UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

ESTUDO E APERFEIÇOAMENTO DA TÉCNICA DE STEERING BEHAVIORS NA SIMULAÇÃO FÍSICA DE FLUIDOS EM UM ESPAÇO TRIDIMENSIONAL

DISSERTAÇÃO DE MESTRADO

Henrique Vicentini

Santa Maria, RS, Brasil

2008

ESTUDO E APERFEIÇOAMENTO DA TÉCNICA DE STEERING BEHAVIORS NA SIMULAÇÃO FÍSICA DE FLUIDOS EM UM ESPAÇO TRIDIMENSIONAL

por

Henrique Vicentini

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do grau de
Mestre em Informática

Orientador: Prof. Dr. César Tadeu Pozzer (UFSM)

Co-orientador: Profª Drª Marcos Cordeiro d'Ornellas (UFSM)

Dissertação de Mestrado Nº 2
Santa Maria, RS, Brasil

2008

Universidade Federal de Santa Maria
Centro de Tecnologia
Programa de Pós-Graduação em Informática

A Comissão Examinadora, abaixo assinada,
aprova a Dissertação de Mestrado

ESTUDO E APERFEIÇOAMENTO DA TÉCNICA DE STEERING BEHAVIORS NA SIMULAÇÃO FÍSICA DE FLUIDOS EM UM ESPAÇO TRIDIMENSIONAL

elaborada por
Henrique Vicentini

como requisito parcial para obtenção do grau de
Mestre em Informática

COMISSÃO EXAMINADORA:

Profª Drª Marcos Cordeiro d'Ornellas (UFSM)
(Presidente/Co-orientador)

Prof. Dr. ISSO TROCAR (UFSM)

Prof. Dr ISSO TROCAR (UFSM)

Santa Maria, 22 de Agosto de 2008.

“Tudo vai ser bom.” — JOACHIM MEYER

RESUMO

Dissertação de Mestrado
Programa de Pós-Graduação em Informática
Universidade Federal de Santa Maria

ESTUDO E APERFEIÇOAMENTO DA TÉCNICA DE STEERING BEHAVIORS NA SIMULAÇÃO FÍSICA DE FLUIDOS EM UM ESPAÇO TRIDIMENSIONAL

Autor: Henrique Vicentini
Orientador: Prof. Dr. César Tadeu Pozzer (UFSM)
Co-orientador: Profª Drª Marcos Cordeiro d'Ornellas (UFSM)
Local e data da defesa: Santa Maria, 22 de Agosto de 2008.

Resumo em português aqui.

Palavras-chave: Simulação de fluidos, stearing behaviors, computação gráfica.

ABSTRACT

Master's Dissertation
Programa de Pós-Graduação em Informática
Universidade Federal de Santa Maria

VXDL: A LANGUAGE FOR INTERCONNECTION AND RESOURCES SPECIFICATION IN VIRTUAL GRIDS

Author: Henrique Vicentini
Advisor: Prof. Dr. César Tadeu Pozzer (UFSM)
Coadvisor: Prof[a] Dr[a] Marcos Cordeiro d'Ornellas (UFSM)

Grid computing has been defined as an infrastructure integrator of distributed resources. Although it is already used on a large scale in many areas, this type of computational infrastructure is still an area of active research, with many open questions. Today, new research works investigate the application of resources virtualization techniques to perform the composition of virtual grids. These grids can be defined as a high level abstraction of resources (computing and network), through which users have a view of a wide range of interconnected computers, that can be selected and virtually organized. In a virtual grid, as well in a real grid, users and middleware must have tools that allow the composition and management of the infrastructure. Among these tools, there are languages for resource description that allow the specification of components that will be used in the infrastructure. In a virtualized environment, the resources descriptions languages should offer attributes that interact with some peculiarities, such as the possibility of allocate multiple virtual resources (computing and network) on the same physical resource. In this context, this work presents VXDL, a language developed for the interconnections and resources description in virtual grids. The innovations proposed in VXDL allow the description, classification and parameter specification of all desirable components, including network topology and virtual routers. VXDL also allow the specification of a execution timeline, which can assist grid middleware in the tasks of resources sharing and scheduling. To evaluate the proposed language, this work presentes I) a comparative study between VXDL and other resources description languages and II) an analysis of results obtained with the benchmarks execution in virtual infrastructures composed using different VXDL descriptions.

Keywords: virtualization, virtual grids, virtual clusters, resources description language.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Por "animação de fluidos"entendemos a geração de uma seqüência de imagens digitais ilustrando a evolução do fluido ao longo de um intervalo de tempo. Este conjunto de imagens deve ser capaz de reproduzir o movimento do fluido de forma convincente, ou seja, o fluido deve escoar com o grau de realismo necessário para o contexto da animação que está sendo gerada.

Dentro desta perspectiva, a animação de fluidos tem natureza multidisciplinar, e seu desenvolvimento depende da interação entre profissionais das áreas de computação e engenharia. Objetivamente, para atingir o grau de realismo necessário, é preciso que as equações de fluidos sejam convenientemente tratadas e que as técnicas de visualização de dados utilizadas possam gerar imagens com a aparência realista. As técnicas utilizadas no processo de visualização são oriundas da computação gráfica. As equações de fluidos são derivadas de conceitos da mecânica clássica. É oportuno fazer a seguir alguns comentários gerais destas áreas. Nos capítulos seguintes, estes temas serão devidamente detalhados.

Dificilmente este sistema apresenta solução analítica, o que obriga os especialistas a lançarem mão dos recursos da análise numérica e de métodos numéricos para discretizar o problema e obter uma solução com um nível de aproximação aceitável.

No caso particular de cenários envolvendo animações de fluidos, a utilização de métodos em Dinâmica de Fluidos Computacional (DFC) tem se mostrado um recurso valioso para os animadores, diminuindo o custo, bem como o tempo de produção de filmes, tais como The Prince of Egypt [Witting (1999)] ou Shrek [Enright (2002)].

A necessidade de se buscar métodos em DFC para auxiliar a geração de efeitos visuais envolvendo fluidos vem da dificuldade para executar tal tarefa, com o grau de realismo necessário, com métodos puramente geométricos. Fluidos são sistemas com um número

muito elevado de graus de liberdade, o que torna impraticável animá-los com realismo sem fazer uso de princípios físicos para guiar o trabalho de designers experientes.

Os métodos encontrados na literatura para animação de fluidos, baseados em modelos de DFC, são fundamentados nas equações de Navier-Stokes, com técnicas de discretização baseadas em diferença finitas implícitas [Stam (1999)] e explicitas [Foster-Metaxas (1997)], bem como em métodos Lagrangeanos tais como Método das Características [Stam (1999)], Smoothed Particle Hidrodynamics (SPH) [Desbrun-Gascuel (1998)] e Moving-Particle Semi-Implicit (MPS) [Premoze at al.(2003)].

O método SPH é independente de malhas (assim como o MPS) e vem sendo aplicado com sucesso em problemas de engenharia, tais como simulação de fluidos compressíveis, para a descrição da dinâmica de materiais, estudo de explosões e fenômenos de transporte [Liu at al. (2003)]. Implementações deste método para arquiteturas paralelas têm sido desenvolvidas com o objetivo de melhorar a performance do mesmo.

