SigemVR: Um Simulador Imersivo e Interativo de Gerência de Memória

Luiz Felipe dos Santos Freitas

Outubro / 2018

Projeto de pesquisa de Mestrado em Ciência da Computação

SigemVR: Um Simulador Imersivo e Interativo de Gerência de Memória

Esse documento corresponde ao Projeto de Pesquisa apresentado à Banca Examinadora para Defesa de Dissertação no curso de Mestrado em Ciência da Computação do UNIFACCAMP – Centro Universitário Campo Limpo Paulista.

Campo Limpo Paulista, 19 de outubro de 2018.

Luiz Felipe dos Santos Freitas

Prof. Dr. Marcelo de Paiva Guimarães (Orientador)

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Dedicatória

Dedico este trabalho a minha esposa Alessandra Carla Mendes, fonte de amor, compreensão, admiração e paciência. Ao meu filho Victor Hugo que sempre me motivou a querer mais e aos meus avós, Síria e Castor (em memória), que estão sempre presentes nos meus pensamentos e no meu coração.

Agradecimentos

Agradeço ao professor e orientador Dr. Marcelo Paiva Guimarães pelo apoio, confiança e ensinamentos. Sem sua ajuda nada disso seria possível. À toda minha família: minha mãe, meu irmão, meus tios, primos e sobrinhos que sempre me apoiaram e incentivaram durante esta jornada. E um agradecimento especial ao amigo, professor e matemático Mauro Luís, que esteve sempre disponível para ouvir, opinar e ajudar na melhora desse texto.

“O tempo às vezes é alheio às nossas vontades, mas só o que é bom dura tempo o bastante para se tornar inesquecível.”

Charlie Brown Jr

Resumo. *A Realidade Virtual (RV) é uma tecnologia voltada para o desenvolvimento de interfaces avançadas entre usuários e sistemas computacionais, com as quais é possível ver, ouvir, tocar, manipular objetos e percorrer espaços virtuais em tempo real, simulando os mais diversos fenômenos, realísticos ou não. A atual redução de custos e o desenvolvimento de dispositivos com alta capacidade de processamento tornaram essa tecnologia acessível, de modo que aplicações em RV passaram a ser usadas como ferramentas auxiliares de ensino. No ensino dos fundamentos e técnicas envolvidas na construção e no papel desempenhado pelo Sistema Operacional (SO), os professores, de maneira geral, utilizam métodos tradicionais, como projeções, quadro negro e mapas conceituais, os quais não estimulam a habilidade de abstração dos alunos em relação aos conceitos teóricos abordados. Com o intuito de desenvolver uma ferramenta de ensino complementar às aulas teóricas da disciplina de SOs, neste trabalho foi desenvolvido um software que permite a simulação das principais técnicas de gerenciamento da memória principal utilizadas por diferentes SOs. O simulador, nomeado como SigemVR (Simulador de Gerência de Memória Principal com Suporte à Tecnologia de RV), apresenta ao aluno uma experiência tridimensional, imersiva e multissensorial através de representações gráficas, textuais e auditivas. Foram consideradas as técnicas de alocação contígua simples, particionada estática, particionada dinâmica e paginação. Todas as etapas do método de análise e projeto no desenvolvimento do simulador (concepção, implementação e testes) foram realizadas e concluídas. A etapa de testes foi realizada em três grupos de quarenta estudantes do curso de informática do Instituto Federal do Mato Grosso do Sul (IFMS) – Campus Corumbá, nos quais foram aplicados questionários para avaliar a eficácia do simulador como ferramenta de ensino e os recursos tecnológicos durante a experiência de uso do software. Os resultados dos testes estatísticos ANOVA e do Teste de Tukey mostraram que a média de acertos em questões sobre gerência de memória principal foi maior nos grupos que fizeram uso do software. Quanto aos recursos tecnológicos, os testes apontaram resultados positivos quanto à usabilidade e performance do mesmo. Portanto, dentro do espaço amostral considerado nos testes, foi possível concluir que o SigemVR pode ser considerado um recurso didático eficiente e uma ferramenta auxiliar no ensino do módulo de gerência de memória principal na disciplina de SO.*

Palavras-chave: *Sistemas Operacionais, Realidade Virtual, Simulador, Ensino, Aprendizagem, Gerência de memória, Técnicas de Alocação.*

