Jogos por Computador – Histórico, Relevância Tecnológica e Mercadológica, Tendências e Técnicas de Implementação

Prof. Dr. André Luiz Battaiola
Departamento de Computação da Universidade Federal de São Carlos
Via Washington Luiz, Km. 235
São Carlos – SP - CEP: 13565-905 - CP: 676
email: andre@dc.ufscar.br
URL: www.dc.ufscar.br/grv/andre.htm

Departamento de Design da Universidade Federal do Paraná
email: albattaiola@ufpr.br
URL: www.design.ufpr.br/lai

JAI2000 / SBC

Resumo: Jogos por computador têm apresentado um surpreendente grau de evolução tecnológica nos últimos anos. O grande interesse no desenvolvimento deste tipo de software se deve ao seu atrativo comercial, cujo mercado mundial se situa na faixa das dezenas de bilhões de dólares. Assim, muitas empresas das áreas de computação e entretenimento estão fazendo pesados investimentos no desenvolvimento de técnicas computacionais sofisticadas, as quais podem ser empregadas em outros programas interativos, principalmente naqueles que envolvam interfaces gráficas complexas, o que significa o uso conjunto de várias mídias, animações com gráficos 2D e 3D, vídeos, som, som 3D, etc, bem como ambientes multiusuário baseados em Internet. Como o interesse por conteúdo na Internet tem aumentado acentuadamente, é razoável se supor que também cresça o interesse por formas mais sofisticadas de apresentá-lo, o que torna bastante atrativas as técnicas de implementação dos jogos por computador.
O objetivo deste curso é ressaltar o potencial mercadológico e técnico-científico da área de jogos por computador através de uma apresentação dos principais conceitos envolvidos na sua implementação. O curso é estruturado para inicialmente apresentar a evolução do hardware e do software dos jogos por computador, destacando sua influência no desenvolvimento de novas tecnologias, e posteriormente, fazer uma apresentação concisa das principais técnicas de implementação. Observe-se que, devido à complexidade destas técnicas, a exposição deverá dosar eficientemente o grau de profundidade da abordagem de cada conceito, buscando seguir a filosofia de um curso introdutório ao tema e a limitação ao tamanho do texto.

Abstract: In the last years, computer games have shown an impressive technological evolution. The great interest in the development of this kind of software is its commercial profit. Great companies in the computer and entertainment market are investing a lot of resources in the development of sophisticated computer techniques that can be applied in the development of other interactive programs, specially the programs that require complex graphics interfaces. This means the simultaneous use of many medias as 2D and 3D graphics animations, videos, sound, 3D sound, etc, and multiusers environments based in the Internet. As the interest for Internet content increases dramatically, it makes sense to consider that will increase the interest for sophisticated ways to show it. In this case, games implementation techniques will be very attractive.
This course objective is to emphasize the market, technological and scientific potential of the computer games area by a presentation of the main concepts involved in its implementation. The course is structured to present initially the computer games hardware and software evolution. After, it is presented a summary of the main implementation techniques. As the issue is very complex, take notice that the presentation will balance the detail level of each concept mentioned, trying to follow an introductory course philosophy and the text size limitation.

Palavras-Chaves: Jogos por Computador e Computação Gráfica.

***Key-words*:** Computer Games and Computer Graphics.

Observação: Este texto contém poucas figuras pela dificuldade de se utilizar imagens preto-e-branco no contexto deste tema, bem como por limitações de número de folhas.

Índice

1- Evolução histórica dos jogos por computador

Jogos por computador são atualmente um produto que dispensa apresentações. A maioria dos usuários de computadores já gastou várias horas se divertindo com algum jogo deste tipo. Se o produto é bem conhecido, o mesmo não se pode dizer de diversas características suas, tais como, componentes, técnicas de implementação, importância tecnológica e comercial, etc. Este texto visa detalhar, de forma compatível a um curso introdutório e a limitação de páginas estipulada, a maior parte destas características.

Para se entender as características atuais dos jogos por computador, bem como a influência que eles tiveram no desenvolvimento de diversas tecnologias, é importante conhecer a sua evolução histórica, a qual é descrita abaixo. [VIDEOT], [DVDEN], [PLAY2], [ATARI], [ARRGH]. Esta descrição enfocara essencialmente os jogos de galeria (*arcade games*), que são unidades operadas por fichas e disponíveis para jogar em lojas especializadas, e os jogos por console, aquele que o usuário compra um console que vem com um conector para o aparelho de TV e um encaixe para algum tipo de unidade de armazenamento, que contém o programa e/ou os dados de um jogo em particular. No Brasil, os jogos de galeria são conhecidos, usualmente, como fliperamas, no caso de máquinas que tem funcionamento eletro-mecânico, ou videogames, com funcionamento eletrônico. O termo videogames é também associado aos jogos por console.

1971 Nolan Bushnell da *Nutting Associatesl* projeta ***Computer Space***, o primeiro videogame explorado comercialmente.

1972 N. Bushenell, com o ganho adquirido em *Computer Space*, cria a *Atari* e incumbe Al Alcorn para construir ***Pong***, o primeiro videogame com sucesso comercial.

1974 A *Atari* e a *Kee Games* lançam ***Tank***, o primeiro videogame a usar um chip ROM (Read Only Memory) para armazenar dados gráficos, o que permitiu que os caracteres de tela fossem exibidos com maior definição. Note-se que a *Computer Space* e a *Pong* utilizavam, respectivamente, grupos de pontos e blocos.

1975 A Taito/Midway lança ***Gunfight*****,** um jogo ambientado no velho oeste e que permitia até dois jogadores. Também foi o primeiro a utilizar microprocessador.

Steve Jobs, o quadragésimo empregado contratado pela *Atari*, projeta, com a ajuda de Steve Wozniak, ***Breakout***. Utilizando partes "emprestadas" de uma unidade Atari, estes especialistas constroem o seu primeiro protótipo de computador na recém fundada *Apple Computer*.

1978 A *Atari* lança ***Fire Truck***, um de seus últimos jogos preto-e-branco, mas o primeiro do tipo cooperativo, que compartilha funções entre dois jogadores. *Fire Truck* dava a 2 jogadores a oportunidade de controlar um rápido caminhão de bombeiro. O jogador sentado na frente dirigia a cabine do caminhão, enquanto o segundo jogador, de pé atrás, dirigia uma carroceria com uma escada que chacoalhava intensamente.

A *Taito/Bally/Midway* lança ***Space Invaders***, o primeiro videogame de grande sucesso comercial e o primeiro a ser disponibilizado fora das galerias de jogos. Era encontrado em bares, restaurantes e sorveterias.

Criado por Dave Subben, o jogo ***Football***, da *Atari*, se torna o primeiro videogame de esportes a ser comercializado. O jogo deixava as pessoas maravilhadas com a ação rápida e a simulação complexa de um time esportivo. Este jogo também marcou a introdução conjunta do *track-ball* e dos vídeos com *scrolling*.

1979 ***Lunar Lander***, da *Atari*, enfoca conceitos das leis de Newton e foi o primeiro jogo a incorporar a tecnologia dos monitores vetoriais, que foi concebida em parte para o

programa espacial Apollo, com o objetivo de criar um sistema capaz de simular uma aterrisagem na Lua.

Asteroids, da Atari, foi um grande sucesso comercial que posicionou os videogames como uma importante mídia de entretenimento. No auge de sua popularidade em 1980, um artigo na revista Newsweek ressaltou que o público deste jogo era constituído de um grande número de profissionais adultos que aliviavam o stress do trabalho diário jogando-o durante a hora do almoço. A nova versão do jogo, ***Asteroids Deluxe***, foi liberada no próximo ano.

Warrior, da Cinematronics, foi o primeiro jogo de luta um-contra-um, onde dois jogadores travavam uma luta de espadas. O jogo contava com gráficos vetoriais espetaculares, um belo pano de fundo e uma refinada arte da cabine interna. Além disso, o jogo produzia imagens mais complexas que o hardware gráfico vetorial da Atari era capaz de produzir. Infelizmente, o sistema vetorial da Cinematronics era também menos confiável.

1980 ***Battlezone***, da *Atari*, ofereceu o primeiro ambiente verdadeiramente tridimensional., de tal forma que as forças armadas americanas contrataram a Atari para construir uma versão especialmente modificada e incrementada para treinamento de pilotos de tanques de guerra.

Berzerk, da *Universal Research Laboratories/Stern Inc*, foi um jogo que apresentou uma máquina que falava. O jogo impressionou porque poucas pessoas na época tinham tido algum tipo de experiência com voz sintetizada. Dado que a digitalização do som era cara, as sentenças faladas no jogo compartilhavam um vocabulário de somente 30 palavras.

Defender, da *Williams Electronics***,** foi o primeiro jogo a compor um mundo artificial no qual eventos do jogo poderiam ocorrer fora da tela onde o usuário via as cenas.

Pac-Man**,** da *Bally/Midway*, foi produzido com licença da companhia Namco. Ele foi baseado em um conto popular japonês. O jogo fez um sucesso tão grande no Japão que causou impacto positivo na cotação do yen. Pac-Man também foi grande sucesso nos Estados Unidos, aparecendo na capa de uma revista Time, estrelando um desenho animado das manhãs de sábado e ganhando uma canção de sucesso.

Missile Command, da *Atari*, foi projetado para refletir o medo americano de um conflito nuclear. Originalmente ele era para ser denominado *Armageddon*.

1981 ***Gorf***, da Bally/Midway, nasceu da combinação de jogos *Space Invaders*, *Galaxian* e outros jogos similares do tipo "mova e atire" da mesma empresa. *Gorf* foi um dos primeiros jogos a oferecer estágios distintos de jogo, bem como um dos últimos com fala.

Donkey Kong, da *Nintendo*, apresentou um estilo brilhante de jogo e uma história esquisita sobre um macaco gigante e um personagem curioso denominado de *Jumpman*. Somente depois de finalizado o projeto e a implementação do jogo, que este personagem recebeu o famoso nome de Mario.

Centipede**,** da *Atari*, foi o primeiro jogo projetado por uma mulher. Gráficos coloridos e um engenhoso modo de jogar fizeram com que o *Centipede* atraísse mais fãs femininos do que masculinos.

Tempest, da Atari, foi um dos primeiros jogos a utilizar um monitor vetorial colorido. Tempest dispunha de gráficos surrealistas do tipo estrutura-de-arame. Este jogo nasceu de um sonho do seu criador e foi a base do famoso jogo *Tempest 2000*.

Warlords, da Atari, foi uma variação para múltiplos jogadores do Breakout, cujo projeto buscava transmitir a idéia de que a cooperação era um elemento importante para os jogadores atingirem os estágios mais altos de jogo, bem como encorajar cada jogador no desenvolvimento de regras e táticas próprias.

Frogger, da *Konami/Sega*, foi um jogo de muitos fãs atraídos pelos belos gráficos e a navegação através de obstáculos móveis.

1982 ***Joust***, da *Williams Electronics*, teve o seu atrativo acentuado pelo magnífico estilo de jogo para múltiplos jogadores. Por este estilo, o jogo alternava constantemente o ganho e a perda de pontuação para jogadas colaborativas entre os jogadores, bem como a destruição do outro. O interessante deste jogo é que ele pode ser usado como um mecanismo de teste da teoria que diz que jogos competitivos diminuem a agressividade e jogos colaborativos aumentam.

Tron, da *Bally Midway*, foi projetado em conjunto com o filme da Disney, tendo o jogo se tornado parte da trama e possivelmente influenciado no final do filme.

Quantum, da Atari, foi um raro jogo, projetado com base em conceitos de mecânica quântica. O jogador, neste jogo, tinha que, através de um *track-ball*, circundar e isolar partículas subatômicas, realizando voltas completas em torno delas.

Star Wars, da *Atari*, é o resultado da conversão de um jogo dois anos mais velho denominado *Warp-Speed*, o qual foi projetado para operar com imagens 3D. *Star-Wars* usou originalmente um *joystick*. O controlador de vôo foi projetado pela modificação de um controlador criado para a versão militar de *Battlezone*.

1983 ***Dragon's Lair***, da *Starcom/Cinematronics*, foi projetado como um filme interativo e se tornou o primeiro jogo a utilizar a tecnologia de disco a laser. O visual de Dragon's Lair's fascinou a todos que o viram e o jogo conquistou a imortalidade.

***Blaster*,** da *Williams Electronics*, foi um exemplo de geração rápida de gráficos através de escalamento. O seu joystick permitia 49 movimentos, o que possibilitava um controle rápido e suave de seu "*space shuttle*".

1984 ***I, Robot***, da *Atari*, foi o primeiro jogo a operar com gráficos gerados por polígonos 3D. Apesar deste jogo não ter sido um sucesso, ele foi o ancestral direto dos jogos mais sofisticados de corrida e de simuladores de vôo.

1986 ***Space Harrier***, da *Atari*, foi um marco na utilização de gráficos baseados no método de *sprites-3D*. Sua interface gráfica era capaz de escalar e rotacionar, de forma suave e rápida, em torno de 32.000 *sprites*. Além disso, o jogo utilizava som estéreo. Apesar desta performance, este tipo de técnica estava com os dias contados, pois gráficos por polígonos 3D já haviam surgido e logo iriam se tornar a técnica dominante.

1989 ***S.T.U.N. Runner***, da *Atari*, bem como *Hard Drivin*, jogo da mesma época, marca a tendência desta empresa em produzir jogos com gráficos poligonais 3D.

1991 ***Time Traveler***, da *Sega*, foi o primeiro jogo, de seus dois produzidos, que utilizavam a técnica de holograma. Os personagens do jogo pareciam estar projetados holograficamente sobre a área de jogo. Ele tinha um toca CD e um monitor de TV, o qual apontava para um vidro circular. A imagem 2D era refletida através da superfície do vidro e produzia uma ilusão de profundidade. As imagens pareciam ter dimensão, mas o jogo na realidade não produzia verdadeiramente hologramas.

1992 ***Virtua Racing***, da *Sega*, começou não só a era dos jogos de corrida com gráficos poligonais 3D de rápida geração, mas também a dos poderosos simuladores para múltiplos jogadores. Ele combinou um estilo de jogo brilhante com uma direção controlada por um sistema de *force-feedback* e os gráficos mais realistas vistos até o momento.

1994 ***Daytona-USA***, da *Sega*, foi projetado para utilizar movimentos de longo alcance de uma câmera virtual, animações suaves e ângulos de visão de múltiplas câmeras, as quais foram desenvolvidas no seu predecessor *Virtua Racing*. *Daytona* introduziu os jogos que começavam a parecer reais. O jogo também aperfeiçoou a técnica de múltiplos jogadores, permitindo que através de 4 cabines duplas, jogassem até 8 jogadores. Para completar, o jogo contava com gráficos espetaculares e um mundo real amparado em leis físicas.

1996 ***Alpine Racer***, da *Namco*, usou um processador de 32 bits para criar gráficos poligonais estonteantes. O seu dispositivo de entrada era inovador e, combinado com seu monitor de 50", permitia, de forma natural, a imersão do jogador em seu mundo artificial.

1997 ***Super GT***, da *Sega*, usou uma arquitetura de 64 bits com o uso de versões avançadas, para a época, do microprocessador PowerPC, similares aqueles utilizados em computadores Macintosh da Apple. Este jogo opera com imagens que exigem o quádruplo da capacidade computacional do Daytona USA.

Em termos de jogos por console, a evolução pode ser sintetizada pelo histórico abaixo.

1972 ***Odyssey***, da *Magnavox*, foi colocado no mercado americano em janeiro de 1972 ao preço de US$ 100,00. Dado que circuitos integrados eram caros na época, este console foi construído usando somente 40 transistores e 40 diodos. Isto permitia a geração somente de poucos e simples efeitos na tela. Para tornar o jogo mais interessante, a unidade era empacotada com 2 controles e outros acessórios diversos (dados, dinheiro de brinquedo, roletas, cartas, etc). Em especial, o jogo vinha com algumas coberturas plásticas para o monitor da TV, cuja aplicação era simular um ambiente de jogo mais complexo. Note-se que o jogador tinha que anotar o placar, porque a máquina não era capaz de fazer isso.

1975 ***Pong***, da *Atari*, foi colocado no mercado em janeiro de 1975 e foi um grande sucesso. O console operava em preto-e-branco e tinha apenas dez variedades de jogos.

1976 ***Channel F.***, da *Fairchild*, posto a venda em agosto de 1976, foi o sistema que refinou o conceito de consoles programáveis. O atrativo da unidade era os jogos adicionais que podiam ser comprados na forma de cartuchos.

1977 ***Video Computer System*** **(VCS)**, da *Atari*, foi liberado em outubro de 1977, ao preço de US$ 200,00 e, pelo seu grande sucesso, foi comercializado até 1990. A parte central do sistema era um processador Motorola 6507 de 8-bit e 1.19 Mhz e 256 bytes de RAM. Foi o primeiro console a permitir jogos coloridos e as imagens se limitavam a pontos e traços.

1979 **Microvision**, da *Milton Bradley*, foi o primeiro sistema a combinar a portatilidade dos jogos eletrônicos manuais (*hand-held*) com a capacidade computacional de consoles do tipo VCS da Atari. A base da unidade era equipada com uma tela de cristal líquido de 2 polegadas^2 e um botão de controle, mas não dispunha de CPU. Cada cartucho vinha com seu próprio microprocessador de 4-bit.

1980 ***Intellivison***, da *Mattel Electronics*, apresentava gráficos bem mais elaborados que o do VCS da Atari, no entanto, apresentava problemas de velocidade. Outra característica deste jogo foi o inovativo *PlayCable*, um serviço disponível 24 horas que entregava jogos nas casas através de transmissões de TV a cabo.

1982 ***Colecovision***, da *Coleco*, continha 48 Kb de RAM de memória, um microprocessador Z-80A de 8-bit e 3.58 MHz e custava US$ 200,00. O fabricante planejava vários módulos de expansão para o sistema, entre eles, um que o transformava em um computador pessoal.
SuperSystem 5200, da *Atari*, era basicamente o computador pessoal Atari 400 (16 Kb de RAM) sem teclado

1985 ***Entertainment System*** **(NES)**, da *Nintendo*, teve como base o computador pessoal japonês *Famicom* (*Family Computer*) de 1983. O console liberado no Natal de 1985 tinha um microprocessador de 6502 de 8-bit, um chip gráfico customizado que gerava 52 cores, controladores, um revolver de luz e um R.O.B. (*Robotic Operating Buddy*).

