Oportunidades de Colaboração empresa-universidade, através da proposição de suporte a ambientes virtuais multiusuário no padrão MPEG-4

Resumo

Um importante catalisador de colaborações entre Instituições de pesquisa e empresas é o processo de desenvolvimento de padrões internacionais. Dentre alguns desses padrões internacionais, o MPEG[**1**] merece destaque. O grupo MPEG (*Moving Picture Experts Group*), formado por especialistas de empresas, pesquisadores de universidades, etc., foi o responsável pelo desenvolvimento dos seguintes padrões internacionais: MPEG-1 (vídeo interativo em CD-ROM); MPEG-2 (televisão digital); e, atualmente, o MPEG-4, MPEG-7 e MPEG-21 (padrões relacionados ao suporte de multimídia na WWW).

O processo de desenvolvimento de um padrão como o MPEG-4 começa com a definição dos requisitos que devem ser atendidos no suporte à geração, armazenamento e transmissão de informação multimídia. A partir dos requisitos, especialistas e pesquisadores interessados podem elaborar propostas que são submetidas, analisadas e refinadas a cada encontro do MPEG-4 (que ocorre, em média, a cada 3 meses). Neste processo, os seguintes documentos são gerados, seqüencialmente, antes do estabelecimento de um padrão internacional: *WorkDraft*, *Commitee Draft*, *Final Commitee Draft*, *Draft International Standard* e, finalmente, o *International Standard*.

O padrão MPEG-4 está, atualmente, aceitando propostas de padronização do suporte a ambientes virtuais multiusuário. Num ambiente virtual multiusuário, múltiplos usuários interagem entre si e compartilham um mesmo ambiente sintético tridimensional gerado pelo computador. A integração da multimídia a ambientes virtuais multiusuário pode contribuir para o aumento do realismo em aplicações do tipo jogos, simulações, educação a distância, treinamento, etc.

Como a interação multiusuário não está, ainda, contemplada dentro do padrão MPEG-4, e esta interação vem tornando-se um requisito cada vez mais importante em aplicações para a WWW, uma proposta foi submetida que permite o acesso e controle de uma cena MPEG-4, como parte de um ambiente virtual. Este controle é feito através da extensão das APIs de um outro padrão emergente, denominado MPEG-J [**6**] que possibilita que uma aplicação ou *applets* Java, acessem e controlem os componentes de um *player* MPEG-4 externamente.

Esta proposta concentra-se principalmente em sistemas 3D, entretanto a maioria das observações também são verdadeiras para sistemas 2D. A proposta adota uma abordagem flexível de padronização das APIs, que deixa as demais funcionalidades para os implementadores da aplicação.

Atualmente, a questão do suporte à interação multiusuário dentro do MPEG-4, encontra-se na primeira fase, que é a da especificação dos requisitos [**7**]. A proposta enviada ao MPEG suporta quase todos os requisitos listados neste documento de especificação. No momento, está-se trabalhando na implementação das APIs e na especificação de novas APIs que não foram cobertas pela proposta.

Este artigo discute os requisitos de aplicações de ambientes virtuais multiusuário integrados a multimídia, bem como a proposta que foi elaborada e enviada ao grupo do MPEG-4 [**2**], e que resultou, até o momento, em pelo menos

duas propostas de colaboração internacional entre o grupo da universidade que submeteu a proposta e empresas participantes do processo de padronização do MPEG-4.

Summary

An important catalyst of collaborations between research institutions and companies is the process of development of international standard. Among some of those international standard, the MPEG[**1**] it deserves prominence. The group MPEG (Moving Picture Experts Group), formed by specialists of companies, researchers of universities, etc., it was the responsible by the development of the following international standard: MPEG-1 (interactive video in CD-ROM); MPEG-2 (digital television); and, now, the MPEG-4, MPEG-7 and MPEG-21 (standard related to the multimedia support in WWW).

The process of development of a standard as the MPEG-4 begins with the definition of the requirements that should be assisted in the support to the generation, storage and transmission of multimedia information. Starting from the requirements, specialists and interested researchers can elaborate proposed that are submitted, analyzed and refined to each meeting of the MPEG-4 (that happens, on the average, to every 3 months). In this process, the following documents are generated, one after another, before the establishment of an international standard: WorkDraft, Committee Draft, Final Committee Draft, Draft Standard International and, finally, Standard International.

The MPEG-4 standard is, now, accepting proposed of standardization of the support the multi-user virtual environments. In a multi-user virtual environment, multiple users interaction to each other and they share a same three-dimensional synthetic environment generated by the computer. The integration of the multimedia in the multi-user virtual environments can contribute to the increase of the realism in applications of the type games, simulations, education the distance, training, etc.

As the multi-user interaction is not, still, contemplated inside of the pattern MPEG-4, and this interaction comes becoming more and more a requirement important in applications for WWW, a proposal was submitted that allows the access and control of a scene MPEG-4, as part of a virtual environment. This control is made through the extension of APIs of another emergent standard, denominated pattern MPEG-J [**6**] that allow that an application or applets Java, can access and control external the components of a player MPEG-4.

This proposal concentrates mainly on systems 3D, however most of the observations are also true for systems 2D. The proposal adopts a flexible approach of standardization of APIs, which leaves the other functionality for the implements of the application.

Now, the subject of the support to the multi-user interaction inside of the MPEG-4, is in the first phase, that is the one of the specification of the requirements [**7**]. The proposal sent to MPEG almost supports all the requirements listed in this specification document. In the moment, it is being worked in the implements of APIs and in the specification of new APIs that were not covered by the proposal.