***Abstract**: Virtual Reality is a technology aimed to developing of advanced interfaces between user and computer systems, in which it is possible to see, hear, touch, manipulate objects and walk through virtual spaces in real time, simulating the most diverse phenomena, realistic or not. Current cost saving and the development of high-capacity devices have made this technology accessible and VR applications have been used as auxiliary teaching tools. In teaching the fundamentals and techniques approached in the construction and performance of Operating Systems (OS), teachers, in general, use traditional methods such as projections, blackboard and conceptual maps, which do not stimulate the abstraction ability of the students. In this context, the purpose of this project is to present a tridimensional, immersive and multisensory simulator, called SigemVR, which can support the memory management teaching, an important topic of operating systems discipline. The techniques of simple contiguous allocation, static partitioning, dynamic partitioning and pagination were considered. All stages in the project development (conception, implementation and testing) were completed. The test stage was performed with three groups of forty computer science students of the Instituto Federal do Mato Grosso do Sul (IFMS)- Campus Corumbá. Questionnaires were applied to evaluate the simulator effectiveness as teaching tool and to evaluate the technological resources of the software during the user experience. The results of ANOVA and Tukey's tests showed that the groups which used the SigemVR presented a higher mean number of correct answers to questions about main memory management. The tests results also showed positive results regarding the usability and performance of the simulator. Therefore, within the sample space considered in the tests, it was possible to conclude that the SigemVR simulator can be used as an auxiliary tool in teaching the main memory management module, important subject of the OS discipline.*

***Keywords:** Operating Systems, Virtual Reality, Simulator, Teaching, Learning, Memory Management, Allocation Techniques.*

Sumário

Glossário

2D – Duas dimensões

3D – Três dimensões

ACM – Association for Computing Machinery

CPU – Central Process Unit

HMD – Head-mounted Display

IEEE – Institute of Electrical and Eletronics Engineers

IFMS – Instituto Federal de Mato Grosso do Sul

PDF – Portable Document Format

QMR – Quadrado médio dos resíduos

QMTR – Quadrado médio do tratamento

RAM – Random Access Memory

ROM – Read-Only Memory

RV – Realidade Virtual

SBC – Sociedade Brasileira de Computação

SigemVR – Simulador de Gerência de Memória em Realidade Virtual

SO – Sistema Operacional

SOsim – Sistemas Operacionais Simulador

SSOG – Simulador de Sistema Operacional Genérico

TADS – Tecnólogo em Análise e Desenvolvimento de Sistemas

TBC-SO/Web: Treinamento Baseado em Computador para Sistemas Operacionais via Web

USP – Universidade de São Paulo

Lista de Tabelas

Lista de Figuras

1. INTRODUÇÃO

De acordo com o Projeto Educom (Nascimento, 2009), que foi desenvolvido por uma comissão de especialistas em educação e tecnologia do Ministério da Educação (MEC) com intuito de criar centros-pilotos para uso de ambientes computacionais em escolas públicas, a informática educativa deu seus primeiros passos em nosso país no ano de 1971, em uma discussão sobre o uso de computadores no ensino de Física, na Universidade de São Paulo (USP) - São Carlos. Desde então, surgiram outras iniciativas para a inclusão do computador no ensino, cada uma delas com sua sustentação teórica e linha de ação (Nascimento, 2009). Com o uso do computador é possível que, professor e escola, estimulem o processo de ensino-aprendizagem com recursos capazes de, por exemplo, ilustrar situações difíceis de serem reproduzidas em sala de aula devido ao custo e/ou segurança dos alunos, como no caso dos simuladores na área de Química. A escola reconhece a influência da informática na sociedade moderna e os seus benefícios na educação, mas aponta que existem inúmeras lacunas a serem preenchidas, como a escolha dos recursos tecnológicos que melhor se adeque à necessidade do docente e o conhecimento necessário para utilização desses recursos (Tajra, 2000)

Os cursos de computação, sejam de nível técnico ou nível superior, possuem disciplinas que envolvem conceitos abstratos, difíceis de serem repassados pelos professores e assimilados pelos alunos, como os relacionados à disciplina de Sistemas Operacionais (SOs), que faz parte do núcleo de fundamentos da Computação (Zorzo, Nunes, Matos, Steinmacher, Leite, Araújo, Correia e Martins, 2017). Assim, o emprego de soluções computacionais com suporte à tecnologia de Realidade Virtual (RV) pode ser utilizado como ferramenta auxiliar na apresentação dos conceitos teóricos, de forma prática e interativa, desenvolvendo a habilidade de abstração dos alunos. Segundo Cardoso (2007), aplicações em RV possibilitam desenvolver uma nova metodologia de exploração, observação e construção do conteúdo estudado.