1989 ***Game Boy***, da *Nintendo*, um sistema manual portátil, usava uma tela de cristal líquido preta e verde e era programável a partir de cartuchos passíveis de troca. Ele continha um microprocessador de 8-bit e 1.1 MHz, custava US$ 100,00 e foi um grande sucesso.

Lynx, da *Atari*, foi projetado pelos projetistas do computador pessoal Commodore Amiga, que decidiram criar o primeiro jogo portátil manual colorido. O jogo operava com um microprocessador de 8-bit e uma tela suficientemente grande e capaz de exibir imagens coloridas com detalhes finos. Foi comercializado a partir de US$ 149,00.

Turbografx-16, da *NEC*, foi anunciado como um jogo 16-bit, no entanto, a sua CPU era de 8-bit e o chip gráfico 16-bit. Foi o primeiro console a ter um encaixe para um leitor de CD.

Genesis, da *Sega*, foi produzido a partir de jogos de galeria de 16-bit. A concepção da Sega era de que boas máquinas de jogos apoiavam a venda de bons consoles. O Genesis contava com um processador Motorola 68000 de 16-bit e custava inicialmente US$ 200,00.

1991 ***Super NES***, da *Nintendo*, foi lançado como um console de 16-bit, com gráficos admiráveis e coloridos com mais de 30 mil cores, bem como com jogos excelentes.

1993 ***3DO*****,** da *Interactive Multiplayer*, foi o primeiro jogo baseado somente na tecnologia de CD. O console podia exibir vídeo com qualidade VHS, bem como som com qualidade de CD. O interessante sobre este jogo é que a empresa que o projetou não o manufaturou. O jogo foi disponibilizado para outros fabricantes na base de uma comissão. O Panasonic REAL FZ-1 3DO usava um microprocessador de 32 bit e 12.5 MHz e custava US$ 700,00.

1995 ***Playstation***, da *Sony*, originou-se de uma conexão de CD ao Super NES. Desacordos entre a Sony e a Nintendo sobre a forma de comercialização do sistema, fizeram com que a Sony decidisse por desenvolver a sua própria máquina, no caso, o *Playstation*, que contava com microprocessador de 32-bit e 33 MHz, especialmente projetado para produzir gráficos poligonais.

Saturn, da *Sega*, um sistema equipado com 2 microprocessadores de 32-bit e 28.8 MHz, foi projetado para operar com processamento paralelo e usufruir da popularidade dos jogos de galeria da Sega.

Virtual Boy, da *Nintendo*, foi projetado para operar com um monitor próprio, constituído de 2 visores, um para cada olho, em arranjo similar ao de um par de óculos de realidade virtual apoiado em um tripé. Cada visor operava com 4 cores e 32 níveis de intensidade. Apesar do sistema gerar imagens que realmente pareciam 3D, foi noticiado que alguns usuários reclamavam de dificuldades no ajuste da distância entre os olhos e os visores. A CPU do sistema era um microprocessador de 32 bit e 10 MHz. O preço era de US$ 180,00.

1996 ***Nintendo 64*** incorporou a tecnologia de supercomputadores da década de 80 através do uso de processadores de 64 bit e 93.75 MHz. O projeto foi feito em conjunto pela Nintendo e a Silicon Graphics. O sistema de controle, por si só, já era revolucionário, pois foi projetado especificamente para jogos 3D. O console foi comercializado por US$ 150,00.

2000 ***Playstation 2***, da *Sony*, lançado em março de 2000 no Japão ao preço de aproximadamente US$ 370,00, é o console mais avançado até o presente momento. Ele é equipado com um processador de 128-bit (RISC / MIPS-IV) e 294.91 MHz, com memória de 32 MB, três co-processadores, um de ponto-flutuante e dois vetoriais, atinge a performance de 6.2 Gflops em cálculos vetoriais. Com iluminação processa 34 milhões de polígonos por segundo. Com iluminação e névoa (*fog*) processa 30 milhões. Com iluminação, névoa e curvas de Bézier, processa 13 milhões. Apresenta taxa de desenho de *sprites* (8x8) de 18,75 milhões, tem encaixe para CD e DVD e pode exibir filmes em DVD. Este sistema, por suas características e pelo impacto inicial que teve, oferece as condições para direcionar os jogos por computador para limites incomensuráveis, bem como acirrar drasticamente a disputa por este mercado milionário. Aguarde-se, por exemplo, o projeto XBOX da Microsoft.

2- Tipos de jogos por computador

O tipo de um jogo define algumas de suas características de interface e de motor, com implicações claras nas técnicas de implementação. A descrição dos principais tipos é mencionada a seguir [ARAUJ98], [FALST97], [MOTHE94], [VIDEOT].

Para melhor definir os tipos de jogos é importante salientar que, dependendo do tipo, a interface define uma forma particular de atuação do usuário no universo do jogo. Uma possibilidade é o usuário atuar como uma terceira pessoa (o usuário se vê na cena), com uma visão descrita na literatura de *God's Eye* (olhar de Deus) ou onipotência, usada geralmente em narrações por ser normalmente uma visão superior de todo o sistema. Por exemplo, em jogos de lutas, o usuário se vê no jogo como um lutador. Outra possibilidade é o usuário atuar em primeira pessoa (a cena exibida representa o que os olhos do usuário vêem), como em alguns jogos de aventura em que o usuário caminha por túneis atirando em inimigos.

Quando a interface é em terceira pessoa, o personagem que representa o usuário na cena é denominado de ator. Quando é em primeira pessoa, o personagem é denominado de *avatar*. O ator é parte integrante da cena e deve ser bem caracterizado, o que exige um visual sofisticado e bem elaborado. O *avatar* simplesmente marca o usuário na cena e a sua sofisticação deve ir até o limite em que não prejudique a ilusão de imersão do usuário no jogo. [GARD00].

A interface também pode variar em termos de projeção gráfica. Quando o jogo é feito sobre um grande mapa, é usual se utilizar projeções planares ortográficas (planta, vista lateral, vista frontal e axonométrica isométrica). Estas projeções são aquelas que compõem a descrição de um objeto em uma folha de desenho técnico tradicional. Projeções perspectivas são utilizadas quando a noção de profundidade é requerida, como, por exemplo, quando um personagem de um jogo entra atirando em um túnel.

Estratégia: são jogos idealizados com o objetivo do usuário tomar decisões de grandes conseqüências. Pertencem a esta categoria os jogos de planejamento de cidades, como o *SimCity*, o *SimCity 2000* e o *Colonization*, os de planejamento de países e planetas, jogos de guerra, como o *WarCraft*, nos quais o usuário comanda exércitos ou esquadras, e qualquer outro jogo cuja função principal é a conquista de um objetivo através de análise crítica da situação e que possibilite um desafio mais intelectual do que de reflexo. Em termos de interface, estes jogos se caracterizam por serem usualmente programas 2D ou $2^{1/2}$D (simulação do 3D através de texturas). A lógica operacional destes jogos é normalmente complexa por possuírem estruturas e bases de dado extensas. Por outro lado, não requerem interação em tempo real. O esquema de implementação destes jogos é próximo ao de aplicativos mais convencionais.

Simuladores: são, normalmente, jogos de âmbito tático, com uma visão em primeira pessoa. São jogos que, salvo os de ficção científica, buscam levar em consideração a física do ambiente, sendo seu principal objetivo a imersão do usuário no ambiente proposto. A visibilidade do usuário se restringe ao que é fisicamente (ou tecnologicamente) possível ver. Nesta categoria se enquadram os simuladores de carro, simuladores de avião e qualquer outro simulador de máquinas que tente modelar o real e ponha o jogador numa perspectiva de primeira pessoa. Normalmente, a interface é 3D e utiliza polígonos com texturas de alta qualidade para maior realismo. Em termos de lógica operacional, a complexidade destes jogos é alta, porém é limitada pela necessidade de interação em tempo-real. As estruturas são, usualmente, grandes, devido principalmente aos dados físicos do ambiente, e são necessárias formas de programação não convencionais para permitir a continuidade do jogo sem a intervenção do usuário.

Aventura: é uma classe de jogos que combina ações baseadas em raciocínio e reflexo. O objetivo do jogador é ultrapassar estágios que envolvam a solução de enigmas e quebra-

cabeças para chegar ao final do jogo. Note-se que, para pertencer a esta classe, os quebra-cabeças dos jogos devem ser implícitos. Por exemplo, no jogo *The Dig* há um esqueleto de tartaruga alienígena que deve ser remontado para poder avançar na estória, o que caracteriza um quebra-cabeça disfarçado. Também nesta classe se enquadram os jogos de detetive, os derivados de filmes de ação, como, por exemplo, *Indiana Jones: Last Crusade* e *The Dig*. A interface destes jogos é, em sua maioria, $2^{1/2}$D, utilizando técnicas de movimentação de textura. O usuário atua em terceira pessoa, sendo que este, normalmente, está como auxiliar de um personagem principal. O tempo, nestes jogos, pode ou não ser importante para a pontuação, no entanto, o objetivo central é a resolução de problemas. A base de dados é, em geral, grande, porque o usuário desfruta de muitos graus de liberdade. Porém, o grau de complexidade do motor diminui com o uso de texturas com padrões constantes.

Infantil: são jogos que tem como público alvo as crianças e enfocam quebra-cabeças educativos ou histórias simples com o objetivo de divertir a criança. Estes jogos são caracterizados por imagens bonitas e coloridas, tendo visual próximo ao de desenhos animados. A maior prioridade é no visual e na facilidade de interação. A criança, normalmente, atua como uma terceira pessoa, auxiliando um personagem principal. A trama do jogo é usual e simples e segue, normalmente, o método de histórias em série, no qual a criança progride quando resolve um quebra-cabeça. Nesta classe de jogos, um destaque é para Gustavinho, um jogo infantil brasileiro que mescla filmagens de uma atriz com o personagem principal, o Gustavinho, que é um desenho. Outro jogo interessante é *JumpStart Baby* destinado a crianças de nove meses a dois anos de idade cujo objetivo é divertir e favorecer o reconhecimento de palavras. O Mestre, jogo educativo para crianças desenvolvido por pesquisadores do Departamento de Psicologia da UFSCar, também merece destaque.

Passatempo: são programas simples, com quebra-cabeças rápidos e sem nenhuma história relacionada, cujo objetivo essencial é atingir uma pontuação alta. Os jogos de passatempo podem imitar jogos de tabuleiro (damas, xadrez, go, etc), de cartas (*poker*, paciência, sete-e-meio, etc) ou de qualquer outra forma de jogo onde o objetivo é simplesmente o entretenimento puro. Estes jogos têm, usualmente, interfaces 2D simples. O motor, no entanto, pode ter uma lógica complexa.

RPG: *Role Playing Game* são implementados em computador com o mesmo objetivo de um RPG convencional. Sua perspectiva é, normalmente, feita em primeira pessoa, apesar de atualmente estar se mudando esta tendência, pois os programas mais recentes são, em sua maioria, em terceira pessoa. A implementação é complexa, devido à gigantesca base de dados que este tipo de jogo deve ter, pois os fatos inventados devem possibilitar ao jogador vários caminhos. Os objetos são, normalmente, texturas.

Esporte: são programas que simulam esportes populares, como os jogos de futebol, futebol americano, vôlei, basquete, boxe, basebol, etc. Como o usuário comanda times inteiros, os jogos têm interfaces 3D que, em geral, fazem uso extensivo de *sprites* 3D. Em termos de programação, os motores destes jogos têm os mesmos problemas dos simuladores porque a ação é normalmente em tempo real.

Educação/Treinamento: Jogos educacionais podem envolver as características de qualquer um dos jogos anteriores. Por exemplo, um jogo de aventura pode envolver enigmas relativos a um templo maia ou a um castelo medieval e, assim, se tornar um jogo educacional sobre história antiga. Um simulador de vôo pode ser aperfeiçoado para treinar pilotos de avião. O que diferencia os jogos de educação ou treinamento dos jogos somente para diversão é que eles levam em conta critérios didáticos e pedagógicos associados aos conceitos que visam difundir.

Os jogos também podem ser classificados em termos do número de usuários, ou seja, mono ou múltiplos usuários. Alguns jogos, mono ou múltiplos usuários, também podem ser jogados via Internet. Estes tipos de jogos são discutidos no capítulo 7.4.

Note-se que, em função da crescente capacidade gráfica do hardware disponível para jogos, a tendência é se implementar jogos que façam uso extensivo de interfaces 3D sofisticadas com objetos constituídos de polígonos que são renderizados (item 6.3) em alta velocidade. Outra tendência é disponibilizar jogos sofisticados em rede para múltiplos usuários, no entanto, os recursos para isto ainda não estão totalmente disponíveis.

3- Atual importância mercadológica, técnica, científica e educacional de jogos por computador

Atualmente, jogos por computador têm sido largamente explorados em termos comerciais. Dados da *Interactive Digital Software Association* (IDSA), a associação que organiza a *Electronic Entertainment Expo* (E3) demonstram que em 1999 foram comercializado em torno de US$ 6,1 bilhões em software de entretenimento só nos Estados Unidos. Em 1996, este volume era US$ 3,7 bilhões, com venda de 109 milhões unidades de software de jogos, sendo 59 milhões de programas para PCs. Pesquisa da IDSA também mostra que, em 1996, jogos em PCs eram mais usados do que processadores de texto. A pesquisa também mostra que 72% dos jogadores têm menos de 18 anos e que 40% dos usuários adultos são mulheres, sendo elas também responsáveis por 33% das compras de consoles de jogos. Todos estes dados justificam uma empresa gastar em um único projeto US$ 40 milhões de dólares e um tempo de implementação de 3 anos, caso do jogo japonês *Fantasy VII* da *Square*, que ganhou no Japão o título de melhor jogo do ano de 1997, ou, então, porque a Sony investe centenas de milhões de dólares no Playstation 2.

Fontes extra-oficiais estimam que o investimento em pesquisa e desenvolvimento na área de jogos é da ordem de US$ 2,7 bilhões por ano, o que é maior do que o investimento da indústria farmacêutica. No Brasil, jogos disponíveis em CDs têm uma venda estimada em 200.000 unidades por ano, o que corresponde a 6% do mercado total de venda de CDs. Isto sem considerar a pirataria.

Detalhes comerciais à parte, um ponto importante a ser considerado nesta área é a possibilidade de se combinar entretenimento com educação. Infelizmente, os jogos de maior sucesso comercial atualmente são os que melhor combinam violência com efeitos visuais sofisticados. Estima-se que menos de 20% dos jogos disponíveis tem algum tipo de enfoque educacional, sendo que, em geral, os jogos mais utilizados só servem para desenvolver rapidez de raciocínio e reflexo. Logo, há um grande campo para pesquisas nesta área e, considerando a possibilidade de se disponibilizar jogos educacionais via Internet, pode-se pensar em integrar este tipo de software a programas educacionais. [ARAUJ98], [MOTHE94], [VIDEOT].

Além da questão educacional, jogos podem contribuir com dois outros efeitos interessantes: capacitação técnica e abertura de nicho de mercado.

A capacitação técnica advém do fato de que os conceitos envolvidos na implementação de um jogo são multidisciplinares, abrangendo múltiplas subáreas computacionais, tais como, linguagens, sistemas operacionais, computação gráfica, inteligência artificial, etc, bem como as áreas de psicologia e pedagogia. Ressalte-se que jogos por computador estão caminhando para se tornar filmes interativos para múltiplos usuários com acesso via Internet, assim, a tecnologia empregada no desenvolvimento de um jogo de ação também pode ser empregada na implementação de qualquer software de interação sofisticada na Internet, em especial, aqueles que envolvam educação. [ARAUJ99], [BATTA98b], [BATTA99].

Um nicho técnico que se desenvolveu de forma acentuada no país é o de software e hardware relacionado às aplicações bancárias, o que abrange sistemas operacionais de alta segurança, técnicas de criptografia, de comunicações, etc. A qualidade destes produtos está associada ao grande investimento feito pelos bancos para automatizar agências bancárias, de forma a diminuir custos operacionais e fornecer maior comodidade aos usuários. Este investimento capacitou o Brasil com um dos sistemas bancários mais evoluídos do mundo, no entanto, fora desta área é difícil enxergar outra na qual o país se destaque acentuadamente em termos de hardware e, em especial, de software. Várias razões explicam este desnível, as principais estão relacionadas a domínio tecnológico, investimento financeiro e nicho de mercado.

Neste contexto, uma questão importante para análise é a definição de uma área de atuação com valor técnico-científico suficiente para assegurar o desenvolvimento em universidades e centros de pesquisa, e que, ao mesmo tempo, permita, abrir um nicho de mercado para o desenvolvimento de software no país. Estes requisitos podem ser satisfeitos por programas sofisticados que associem entretenimento e educação, com características de jogos por computador.

Outro aspecto importante observado sobre jogos por computador é que eles, ao contrário de pacotes tradicionais, como, por exemplo, formatadores de texto, compiladores, sistemas gráficos, etc, são potencialmente mais competitivos em termos de mercado. Este fato se justifica porque cada jogo é um produto em particular e o seu sucesso não está totalmente relacionado a sua sofisticação computacional, mas sim aos atrativos lúdicos que ele fornece aos usuários. Neste sentido, a tão propalada "imaginação do brasileiro" pode ser um fator bastante positivo no desenvolvimento deste tipo de produto.

4- Definição de um jogo por computador

Um JC pode ser definido como um sistema composto de três partes básicas: **enredo**, **motor** e **interface interativa**. O sucesso de um jogo está associado a combinação perfeita destes componentes. [ARAUJ98], [MOTHE94], [VIDEOT].

O enredo define o tema, a trama, o(s) objetivo(s) do jogo, o qual através de uma série de passos o usuário deve se esforçar para atingir. A definição da trama não envolve só criatividade e pesquisa sobre o assunto, mas também a interação com pedagogos, psicólogos e especialistas no assunto a ser enfocado pelo jogo. Este elemento é tratado no capítulo 5.

A interface interativa controla a comunicação entre o motor e o usuário, reportando graficamente um novo estado do jogo. O desenvolvimento da interface envolve aspectos artísticos, cognitivos e técnicos. O valor artístico de uma interface está na capacidade que ela tem de valorizar a apresentação do jogo, atraindo usuários e aumentando a sua satisfação ao jogar. O aspecto cognitivo está relacionado à correta interpretação da informação gráfica pelo usuário. Note-se que em termos de jogos educacionais, a interface deverá obedecer a critérios pedagógicos.O aspecto técnico envolve performance, portatilidade e a complexidade dos elementos gráficos. Técnicas e conceitos de implementação de interfaces são abordados no capítulo 6.

