METAVERSO

Prof. André Roberto Guerra

INTRODUÇÃO

Até aqui, abordamos mais profundamente os requisitos básicos (a tríade *imersão*, *interação* e *envolvimento*) de acesso ao metaverso, a fim de proporcionar uma compreensão detalhada desse novo universo como um ambiente vitual, versando sobre quais são os elementos para desenvolver uma simulação de qualidade em sistemas de realidade virtual (SRV).

Sequencialmente, esta etapa de estudos sobre metaverso irá trazer detalhes de modelagem e programação, os quais relacionam-se com o processo de desenvolvimento de SRV e, consequentemente, também impactam nas experiências proporcionadas pelo metaverso. Por fim, serão apresetados os principais conceitos e distinções entre realidade virtual (RV) e realidade aumentada (RA).

O desenvolvimento de ambientes virtuais interativos exige predefinição dos elementos virtuais que compõem cada uma das cenas, além da organização desses elementos e a definição de quais deles serão interativos. Nesse sentido, pretende-se destacar os principais modelos de processo propostos atualmente e apresentar o processo de desenvolvimento de SRV, por meio das etapas de análise de requisitos, projeto, implementação, avaliação e implantação.

Nesse contexto, esta etapa tem como objetivos:

a. Apresentar as definições de modelagem e programação em ambientes virtuais, além de destacar o processo de desenvolvimento de SRV;
b. Abordar os principais conceitos e distinções entre realidade virtual (RV) e realidade aumetada (RA);
c. Explicar as vantagens de utilização da RA e suas aplicações.

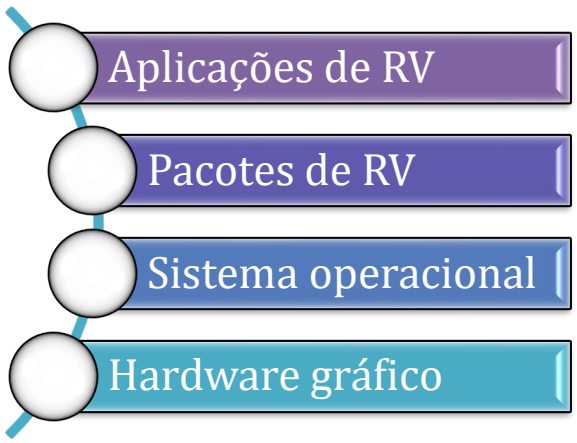
E seguirá a seguinte estrutura:

a. Modelagem e programação em ambientes virtuais;
b. Processo de desenvolvimento de um SRV;
c. Realidade virtual (RV) X Realidade aumentada (RA);
d. Vantagens e aplicabilidade da RA;
e. Conclusão e próximos passos.

TEMA 1 – MODELAGEM E PROGRAMAÇÃO EM AMBIENTES VIRTUAIS

Aplicações gráficas tridimensionais (3D) exigem um esforço computacional considerável para processá-las. É possível distribuir esse esforço entre processadores alocados em placas gráficas ou na placa principal do computador. Conhecer esses detalhes dos modelos arquiteturais disponíveis e desenvolver aplicações que explorem ao máximo a potência computacional das placas e processadores faz com que as aplicações percam portabilidade, isto é, sejam fortemente dependentes do *hardware* para o qual foram desenvolvidas. No caso de a portabilidade da aplicação ser o fator preponderante sobre o desenvolvimento, opta-se por um modelo de desenvolvimento que utilize o conceito de camadas de abstração sobrepostas, conforme ilustra a Figura 1 (Calonego Júnior et al., 2006).

Figura 1 – Camadas de abstração



Fonte: Baseado em Calonego Júnior et al., 2006.

A camada denominada *hardware gráfico* corresponde a algum dispositivo de saída gráfica, por exemplo, uma placa gráfica usada em jogos. Uma vez inserida a placa no computador, a camada *sistema operacional* deve ser configurada. Os fabricantes dos dispositivos gráficos disponibilizam os programas que permitem ao sistema operacional ter acesso ao dispositivo gráfico. O sistema operacional gerencia o acesso ao *hardware* gráfico e oferece um conjunto de primitivas que viabilizam o acesso indireto de outras aplicações ao *hardware* gráfico. Isso significa que o programador não tem necessidade de conhecer como o *hardware* opera. Além disso, há a possibilidade de troca do *hardware* sem que haja a necessidade de modificação dos programas que utilizam as primitivas do sistema operacional (Calonego Júnior et al., 2006).