Atualmente, a redução de custos e os avanços tecnológicos dos dispositivos de *hardware* e das ferramentas de desenvolvimento de aplicações em RV tornaram esta tecnologia acessível. Segundo Burdea (1996), a RV permite construir interfaces computacionais avançadas, compostas por simulações em tempo real e interações com os mais diversos componentes envolvidos na experiência, as quais podem ser usadas como ferramentas de ensino. A utilização de um simulador desenvolvido com base nessa tecnologia permite que computador e mente

humana sejam integrados em uma interface avançada, a qual oferece aos usuários uma experiência imersiva, interativa e exploratória (Machado, 1995 e Reis, 2009).

Diante das inúmeras vantagens e variedade de aplicações de simuladores que exploram os recursos da tecnologia da RV, neste trabalho foi desenvolvido um *software* que permite a simulação das principais técnicas de gerenciamento da memória principal utilizadas por diferentes SOs. O simulador 3D, nomeado como *SigemVR* (Simulador de Gerência de Memória Principal com Suporte à Tecnologia de RV), pode ser usado como ferramenta de ensino auxiliar à disciplina de Sistema Operacional (SO) e apresenta ao aluno uma nova abordagem de ensino, com uma experiência imersiva e interativa através de representações gráficas, textuais e auditivas.

1.1 Contexto e Motivação

Um sistema computacional é composto por um conjunto de componentes eletrônicos e físicos, denominados *hardware*, juntamente com um grupo lógico de programas, os quais são chamados de *softwares*. As memórias primária e secundária, o processador e a fonte de alimentação são exemplos de dispositivos de *hardware* considerados como essenciais para o funcionamento do computador. Já o SO figura como um dos principais elementos de *software*, pois é responsável pelo gerenciamento do computador. Compreender os fundamentos e as técnicas usadas na construção de um SO possibilita o entendimento de conceitos fundamentais das mais diversas áreas da Computação, como: desenvolvimento de aplicações, otimização de desempenho e segurança computacional.

O papel do SO se estende desde a abstração e o entendimento do *hardware,* até o controle, compartilhamento e performance dos recursos oferecidos pelo mesmo (Silberschatz, 2015), além de servir de plataforma para desenvolvimento de outros *softwares* (Machado e Maia, 2013). O SO pode ser dividido em módulos internos responsáveis por funções distintas, mas que são complementares e dependentes entre si, dentre os quais: o módulo de gerência de processador, de gerência de memória principal, de gerência de arquivos e de gerência de dispositivos de entrada e saída. Neste trabalho foi dada ênfase ao módulo de gerência da memória principal, por se tratar de um recurso primordial para execução de qualquer programa, incluindo o SO. As funções do SO podem ser classificadas através de níveis, onde a

comunicação direta com o *hardware* constitui o mais baixo nível, enquanto que o nível mais alto constitui a plataforma de execução para as demais aplicações do sistema computacional (Machado e Maia, 2013). Quando tais conceitos são apresentados aos alunos na disciplina de SO, espera-se que os mesmos compreendam o funcionamento do computador nos seus diversos níveis.

Há basicamente duas formas de abordagem prática no ensino do seu conteúdo: através da utilização de SOs reais ou de simuladores genéricos. O uso de sistemas reais para o ensino da disciplina pode ser explorado e analisado, como por exemplo o MINIX[1] e TROPIX[2], que permite interação direta com o código fonte do sistema. O código fonte aberto (*opensource*[3]) possibilita que estudantes e professores façam alterações nos módulos do SO por meio de uma interface com recursos gráficos limitados, exigindo conhecimentos sólidos e profundos sobre a linguagem de programação, fator que dificulta o entendimento dos conceitos abordados. Em contrapartida, simuladores genéricos, apesar de não serem sistemas reais, apresentam um enfoque muito mais didático, pois traduzem os conceitos da disciplina de forma visual e animada. São feitas analogias através de cenários e representações gráficas dos módulos do SO.