O motor do jogo é o seu sistema de controle, o mecanismo que controla a reação do jogo em função de uma ação do usuário. A implementação do motor envolve diversos aspectos computacionais, tais como, a escolha apropriada da linguagem de programação em função de sua facilidade de uso e portatilidade, o desenvolvimento de algoritmos específicos, o tipo de interface com o usuário, etc. Técnicas relacionadas à implementação do motor de um jogo são mencionadas no capítulo 7.

5- Técnicas de criação de um roteiro

A trama (enredo) de um jogo surge de diferentes formas. Ela pode surgir de uma idéia original ou então ser baseada em uma história em quadrinhos, um personagem de um desenho animado, um fato histórico, um enredo de um filme de sucesso, um esporte, um objetivo educacional, etc. Uma definição clara da trama e do que a originou inspira a imaginação dos desenvolvedores e faz com que eles implementem um jogo em que a narrativa, o processo de interação, o visual da interface e o desenrolar das jogadas transpareçam de forma absolutamente coerentes ao usuário. Neste sentido, note-se que muitos jogos, tais como, *Wing Commander IV*, *Spycraft* e *The 11th Hour*, são aventuras narradas interativamente.

Definida a trama, a etapa seguinte é a definição do roteiro. A menos da interatividade, a elaboração do roteiro de um jogo obedece a algumas regras similares ao dos roteiros de filmes ou de séries de televisão. Um objetivo destas regras é enfatizar a importância da integração entre a concepção do idealizador, a equipe que elabora o enredo e aquela que faz a implementação. Outro objetivo é compatibilizar o fator lúdico do enredo com as técnicas disponíveis para a sua implementação, bem como com as restrições de prazo e custo [LEWIN98].

Como na indústria do cinema e da TV, o roteiro (*story bible*) na área de jogos também documenta de forma linear o enredo, define os personagens, a sua ação, os cenários e qualquer detalhe adicional importante, bem como o que originou a trama e possíveis desdobramentos dela para um futuro jogo. Observe-se que o roteiro não é a especificação da implementação do jogo, o roteiro apenas trata da história e de seus elementos.

Desde que muitos jogos apresentam uma estética cinematográfica, é natural que o roteiro seja escrito em um formato similar ao de filmes ou séries de TV, onde os personagens, mesmo aqueles que foram sintetizados computacionalmente, tenham sua fala, ação, caráter, vestimenta e tomadas de câmeras bem definidas. A característica adicional que o roteiro de um jogo deve enfocar é a interatividade. Enquanto um roteiro para cinema ou TV ocupa uma página para cada minuto de ação, o que perfaz em torno de 120 páginas para um filme e 60 para uma série de TV, o roteiro de um jogo pode requerer algumas centenas de páginas. Isto porque o roteiro de um jogo deve considerar as ramificações da trama, o que acarreta, dependendo das ações do jogador, múltiplas cenas e vários finais para o jogo.

A estrutura básica do roteiro de um jogo é composta de quatro itens: sinopse, personagens, cenários e comentários finais. Cada uma delas é abordada a seguir.

Em sinopse é sintetizada toda a trama do jogo. No caso de um jogo baseado em um enredo cinematográfico, a sinopse também deve conter uma análise da ação que transcorre do primeiro ao terceiro ato, no mínimo. Esta parte do roteiro deve ser elaborada prioritariamente pelo idealizador do jogo.

No item personagens é descrito as características centrais de cada personagem. Este item é dividido em: 1) descrição sucinta de suas características e papel na trama (nobre atlético e inteligente que salvara a princesa e o seu castelo), 2) nome, 3) idade, 4) aparência (jovem musculoso que se veste com roupas de grife), 5) equipamentos que opera (revólver, canhão laser, etc), 6) caráter (gentil e defensor dos fracos e oprimidos) e 7) origem (nascido de família nobre e educado em escola de renome).

No item cenários é descrito, para cada etapa do jogo, o local físico onde ocorre a ação, a era, o clima, a temperatura, o som, etc. Se um determinado local é habitado por fantasmas ou monstros, ou então escondem armas ou armadilhas, todos estes elementos devem constar da descrição daquele cenário em particular.

O item comentários finais contém observações e sugestões variadas. Em especial, pode haver sugestões de como o final do enredo pode ser a ponte para um novo jogo, de como transformar o personagem principal em um ator de desenho animado ou história em

quadrinho, de como associar uma música de sucesso ao jogo para ressaltar sua trilha sonora, de como se aproveitar o jogo para fazer propaganda de um produto, etc.

6- Técnicas de implementação da interface gráfica de um jogo por computador

A interface gráfica de um jogo é fundamental para aumentar o seu realismo e o nível de engajamento do usuário ao seu ambiente. Além disso, o atrativo visual desta interface, baseado na sofisticação artística e beleza das imagens, é um fator de grande importância para o sucesso de um jogo. Adicionalmente, a interface deve também considerar outros fatores importantes como a facilidade de interação, a rapidez de resposta, a incorporação ou não de vídeo, trilhas sonoras dependentes da cena, som 3D, etc. Em termos de implementação, deve se considerar também o ambiente gráfico a ser utilizado. Parte destas características serão abordadas nos itens subseqüentes.

6.1- Animação, *sprites* e renderização de polígonos

Jogos por computador têm como um dos seus principais atrativos a interatividade, o que requer a movimentação de personagens e objetos. O termo animação, no contexto da computação gráfica e dos jogos por computador, se refere a qualquer alteração em uma cena em função do tempo. As características passíveis de modificação, neste caso, são: posição, forma, cor, iluminação, material, transparência, etc. [MOTHE94], [ARAUJ98], [HOOK97].

Entre as diversas técnicas de animação, três se destacam em ambiente computacional: quadro-a-quadro, *sprite* e renderização de polígonos.

A animação quadro-a-quadro segue o mesmo método da exibição cinematográfica onde a idéia de movimento é alcançada pela exibição seqüencial de quadros a uma determinada taxa, usualmente, 24 quadros por segundo. Esta animação é geralmente pré-definida e não-interativa, o que, mais a exibição em seqüência fixa dos quadros, limitam a utilização desta técnica em jogos por computador. Assim, as técnicas de *sprites* e de renderização de polígonos são as de maior interesse de jogos por computador.

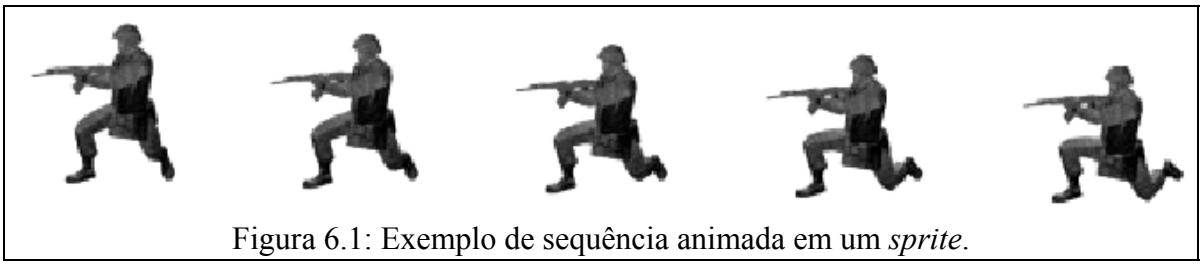

Figura 6.1: Exemplo de sequência animada em um *sprite*.

Os implementadores dos primeiros jogos, em função da limitação dos recursos computacionais, tiveram que elaborar técnicas de animações interativas compatíveis com o nível destes recursos. Uma destas técnicas é a de *sprites* (fig. 6.1), um objeto gráfico bidimensional que se move sobre a tela, sem deixar marcas sobre a área que passa, como se fosse um espírito. Note-se que a palavra em inglês *sprite* é derivada da palavra em latim *spiritus*, espírito, e significa duende, fada, espírito, alma ou fantasma. [PRITC95].

Um *sprite* é composto de uma seqüência de imagens (texturas), usualmente retangulares, que definem a movimentação de um determinado elemento. Em função desta movimentação, as imagens são substituídas na tela. Para permitir que não só elementos retangulares sejam exibidos através de um *sprite*, as imagens, normalmente, contém áreas que devem ficar transparentes durante a sua exibição. Esta operação pode ser implementada via cores transparentes ou aplicação de máscaras. No segundo caso, associa-se uma máscara ao *sprite*. Para se sobrepor à imagem sobre a tela, inicialmente, se faz um AND entre a máscara e a área do fundo onde ele será colocado, depois uma operação XOR entre esta área do fundo e

o *sprite*. Para se recompor o fundo basta se armazenar em memória a área em questão e recuperá-la posteriormente.

O uso de *sprites* sobrepostos ao mapa de bits da tela de fundo, usualmente denominado de *tile*, permite implementar uma animação rápida do objeto. Ao *sprite* se associa altura, largura, posição, estado de visibilidade, prioridade sobre outros *sprites*, transformações de escala, translação e rotação, controle de colisão com outros *sprites*, manipulação de tabela de cores, etc. Os *sprites* podem ser 2D (movimentação em x e y) ou 3D (movimentação em x, y e z).

Em termos de interface, os jogos podem ser 2D, $2^{1/2}$D e 3D. Os jogos 2D e $2^{1/2}$D fazem uso de *sprites* e/ou texturas e os jogos 3D renderizam em tempo real os personagens e cenários.

Jogos 2D se caracterizam essencialmente por utilizar mapas de bits, através do uso preferencial de *sprites* e de técnicas relacionadas, como *double-buffering* e *scroll*. Jogos $2^{1/2}$D, ou *sprite* 3D, fazem uso extensivo de técnicas computacionalmente simples para simular uma cena 3D sem renderizações custosas. Esta técnica se divide em 2 classes: *sprite*-planares e *geo-sprites*. A primeira faz uso de texturas aplicadas a objetos 3D simples, como planos. Para se compreender o seu funcionamento pode-se tomar como exemplo um homem que caminha na direção do ponteiro das horas de um relógio. Se uma textura mapeada sobre um plano inidica o homem caminhando na direção das 2 horas, basta espelhar verticalmente o plano e a textura para se indicar o homem caminhando na direção das 10 horas. A técnica de *geo-sprites* é mais sofisticada. Neste caso, os elementos do jogo são descritos em termos de polígonos, mas ao invés de serem renderizados, eles são preenchidos com texturas pré-definidas e armazenadas em memória. Para economizar espaço em memória, tenta-se definir personagens simétricos, para que se armazene somente metade das texturas associadas ao personagem. Quando o personagem não é simétrico, por exemplo, ele tem um armamento pesado encaixado em apenas um dos seus braços, esta técnica não funciona. [SICKS97a], [SICKS97b].

Jogos 3D têm vários objetos constituídos de polígonos ou malhas de triângulo passíveis de renderização. Este tipo de técnica permite um aspecto visual e uma movimentação mais natural dos personagens e dos cenários, no entanto, é extremamente custosa em termos computacionais. A melhor solução para contornar este problema é aumentar o poder computacional do hardware, o que a Sony tenta viabilizar com o *Playstation 2*.

6.2- A importância do DirectX para Jogos por Computador

Nos idos do sistema DOS, os implementadores de jogos se preocupavam em implementar jogos com várias rotinas em *Assembler* para gerar código eficiente e pequeno e controlar dispositivos de entrada (*joystick*, *mouse*, teclado) e saída (placas gráficas tipo *VGA*, placas de som, etc).

Com o surgimento do MS/Windows, os implementadores continuaram a utilizar o DOS como plataforma de desenvolvimento de jogos devido à dificuldade de se implementar código suficientemente eficiente para este tipo de programa no MS/Windows [MOTHE94]. Para resolver este problema, a Microsoft lançou o DirectX, que é um conjunto de funções que permitem aos jogos usar recursos do Windows 95 de uma forma eficiente [KOLB97]. O DirectX alterou o rumo do desenvolvimento dos jogos por computador, dado que esta tecnologia provê estratégias, tecnologia e ferramentas que auxiliam na implementação da próxima geração de jogos e aplicações multimídia. O DirectX fornece ao programador várias rotinas necessárias para implementar um jogo, além disso, como ele está intimamente ligado ao Windows, ele aumenta a portatilidade do jogo, pois se o usuário possuir uma placa de som e um *joystick* que o Windows reconheça, não há a necessidade de se reconfigurar o hardware porque o DirectX se encarrega do interfaceamento. Isto se aplica também a monitores, placas de vídeo, *mouse* e outros periféricos tratados pelo Windows.

O *DirectX* é dividido em seis módulos: *DirectDraw*, *DirectInput*, *DirectPlay*, *Direct3D*, *DirectSound* e *Direct3Dsound*. Todos merecem destaque, no entanto, o destaque neste texto será para o *Direct3D*, devido a um questionamento importante que ele traz ao implementador.

O Direct3D [GLID97] trata primitivas gráficas 3D no mesmo nível que a biblioteca gráfica OpenGL [WRIG96] da OSF. Ele pode operar em dois modos: *Imediate* e *Retained*. O modo *Imediate* é mais básico e contém um conjunto pequeno e enxuto de rotinas que permitem o traçado imediato de uma primitiva gráfica 3D. Suas funções são mais parecidas com as funções básicas da *OpenGL*. O modo *Retained*, mais complexo, assume algumas funções que o implementador de um jogo pode preferir controlar de forma particular, como, por exemplo, o de agrupamento de um conjunto de primitivas gráficas para formar um único objeto (arma, casa, carro, etc). Assim, se o implementador quer ter maior controle sobre a sua interface gráfica, necessitando apenas das rotinas gráficas que façam traçados e renderizações, deve optar pelo modo *Imediate*. Ao contrário, se quiser dispor de mais recursos da biblioteca gráfica, deve utilizar o modo *Retained*. [THOMP96].

6.3- Técnicas importantes de modelagem de objetos em computação gráfica

A definição de todas as técnicas de modelagem gráfica exigiria um curso a parte. Assim, a idéia deste item é apenas fornecer uma noção de como esta modelagem é realizada em ferramentas gráficas do tipo 3D Studio Max da Kinetix, Maya da Alias|Wavefront, etc.

Inicialmente, o primeiro conceito que se deve ter é o de projeções planares, dado que a tela do computador pode ser considerada o plano de projeção de um objeto 3D e, assim, o usual é modelar este objeto visualizando-o de diferentes pontos de vista. Neste caso, as projeções mais importantes de serem conhecidas são as projeções ortográficas, os raios projetores partem do infinito e incidem de forma perpendicular ao plano, e as projeções perspectivas, os raios projetores partem de um ponto no finito (fig. 6.2). A vantagem das projeções ortográficas é que elas mantêm as proporções de comprimento, o que facilita alterações no objeto a ser modelado. As projeções perspectivas, por outro lado, possibilitam uma visão mais natural do objeto por manterem a noção de profundidade. [BATTA98a].

Uma forma simples de se modelar um objeto é a combinação de objetos 3D básicos. A maioria dos editores gráficos permite a geração de paralelogramos, esferas, pirâmides, cilindros, etc. Operações especiais podem ser aplicadas sobre estes sólidos, do tipo unir ou subtrair um do outro, ou então calcular a intersecção de ambos. Utilizando estas operações é possível se gerar um objeto 3D complexo (fig. 6.3).

A operação de *loft* permite a geração de um objeto 3D através da definição de formas 2D que são associadas a um caminho (fig. 6.4). A extrusão funciona de forma similar ao *loft*, permitindo que após o desenho de uma forma 2D, um texto por exemplo, ela seja replicada ao longo do eixo vertical, formando, por exemplo, um texto 3D. A revolução permite que uma forma 2D seja rotacionada em torno de um eixo para a geração de um sólido (fig. 6.5). Outros métodos de geração de sólidos são disponíveis, a escolha de um deles é influenciada por parâmetros do tipo: qualidade visual do objeto gerado, complexidade de uso da ferramenta, flexibilidade para permitir modificações e ajustes, etc. [ELLIO98], [KAKER99].

Os objetos são gerados usualmente na representação de arame, que é uma forma com menor custo computacional e serve apenas como um guia para o processo de geração. Finalizada a geração de um ou mais objetos de uma cena, pode-se passar para o processo de renderização quando ao objeto se associam as características que vão torná-lo similar a um objeto real, tais como, cores, sombras, rugosidade, brilho, etc. O termo renderização vem da palavra inglesa *rendering*, que se origina na palavra francesa *rendre*, que por sua vez, se presume originar da palavra *rendere* do Latim popular. Dentre os muitos significados de *rendering* em Inglês, um é o de "aplicação de uma camada de plástico ou cimento", operação

esta normalmente utilizada em construção civil. Assim, *rendering* significa o processo de associação de paredes, usualmente pré-fabricadas no Estados Unidos, à estrutura levantada de uma casa, o que é similar ao processo, em computação gráfica, de preenchimento da estrutura de arame dos objetos de uma cena.

Para que o processo de renderização se torne mais real, o usuário pode associar fontes de iluminação aos objetos de uma cena. A iluminação pode ser programada para usar um ou vários tipos de fontes de luz. Alguns destes tipos são descritos abaixo. [ELLIO98], [KAKER99].

1) Luz Ambiente – é a luz que permeia toda a cena e que, se não for alterada pelo usuário, produz uma luz mínima, na quantidade suficiente para tornar todos os objetos visíveis.
2) Luz *Omni* – o nome vem da palavra latina *omni* que indica em todas as direções, no caso, uma luz que se comporta como um pequeno sol, enviando iluminação em todas as direções.
3) Luz de Feixe Cônico (*Spot Light*) – equivale no mundo real às luzes de lanternas, espotes ou refletores usados em teatro, os quais criam um feixe de luz que se abre (raios luminosos não paralelos).
4) Luz de Feixe Cilíndrico (*Target Light*) – equivaleria no mundo real a um canhão *laser*, cujo feixe de luz não altera a sua forma inicial, mantendo o paralelismo entre os raios luminosos.
5) Luz Solar – luz gerada pelo sol e que ilumina um ambiente em função de sua posição geográfica e hora do dia.

Note-se que as luzes de feixe cônico ou cilíndrico, podem ser configuradas para ter seus feixes apontados para determinadas posições, as quais não se alteram mesmo que a fonte de luz se mova.

