Universidade Regional de Blumenau
Centro de Ciências Exatas e Naturais
Departamento de Sistemas e Computação

FURB

Protótipo de um ambiente virtual distribuído multiusuário

Acadêmico: Leonardo Willrich
Prof. orientador: Dalton Solano dos Reis

Apresentação para defesa em banca do Trabalho de Conclusão de Curso da primeira fase do ano de 2002, para obtenção do título de Bacharel em Ciências da Computação

Tópicos abordados

- **Introdução**
  - Contextualização / objetivos
- **Fundamentação teórica**
  - Ambientes virtuais distribuídos
  - Modelos/Protocolos de comunicação
  - OpenGL
  - DIS-Java-VRML
  - Objetos Distribuídos/CORBA
- **Desenvolvimento do trabalho**
  - Ferramentas utilizadas
  - Especificação e implementação
  - Operacionalidade do Protótipo
- **Considerações finais**
  - Conclusões e extensões

Introdução - Contextualização

- Imagens em 3D torna a interface com o usuário mais próxima da realidade
- Surgimento dos ambientes virtuais (imersivos, não imersivos, mono ou multiusuário)
- Necessidade de apoio de outras ferramentas (Biblioteca gráfica OpenGL e DIS-Java-VRML)
- Necessidade comunicação entre objetos distribuídos (Padrão CORBA)

Introdução - Objetivos

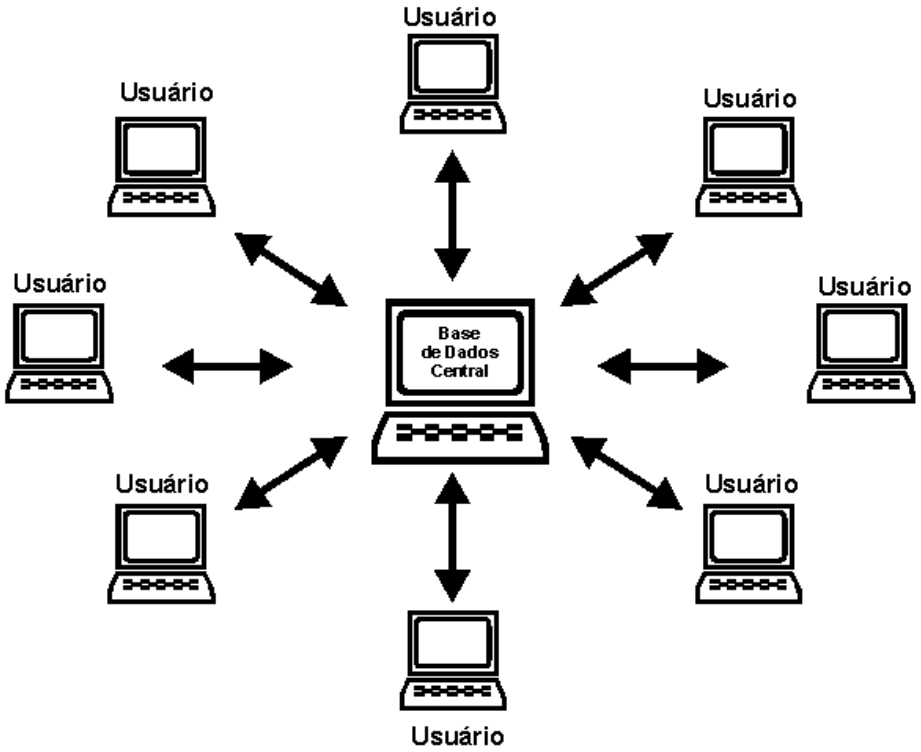
- **Implementar um protótipo de ambiente virtual distribuído sobre uma rede local, com suporte a multiusuários e com uma interface não imersiva**
- **Mais especificamente:**
  - Composto por um cenário e objetos simples
  - Permitir interação dos usuários com o ambiente virtual

Fundamentação – AVD's

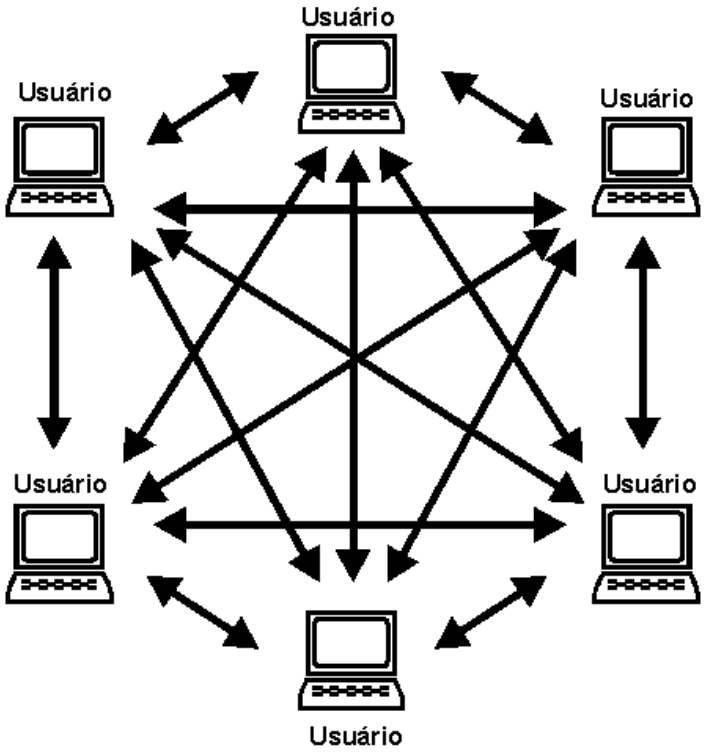
- **São simulações de computador interativas que submergem os usuários a uma realidade**
- **Primeiras experiências estão relacionadas a aplicações para simulações militares**
- **Projetos que servem como referência: SIMNET e NPSNET**
- **Resultaram como referência nos seguintes aspectos de construção de um AVD:**
  - modelo geral de comunicação
  - técnicas (*heartbeats)* e protolocos (*broadcast e* DIS) mais indicados

Modelos de Comunicação

Basicamente existem dois modelos de comunicação: o modelo centralizado e o distribuído.