Do ponto de vista da computação gráfica, o fato do SPH não depender de uma malha previamente definida é uma vantagem, se comparado com os métodos clássicos. As cenas em um filme são em geral dinâmicas, com fronteiras variáveis em função de novos objetos que entram na cena à medida que esta se desenvolve. Desta forma, métodos que não fazem uso de malhas, podem ser vantajosos por evitar o custo extra de re-gerar a malha sempre que tais alterações ocorrem. Estas vantagens vêm despertando interesse por pesquisadores em computação gráfica, e propostas enfocando a utilização do SPH para a geração de efeitos visuais já podem ser encontradas na literatura [Müller at al. (2003)], [Amada(2004)].

As pesquisas em animação de fluidos se dividem basicamente em três etapas. Primeiramente, temos a busca de novos modelos em DFC que sejam mais eficientes do ponto de vista da computação gráfica. Esta etapa envolve tanto a pesquisa de novos modelos físicos quanto o ajuste de modelos já conhecidos, sem perder de vista o fato de que o objetivo final é a geração de efeitos visuais, e não a descrição de fenômenos naturais.

1.1 Contexto e Motivação

teste 2.7 *italico-asdfasf* teste-ifem **negrito** conforme a imagem **??**

1.2 Objetivos e Contribuição

1.3 Organização do Texto

2 REVISÃO DE LITERATURA

2.1 Steering Behaviors

Em seu estudo de 1987 [11] Reynolds desenvolveu uma metodologia diferente dos caminhos pré programados existentes no tratamento comportamentais ligados a grupos de indivíduos. A simulação de grupos pode ser relacionado como uma modificação de um sistema de partículas e sua simulação é criada através de um sistema distribuído de um modelo comportamental, sistema esse parecido com um sistema natural de movimentação em grupo. A movimentação e escolha de caminho é feita através da percepção do ambiente em que o mesmo é inserido, as regras físicas e de movimentação e por um conjunto de comportamentos programados.

A utilização de forças para direcionar elementos em uma simulação de grupo foi proposta por Reynolds em 1999 [11] chamando-a de *steering behavior* a qual é uma versão melhorada de seu estudo de 1987. As simulações podem ser utilizadas em comportamentos como: busca, perseguição, fuga, perambular, aproximação, desvio de obstáculos e forças de direcionamentos relacionadas a grupo de personagens, aonde modelo proposto é estruturado em três forças, as quais direcionam os elementos do grupo individualmente baseado na velocidade e posição dos elementos vizinhos, essas forças são: separação, alinhamento e coesão.

O comportamento de personagens autônomos podem ser dividido em camadas comportamentais para melhor compreensão. Essas camadas podem ser vistas na figura 2.1, são elas: seleção de ação, direcionamento e locomoção.

- Selecão de ação: é responsável pela percepção do mundo e determinação de objetivos;

- Direcionamento: decompõem os objetivos em sub tarefas de movimentação (aprox-

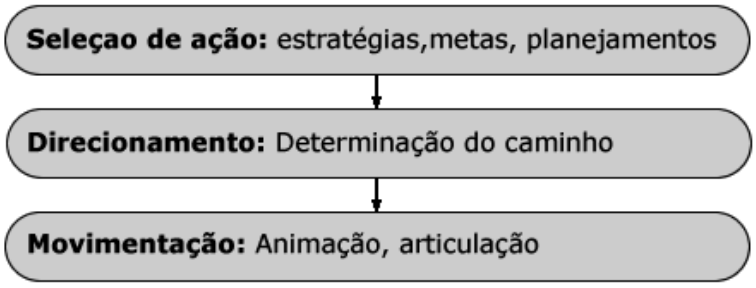

Figura 2.1: Hierarquia do comportamento de movimentação

imação do objetivo, desvio de obstáculos);

- Locomoção: utiliza as informações passadas pela camada de direcionamento para realizar o deslocamento físico do personagem.

O steering behavior é focado na camada do central de direcionamento o qual é responsável pelas forças que atuam sobre o elemento a fim de que ele alcance o objetivo inicial definido na camada de seleção de ação. Como forma de representar essa camada foi utilizado um modelo veicular simples. Esse modelo é simples o suficiente para representar os mais variados tipos de transporte ou formas de movimentação que se deseja.

O veículo é baseado em uma aproximação do ponto de massa. Essa abordagem proporciona um simples e computacionalmente barato modelo físico, porém o mesmo não pode ser considero um modelo físico completo pois o mesmo é capaz de representar o momento linear (velocidade) mas não é capaz de representar o momento rotacional pois o veículo é representado por um ponto de massa não dimensional.

Um ponto de massa é definido pelas propriedades *position* e *mass* que representam respectivamente a posição e a massa do elemento. O veículo ainda possui a propriedade *velocity* representando a velocidade, a velocidade é modificada pela aplicação de forças. As forças e a velocidade aplicada ao veículo possuem um limitador, sendo esse a representação das limitações físicas do próprio veículo, como a aceleração, representada pela propriedade *max_force* e uma limitação de velocidade, causada pelo atrito ou outros fatores, representado pela propriedade *max_speed*. A orientação (*orientation*) representa a direção do veículo a qual junto com posição do veículo representa a coordenada espacial na qual o modelo geométrico do veículo pode ser anexado.

```
   Simple Vehicle Model:
mass scalar
```

```
position vector
velocity vector
max_force scalar
max_speed scalar
orientation N basis vectors
```

A física do modelo veicular simples é baseada no FORWARD EULER INTEGRATION. A cada iteração da simulação, as forças determinadas pelo comportamento são aplicadas ao ponto de massa do veículo. Isso produz uma aceleração igual a força de deslocamento dividido pela massa do veículo. A aceleração é adicionada a antiga velocidade produzindo uma nova velocidade, a qual é truncada por max_speed. E por fim a velocidade é adicionada à antiga posição do veículo.

```
   steering_force = truncate (steering_direction, max_force)
acceleration = steering_force / mass
velocity = truncate (velocity + acceleration, max_speed)
position = position + velocity
```

O modelo veicular simples mantém-se alinhado com a velocidade por ajustes incrementais das iterações prévias. O sistema local de coordenadas é definido em por quatro vetores: vetor posição especificando o local da origem, e três vetores de direção servindo como base vetorial do espaço. A base vetorial indica a direção e comprimento das unidades de coordenadas, no qual três direções mutualmente perpendicular relativas ao veículo. Esses eixos serão referenciados por *forward, up* e *side* (esses correspondem aos eixos x, y e z do R3).