No ensino dos fundamentos e técnicas envolvidas na construção e no papel desempenhado pelo SO, os professores desta disciplina, de maneira geral, utilizam métodos tradicionais, como: projeções, quadro negro e mapas conceituais (Scamati, 2017), que muitas vezes são limitados em apresentar conceitos abstratos aos alunos e difíceis de serem assimilados. Como ferramenta de ensino complementar às aulas teóricas, neste trabalho foi desenvolvido um simulador, baseado na tecnologia de RV, nomeado de *SigemVR*, o qual oferece uma experiência imersiva e interativa sobre as principais técnicas de gerenciamento de memória principal utilizadas por diferentes SOs.

1.2 Objetivos

A RV tem um potencial enorme de enriquecer o ensino por meio de simulações e experiências virtuais. Resultados da utilização desta tecnologia no aprendizado de conceitos científicos, na grande maioria abstratos, sugerem que a imersão em ambientes virtuais pode

[1] *Minix* é um SO *Unix-Like*, escrito em linguagem C e Assembly.

[2] *Tropix* é um SO Unix, desenvolvido pelo Núcleo de Computação da Universidade Federal do Rio de Janeiro.

[3] *Opensource* é o *software* de computador com código-fonte disponibilizado e licenciado ao qual o desenvolvedor possui direito de estudar, modificar e distribuir.

facilitar a compreensão destes conceitos, antes limitada à capacidade de abstração dos alunos e à apresentação teórica dos mesmos (Pantelidis e Vinciguerra, 2008). Neste sentido, o objetivo deste trabalho é avaliar a eficiência de um recurso didático interativo e 3D como ferramenta complementar às aulas teóricas da disciplina de SO, com foco na apresentação das principais técnicas de gerenciamento de memória principal utilizadas por diferentes SOs, como o *Microsoft MS-DOS* e *Multics*. Tal recurso consiste em um *software,* denominado *SigemVR*, desenvolvido com base na tecnologia de RV e que apresenta as seguintes características básicas:

- Proporciona, tanto aos professores quanto aos estudantes da disciplina de SO, uma experiência imersiva, interativa e em tempo real com a simulação.
- Dentre as principais técnicas de gerência de memória principal utilizadas por SOs foram consideradas as técnicas de: Alocação Contígua Simples, Alocação Particionada Estática, Alocação Particionada Dinâmica e Alocação por Paginação (Tanembaum, 2016).
- Possui interface gráfica representativa da memória principal, a qual facilita a compreensão da distribuição e controle dos processos na mesma.
- Permite interações com o ambiente através de reações gráficas e orientações audiovisuais, as quais facilitam a experiência e o entendimento dos fenômenos considerados no simulador.

Portanto, o simulador, que explora os recursos da interface de RV, tem como propósito principal simplificar o entendimento do módulo de gerência de memória principal, com enfoque nas principais técnicas utilizadas na construção de SOs modernos.

1.3 Organização do Trabalho

O presente trabalho é organizado da seguinte forma:

O capítulo 2 apresenta, em cada um dos seus tópicos, os principais conceitos abordados na disciplina de SO, como: breve histórico, classificação, conceito de interrupção e processos, além das teorias necessárias para compreensão das técnicas de alocação de memória principal utilizadas por SOs, assim como sua evolução temporal.

O capítulo 3 apresenta uma revisão bibliográfica dos principais conceitos, características e aplicações da tecnologia de RV.

O capítulo 4 faz um estudo comparativo entre os simuladores já existentes no mercado, bem como as características e funções apresentadas pelos mesmos, encerrando com uma tabela comparativa entre os *softwares* analisados.

O capítulo 5 é dedicado à metodologia adotada no desenvolvimento deste projeto, enquanto que o capítulo 6 apresenta as etapas de desenvolvimento do simulador, desde sua concepção até a sua avaliação.

O capítulo 7 apresenta os resultados obtidos a partir do uso do simulador como ferramenta de auxílio ao ensino da disciplina de SO, mais especificamente no módulo de gerência da memória principal. O capítulo 8 apresenta as conclusões e as sugestões de trabalhos futuros.