This article discusses the requirements of applications of multi-user virtual environments integrated with the multimedia, as well as the proposal that was elaborated and correspondent to the group of the MPEG-4 [**2**], and that it resulted, until the moment, in at least two proposed of international collaboration among the

group of the university that submitted the proposal and participant companies of the process of standardization of the MPEG-4.

1. Introdução

A multimídia pode ser caracterizada pela combinação de diferentes tipos de mídias em uma mesma apresentação, em que pelo menos uma das mídias seja contínua, isto é, dependente do tempo. Já os ambientes virtuais distribuídos são caracterizados pelo compartilhamento simultâneo de um mesmo ambiente tridimensional, sintetizado pelo computador, por múltiplos usuários. Estes usuários interagem com o ambiente e com os outros usuários participantes em tempo real. Com o surgimento e a integração das linguagens VRML [**8**] e Java, a disponibilização destes ambientes virtuais distribuídos na WWW é uma realidade, porém com várias limitações (suporte a multimídia, tempo de resposta, qualidade, quantidade de usuários suportados, etc.) devido principalmente à complexidade de projeto da linguagem VRML.

A integração da multimídia à ambientes virtuais é uma aliança poderosa que visa aumentar o realismo e a sensação de imersão do usuário em aplicações que suportem ambientes virtuais do tipo telecolaboração, aplicações educacionais ou de treinamento. A demanda por aplicações destes tipos, vem se tornando, cada vez mais, uma necessidade crescente na WWW.

O MPEG-4 é um padrão ISO/IEC (ISO/IEC JTC1/SC29/WG11) que esta sendo desenvolvido pelo MPEG (*Moving Picture Experts Group*), responsável também pelo desenvolvimento dos padrões MPEG-1, MPEG-2, MPEG-7 e MPEG-21, e vem se tornando uma alternativa interessante para o desenvolvimento de aplicações multimídia na WWW, permitindo também a criação de ambientes virtuais integrados a multimídia. Porém o suporte multiusuário esta em fase de padronização e poderá ser feito via MPEG-J. O MPEG-J é um conjunto de APIs desenvolvidas em Java que permitem que aplicações e *applets* Java, possam controlar os componentes de um *player* MPEG-4, possibilitando acessar e controlar a cena, além de realizar interações com a rede (servidores remotos, sistemas de arquivos, etc.).

Uma proposta foi submetida ao MPEG-4 [**5**], que estende as atuais APIs do MPEG-J para permitir a padronização do suporte multiusuário. Esta proposta concentra-se principalmente em sistemas 3D, entretanto a maioria das observações também são verdadeiras para sistemas 2D. A proposta adota uma abordagem flexível de padronização das APIs, deixando as funcionalidades específicas para os implementadores da aplicação.

Atualmente, a questão do suporte à interação multiusuário dentro do MPEG-4, encontra-se na primeira fase, que é a da especificação dos requisitos [**7**]. A proposta enviada ao MPEG suporta quase todos os requisitos listados neste documento de especificação. No momento, está-se trabalhando na implementação das APIs e na especificação de novas APIs que não foram cobertas pela proposta.

Este artigo discute os requisitos de aplicações de ambientes virtuais multiusuário integrados a multimídia, bem como a proposta que foi elaborada e enviada ao grupo do MPEG-4 [**2**], e que resultou, até o momento, em pelo menos duas propostas de colaboração internacional entre o grupo da universidade que submeteu a proposta e empresas participantes do processo de padronização do MPEG-4.

2. Requisitos para Interações Multiusuário

Interações Multiusuário (MUI – Multiuser Interactions) podem ser usadas em diferentes tipos de aplicações. Este artigo se concentra, principalmente, em sistemas multiusuário 3D, entretanto a maior parte das observações também são verdadeiras para sistemas multiusario 2D. Atualmente, o suporte multiusuário dentro do MPEG-4, se encontra em fase de especificação dos requisitos [**7**]. Por exemplo: em um prédio virtual, um usuário pode mover seu avatar, bem como ver os outros avatares participantes. Um avatar é a representação gráfica de um usuário na cena. É necessário existir um sistema de comunicação entre os terminais MPEG-4, para comunicar eventuais trocas de estado de cada avatar. Objetos podem variar de estado ao longo do tempo, por exemplo, portas podem estar abertas ou fechadas (com ou sem chave), sendo necessário reportar o estado das portas para todos os terminais participantes. O prédio virtual pode conter também uma TV com um vídeo cassete. Neste caso, o avatar pode assistir, parar, avançar ou retroceder o vídeo. A imagem e o áudio apresentados na TV devem ser sincronizados entre todos os terminais participantes. Um avatar pode ainda manipular objetos, pega-los, modifica-los, move-los ou até mesmo cria-los. Tanto o vídeo como o áudio podem ser utilizados para comunicações entre os avatares. Uma câmera pode ser usada para capturar a face do usuário e esta ser utilizada como face do avatar, o mesmo é válido para o áudio quando o usuário utilizar um microfone. Porém o áudio vai precisar de um tratamento 3D dentro do ambiente, conforme a posição de cada avatar em relação à fonte do áudio. Outros tipos de requisitos poderão existir conforme o tipo de cada aplicação e desta forma, não é interessante procurar especificar todos os requisitos para os vários tipos de aplicações MUI, e sim aspectos genéricos relacionados a todos os tipos aplicações MUI e deixar os aspectos específicos a serem implementados pelo desenvolvedor.

3. MPEG-4

O grupo MPEG (*Motion Picture Experts Group*) vem trabalhando no padrão MPEG-4 desde 1993. O MPEG-4 deverá prover serviços de televisão digital, aplicações gráficas interativas e multimídia interativa (em especial na WWW), satisfazendo as necessidades de autores, provedores de serviços e usuários finais.

