor

custo computacional é significativo.

Collider

Cada segmento do fio tem um colisor em formato de cápsula (TECHNOLOGIES, 2024d). Esse formato foi escolhido pois é o que apresentou uma colisão mais estável, visto que as arestas da cápsula são mais suaves e tem uma superfície de contato menos variável. Um colisor mais simples poderia diminuir a complexidade das colisões, mas afeta negativamente o comportamento dos fios em alguns tipos de colisão. Utilizamos um material (TECHNOLOGIES, 2024i) com 0,2 de fricção, tanto dinâmica (quando dois objetos estão deslizando um contra o outro), quanto estática (quando dois objetos estão colidindo parados).

Configurable Joint

Componente que conecta um objeto a uma âncora e oferece diversas formas de

Figura 4.6 – Formato dos colisores de alguns segmentos.

Fonte: O Autor

controle do movimento desse objeto em relação a essa âncora, por exemplo, simulando o comportamento de molas ou articulações (TECHNOLOGIES, 2024f). Neste projeto cada segmento de fio é conectado ao seu objeto pai na hierarquia por meio de uma junta ancorada no final do segmento anterior, como mostra na Figura 4.7.

Cada junta tem movimento linear (o quanto o objeto se distancia da âncora) e movimento angular (o quanto o objeto rotaciona ao redor da âncora). Esses movimentos podem ser limitados de diversas formas. Para o caso de uso desse protótipo, que é um fio de cabelo, podemos limitar ao máximo o movimento linear, visto que a elasticidade de um fio é tão baixa que pode ser ignorada nesse contexto. O importante, nesse caso, é o movimento angular.

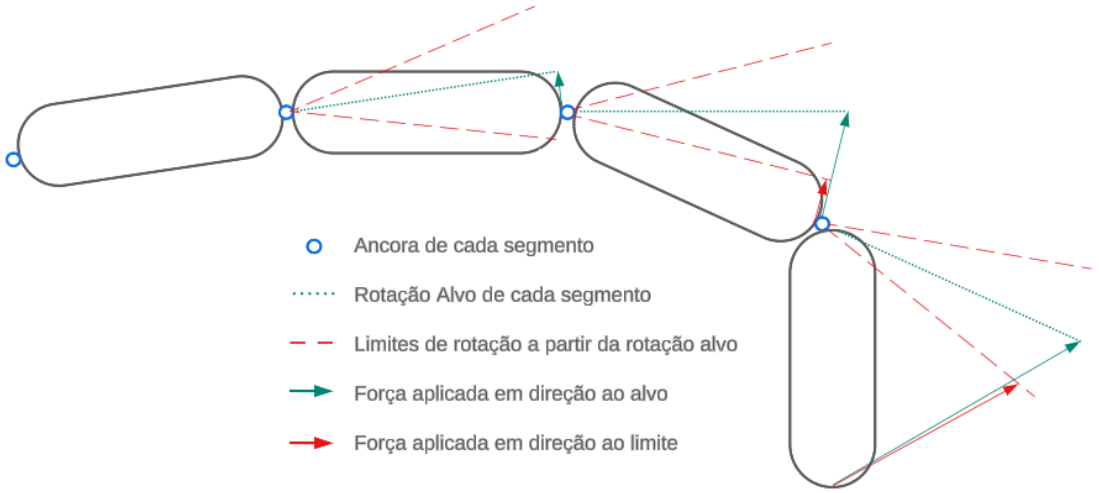
Também optamos por limitar a rotação em ±30 graus nos eixos X e Z, e ±10 graus no eixo Y, sendo Y o eixo de torção do fio. Idealmente o eixo Y seria fixo, sem nenhuma rotação, porém limites muito baixos tendem a diminuir a estabilidade da simulação. Como a malha de renderização é afetada por essa torção, precisamos incluir algum limite, caso contrário, poderíamos deixar esse eixo livre. Por definição, esses limites são flexíveis, ou seja, cada um tem uma definição de mola atrelada que vai aplicar força angular ao objeto para que ele volte a ter a rotação dentro dos limites estipulados.

Além desses limites de posição e rotação, temos o estado de repouso da junta, que é definido por uma posição, rotação, velocidade e velocidade angular. Além disso, para reforçar esse estado também são definidas molas. Por fim, definimos um limite de projeção. Caso a junta passe desse limite, ela é redefinida ao estado de repouso.

4.3.3 Fio de Cabelo Seguidor

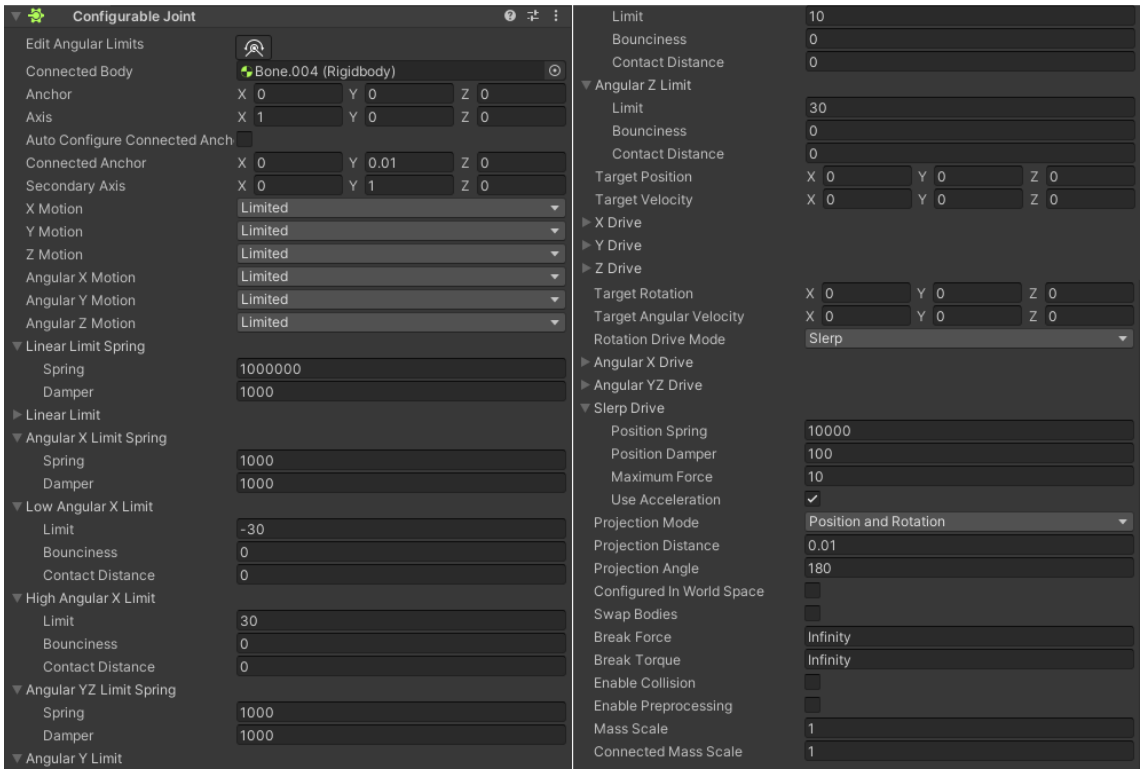
Os fios seguidores utilizam o mesmo objeto 3D usado nos fios guia, com a diferença de não possuir componentes adicionados.