Para manter o alinhamento com a velocidade em cada iteração, o vetor base deve ser rotacionado para a nova direção. Ao invés de usar rotações explicitas, o sistema local é reconstruído usando a combinação de substituição, aproximação e reortogonalização. A nova velocidade é utilizada para calcular a nova direção e uma nova aproximação para a nova direção de *up*. Utilizando o produto vetorial é reconstruído o no sistema vetorial base:

```
   new_forward = normalize (velocity) approximate_up = normalize
(approximate_up) // if needed new_side = cross (new_forward,
approximate_up) new_up = cross (new_forward, new_side)
```

A ideia básica é que o *up* aproximado é quase perpendicular a nova direção de *forward*,

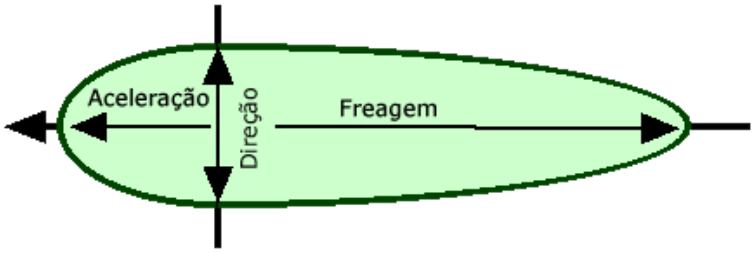

Figura 2.2: Forças de direcionamento assimétricas

porque as diferenças a cada iteração da orientação são tipicamente pequenas. A nova direção *side* será perpendicular ao novo *forward*, pela definição do produto vetorial. O novo *up* será o produto vetorial entre *forward* e *side* logo é perpendicular a cada um deles.

O conceito do alinhamento da velocidade não especifica somente uma orientação. O grau de liberdade correspondente a rotação sobre o eixo *forward* (também conhecido como inclinação) permanece não limitado. Construindo um novo espaço local relativo ao primeiro é garantido que a inclinação permanece consistente. Definir o valor correto de inclinação requere heurísticas futuras, baseada na intenção de uso do modelo veicular.

Nesse sistema veicular simples, o sinal de controle passado da camada comportamental de direcionamento para a camada de movimentação consiste em exatamente um vetor quantidade: uma força de direcionamento desejada. Mais realísticos modelos veiculares poderão ter vários diferentes conjuntos de sinais de controles. Por exemplo um automóvel teria um volante de direção, acelerador e freio os quais cada um podem ser representados por quantidades escalares. É possível mapear um vetor força de direcionamento generalizado nesses sinais escalares: o componente *side* do vetor direcionamento pode ser interpretado como o sinal de direção, o *forward* pode ser mapeado para o acelerador caso positivo ou freio caso negativo. Esse mapeamento pode ser assimétrico, por exemplo um automóvel pode desacelerar através da frenagem muito mais rápido que acelerar através do impulso do motor como mostrado na Figura 2.2.

Por causa dessa concepção de alinhamento à velocidade, esse sistema veicular simples não pode simular efeitos como derrapagem. Além do mais esse modelo permite que o veículo gire com sua velocidade em zero, esse problema pode ser resolvido adicionando um limitador na mudança de orientação, ou limitando o componente de direcionamento lateral em velocidades baixas, ou simulado o momento de inércia.

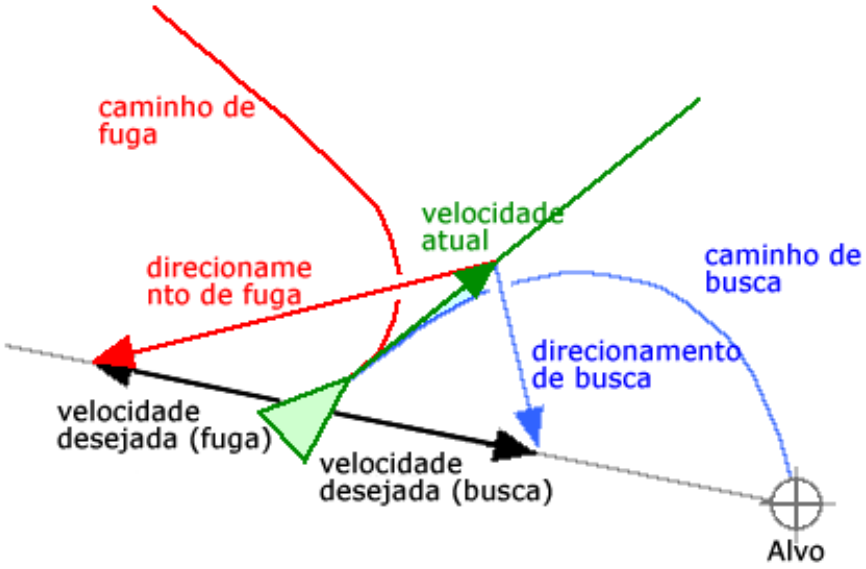

Figura 2.3: Comportamento de *seek* e *flee*

2.1.1 Modelos de *Steering Behaviors*

Para cada *steering behavior* específico assumiremos que a movimentação seja implementada pelo modelo veícular simples previamente descrito, o qual é parametrizada por um simples vetor força de direcionamento.

2.1.1.1 Seek

O comportamento *seek* (perseguição a um ponto estático) atua no direcionamento do personagem a uma posição fixa especificada no mundo virtual. Esse comportamento coordena o personagem em uma velocidade radialmente alinhada para o alvo. A velocidade desejada é um vetor na direção do personagem para o ponto de objetivo. O módulo da velocidade desejada pode ser `max_speed`, ou pode ser a velocidade corrente do personagem, dependendo da aplicação. O vetor de direcionamento é a diferença entre a velocidade desejada e a velocidade corrente do personagem conforme a Figura 2.3.

```
desired_velocity = normalize (position - target) * max_speed
steering = desired_velocity - velocity
```

Caso o personagem continue com o comportamento de *seek*, ele eventualmente passará pelo objetivo e após voltará para uma nova aproximação. Isso produzirá um movimento parecido com o movimento de moscas ao redor de uma lâmpada, diferente do comportamento de *arrival* a seguir.

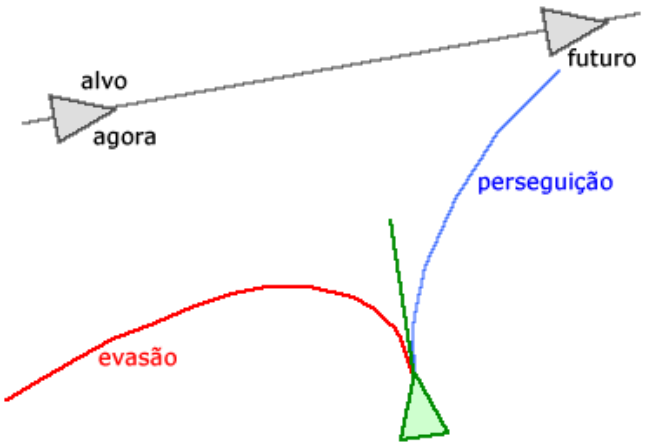

Figura 2.4: Comportamento de *pursuit* e *evasion*

2.1.1.2 Flee

O comportamento de *flee* é simplesmente o inverso do *seek*, atuando no direcionamento do personagem a se afastar de um ponto fixo especificado. A velocidade desejada apontará para a direção oposta formada entre o personagem e o ponto de objetivo verificado na Figura 2.3.

2.1.1.3 Pursuit

O comportamento de *pursuit* é similar ao *seek* exceto que o alvo é outro personagem móvel. Uma perseguição efetiva requer a previsão da futura posição do alvo. Uma das abordagens é usar um simples previsor que reavalia a cada iteração a futura posição do alvo. Como exemplo pode se usar um previsor linear basedo na velocidade o qual leva em conta que o alvo não mudara de direção durante o intervalo da previsão. O mesmo avalia a posição do personagem T unidades de tempo no futuro e ajusta a velocidade escalonando-a pelo período T previsto. O direcionamento de *pursuit* é um simples resultado da aplicação do comportamento de seek na posição prevista do alvo. Verifique a Figura 2.4.