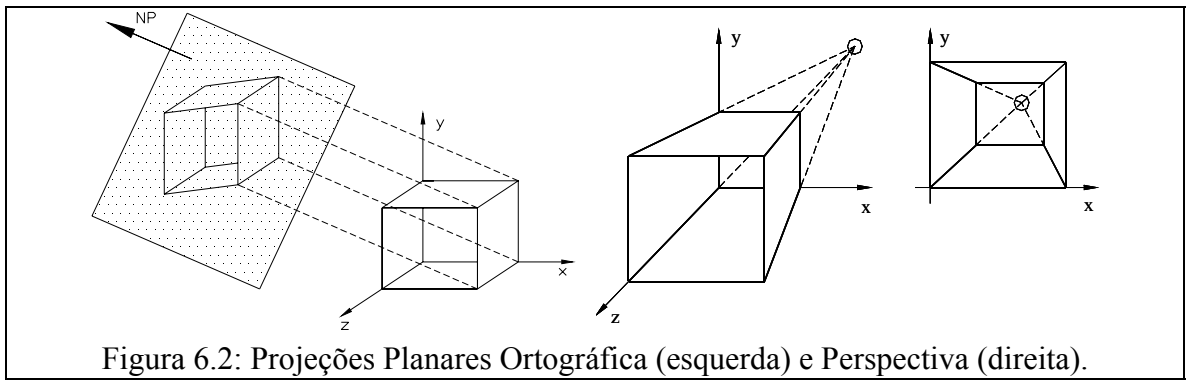

Figura 6.2: Projeções Planares Ortográfica (esquerda) e Perspectiva (direita).

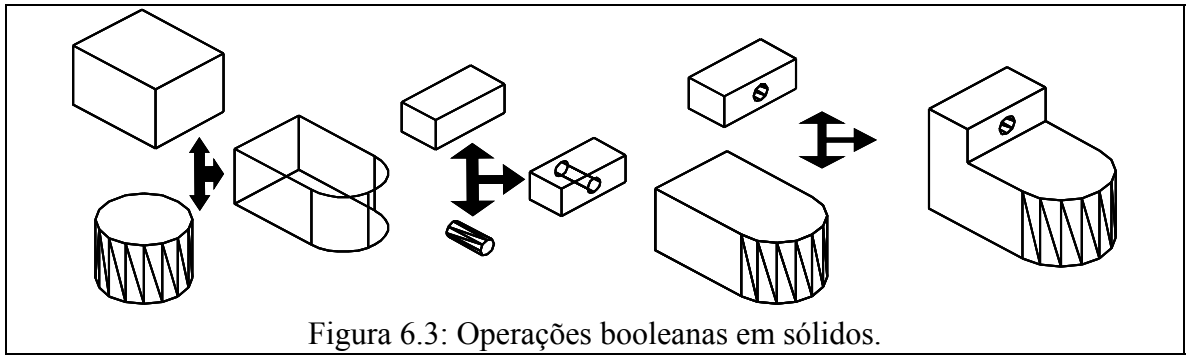
Figura 6.3: Operações booleanas em sólidos.

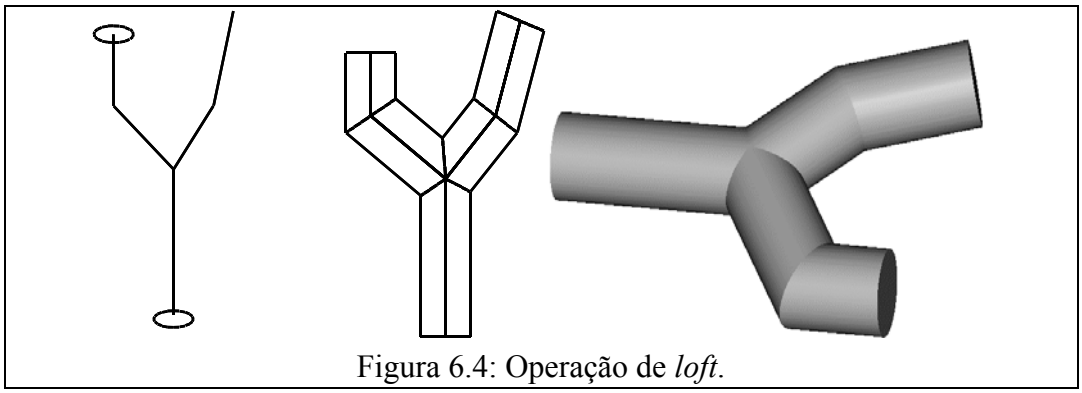

Figura 6.4: Operação de *loft*.

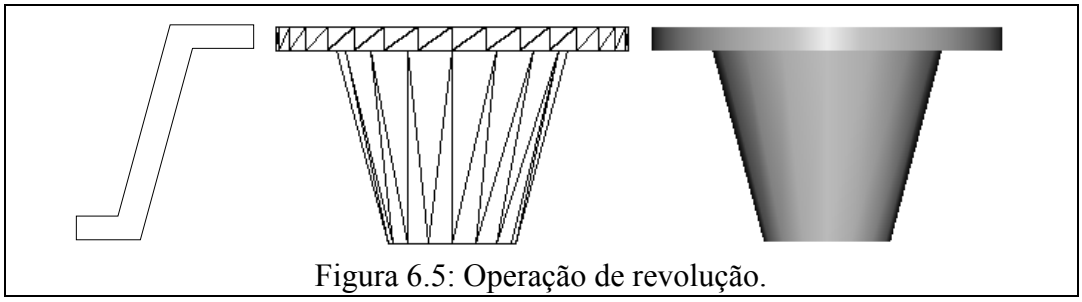

Figura 6.5: Operação de revolução.

6.4- Técnicas de diminuição do número de polígonos de um objeto

Implementadores de jogos fazem usualmente um balanço entre a qualidade visual do jogo e o seu desempenho. Técnicas de diminuição do número de polígonos de um objeto do jogo permitem que ele seja processado mais rapidamente pelas funções gráficas e, conseqüentemente, exibido mais rápido. Estas técnicas podem ser baseadas em ações tomadas diretamente sobre o objeto pelo seu criador ou então pelo uso de programas especiais, que realizam a diminuição de forma automática. [MELAX98], [STTED98]. As duas técnicas podem trazer bons resultados.

O uso repetido de operações booleanas para a obtenção de um sólido complexo implica, normalmente, na geração de muitos polígonos adicionais. Programas específicos eliminam estes polígonos através de cálculos matemáticos que se baseiam em algumas heurísticas. A principal delas é a de que uma região de pouca curvatura, formada por superfícies quase coplanares, precisa de menos polígonos para ser representada do que uma região com grande curvatura. Esta questão é crítica, pois no caso do personagem humano de um jogo, a região do seu cotovelo apresenta uma alta curvatura e, se houver eliminação de muitos polígonos nesta região, o movimento do braço do personagem ficará pouco natural.

O processo algorítmico de eliminação de polígonos conhecido como *decimation* se baseia, geralmente, na determinação de vértices de uma região que podem ser colapsados (fig. 6.6). Neste processo, para cada vértice excluído ocorre a eliminação e a geração de polígonos, sendo que, na média, pelo menos uma face e duas arestas são excluídas. Análises sucessivas dos vértices dos polígonos de um objeto (fig. 6.7) permitem que alguns programas eliminem em até 10 vezes o número de polígonos. No entanto, como esta taxa pode ser de objetos produzidos com características especiais que facilitam a obtenção destas taxas, o ideal é o implementador de jogos optar por um destes programas depois de testa-lo em algum personagem básico de seu jogo. Um exemplo de programa que realiza este tipo de operação é o *Decimator* da Raindrop Geomagic.

Seguindo as mesmas heurísticas, os vértices podem ser eliminados através de uma inspeção visual do objeto (fig. 6.8), o que indica que todo o processo de eliminação de polígonos pode ser feito visualmente. Esta forma manual de se realizar o processo é recomendável quando o objeto tem um número razoável de polígonos, no máximo em torno de 1000, ou então o objeto é construído e simplificado em partes e, principalmente, quando o modelador tem grande experiência neste tipo de tarefa. Neste último caso, quando o modelador constrói, por exemplo, o tentáculo de um monstro, ele deve saber que este objeto é usualmente mais flexível nas pontas do que na parte ligada ao corpo e, assim, deve ter mais polígonos nas pontas.

Outra técnica importante é ilustrada na figura 6.9, onde se define uma série de passos para a obtenção de um vértice intermediário em um paralelogramo. Os vértices que definem a aresta onde se situa o vértice intermediário serão movidos para ele, de forma que fique apenas o vértice intermediário. Este processo é a base para se produzir uma ponta no objeto da figura 6.10.

O número de faces escolhido para modelar um determinado objeto é outro parâmetro importante (fig. 6.11). A forma de se modelar um tubo, o qual pode ser o cabo de conexão de uma arma laser a um gerador em um jogo, pode ser modelado através de um pentágono (5 faces) com bom resultado visual. No entanto, ele também pode ser modelado através de um quadrado (4 faces) com bom resultado visual, se este tiver as arestas inclinadas em relação a horizontal (fig. 6.12).

A renderização de um sólido se dá através da coloração dos polígonos que o compõem. Esta coloração pode ser feita de diferentes formas, seguindo métodos que usualmente tomam como base a normal ao polígono (fig. 6.13). Quanto maior o número de polígonos, maior a suavidade do objeto na renderização, o que relega ao modelador o balanceamento entre a qualidade visual e o número de polígonos. *Tesselation* é o termo que se refere ao processo de se adicionar mais polígonos à representação gráfica de um objeto.

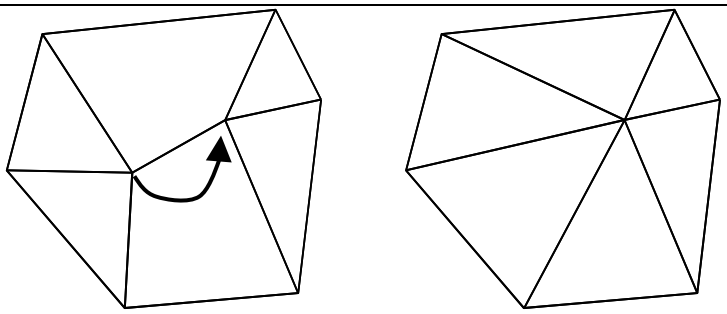

Figura 6.6: Colapso de vértices e reajuste dos triângulos de uma região.

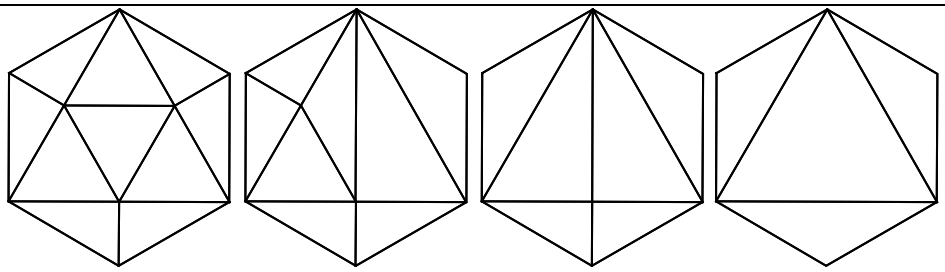

Figura 6.7: Eliminação sucessiva de vértices e polígonos.

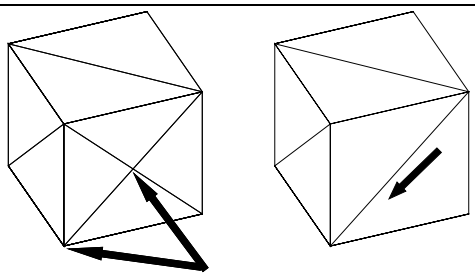

Figura 6.8: Eliminação de vértice e polígonos por inspeção visual.

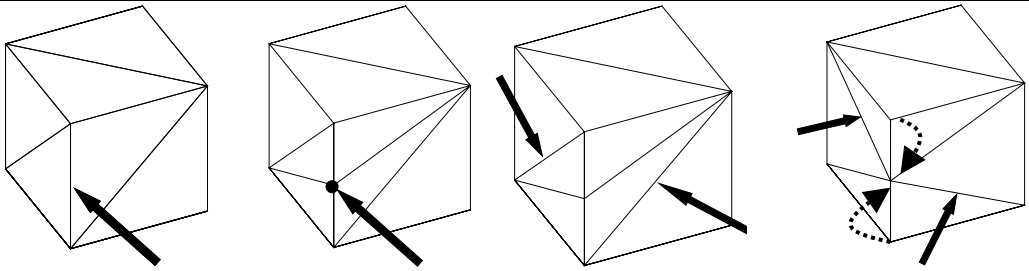

Figura 6.9: Obtenção de um vértice intermediário para colapsar.

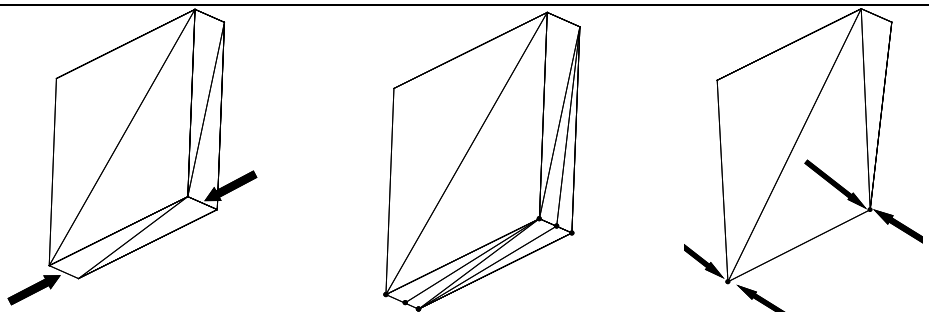

Figura 6.10: Obtenção de uma ponta através de vértices intermediários.

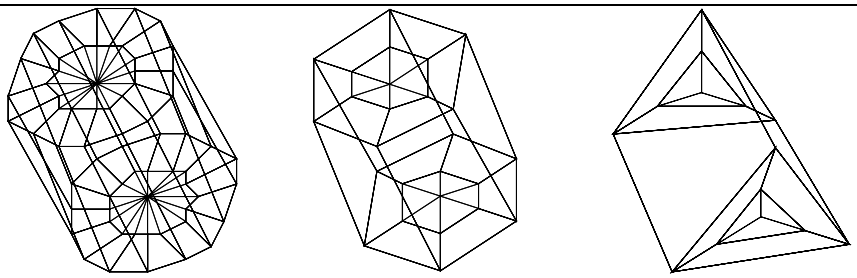

Figura 6.11: Objeto gerado com diferente número de faces.

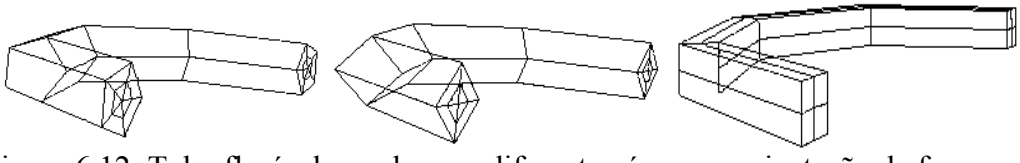

Figura 6.12: Tubo flexível gerado com diferente número e orientação de faces.

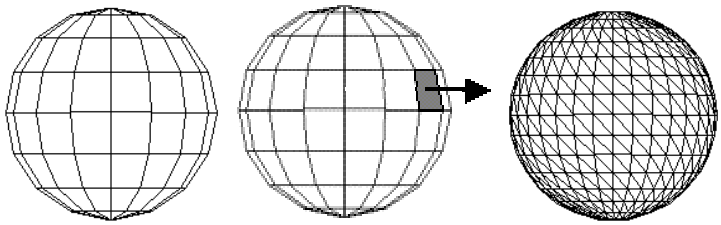

Figura 6.13: Diferentes números de polígonos usados na modelagem de um objeto.

6.5- Técnicas gerais de animação

O processo de animação que ocorria tradicionalmente nos estúdios de produção de desenhos animados estabelecia que o animador mestre desenhava o personagem em posições chaves, ou seja, por exemplo ele gerava os quadros n e n+i, e os desenhistas auxiliares os quadros intermediários, ou seja, os quadros de n+1 ao n+i-1. Este método é denominado de *keyframe*, ou quadro-chave.

Atualmente, os aplicativos gráficos fazem o papel dos desenhistas intermediários, o usuário é o animador mestre, ele estabelece os quadros-chave e o aplicativo se encarrega de gerar os quadros intermediários. De um quadro-chave para outro, o usuário praticamente pode alterar qualquer parâmetro do desenho: posição, ângulo, cor, iluminação, material, etc.

Apesar da potencialidade desta técnica, há determinadas aplicações que requerem uma forma diferente de produção da animação. Por exemplo, animar uma bailarina dançando requer que o usuário seja capaz de conhecer os passos da dança ao longo do tempo. Se o usuário criar de forma pouco criteriosa os quadros-chave da bailarina, é bem provável que a animação associe movimentos artificiais a bailarina. Para se corrigir este problema, dois métodos são passíveis de uso: *rotoscoping* e *motion-capture*.

A técnica de *rotoscoping*, no contexto da animação gráfica, utiliza os quadros de um vídeo como fundo, ou textura, para a produção de uma animação [GUYM99]. Assim, considerando-se novamente o caso da bailarina, pode-se filma-la dançando, importar o vídeo e produzir a animação considerando os seus movimentos para cada intervalo de quadros determinado.

Note-se que a técnica de *rotoscoping* também considera a possibilidade de se criar uma animação composta através da combinação de vídeos e/ou animações gráficas. Um exemplo simples neste caso é a sobreposição sobre um vídeo de um texto animado. O texto é produzido a parte e combinado com o vídeo posteriormente. Neste sentido, no caso específico do aplicativo gráfico 3D Studio MAX 3.0 da Kinetix [ELLIO98], [KAKER99], existem três modos básicos de se criar uma cena composta com *rotoscoping*: substituir o fundo pelo vídeo, aplicar o vídeo a um material ou utilizar uma ferramenta de pós-produção, o *Vídeo Post*, para processar a imagem já renderizada.

No processo de composição é importante considerar o tamanho final da animação, porque os *pixeis* de um vídeo têm escala de cores bastante variada, o que diminui o grau de correlação entre *pixeis* adjacentes e, assim, dificulta o processo de compressão.

