VIRTUAL WORLDS
UMA ARQUITECTURA PARA AMBIENTES VIRTUAIS

Manuel Fradinho Oliveira[1]
João Madeiras Pereira[1,2]

[1] Instituto Superior Técnico
[2] INESC, Rua Alves Redol Nº9 Apartado 13069, 1000 Lisboa

Sumário

Apresentamos Virtual Worlds como uma arquitectura aberta para ambientes virtuais em larga escala com utilizadores múltiplos. É baseada na integração das linguagens VRML e JAVA e ainda da Internet, constituíndo deste modo como uma solução, em termos de plataforma de trabalho de grande abragência. A arquitectura Virtual Worlds é fornecida com um conjunto de ferramentas e API que permite aos utilizadores ampliar e personalizar o seu próprio ambiente virtual.

Palavras e expressões chave – mundos virtuais, ambientes virtuais, realidade virtual, arquitectura distribuída, utilizadores múltiplos.

0. Introdução

A Virtual Worlds está dirigida para a criação de uma arquitectura aberta para ambientes virtuais em larga escala (AVLE), tirando o máximo partido dos benefícios funcionais e técnicos proporcionados pelas Internet, JAVA e VRML.

AVLE é uma das muitas novas expressões criadas para descrever o mundo virtual. Adoptámos o termo AVLE porque um universo é naturalmente povoado por uma quantidade enorme de modelos. No entanto, há outras formas de informação que também precisam de ser tidas em consideração e que conduzem a um vasto conjunto de dados. Contudo, num determinado momento, o utilizador apenas consegue focar a sua atenção numa parte restrita desse conjunto de dados, sendo, por isso, possível a partição de informação.

Entendemos os AVLE como locais onde as pessoas podem inter-relacionar-se com várias outras e respectivos ambientes. Os AVLE dizem não só respeito a imagens gráficas de alta qualidade, mas também a ambientes credíveis que imitam com perfeição as complexidades da realidade. Aquilo que se vê não tem necessariamente que ser uma cópia visual exacta do mundo real; contudo, todos os elementos de um ambiente AVLE têm que movimentar-se correctamente e movimentar-se de forma credível, sem o que a ilusão pode desvanecer-se.

Os ambientes AVLE têm aplicação imediata nos jogos e entretenimentos informáticos e também na educação. É importante ter em conta que os ambientes AVLE não se destinam exclusivamente a comunidades do ciberespaço, mas podem também dar apoio

a empresas virtuais, tendo assim aplicação no mercado industrial. É necessário proceder a um ajuste cultural antes de os utilizadores começarem a adoptar o ambiente AVLE como suporte para o seu trabalho. As vantagens das novas tecnologias não são, só por si, suficientes para serem adoptadas pelos utilizadores finais. É obrigatório que sejam criados novos paradigmas de trabalho e que os utilizadores se sintam confortáveis a utilizá-los. É o caso típico da video-conferência. A tecnologia não é nova, mas os paradigmas de trabalho são recentes, razão pela qual só muito recentemente é que as empresas adoptaram a videoconferência como ferramenta para o trabalho diário. As vantagens imediatas mais óbvias são uma redução de custo e maior produtividade.

Os AVLE tornaram-se um objectivo atingível e não apenas uma fascinante quimera de ficção científica. Com a implementação e adopção de ambientes de AVLE, será possível construir comunidades virtuais. Estamos convictos que estas comunidades se tornarão habituais e correntes num futuro próximo e que as pessoas aprenderão novos processos de trabalho e adoptarão novos paradigmas sociais.

Apesar da sigla AVLE ser nova, o conceito não é de modo nenhum novo, tal como se pode verificar por trabalhos [JON87] já publicados. Contudo, as vantagens do ambiente AVLE não estavam disponíveis para o comum dos utilizadores, que não tem nem acesso à tecnologia de ponta nem possui as disponibilidades financeiras necessárias. Estas vantagens deveriam estar disponíveis para todos, mas só recentemente é que passaram a existir as condições necessárias para o efeito.

Este documento está subdividido em três partes principais: Introdução, Sistemas AVLE e Virtual Worlds. A primeira parte contém uma descrição abreviada das tecnologias escolhidas. A segunda parte evidencia as propriedades mais importantes que deverão ser tidas em conta no que respeita às estruturas AVLE. A terceira parte descreve com detalhe os conceitos-chave que constituem a base da arquitectura Virtual Worlds.

0.1 VRML

Virtual Reality Modeling Language (VRML) é uma norma para descrever mundos dinâmicos a três dimensões para a Internet. Torna possível a criação de cenários virtuais que podem ser preenchidos com objectos, com os quais os utilizadores podem relacionar-se interactivamente.