As cenas audiovisuais são compostas por objetos de mídia, organizados de forma hierárquica, em uma estrutura de árvore. Os objetos de mídia (imagens, vídeo, áudio, etc.) são representados nas folhas da árvore – *Figura 1*. A árvore não é necessariamente estática - nós podem ser adicionados, substituídos ou removidos. As propriedades dos nós podem ser alteradas, como por exemplo, os parâmetros de posicionamento de um avatar.

O MPEG-4 [**2**] implementa a descrição da cena no formato BIFS (*BInary Format for Scenes*). A descrição de uma cena em MPEG-4 estende os conceitos da linguagem VRML, e desta forma, também permite uma maior facilidade para que implementadores VRML possam migrar para o MPEG-4, devido a grande semelhança existente em ambos os casos para descrever uma cena.

Objetos de mídia e BIFs são todos convertidos em segmentos de dados (*streams*) que são transportados em um ou mais segmentos elementares (ES - *Elementary Stream*). Todos os segmentos associados a um objeto de mídia são identificados por um descritor de objeto, o que permite manipular os dados de forma hierárquica, bem como associar meta informações sobre o conteúdo dos dados e direitos autorais, além de trazer também informações de configuração, como por exemplo: determinação dos decodificadores necessários e o tempo de

decodificação. Os descritores podem trazer também, indicações de QoS necessárias para a transmissão da informação multimídia, como prioridade, taxa máxima de velocidade, etc.

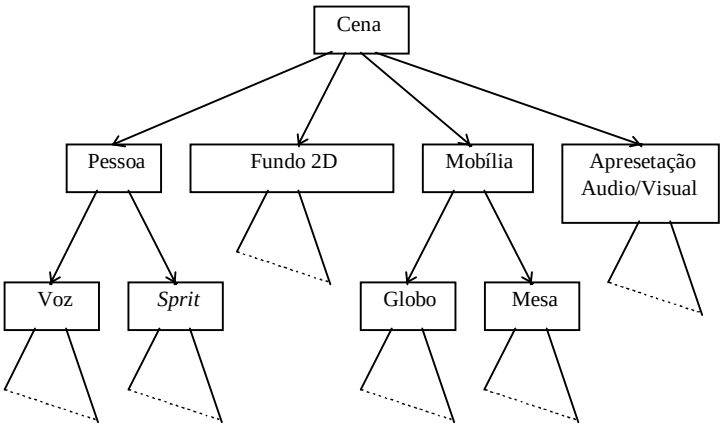

Figura 1 – Estrutura Hierárquica de uma Cena

A sincronização dos segmentos elementares – SE, é realizada através de *time stamping* de unidades de acesso individuais dentro dos próprios SE. A identificação de tais unidades de acesso e dos *time stampings* são executados na camada de sincronização. Independente do tipo de mídia, esta camada permite a identificação das unidades de acesso (como por exemplo, quadros de vídeo ou áudio, comandos de descrição da cena) em segmentos elementares, recriando os tempos base dos objetos de mídia ou a descrição da cena, permitindo assim a sincronização entre eles.

Através do uso de um *player* [**3**], um usuário pode navegar e interagir no ambiente virtual da mesma maneira que acontece com um browser VRML para a visualização de ambientes virtuais, sendo que as duas interfaces, ou seja, a maneira na qual o usuário vai interagir com o *player* ou *browser* são bastante parecidas. Um *player* MPEG-4 que além de compor e sintetizar o ambiente virtual, como acontece em um *browser* VRML, também é responsável pela descompressão, sincronização e recepção dos dados.

4. MPEG-J

O MPEG-J [**4**] foi introduzido na versão 2 do MPEG-4. Ele contém um conjunto de APIs (*Appication Programming Interface*) que possibilitam que aplicações e *applets* Java possam controlar um *player* MPEG-4, permitindo respostas apropriadas para os eventos na rede, servidor ou entradas do usuário. As aplicações Java podem ser distribuídas da mesma maneira que os demais fluxos (vídeo, áudio, descrição da cena – BIFs, etc.) através de um Fluxo Elementar - ES, ou seja, uma aplicação Java possui um descritor de objeto, e ela pode ser multiplexada e distribuída dentro de um arquivo no formato MPEG-4.

A arquitetura do MPEG-J, incorporado em um sistema MPEG-4 é mostrada na *Figura 2*. As aplicações Java podem acessar todos os componentes do *player* MPEG-4 através das APIs. Tais componentes incluem os Recursos de Apresentação e Execução, Decodificadores, Recursos de Rede e a Cena Gráfica.

4.1 Iniciando uma aplicação MPEG-J

Uma sessão MPEG-J não é iniciada enquanto o *player* MPEG-4 não receber um descritor de objeto MPEG-J. Após o recebimento, o *player* segue os seguintes passos: Abre um novo canal para o fluxo elementar da aplicação MPEG-J. O fluxo é tratado da mesma maneira que os demais fluxos (vídeo, áudio, etc.). O fluxo elementar é dividido em Unidades de Acesso. As Unidades de Acesso MPEG-J são enviadas para o *Class Loader*, no qual realiza a carga de todas as classes da aplicação. Um "decodificador" MPEG-J é responsável em receber as Unidades de Acesso e tomar as decisões para executar a aplicação. Podem existir um ou mais pontos de entrada dentro de um fluxo MPEG-J. A cada ponto de entrada, é disparado uma nova *thread* para a sua execução.