Outro ponto de extrema relevância é o da sincronização do vídeo com a animação. Filmes e vídeos geralmente utilizam 24 qps (quadros por segundo), enquanto animações digitais rodam a 30 qps, logo é necessário um estudo sobre a qualidade final da animação para determinar qual conversão é a mais adequada, da taxa de vídeo para a de animação, ou vice-versa.

Captura de movimento *(Motion Capture)*, [LANDE98], [KNIG97], é a técnica utilizada para capturar, através de equipamentos especiais, os movimentos de objetos reais e mapeá-los

a objetos sintetizados no computador. Esta técnica é geralmente utilizada para capturar movimentos humanos, como, por exemplo, o de um lutador de artes marciais. Opta-se por esta técnica quando se exige maior naturalidade ou suavidade no movimento do personagem ou quando o movimento apresenta dificuldades para ser gerado por outras técnicas. Normalmente, os dispositivos para realizar esta captura são sensores eletromagnéticos sem fio que são acoplados a diferentes partes do corpo de um ator. Estes sensores capturam qualquer movimento do ator, desde que ele permaneça em uma área definida cujas dimensões são de aproximadamente 7.6 x 7.6 m^2 (25 x 25 ft^2). O equipamento mais comum inclui uma vestimenta de lycra, sobre a qual sensores são presos, usualmente em número de 16, dispositivos de recepção e transmissão, um servidor e software específico. Os movimentos captados pelos sensores são processados e armazenados em arquivos, cujo formato permite que eles sejam associados aos objetos gerados no computador. Eventualmente, antes da gravação dos dados, é feito um pré-processamento das tomadas para que se selecione somente aquelas que são interessantes para uso.

Outros equipamentos de captura de movimentos incluem câmeras, *flashes* infravermelhos de alta luminosidade, sincronizadores, marcadores reflexivos passivos, etc.

O uso de uma técnica de animação mais sofisticada exige um tratamento mais sofisticado do personagem. Por exemplo, considerando-se que o movimento capturado vai ser associado ao esqueleto do personagem para lhe permitir uma movimentação suave e natural, então o seu corpo deve refletir estas características. Neste caso, por exemplo, o corpo do personagem deve ter as junções de seu corpo (joelho, cotovelo, etc) modeladas com uma quantidade maior de polígonos.

Outro cuidado que se deve ter com esta técnica se refere à coerência inicial das seqüências de movimento. Se um espadachim parte de uma posição inicial para começar uma luta, então todas as tomadas realizadas devem considerar sempre este ponto de partida. Isto pode dificultar as tomadas, porque é difícil de se exigir um comportamento tão preciso de um ator humano. Note-se que uma tomada para captura de movimentos difere das tomadas cinematográficas, pois esta última apresenta uma seqüência linear ditada por um roteiro.

Uma boa estratégia de utilização da técnica de captura de movimentos em jogos é avaliar e planejar cuidadosamente o seu uso, pois é uma técnica cara e sofisticada e, por isto, não permite desperdício de investimento.

Aplicativos gráficos, como o 3D Studio MAX, permitem que se capture movimentos através do uso de periféricos como *joystick*, *mouse*, teclado e até mesmo um teclado músical MIDI. A captura de movimentos pelo teclado se dá pelo processo de quadros-chave, que é de fácil uso, e requer apenas a definição das posições inicial e final do objeto para que o computador se encarregue de animar a seqüência. A captura de movimento pelo *joystick* requer uma configuração especial, uma vez que o 3DS se baseia no *joystick Microsoft Sidewinder 3D*, e, assim, é preciso definir qual movimentos dos eixos do *joystick* afetará os eixos X, Y e Z do personagem.

6.6- Técnicas de animação de faces

Associar falas e emoções as faces dos personagens de um jogo é um problema de crucial importância porque dela também depende, em grande parte, a naturalidade dos personagens. Animar faces é um processo que segue usualmente os seguintes passos: a) a construção da face, b) a síntese da imagem facial e c) a animação da face. [RODGE98].

Não há uma maneira correta para construção de um rosto que será utilizado em uma animação. O processo mais comum divide a construção em 3 partes: 1) criar a boca pelo método radial, 2) derivar outras superfícies a partir da boca e 3) criar e inserir os olhos e o nariz.

O método radial deriva seu nome do primeiro passo de criação da boca, que é a geração de uma curva NURB 2D representando o lábio e a cavidade da boca. A curva é utilizada para, através do processo de revolução, gerar uma malha de aspecto igual ao de um vaso, a qual é modelada para atingir a forma de uma boca.

Após a criação da boca, cria-se a superfície do rosto, ou, mais especificamente, a superfície das bochechas. A boca é concatenada à superfície criada, o que gera uma única malha. O nariz e os olhos são modelados a parte e depois fundidos à malha da boca. Um tratamento especial deve ser dado a esta fusão, de forma a remover rugosidades na face.

A malha da cabeça é gerada pelo mesmo processo de modelagem da face. Posteriormente, a malha da face é fundida à da cabeça. Neste processo, pode ser preciso a edição de alguns vértices para atribuir um toque de realismo à figura final. Note-se que pode ser necessário repetir a criação da cabeça para cada expressão facial utilizada na animação.

Gerar expressões faciais complexas e reais de uma face 3D através da modelagem de malhas é uma tarefa complexa. Um método que pode simplificar essa tarefa é a combinação de modelos de faces 3D com texturas de cada expressão facial, as quais são geralmente projetadas de forma cilíndrica sobre a face 3D.

A animação da face é realizada, geralmente, através de dois métodos principais:

1) Quadro-chave – As formas de expressões específicas do modelo de um ator compõem um conjunto de quadros denominados quadros-chave que serão a guia para a geração dos quadros-intermediários via interpolação. Por exemplo, pode se associar uma curva ao formato de um olho aberto e de outro fechado, assim, o piscar de olho é conseguido através de interpolações de uma curva para a outra. As interpolações podem requerer operações de rotação, translação e escalamento dos vértices das curvas.
2) Parametrização - Partes individuais de um modelo são definidos e combinados em grupos separados em função de critérios ou parâmetros em comum. Os parâmetros podem ser: o tamanho da pupila, a altura da boca, o comprimento do nariz, etc. Neste caso, em vez de se definir posições e rotações de vértices em quadros-chave, são definidos somente valores de comprimento, altura e tamanho de elementos do modelo 3D.

A associação de falas à face é usualmente um processo complexo, cujo problema principal é a dificuldade de sincronização do movimento da boca e da face com a voz. Outros problemas são: a) a preservação do realismo da fala e da expressão em modelos com número menor de polígonos, como muitos utilizados em jogos por computador e b) o ajuste da animação total a um determinado tempo. A associação de falas a face pode ser realizada por um dos seguintes métodos: 1) pesquisa, 2) combinação de fonemas, 3) captura facial 2D e 4) captura facial 3D.

O método da pesquisa, baseado na análise de todos os movimentos dos músculos do rosto humano, requer um trabalho extenso de uma numerosa equipe de artistas. Este método foi utilizado pela Pacific Data Images para a animação do filme *Antz* da Dream Works.

Fonema é a menor unidade fonética significativa para uma língua, por exemplo, *r* e *rr* em português. O método da combinação de fonemas associa a cada fonema um padrão de malha de polígonos. Existem vários métodos e ferramentas para a realização deste processo, mas, em geral, todos seguem a mesma idéia. O processo envolve a criação de expressões para os fonemas e a associação de cada fonema a um canal MIDI controlado por uma barra deslizante. A posição de cada barra determina a percentagem de contribuição de seu fonema à expressão total. Cada canal atualiza apenas os vértices dos polígonos da face que são associados ao seu fonema, o que permite a combinação de um franzir de sobrancelhas com um bocejo. Este método possibilita uma boa interatividade, no entanto, pode apresentar problemas no controle geral do tempo da animação.

O método de captura facial 2D é amplamente utilizado. A idéia básica é capturar movimentos faciais do rosto de ator de um ponto de vista frontal. A captura dos movimentos pode ser feita por marcadores presos ao rosto do ator, os quais são rastreados por um dispositivo montado sobre a sua cabeça. Outro método utiliza uma câmera de vídeo comum e um software específico que, através de um sofisticado processo de detecção de bordas, permite rastrear somente o movimento dos lábios do ator. Os problemas associados a este método são a falta de detalhes da face e as limitações ao movimento natural do ator. No caso dos marcadores há também o problema da dificuldade de colocação em determinadas partes do rosto, como, os olhos, por exemplo.

O método de captura facial 3D retorna a maioria das informações sobre o movimento da face, o que inclui movimentos de cabeça e quase todas as expressões faciais. A captura é realizada por equipamentos especiais que permitem o escaneamento 3D do ator, os quais retornam uma grande quantidade de dados que requerem a análise e a filtragem adequada. Um problema deste tipo de método é a dificuldade de se manipular a geometria da face após a aplicação do movimento a ela. Para evitar este problema, pesquisadores trabalham em um método alternativo que combina este método com o de fonemas.

6.7- Inserção de vídeos em jogos

Vídeo é um dos elementos dos jogos por computador que tem apresentado cada vez maior uso. No entanto, muitas vezes, os implementadores de jogos não sabem como utiliza-lo da melhor forma. Note-se que quando bem usado, o vídeo pode aumentar o prazer e a compreensão da trama do jogo, reforçar uma narrativa, introduzir um personagem, etc. Quando mal usado, ele pode matar o jogo. [WAGG99].

Os vídeos imersos em ambientes interativos foram denominados inicialmente de vídeos interativos. Atualmente, a nomenclatura adotada é FMV (*Full-Motion Video*), significando vídeo com movimentação total.

A história de uso de vídeos em jogos começa em 1983 com *Dragon's Lair*, o primeiro a usar vídeo analógico. *Sherlock Holmes: Consulting Detective*, liberado em 1992, foi o primeiro a usar vídeo digital. O primeiro jogo com vídeo de sucesso foi *The 7th Guest*, da Trilobyte, liberado em 1993. O mais popular de todos foi *Myst*, da Cyan, jogo liberado para PC em 1994 e que contribuiu para expandir significativamente o mercado de entretenimento interativo e de *drivers* para CD. *Myst* utilizou adequadamente o vídeo, através de pequenas inserções que se casam com o contexto geral do ambiente do jogo. Na seqüência, vieram os bons jogos *Phantasmagoria*, 1996, e *Wing Commander III*.

Uma regra importante para a utilização de vídeo em jogos é o da naturalidade da transição, ou seja, a mudança de vídeo para a animação gráfica e vice-versa deve estar totalmente de acordo com o contexto do jogo, possibilitando que, em alguns casos, o usuário nem a perceba. O uso de um vídeo longo e maçante no começo do jogo, ou então, de vídeos curtos e desconexos da trama no começo de cada nível não é um procedimento que vai de encontro a esta regra.

Além do encaixe perfeito com a trama e a interface do jogo, a inserção de vídeo em jogos também deve obedecer algumas regras técnicas de produção. Estas regras mais aquelas necessárias para a utilização da técnica do cromaqui são citadas abaixo.

O cromaqui é uma técnica que consiste em filmar uma cena em que o cenário de fundo é totalmente pintado de verde ou azul, o qual depois será removido e substituído por uma imagem, animação gráfica ou outro vídeo. Tradicionalmente, a técnica utilizava equipamentos ópticos especiais para a eliminação do fundo. Atualmente, placas gráficas permitem a captura do vídeo por um computador e a sua edição através de programas específicos que possibilitam também a realização do cromaqui.

As regras para o uso eficiente do cromaqui são:

1) Qualidade do cenário de fundo. Fundos verdes ou azuis mal pintados acarretam efeitos indesejáveis no momento da eliminação do fundo, fazendo com que um personagem de cabelos longos apresente pequenas manchas ao redor dos fios.
2) Curvatura do cenário. Uma pequena curvatura no eixo vertical do cenário suaviza a luz e ameniza brilhos indesejáveis.
3) Iluminação. O cenário deve ser iluminado suavemente de frente, com as iluminarias posicionadas acima ou do lado deste. Nunca se deve posicionar as iluminarias por trás ou muito perto da câmera. O objetivo é eliminar variações de contraste. Um instrumento denominado de *waveform monitor* é utilizado normalmente para medir a iluminação sobre o cenário e testar a sua homogeneidade. Outro cuidado é evitar objetos lustrosos no cenário.
4) Tamanho do cenário. O cenário deve ter dimensões suficientemente grandes para possibilitar a movimentação apropriada ao ator, bem como permitir a iluminação diferenciada do ator e do cenário.
5) Distinção entre a iluminação do ator e a do cenário. Uma iluminação homogênea sobre o cenário e outra forte sobre o ator traz normalmente problemas para o processamento em cromaqui. O ideal é manter o ator suficientemente afastado do cenário, para que as luzes sobre ele não atinjam o cenário.
6) Formato da fita de gravação. Formatos de gravação de fitas para uso doméstico, como o VHS e Hi-8, não manipulam imagens complexas e altamente saturadas de forma satisfatória. Sistemas como o Digital Betacam e Digital S são mais adequados.
7) Campo de profundidade. O foco da câmera é o ator, assim, se o campo de profundidade estiver o mais próximo possível dele, o cenário ficara meio desfocado, permitindo se eliminar possíveis imperfeições dele, bem como facilitando o uso do cromaqui.
8) Iluminação do ator por trás. Uma luz quente com cor próxima do vermelho, como um alaranjado, que atinja o ator por trás define de forma marcante o seu contorno, possibilitando até que unhas e cabelos não desapareçam durante o corte do fundo.
9) Evitar a cor chave no ator. A roupa do ator, bem como algumas de suas características (olhos, cabelos, etc), devem ter cores distintas da cor chave (cor do fundo). Devem ser evitadas também cores que reflitam o azul, como, em especial, as cores metálicas e brancas. Se alguma cor do ator deve ser azul, a melhor opção é usar um fundo verde.
10) Antecipe problemas no planejamento. Como o custo do aparato para produzir uma tomada para uma operação de cromaqui é alto, o ideal é planejar a operação cuidadosamente para evitar problemas depois da tomada.

Os cuidados que devem ser tomados com a inserção de vídeos em jogos são:

1) Vídeo entrelaçado. O vídeo analógico é entrelaçado, ou seja, a imagem é dividida em duas partes, um conjunto de linhas horizontais pares e impares (1, 2, 3, 4, ...). As linhas pares são capturadas um 1/60 de segundo depois das linhas impares. Quando os objetos filmados se movimentam muito rápido, durante a exibição do vídeo no computador surgem borrões com padrões que evoluem em escada e ocasionam um efeito visual muito ruim. A maioria dos programas de edição digital de vídeo permite o desentrelaçamento do vídeo.
2) Taxa de exibição de quadros errada. Vídeos normalmente têm taxa de 30 qps (quadros por segundo) e filmes de 24 qps. Estas taxas têm de ser compatibilizadas para evitar problemas de sincronização nas cenas. Programas como o *Media Cleaner Pro 4* da Terran Interactive tratam deste tipo de conversão.

3) Relação de aspecto errada. Muitos formatos de vídeo digital utilizam uma resolução de 720x486 pixeis por quadro, onde os pixeis são mais altos do que largos. Quando o vídeo é exibido no computador, há um ajuste que torna a imagem mais larga. Neste caso, é necessário se escalar o vídeo tomando como base a manutenção da relação de aspecto de circunferências e quadrados.
4) Manchas nas bordas. Quase sempre é possível se encontrar manchas nas bordas de vídeos analógicos que não são vistos em um monitor de TV, mas sim no de um computador. Estas manchas tem de ser cortadas das bordas.
5) Atores ruins. A escolha dos atores deve ser feita com cuidado pois há um ditado entre os diretores de produção de vídeo de que um bom ator não repete a sua boa performance em um grande palco com platéia na frente da objetiva de uma câmera.
6) Diálogos ruins e pobres. Usualmente o diálogo é mais difícil de ser finalizado do que a encenação. Isto porque sons são uma constante em jogos e normalmente se considera sua produção mais simples do que a do vídeo, o que nem sempre é verdadeiro. Além do esmero técnico, é necessário que os atores responsáveis pela dublagem tenham experiência e aptidão para a tarefa. Adicionalmente, é preciso que os técnicos de som tenham a qualificação requerida para a produção de um som limpo e claro.
7) Áudio de 8-bit. O recomendável é nunca se usar áudio de 8 bit. Os modernos compactadores de som (*codecs*) geram arquivos maiores e de menor qualidade sonora para o áudio de 8-bit do que para o de 16-bit.
8) Combinação inadequada do vídeo com o fundo renderizado. Deve haver um casamento perfeito entre os elementos de ação no vídeo (ator, armas, motos, etc) recortados pela técnica do cromaqui e o fundo renderizado do jogo. O ideal é o produtor do cromaqui ter uma idéia do fundo do jogo em termos de cores, luzes, materiais, etc, para poder planejar as tomadas e possibilitar este casamento.
9) Requisitos técnicos do vídeo diferentes dos de todo o jogo. Os jogos são planejados para serem jogados em um computador com uma configuração mínima. Se a qualidade do vídeo exigir uma configuração mais sofisticada, detalhes sofisticados de cenas, ou até o todo, poderão não ser vistos.

A incorporação de vídeo aos jogos deve se acentuar rapidamente em função de três fatos:

1) O aumento expressivo da capacidade de armazenamento das mídias. Um CD armazena 640 Mbytes. Um DVD, sigla para *Digital Versatile Disk*, originalmente *Digital Video Disk*, armazena 9 Gbytes, capacidade suficiente para armazenar 1 filme. Um DVD-NG, DVD *New Generation*, 20 Gbytes, capacidade para 2 filmes. Um novo tipo de disco denominado FMD, sigla para *Fluorescent Multi-Layer disk*, armazena 140 Gbytes, capacidade para 30 filmes.
2) As técnicas de compressão estão se sofisticando, vide os esforços para a definição do padrão MPEG-4.
3) Os equipamentos para a produção de vídeos estão aumentando em qualidade e diminuindo em preço, o que diminui os custos de produção e populariza o uso.

6.8- Características Técnicas do Som Estéreo, do Som 3D e de sua Inserção em um Jogo

Este item discute a definição e a funcionalidade do som, do som interativo, do som digital, do MIDI e do som 3D. [GRAHA97], [MILLER97], [MOTHE94]. Uma discussão sobre o som requer uma descrição à priori de seus principais conceitos, os quais são mencionados abaixo.

Som: Tipo de energia que é transmitida através de ondas.