Tudo começou com o Labyrinth[PES94], que é uma das mais importantes etapas na via rápida que conduziu aos ambientes virtuais. Esta ideia deu origem ao formato gráfico a três dimensões para ser usado na Internet e que resultou na versão VRML 1.0. Esta versão inicial era muito limitada e pouco satisfatória, porque se restringia a cenas estáticas, mas o movimento de criação de mundos virtuais na Internet tinha começado. Presentemente, as suas especificações foram revistas, o que deu origem à versão VRML 2.0 [ISO97]. Resumidamente, é agora possível descrever não só uma cena, mas também integrar comportamentos e, se necessário, ampliar as disponibilidades construtivas com a utilização de protótipos. O objectivo principal da linguagem VRML consiste e

disseminar comunidades do ciberespaço. Contudo, e tal como acontecia com a versão VRML 1.0, a versão corrente ainda apresenta algumas limitações, que não permitem utilizá-la como a principal infraestrutura para ambientes AVLE [ROC96]. O principal óbice da linguagem VRML neste momento é a inexistência de definição no que respeita à participação de utilizadores múltiplos, o que exige a implementação de soluções desenvolvidas comercialmente, a adquirir no mercado.

0.2 Internet

Aquilo que começou como uma simples rede pondo em contacto diversas univerdades americanas, tornou-se na maior WAN actualmente existente. A explosão da Internet ultrapassou as previsões mais optimistas e o seu crescimento continua exponencial, sem sinais de futura estagnação. A simplicidade do conceito da Internet é a sua principal virtude e é essa simplicidade que permite que esteja tão divulgada. Quando se faz a ligação à Internet, é possível contactar qualquer outro computador sem nos preocuparmos com a sua localização geográfica. A vastíssima distribuição e cobertura da Internet torna o espaço uma noção perfeitamente virtual sendo, por isso, um suporte ideal para o ambiente AVLE.

Recentemente, tem-se constatado que o crescimneto expotencial da Internet começa a originar sérios problemas na sua infraestrutura. Consequentemente, o correspondente protocolo está em vias de ser alterado [DEE97], a fim de serem introduzidos melhoramentos que dêm solução aos problemas actuais e também a alguns outros que se prevêem no futuro. As tecnologias para as redes continuam a avançar, mas a Internet não desaparecerá, antes passará a integrar as novas soluções.

0.3 JAVA

A JAVA é uma linguagem [GOS96] que tem vindo a merecer grande aceitação junto da comunidade de programadores. Apresenta boas capacidades para as redes com mecanismos de distribuição promissores, que tornam bastante fácil a utilização de software em rede. Associadas a estas características, existem outras, tais como fiabilidade, segurança e multi-plataforma. Estas características combinam-se para que o JAVA seja a linguagem ideal para utilizar em Internet e VRML [LEA96]. A linguagem é completamente independente do sistema operativo sendo, por isso, necessário uma máquina virtual que forneça a necessária abstracção. Este mecanismo requer que os programas sejam transferidos para um código intermediário, denominado *bytecode*, que mais tarde será interpretado por uma implementação específica da máquina virtual JAVA. Este processo permite independência de plataformas, mas tem uma grave deficiência quanto a eficácia. O aparecimento de compiladores JIT (*just-in-time*) reduz a diferença de eficácia entre a JAVA e outras linguagens de programação, tais como a C/C++ e a PASCAL.

Outro importante factor que reduz esta diferença é o contínuo avanço tecnológico na microelectrónica, o que torna a eficácia dos códigos compilados menos relevante.

A estratégia [KRA96] subjacente à disseminação da linguagem JAVA consiste em proporcionar uma plataforma computacional completa e, consequentemente, diversos APIs estendem a sua tecnologia de base.

A característica multi-plataforma da JAVA permite que os ambientes AVLE sejam acedidos a partir de qualquer posto de trabalho/utilizador, sem ter em conta o sistema operativo subjacente.

0.4 *Hardware* AVLE

O papel do *hardware* [LAS97] na criação de ambientes AVLE é muito importante. O sucesso na aproximação aos limites da realidade depende do facto de a ilusão criada ser aceite como sendo o ambiente percepcionado. De momento, o sucesso desta ilusão depende de equipamentos que não estão facilmente disponíveis, se tivermos em conta uma relação custo-eficácia que esteja nos limites do razoável. Consequentemente, a ampla disseminação de experiências em sistemas AVLE encontra alguns obstáculos.

Algumas empresas têm-se dedicado ao desenvolvimento do *hardware* necessário para as máquinas *desktop*, tornando-o um equipamento semelhante ao computador pessoal.

1. Sistemas AVLE

A conjunção das condições acima descritas tornou possível a construção extensiveis. Existem, neste momento, diversos projectos em desenvolvimento que têm como objectivo a conceptualização e a implementação de arquitectura de software AVLE. As estruturas existentes baseiam-se em normas rígidas e pesadas ou, então, em soluções comerciais. Ambas apresentam vantagens e desvantagens e estão altamente dependentes do mercado que se pretende atingir.