Modelo Centralizado



Modelo Distribuído

Modelos de Comunicação

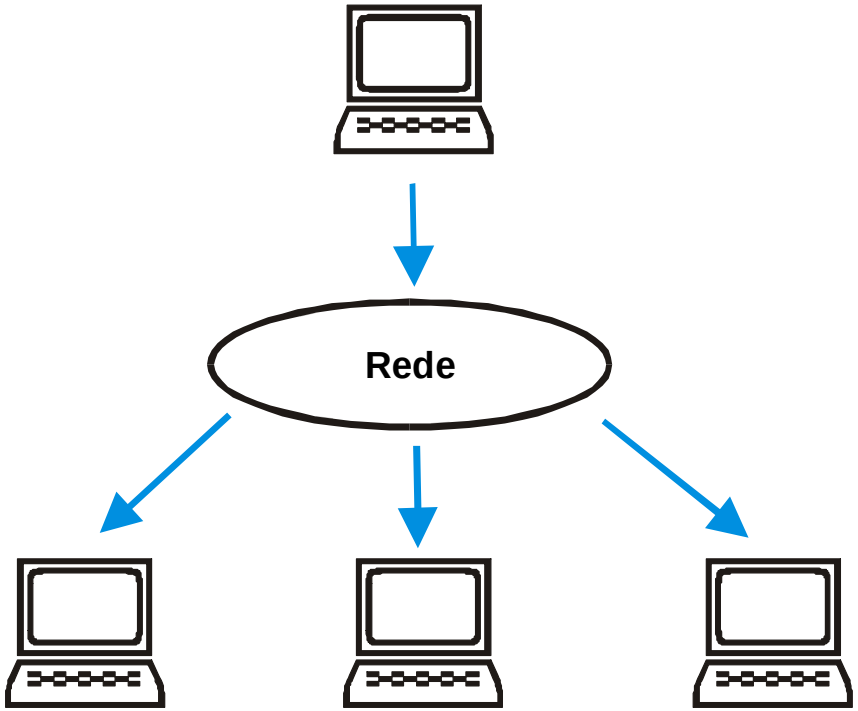
FURB

Modelo/Aspecto	Centralizado	Distribuído
Positivo	• Simplifica o mecanismo de controle da comunicação	• Cada usuário usa seus próprios recursos computacionais • Mais tolerante a falhas, evita congestionamento de mensagens sobre um único computador
Negativo	• Mais vunerável a falhas • Problema de escalabilidade • Sobrecarga de mensagens para o computador central	• Problema de geração excessiva de mensagens

Protocolos de comunicação

Protocolo de rede Broadcast:

- Não é necessário uma conexão para cada usuário
- Uma mensagem enviada, todos recebem
- Útil principalmente sobre LAN's
- Não aplicável a WAN's: UDP *broadcast* não roteável
- Não existe a garantia da entrega dos pacotes



Broacast (apenas uma mensagem enviada, todos recebem)

Protocolos de comunicação

Distributed Interaction Simulation (DIS)

- É um grupo de padrões definido pelo Departamento de Defesa dos Estados Unidas e indústrias interessadas
- Sua origem está ligada ao SIMNET
- Visa determinar uma arquitetura de comunicação comum para AVD's
- Consiste de PDU's (originalmente 27 tipos) para comunicar eventos e estados do AVD
- Originalmente definido no padrão IEEE 1278-1993

Técnica Associada

- ***Heartbeat***
  - Consiste em enviar PDU's de atualização de estado do usuário periodicamente para os outros usuários
  - Importante para objetos com baixa ou nenhuma freqüência de atualização no AVD (estáticos)
  - Essencial para manter novos usuários que entram no AVD atualizados
  - Causa um aumento na utilização da largura de banda

OpenGL

- **Características**
  - Significa *Open Graphics Library*
  - É uma biblioteca gráfica e de modelagem 3D
  - Desenvolvida pela Silicon Graphics para suas estações gráficas de trabalho - 1992
  - Portável: Independente de plataforma e de sistema operacional
  - Consistem em 150 comandos aproximadamente
  - Possibilita aos desenvolvedores produzirem animações 3D com simples linhas de comando

OpenGL

Vários jogos: QUAKE I, II, e III, Half-Life, Soldier of Fortune, e outros

Imagens do Jogo Half-Life (fonte: www.amazon.com)

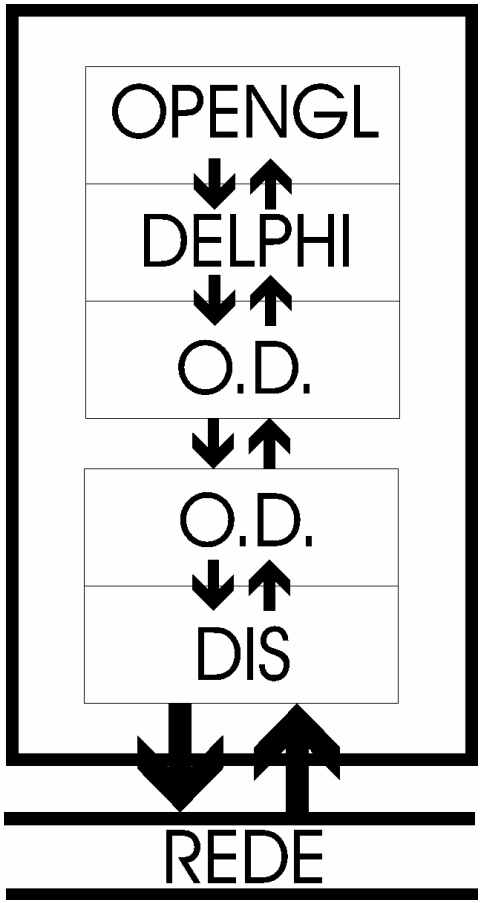
DIS-Java-VRML

FURB

- **Características**
  - Resultado de um dos grupos de trabalho do consórcio Web3D
  - API visa facilitar a construção de AVD's portáveis, oferecendo uma interface entre a linguagem Java e o ambiente virtual feito em VRML
  - Nelas estão contidas várias classes que implementam a maioria dos PDU's do protocolo DIS, bem como classes que controlam o processo de envio e recebimento dos mesmos