O desenvolvimento de programas para um dado sistema operacional apresenta pouca portabilidade. O aumento dessa portabilidade é tratado na camada *biblioteca gráfica*. Ela implementa um padrão de comunicação com as primitivas do *hardware*, via sistema operacional, que aumenta essa portabilidade. Os exemplos mais comuns de camadas análogas à camada *biblioteca gráfica* são o *OpenGL* e o *DirectX* (Quadro 1). Programas que usam essas camadas podem ser transportados para diferentes sistemas operacionais, desde que haja uma versão correspondente à camada usada pela aplicação que execute naquele sistema operacional. A proliferação dessas bibliotecas gráficas possibilitou a criação de pacotes de desenvolvimento de RV (Calonego Júnior et al., 2006).

Quadro 1 – Diferenças entre *OpenGL* e *DirectX*

OpenGL®	Microsoft® DirectX®
API de plataforma cruzada para gráficos de renderização	API proprietária desenvolvida pela Microsoft
Desenvolvido pelo *Khronos* Grupo	Desenvolvido pela Microsoft
Código aberto	Proprietário
Disponível em vários sistemas operacionais, incluindo *Windows*	Disponível apenas no sistema operacional Windows
Suporta uma gama mais ampla de plataformas, incluindo dispositivos móveis	Projetado para *desktop* e jogos por console
Não inclui suporte para áudio ou entrada	Inclui suporte para áudio e entrada
Maior flexibilidade e personalização	API mais padronizada e simplificada
Mais complicado e mais difícil para aprender	Mais fácil para aprender e usar, especialmente para iniciantes
Fornece maior controle sobre o gráfico pipeline	O *pipeline* é mais limitado e estruturado no DirectX
Suporta uma gama mais ampla de programação de linguagens	Principalmente projetado para uso com C++
Não fornece suporte embutido para linguagem shader	Inclui suporte integrado para linguagem HLSL shader
Usa um sistema de coordenadas baseado em coordenadas destras	Usa um sistema de coordenadas baseado em coordenadas canhotas
Uso imediato do modo de renderização por padrão	Usos retidos do modo de renderização por padrão
O OpenGL usa um pipeline de função fixa por padrão	O DirectX usa um pipeline programável por padrão
OpenGL requer mudanças de estado mais explícito	DirectX tem os estados de mudanças implícitos

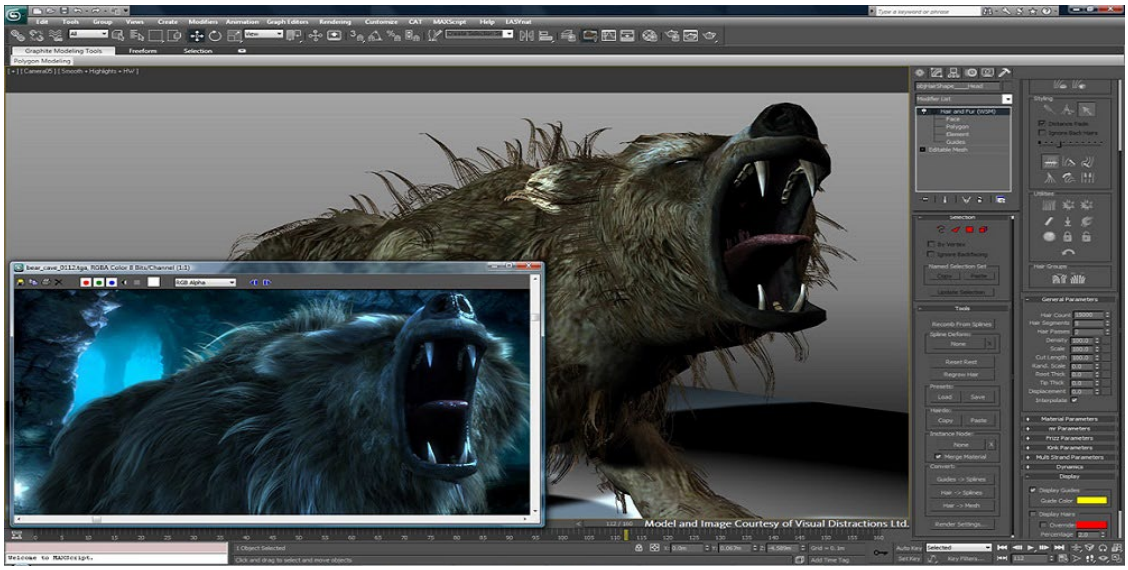
Fonte: Baseado em Monaleesa, 2023.