Figura 4.7 – Principais aspectos da modelagem de um fio de cabelo.



Fonte: O Autor

Figura 4.8 – Exemplo de configurações de uma *configurable joint*.

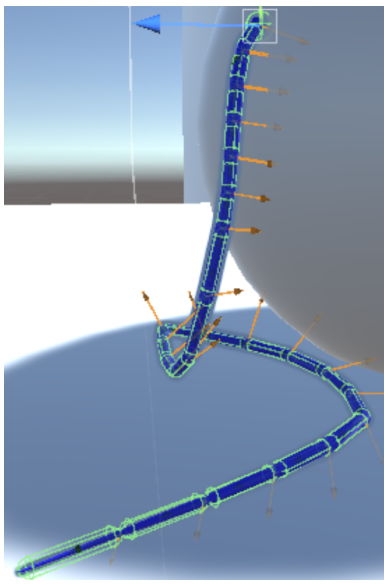


Fonte: O Autor

4.4 Distribuição dos Fios

Para a distribuição dos fios de cabelo de forma automatizada e eficiente, utilizamos o Modelo de Esfera de Fibonacci (GONZÁLEZ, 2010). Aplicamos o modelo para os fios

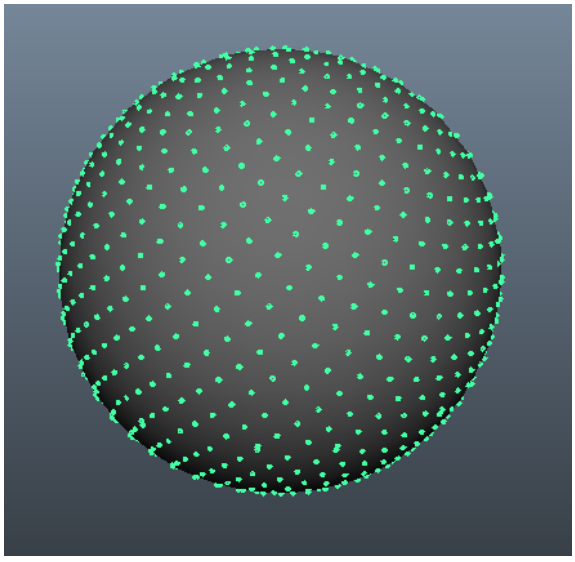
Figura 4.9 – Fio de cabelo guia.



Fonte: O Autor

guia e seguidores de forma independente e preenchemos apenas uma parte da esfera.

Figura 4.10 – Modelo de Esfera de Fibonacci.



Fonte: (INTERNET, 2024)

4.5 Seleção de Vizinhos

Após a distribuição dos fios pela base esférica, calculamos a distância da raiz de cada fio seguidor para os fios guia e selecionamos, de forma ordenada, os dois vizinhos mais próximos. O vizinho mais próximo é quem vai definir a maior parte da movimentação do fio seguidor, e o segundo mais próximo vai ser utilizado para calcular a variação desse movimento.

4.6 Simulação

Esta seção aborda o cálculo e a frequência de atualização dos fios de cabelo.

4.6.1 Fio Guia

Os fios guia são atualizados de acordo com o intervalo de atualização do motor físico, definido nas configurações do projeto (TECHNOLOGIES, 2024k). Neste projeto, esse valor é 0,02 segundos.

4.6.2 Fio Seguidor

Cálculo

Já que os fios seguidores não têm nenhum componente físico, a menos que eles sejam manualmente atualizados, eles não serão afetados por nenhum outro objeto ou força. Como os segmentos são organizados de forma hierárquica, e uma movimentação em um objeto afeta todos os seus objetos filhos, sabemos que as posições no sistema de coordenadas global são atualizadas de forma automática quando um objeto anterior da hierarquia se move. Considerando também que um segmento é posicionado ao final do segmento anterior, e eles devem estar sempre conectados, podemos assumir que o único tipo de movimentação que um segmento pode fazer é rotacionar ao redor desse ponto de conexão. Com essas informações em mente, os segmentos de um fio seguidor só precisam ter as suas rotações atualizadas de forma manual.

Para cada segmento de fio, calculamos a nova rotação como sendo a interpolação linear esférica (*slerp*) entre as rotações dos dois vizinhos mais próximos, utilizando um

fator de deslocamento composto de um deslocamento fixo, definido por uma função quadrática, dependente do índice do segmento, somado um deslocamento variável, baseado em um fator pseudoaleatório (único para cada segmento de fio) escalado pela velocidade do vizinho mais próximo.

Algorithm 1 Atualização das Rotações

$qs =$ Metade da quantidade de segmentos em um fio
$rand =$ Valor pseudoaleatório único para cada segmento de cabelo, no intervalo $[-1, 1]$
for A cada iteração do motor físico (FixedUpdate) **do**
    Salva as rotações e velocidades atuais dos fios guia em um buffer
    for Cada fio x em todos os fios seguidores **do**
        for Cada segmento y em todos os segmentos do fio x **do**
            $v_1 =$ Fio guia mais próximo
            $v_2 =$ Segundo fio guia mais próximo
            Calcula o deslocamento fixo $d_f = 1 - ((y - qs)^2)/(qs^2)$
            Calcula o deslocamento variável $d_v =$ Velocidade de v_1 limitada ao intervalo $[0, 0.1]$
            Calcula o deslocamento resultante $d_r = 0.2 * d_f + rand_{xy} * d_v$
            Calcula a nova rotação de y como sendo $y_{rot} = slerp(v_{1rot}, v_{2rot}, d_r)$
        end for
    end for
    Atualiza as rotações de cada fio seguidor com os valores calculados
end for

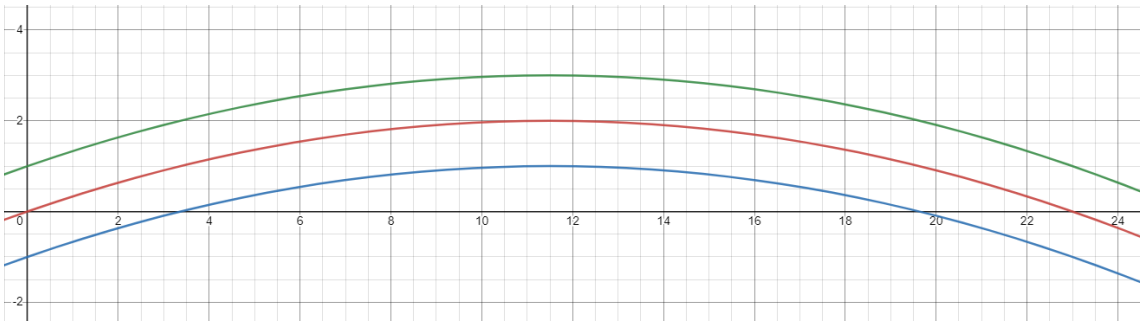
Em momentos que o vizinho mais próximo está em deslocamento, a variação de deslocamento de cada segmento gera um formato único para cada fio, causando o efeito de diferentes deslocamentos em vários fios seguidores com o mesmo vizinho mais próximo. Em contrapartida, nos momentos de repouso, com a anulação da variação do deslocamento, os fios ficam mais alinhados, evitando assim fios em posições não naturais.