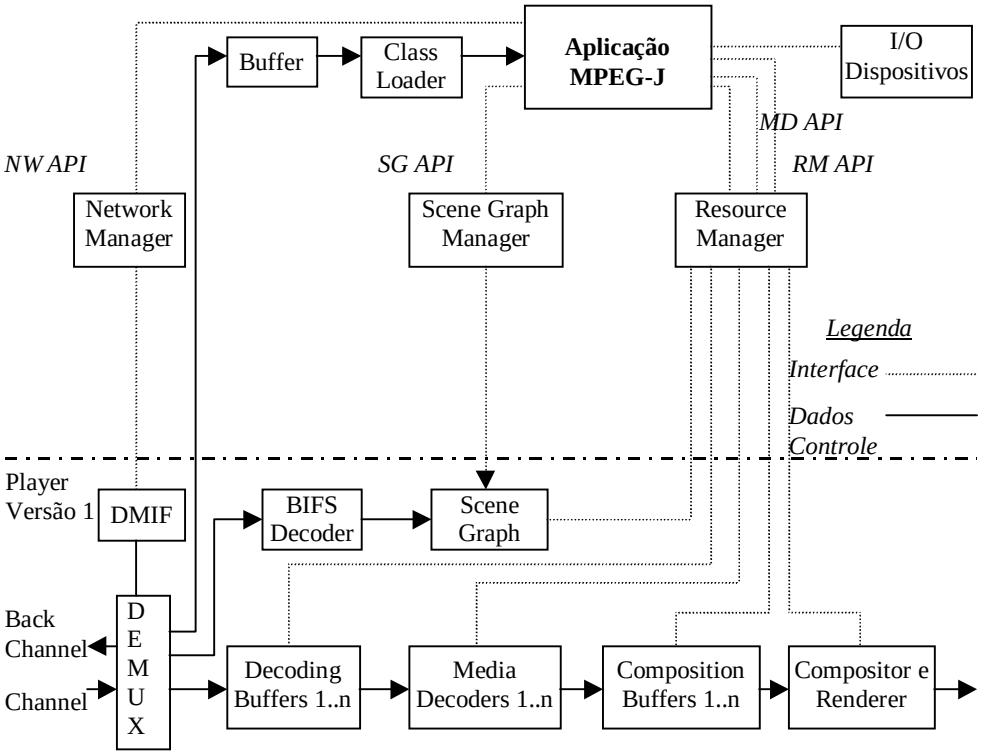

Figura 2 – Interação dos Componentes MPEG-4 com o MPEG-J
**ISO/IEC JTC1/SC29/WG11*

Existem várias questões que envolvem a distribuição das aplicações MPEG-J que estão fora do escopo deste artigo como segurança, perda de pacotes, dependência de classes, compressão, etc.

4.2 MPEG-J APIs

Os pacotes que compõem atualmente as APIs do MPEG-J são listados na Tabela 1:

Tipo API e principal classe/interface	**Descrição**

classe/interface	
Cena Gráfica: *Scene Graph*	Acesso à cena gráfica, permitindo a criação e modificação dos nós da cena.
Recursos: *Resource Manager* *Capability Manager*	Gerenciamento dos recursos do sistema. Acesso estatístico e as características dinâmicas do *player*
Decodificadores: *MPDecoder*	Acesso e controle dos decodificadores usados.
Rede: *Network Manager*	Acesso e controle dos componentes de rede de um *player* MPEG-4.

Tabela 1 – MPEG-J APIs

Para que uma aplicação MPEG-J possa acessar os gerenciadores implementados em cada terminal, ela deve fazer uso da classe MPEG-J Terminal, que possui os métodos específicos para o acesso a cada gerenciador (*getNetworkManager( )*, *getResourceManager( )* e *getrSceneManager ( )*).

4.3 Extensões para o suporte Multiusuário

Na seção 2 foram vistos alguns dos requisitos para interação multiusuário em ambientes virtuais. Alguns destes requisitos podem ser preenchidos estendendo o padrão MPEG-4 com novas capacidades, porém, os maiores parte dos requisitos necessários para a interação multiusuário são conseguidos através da extensão do MPEG-J. A *Tabela 2* mostra a proposta de extensões para preencher alguns dos requisitos vistos. Três novas APIs são propostas e uma estendida. As APIs propostas abaixo não especificam todos os requisitos vistos na seção 2, porém elas servem de base para que estes requisitos sejam especificados. As demais APIs da *Tabela 1*, não sofreram nenhuma alteração.

API	**Tipo**	**Descrição**
Scene Graph Manager	Version 2 + Sugestão	Acesso a cena gráfica. Permitindo a criação e modificação de nós e DEF IDs.
Message Manager	Sugestão	Transmissão das Mensagens entre terminais/servers.
User Interface Manager	Sugestão	Acesso aos dispositivos de entrada do usuário.
Shared State Manager	Sugestão	Gerenciamento dos estados compartilhados entre terminais/servidores

Tabela 2 – APIs para o suporte multiusuário

4.3.1 *Scene Graph Manager*

A API *Scene Graph Manager* (*Figura 3*) é a parte mais importante do MPEG-J para a implementação de sistemas multisuário. Através do acesso à cena gráfica, uma aplicação pode modificar e controlar o modo em que a cena é apresentada. A única maneira de obter acesso à cena, é implementado a interface *SceneListener*, que permite monitorar as alterações ocorridas na cena através do método *notify()*. A interface *Scene* permite o acesso aos nós da cena. A interface *Node* representa um nó na cena gráfica. Esta interface contém os métodos para acessar e modificar os atributos de cada nó.