A chave para a implementação do *pursuit* é o método usado para estimar o intervalo T de predição. No caso ideal o intervalo T deveria ser o tempo até a interceptação, mas o valor é desconhecido pelo fato do alvo realizar mudanças de rota imprevisíveis. T pode assumir um valor constante, o qual deve produzir melhor perseguição que o comportamento simples *seek* (o qual corresponde T=0). No entanto, para uma performance aceitável T

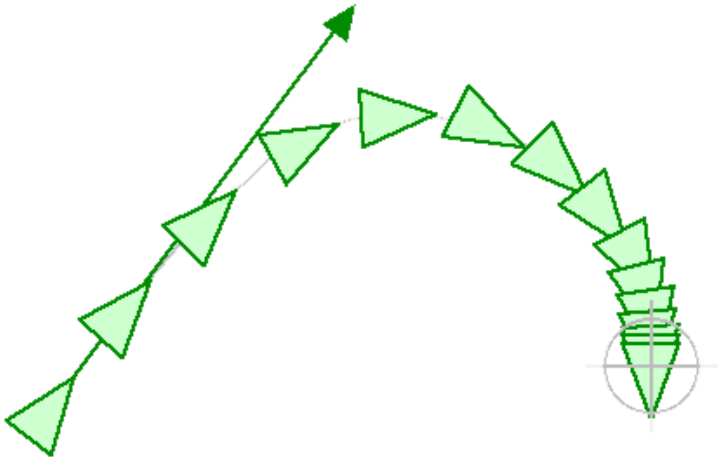

Figura 2.5: Comportamento de *Arrival*

deve ser maior quando o perseguidor está longe do objetivo, e menor quando ele está próximo. Outros métodos para estimar o valor de T podem ser utilizados dependendo do ambiente no qual será aplicado e do comportamento esperado.

2.1.1.4 Evasion

O comportamento de *evasion* é análoga a de *pursuit*, exceto que o comportamento de *flee* é utilizado para direcionar para longe da posição futura estimada.

As técnicas de perseguição e evasão dadas aqui tem a intenção de serem computacionalmente leves e são não-ótima, existem técnicas ótimas na literatura porém em um sistema natural a evasão é intencionalmente não-ótima com o objetivo de ser imprevisível, permitindo assim que frustre estratégias de perseguição previsíveis [2]

2.1.1.5 Arrival

O comportamento de *Arrival* é idêntico ao *seek* aonde o personagem está longe do objetivo. Porém ao invés de mover através do alvo com sua velocidade máxima, este comportamento causa uma diminuição de velocidade para o personagem conforme se aproxima do objetivo, eventualmente reduzindo a velocidade até parar no alvo, conforme mostrado na Figura 2.5. A distância a qual a desaceleração inicia é um parametro do comportamento. Esta implementação é similar ao *seek*: uma velocidade desejada é determinada direcionando o personagem para o objetivo. Fora do raio de parada a velocidade é cortada em `max_speed`, interior ao raio de parada, a velocidade é reduzida até atingir zero (ao encontrar o alvo).

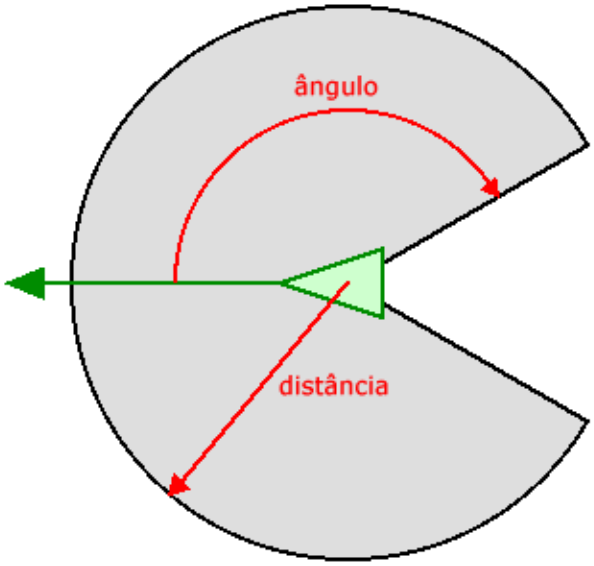

Figura 2.6: Vizinhança local

```
   target_offset = target - position
distance = length (target_offset)
ramped_speed = max_speed * (distance / slowing_distance)
clipped_speed = minimum (ramped_speed, max_speed)
desired_velocity = (clipped_speed / distance) * target_offset
steering = desired_velocity - velocity
```

2.1.2 Comportamentos de grupo

Os próximos três *steering behaviors*: *separation*, *cohesion* e *alignement*, são destinados a grupos de personagens. Em cada caso, o *steering behavior* determina como o personagem reage com outro personagem em sua vizinhança local. personagens fora da vizinhança local são ignorados. Como mostrados na Figura 2.6, a vizinhança é especificada por uma distância (*distance*) o qual define quando os dois personagens estão próximos, e um ângulo (*angle*) o qual define o campo de visão do personagem.

2.1.2.1 Separation

O comportamento de *separation* dá ao personagem a habilidade de manter certa distância de separação de outros personagens próximos. Esta poe se usada para prevenir que personagem se aglomerem. Para calcular a força de separação, primeiro é realizada uma busca para encontrar personagens dentro da vizinhança especificada. Esta pode ser uma

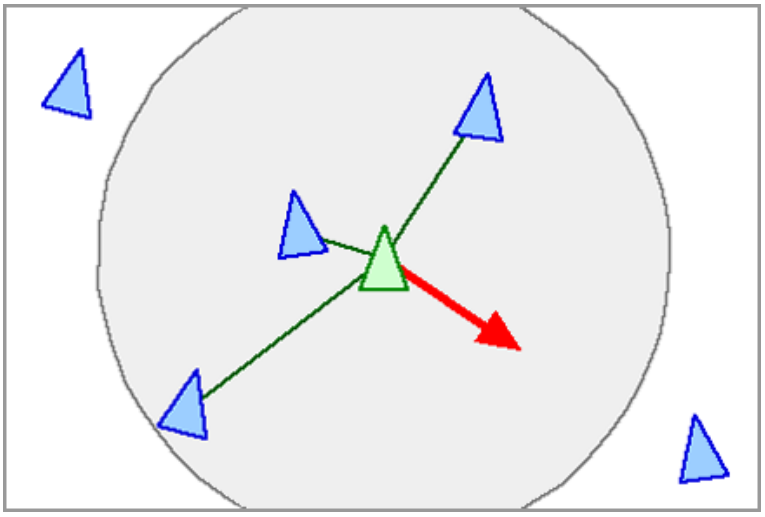

Figura 2.7: Comportamento de separação (*separation*)

busca exaustiva de todos os personagens da simulação, ou pode ser utilizado algum tipo de partição espacial ou um sistema de cache para limitar a busca a personagens locais. Para cada personagem local, a força de repulsão é calculada pela subtração das posições entre o personagem e seus vizinhos, normalizando e aplicando um peso inversamente proporcional à distância entre eles (r). A força de repulsão para cada personagem próximo é somada para produzir uma força de direcionamento global, veja a Figura 2.7.