Um exemplo de uma solução que recorre a uma estrutura de base comercial vem descrita na proposta *Open Community* [AND96], que foi concebida por um consórcio em que as empresas Chaco Communications, Velocity e Worlds, Inc. são as principais entidades envolvidas. Esta estrutura baseia-se numa plataforma extensível para ambientes interactivos em redes de grande dimensão [AND95] (SPLINE), que foi o resultado de um projecto de pesquisa nos laboratórios de investigação da Mitsubishi Electric (MERL). Um ambiente para utilizadores múltiplos denominado Diamond Park foi desenvolvido com base no SPLINE para demonstrar a sua funcionalidade. Outro exemplo de uma solução comercial é a que foi desenvolvida pelo Instituto Sueco de Informática, denominada DIVE (*Distributed Interactive Virtual Environment*).

Em constrate com as soluções comerciais, existem outras que se baseiam em standards de implementação já existentes. O DIS [IST93] (*Distributed Interactive Simulations*) é

um bom exemplo de um standard para ambientes virtuais baseado na simulação. O Ministério Americano da Defesa (*Department of Defense*) está a desenvolver esta estrutura. Tem havido um número crescente de projectos e de aplicações que adoptaram o DIS como estrutura de base.

A nossa proposta tem semelhanças com as estruturas que possuem soluções comerciais e soluções standard.

A utilização do ambiente AVLE está presentemente limitada a usos no domínio do entretenimento informático. Contudo, algumas empresas [ORI97] já começaram a tirar partido da integração do ambiente AVLE nas suas actividades.

Alguns projectos de pesquisa anteriormente desenvolvidos demonstraram que a criação de ambientes AVLE depende de uma estrutura que forneça soluções para problemas específicos e bem identificados, relacionados com largura de banda, distribuição, latência e fiabilidade [MAC97]. São as necessidades do utilizador que determinam os enfoques que são considerados necessários.

1.1 Largura de banda

Um princípio dogmático do AVLE relativamente a este tópico consiste em constatar que por maior que seja a largura de banda disponível, nunca é suficiente. A tecnologia das redes continua a permitir a expansão dos limites da largura de banda, mas as aplicações rapidamente consomem o excesso e exigem ainda mais. É nosso entendimento que a largura de banda influencia fortemente a arquitectura dos AVLE e é inversamente proporcional ao número de utilizadores que têm acesso ao sistema. Considerem-se dois exemplos opostos. Na Internet, a largura de banda é pequena, no entanto, a base potencial de utilizadores é enorme. As soluções que estão presentemente a ser implementadas baseiam-se em técnicas de grupagem e de filtragem. Inversamente, considere-se a aplicação AVLE desenvolvida na Universidade de Illinois [ROY95], que utilizou uma combinação de redes de banda larga. Neste caso concreto, um computador mainframe CS5 procedeu às simulações necessárias e, seguidamente, encaminhou as cenas resultantes para um sistema CAVE [CRU93] (*Cave Automatic Virtual Environment*). Contudo, neste ambiente AVLE, a base do utilizador restringiu-se à equipa de pesquisa.

1.2 Distribuição

Esta propriedade está directamente relacionada com a extensibilidade do ambiente AVLE. Existem três componentes principais que definem a distribuição das arquitecturas AVLE. Um é o modelo de arquitectura adoptado; outro é o método de distribuição de comunicações adoptado; e, finalmente, o terceiro é a gestão de dados.

1.2.1 Modelo

O modelo mais comum usado em aplicações software é o modelo cliente-servidor, mas tal conduz a fortes restrições na expansão do AVLE. Hoje em dia, as arquitecturas AVLE têm em vista uma utilização mais distribuída, evitando um servidor central. Este esquema alivia a carga do servidor, mas introduz novos problemas, tal como coerência do mundo.

1.2.2 Comunicação

Este componente define a forma como a distribuição da informação é levada a cabo. Existem três métodos principais, tal como evidenciado na **Fig. 1**.