VRML - OpenGL

Substituição VRML – OpenGL



- Classes que interfaceiam Java e VRML foram desconsideradas
- Uso somente das classes que implementam os PDU's e mecanismos de controle de envio e recebimento dos mesmos
- Necessidade de objetos distribuídos em duas aplicações desenvolvidas em linguagens diferentes

Objetos Distribuídos

- **Características**
  - Localização: Podem estar todos em uma máquina ou em máquinas distintas de uma rede
  - São capazes de executar funções para um sistema distribuído
  - Teve sua origem com a orientação a objetos e as aplicações distribuídas
  - Os padrões mais conhecidos são o DCOM e o CORBA

CORBA

FURB

- **Características**
  - Padrão que define como objetos devem interoperar em um ambiente distribuído
  - Define e implementa a estrutura necessária para comunicação entre aplicações distribuídas em diferentes plataformas, sistemas operacionais e linguagens de programação.
  - Desenvolvido pela OMG
  - É um padrão aberto
  - Duas maneiras de implementar objetos:
    - SSI (*Static Invocation Interface*)
    - DII (*Dynamic Invocation Interface*)

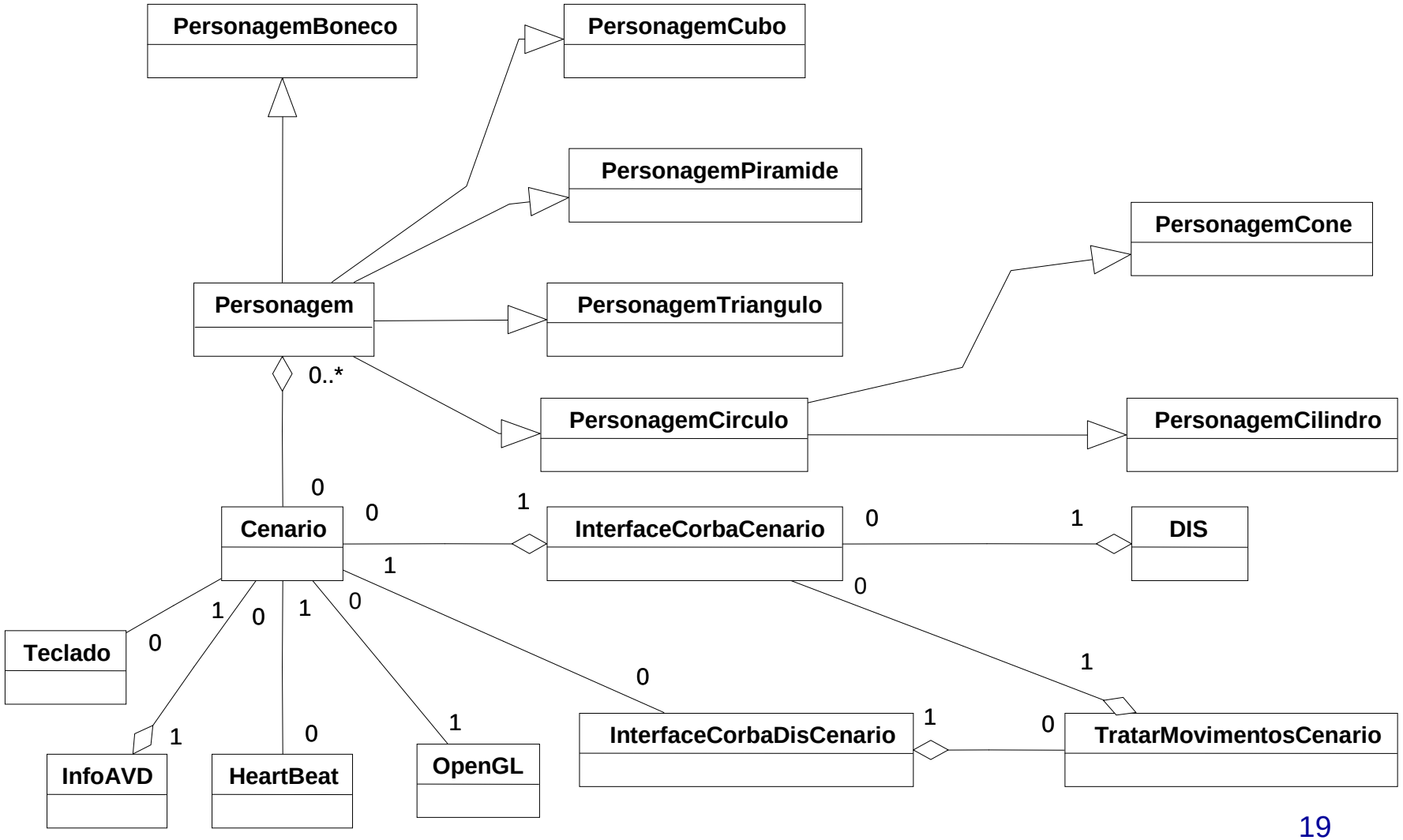
Desenvolvimento do protótipo

- **Objetivo:**
  - Ambiente virtual distribuído sobre uma rede local, com suporte a multiusuários e com uma interface não imersiva
- **Requisitos identificados:**
  - Modelo de comunicação distribuído
  - Envio de mensagens através de *broadcast* UDP
  - Protocolo de comunicação baseado no DIS
  - Comunicação entre objetos distribuídos através do CORBA
  - Criação de cenário e dos personagens com OpenGL
  - Entrada de dados no AVD
  - Utilização da técnica de *heartbeat*