Ondas Sonoras: Considere a pressão do ar como a densidade de suas moléculas. Quando um objeto vibra ou se move, ele modifica esta densidade e, conseqüentemente, causa uma mudança de pressão que é captada pelo ouvido através de vibrações de ossos internos Estas vibrações são transformadas em sinais elétricos que o cérebro interpreta como som. As ondas sonoras são caracterizadas por amplitude e freqüência.

Amplitude e Freqüência: A amplitude é a altura máxima da onda em relação ao seu ponto de equilíbrio onde a altura é zero . A freqüência é o número de ondas detectadas em um determinado intervalo de tempo. A freqüência é medida em hertz (Hz), unidade que define o número de ondas por segundo. O ouvido humano é capaz de perceber vibrações dentro da faixa de 20 Hz a 18.000 Hz. Os sons fundamentais se localizam em uma faixa de 32 Hz à 4.000 Hz.

Música: Quando algum objeto vibra de forma completamente desordenada, dizemos que o som produzido por esta vibração é um ruído, como por exemplo o barulho de uma explosão ou de um trovão. O ruído é o resultado da soma de um grande número de freqüências, sendo por isto difícil de descrevê-lo em termos de alguma notação. Os sons musicais, por sua vez, utilizam apenas algumas dentre as inúmeras freqüências possíveis, as quais foram estabelecidas por convenção, constituindo-se nas notas musicais.

Escala Musical e Notas: As notas musicais por sua vez podem ser agrupadas de modo a formar um conjunto. Este conjunto recebe o nome de gama e um conjunto de gamas se constitui numa escala musical. Tanto as gamas quanto as escalas musicais podem ser construídas de diversas maneiras, por exemplo, a música oriental usa uma gama de cinco notas musicais e a ocidental uma gama de sete. Entre as diversas gamas existentes, a mais popular de todas é a chamada GAMA NATURAL ou GAMA de ZARLIN, que utiliza as notas denominadas dó, ré, mi, fá, sol, lá si e novamente dó. Estes nomes foram atribuídos a Guido de Arezzo, que foi um músico italiano que viveu no século XI. [MUSICA].

Intervalo Acústico: Na Gama de Zarlin é definida a variação de uma nota para outra através de um intervalo acústico I que é a razão entre as notas F1 e F2 (I = F1/F2), onde F1 é maior ou igual a F2. Assim, conforme descrito na tabela Notas_Musicias, os intervalos entre as notas consecutivas da Gama Natural podem assumir apenas os valores 1 (Uníssono), 9/8 (Tom-Maior), 10/9 (Tom-Meno), 16/15 (Semiton) e 2 (Oitava). Note-se que para introduzir uma nota intermediária entre duas notas consecutivas de freqüências F_1 e F_2 pode-se proceder de duas maneiras: aumentar a freqüência de F_1 ou reduzir a freqüência da nota F_2. O primeiro método chama-se sustenir e o segundo bemolizar.

Freqüência Padrão: Na Gama de Zarlin, a freqüência universalmente aceita como padrão é a da nota LÁ de índice 3 ($LÁ_3$), cujo valor é igual a 435 Hz. Assim, a freqüência do DO_1, a nota fundamental, é de 65,25 Hz.

Cifras: As notas, de acordo com o sistema americano, também podem ser definidas através de letras denominadas cifras, conforme descrito na tabela Notas_Musicais.

Pitch: O Mensuralismo, sistema inventado por Walter Oddington e Franco de Colônia no século XII, também ajudou a evoluir a técnica musical, por permitir a medição do tempo sonoro, determinando uma duração específica para cada nota (breve, semibreve, mínima, semínima, etc.). Assim, uma partitura musical é composta de notas e tempos. Em inglês, em especial, nos programas de edição de som, o tempo de duração é denominado de *pitch*.

Notas_Musicais			
NOTA	**FREQÜÊNCIA**	**INTERVALO ACÚSTICO**	**CIFRA**
$Dó_n$	F		C_n
$RÉ_n$	9/8f	9/8	D_n

MI_n	5f/4	10/9	E_n
$FÁ_n$	4f/3	16/15	F_n
$SÓL_n$	3f/2	9/8	G_n
$LÁ_n$	5f/3	10/9	A_n
SI_n	15f/8	9/8	B_n
$DÓ_{n+1}$	2f	16/15	C_{n+1}

Intensidade: A intensidade de um som está associada a sua amplitude, maior a amplitude, maior a intensidade do som e vice-versa. A intensidade de um som é medida em decibéis (dB) que é uma escala logarítmica. Uma onda de 0 dB está no nível mínimo de audição do ser humano. Um trovão apresenta uma intensidade sonora de aproximadamente 120 dB, o que está perto do nível máximo que um ser humano pode ouvir sem provocar dor ou danos ao ouvido.

Timbre: Uma única onda sonora produz um único som. A combinação de diversos tipos de som, como no caso dos sons gerados por um instrumento, produz uma onda sonora composta que no conjunto atribui uma qualidade (timbre) ao som de cada instrumento.

Sinal e Gravação Analógica: As vibrações sonoras foram e são usualmente capturadas através de um método mecânico, onde em geral um diafragma de metal fino vibra e os aparatos conectados a ele transformam esta vibração em um sinal elétrico que é gravado em um determinado material do tipo disco de vinil ou fitas cassete. Este método e este sinal elétrico são conhecidos como analógicos. Este tipo de gravação ou reprodução apresenta algumas limitações físicas, uma é a degradação do sinal em função do uso, já que dependem do tipo de material utilizado, como, por exemplo, discos de vinil ou fitas cassete. Uma vantagem deste método é a faixa de freqüências capaz de ser operada. Sistemas analógicos profissionais podem operar em faixa de 10 Hz a 100 KHz.

Playback: Para transformar um sinal analógico em som (*playback*), é necessário utilizar alto-falantes que vibram de forma a gerar as ondas sonoras.

Gravação Digital: Consiste na conversão do sinal analógico em números binários e gravação em mídia apropriada, como, por exemplo, os discos a laser (CD). A gravação digital é livre de impurezas ou ruídos e a reprodução do som original é clara, no entanto, apresentam duas limitações: 1) banda superior da faixa de freqüências limitada a 20 KHz e 2) a descrição de um sinal suave, como o analógico, ser feito através de uma discretização por amostragem (quantização). Usualmente, estas limitações não trazem resultados perceptíveis para os ouvintes.

Taxa de Amostragem: É o número de vezes em que algum tipo de equipamento analisa uma onda sonora por segundo para, em especial, transformá-lo em um código digital. Assim, quanto maior a taxa de amostragem, maior a quantidade de códigos de descrição do som em uma unidade de tempo, o que torna a gravação digital mais fiel ao som original. Como exemplo, pode-se citar as seguintes taxas de amostragem por segundo: Padrão Telefônico-8000, Rádio Digital-32000, Padrão CD-ROM/XA de alta qualidade-37800 e Padrão DAT (*Digital Audio Tape*)-48000.

Qualidade do Som: Uma partitura associa notas musicais com instantes de tempo, o que define uma duração fixa para a música e requer que as informações sobre um som, ou um trecho dele, devem ser enviadas sempre no mesmo intervalo de tempo para o equipamento que irá reproduzi-lo. Se houver limitações em termos da capacidade de transmissão, o que se pode fazer é diminuir a taxa de amostragem do som e, conseqüentemente, o número de informações sobre ele. Isto, claramente, também diminui a qualidade do som. A tabela Qualidade_Som ilustra esta idéia.

Qualidade_Som		
Qualidade	**Taxa de Transmissão Kbytes/minuto**	**Tamanho do Arquivo MB**
CD	10.584	42,3
MP3	960	3,8
FM	480	1,9
AM	240	0,9

Resolução da Amostragem (*Bit-Depth*): Um modelo simples de alto-falante como o de um microcomputador PC possui um cone que vibra de forma controlada. Quando a vibração é definida por um número binário, quanto maior o número de bits maior a flexibilidade na reprodução de um som e, portanto, maior a sua fidelidade ao som original. Por exemplo, um som definido por 8 e 16 bits permite, respectivamente, 256 e 65.536 diferentes vibrações.

Formatos de Arquivos de Som: Atualmente há vários formatos de arquivos para o armazenamento de som e utilização em computadores. Destes formatos, seis merecem destaque e, assim, são mencionados a seguir.

WAV: O Microsoft *Windows Riff Wave* popularizou-se em função do grande número de plataformas com o Microsoft *Windows* e seus aplicativos. O Wav é atualmente a base do áudio digital, sendo largamente utilizado em efeitos sonoros. Existem vários tipos de ferramentas que manipulam este formato, permitindo a geração dos mais variados tipos de efeitos, bem como a utilização das mais variadas taxas de amostragem. Dependendo da qualidade utilizada para um arquivo Wav, ele pode requerer uma taxa de transmissão de 40 MBytes por minuto, o que o torna muito pesado para operar em computadores de pequeno porte. Em função da sua grande difusão, todos os outros formatos possuem conversores para Wav.

MIDI: A interface MIDI (*Musical Instrument Digital Interface*) define uma linguagem de transmissão de dados digital entre sistemas computacionais, sintetizadores e instrumentos musicais. A comunicação pode ser feita com base no protocolo ou em arquivos MIDI. Por exemplo, quando uma nota é tocada em um teclado, a porta de comunicação MIDI envia para um outro instrumento ou computador conectado a ela, todas as informações pertinentes, tais como, a nota, a sua velocidade, tipo, etc. O MIDI não é áudio digital, portanto não há a possibilidade de se gravar vozes ou efeitos sonoros, ele somente mantém a seqüência de notas tocadas e armazena-as com um tipo de som pré-definido, sendo eles variados, como guitarra, teclado, gaita, etc. Isto acentua a flexibilidade deste tipo de som, pois permite que todas as suas características possam ser mudadas facilmente de forma manual (direta) ou via programa. A característica mais interessante do MIDI é o pouco espaço em disco que ele ocupa, por exemplo, 1 minuto de som com só uma trilha e sem nenhum evento a ele ligado requer 3Kb por minuto, o que comparado com o formato Wav e muitos outros é extremamente pequeno. Note-se que esta comparação não considera a questão da qualidade.

MP3: O formato MP3, *Moving Pictures Experts Group Audio Layer 3 compressed audio*, objetiva armazenar som compactado. A sua definição é baseada em um modelo psico-acústico, ou seja, o ouvido humano não é capaz de ouvir todas as freqüências, há um limite entre 20Hz e 20KHz e ele é mais sensitivo entre 2KHz e 4 KHz. Assim, um algoritmo elimina grande parte das freqüências que um ouvido humano possa não escutar e ainda algumas que possam ser retiradas sem que haja perda na qualidade sonora. Ressalte-se que esta compactação é destrutiva, ou seja, a compactação do MP3 elimina informações que nunca mais poderão ser recuperadas. A maioria dos *rippers*, dos codificadores e dos decodificadores trabalha com 44 KHz e 128 Kbits estéreo o que torna o arquivo com qualidade igual ao CD e 12 vezes menor que um arquivo igual do formato Wav.

MP4: Utiliza o mesmo conceito do MP3, porém apresenta melhor algoritmo de compactação, o que tem aumentado o interesse por ele. O MP3 compacta a razão de 11:1 enquanto o MP4 a 16:1. O ganho de 30% advém de otimizações realizadas no algoritmo.

VQF: Este formato, *Transform-domain Weighted Interleave Vector Quantization*, ou TwinVQ, ou ainda VQF, permite um nível de compactação de 18:1 ou mais vezes.

RA: O formato *Real Audio* foi desenvolvido para uso na Internet. Ele permite que o som possa ser ouvido com boa qualidade, diretamente pela rede, em tempo real, sem a necessidade de *download*.

6.8.1 Áudio Interativo

Esta técnica consiste na modificação em tempo real do som para que este pareça mais real em uma determinada circunstância. Como usualmente as aplicações apresentam limitações de espaço, de hardware e de software, e requerem flexibilidade, se opta pela utilização do formato MIDI. Neste caso, através de ferramentas especiais é feita a conversão de áudio digital em MIDI.

O áudio digital é utilizado prioritariamente em aberturas ou cenas sem interatividade, dado que os formatos para este tipo de som são pesados e pouco flexíveis. No Wav, por exemplo, o processo de adicionar efeitos e modificar o som é demorado e complexo. Uma alternativa para tornar o áudio digital um pouco mais interativo é a gravação de música ou efeitos sonoros em pequenos trechos e das mais variadas formas, incluindo mudanças de tempo e velocidade, freqüência, modulações e efeitos como *reverb*, *wah-wah*, envelope, etc.

Em função destes problemas e com as otimizações no formato MIDI, ele se tornou a opção mais atrativa para áudio interativo. Isto, tanto pelo tamanho dos arquivos, quanto pela facilidade de modificações e alterações. Com a utilização de ferramentas pode-se fazer modificações radicais em um MIDI sem muita demora e esforço. Por exemplo, tendo-se um MIDI de alguma música do Beethoven em piano, pode-se, dentre os vários efeitos, modificar o volume da trilha, realizar *pan* (translação) entre vários trechos, modificar o estilo do som pré-definido de piano para guitarra em algum trecho, em outro colocar o som de uma gaita e depois ainda adicionar o som de uma sirene de bombeiros.

Devido à facilidade de manipulação do MIDI, foram criadas linguagens de programação simples para se alterá-lo. Neste caso, por exemplo, em um jogo, o personagem está para entrar em uma situação de combate, antes o som ambiente estava calmo, no momento em que os outros personagens aparecem o som muda, fica mais pesado, mais rápido. Isto é facilmente programável em termos de recursos MIDI. Neste contexto, um algoritmo para a simulação de som interativo em um jogo é ilustrado abaixo.

```
se personagens.quantidade>5 e personagens.posição==esquerda
então
        SomMIDI.Track1.velocity=6
        SomMIDI.Track1.patch='Distortion Guitar'
        SomMIDI.Track1.pan=34 …
fim
```

Muitos profissionais da área de jogos afirmam que o futuro dos jogos está no áudio interativo. Com melhorias em hardware e software, os processos disponíveis para o MIDI poderão estar disponíveis para o áudio digital, o que acentuará a qualidade deste tipo de som.

Outros, no entanto, argumentam que na vida real não existe uma música ambiente e, assim, não se deve acentuar a importância do som. Neste caso, pode até se diminuir a

quantidade de música nos jogos, mas não os efeitos sonoros, pois, com certeza, eles incrementam o prazer de jogar.

Um problema que o áudio interativo em jogos apresenta é o fato de que muitos programadores não dispensam a atenção devida ao som, consideram, usualmente, o som de difícil programação e também não procuram desenvolvê-lo em parcerias com músicos ou compositores.

6.8.2 Áudio 3D

A idéia do áudio 3D acompanha as tendências 3D de jogos e outros aplicativos atuais. No entanto, diferentemente da imagem, o som 3D é mais complicado de ser manipulado e requer uma visão correta do que seja. Áudio 3D não é a manipulação do som entre esquerda/direita ou frente/trás. Usualmente se confunde este conceito com o efeito *pan* no som estéreo. Áudio 3D é a possibilidade de se colocar o som em qualquer lugar em torno da cabeça do ouvinte. Para alcançar este efeito, o ideal é o ouvinte estar equipado com um fone de ouvido porque a posição do ouvido está bem definida em relação aos ouvidos. Outra possibilidade é se utilizar som quadrifônico ou *surround*, com as caixas em posições definidas. Se o som vier de duas caixas à frente do ouvinte, fica difícil de gerar som 3D.

Para evitar confusões acerca de som 3D, é interessante se reportar às definições abaixo.

Áudio posicional 3D: técnica de manipulação do sinal do som que visa explorar todo o espaço à volta do espectador.

Virtualização: técnica aplicada ao sinal do som para que se perceba mais caixas de som do que realmente existe.

Espacialização: forma como o sinal do som é modificado para que não haja uma idéia precisa da localização das caixas de som.

Aperfeiçoamentos têm ocorrido nesta área, mas ainda existem barreiras a serem transpostas, como, por exemplo, o áudio 3D depende da posição da cabeça do ouvinte em relação às caixas, logo para ouvintes diferentes podem ocorrer erros de posicionamento. O ideal neste caso seria o usuário utilizar um fone de ouvido.

Para criar um ambiente com áudio 3D, deve-se observar os 5 itens descritos abaixo.

Reverberação: O som reflete em paredes ou em outros objetos, e este efeito deve ser simulado para diferentes locais. Técnicas para este fim utilizam algoritmos que simulam objetos geométricos para a determinação de distâncias e formatos.

Distância: A distância entre objetos é muito importante em termos de intensidade de som, pois quanto mais perto se está de um prato que se espatifa no chão, mais alto se ouve o som da quebra.

Efeito Doppler: Um objeto em movimento emite sons diferentes à medida que anda.

Absorção do ar: O ar absorve o som à medida que este se propaga. Este efeito pode ser um dos mais difíceis de serem tratados porque devem ser criados ambientes que tenham características especiais para a ocorrência deste fenômeno.

Objetos à frente: O som é emitido por detrás de um objeto e refrata para poder contornar o objeto em outras direções.

7- O motor de um jogo

Como mencionado anteriormente, a definição das características básicas do motor de um jogo é função do seu tipo. No entanto, usualmente, as técnicas envolvidas na sua implementação incluem: estruturação e classificação de dados, mecanismos de comunicação, métodos sofisticados de controle dos personagens e do mundo, de detecção de colisão, de

sincronização animações gráficas, imagens e som, etc. Além disso, como qualquer outro software, se exige do motor níveis satisfatórios de performance, de robustez, de modularidade e de reusabilidade de código [CORLE98], [FALST97], [LANDE99], [MORRI96], [MOTHE94].

Como, em função dos limites impostos a abrangência do curso, não se pode abordar todos estes itens, alguns deles são mencionados abaixo. O maior destaque foi dado para jogos em rede utilizando Realidade Virtual, isto devido à possibilidade de se empregar algumas das técnicas na implementação de software educativo disponível via Internet.