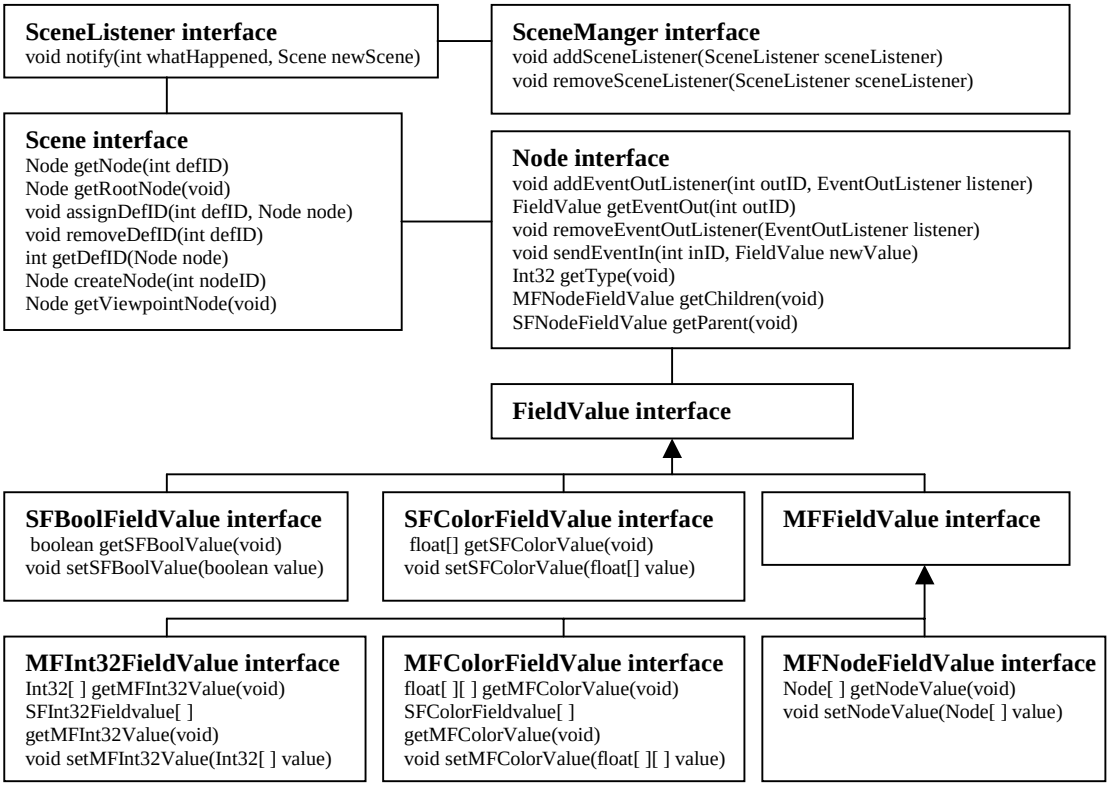

Figura 3 – Scene Graph Manager

Os eventos que podem ocorrer na cena são identificados por duas interfaces: *EventIn* e *EventOut*. A aplicação pode ser notificada da ocorrência de qualquer evento através do uso da interface *EventOutLinstener*.

A API *Scene Graph* possui métodos que podem retornar um valor de um campo (propriedade de um nó). Dois tipos de campos são suportados. *SFField*, para campos com um único valor e *MFField*, para campos com múltiplos valores.

As seguintes alterações foram realizadas:

- Para facilitar a navegação na árvore da cena gráfica, o método *getRoot( )* foi acrescentado, que possibilita o acesso à raiz na arvore (cena). Os métodos *getChildren*( ) e *getParent( )* foram incluídos para facilitar a navegação;
- O método *getViewpointNode( )* facilita o acesso ao ponto de vista do usuário dentro da cena;
- A interface *Scene* contém três novos métodos para assinalar, remover e acessar referencias aos nós da cena. Isto facilita a referência a partes especiais dentro da cena como o controle de uma avatar;
- O método *createNode()* foi adicionado para permitir a criação de novos nós, como objetos ou avatares;
- Subclasses da classe *FieldClass* subclasses foram adicionadas;
- Todos os *FieldClass* subclasses possuem métodos que além de retornar o valor, também alteram o valor dos campos;

4.3.2 *User Interface Manager*

A interface *User Manager* é responsável em acessar os dispositivos de entrada para permitir o tratamento desses eventos dentro da aplicação. Enquanto as características básicas estão disponíveis através dos vários nós, tal como o nó *ProximitySensor* (Sensor de Proximidade), outras atualmente não são tratadas. Por exemplo: entrada de dados pelo teclado para iniciar ações especiais; tratamento movimento do mouse para mover objetos dentro da cena; entrada de vídeo ou áudio utilizando uma câmera ou um microfone; aparência do cursor do mouse, modificação da barra de estados; etc. A *Figura 4* mostra o desenho desta API.