Desenvolvimento do protótipo

- **Técnicas de especificação:**
  - Para representar as classes: **Diagrama de Classes**
  - Para lógica dos processos: **fluxogramas**
- **Linguagens de programação:**
  - Java
  - Object Pascal
- **Ferramentas:**
  - API J2SDK 1.3
  - API OpenGL 1.2
  - API DIS-Java-VRML
  - VisiBroker for Java 4.5.1

Diagrama de classes



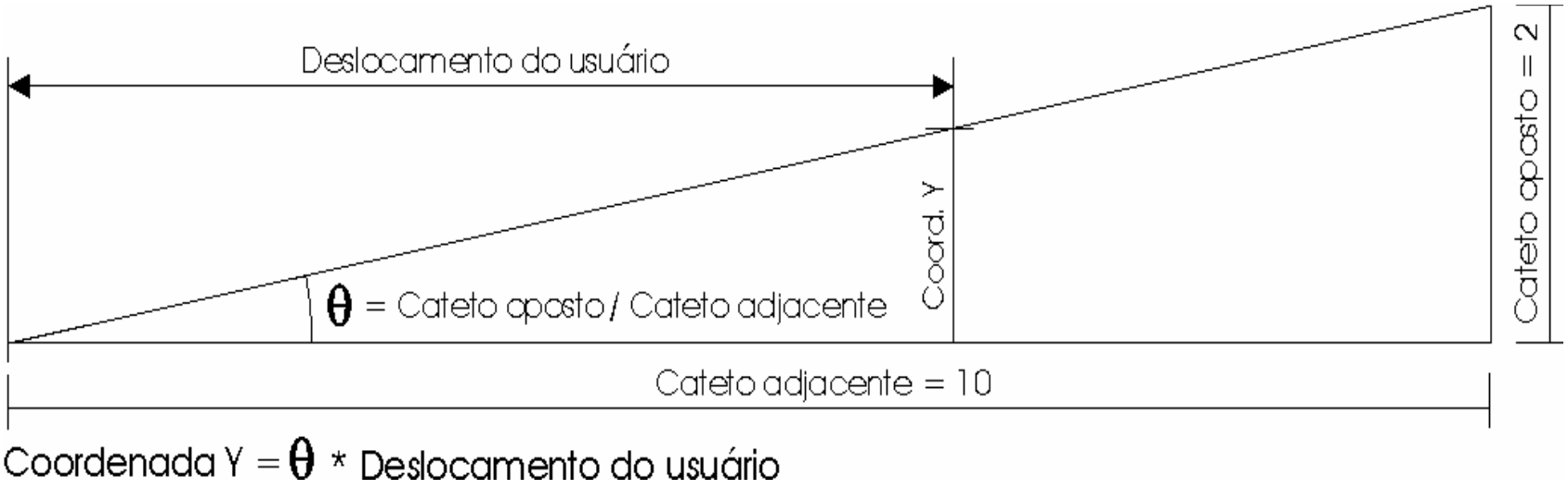
Classe Cenario

Cenario
DesenharCenario( ) DesenharPersonagens( ) CalculaY( ) CriaCenario( ) DestroiCenario( ) Desenhar( ) MoveuParaFrente( ) MoveuParaTras( ) MoveuParaEsquerda( ) MoveuParaDireita( ) OlhouparaCima( ) OlhouParaBaixo( ) AdicionarPersonagem( ) ExcluirPersonagem( ) AtualizarPersonagem( ) AtualizaCenarioDistribuido( ) InfoShow( )

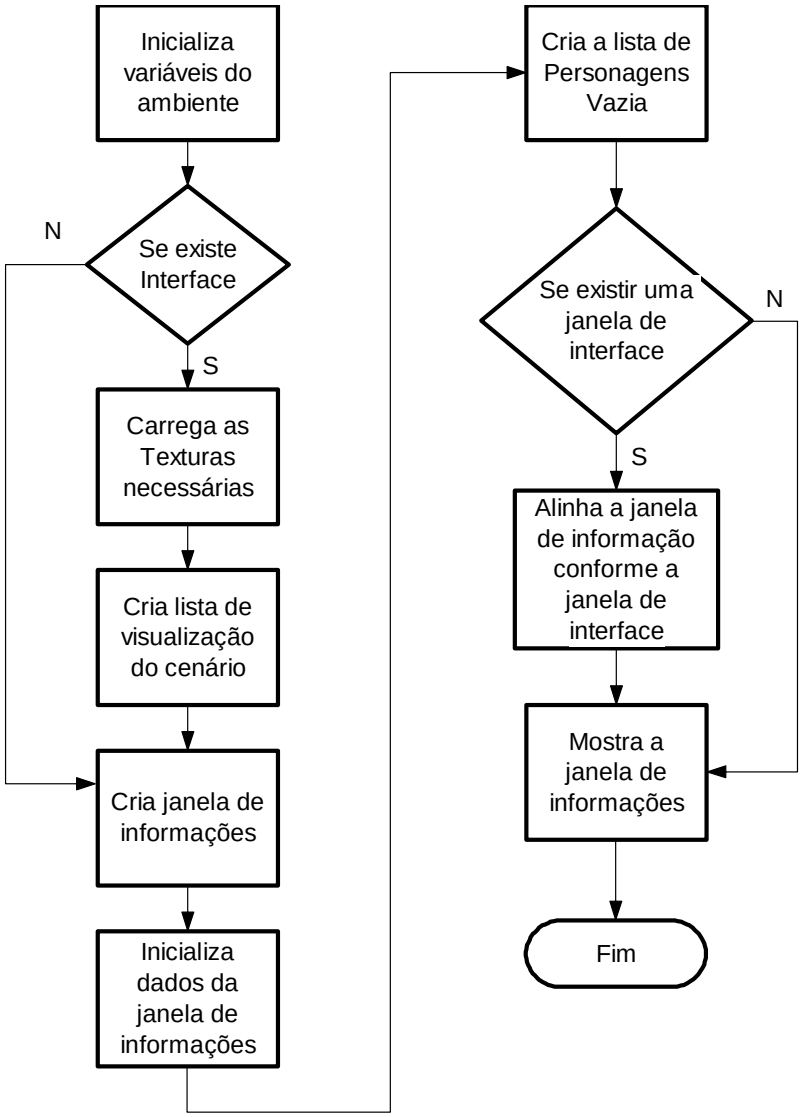
- Responsável pelo gerenciamento do cenário
- Agrega lista de personagens pertencentes ao cenário
- Contém informações do estado atual do usuário
- Atualiza a janela de informações
- Cada usuário terá instanciado um objeto desta classe ao entrar no AVD ou ao consultar informações do mesmo