A biblioteca *OpenGL* permitiu a especificação e o desenvolvimento de uma camada para a descrição de objetos, eliminando a preocupação com a implementação das gráficas primitivas ou com o *hardware* a ser utilizado pela

aplicação. Esse modelo foi utilizado por Gavin Bell para escrever a primeira especificação de uma linguagem de modelagem de mundos virtuais denominada *Virtual Reality Modeling Language* (VRML) (Carey; Bell, 2000).

Diversos *pacotes de RV* utilizam programas escritos na linguagem VRML ou oferecem suporte para a conversão de códigos produzidos em outros formatos para VRML. Por exemplo, programas de desenho 3D tais como o *3D Studio MAX* (Figura 2) e o *Spazz 3D*, exportam código VRML. Esses pacotes são a base para a implementação de aplicações de realidade virtual, que correspondem ao maior grau de abstração. Em geral, as aplicações são desenvolvidas utilizando-se um pacote de desenho 3D, um pacote de RV, e linguagens de programação tais como C, C++, *Java*, *Delphi*, *VB*, *ECMAScript*, entre outras, que permitem ao programador ter acesso ao pacote de RV para desenvolver os controles de interação da aplicação com o usuário (Calonego Júnior et al., 2006).

Figura 2 – Programa *3D Studio MAX*



Fonte: Animação 3D..., S.d.

A programação em VRML, ou a construção de cenários virtuais, é uma tarefa complexa e exige dos gerenciadores o uso de estruturas de dados específicas para a otimização da visualização durante a navegação no ambiente virtual. Um *gerenciador de desenho* simplifica a tarefa de controle fornecendo ao programador uma interface de desenvolvimento de programa, também denominada *Application Programming Interface* (API). O uso de uma API permite ao programador elaborar um cenário virtual para navegação e interação de forma direta. A criação de um mundo virtual é, portanto, a expressão de componentes

do mundo real através de um modelo matemático que permita representar os elementos, bem como as suas interações. Esse modelo é denominado *grafo de cena* e é usado por gerenciadores para otimizar o desenho das cenas 3D. Atualmente, há programas tradutores de código VRML para *Extensible 3D* (X3D), a fim de garantir que as aplicações desenvolvidas em VRML possam ser facilmente transportadas (Calonego Júnior et al., 2006).

TEMA 2 – PROCESSO DE DESENVOLVIMENTO DE UM SRV

A realidade virtual (RV) é uma tecnologia de interface avançada que possibilita ao usuário não somente usar o sistema de *software*, como também perceber-se dentro do ambiente tridimensional gerado por um computador. Nesse contexto, o usuário pode explorar e até mesmo modificar o ambiente virtual, o que lhe é possibilitado por meio de técnicas de navegação, interação e imersão (Vince, 2004).

Os SRV podem ser implementados através de diferentes arquiteturas físicas e lógicas, que compreendem desde a utilização de um único microcomputador até arquiteturas distribuídas de processamento, que permitem, por exemplo, melhor realização de uma imagem digital. Os tipos mais comuns de arquitetura envolvem uma combinação das seguintes características: funcionamento mono ou multiusuário e processamento centralizado ou distribuído (Luz; Kirner, 2006).