A função de deslocamento fixo é calculada de forma que a raiz e a ponta dos fios sejam as partes que menos se desloquem em relação a posição do vizinho mais próximo (Figura 4.11). Utilizamos essa função para que cada fio tenha o formato levemente alterado, mesmo em repouso.

Frequência de Atualização

Assumindo que temos 23 segmentos por fio, em uma simulação de 800 fios seguidores, teríamos que computar 18400 novas rotações por *frame*, o que resultaria em um tempo de execução bem alto. Como cada segmento depende apenas dos seus vizinhos guia e não se afetam entre si, a solução que encontramos para esse problema foi paralelizar o cálculo de rotação. Optamos por fazer essa paralelização na GPU utilizando *compute shaders* (TECHNOLOGIES, 2024e) pois a GPU oferece uma melhor paralelização em

Figura 4.11 – Curva de deslocamento onde o eixo x é o índice do segmento sendo calculado e o eixo y é o deslocamento. A curva em vermelho é o deslocamento final quando o deslocamento variável é zero. Já as curvas em azul e verde representam o deslocamento final quando o deslocamento variável está no seu valor mínimo e máximo respectivamente. Obs.: O eixo y está escalado em 10 vezes para fins de visualização.



Fonte: O Autor

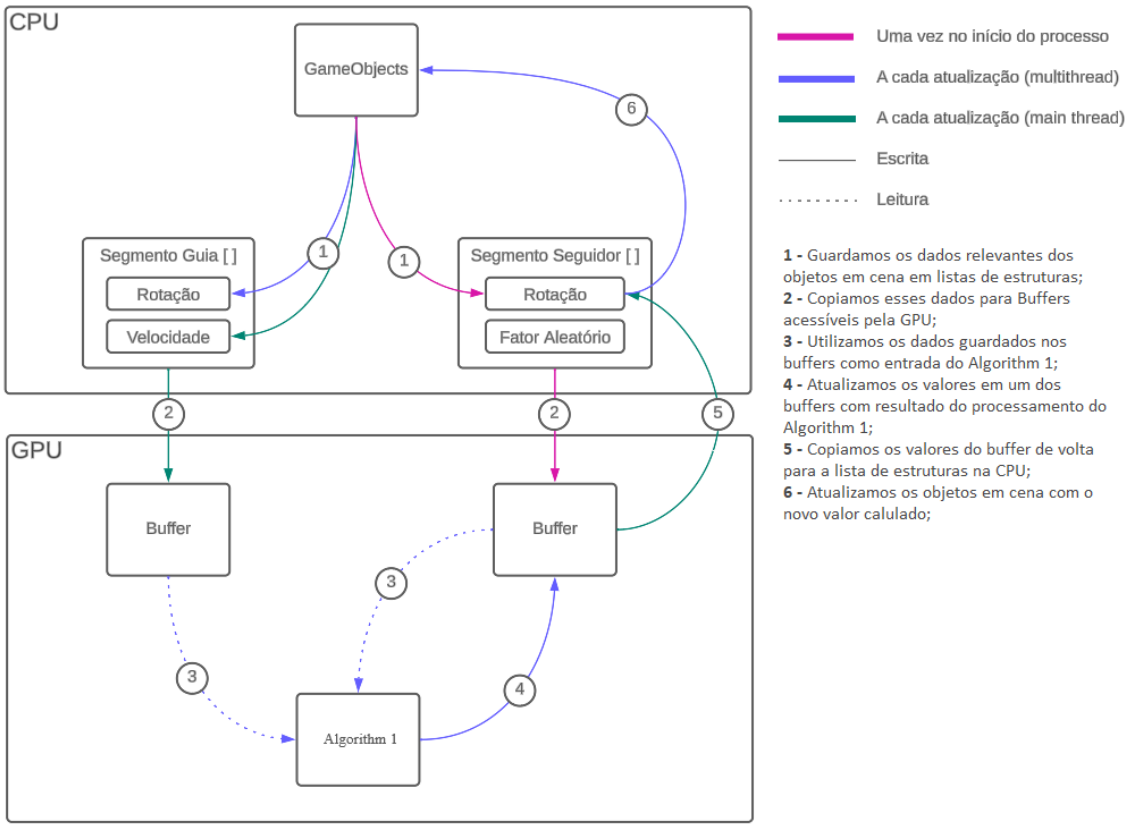
escala.

Um problema encontrado, porém, é que a leitura e escrita de componentes gerenciados pela Unity é exclusivamente feita na *main thread* do projeto (com uma exceção, comentada no próximo parágrafo), o que causa um custo computacional adicional da cópia desses valores entre os componentes gerenciados e *buffers* acessíveis fora da *main thread*. Isso faz com que sempre que vamos recalcular a rotação de cada segmento de fio temos que atualizar esses *buffers* com o estado atual dos fios guia. A Figura 4.12 mostra o fluxo completo de atualização das rotações.

Uma solução seria encontrar uma forma de acessar esses componentes gerenciados fora da *main thread*, que nesse caso são o *Transform* (onde fica a rotação) e *Rigidbody* (onde fica a velocidade). No caso do *Transform*, existe o *IJobParallelForTransform* (TECHNOLOGIES, 2024h), uma paralelização feita na CPU, que permite leitura e escrita fora da main thread, com a restrição de que escritas em *Transforms* que compartilhem uma mesma hierarquia só podem ocorrer em uma mesma thread. Assim, para o caso de escritas, mesmo que não seja feita na *main thread*, ainda é em uma única *thread*, já que todos os fios compartilham a mesma hierarquia. Contudo, para o *Rigidbody* não existe essa opção, tendo sua leitura e escrita sendo feita exclusivamente na *main thread*.

Ainda assim utilizamos *IJobParallelForTransform* para realizar tanto a leitura das rotações dos fios guia, quanto a atualização das rotações dos fios seguidores. No caso da leitura, temos a vantagem de fazer o acesso em paralelo, e no caso da atualização, o ganho é que essa escrita é feita em uma *thread* separada, que nesse caso está sendo usada exclusivamente para esse fim. Dessa forma, o uso desse artifício nos economiza por volta

Figura 4.12 – Fluxo de atualização dos fios seguidores.