Classe Cenario - Métodos

- Método CalculaY():
  - Composto de dois pisos com uma rampa de acesso
  - Se usuário está no piso superior ou inferior então Y será fixo
  - Se usuário está na rampa existe a necessidade de um calculo

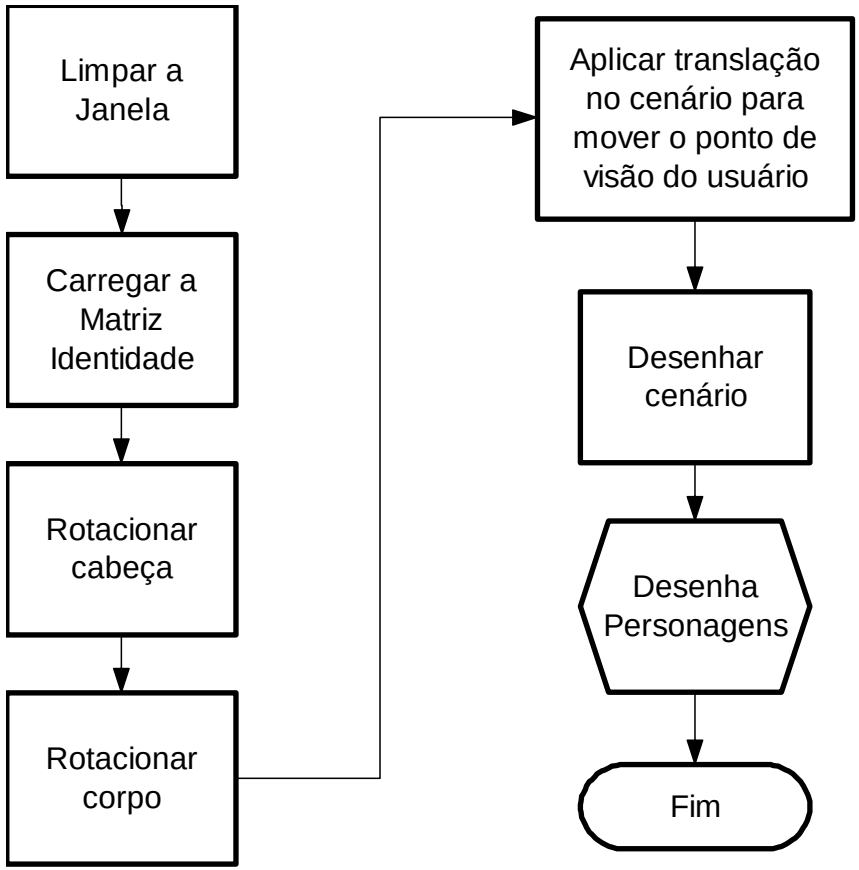


Classe Cenario - Métodos



- Método CriaCenario():
  - Este método é um construtor desta classe
  - Duas maneiras: com interface e sem interface

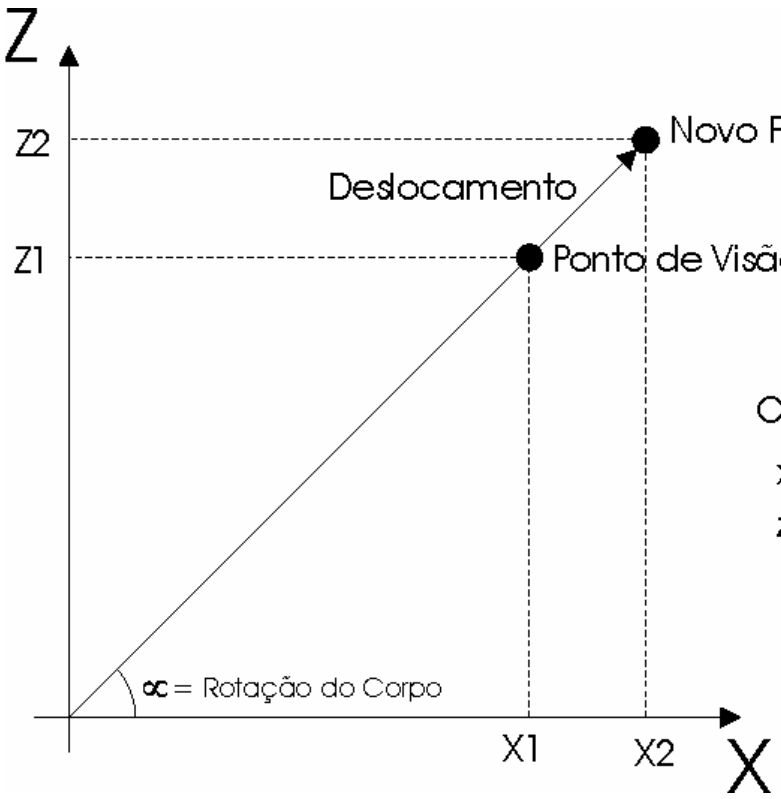
Classe Cenario - Métodos



- Método Desenhar():
  - Desenha o cenário por completo
  - Utilizado quando o cenário sofre alguma alteração