2.1.2.2 Cohesion

O comportamento de *cohesion* da ao personagem a habilidade de se aproximar de um grupo formado de outros personagens próximos, conforme Figura 2.8. No direcionamento para a coesão é realizada a busca de todos os personagens da vizinhança local (conforme descrito previamente para a separação na seção 2.1.2.1), calculando a posição média do grupo de personagens. A força de direcionamento pode ser aplicada na direção da posição média ou pode ser usada como objetivo para o comportamento de *seek* 2.1.1.1.

2.1.2.3 Alignment

O comportamento de *alignment* da ao personagem a abilidade de se alinhar com outros personagens proximos, como mostrado na Figura 2.9. O direcionamento de alinhamento é calculado buscando todos os personagens da vizinhança local (conforme descrito previamente para a separação na seção 2.1.2.1), calculando a média dos vetores velocidade de todos os personagens próximos. Essa média é a velocidade desejada, assim o vetor de direcionamento é a diferença entre a média e a velocidade atual do personagem. Esse

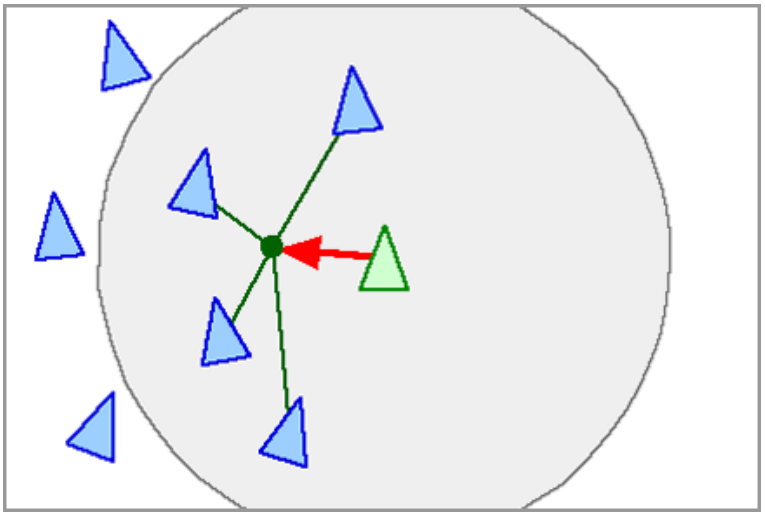

Figura 2.8: Comportamento de coesão (*cohesion*)

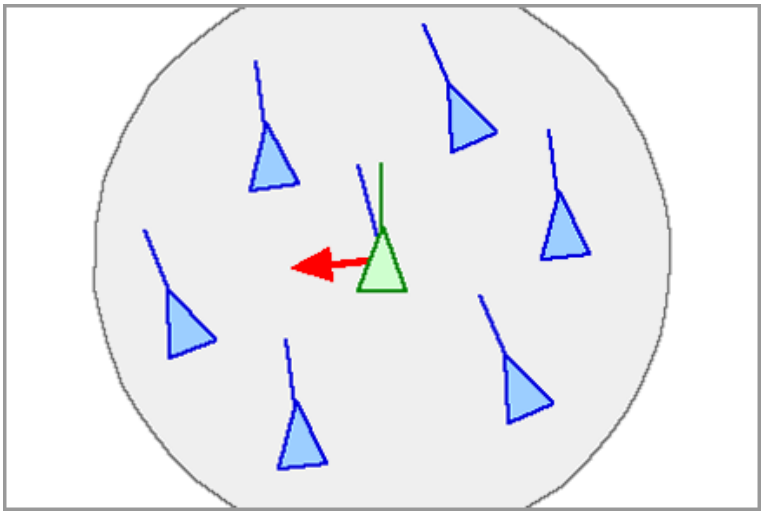

Figura 2.9: Comportamento de alinhamento (*alignment*)

direcionamento tenderá a girar nosso personagem alinhando com o grupo.

2.1.2.4 Flocking

O comportamento de *flocking* é obtido pela combinação dos comportamentos de separação, coesão e alinhamento. Essa combinação é capaz de produzir modelos comportamentais como multidão, rebanho e cardumes [10]. Para algumas aplicações é suficiente a simples soma entre os três vetores para produzir uma simples direcionamento de *flocking*. Porém para um melhor controle os três componentes são normalizados e posteriormente escalonados por três fatores individuais antes de soma-los. Como resultado, o comportamento de *flocking* é especificado por nove parametros numéricos: um peso (colaboração individual do comportamento), uma distância e um angulo (para determinar a vizinhança,

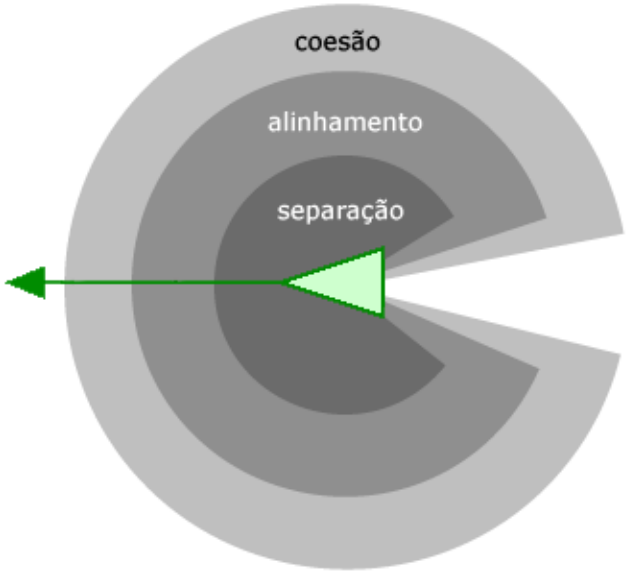

Figura 2.10: Comportamento de (*flocking*)

veja Figura 2.6) para cada um dos três comportamentos, conforme a Figura 2.10.

2.2 Simulação de fluidos

A Dinâmica de Fluidos Computacional (DFC) teve por origem no trabalho de Claude Navier (1822) e George Stokes (1845) os quais formularam a famosa equação de Navier-Stokes que descreve a conservação do momento. Em complemento a esta equação, duas equaçoes adicionais são necessárias para simular fluidos, uma descreve a conservação de massa e outra a conservação de energia. Uma vez definidas as equações é possível empregar a tecnologia de hardware para soluciona-las numericamente. Técnicas de simulação de fluido para propósitos específicos vem sendo desenvolvidas no campo de computação gráfica. Em 1983, [9] introduziu o uso do sistema de partículas como técnica para modelar uma classe de objetos *fuzzy*. A partir desse várias aplicações utilizando abordagem Euleriana (baseada em malhas) e Lagrangeana (baseada em partículas) na simulação de fluidos para computação gráfica. Surgiram uso de partículas para animação de objetos deformáveis [3] e [16], animação de superfícies [1] e fluxo de lava [14]. A abordagem Euleriana tem se mostrado mais popular nos últimos anos para a simulação de fluidos em geral [13], água [5], [4], [15], objetos deformáveis [8]. Porém ainda existem poucas técnicas disponíveis para uso em sistemas interativos. O trabalho baseado em malha de Stam [13] foi certamente um passo importante para a simulação de fluidos em tempo real.