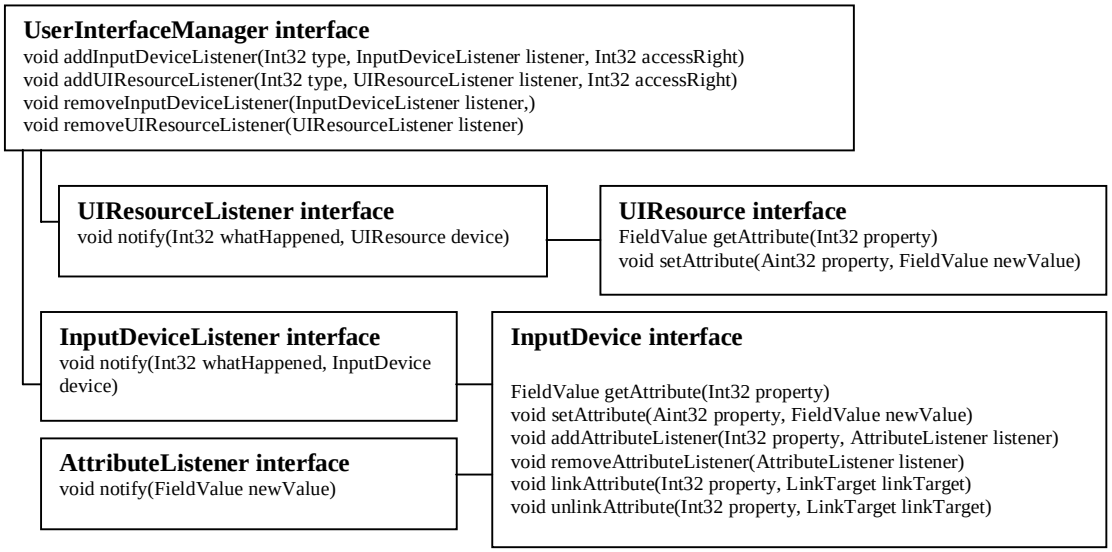

Figura 4 – User Interface Manager

A API foi projetada como uma interface de baixo nível para os componentes interativos de um *player* MPEG-4. A parte principal desta API é a interface *UserInterfaceManager*. Uma aplicação MPEG-J pode registrar um *Listener* nos dispositivos de entrada ou nos recursos de interface, através desta interface. Estes *Listeners* retornam uma mensagem de estado e uma referência para um dispositivo de entrada (*InputDevice*) ou um recurso de interface (*UIResource*). Desde que dispositivos de entrada e recursos de interface do usuário são recursos limitados em um *player* MPEG-4, a aplicação MPEG-J tem que especificar o tipo de acesso (compartilhado ou exclusivo). Em ambos os casos, se a alocação do recurso falhar, ela deve indicar uma exceção apropriada. Se a alocação foi bem sucedida, o método *notify()* receber um *InputDevice* ou um *UIResource* como parâmetro. A interface *InputDevice* oferece métodos para a leitura ou gravação de valores de certos atributos. O número e tipo de atributo dependem do tipo de dispositivo. Por exemplo, um mouse deve suportar dois ou mais valores booleanos como atributos. Para melhorar a facilidade de uso e reduzir redundâncias, os valores são encapsulados em subinterfaces da interface *FieldValue* da API *SceneGraph Manager*. Desde que aplicações MPEG-J também necessitam manipular eventos, a interface *InputDevice* pode registrar um *AttributeListener*. O método *notify()* desta interface é chamado, toda vez que um valor do atributo for trocado.

Para permitir a manipulação de dispositivos que requerem grande largura de banda, como microfones ou câmeras de vídeo, a interface *InputDevice* também

possui dois métodos que ligam um fluxo de entrada com outros componentes do MPEG-4. A interface *UIResource* foi projetada como a interface *InputDevice*. Entretanto ela não tem a habilidade de registar *Listiners* ou fazer a ligação de fluxo de dados com os componentes do MPEG-4. Tipicamente recursos de interfaces do usuário são a aparência do cursor do mouse e barra de estados.

4.3.3 Message Manager

A principal motivação desta API é de transmitir eventos que ocorrem durante a execução de uma aplicação entre os terminais MPEG-4. Exemplos desses eventos são quando o usuário entra do ambientes (*join*), mensagens de reposicionamento dos avatares, envio e recepção de texto entre dois avatares (*chat*), etc.

A principal interface desta API e a *MessageManager*. O método *getMessenger()* permite o acesso a um dos serviços de mensagens instalado, representado pela interface *Messenger*. Um terminal pode suportar diferentes tipos de mensagens. Os detalhes de implementação, como o protocolo de rede, são deixados como detalhes de implementação. Entretanto, a padronização de um especial serviço de mensagem, no qual é presente em todos os *players* conformantes com o MPEG-4, deve ser adotado.

A interface *Messeger* possui métodos para criar novas mensagens, enviar as mensagens para um endereço (representado pela interface *AddressParam*) ou para toda a lista de endereços (representado pela interface *ParamList*, no qual contem um conjunto de *AddressParams*). Endereços de outros terminais são obtidos através do método *findAddress()*. O recebimento da mensagem é manipulado pela interface *MenssageListener.*

Uma mensagem é representada pela interface *Message*. Ela contém métodos que permitem a adição, remoção e avaliação dos parâmetros das mensagens. Cada parâmetro tem um identificador (ID) único para distinguir um parâmetro do outro. Enquanto alguns identificadores de parâmetros já estão determinados, cada aplicação MPEG-J pode adicionar outros parâmetros nas mensagens. Cada mensagem deve ter sempre um parâmetro do tipo *messageID* (identificador da mensagem). Como também alguns identificadores de mensagem já estão definidos (*avartarJoin*, *avatarTranslation*, *avatarRotation*, *avatarLeave*, etc.), novos tipos de mensagens podem ser criadas. Quando uma mensagem é enviada, o método *sendMessage()* automaticamente adiciona o endereço origem como um parâmetro do tipo *senderAddress* a mensagem. Um parâmetro é representado por uma das três subinterfaces: um *AddresParam* contem o endereço destino ou origem. Um parâmetro *AddressParam* pode ser unicamente obtido através do método *findAddress()*. O único parâmetro deste método é uma *string* descrevendo um ou mais endereços dos terminais, neste caso, o objeto retornado será do tipo *ListParam*, que conterá mais que um endereço. A *Figura 5* mostra o projeto desta API.