Classe Cenario - Métodos

- Métodos MoveuParaFrente() e MoveuParaTraz():
  - Calcula-se novas coordenadas através da rotação do corpo

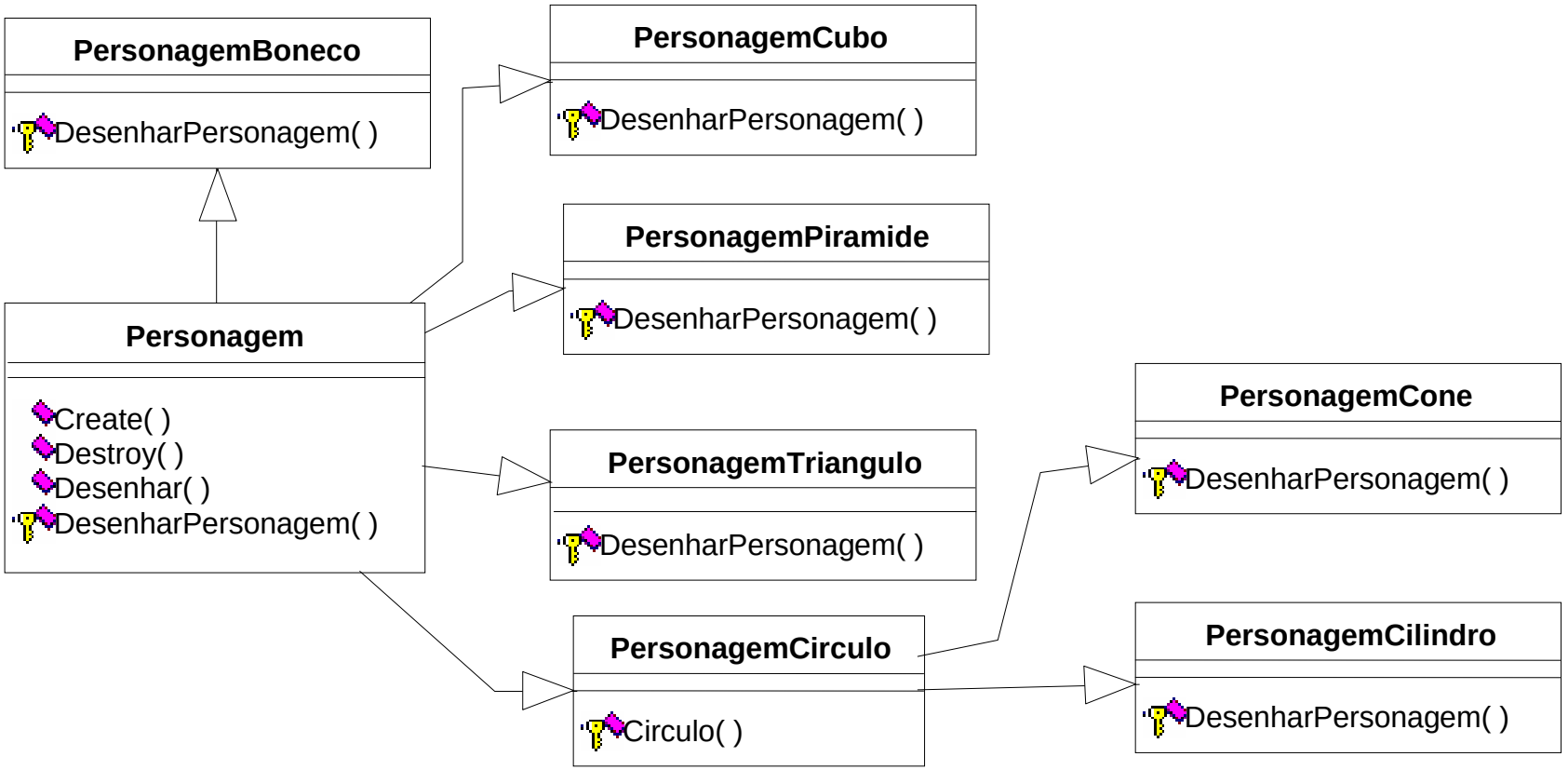


Onde:

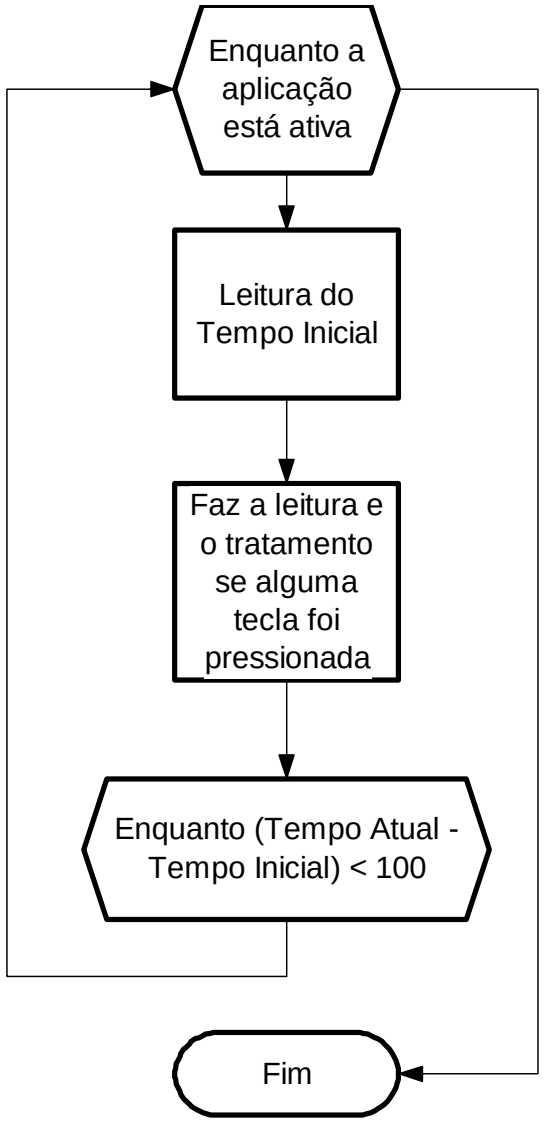
x2 = x1 (+ ou -) (seno(∝) * Deslocamento)

z2 = z1 (+ ou -) (coseno(∝) * Deslocamento)