2.2.1 Baseadas em Malha (Eulerian)

Stable fluids

2.2.2 Baseadas em Partículas (Lagrangian)

SPH (smoothed particle hydrodynamics)

3 PROPOSTA

3.1 Entendimento do Problema

A proposta desse trabalho visa utilizar a técnica de *steering behaviors* na simulação de fluido, como previamente explicado essa técnica utiliza a combinação de forças de direcionamento para direcionar os elementos em um cenário. Em contrapartida as técnicas tradicionais de simulação de fluídos baseado em partículas utilizam a transferência de energia entre as partículas para realizar a simulação, as constantes trocas de forças (ou energia no caso da simulação física) entre os elementos movimentam as partículas.

A simulação de fluidos para computação gráfica não é baseado em realismo mas sim no resultado visual, assim a interação entre as partículas de *behaviors* adequando corretamente as forças podem resultar em uma simulação coerente para a utilização da mesma em ambientes tridimensionais de jogos ou animações. As forças e os próprios steerings devem ser modificados ou ajustados para que o resultado adequado seja alcançado.

Como futuro trabalho pode ser vinculado a este relacionando os *behaviors* da simulação com propriedades dos fluídos como exemplo: quais forças devem ser alteradas para mudarmos o comportamento de viscosidade do fluído, se é possível alterar a viscosidade trabalhando com as forças e o tamanho da região de interação dos *behaviors* ; ou a possibilidade de simularmos a interação de fluidos de densidades diferentes alterando as regiões de vizinhança dos elementos as forças que parametrizam essas regiões. O objetivo desse trabalho é de relacionar a possibilidade de simular algumas características ou casos específicos de fluidos utilizando a metodologia de direcionamento por forças.

3.1.1 Relação entre as abordagens

A interação de um corpo estranho com a superfície do fluído segue a ordem: repulsão das partículas que formam a superfície do fluído, afastamento do volume liquido para a

entrada do novo corpo, retorno do fluido e preenchimento do espaço deslocado.

A repulsão pode ser relacionado com o comportamento de repulsão do *steering behaviors* aonde os elementos são repelidos ao entrarem na região de proximidade limite, o afastamento do fluído é similar à repulsão da primeira etapa, diferenciando na força de repulsão aplicada, esse comportamento pode ser melhor visualizado na sequência de imagens da Figura [adicionar figura aqui]. A diferença de forças funcionará de forma automática por causa da velocidade de aproximação do corpo estranho, quanto maior a velocidade mais próximo do elemento o mesmo se situará ao ser computado as forças que atuarão sobre o elemento fazendo com que a força de repulsão seja relativamente alta quando comparado com a próxima etapa aonde ocorre a diminuição na velocidade do novo corpo (reação a força de repulsão dos elementos que já se encontram no recipiente) e os elementos são repelidos quando estão mais afastados.

O retorno se da de forma similar a repulsão, como as partículas foram comprimidas com o deslocamento lateral as mesmas saem de uma região de estabilidade e são forçadas a entrar em uma região de repulsão entre os elementos vizinhos fazendo com que eles sejam forçados a se reorganizar e preencher os espaços vazios do recipiente.

3.1.2 Representação

A representação realista é muito importante para uma melhor visualização dos elementos, porém a mesma é somente deve ser levada em consideração após um resultado satisfatório na simulação. A representação dos elementos será dada através de esferas, as esferas representarão as partículas do fluido as quais por métodos previamente utilizados por Müller [7] podem ser usadas para formar uma superfície do fluido as quais podem ser vistas na Figura 3.1. Na Figura 3.1 (a) verificamos as partículas utilizadas na simulação realizada, em (b) foi utilizada a técnica de *marching cubes* [6]. O trabalho tem por objetivo buscar o realismo da simulação de fluidos para computação gráfica e não de representação visual o qual pode ser tratado posteriormente, por esses motivos as técnicas de criação de superfícies não serão tratadas nesse trabalho.

3.1.3 Definição dos steerings

Falar dos steerings utilizados e como eles afetam os elementos Falar da interação dos elementos

Figura 3.1: Formas de representação de fluídos.

3.1.4 Controle da Entropia

Falar do problema de adição constante de energia ao sistema, e falar que é uma característica do próprio sistema de steering behaviors criar novas forças para simular o comportamento dos elementos, sendo necessário um controle para que o sistema se estabilize.

3.2 Estratégias de implementação

Como base de implementação é necessário a implementação dos comportamentos de direcionamentos necessários como base programacional da solução proposta, surgiu a proposta da criação de um sistema de behaviors próprio o qual resultou na compreensão do funcionamento da biblioteca OpenSteer. O OpenSteer é uma biblioteca desenvolvida em C++ destinada a ajudar a implementação de steering behaviors para personagens autônomos aplicado a jogos e animação. A escolha do uso da biblioteca OpenSteer se deve pelos seguintes fatores: implementação validada e testada [citation needed], desenvolvido pelo Craig Reynolds criador da técnica de *steering behaviors*, sua implementação abrange as especificações e técnicas especificadas no trabalho original [11].

3.2.1 OpenSteer

Esta seçao resumirá os componentes relevantes do fremework desenvolvido criado por Reynolds denominado OpenSteer. OpenSteer é uma biblioteca *open source* de componentes que facilita a construção de ambientes para jogos e simulações multi-agentes que façam uso de *steering behaviors* para personagens autônomos. Esses agentes podem representar personagens (humanos, animais), veículos (carros, aviões, espaçonaves) ou outros tipos de agentes móveis. OpenSteer é multiplataforma, suportando atualmente Linux, Windows e Mac OS X, informações não encontrado nesse documento podem ser

encontradas na página do projeto [12].

OpenSteer prove um kit de ferraments para *steering behaviors*, definidas em termos deum agente movel chamado *vehicle*. Para permitir uma facil integraçao com engines de jogos existentes o OpenSteer pode ser adicionado como uma camada ou por herança de código. O mesmo foi criado para ser utilizado por programadores e não da suporte como ferramenta interativa de produção.

3.2.1.1 OpenSterDemo

Em adição a biblioteca, OpenSteer fornece uma aplicação interativa chamada OpenSteerDemo o qual possui vários behaviors implementados e prove um ambiente que facilita a vizulização de resultados e facilidade para corrigir problemas através de sua interface. O OpenSteerDemo é desenvolvido em C++ e usa a API Gráfica OpenGL e todas as chamas OpenGL são separadas em um módulo possibilitando a troca da mesma caso seja necessário.

O OpenSterDemo é baseado em uma arquitetura *plug-in*, isso é, módulos podem ser facilmente adicionados. Um *plug-in* para OpenSteerDemo especifica várias ações genéricas requeridas pelo OpenSteerDemo framework: `open, close, reset, run one simulation step, display one frame, etc`, o mesmo também define as classes veícular e gerencia a simulação.

O sistema de *plug-in* do OpenSteerDemo permite que o desenvolvedor crie rapidamente um protótipo de um comportamento durante o *game design*.