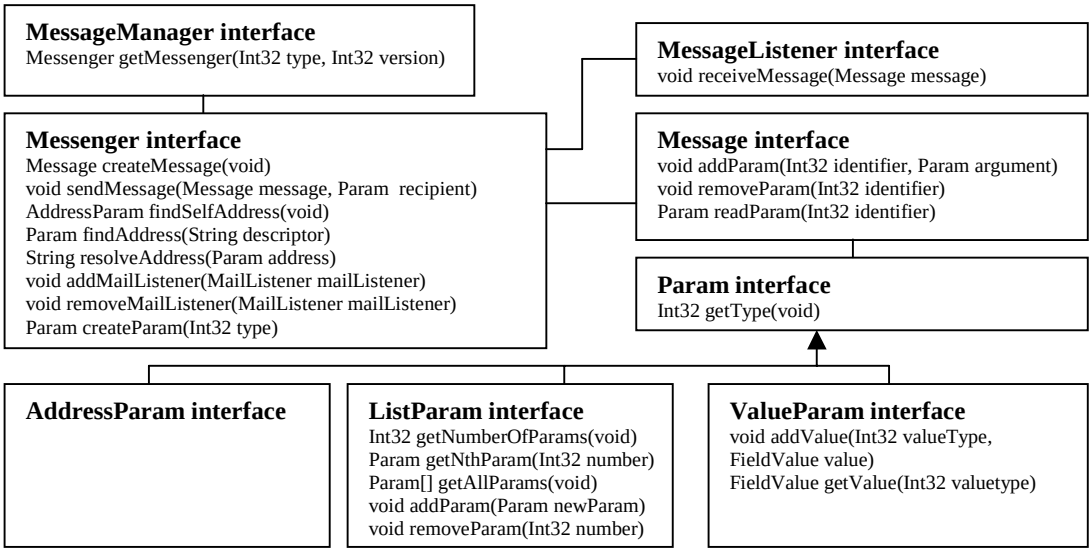

Figura 5 – Message Manager

4.3.4 *Shared State Manager*

Aplicações multiusuário, muitas vezes, necessitam controlar os estados dos objetos dentro da cena: uma porta pode estar aberta ou fechada, um objeto pode estar sendo usado ou não por uma avatar, um vídeo pode ser adiantado, atrasado ou congelado, etc. Todos estes estados devem estar acessíveis para todos os participantes do ambiente. Se existir um grande número de estados e um grande número de terminais que compartilham estes estados, este pode ser um ponto de gargalo. Uma solução para este problema é minimizar a troca de estados entre os terminais, nos quais não são diretamente influenciados por estes estados. Apesar disso, uma eficiente implementação dos estados compartilhados pode ser um desafio. O propósito desta API (*Shared State Manager*) é prover um modo flexível na gerência destes estados em um sistema multiusuário. Do mesmo modo que existem uma grande quantidade de mensagens, existem também várias formas de gerenciamento dos estados compartilhados. Desta forma, não é interessante escolher um tipo e adotá-lo como padrão. Em vez disto, é implementado unicamente a interface da API, no qual os detalhes de implementação são deixados para os implementadores.

A troca de um estado compartilhado pode comunicado aos demais terminais através do envio de uma simples mensagem. Um pode sugerir que a API *Message Manager* possui todas as necessidades para controlar os estados compartilhados, tornando desnecessário o uso da API *Shared State Manager*. Isto pode ser verdadeiro para estados que podem ser compartilhados através de troca de mensagens, porém existem outros requisitos que esta API não cobre, como por exemplo: a sincronização do acesso a um estado compartilhado, *deadlocks* ou manipulação do tráfico da rede causado pela quantidade de estados compartilhados.

A API *Shared State Manager* contém a classe *SharedStateManager* que fornece o acesso à funcionalidade principal. A implementação do modo em que os estados compartilhados serão gerenciados, não é especificada pelo padrão. A aplicação deve usar o método *getProvider()* para obter acesso a interface *SharedStateProvider*, no qual representa o gerenciador de estados instalado no

player MPEG-4. A interface *SharedStateProvider* oferece métodos para a criação, registro e subscrição dos domínios dos estados compartilhados. Um domínio contém um grupo de estados compartilhados, organizados em estrutura de árvore. Cada terminal pode criar novos domínios e registra-los em um *broker* central. Após o registro, outros terminais podem realizar chamadas aos serviços do *broker* por domínios disponíveis e receber uma descrição dos domínios que ele deseja acessar. Esta descrição é então usada para subscrever ao domínio usando o método *subscribeDomain()*. Os serviços do *broker* não são especificados aqui, porque cada aplicação pode ter diferentes tipos de requisitos. O dono do domínio pode criar novos estados e adicioná-los ao domínio. Ele pode também estabelecer os direitos de acesso para os demais terminais através do método *setAccessPermission()*. Um domínio pode conter um ou mais estados compartilhados, que são representados pela interface *State*. A interface *ValueState* contém um estado compartilhado e o seu valor atual. O valor pode ser alterado ou obtido com os seus respectivos métodos. Ao contrário dos possíveis tipos de parâmetros que uma mensagem pode ter, o valor de um estado só possui um tipo de parâmetro.

A interface *ListState* contém um grupo de estados. Ela contém métodos para adicionar, remover e acessar o conjunto de estados. Cada domínio contém uma raiz do tipo *ListState*, no qual pode ser obtida usando o método *getRootState()*. O método *getState()* possibilita o acesso rápido os estados na árvore. Por exemplo: *getState("Sala.Porta.locked")* recupera o *ValueState* "*locked*" contido no *ListState "Porta"* , no qual faz parte do *ListState "Sala"*. Para prevenir condições de *deadlocks* um terminal pode travar um ou mais estados. Os estado são protegidos de gravação até que a trava seja liberada. E finalmente, a interface *StateListener* pode ser utilizada para notificar trocar de estados para a aplicação. A *Figura 6* mostra esta API.