Os seguintes documentos se encontram em anexo: o Questionário de análise para o desenvolvimento do simulador SigemVR (Apêndice I), o Questionário teste de conhecimentos em SO (Apêndice II) e o Questionário de teste dos recursos tecnológicos do SigemVR (Apêndice III).

2. SISTEMAS OPERACIONAIS: CONCEITOS IMPORTANTES

Esse capítulo apresenta um referencial teórico dos principais conceitos e definições aplicados na disciplina de SO e que são utilizados no desenvolvimento do SigemVR.

A seção 2.1 é dedicada a fornecer uma visão geral sobre os SOs, definindo conteúdos básicos e comuns à maioria deles, os quais são abordados frequentemente no ensino dos mesmos. A seção 2.2 apresenta um breve histórico no desenvolvimento dos SOs. Na seção 2.3 e 2.4 são apresentados aspectos importantes na compreensão de como os SOs definem a execução de um processo e suas principais características. A seção 2.5 aborda o gerenciamento de memória principal e seus conceitos fundamentais. A seção 2.6 trata da evolução temporal das principais técnicas de alocação de memória usadas pelos SOs.

2.1 Conceituação de SOs

Um modelo computacional moderno é um sistema complexo, formado por um ou mais processadores, memória principal, memória secundária, teclado, mouse e outros dispositivos de entrada e saída. Esse conjunto de componentes eletrônicos e físicos corresponde ao *hardware* do computador. Entender como tais componentes funcionam em detalhes e gerenciar o uso dos mesmos, de maneira eficiente e segura, não é um trabalho simples. Para desempenhar tal papel, é necessário um dispositivo de *software*, chamado de Sistema Operacional (Tanenbaum, 2016). Tal dispositivo não permite apenas a abstração e simplificação do funcionamento do *hardware* do computador, mas, segundo Silberschatz (2015), um SO também estabelece uma relação com o usuário do computador, fornecendo um ambiente que permite executar os programas do mesmo, deixando o sistema computacional como um todo mais conveniente e adequado às necessidades do usuário. Para simplificar essa definição, o autor descreve um sistema computacional dividido em quatro componentes: o *hardware*, o SO, os programas aplicativos e o usuário.

Não existe uma definição universalmente aceita do que faz ou não parte do SO, uma vez que os requisitos básicos, as características e as aplicações são específicas de cada SO e sofreram modificações ao longo do tempo, como será descrito no próximo tópico.

2.2 Histórico

A evolução dos SOs está, em grande parte, associada ao desenvolvimento dos computadores como um todo. Segundo Machado e Maia (2013) esta evolução pode ser dividida ao longo das últimas décadas, como mostrado abaixo:

- Década de 30: O matemático, Alan Turing, desenvolveu a máquina de Turing, capaz de executar sequências de instruções, as quais são atualmente chamadas de algoritmos. Essa máquina criou o conceito de processamento de símbolos, que é a base da ciência da computação moderna.
- Década de 40: Em decorrência da explosão da segunda guerra mundial, houve a necessidade de se criar máquinas que pudessem agilizar os processos manuais realizados para fins militares, fato que impulsionou a criação dos primeiros computadores eletromecânicos. O ENIAC (*Eletronic Numerical Integrator And Calculator*) é considerado o primeiro computador eletrônico digital. Com estrutura de 17 mil válvulas, 10 mil capacitores, 70 mil resistores e massa total igual a 30 toneladas, a sua utilização era baseada em "linguagem de máquina" (*assembly*), cuja programação era feita por meio de painéis e necessitava de longos períodos (dias) para a realização de tarefas. Este tipo de programação é fortemente dependente das características do *hardware*. Nesta época, os computadores não possuíam dispositivos que estabeleciam relação com os usuários, como teclados e monitores, de modo que o conceito de SO ainda não existia.
- Década de 50: O desenvolvimento do transistor[4] e da memória magnética possibilitou um grande avanço na área da Ciência da Computação da época. O transistor permitiu um aumento da performance e confiabilidade no processamento de dados, enquanto que a memória magnética permitiu o armazenamento de uma maior quantidade de informações. Em 1951, os programas, até então conhecidos como *Jobs,* eram codificados em cartões perfurados, os quais eram submetidos, um a um, a uma leitora de dados em uma fita magnética (fita de entrada). Essa fita era processada pelo computador, que executava uma tarefa por vez, armazenando o resultado desse processamento em uma fita de saída, a qual era lida e impressa. Este tipo de processamento foi chamado de *batch*. Em 1953, usuários do computador IBM[5] 701, criaram o primeiro SO, chamado de Monitor, o qual era usado

[4] O transistor é um componente semicondutor utilizado como amplificador ou interruptor de sinais ou energia elétrica.