Fonte: O Autor

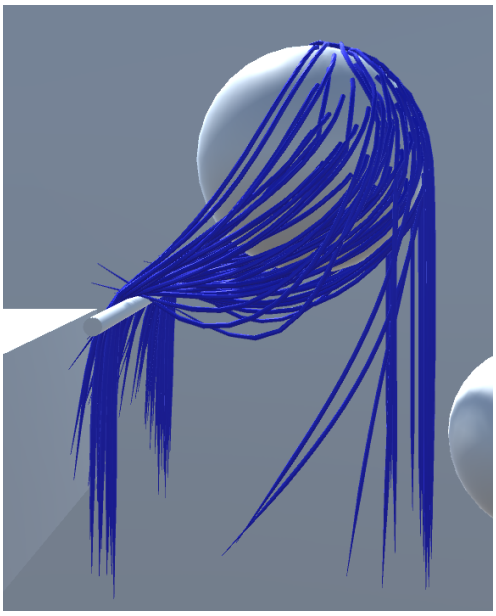
de 2,5ms no tempo de execução de um *frame*.

5 RESULTADOS E DISCUSSÃO

O objetivo do nosso trabalho é estudar maneiras de otimizar o tempo de simulação e testar a hipótese de uma implementação utilizando os sistemas de simulação física de propósito geral utilizados em jogos. A quantidade e proporção dos fios foi escolhida baseada em testes empíricos.

Para avaliarmos a qualidade e eficiência do nosso modelo, começamos por testar o modelo que implementamos sem nenhuma otimização (Figura 5.1). Podemos analisar pela Tabela 5.1 que a simulação dos fios guia não escala nada bem, visto que com apenas 845 fios a aplicação já não é capaz de executar como esperado.

Figura 5.1 – Modelo básico contendo 94 fios guia.

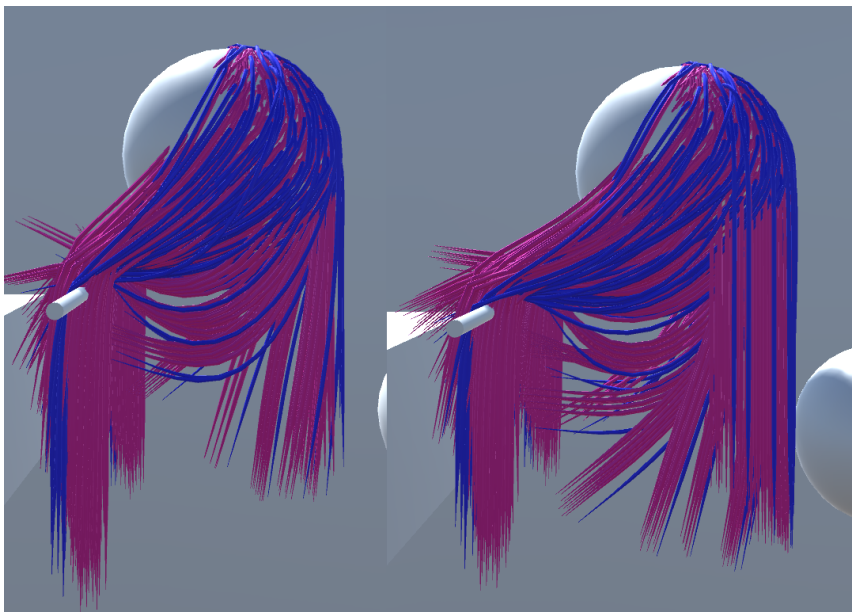


Fonte: O Autor

A partir disso já sabemos que não podemos simular muitos fios, então começamos a testar hipóteses e analisar como elas se comportam. Começamos com a proposta de implementar alguma variação do modelo de agrupamento (Capítulo 3), visto que é uma implementação simples que economiza bastante processamento e tem grande potencial para modificações. Testamos primeiro o modelo base, onde fios de um mesmo agrupamento têm exatamente a mesma movimentação, para poder comparar com futuras alterações (Figura 5.2).

O primeiro pensamento que vem à mente quando testamos o modelo de agrupamento, é a constatação de que os agrupamentos são muito óbvios e o cabelo perde a

Figura 5.2 – Modelo com agrupamento contendo 94 fios guia e 751 seguidores.



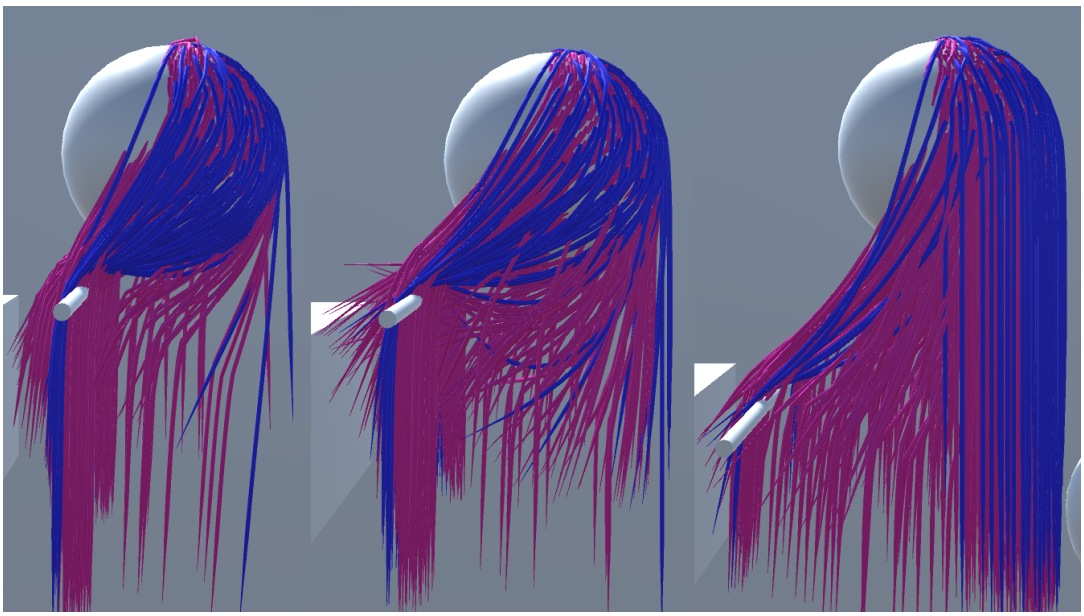
Fonte: O Autor

sensação de continuidade quando em movimento. Então, o primeiro teste que fizemos foi interpolar entre os dois fios guia mais próximos, considerando a distância entre eles como o fator de deslocamento da interpolação (Figura 5.3). Enquanto essa abordagem traz um visual menos dividido e com mais movimento, em situações que o cabelo se divide, ou fios próximos estão em uma configuração muito diferente, as interpolações geram posições nada naturais (Figura 5.4).