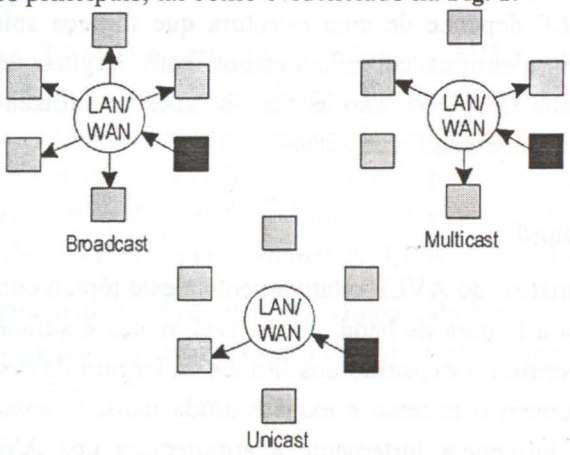

Fig. 1 – Modos de distribuição de comunicação

O primeiro é a divulgação generalizada (*broadcast*) que é altamente indesejável, porque transmite os dados a todos os clientes do AVLE. Este método é muito exigente para as infraestruturas de todo o AVLE e, portanto, extremamente oneroso. Até 1994, a implementação da estrutura DIS usava exclusivamente o processo broadcast, se bem que a comunidade DIS tenha modificado a sua filosofia desde então. No entanto, esta abordagem não é utilizável na Internet, porque a rede ficaria congestionada. A segunda abordagem baseia-se na transmissão dirigida a um único destinatário (*unicast*) e refere-se a comunicações em pé de igualdade (*peer-to-peer*). Muitos ambientes em AVLE adoptaram este esquema. Contudo, ele não é expansível e torna-se ineficaz para um grupo de utilizadores de maior dimensão. O terceiro e último método é designado por *multicast*, em que os clientes são agrupados e os dados são transmitidos em sistema *peer-to-peer* ou no sistema "um para muitos", tudo isto sem sobrecarregar a rede. Actualmente, a maior parte dos ambientes AVLE estão a pôr em prática soluções multicast, tendo em vista as vantagens desse sistema. Não existem standards para os critérios de agrupamento ou para os mecanismos, apesar de existirem diversas soluções eficazes. O laboratório MERL desenvolveu e pôs em prática uma estrutura de distribuição denominada SPLINE, que usa o sistema "*peer-to-peer multicast*". O agrupamento dos clientes consegue-se subdividindo o universo em regiões. Todos os

utilizadores de uma mesma região partilham o mesmo porto multicast. O sistema DIVE propõe outra solução *multicast*, uma vez que recorre à interacção espacial entre entidades.

1.2.3 Armazenamento de dados

A distribuição de um ambiente AVLE conduz à necessidade de se tomar uma importante decisão de arquitectura relacionada com a localização dos dados. A solução que for adoptada tem um impacte directo nos mecanismos de gestão. É extremamente importante que todos os utilizadores visionem os dados da mesma maneira, razão pela qual a base de dados tem que ser consistente.

A solução deste problema terá que ser uma de entre as seguintes:

- *Replicação total da base de dados* – é uma solução grosseira, em que todos os clientes do ambiente AVLE possuem a base de dados do universo na sua totalidade. Tal solução não é exequível, porque, à medida que o conteúdo aumenta, assim aumentam também as bases de dados em cada máquina "cliente". Outra dificuldade consiste nas divergências que se acumulam nas bases de dados à medida que o tempo passa, devido à perda de pacotes de informação durante a transmissão.
- *Base de dados centralizada* – é uma arquitectura em que os dados do universo são guardados num servidor central. Os clientes distantes têm acesso aos dados, relacionando-se interactivamente com o servidor. É este o princípio subjacente na maioria dos MUD, no qual a maior parte do processamento de dados é executado pelo servidor. Tal como previamente referido, este esquema não é expansível, limitando o número de clientes que se podem ligar interactivamente ao servidor.
- *Base de dados distribuída* – é uma outra solução, que se baseia na partilha da base de dados por entre os clientes. No entanto, torna-se necessária a existência de uma entidade mediadora, para identificar qual dos clientes é que é responsável por uma determinada parte da base de dados.

1.3 - Latência

A latência existe devido às comunicações em rede e às restrições referentes às aplicações. Dado que a latência é um problema inerente ao ambiente AVLE e poderá afectar a ilusão de imersão do utilizador final, é altamente conveniente conceber meios para reduzir os seus efeitos. Uma vez que a latência da rede está intrinsecamente ligada à infraestrutura subjacente, torna-se óbvio que a atenção deverá centrar-se na aplicação.

A latência afecta não só a eficácia global do cliente com processamento por pacotes, mas também poderá levar o ambiente AVLE a sérios estados de incoerência, destruindo a ilusão de realidade. Como exemplo, considere-se o caso que se verifica quando um *avatar* escolhe um objecto e o correspondente pacote com esse acontecimento é transmitido aos restantes *avatars*. Se outro *avatar* estiver interessado em escolher o

mesmo objecto, então tal não deverá ser possível devido à sua indisponibilidade. Contudo, se o pacote for sujeito a demora, então está-se na presença de um estado incoerente, porque dois *avatars* terão escolhido o objecto e ambos receberão um pacote contendo a informação de que uma outra entidade executou a mesma acção sobre esse mesmo objecto.

A melhor maneira de reduzir a latência é através da transmissão mínima de dados, extrapolando-se tudo o que for necessário. Algumas técnicas podem ser usadas conjuntamente, a fim de explorar as vantagens de cada uma delas e minimizar os seus aspectos negativos. A natureza da aplicação ditará quais os compromissos que deverão ser assumidos.

1.4 Fiabilidade

Esta propriedade indica o grau de fiabilidade da infraestrutura de rede de um ambiente AVLE. Terá que se chegar a um compromisso entre a latência e a largura de banda, para satisfazer os requisitos do sistema. A principal questão consiste em analisar que dados serão transmitidos através da rede e se a sua natureza permite a perda de pacotes ou se, pelo contrário, exige fiabilidade de entrega. Na arquitectura Virtual Worlds, este compromisso consegue-se separando os dados que são de natureza tempo real e os outros.