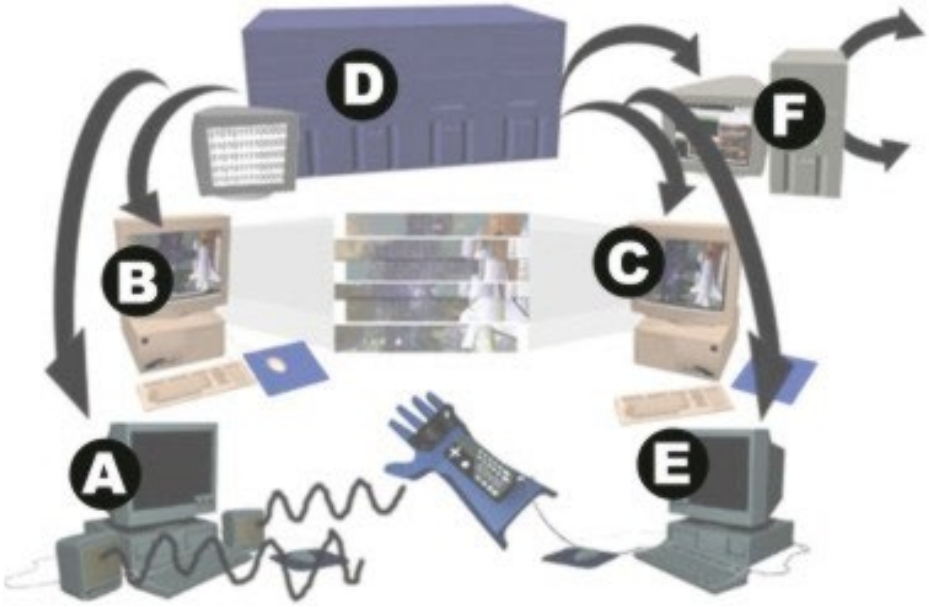
Em linhas gerais, uma arquitetura de SRV é composta de dois conjuntos de componentes (Luz; Kirner, 2006).

a. Interfaces físicas e lógicas, que incluem as entradas e saídas do sistema, representadas por sensores e atuadores, respectivamente. Essas interfaces permitem a integração do ser humano com o sistema;
b. Processador lógico do mundo virtual, sendo esse componente responsável pelo controle do sistema.

A Figura 3 apresenta um exemplo de arquitetura de SRV que envolve processamento distribuído. Nesse tipo de arquitetura, diversos aspectos do mundo virtual são processados por diferentes computadores. O computador A processa os dados referentes à geração do som, enquanto os computadores B e C geram as imagens que, em conjunto, geram a visão estereoscópica que alimenta o capacete de imersão, causando no usuário a sensação de

profundidade. O computador D é responsável pelo processamento computacional das tarefas em tempo real e pela integração do sistema, enviando e recebendo pacotes de tarefas que compõem a interface externa do sistema (tanto com o usuário, quanto com outros sistemas). Já o computador E refere-se ao controle do dispositivo háptico, de retorno da força enviada pelo usuário. O computador F, por sua vez, é responsável pela base de dados do sistema e pela atualização de dados referentes a agentes externos, utilizando, para isso, um meio de comunicação com outras bases de dados distribuídas (Luz; Kirner, 2006).

Figura 3 – Esquema de um SRV de processamento distribuído



Fonte: Luz; Kirner, 2006, p. 111.

Como qualquer sistema de *software*, o desenvolvimento de SRV pode basear-se nos modelos e métodos tradicionalmente indicados pela engenharia de *software*. Os modelos existentes, desde o tradicional Cascata (Sommerville; Sawer, 1997) até o atual Programação Extrema (Beck, 1999), podem ser adotados. Mesmo com a existência de diversos métodos, cada empresa deve utilizar o que melhor lhe convier e, se necessário, adaptá-los ou até mesmo criar o seu próprio processo. Um resumo dos principais modelos de desenvolvimento de *software* está representado no Quadro 2.