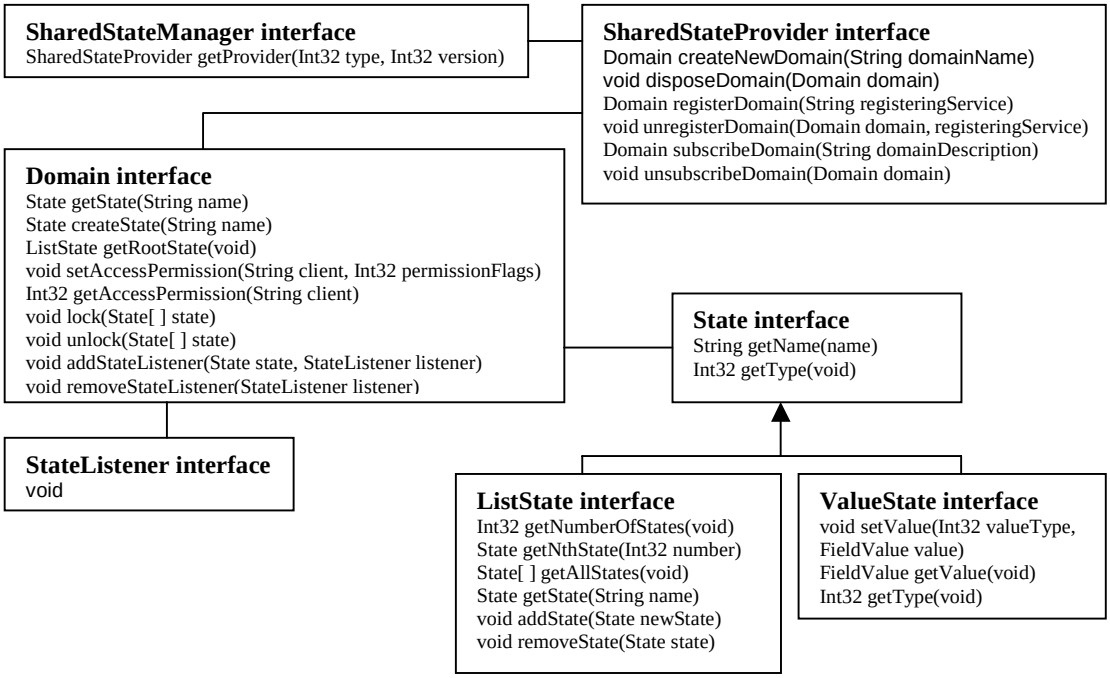

Figura 6 – Shared State Manager

4.4 Outras Extensões

As extensões acima são o primeiro passo para a padronização de sistemas multiusuário no MPEG-4. Outras modificações são necessárias, mas isto esta fora do escopo deste artigo. Algumas destas modificações são:

- Criação de novos nós, tal como *SharedNodes* proposto pelo LivingGroup [**9**] ou um nó MPEG-J *applet*;
- Uma *Server* API, pois muitas aplicações multiusuário são do tipo cliente/servidor;
- Serviços de alto nível: a proposta apresenta neste artigo especifica um conjunto de APIs de baixo nível, por motivo de flexibilidade. Porém um conjunto de APIs de alto nível pode também ser especificado.

5. Conclusões

Este artigo estendeu o conjunto de APIs do MPEG-J para possibilitar o suporte multiusuário dentro do MPEG-4. Alguns detalhes desta especificação podem sofrer alterações, pelo fato de não serem testadas. Entretanto, esta arquitetura deve servir como ponto de partida para o suporte multiusuário dentro do padrão MPEG-4.

Através da proposta enviada para o MPEG-4, duas propostas de colaboração internacional com as empresas participantes do processo de padronização do MPEG-4 foram estabelecidas. Todo o processo de padronização é muito longo e exige uma grande demanda de mão de obra, pois existem várias frentes dentro do próprio MPEG-4 e muitos pontos em aberto. Com isso, várias oportunidades de colaboração entre empresas e universidades podem surgir dentro deste processo, onde a universidade pode desenvolver trabalhos que irão contribuir para o desenvolvimento de padrões internacionais, além de permitir a pesquisa e a capacitação dos pesquisadores brasileiros.

6. Agradecimentos

A CAPES pelo financiamento de uma bolsa de estudo (Mestrado). A FAPESP (Processo 97/11038-4) pelo financiamento dos equipamentos do laboratório.

7. Referências Bibliográficas

[1] MPEG Home Page – http://www.cselt.stet.it/mpeg/.

[2] MPEG-4 Mutlimedia for Our Time – Koenen, R. - IEEE Spectrum, Vol. 36, No. 2,February 1999, pp. 26-33.

[3] MPEG-4 Support to Multiuser Virtual Environments - Todesco, G., Araujo, R.B. ICDCS'2000 – IEEE Publishing - Apr/2000.

[4] *Alex MacAulay, Julien Signes: Review of the MPEG-J specification, May 1999, contribution to the ISO/IEC JTC1/SC29/WG11 standardization process.*

[5] m5660, MPEG-4 Proposal Document – ISO/IEC JTC1/SC29/WG11 – MPEG-J API Multiuser Proposal – Todesco, G., Rueckert, U., Araujo, R.B. – Jan/2000.

[6] m4646, Alex MacAulay, Julien Signes: Review of the MPEG-J specification, May 1999, contribution to the ISO/IEC JTC1/SC29/WG11 standardization process.

[7] n3205 - Multi-users technology (Requirements and Applications) - ISO/IEC JTC1/SC29/WG11 standardization process. December 1999.

[8] http://www.vrml.org

[9] *http://www.vrml.org/WorkingGroups/living-worlds/*