Uma vez que o protocolo de rede que está subjacente se baseia no protocolo da Internet (IP), então estão disponíveis dois tipos de transporte para o caso de o serviço ser orientado à ligação ou sem ligação.

A informação em tempo real é fortemente dependente do factor tempo e, portanto, não pode coadunar-se com o peso de um protocolo contendo confirmação de chegada e recuperação de erros. O facto de os dados em tempo real estarem subdivididos em pequenos pacotes permite que seja possível extrapolar dados perdidos, no caso de algum pacote não chegar ao seu destino. Consequentemente, é utilizado um esquema envolvendo um protocolo UDP (*User Datagram Protocol*) para suportar dados transmitidos em tempo real. Este protocolo não tem conexões e não garante as entregas.

A informação off-line não é tão exigente quanto ao factor tempo, mas exige fiabilidade na entrega. Tal exigência obriga ao uso de um protocolo TCP (*Transport Control Protocol*) como base para as comunicações. São exemplos de informação off-line a transferência de ficheiros VRML e pedidos de log-on.

2. Virtual Worlds

A ideia por detrás da arquitectura Virtual Worlds (VW) consiste em dispôr de uma estrutura simples para o ambiente AVLE, com características multi-plataforma e expansibilidade, mas com a possibilidade de aperfeiçoamentos contínuos.

A plataforma de desenvolvimento escolhida foi a JAVA, já que as suas características de

multi-plataforma, segurança, fiabilidade e actuação em rede são ideais para o ambiente AVLE. A VW é uma arquitectura aberta para ambiente AVLE, com um conjunto de classes que podem ser ampliadas, por forma a ajustarem-se às necessidades de um determinado universo. A arquitectura VW encontra-se esquematicamente delineada no diagrama da **Fig. 2**, onde se identificam cinco componentes: servidor do universo (WS), gestor de região (RM), gestor de grupo (GM), gestor de sessão (SM) e o cliente (C). Existem dois tipos de clientes: *avatar* e agente.

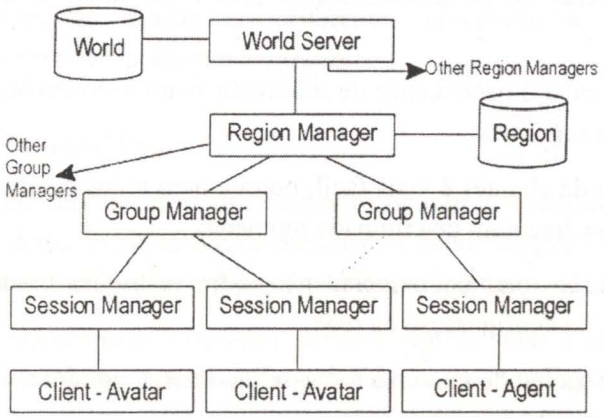

Fig. 2 – Arquitecture Funcional do Virtual Worlds

A base da arquitectura é o WS, que tem uma relação do tipo "um para muitos" com o RM. Cada RM tem uma relação do tipo "um para muitos" com o GM. Estes, por sua vez, têm uma relação do tipo "um para muitos" com o SM. A relação entre o SM e C é do tipo "um para um".

2.1 Base de Dados

A informação contida na base de dados depende das responsabilidades e atribuições do componente que supervisiona. Em consequência, existe uma hierarquia de dados, tal como pode ser visto na **Fig. 3**. As relações entre dados são do tipo "um para muitos".

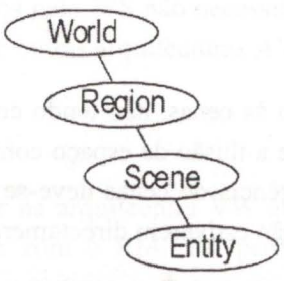

Fig. 3 – Hieraquia dos dados

O formato dos ficheiros de dados que é utilizado é o VRML 2.0, tendo em vista as vantagens que foram descritas no parágrafo 1.1 – VRML. A forma como o ficheiros são estruturados depende das convenções adoptadas pelos administradores do universo. A arquitectura VW não impõe nenhuma estrutura rígida, embora se sugira que se utilize a proposta.

2.1.1 Universo (*World*)

O universo engloba a totalidade dos dados relativos a um ambiente AVLE. Consiste não só nos elementos referentes à geometria, mas também todas as normas que regem tudo o que nele estiver incluído. A estrutura dos dados propriamente dita está subdividida em regiões.

Associados ao universo, estão mecanismos que melhoram a eficácia da pesquisa. Estes mecanismos baseiam-se no processamento prévio de dados e, de momento, são em regime off-line.

As razões subjacentes à necessidade de subdividir o universo em regiões podem resumir-se aos parágrafos seguintes:

- A gestão de clientes é mais fácil, porque nem todos os utilizadores estão dentro da mesma área num determinado momento.
- A dimensão total do universo não sofre restrições causadas pelo hardware, software ou configuração da rede.
- A gestão global do universo torna-se modular e, portanto, é mais fácil de manter em bom estado de conservação.
- Cada região pode ter normas ambientais independentes, que afectam as entidades nelas contidas.