Analisando os experimentos até agora, chegamos a conclusão de que precisamos do movimento e aspecto de comportamento individual que a interpolação traz, mas também precisamos que os fios respeitem o movimento em conjunto e não fiquem em posições não naturais (Figura 5.4). Portanto, decidimos dar mais peso ao fio guia mais próximo. Assim, o fio seguidor vai seguir, em geral, um comportamento natural bem definido, evitando casos em que os fios próximos estão fazendo movimentos totalmente diferentes e o meio termo não faz sentido. Somado a isso, para o movimento não ficar com agrupamentos tão óbvios, como no método de agrupamento simples, adicionamos variação ao movimento, com variações únicas por fio, ativadas pela velocidade do fio guia (Figura 5.5).

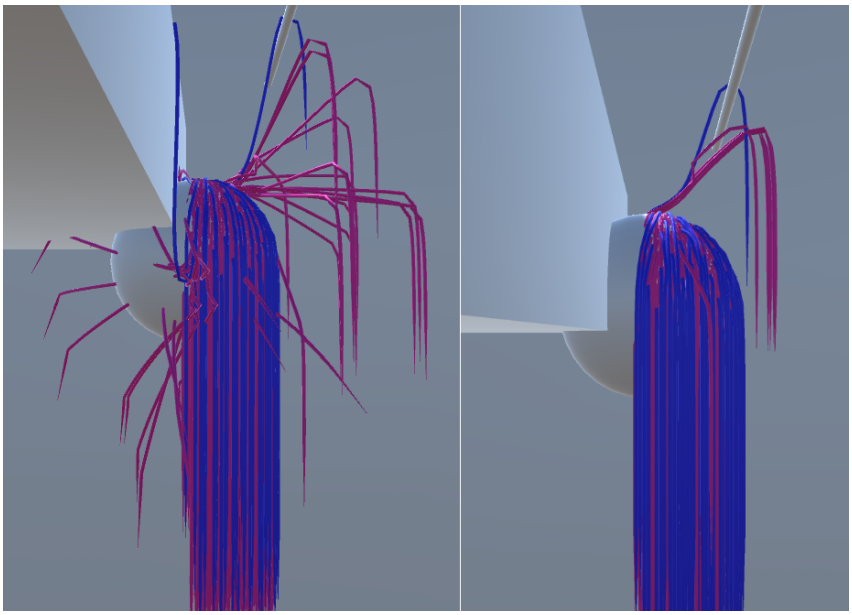
Este método, apesar de manter os agrupamentos de forma geral, é menos óbvio visualmente em relação a eles. Em termos de performance, os três modelos de agrupamento testados apresentaram resultados bem parecidos, com uma contagem de FPS similar.

Figura 5.3 – Modelo com interpolação contendo 94 fios guia e 751 seguidores.



Fonte: O Autor

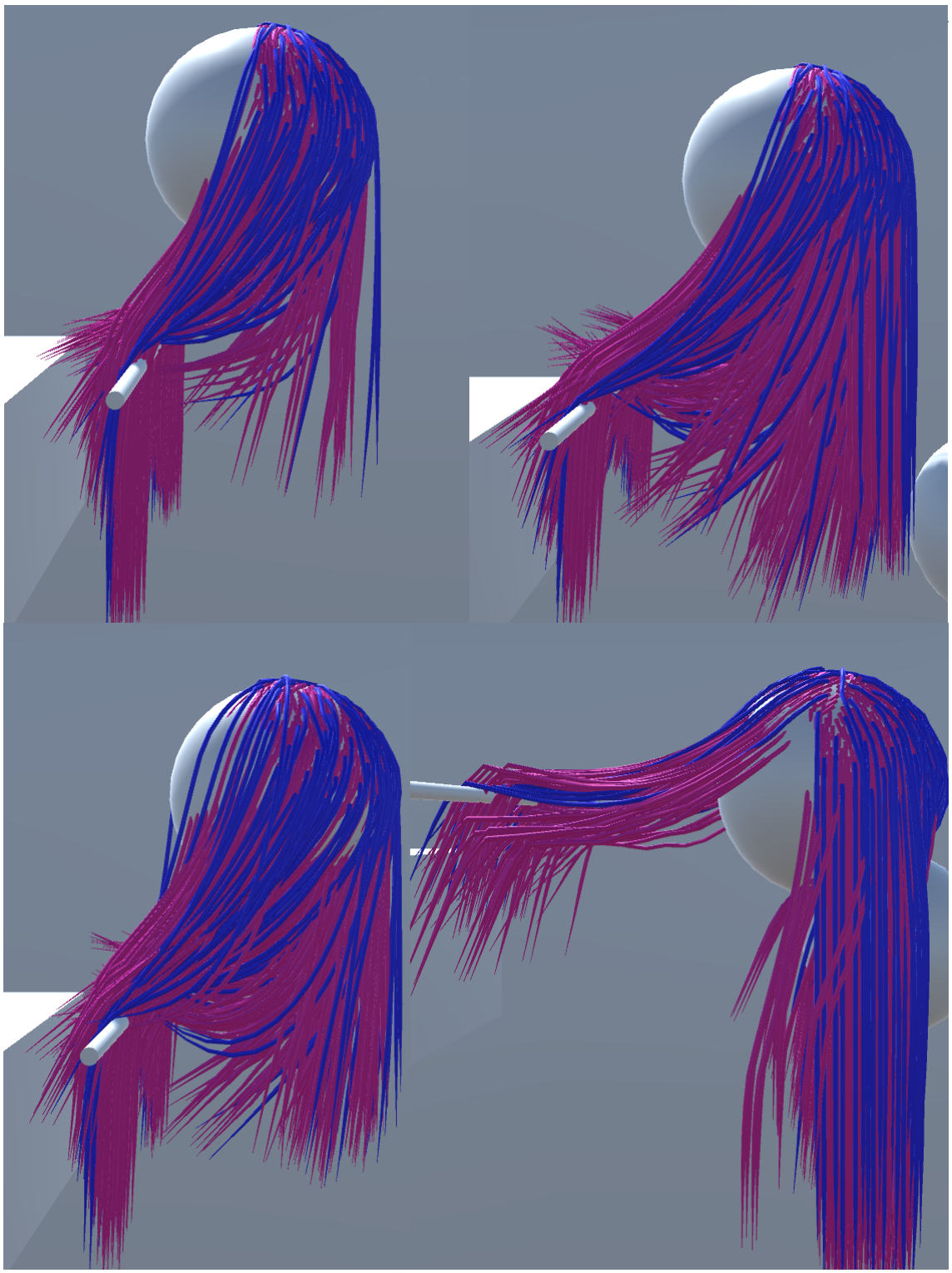
Figura 5.4 – Modelo com interpolação contendo 94 fios guia e 751 seguidores à esquerda. Modelo com interpolação variada contendo 94 fios guia e 751 seguidores à direita.