Quadro 2 – Principais modelos de desenvolvimento de *software* para SRV

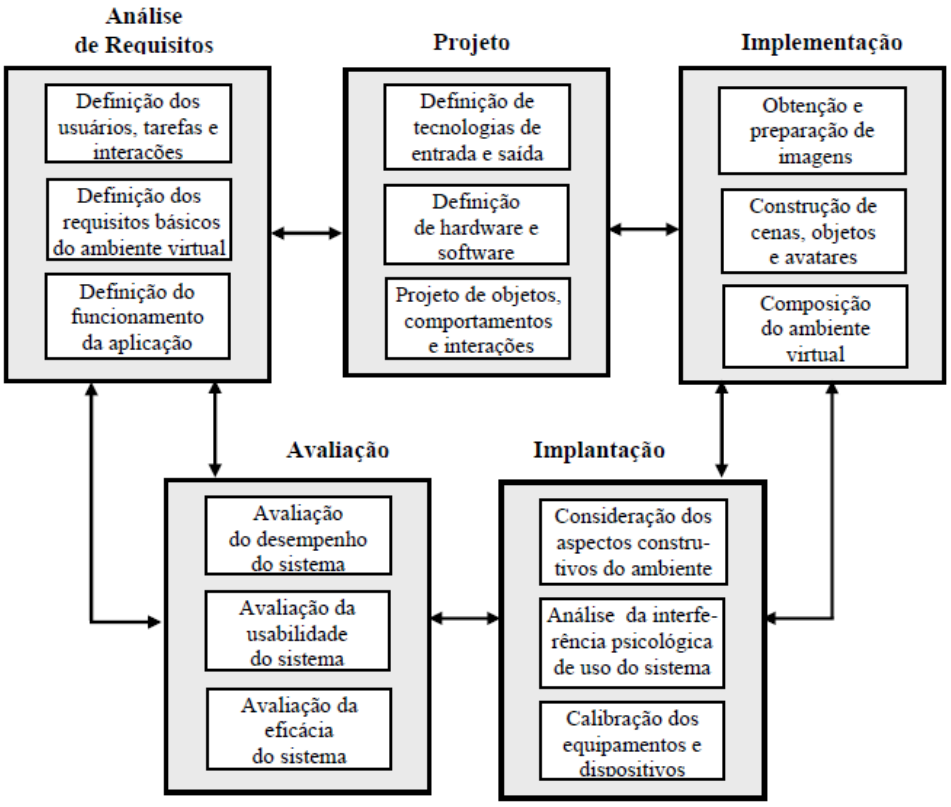
Modelo Cascata (*waterfall*)	É um processo tradicional de desenvolvimento de *software*, que envolve a consecução das etapas de levantamento dos requisitos, análise dos requisitos, projeto, projeto detalhado, implementação, testes, implantação e manutenção. Apesar de ser largamente utilizado, possui desvantagens, pois projetos dessa natureza possuem um ciclo de desenvolvimento longo e, nesse caso, muitas das tecnologias, soluções e até mesmo metáforas podem ter que ser modificadas antes da finalização do ciclo de desenvolvimento.
Prototipagem	Este modelo mostra-se adequado por permitir a criação de um protótipo ou produto final do sistema e colocá-lo à prova junto aos usuários finais, em um tempo relativamente curto. Entretanto, não prevê a reformulação do sistema ao longo do tempo, pois o protótipo é criado apenas uma vez e depois o ciclo de desenvolvimento transcorre linearmente.
Desenvolvimento Iterativo	É um modelo importante, embora exija que as principais funcionalidades do sistema sejam cobertas já no primeiro estágio de desenvolvimento. Em muitos casos, este modelo não leva a uma solução adequada, pois essas funcionalidades podem exigir um tempo e custo de desenvolvimento elevado, além de contribuir para distanciar o cliente do produto final, aumentando os riscos.
Modelo evolucionário	Apresenta características importantes, pois o ciclo de desenvolvimento de cada versão do sistema é reduzido em relação aos modelos anteriores. Neste modelo, o sistema é disponibilizado em versões que cumprem alguns dos requisitos totais do sistema.
Programação extrema	A *Extreme Programming*, com sua abordagem focada no problema e com contato constante com o cliente e usuário final, procura aumentar as chances do sistema ser desenvolvido conforme as reais necessidades do usuário. O modelo estimula a criação rápida de versões, para que o usuário final possa avaliar e interferir no próximo ciclo de desenvolvimento. Este modelo é adequado quando os requisitos não forem totalmente esclarecidos e o contato com o cliente for possível. Para um SRV completo, este modelo pode não ser adequado, principalmente quando for exigida a integração de equipamentos especiais de alto custo. A criação de uma réplica do sistema junto ao cliente e/ou o constante deslocamento do cliente ou dos desenvolvedores para a averiguação de cada nova versão pode também elevar muito os custos.

Fonte: Baseado em Luz; Kirner, 2006, p. 116.

O processo de desenvolvimento de um SRV compõe-se de etapas (Figura 4), realizadas iterativamente, que são:

a. Análise de requisitos;
b. Projeto;
c. Implementação;
d. Avaliação;
e. Implantação.

Figura 4 – Etapas do processo de desenvolvimento de SRV



Fonte: Luz; Kirner, 2006, p. 118.