Classe Personagem/herdadas

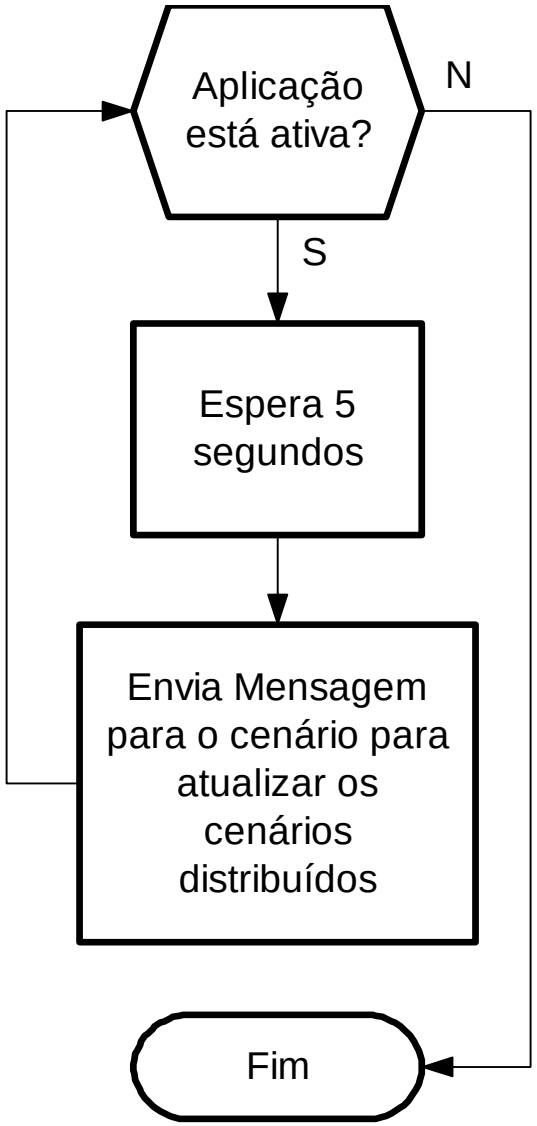


Classe Teclado



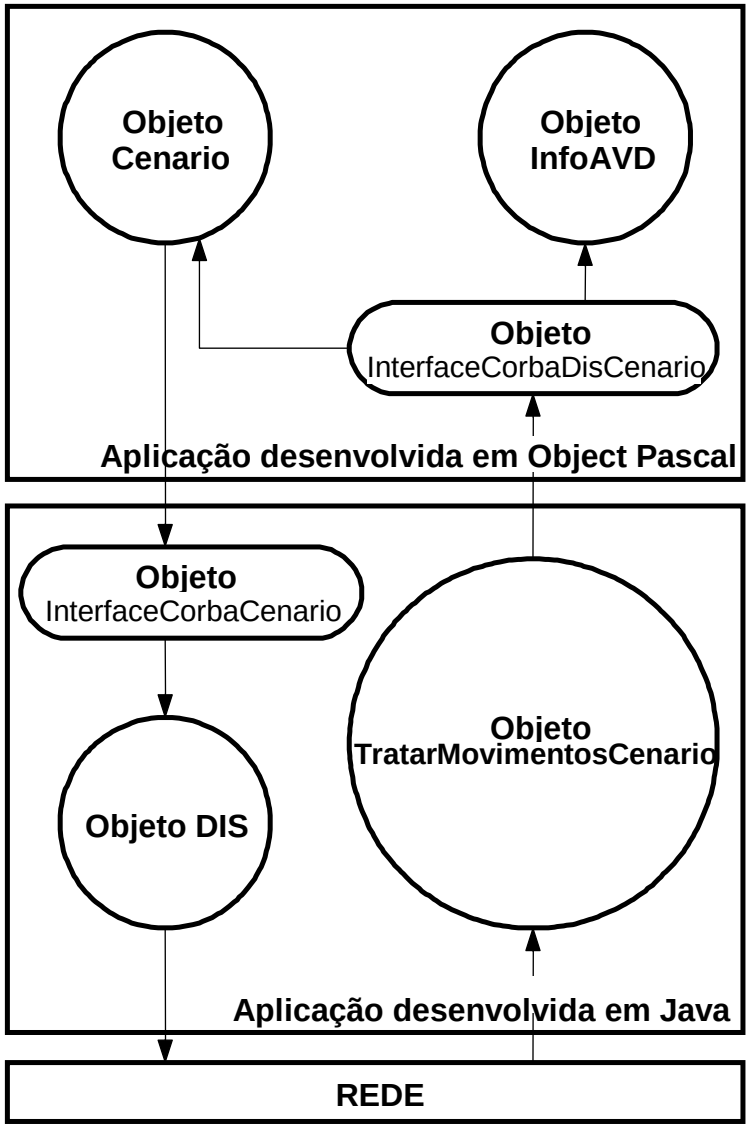
- Método execute():
  - Laço de repetição
  - Processamento paralelo
  - Processamento em tempo real