Fonte: O Autor

A partir da análise da Tabela 5.1, que representa a performance dos experimentos em FPS, e comparando as linhas 4 e 6, conseguimos validar que o modelo transforma uma simulação incapaz de ser processada em tempo real, em uma com essa possibilidade. Conseguimos também validar, comparando as linhas 5 e 6, que o uso da GPU para o

Figura 5.5 – Modelo com interpolação variada contendo 94 fios guia e 751 seguidores.

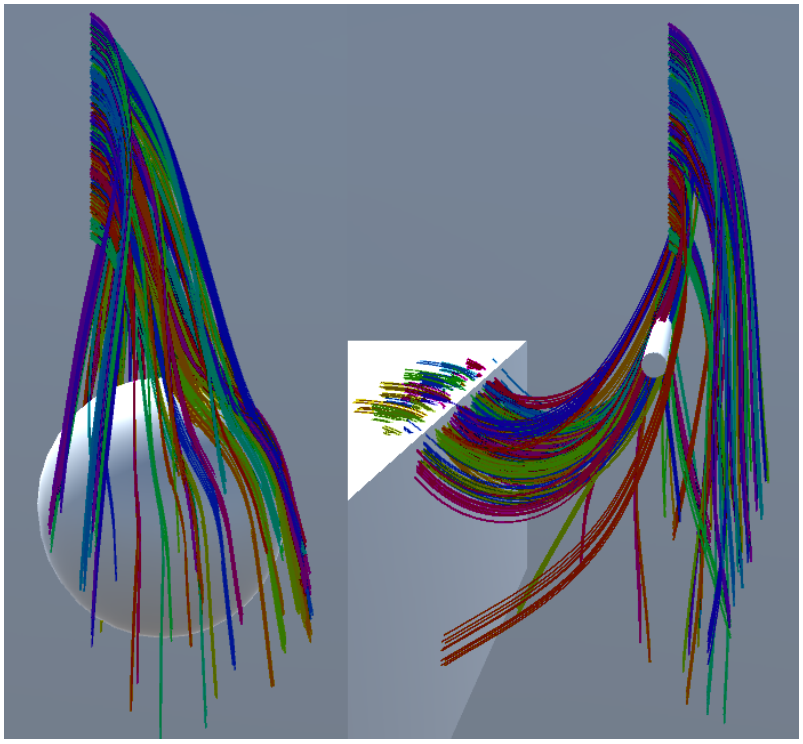


Fonte: O Autor

cálculo de atualização das rotações melhora consideravelmente a eficiência do modelo. Já as linhas 6 e 7 nos mostram que colisões são uma grande parte do processamento, deixando evidência de que otimizar os cálculos de colisões traria bastante benefício.

Além das comparações do nosso próprio modelo, testamos o *Hair System* desenvolvido pela própria Unity, um simulador físico altamente configurável, no qual pudemos configurar alguns parâmetros semelhantes aos usados nos testes do nosso simulador. De modo geral, como mostra a Tabela 5.1, o simulador da Unity é muito mais eficiente e escalável. Porém, além da colisão não ser confiável (os cabelos atravessam alguns objetos), o comportamento dos fios em relação a maleabilidade é um pouco diferente, como mostram as Figuras 5.6 e 5.7.

Figura 5.6 – Unity Hair System com 94 clusters de 9 fios.

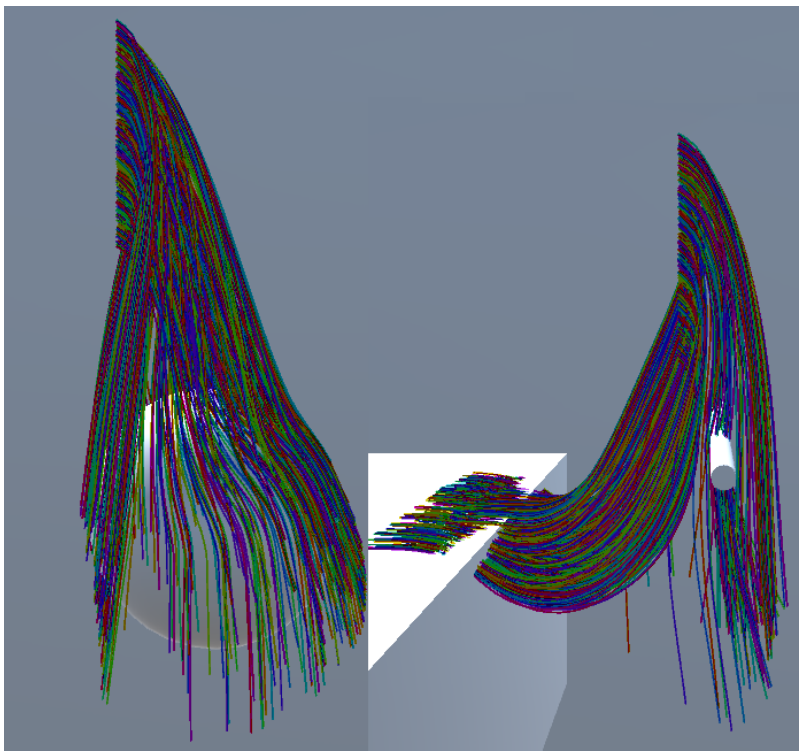


Fonte: O Autor

Dados os resultados obtidos nos experimentos, podemos concluir que o modelo de agrupamento com interpolação variada tem um visual mais agradável em relação aos modelos de agrupamento simples e agrupamento com interpolação sem variação. Já em termos de performance os três não têm diferenças significativas entre si. A diferença em performance vem do uso ou não de paralelização e detecção de colisão entre fios, que pode ser aplicado a qualquer um dos modelos. Portanto, entre os modelos experimentados, a recomendação geral é o uso do modelo de agrupamento com interpolação variada com paralelização, pelos motivos citados anteriormente. Com a opção de detecção de colisão entre fios sendo dependente do caso de uso, visto que afeta bastante na performance.

Para uma melhor visualização dos resultados e comparações, acessar o video dis-

Figura 5.7 – Unity Hair System com 845 fios.



Fonte: O Autor

ponibilizado no *YouTube* (GOMES, 2024).