TEMA 3 – REALIDADE VIRTUAL (RV) X REALIDADE AUMENTADA (RA)

A evolução das Tecnologias de Informação e Comunicação (TICs), incluindo o poder de processamento dos computadores, o barateamento dos dispositivos, a velocidade da comunicação e a disponibilidade de aplicativos gratuitos vem promovendo a consolidação de várias tecnologias, dentre elas a realidade aumentada (RA). A RA enriquece o ambiente físico com objetos sintetizados computacionalmente, permitindo a coexistência de objetos reais e virtuais, podendo ser ser considerada uma vertente da realidade virtual (RV), ainda que inicialmente tenham sido desenvolvidas indistintamente (Hounsell; Tori; Kirner, 2021).

Diferentemente da RV, que transporta o usuário para o ambiente virtual fazendo-o abstrair completamente o ambiente físico e local, a RA mantém referências do entorno real, transportando elementos virtuais para o espaço do usuário. O objetivo é que o usuário possa interagir com o mundo e os elementos virtuais, de maneira mais natural e intuitiva, sem necessidade de treinamento ou adaptação. Essa interação pode ser feita de maneira *direta* (com a mão ou com o

corpo do usuário) ou *indireta* (auxiliada por algum dispositivo de interação). Se vários dispositivos competem para facilitar a interação, a interface é denominada *multimodal*. A possibilidade de usar uma interação natural e, principalmente, as próprias mãos para segurar instrumentos físicos reais ao mesmo tempo em que se pode interagir com informações e modelos virtuais, é um dos maiores benefícios da RA (Hounsell; Tori; Kirner, 2021).

Comparando RA e RV, já foi dito (Billinghurst et al., 2015) que o principal objetivo da RV é usar a tecnologia para substituir a realidade ao passo que o principal objetivo da RA é melhorar a realidade. Dessa forma, um ambiente em RA pode ser representado por uma mesa real e um vaso virtual, como mostrado na Figura 5 (Hounsell; Tori; Kirner, 2021).

Figura 5 – Exemplo de RA com vaso e carro virtuais sobre uma mesa

Fonte: Kirner e Tori, 2006, p. 25.

A RA e a RV pode ter suas diferenças estudadas quando vistas em um diagrama que considera a dimensão da artificialidade e a dimensão do espaço (Figura 6). Ambos os casos tratam de objetos gerados por computador, mas, no mundo físico, a RA está ligada com a realidade física, enquanto a RV refere-se ao sentido de telepresença. Assim, pode-se comparar RA com RV levando-se em conta três fatores principais:

a. A RA enriquece a cena do mundo real com objetos virtuais, enquanto a RV é totalmente gerada por computador;
b. No ambiente de RA, o usuário mantém o sentido de presença no mundo real, enquanto que, na RV, a sensação visual é controlada pelo sistema;
c. A RA precisa de um mecanismo para combinar o real e o virtual, enquanto que a RV precisa de um mecanismo para integrar o usuário ao mundo virtual (Hounsell; Tori; Kirner, 2021).

TEMA 4 – VANTAGENS E APLICABILIDADE DA REALIDADE AUMENTADA (RA)

Algumas das vantagens e aplicabilidades da RA se confundem com as da RV, mas pode-se destacar algumas que são próprias da RA. Dentre as vantagens da RA destacam-se (Wang; Ong; Nee, 2016):

a. Não é necessário fazer toda a modelagem do mundo virtual (o que normalmente demanda esforço manual, aumentando a dificuldade de integração com os sistemas e também esforço computacional para a renderização);
b. O usuário pode agir no real (usar ferramentas, atuar sobre dispositivos, manipular objetos, se mover em torno do objeto) de forma natural com suas propriedades responsivas (hápticas: peso/inércia, textura, rigidez), o que dá maior senso de realismo e imersão no mundo enriquecido, trazendo o benefício tanto do real quanto do virtual;
c. Podem-se explorar novos elementos (virtuais) e sua interação com o ambiente (real), sem a necessidade de construir ou desenvolver os elementos, economizando tempo e recursos;
d. Proporciona um ambiente seguro, flexível, controlado e intuitivo para experimentar interações físicas.