2.1.2 Regiões (*Region*)

Cada região está ainda subdividida em cenas. A região considera-se activa, quando um cliente penetra nas suas fronteiras. A fim de evitar os problemas de dados e de *rendering* associados às transições, uma determinada região não se considera isolada, mas sim em parceria com os seus vizinhos circundantes. O gerente de região (RM) é responsável pela gestão de uma região activa.

2.1.3 Cenas (*Scene*)

Os clientes apenas têm acesso às cenas, não tendo conhecimento da restante estrutura. Tal situação confere ao cliente a ilusão de espaço contínuo e de coordenadas contínuas. A principal razão para a existência de cenas deve-se ao facto de elas permitirem uma pré-selecção de objectos que não estiverem directamente expostos.

2.1.4 Entidades (*Entity*)

Estes componentes são os elementos básicos que integram o universo. As entidades podem ter os seus próprios comportamentos, geometria e aparência. Cada entidade tem uma marcação temporal no momento da sua criação e, portanto, poderá "envelhecer". Existem alguns tipos especiais de entidades, os quais são referenciados por nomenclatura

apropriada que tem em conta as suas propriedades características:

- **Utilizador.** Esta entidade representa um utilizador contido no universo. Consoante se tratar de controle pessoal ou inteligência artificial (IA), o utilizador será respectivamente denominado um *avatar* ou um agente.
- **Cenas e regiões.** São entidades que representam o cenário espacial, contendo uma variedade de outras entidades.

As entidades podem ser usadas em diferentes universos, a não ser que os administradores de universo decidam o contrário.

2.2 Servidor de universo (WS)

O principal papel do servidor é o de manter e garantir a estabilidade do universo. Mantém, para além disso, todos os outros parâmetros do utilizador, validando quaisquer novas entradas ou modificações surgidas no anterior parâmetro.

É possível aos utilizadores criar entidades, cenas e regiões, mas a sua construção terá que se subordinar às restrições determinadas pelos administradores do universo.

O modelo de distribuição adoptado permite que a responsabilidade da gestão seja delegada no RM. No entanto devido ao problema da rede nao ser segura , por isso torna-se necessário que WS atribuia responsabilidades de RM a máquinas de confiança. O WS possui tabelas identificando os potenciais RM e indica se alguma região está já activa. No caso de não estar disponível nenhuma das máquinas dedicadas, então o WS utilizará os recursos disponíveis e lançará o RM dentro de si próprio.

O WS não faz nenhuns cálculos intensivos em tempo real. Os utilizadores finais são muito exigentes para o ambiente AVLE, exigindo respostas instantâneas aos seus inputs, necessitanto, portanto, de capacidade de trabalho em tempo real ou algo suficientemente parecido. Contudo, os utilizadores finais permitem compassos de espera durante a fase de log-on e log-off, que são os únicos momentos em que a aplicação cliente se relaciona com o WS. Os serviços fornecidos pelo WS não necessitam do poder de processamento de um mainframe, ao contrário de outras arquitecturas AVLE.

2.2.1 Segurança

Todos os clientes têm que entrar na arquitectura VW através do WS e, seguidamente, terminar o protocolo de entrada com o RM. Tal permite um protocolo baseado no KEREBEROS [ATK93], em que o sistema garante segurança e autenticidade a todos os componentes da arquitectura.

Este mecanismo permite que utilizadores com diferentes níveis de privilégio acedam ao sistema e se relacionem com ele de forma interactiva.

2.2.2 Consistência

A responsabilidade de manutenção da consistência do universo é uma tarefa partilhada pelo WS e pelo RM. É importante ter a certeza de que:

- Não é permitida nenhuma clonagem ou criação de entidades, a não ser que a acção pretendida esteja disponível para o utilizador.
- WS tem um referencial de tempo absoluto, enquanto que o RM apenas tem um referencial de tempo relativo. O WS tem que se certificar que todos os acontecimentos e entidades obedecem a uma ordem cronológica.
- As cenas têm uma elevada probabilidade de sofrer modificações quando são visitadas por utilizadores. O WS não é responsável por uma região enquanto ela estiver activa. No entanto, não poderá esperar até que a RM devolva a responsabilidade para se auto-actualizar. Por isso, o WS actualiza periodicamente a região, tendo o cuidado de verificar se o RM não está sujeito a uma fase crítica.
- WS supervisiona o RM quando cada um deles tem regiões que são vizinhas. Nestes casos, é criado um gestor dedicado para controlar as zonas adjacentes de cada região.

2.3 Gestor de região (RM)

Cada RM é responsável por apenas uma região activa, se bem que uma máquina possa processar mais do que um RM se os recursos o permitirem.