O OpenSteerDemo possui um sistema de relogio interno com dois tipos de contagem diferentes: *real time* e *simulation time*. O tempo de simulação é normalmente segue o tempo real da simulação mas também pode ser pausada, congelando a simulação. Pausando a simulação permite examinar as informações em detalhes ou reposicionar a câmera. O relógio pode continuar sua execução de três formas diferentes conformes descritas abaixo:

- *Frame rate* variável: utilizado para visualização de simulações em tempo real. As atualizações ocorrem o mais rapido possível. Os passos da simulação são proporcionais ao tempo passado desde o último update;

- *Frame rate* fixo: utilizado para visualização de simulações em tempo real. As atualizações são forçadas a ocorrer em quantias fixas de *frame rate* aguardando até

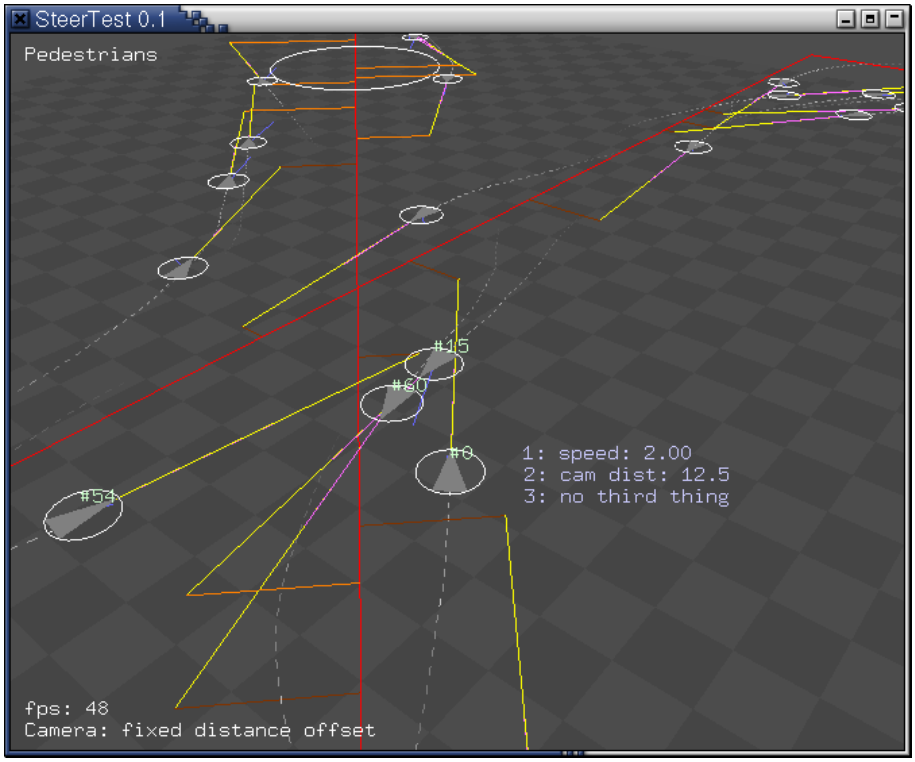

Figura 3.2: Simulação de pedestres utilizando OpenSteer

o início do próximo frame. Caso o update leve muito tempo o mesmo deve aguardar até o início da próxima janela de atualização;

- Modo de animação: Para executar passos fixos da simulação, geralmente não é em tempo real, ignorando o tempo real para produzir resultados consistentes da simulação.

A Figura 3.2 mostra uma simulação de pedestres em um ambiente bidimensional utilizando o framework. Esse exemplo faz parte dos *plug-in* que acompanham o OpenSteerDemo.

3.2.1.2 Steering Library

Wander behavior

```
Vec3 steerForWander (float dt);
```

Retorna a força para o comportamento de *wander* (perambular).

Seek behavior

```
Vec3 steerForSeek (const Vec3& target);
```

Retorna a força para o comportamento de *seek*, faz com que o veículo seja direcionado para o objetvo e o move para esse.

Flee behavior

```
Vec3 steerForFlee (const Vec3& target);
```

Retorna a força para o comportamento de *flee*, a qual faz com que o veículo se afaste do alvo.

Pursuit behavior

```
Vec3 steerForPursuit (const AbstractVehicle& quarry);
Vec3 steerForPursuit (const AbstractVehicle& quarry, const
float maxPredictionTime);
```

Retorna a força de direcionamento de perseguição a outro veículo em movimento

Evasion behavior

```
Vec3 steerForEvasion (const AbstractVehicle& menace, const
float maxPredictionTime);
```

Retorna a força de direcionamento para evadir da aproximação de outro veículo, se afastando do ponto de interceptação.

Separation behavior

```
Vec3 steerForSeparation (const float maxDistance, const float
cosMaxAngle, const AVGroup& flock);
```

Retorna a força para afastar o veículo dos elementos vizinhos. **Alignment behavior**

```
Vec3 steerForAlignment (const float maxDistance, const float
cosMaxAngle, const AVGroup& flock);
```

Retorna a força que alinha o veículo aos seus vizinhos.

Cohesion behavior

```
Vec3 steerForAlignment (const float maxDistance, const float
cosMaxAngle, const AVGroup& flock);
```

Retorna a força que tende a mover o veículo para o "centro de massa"do grupo de vizinhança.

Path Following behavior

```
Vec3 steerToFollowPath (const int direction, const float predictionTi
Pathway& path)
Vec3 steerToStayOnPath (const float predictionTime, Pathway&
path)
```

Retorna a força para para acompanhar um caminho predefinido. `steerToStayOnPath`

apenas tenta manter o veículo no caminho enquanto `steerToFollowPath` prove um acompanhamento direcionado de caminho.

3.2.1.3 Funcionalidades não implementadas

Alguns comportamentos principais detalhados no artigo de Reynolds [11] não foram implementadas ainda no OpenSteer. São eles: *offset pursuit*, *arrival*, *wall following*, *containment*, *flow field following* e *leader following*. Outros comportamentos citados de passagem no artigo não foram implementados.

O OpenSteer não prove uma restrição de não penetração em outros elementos (obstáculos e outros veículos) o que pode ocasionar falhas visuais na simulação.

Alguns componentes da API do OpenSteer não estão documentos além dos comentários existentes nos arquivos de cabeçalho (*header files*).

4 IMPLEMENTAÇÃO

4.1 Definição de forças

4.1.1 Gravidade

Para simulação de grupo de elementos em um sistema de steering behaviors a geralmente gravidade no mundo não é relevante para o resultado esperado, porém em uma representação gráfica de fluídos a gravidade é importante para o correto fluxo dos elementos e da representação correta dos elementos. A gravidade foi implementada como uma força que age constantemente sobre todos os elementos do sistema. Essa força faz com que os elementos formam um fluxo em direção em que força está atuando.

4.1.2 Coesão

A coesão consiste em uma região, geralmente mais externa, na qual os elementos tendem a se aproximar do centro do grupo. A coesão entre as partículas faz com que as mesmas tendam a não se separar do grupo, esse comportamento é esperado na representação de fluídos uma vez que as partículas tendem a manter-se unidas na simulação. Em nossa implementação estudamos a força de coesão em nossa simulação nas seguintes condições:

- Comportamento original: os elementos se atraem com igual força não importando em qual região de vizinhança o elemento se encontra, nesse caso em especial o comportamento de agrupamento dos elementos funcionava conforme esperado, porém com a força de separação 4.1.3 adicionando força de forma exponencial ao inverso da distância entre os elementos, o sistema reagia com divergência do funcionamento esperado.