Da mesma maneira que a RV, a RA pode ser aplicada às mais diversas áreas do conhecimento, em muitos casos com vantagens adicionais por se integrar simbioticamente com os ambientes reais. Essas aplicações consistem, por exemplo, em: reparo mecânico; modelagem e projeto de interiores; cirurgia apoiada por computador; manufatura e diagnóstico de placas de circuito impresso; montagem de equipamentos; experimentação de adornos; manutenção de instalações industriais; visualização de instalações embutidas; visualização de temperaturas em máquinas e tubos; ferramentas para educação e treinamento; exposições em museus virtuais; visualização de dados; e muitas outras (Hounsell; Tori; Kirner, 2021).

TEMA 5 – CONCLUSÃO

Com a evolução da tecnologia relacionada à visualização e dispositivos especiais, a RV vem obtendo uma avanço crescente. Cada vez mais, os SRV estão fazendo parte do cotidiano das pessoas, nas mais diferentes áreas de

aplicação. Consequentemente, o domínio de um processo sistemático de desenvolvimento, adaptado às peculiaridades dos SRV, tornou-se um fator altamente relevante para as empresas de *software*.

Acompanhando a tendência do desenvolvimento de jogos para dispositivos móveis, em especial para celulares, aliado ao aumento da capacidade de processamento desses dispositivos (*smartphones*), a RA tende a acompanhar tal evolução, ficando cada vez mais popular. Mas, tudo indica que esse recurso tecnológciso não será somente destinado ao entretenimento, sendo destinado a aplicações mais sérias, como nas áreas da saúde, educação e comercial por exemplo.

Vale destacar que os conceitos de RV e RA ainda são consideravelmente recentes, apresentando limitações. As principais desvantagens da RA estão associadas com a forma com que se promove a integração entre os dispositivos com o processamento e a tarefa em questão. Ou seja, não existem soluções prontas de como abordar uma determinada área. Portanto, muita pesquisa precisa ser feita para analisar as formas mais intuitivas e naturais dessa integração, a fim de permitir experiências cada vez mais qualificadas aos usuários.

REFERÊNCIAS

ANIMAÇÃO 3D Modelagem 3D. **Escola Pro Arte**, S.d. Disponível em: <https://escolaproarte.com.br/animacao-3d-modelagem-3d/>. Acesso em: 31 jul. 2023.

BECK, K. **Extreme programming explained**: embrace change. Boston: Addison Wesley Professional, 1999.

BILLINGHURST, M. et al. A survey of augmented reality. **Foundations and Trends® in Human-Computer Interaction**, v. 8, n. 2-3, p. 73-272, 2015.

CALONEGO JÚNIOR, N. et al. Modelagem e programação de ambientes virtuais interativos. In: TORI, R.; KIRNER, C.; SISCOUTTO, R. (eds.). **Fundamentos e tecnologia de realidade virtual e aumentada**. Porto Alegre: SBC, 2006. p. 98-108.

CAREY, R.; BELL, G. **The Annotated VRML 2.0 Reference Manual**. 3. ed. Boston: Addison-Wesley. 2000.

HOUNSELL, M. S.; TORI, R.; KIRNER, C. Realidade aumentada. In: TORI, R.; HOUNSELL, M. S. **Introdução a realidade e aumentada**. 3. ed. Porto Alegre: SBC, 2021. p. 11-29.

KIRNER, C.; TORI, R. Fundamentos de realidade aumentada. In: TORI, R.; KIRNER, C.; SISCOUTTO, R. (eds.). **Fundamentos e tecnologia de realidade virtual e aumentada**. Porto Alegre: SBC, 2006. p. 22-38.

LUZ; R. P.; KIRNER, T. G. Processo de desenvolvimento de sistemas de realidade virtual. In: TORI, R.; KIRNER, C.; SISCOUTTO, R. (eds.). **Fundamentos e tecnologia de realidade virtual e aumentada**. Porto Alegre: SBC, 2006. p. 109-127.

MONALEESA. Top 50 differences between OpenGL and DirectX – Open GL versus DirectX. **Freshers Now**, 3 fev. 2023. Disponível em: <https://www.freshersnow.com/opengl-vs-directx/>. Acesso em: 31 jul. 2023.

SOMMERVILLE, I.; SAWER, P. **Requirements engineering**: a good practice guide. New York: Wiley, 1997.

VINCE, J. **Introduction to Virtual Reality**. Verlag: Springer-Verlag, 2004.

WANG, X.; ONG, S. K.; NEE, A. Y. C. A comprehensive survey of augmented reality assembly research. **Advances in Manufacturing**, v. 4, n. 1, p. 1-22, 2016.