7.1- Estrutura de dados de um jogo

Ao projetar a estrutura de dados de um jogo, a primeira questão que surge é "como as características do jogo irão influenciar na definição desta estrutura?". Um exemplo simples, neste caso, é o fato do jogo ser em níveis ou não. Se o jogo for em níveis, o projeto pode ser mais simples, dado que as estruturas de dados dos diferentes níveis poderão ser independentes e, assim, o conjunto de elementos do jogo tratado por nível pode ser menor. Se o jogo se concentrar em um único nível, por exemplo, ele se desenrola somente em um grande campo de batalha, onde todos os componentes estão relacionados, a estrutura de dados pode ser mais complexa em função da extensão e do controle de todos os dados. Outra característica que se nota influenciar na estrutura de dados é a do jogo ser para múltiplos jogadores em rede. Esta característica mais a arquitetura de comunicação escolhida, cliente/servidor ou par-a-par (*peer-to-peer*), influenciam a definição da estrutura de dados, pois alteram a forma como os dados são atualizados para os diversos jogadores. [VINCK97].

Como, atualmente, o requisito de se reusar código é uma constante nos projetos de software, pois economiza tempo (e dinheiro) no desenvolvimento, outra questão que surge é "como se define uma estrutura de dados passível de uso em vários jogos?". Certamente, esta não é uma questão fácil de ser respondida de forma precisa para todos os casos, no entanto, é possível se definir algumas alternativas com base em dois parâmetros fundamentais: abstração e compressão de dados.

Em termos de abstração de dados, os elementos do jogo podem ser definidos como objetos (Prog. Orientada a Objetos), cujas propriedades são determinadas pela sua ação no jogo. Em função das características do jogo, os seus elementos podem ser abstraídos de diferentes formas, como, por exemplo, elementos estáticos (terreno, prédios, árvores, etc) e elementos dinâmicos (personagens, carros, armas, etc), os quais podem ser classificados como jogáveis (controlados pelo jogador) ou não-jogáveis (controlados pelo computador), etc.

Cada elemento terá um conjunto de informações que definem as ações que eles podem realizar. No caso dos elementos dinâmicos, por exemplo, estas informações contemplam a sua movimentação e as ações que eles podem realizar sobre os outros elementos dinâmicos, bem como sobre os estáticos.

O que se almeja com a estruturação de dados é a definição de um mapa hierarquizado que contenha as informações sobre todos os elementos do jogo e que possa ser editado de uma forma coerente, segura e rápida. Em especial, esta edição deve facilitar a manipulação dos elementos presentes na cena corrente do jogo, dado que nem todos os elementos podem estar sendo afetados quando uma determinada jogada está ocorrendo. Neste caso, é como se trabalhasse com uma janela de recorte sobre o mapa dos elementos do jogo, o que permite que os dados relacionados à janela sejam, por exemplo, armazenados em memória para acesso mais rápido.

Dado que grandes bases de dados podem requerer compressão, a questão seguinte é "como abstrair os dados para que o processo de compressão se adapte ao da janela de recorte?". Uma alternativa é estruturar os dados em duas camadas de abstrações, uma

altamente comprimida e outra fracamente. A camada altamente comprimida contém todos os dados do jogo e a fracamente apenas os dados de interesse no estado atual do jogo. Isto equivale a trabalhar com dois mapas de dados, um fortemente (mapa-1) e outro fracamente (mapa-2) comprimido (fig. 7.1). A transferência de dados do mapa-1 para o mapa-2 é realizada quando necessário, o que permite dividir o trabalho entre diversas jogadas e, assim, diminuir o impacto sobre a velocidade do jogo. A atualização do mapa-1 pode ser feita quando estritamente necessária, por exemplo, quando uma grande quantidade de dados no mapa-2 não se relacionar mais aos elementos da janela corrente, ou quando o usuário salvar uma seção ou, então, em momentos particulares do jogo, quando a resposta ao usuário puder sofrer algum atraso.

Certamente, várias outras técnicas podem ser empregadas para a definição da estrutura de dados de um jogo, no entanto, o importante é ressaltar o quão complexo esta definição pode ser e o quanto ela irá influenciar na performance do jogo.

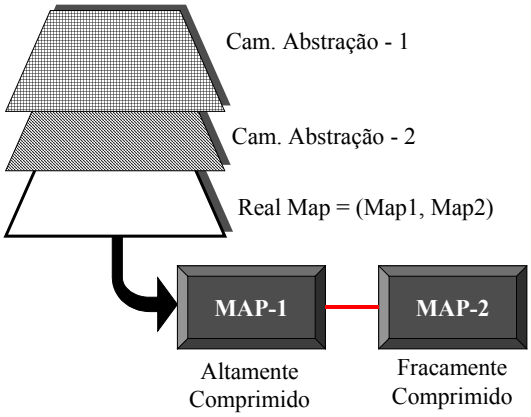

Fig. 7.1: Camadas de abstração de dados.

7.2- Implementação de um protótipo

Jogos por computador são programas e, portanto, podem utilizar as várias técnicas de gerenciamento e de desenvolvimento de software, no entanto, a sua característica especial de propiciar divertimento ao usuário permite que se adote algumas regras particulares de prototipação. Estas regras incluem definição da estrutura do jogo através de fluxogramas (retrato de níveis do jogo), definição do ambiente, o uso, quando possível, de maquetes e objetos reais para uma avaliação do mundo, das regras e do atrativo do jogo, o processo gradual de sofisticação da interface gráfica, etc. [LAZZA98].

A concepção inicial de um jogo, ao contrário de programas tradicionais como formatadores de texto, planilhas de cálculo, etc, pode sofrer modificação de maior ou menor intensidade ao longo do desenvolvimento do projeto. Neste sentido, implementadores recomendam que o desenvolvimento de jogos siga um ciclo que envolva a passagem pelos seguintes processos: projeto, prototipação, teste e aprendizado. Quantos mais ciclos são percorridos antes da implementação final, menor a possibilidade de ocorrências dramáticas no final do desenvolvimento do jogo.

Certamente, elementos como a direção de arte e a performance do código não serão prototipados até uma fase adiantada do projeto, no entanto, a concepção geral do jogo, a interação básica entre personagens, conceitos presentes no roteiro, bem como o divertimento propiciado ao jogador e a forma de se mensurar este parâmetro [GEIG98], devem ser definidos antes de se escrever uma linha de código. Neste contexto, um procedimento

interessante é a prototipação do divertimento que o jogo oferece em um ambiente fora do computador. Como muitos jogos envolvem mapas, alguns personagens e quebra-cabeças, este tipo de prototipação pode ser realizado através da simulação do jogo com maquetes simples, brinquedos de plástico, desenhos em papel, atores vestidos a caráter, etc. A "brincadeira" pode ocorrer em reuniões com os desenvolvedores e até com um conjunto representativo do público alvo do jogo. Neste tipo de procedimento, os pontos de interesse avaliados são: a determinação de como a trama do jogo se desenrola fora da tela do computador e do apertar de teclas do *mouse*, daquilo que as pessoas consideram interessante, de como as pessoas assimilam a trama e visualizam o desenrolar da história de um nível para outro, etc.

Depois de definida a trama, o ambiente e os personagens, segue a fase de refinamento artístico. Este processo começa com esboços simples de ambientes e personagens e eles vão sendo aperfeiçoados gradativamente. Isto pode ser feito até pelo escaneamento dos esboços em papel e aprimoramento através de ferramentas especiais disponíveis no computador. Além disso, a exibição de uma seqüência de esboços pode transmitir a sensação dos movimentos presentes no jogo.

Posteriormente, modelos 3D, tais como personagens e elementos estáticos do jogo (terreno digitalizado, árvores, etc), são exibidos no jogo, em estrutura de arame ou coloridos através de renderizações rápidas, para testar a interação presente nele. Nesta fase, podem ser aprimorados os posicionamentos de câmeras, o estilo da interação e a forma de movimentação de alguns personagens no ambiente. Se o jogo incorpora movimentos de personagens reais através do uso do vídeo e da técnica de cromaqui, o interessante é montar o cenário com modelos virtuais e estudar a sua ação, de forma a facilitar o processo de movimentação e iluminação dos atores no cenário do cromaqui.

Definidos todos os elementos do jogo, é feita mais uma avaliação de todo o processo de interação do jogo com o usuário para que, caso necessário, se faça ajustes. Finalmente, passa-se para a etapa final que é o da renderização de alta qualidade dos elementos do jogo. Este processo pode levar horas ou dias. A avaliação final do jogo será feita pelo usuário.

7.3- Sistemas de autoria de jogos

No contexto dos jogos, ferramentas de autoria são programas que fornecem o máximo de recursos para facilitar o trabalho de desenvolvimento de um jogo, possibilitando a economia de esforço e tempo. Estas ferramentas são elementos essenciais de projeto.

Os implementadores do jogo *Hyperblade*, da Activision, consideraram que era necessário disponibilizar um ambiente de criação de jogos que reduzisse a quantidade de código a ser escrito e que fosse acessível para não-programadores. O sistema de desenvolvimento ADLIB (*Authoring and Design Language for Interactive Behavior*) foi criado com esta finalidade. O ADLIB é constituído de três partes: 1) uma linguagem para descrever o comportamento e a interação dos personagens, 2) um ambiente de criação e 3) um sistema de tempo de execução. [ROSEN96].

A linguagem do ADLIB é baseada em planos, que são coleções de declarações de procedimentos e interações, determinando como um personagem reage em situações variadas. Os planos controlam também outros aspectos da cena 3D, como o trabalho das câmeras, a música, etc. A linguagem é declarativa, orientada a objetos, e possui vocabulário extensível capaz de descrever em alto nível entidades, circunstâncias, interações, personagens, ambientes, obstáculos, salas, seqüências de luta e animações.O ambiente de criação é um conjunto de ferramentas que define, desenvolve e mantém uma base de dados (modelos, animações e vocabulário). Modelos 3D são importados em um formato padrão de ferramentas de modelagem e convertidos para um modelo próprio visando aumento de desempenho

A animação é feita por uma ferramenta própria que permite a criação de segmentos de animação a partir de movimentos capturados. Os quadros podem ser visualizados, montados, misturados com velocidade independente e com controle direcional para cada um dos segmentos e transições.

A necessidade de acesso a ferramentas gratuitas, para múltiplas plataformas, com código aberto e amplo, deu origem a algumas iniciativas para a definição e implementação de ferramentas de autoria para jogos. Uma destas iniciativas originou o *Allegro*, sigla recursiva para *Allegro Low Level Game Routines*, que é uma biblioteca que disponibiliza aos programadores C/C++ as rotinas de baixo nível que são normalmente utilizadas na implementação de jogos, tais como, as rotinas para o controle de entrada de dados, de tempo das animações, de som MIDI, e de geração de gráficos e de efeitos de som. A biblioteca começou a ser desenvolvida por Shawn Hargreaves e, atualmente, o trabalho conta com a ajuda de programadores de todo o mundo. [ALLEG1], [ALLEG2]. Alguns dos recursos da *Allegro* incluem:

1) Facilidade de uso – vem com ampla documentação e exemplos explicativos;
2) Extensão - disponibiliza a grande maioria das funções necessárias para a implementação de um jogo;
3) Portatilidade – sem praticamente a mudança de uma linha de código, é possível mover um programa Allegro para compiladores em diferentes ambientes, tais como, Windows, DOS, Linux, BeOS e outros. Diversos compiladores são passíveis de uso, incluindo DJGPP (um compilador 32-bit para as linguagens C/C++ que é grátis e completo) [DJGPP] e VC++;
4) Código aberto – o código é aberto e qualquer programador pode contribuir para a sua modificação;
5) Grátis – é totalmente gratuito.

Uma das bibliotecas *Allegro* disponíveis é a *Allegro-GL*, que permite o uso das suas rotinas em conjunto com as da biblioteca gráfica *OpenGL*. A *jaw3d* é outra biblioteca disponível que permite uma programação 3D ampla e pode ser usada para criar aplicações 3D simples em praticamente qualquer sistema operacional.

7.4- Jogos em Rede

Os jogos por computador, em sua maioria, são monousuário. Aqueles que suportam múltiplos jogadores, normalmente o fazem em pequeno número, por questões de limitações tecnológicas nas áreas de *software* e *hardware* (modelagem e sintetização de ambientes realistas em tempo-real) e de redes de comunicação (o estado dos objetos no ambiente do jogo muda no decorrer do jogo, e esta mudança tem que ser refletida, em tempo-real, no ambiente compartilhado por todos os jogadores, o que nem sempre é possível satisfatóriamente quando os jogadores estão separados geograficamente e interligados através de redes de comunicação de longa distância). [ARAUJ98], [MORRI96].

No início dos anos 80, com a disponibilização das redes de comunicação em escala crescente, uma variação dos jogos centrados em gráficos 2-D que fizeram, e ainda fazem, enorme sucesso são os MUDs *(Multi-User Dungeons)*. MUDs são ambientes imaginários cujos contextos variam desde a resolução de quebra-cabeças até roteiros complexos de aventura e violência. O que diferencia os MUDs de jogos tradicionais, é que o conteúdo dos MUDs é basicamente texto. Os MUDs são um grande sucesso. Um sucesso que muitos jogos para múltiplos usuários, com interfaces gráficas complexas, não conseguem repetir. Um exemplo é o software *Community Place* (CP) da Sony, que disponibiliza para múltiplos usuários da Internet, ambientes virtuais 3D com potencial de interação. Entretanto, esses

ambientes não conseguem cativar um número de "devotos", como os MUDs o fazem, apesar deles não terem qualquer apelo gráfico.

Um dos fatores de sucesso dos MUDs, mas não o principal, é que eles não requerem qualquer equipamento sofisticado para que o usuário participe do jogo, já que até um terminal burro *(dumb terminal)* pode ser utilizado para acessar um MUD. Alguns MUDs populares, como, o LambdaMOO, têm milhares de usuários habituais.

Um fator fundamental que diferencia um jogo tradicional de um MUD é o da interação entre os participantes. O fator de atração dos MUDs, ao contrário dos jogos tradicionais, é a comunicação [CURTIS], em que o atrativo é a possibilidade de se encontrar a qualquer hora pessoas interessadas em manter conversações. Como não existe o apelo gráfico, um participante tem que descrever o ambiente e suas emoções, assim, o sucesso dos MUDs mostra a força do texto, mesmo na atual era da mídia visual. O principal objetivo nos MUDs é ganhar poder para modificar o ambiente onde acontece o jogo. Outro fator que é considerado importante para o sucesso dos MUDs é a dissimulação da identidade, isto é, os participantes fingem que são outras pessoas, podem mudar de sexo, de atitude, sem se expor, pois não há comandos para revelar a sua verdadeira identidade. Muitas variações de MUDs são encontradas hoje na Internet, tais como, MOOs, MOOSHES, MUCKs, etc.

Em termos educacionais, há um grande potencial de uso da idéia dos MUDs no estimulo a crianças e jovens a participar ativamente de ambientes que exploraram o mundo virtual, comunicar com outros participantes deste mundo, criar objetos, compartilhar conhecimento, modificar o ambiente etc. Exemplos de ambientes virtuais educativos baseados em MUDs incluem MOOSE Crossing [BRUCK] e Pet Park [RESNIC].

7.4.1. Jogos em Ambientes Virtuais

A realidade virtual (RV) é uma forma mais natural e poderosa de interação entre o usuário e seu computador. A RV é uma evolução de várias tecnologias que surgiram com a construção do primeiro simulador de vôo em 1929. O desenvolvimento dos simuladores de vôo, com o investimento em capacetes estereoscópicos e ferramentas para a construção de simulações 3D realistas, deram origem à realidade virtual imersiva, em que um usuário se sente "presente" no ambiente gerado sinteticamente pelo computador. Este sentimento de presença se deve à isolação, total ou parcial do usuário, do mundo real, através de um capacete. O usuário vê apenas as imagens que são sintetizadas nos dois monitores estereoscópicos acoplados ao capacete. Quando o usuário vira a cabeça para trás, por exemplo, os objetos que fazem parte deste novo cenário são sintetizados.

Os jogos por computador 2D têm interatividade implementada, usualmente, através de *sprites*, conforme explicado no item 6.1. Nestes jogos, o ponto de vista é fixo e não há visão em perspectiva. Na realidade virtual, o conceito de *sprite* é substituído pelo conceito de *avatar* e de objetos dirigidos por simulação (ODS).

Um *avatar* representa o usuário num ambiente virtual e pode assumir qualquer forma, dependendo da aplicação, o que vai de figuras geométricas, como círculos ou quadrados, até automóveis, aviões, partes de um corpo humano, como mãos, pés, cabeça, etc. Os *avatares* refletem as ações dos usuários no ambiente virtual. Estas ações são capturadas através de sensores ligados ao usuário ou direcionados a ele. Assim, diferentemente dos jogos 2D, eles têm ponto de vista dinâmico, calculado em função da orientação e movimentação do *avatar*. [DAMER98].

Os ODSs são objetos animados cuja complexidade pode variar de simples objetos dirigidos por um conjunto de ações pré-definidas *(scripts)* até objetos altamente complexos com ações obtidas através de técnicas de inteligência artificial. Os ambientes virtuais possuem também, assim como os jogos tradicionais, objetos estáticos que são fixos no ambiente virtual e

caracterizam o contexto da aplicação, como, por exemplo, árvores, edifícios, avenidas, rios, montanhas, etc.

Para que o sentimento de imersão da RV seja alcançado, a sintetização do ambiente tem que ser feita em tempo-real. Como as tecnologias de sintetização e de dispositivos de saída ainda estão em evolução, é questionado se a realidade virtual imersiva garante a imersão do usuário em um grau maior do que um tradicional jogo 3D, ou mesmo 2D ou $2D^{1/2}$, com interfaces altamente interativas. Isto gera controvérsias sobre o sentido da imersão, ou seja, um ambiente de RV que utiliza capacete, mas que apresenta atrasos sensíveis na sintetização das imagens e na resposta as interações, pode ser considerado menos imersivo do que um jogo tradicional com resposta rápida e maior grau de realismo? Possivelmente não, no entanto, com a evolução tecnológica dos dispositivos de E/S, como aqueles que projetam imagens diretamente na retina (em substituição aos capacetes), rastreadores de movimento mais precisos e rápidos, e nova geração de *chips* gráficos e placas adicionais de baixo custo, jogos mais atraentes e divertidos, com interface de realidade virtual, vão, provavelmente, aquecer o mercado de entretenimento.