Tabela 5.1 – Tabela de FPS médio. Alguns testes em máquinas específicas apresentaram um limite máximo de 60 FPS. A primeira coluna numera as linhas.

		i5-9600K 16GB RAM GTX 1060 6GB	Ryzen 5 3600XT 24GB RAM RTX 2070 8GB	i7-13850HX 64GB RAM RTX 2000 Ada Gen Laptop
1	Unity Hair System (845 fios)	164	165	60*
2	Unity Hair System (94 clusters de 9 fios)	164	165	60*
3	Protótipo Básico (94 fios guia)	150	130	60*
4	Protótipo Básico (845 fios guia)	< 1	< 1	< 1
5	Protótipo sem Paralelização (94 guias e 751 seguidores)	35	35	40
6	Protótipo com Paralelização (94 guias e 751 seguidores)	50	50	60*
7	Protótipo com Paralelização sem colisão entre fios (94 guias e 751 seguidores)	73	80	60*

Fonte: O Autor

6 CONCLUSÃO

Neste trabalho, implementamos parcialmente um simulador de cabelo estável, utilizando o motor de jogo Unity para calcular a simulação física. Também investigamos e testamos otimizações com técnicas de agrupamento de mechas e paralelização em GPU. Conseguimos passar de uma simulação que estava com um tempo de processamento tão alto que o FPS estava abaixo de 1, para uma simulação com uma performance na faixa de 50 a 60 FPS, o que é um resultado aceitável.

Considerando isso, acreditamos que fizemos um bom trabalho de otimização do modelo proposto com uma técnica que pode, possivelmente, ser replicada em outros modelos.

6.1 Limitações

Modelar fios de cabelo estáveis se mostrou um grande desafio. A modelagem como uma cadeia de juntas se mostrou difícil de configurar, com muitos parâmetros que muitas vezes não geravam o efeito esperado, e sim os mais diversos efeitos caóticos. Um problema é o motor de física não conseguir simular muito bem grandes cadeias de juntas, problema esse que é resolvido, ou simplificando essas cadeias, ou aumentando o número de interações do *solver*, o que impacta negativamente na performance.

Outra dificuldade é a incapacidade de paralelizar certas operações na Unity, como por exemplo, a atualização de componentes gerenciados pelo próprio motor (a não ser em situações muito específicas, citadas no Capítulo 4). O processamento para atualizar as rotações leva muito mais tempo do que calcula-las, provavelmente teríamos um ganho de eficiência se conseguíssemos paralelizar essa tarefa.

6.2 Trabalhos futuros

Com o intuito de aperfeiçoar este trabalho, diversas melhorias podem ser realizadas, uma delas sendo o aprimoramento do visual do fio de cabelo. Isso pode ser feito através da implementação de um modelo de densidade para gerar mechas de cabelo mais realistas, ou com técnicas de textura e iluminação. A melhora nesse aspecto contribuiria significativamente para a percepção geral da simulação.

Outra possibilidade seria testar a hipótese de agrupamento proposta neste trabalho em outros simuladores mais complexos e eficientes, que utilizem modelos geométricos similares. A possibilidade de aumento no número de fios de cabelo traria experimentos mais realistas e interessantes, permitindo uma melhor avaliação da efetividade da técnica proposta.

A implementação de outras texturas de cabelo, especialmente cabelos cacheados, seria interessante também, visto que existe uma complexidade inerente a fios com essas curvaturas.

Outra melhoria, seria uma análise com distribuições e posicionamentos diferentes de fios guia e seguidores, podendo, possivelmente, melhorar aspecto e performance. Além disso, um método mais robusto para calcular posição dos fios em situações com fios *outliers* seria interessante.

REFERÊNCIAS

ANJYO, K.-i.; USAMI, Y.; KURIHARA, T. A simple method for extracting the natural beauty of hair. In: **Proceedings of the 19th annual conference on Computer graphics and interactive techniques**. [S.l.: s.n.], 1992. p. 111–120.

BANDO, Y.; CHEN, B.-Y.; NISHITA, T. Animating hair with loosely connected particles. In: WILEY ONLINE LIBRARY. **Computer Graphics Forum**. [S.l.], 2003. v. 22, n. 3, p. 411–418.

BERGOU, M. et al. Discrete elastic rods. In: **ACM SIGGRAPH 2008 papers**. [S.l.: s.n.], 2008. p. 1–12.

BERTAILS-DESCOUBES, F. et al. A nonsmooth newton solver for capturing exact coulomb friction in fiber assemblies. **ACM Transactions on Graphics (TOG)**, ACM New York, NY, USA, v. 30, n. 1, p. 1–14, 2011.

BERTAILS, F. et al. Super-helices for predicting the dynamics of natural hair. **ACM Transactions on Graphics (TOG)**, ACM New York, NY, USA, v. 25, n. 3, p. 1180–1187, 2006.

BLENDER. **Blender**. 2024. Accessed: 2024-02-05. Available from Internet: <https://www.blender.org/>.

BRIDSON, R.; MARINO, S.; FEDKIW, R. Simulation of clothing with folds and wrinkles. In: **ACM SIGGRAPH 2005 Courses**. [S.l.: s.n.], 2005. p. 3–es.

CASATI, R.; BERTAILS-DESCOUBES, F. Super space clothoids. **ACM Transactions on Graphics (TOG)**, ACM New York, NY, USA, v. 32, n. 4, p. 1–12, 2013.

CHAI, M.; ZHENG, C.; ZHOU, K. A reduced model for interactive hairs. **ACM Transactions on Graphics (TOG)**, ACM New York, NY, USA, v. 33, n. 4, p. 1–11, 2014.

CHANG, J. T.; JIN, J.; YU, Y. A practical model for hair mutual interactions. In: **Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation**. [S.l.: s.n.], 2002. p. 73–80.

DAVIET, G. Simple and scalable frictional contacts for thin nodal objects. **ACM Transactions on Graphics (TOG)**, ACM New York, NY, USA, v. 39, n. 4, p. 61–1, 2020.

DAVIET, G. Interactive hair simulation on the gpu using admm. In: **ACM SIGGRAPH 2023 Conference Proceedings**. [S.l.: s.n.], 2023. p. 1–11.

DAVIET, G.; BERTAILS-DESCOUBES, F.; BOISSIEUX, L. A hybrid iterative solver for robustly capturing coulomb friction in hair dynamics. In: **Proceedings of the 2011 SIGGRAPH Asia Conference**. [S.l.: s.n.], 2011. p. 1–12.

FINK, B. et al. Age, health and attractiveness perception of virtual (rendered) human hair. **Frontiers in psychology**, Frontiers Media SA, v. 7, p. 1893, 2016.

GOMES, F. **Estudo de Otimização de Simulação de Cabelo em Tempo Real Baseado em Física em Unity**. 2024. Accessed: 2024-02-22. Available from Internet: <https://youtu.be/Yw9DTPk2bFQ>.

GONZÁLEZ, Á. Measurement of areas on a sphere using fibonacci and latitude–longitude lattices. **Mathematical Geosciences**, Springer, v. 42, p. 49–64, 2010.

GUANG, Y.; ZHIYONG, H. A method of human short hair modeling and real time animation. In: IEEE. **10th Pacific Conference on Computer Graphics and Applications, 2002. Proceedings.** [S.l.], 2002. p. 435–438.