- Força exponencial: os elementos são atraídos através de uma força exponencial

proporcional à distância entre os elementos do grupo, assim quanto mais afastado do centro e mais próximo da fronteira da região de vizinhança 4.1.5 maior será a força de atração entre os elementos, e quanto mais próximo da região de separação menor a força fazendo com que os elementos encontrem uma região de transição estável entre a força de separação e coesão. Esse comportamento atuará como um facilitador na solução do problema de estabilização das forças internas do sistema 4.1.7.2.

Além das modificações comportamental dos steering behaviors citadas, foram realizadas modificações na região de atuação da força e coesão e no seu peso na contribuição final na força resultante do processo de interação com os demais elementos. Com essas modificações espera-se chegar a representação da viscosidade de um fluido. Esse resultado é esperado pelo fato de que a região de coesão é determina a resistência em que os elementos são capaz de separar do grupo principal, quanto menor o for a colaboração na força final de interação mais fácil é da partícula se separar do grupo a qual ele pertence e quanto maior a sua contribuição, mais coeso é o grupo.

4.1.3 Separação

A força de separação tende a afastar os elementos quando internos à região de separação. A força de separação é um elemento importante na simulação de fluídos, a mesma vem em substituição da colisão das partículas na simulação clássica de partículas [CITAÇÃO FALTANDO]. Como no steering behaviors os elementos não trabalham com colisão é necessária uma força capaz de separar os elementos quando muito próximos ou quando em rota de colisão. Essa força é diretamente proporcional ao inverso da distância ao quadrado fazendo com que a mesma cresça rapidamente ao se aproximar de outro elemento, fazendo com que ambos sejam repelidos como resultado. A região de atuação da força de separação corresponde com a distância esperada em que as particulas permanecam separadas uma das outras, analogamente podemos relacionar com a densidade do fluído, pois quanto mais próximas as partículas se estabilizarem mais difícil sera desloca-las na interação com outros elementos, dado que a força de repulsão do grupo a um elemento novo é maior devido ao somatório das forças resultantes do grupo.

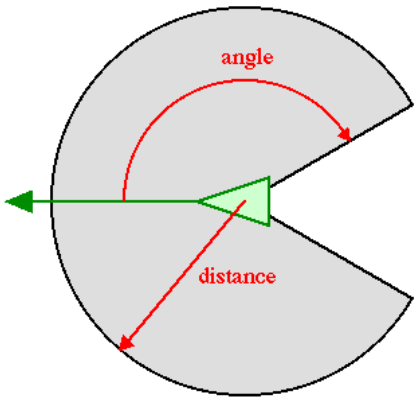

Figura 4.1: Região de vizinhança de um elemento

4.1.4 Alinhamento

A força de alinhamento direciona os elementos a se alinharem com o grupo que pertence a região de alinhamento. [Acredito que não sera utilizado]

4.1.5 Região de atuação da vizinhança

Na implementação dos steering behaviors cada elemento possui um campo de visão baseado no angulo definido de sua frente, essa região define qual a área em que é considerada na interação entre outros elementos. Os elementos fora dessa área ou raio não são relevantes para determinar as forças que estão atuando atualmente no elemento sendo descartados no processamento e restringindo a interação a um grupo de proximidade. Essa região pode ser visualizada na figura 4.1. A restrição da atuação é utilizada nos casos em que os elementos devem manter-se em movimentacao. TODO: CONTINUAR COM A IDEIA.

4.1.6 Interação com paredes do recipiente

4.1.7 Comportamento das forças

4.1.7.1 Atrito / Viscosidade

4.1.7.2 Estabilidade e entropia

O problema de instabilidade ocorre pela funcionamento básico do *steering behavior*, o mesmo não utiliza a energia existente no elemento na adição das forças de interação com a vizinhança, a força é criada fazendo com que o sistema não se estabilize. A criação de forças é uma característica intrínseca ao funcionamento do *steering behavior* fazendo

com que a mesma não possa ser alterada ou removida, sendo necessário desenvolver uma metodologia baseada em forças para que essas a entropia do sistema se estabilize. Algumas regras foram definidas para que o sistema tendesse ao equilíbrio, essas regras são de perda e adição de energia ao sistema.

- Perda de energia pelo sistema. Como no mundo real o atrito e a interação das moléculas fazem com que o sistema tenda a se estabilizar, em nossa abordagem os elementos perdem sua velocidade quando próximos a outros elementos (caracterizando a viscosidade) e quando interagindo com a parede do recipiente (caracterizando o atrito com outros elementos

- Adição de energia somente. A força somente será criada quando for necessária e guardando as proporções dos eventos.

5 RESULTADOS

6 CONCLUSÃO

teste *italico-asdfasf* teste-ifem **negrito** conforme a imagem 6

REFERÊNCIAS

[1] Jean christophe Lombardo and Claude Puech. Oriented particles: A tool for shape memory objects modelling. In *In Proc. Graphics Interface*, pages 255–262, 1995.

[2] D. Cliff and G. Miller. Co-evolution of pursuit and evasion ii: Simulation methods and results, 1995.

[3] Mathieu Desbrun and Marie paule Gascuel. Smoothed particles: A new paradigm for animating highly deformable bodies. In *EG Workshop on Animation and Simulation*, 1996.

[4] Douglas Enright, Stephen Marschner, and Ronald Fedkiw. Animation and rendering of complex water surfaces. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 736–744, New York, NY, USA, 2002. ACM.

[5] Nick Foster and Ronald Fedkiw. Practical animation of liquids. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 23–30, New York, NY, USA, 2001. ACM.

[6] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4):163–169, 1987.

[7] Matthias Müller, David Charypar, and Markus Gross. Particle-based fluid simulation for interactive applications. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 154–159, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[8] Daniel Nixon and Richard Lobb. A fluid-based soft-object model. *IEEE Comput. Graph. Appl.*, 22(4):68–75, 2002.

[9] W. T. Reeves. Particle systems - a technique for modeling a class of fuzzy objects. In *ACM Transactions on Graphics*, 1987.

[10] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *SIGGRAPH 1987, Computer Graphics*, 1987.

[11] Craig W. Reynolds. Steering behaviours for autonomous characters. In *Game Developers Conference 1999*, 1999.

[12] Craig W. Reynolds. Opensteer: Steering behaviors for autonomous characters, 2003.

[13] Jos Stam. Stable fluids. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 121–128, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[14] Dan Stora, Pierre olivier Agliati, Marie paule Cani, Fabrice Neyret, and Jean dominique Gascuel. Animating lava flows. In *in Graphics Interface*, pages 203–210, 1999.

[15] Tsunemi Takahashi, Heihachi Ueki, Atsushi Kunimatsu, and Hiroko Fujii. The simulation of fluid-rigid body interaction. In *SIGGRAPH '02: ACM SIGGRAPH 2002 conference abstracts and applications*, pages 266–266, New York, NY, USA, 2002. ACM.

[16] David Love Tonnesen. *Dynamically coupled particle systems for geometric modeling, reconstruction, and animation*. PhD thesis, Toronto, Ont., Canada, Canada, 1998. Adviser-Terzopoulos,, Demetri.