O RM mantém a consistência da região e verifica se as normas são respeitadas permanentemente. No âmbito do RM, existem dois componentes que podem ter várias instâncias e funcionar concorrentemente: gestor de grupo (GM) e gestor de sessão (SM).

Para melhor ilustrar o relacionamento entre RM, GM e SM, considere-se o seguinte exemplo: no âmbito de um universo, há uma região que representa um edifício, cada GM tem uma área de influência que é equivalente a um piso e, finalmente, cada SM é responsável por um cliente. Para o caso de todos os clientes estarem em pisos diferentes, haverá o mesmo número de GM e SM. No entanto, no caso de alguns clientes estarem no mesmo piso, então eles partilharão o mesmo GM.

2.3.1 Gestor de grupo (GM)

Um GM cria e mantém diversas portos de multicast, dependendo da natureza dos dados:

- As actualizações de movimento são, de longe, a informação mais frequentemente transmitida ao longo da rede. Cada *avatar* envia um pacote apropriado com a actualização da sua posição e do seu movimento.
- As actualizações de cena são muito importantes e necessitam de transmissão rápida com um certo grau de fiabilidade, a fim de evitar inconsistências.

Quando um RM aceita um cliente, duas situações podem ocorrer: é criado um novo GM ou, se o cliente está na proximidade de um GM já existente, então é esse o escolhido. Em qualquer caso, é dado ao cliente o porto multicast do GM seleccionado.

Cada GM é responsável por um grupo de um ou mais clientes. A sua porta de entrada recebe as mensagens que são destinadas a todos os clientes que compõem esse grupo.

2.3.2 Gestor de sessão (SM)

A utilização da arquitectura JAVA obriga-nos a adoptar uma aproximação de *thin-client* e, consequentemente, dotamos o SM de algumas capacidades que permitam aliviar a carga de trabalho do cliente.

O SM executa algumas tarefas que exigem forte poder computacional, sendo os resultado seguidamente comunicados ao cliente através de um canal *peer-to-peer*.

O SM tem acesso a todos os dados da região, assim como à sua topografia. O cliente apenas se apercebe da sua localização imediata, que traduz para algumas cenas. Uma das responsabilidades do SM consiste em supervisionar o utilizador e informar o cliente àcerca de quais as cenas a carregar e a descarreger, dependendo dos algoritmos escolhidos.

2.4 Cliente

O utilizador relaciona-se interactivamente com a arquitectura VW através de uma aplicação cliente. Este cliente baseia-se numa arquitectura leve.

Apesar da arquitectura VW estar orientada para fornecer uma arquitectura aberta para ambiente AVLE, não negligenciámos o facto de o utilizador necessitar ter um *avatar* com uma representação credível e com um elevado grau de interactividade [CAP97]. Estas premissas são necessárias para se obter a ilusão de um universo virtual real. A aplicação cliente possui uma natureza multi-processo que é propícia a uma organização modular. A arquitectura resultante está patente no diagrama da **Fig. 4**.

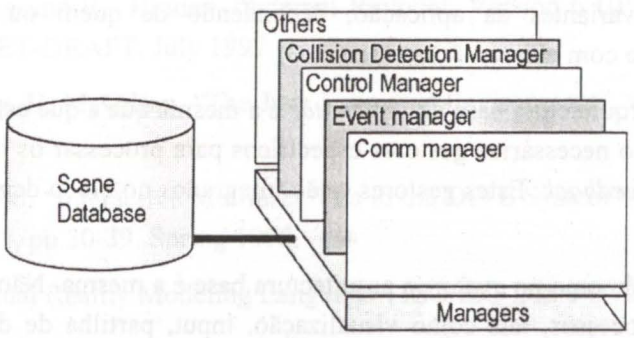

Fig. 4 – Arquitectura do cliente

- Base de dados da cena (*Scene Database*). Não se trata de uma base de dados

propriamente dita, nem orientada a objectos nem relacional, mas sim um repositório gráfico de cenas dinâmicas. Existem mecanismos de cache que são controlados remotamente pelo SM residente no RM.

- Gestores (*Managers*). Trata-se de um objecto que fornece a estrutura a todos os gestores operativos. Contém todas as referências dos gestores existentes no cliente. Quando um gestor é integrado no sistema, tem que ficar registado no registo de gestores.
- Gestor de comunicações (*Comm Manager*). Este gestor gere todas as comunicações com a rede. Ocupa-se tanto das comunicações UDP como TCP. Pode haver caminhos simultâneos mas separados, dependendo da natureza das interacções do utilizador.