7.4.2. Limitações dos Atuais Jogos Multiusuários na Internet

Desde o surgimento do primeiro navegador WWW, no início dos anos 90 (*Mosaic*), um esforço enorme vem sendo aplicado no sentido de oferecer tecnologias de multimídia e 3D na Internet. Um impulso nas duas direções foi dado com o advento das linguagens Java e VRML 2.0. A linguagem Java é importante porque suporta animação, interatividade e portatilidade. A linguagem VRML *(Virtual Reality Modelling Language)* surgiu da necessidade de criação de um padrão para a manipulação de ambientes virtuais 3D na WWW. A versão 2.0 do VRML, conhecida também como *Moving Worlds,* possibilita a interação com o ambiente virtual. Tanto a movimentação de objetos inanimados, quanto o controle de múltiplos usuários dentro do ambiente virtual, podem ser suportados através de uma integração entre as linguagens VRML e JAVA. [FAN96]

Essa integração entre as duas linguagens promove a possibilidade de implementação de jogos 3D para múltiplos usuários na Internet, os quais se destacam como uma das mais promissoras (e lucrativas) aplicações em rede. Entretanto, para aqueles que já tentaram participar de jogos para múltiplos usuários que utilizam VRML e Java e estão disponíveis na Internet, sabem que o acesso não é imediato, ao contrário do acesso aos MUDs. Normalmente, o usuário, para poder jogar, tem que preencher formulários, carregar e instalar *plug-ins*, *browsers* e/ou *software* cliente. Finda a fase de configuração, a interação é muitas vezes pobre, com tempo de resposta muito alto, bem como a interface não é realista, ao contrário, é bastante simplificada, ou simplesmente, o jogo não funciona. Alguns provedores de jogos suportam apenas um número limitado de usuários, chegando a fechar o registro para novos usuários, até que possam operar com um número maior.

Note-se que, para compartilhar um ambiente virtual entre múltiplos usuários, uma cópia deste ambiente tem que existir na máquina de cada um dos usuários remotos participantes, juntamente com as réplicas de todos os *avatares* e outros objetos animados. O esperado é que as réplicas dos *avatares* possam posicionar-se, comportar-se e parecer-se exatamente como os seus *avatares* originais, de tal forma que uma visão consistente do ambiente possa ser mantida entre todos os usuários participantes. A manutenção dessa consistência em tempo-real é um dos maiores desafios no desenvolvimento de jogos em rede utilizando RV.

A causa central das dificuldades da RV em rede é uma limitação tecnológica da Internet que não suporta uma forma de comunicação, denominada de comunicação multidestinatária, que minimiza o tráfego na rede consideravelmente. Steve Deering [DEER91] estendeu os mecanismos da rede Internet, o que permitiu o suporte à comunicação multidestinatária na

Internet em âmbito global, contribuindo para a geração da rede denominada de MBone *(Multicasting Backbone on the Internet)*. O MBONE é uma rede virtual que utiliza a mesma estrutura física da Internet para permitir o compartilhamento de informação multimídia entre múltiplos usuários. Várias aplicações multimídia que envolvem usuários dispersos geograficamente se utilizam atualmente do MBONE, através de ferramentas gratuitas, para vídeo e áudio-conferência, espaço de trabalho compartilhado, etc. [KUMA98].

O MBONE, entretanto, é uma rede que surgiu de um experimento acadêmico (tese de doutorado) e, assim, uma infraestrutura projetada mais cuidadosamente se faz necessária para suportar as aplicações de rede que vêm surgindo. Um grande passo nessa direção está sendo dado atualmente nos EUA, com a criação da Internet II, cujo objetivo é o estabelecimento de um *backbone* de alta velocidade interligando as principais universidades americanas para o desenvolvimento de uma nova geração de aplicações. Desta forma, as limitações para o desenvolvimento de jogos em rede começarão a ser minimizadas à medida que se tornem disponíveis redes de comunicação de alta velocidade e ferramentas para a modelagem e simulação em tempo-real de mundos virtuais.

Um dos poucos exemplos de jogos na Internet que pretende suportar um número muito grande de usuários é o *Mimaze* [GAUT97]. O *Mimaze* é uma versão 3D do jogo *Pacman* que possibilita a participação de múltiplos usuários através da rede MBONE. No *Mimaze*, o ambiente virtual do jogo é carregado de um servidor único que, dependendo do número de usuários envolvidos, pode tornar-se um gargalo.

8- Implementação de Jogos no Brasil.

No Brasil, a produção de jogos, ou mesmo a pesquisa tecnológica, é ainda bastante incipiente. Abaixo é descrita uma iniciativa nesta direção, tanto em termos acadêmicos quanto empresariais.

A *Continuum Entertainment* [CONTIN] do Paraná é uma das poucas empresas nacionais que trabalham com o desenvolvimento de jogos. *Othello* e *Outlive* são jogos sofisticados produzidos por esta empresa. Com o intuito de ilustrar esta experiência, o autor deste curso fez um conjunto de perguntas à direção desta empresa, as quais, juntamente com as respostas, seguem abaixo.

1- Por que e quando vocês decidiram começar a produzir jogos?
A Continuum foi constituída por cinco sócios fundadores que trabalham informalmente com jogos deste o final de 1995. A maioria foi estudante de informática da Universidade Federal do Paraná e resolveram criar a empresa devido aos seguintes fatores:

a) Todos eram e são jogadores fanáticos de jogos por computador.
b) Nenhum sentia a vontade de trabalhar com as áreas mais tradicionais da informática (banco de dados, sistemas comerciais, etc.).
c) O desenvolvimento de jogos é desafiante, pois os jogos normalmente puxam a tecnologia de software e de hardware, ou seja, no caso de microcomputadores, os jogos são os primeiros aplicativos que utilizam, de forma mais abrangente, as características e capacidades do hardware mais avançado.
d) Desenvolver jogos, apesar de ser um trabalho como qualquer outro, para quem gosta é algo muito divertido, e quando se faz o que gosta, se produz muito e a vontade de fazer mais persiste.

2- Quantas pessoas trabalham na empresa e qual a formação delas?
Trabalham cinco sócios que possuem a seguinte formação: 4 graduados, 1 em graduação, no Bacharelado em Informática da UFPR. Um está inscrito no programa de

mestrado em Informática da UFPR. Adicionalmente, há três contratados que possuem formação na área de Desenho Industrial.

3- Como vocês vêem a dificuldade de se penetrar no mercado nacional e internacional?
A maior dificuldade é conseguir que distribuidores nacionais e internacionais distribuam o seu produto. Esta dificuldade é gerada basicamente pelos seguintes fatores:

a) No Brasil não existe uma tradição de desenvolvimento em jogos, assim, os distribuidores nacionais não têm um processo bem organizado de análise dos jogos brasileiros. No caso dos distribuidores internacionais, muitos nem sabem aonde é o Brasil.

b) Empresas novas, que não possuam nenhum projeto lançado no mercado, são evitadas por distribuidores internacionais, ou seja, eles preferem trabalhar com jogos de equipes experientes.

c) O processo de avaliação e negociação é extremamente lento. Isto devido a dificuldades ocasionadas pela distância e pelo processo de análise de jogos pela maioria das grandes empresas internacionais. Elas recebem semanalmente, de várias partes do mundo, dezenas de jogos para serem avaliadas.

4- Considerando-se o investimento que é feito na área, com algumas empresas investindo em torno de US$ 40 milhões na produção de um jogo ou centenas de milhões de dólares na produção de um console, como o caso da Sony, como vocês vêem a capacidade competitiva de uma empresa nacional neste setor?
Pelo nosso conhecimento de mercado podemos afirmar que a grande maioria desse investimento é representada pela campanha de marketing e pela licença de alguma marca ou nome famoso. O que nos propomos a fazer no Brasil é o desenvolvimento de um jogo que custe em torno de US$ 2 milhões. Mas, no atual estágio da indústria nacional de jogos, um investimento desse montante ainda não é viável, o que nos leva a recorrer a criatividade e a capacidade do profissional brasileiro para contornar essa barreira e conseguir desenvolver jogos de alto padrão com orçamentos bem mais modestos.

5- Qual a ajuda de órgãos oficiais no desenvolvimento de um produto como um jogo?
O apoio governamental foi importante para a Continuum, pois através da INTEC (Incubadora Tecnológica de Curitiba) foi possível reduzir custos de desenvolvimento dos produtos. Note-se que a INTEC está instalada dentro do TECPAR (Instituto de Tecnologia do Paraná), o qual é ligado ao Governo do Estado do Paraná.
Contudo, este tipo de apoio parece ser raro. Normalmente os órgãos governamentais não apóiam este tipo de empresa. Apesar desse mercado movimentar cerca bilhões de dólares anualmente, as pessoas ainda não conseguiram enxergar o entretenimento eletrônico como um mercado promissor.

6- Vocês já estão fazendo alguns planos para produzir, caso possível, algum jogo para o *Playstation 2*?
Plataformas alternativas como o Playstation 2, o Dreamcast, o X-Box, o Nintendo, etc, estão em nossos planos, mas atualmente estamos concentrados na plataforma PC.

No Departamento de Informática da Universidade Federal de Pernambuco está sendo desenvolvido um jogo denominado Guararapes. O jogo retrata a Batalha de Guararapes que ocorreu perto de Recife, Pernambuco, em 1648, e envolveu quatro etnias: luso-brasileiros,

holandeses, índios e negros. O jogo de aventuras Enigmas no Campus também está sendo desenvolvido neste departamento. O objetivo do jogo é salvar da extinção criaturas sábias que possuem o conhecimento em um mundo fictício representado pelo Campus Virtual da UFPE. O intuito da sua criação é motivar os usuários a explorar este Campus. [ALBU99].

Os dois jogos utilizam um agente inteligente denominado Gaia, cuja função é adicionar mais dinamismo aos elementos do jogo, realçando a vivacidade de seu ambiente, bem como manter as regras do jogo e impedir que ações proibidas sejam realizadas pelos personagens.

No Departamento de Computação da Universidade Federal de São Carlos está sendo estruturado um trabalho multidisciplinar em Educação a Distância. Uma parte deste projeto deverá utilizar técnicas de jogos por computador para sintetizar um ambiente lúdico de aprendizagem.

9- Considerações Finais e Agradecimentos

A idéia central deste curso foi o de apresentar, de forma sucinta e modesta, alguns dos principais conceitos envolvidos na implementação de jogos por computador. No entanto, outro objetivo importante foi o de chamar a atenção para a área, ressaltando a importância de se começar a desenvolver no país, em centros acadêmicos e empresariais, tanto a tecnologia como a cultura necessária para a implementação deste tipo de aplicativo. Observe-se, novamente, que, além da importância comercial deste tipo de software, outro fator de interesse é a possibilidade de utilização da tecnologia envolvida em jogos na implementação de software educativo, cujo poder atrativo seja baseado em critérios lúdicos.

Neste contexto, o autor se considerará satisfeito se tiver contribuído para a diminuição do preconceito em relação a pesquisas nesta área, bem como para o crescimento de trabalhos que explorem, de forma direta ou indireta, o tema.

Pela contribuição na confecção de alguns capítulos deste curso, o autor agradece aos seguintes alunos do Departamento de Computação da UFSCar: Rafael Ogassawara Togami, item 6.5, Gustavo Antônio Furquim, item 6.4, Luis Henrique Lima Brochado, itens 6.8, 6.8.1 e 6.82.

O autor também agradece a professora Dra. Regina Borges de Oliveira pela ajuda na confecção do capítulo 7.4 do curso.

10- Bibliografia:

[ALBU99] Gaias Virtuais: cenários reagentes para jogos de computador – C. Albuquerque et al. – Anais do Segundo Workshop Brasileiro de Realidade Virtual – Marília, São Paulo, 1999.

[ALLEG1] *www.talula.demon.co.uk/allegro/index.html*

[ALLEG2] *www.allegro.cc/*

[ARAUJ98] Jogos 3D Interativos Multiusuários na Internet: Estado Atual, Perspectivas e Desafios – R. B. de Araujo e A. L. Battaiola – Relatório Interno do DC/UFSCar, 1998.

[ARAU99] Exploração do Uso de Realidade Virtual no Aprendizado de Habilidades Acadêmicas para o Ensino Fundamental - R. B. Araujo, A. L. Battaiola e C. Goyos - Revista Brasileira de Informática na Educação da SBC, Abril de 1999.

[ARRGH] www.arrgh.co.uk/hardware/index.html

[ATARI] www.atari-computer.de/mjaap/computer/english/hc_index.htm

[BATTA98a] Projeções e o seu Uso em Computação Gráfica - A. L. Battaiola - JAI98, XVIII Congresso Nacional da SBC, Belo Horizonte, MG, 1998.

[BATTA98b] Aplicação e avaliação do uso integrado das tecnologias de realidade virtual e hipermídia em sistemas de aprendizado - A. L. Battaiola, C. Goyos e R. B. Araujo - V Congresso Internacional de Educação a Distância, 1998, São Paulo, SP.

[BATTA99] EduGraph: Sistema para aprendizado de conceitos de computação gráfica – A. L. Battaiola, C. Goyos e R. B. Araujo - WISE-99, Workshop Internacional sobre Educação Virtual, organizado pela Universidade Estadual do Ceará com o apoio da UNESCO e IFIP, 1999.

[BRUCK] *MOOSE Crossing: Construction, Community, and Learning in a Networked Virtual World for Kids* - A. Bruckman – Tese de doutorado em www.cc.gatech.edu/fac/Amy.Bruckman/thesis/index.html

[CONTIN] *www.continuum.com.br*

[CORLE98] *Architecting a 3D Animation Engine* - S. Corley - Game Developer, Abril, 1998.

[CURTIS] *Mudding: Social Phenomena in Text-Based Virtual Realities* – P. Curtis - bin.ponton.de/~ole/ literatur/mudding.html

[DAMER98] *Avatars!* – B. Dammer - *Peachpit Press*, 1998.

[DJGPP] *www.delorie.com/djgpp/*

[DEER91] *Multicast Routing in a Datagram Internetwork.* - S. Deering - *PhD Dissertation, Stanford University,* Dec., 1991.

[DVDEN] *www.dvdden.com/console/psx2specs.htm*

[ELLIO98] *Inside 3D Studio MAX 2*, volume 1 – S. Elliot, P. Miller et. al. – *New Riders*

[FALST97] *Game Design: How Platform Choices Dictates Design* – N. Falstein - *Game Developer*, Maio, 1997.

[FAN96] *Black Art of Java Game Programming* – J. Fan, E. Ries e C. Tenitchi – *Waite Group Press,* 1996.

[GAMERE] *www.game-revolution.com*

[GARD00] *Building Character* – T. Gard – *Game Developer*, Maio, 2000.

[GAUT97] *Mimaze, a Multiuser Game on the Internet* - L. Gautier e C. Diot - Rapport de Recherche no. 3248, Setembro, 1997.

[GEIG98] Psychological Research Methods for Game Design – B. J. Geiger - *Game Developer*, Maio, 1998.

[GLID97] *Graphics Programming with Direct3D* – R. Glidden - Addison Wesley, 1997.

[GRAHA97] *Exploiting Surround Sound Using DirectSound3D* - G. Graham - *Game Developer*, Janeiro, 1997.

[GUYM99] *And Now for Something Completely Different* – M. Guymon - *Game Developer*, Março, 1999.

[HOOK97] *A Recent History of Interactive 3D Computer Graphics* – B. Hook - *Game Developer*, Julho, 1997.

[KAKER99] Aprenda em 14 dias 3D Studio MAX 2.5 – P. Kakert e D. J. Kalwick – Editora Campus

[KNIG97] *When Motion Capture Beats Keyframing* – D. Knight et al. - *Game Developer*, Setembro, 1997.

[KOLB97] *Win game developer's guide with DirectX 3* – J. Kolb - Waite Group Press, 1997.

[KUMA98] *MBONE Interactive Multimedia on the Internet* – V. Kumar - - New Riders Pub., Indiana, USA, 1996.

[LANDE98] *Working with Motion Capture File Formats* – J. Lander - *Game Developer*, Janeiro, 1998.

[LANDE99] *When Two Hearts Collide* – J. Lander - *Game Developer*, Fevereiro, 1999.

[LAZZA98] *Prototyping 3D Games: Lessons Learned from Riven* – N. Lazzaro - *Game Developer*, Outubro, 1998.

[LEWIN98] *The Role of Story Bibles in Games* – J. S. Lewinsk - *Game Developer*, Fevereiro, 1998.

[MELAX98] *A Simple, Fast, and Effective Polygon Reduction Algorithm* – S. Melax - *Game Developer*, Novembro, 1998.

[MILLER97] *Producing Interactive Audio: Thoughts, Tools, & Techniques* – M. S. Miller - *Game Developer*, Outubro, 1997.

[MORRI96] *Networking your Game using DirectPlay* – M. Morrison - *Game Developer*, Julho, 1996.

[MOTHE94] *Tricks of the Game Programming Gurus* – A. LaMothe, J. Ratcliff, M. Seminatore, D. Tyler - *Sams Publishing*, 1994.

[MUSICA] *www.cdcc.sc.usp.br/ondulatoria/musica1.html*

[PLAY2] www.consoledomain.com/psx/articles/Playstation_2_Its_Official.html

[PRITC95] *Supercharge Your Sprites* - M. Pritchard - *Game Developer*, Abril, 1995.

[RESNIC] *Learning in Virtual Communities - The PetPark* – M. Resnick – documento de projeto em mres.www.media.mit.edu/people/mres/

[RODGE98] *Animating Facial Expressions* – J. Rodgers - *Game Developer*, Novembro, 1998.

[ROSEN96] *Bringing Life to HyperBlade* – S. Rosen, R. Duisberg - *Game Developer*, Setembro, 1996.

[SICKS97a] *Different Perspectives on 3D Sprites* - D. Sicks - *Game Developer*, Maio, 1997.

[SICKS97b] *Dawn of the 3D Pixel Sprite* – D. Sicks - *Game Developer*, Junho, 1997.

[STTED98] The Art of Low Polygon Modeling – P. Steed - *Game Developer*, Junho, 1998.

[THOMP96] *Building a Scene Using Retained Mode Direct3D* – N. Thompson - *Game Developer*, Setembro, 1996.

[VIDEOT] www.videotopia.com - Site com documentação diversa sobre jogos eletrônicos.

[VINCK97] *From LedWars to LMK: Lessons in Structuring Data* – S. Vincke - *Game Developer*, Outubro, 1997.

[WAGG99] *Video in Games: The State of Industry* – B. Waggoner e H. York - *Game Developer*, Março, 1999.

[WHITE95] *Real Time 3D Modeling* – J. White - *Game Developer*, Setembro, 1995.

[WHITE98] *Oldtimer's Guide to Better Textures* - J. White - *Game Developer*, Junho, 1998.

[WRIG96] *OpenGL SuperBible* – R. S. Wright et al. - Waite Group Press, 1996.