HADAP, S.; MAGNENAT-THALMANN, N. Modeling dynamic hair as a continuum. In: WILEY ONLINE LIBRARY. **Computer Graphics Forum**. [S.l.], 2001. v. 20, n. 3, p. 329–338.

IBEN, H. et al. Artistic simulation of curly hair. In: **Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation**. [S.l.: s.n.], 2013. p. 63–71.

INTERNET. **Esfera de Fibonnacci**. 2024. Accessed: 2024-02-03. Available from Internet: <https://i.stack.imgur.com/NsCif.png>.

JIANG, C.; GAST, T.; TERAN, J. Anisotropic elastoplasticity for cloth, knit and hair frictional contact. **ACM Transactions on Graphics (TOG)**, ACM New York, NY, USA, v. 36, n. 4, p. 1–14, 2017.

JIANG, J. et al. Real-time hair simulation with heptadiagonal decomposition on mass spring system. **Graphical Models**, Elsevier, v. 111, p. 101077, 2020.

JIMENEZ, S.; LUCIANI, A. Animation of interacting objects with collisions and prolonged contacts. In: SPRINGER. **Modeling in Computer Graphics: Methods and Applications**. [S.l.], 1993. p. 129–141.

KAUFMAN, D. M. et al. Adaptive nonlinearity for collisions in complex rod assemblies. **ACM Transactions on Graphics (TOG)**, ACM New York, NY, USA, v. 33, n. 4, p. 1–12, 2014.

KOH, C. K.; HUANG, Z. Real-time animation of human hair modeled in strips. In: SPRINGER. **Computer Animation and Simulation 2000: Proceedings of the Eurographics Workshop in Interlaken, Switzerland, August 21–22, 2000**. [S.l.], 2000. p. 101–110.

KOH, C. K.; HUANG, Z. A simple physics model to animate human hair modeled in 2d strips in real time. In: SPRINGER. **Computer Animation and Simulation 2001: Proceedings of the Eurographics Workshop in Manchester, UK, September 2–3, 2001**. [S.l.], 2001. p. 127–138.

LIANG, W.; HUANG, Z. An enhanced framework for real-time hair animation. In: IEEE. **11th Pacific Conference onComputer Graphics and Applications, 2003. Proceedings.** [S.l.], 2003. p. 467–471.

MCADAMS, A. et al. Detail preserving continuum simulation of straight hair. **ACM Transactions on Graphics (ToG)**, ACM New York, NY, USA, v. 28, n. 3, p. 1–6, 2009.

MÜLLER, M.; KIM, T.-Y.; CHENTANEZ, N. Fast simulation of inextensible hair and fur. **VRIPHYS**, v. 12, p. 39–44, 2012.

NVIDIA. **NVIDIA PhysX SDK 4.1 Documentation**. 2021. <https://gameworksdocs.nvidia.com/PhysX/4.1/documentation/physxguide/Index.html>. Accessed: 2024-02-01.

PETROVIC, L.; HENNE, M.; ANDERSON, J. Volumetric methods for simulation and rendering of hair. **Pixar Animation Studios**, v. 2, n. 4, p. 1–6, 2005.

ROSENBLUM, R. E.; CARLSON, W. E.; III, E. T. Simulating the structure and dynamics of human hair: modelling, rendering and animation. **The Journal of Visualization and Computer Animation**, Wiley Online Library, v. 2, n. 4, p. 141–148, 1991.

SALES, G. L.; MAIA, M. C. **Física Básica I**. [s.n.], 2018. Available from Internet: <{https://educapes.capes.gov.br/handle/capes/42954}.>

SELLE, A.; LENTINE, M.; FEDKIW, R. A mass spring model for hair simulation. In: **ACM SIGGRAPH 2008 papers**. [S.l.: s.n.], 2008. p. 1–11.

SGOUROS, S. **The Role of Simulation in Hair Consulting: Advantages and Disadvantages**. 2024. Accessed: 2024-02-05. Available from Internet: <https://www.linkedin.com/pulse/role-simulation-hair-consulting-advantages-sakis-sgouros/>.

SHIRLEY, P.; ASHIKHMIN, M.; MARSCHNER, S. **Fundamentals of computer graphics**. [S.l.]: AK Peters/CRC Press, 2009.

SUGISAKI, E. et al. Simulation-based cartoon hair animation. UNION Agency, 2005.

TAŞKIRAN, H. D.; GÜDÜKBAY, U. Physically-based simulation of hair strips in real-time. Václav Skala-UNION Agency, 2005.

TECHNOLOGIES, U. **Case studies by industry**. 2024. <https://unity.com/case-study>. Accessed: 2024-02-01.

TECHNOLOGIES, U. **Unity - Manual**. 2024. <https://docs.unity3d.com/Manual/index.html>. Accessed: 2024-02-02.

TECHNOLOGIES, U. **Unity - Manual: Built-in 3D Physics**. 2024. <https://docs.unity3d.com/Manual/PhysicsOverview.html>. Accessed: 2024-02-01.

TECHNOLOGIES, U. **Unity - Manual: Capsule collider component reference**. 2024. <https://docs.unity3d.com/Manual/class-CapsuleCollider.html>. Accessed: 2024-01-31.

TECHNOLOGIES, U. **Unity - Manual: Compute shaders**. 2024. <https://docs.unity3d.com/Manual/class-ComputeShader.html>. Accessed: 2024-01-31.

TECHNOLOGIES, U. **Unity - Manual: Configurable Joint component reference**. 2024. <https://docs.unity3d.com/Manual/class-ConfigurableJoint.html>. Accessed: 2024-01-31.

TECHNOLOGIES, U. **Unity - Manual: Order of execution for event functions**. 2024. <https://docs.unity3d.com/Manual/ExecutionOrder.html>. Accessed: 2024-02-01.

TECHNOLOGIES, U. **Unity - Manual: Parallel jobs**. 2024. <https://docs.unity3d.com/Manual/JobSystemParallelForJobs.html>. Accessed: 2024-02-01.

TECHNOLOGIES, U. **Unity - Manual: Physic Material asset reference**. 2024. <https://docs.unity3d.com/Manual/class-PhysicMaterial.html>. Accessed: 2024-01-31.

TECHNOLOGIES, U. **Unity - Manual: Rigidbody physics**. 2024. <https://docs.unity3d.com/Manual/rigidbody-physics-section.html>. Accessed: 2024-01-31.

TECHNOLOGIES, U. **Unity - Manual: Time**. 2024. <https://docs.unity3d.com/Manual/class-TimeManager.html>. Accessed: 2024-01-31.

WARD, K. et al. A survey on hair modeling: Styling, simulation, and rendering. **IEEE transactions on visualization and computer graphics**, IEEE, v. 13, n. 2, p. 213–234, 2007.