Classe HeartBeat

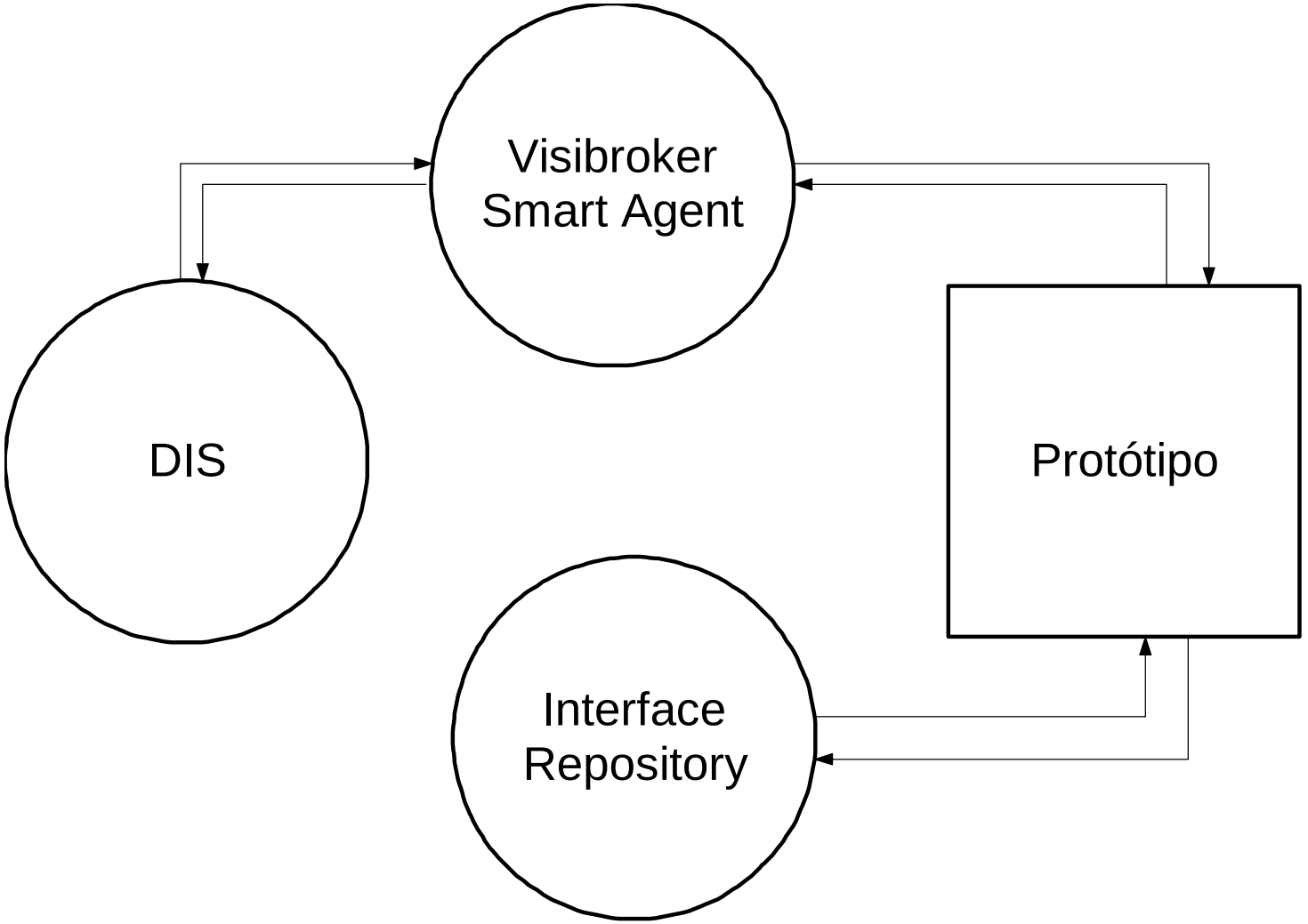


- Método execute():
  - Laço de repetição
  - Processamento paralelo
  - Envia estado do usuário para demais usuários

Classes de Interface



Funcionamento



Demonstração do protótipo

Resultados e discussão

Resultados alcançados e restrições encontradas:

- Abstração na criação do cenário através da OpenGL
- Devido a utilização da linguagem Object Pascal, OpenGL ficou vinculada ao sistema operacional Windows
- OpenGL não tem objetos pré definidos, exisitiu a necessidade de cria-los através de primitivas gráficas
- A API DIS-Java-VRML abstraiu a implementação de mecanismos para enviar PDU´s e os próprios PDU´s
- A utilização do protocolo UDP facilitou a implementação, pois não ele é um protocolo não orientado a conexão
- Porém, pode exisitir o não recebimento de algum usuário de algum pacote enviado e seu funcionamento esta limitado a uma rede local

Resultados e discussão

Resultados alcançados e restrições encontradas:

- Técnica de *HeartBeat* foi essencial para o funcionamento, apesar do tráfego gerado na rede
- a API do DIS-Java-VRML é mais especifica para simulações militares, dessa forma, ela não seria bem aproveitada em qualquer AVD
- CORBA atendeu plenamente a comunicação entre objetos distribuídos
- Com um sistema distribuído desta forma, o custo de processamento aumentou muito, o que é ruim para a geração das imagens gráficas
- Ideal ter todas as técnicas encapsulados em uma aplicação somente

Considerações Finais

Conclusões:

- É viável a construção de AVD utilizando-se OpenGL, DIS-Java-VRML e CORBA
- Trouxe experiência na criação da interface gráfica para AVD´s com OpenGL e na utilização de objetos distribuídos para criar um protótipo na área de realidade virtual
- A API do DIS-Java-VRML não foi tão relevante ao trabalho, pois com a conclusões de outros trabalhos sua definição passou a ser mais compreendida

Considerações Finais

Extensões:

- Melhorar a aparência do ambiente virtual, utilizando-se técnica de realismo para isso
- Permitir interação dos usuários com os objetos do AVD
- Melhorar o processo de comunicação do AVD, fazendo uso de mais tipos de PDU's do protocolo DIS
- Fazer com que o ambiente funcione em WAN´s
- Implantar técnicas de colisão entre usuários e, se existir, com objetos do cenário
- Melhorar a performance do AVD, como por exemplo, utilizando-se a técnica de *Dead Reckoning*
- Implementar a especificação do protocolo DIS, conforme a IEEE, na linguagem Object Pascal