  Associado a este gestor, está um conjunto de classes que gerem os diferentes protocolos que são suportados pelo sistema. Se houver um novo protocolo, tal necessitará da inserção de uma nova classe no conjunto.
- Gestor de acontecimento (*Event Manager*). Este gestor processa todos os acontecimentos relacionados com uma acção executada por um utilizador, um objecto, um agente, ou um *avatar* de outro utilizador. O acontecimento poderá, assim, ser gerado localmente ou à distância.
- Gestor de controle (*Control Manager*). Trata-se do gestor que manuseia todos os pacotes de controle que têm origem no SM que supervisiona o cliente e executa todos os pedidos de controle. Está intimamente ligado à porta dedicada TCP, criada pelo gestor de comunicações.
- Gestor da detecção de colisões (*Collision Manager*). A acção básica de uma aplicação cliente consiste em navegar no ambiente virtual e, portanto, terá que detectar colisões com o ambiente que a rodeia.
- Outros (*Others*). Trata-se de gestores futuros, que serão integrados numa fase posterior.

Existem duas variantes da aplicação, dependendo de quem ou quê se relaciona interactivamento com ela:

- *Avatar*. A arquitectura base de um *avatar* é a mesma que a que está descrita na **Fig.4**. Contudo, são necessários gestores específicos para processar os inputs do utilizador e fornecer *feedback*. Estes gestores estão integrados no bloco denominado *others* na **Fig. 4**.
- Agentes. Tal como no *avatar*, a arquitectura base é a mesma. Não há necessidade de diversos processos, tais como visualização, input, partilha de dados e partilha de aplicações. Contudo, há necessidade de incluir outros processos, tais como avaliação, orientação, comportamento e interactividade.

3. Desenvolvimentos futuros

No momento em que escrevemos este documento, o API da JAVA 3D [SUN97] da SUN ainda não está implementado. Por esse motivo, a visualização das cenas está a ser feita através de um EAI (*External Authoring Interface*). O EAI permite que uma aplicação comunique com o browser VRML existente ou com o plug-in. Este interface está claramente definido na especificação VRML.

Algumas soluções alternativas de visualização estão a ser tentadas, tal como a Liquid Reality da Dimension X, mas têm a desvantagem de reduzir a capacidade multi-plataforma da aplicação.

Referências

[AND95] Waters, "Building Multi-User Interactive Multimedia Environments at MERL", IEEE MultiMedia, 2(4), Winter 1995, pp. 77-82

[AND96] D. B. Anderson, J. W. Barrus, J. H. Howard, C. Rich and C. Shen and R. C. D. Anderson, D. Greening, M. Ma, M. Marvit and R. Waters, "Open Community Overview", Proposal, November 1996

[ATK93] D. Atkins, "Charon: Keberos Extensions for Authentication Over Secondary Networks", MIT, May 1993

[CAP97] T. K. Capin, H. Noser, D. Thalmann, I. S. Pandzic and N. M. Thalmann, "Virtual Human Representation and Communication in VLNet", IEEE Computer Graphics and Applications, Vol. 17, Nº2, Mar-Apr 1997, pp 42-53

[CRU93] C. Cruz-Neira et al., "Scientists in Wonderland: A Report on Visualization Applications in the CALSVE Virtual Reality Environments", Proc. IEEE Symp. Research Frontiers in Virtual Reality, 1993, pp. 59-65

[DEE97] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) specification", INTERNET-DRAFT, July 1997

[GOS96] J. Gosling, H. McGilton, "The Java Language Environment", white paper, , May 1996

[HAG96] O. Hagsand, "Interactive Multiuser VEs in the DIVE System", IEEE Multimedia, Vol.3, Nº1, pp.30-39, Spring 1996.

[ISO97] "The Virtual Reality Modeling Language", ISO/IEC DIS 14772-1, April 1997

[IST93] Institute for Simulation and Training, "Communication Architecture for Distributed Interactive Simulation (CADIS), University of Central Florida, Florida, June 1993

[JON87] R. S. Johnston, "The SimNet Visual System", Proc. 9th International/Service/IndustryTraining Systems and Education Conf. Nov 1987, pp 264-273

[KRA96] D. Kramer, B. Joy and D. Spenhoff, "The JavaTM Platform", white paper, May 1996

[LAS97] A. A. Lastra, "Technology for Virtual Reality", Siggraph 97 course, Los Angels, August 1997

[LEA96] R. Lea, K. Matsuda nad K. Miyashita, "Java for 3D and VRML worlds", New Riders, Indianapolis, 1996

[MAC97] M. R. Macedonia and M. J. Zyda, "A Taxomnomy for Networked Virtual Environments", IEEE Multimeda, Vol. 4, Nº1, Jan-Mar 1997.

[ORI97] A. Oringel and K. Guericke "The Business Benefits of Online Communities", vrmlsite.

[PES94] M. D. Pesce, P. Kennard, A. D. Parisi, "Cyberspace", World Wide Web conference 1994, CERN, Geneva, May 1994,

[ROC96] R. Rockwell, "Infrastructure and Architecture for Cyberspace Communities", Computer Graphics and Applications, Vol. 30, Nº4, November 1996, pp 19-24

[ROY95] T.M. Roy et al., "Steering a High Performance Computing Application from a Virtual Environment," Presence, Vol. 4, Nº2, Summer 1995, pp. 110-120

[SUN97] Javasoft, "Java 3D API Specification", Sun Microsystems, May 27, 1997