[5] IBM (*International Business Machines*) é uma multinacional americana voltada para área de tecnologia.

para automatizar tarefas manuais. Nessa mesma década foram criadas as primeiras linguagens de programação de alto nível, como o COBOL[6] e o FORTRAN[7]. No final dos anos 50, a Universidade de *Manchester*, na Inglaterra, desenvolveu o SO Atlas. Nesse SO, a memória era organizada de forma hierárquica, base do conceito de memória virtual usada na maioria dos SOs modernos, como será explicado com maiores detalhes na Seção 2.5.

- Década de 60: Com o surgimento dos circuitos integrados, o custo e o tamanho físico dos computadores reduziram, enquanto o poder de processamento aumentou consideravelmente. Outras inovações dessa época são utilizadas até os dias atuais, como a multiprogramação, o multiprocessamento, *time-sharing* e a memória virtual, tópicos que serão descritos com maiores detalhes na Seção 2.3. Nessa época foram criados os primeiros SO desenvolvidos com linguagem de programação de alto nível, como o *Master Control Program* (MCP), com características que englobam a multiprogramação, memória virtual com segmentação e multiprocessamento assimétrico. Para melhorar a interação entre usuário e computador, novos dispositivos de entrada e saída foram introduzidos, como o teclado e o terminal de vídeo. Além disso, a divisão do tempo de processamento dos programas em pequenos intervalos, chamada de *time-sharing* ou tempo compartilhado, ofereceu aos usuários da época um menor tempo de resposta às suas atividades.
- Década de 70: O desenvolvimento de novas tecnologias em circuitos integrados conhecidas como *Large Scale Integration* (LSI) e *Very Large Scale Integrations* (VLSI), estimulou ainda mais a miniaturização e o barateamento dos equipamentos de *hardware*. Novas arquiteturas, com diversos processadores, permitiram acelerar o desempenho dos computadores, exigindo dos SOs novos mecanismos de controle e sincronismo. O multiprocessamento permitiu que mais de um programa pudesse ser executado simultaneamente. Em 1975 surgiu a linguagem de programação C, usada, até hoje, nos principais SOs.
- Década de 80: A IBM introduziu o *Personal Computer* (PC), popularizando a utilização de computadores no ambiente doméstico. Embora monousuário, uma única estação permitia a execução de diversas tarefas concorrentemente. Nesta época surgiram os primeiros SOs

[6] COBOL (*Common Business Oriented Language*) é uma linguagem de programação orientada para o processamento de banco de dados comerciais, desenvolvida em 1959.

[7] FORTRAN (*Mathematical Formula Translation System*) é uma linguagem de programação desenvolvida pela IBM, a partir de 1950 e usada até hoje.

com interface gráfica (GUI[8]), como o *Microsoft Windows* e OS/2, facilitando a comunicação e a interação do usuário com o computador. A popularização da utilização de redes LAN[9] trouxe a necessidade de SOs específicos, como o *Novell Netware* e o *Microsoft LAN Manager*.

- Década de 90: Na microeletrônica, os circuitos integrados rapidamente evoluíram com a tecnologia conhecida como *Ultra Large Scale Integration* (ULSI), estimulando ainda mais a minuaturização, o barateamento dos dispositivos de hardware e o aumento da velocidade de processamento da informação. Nessa mesma época, o crescimento da Internet exigiu a criação de um protocolo padrão de comunicação, o *Transmission Control Protocol* em conjunto com o *Internet Protocol* (TCP/IP). As interfaces gráficas também foram consolidadas, bem como o amadurecimento e a popularização dos *softwares* de código aberto, como: bancos de dados, servidores *web* e servidores de correio eletrônico.
- Anos 2000 até a atualidade: A crescente redução dos custos, tanto do *hardware* quanto do *software*, facilitou o desenvolvimento de aplicações que necessitam de mais recursos gráficos e memória. Os sistemas computacionais passaram a possuir interfaces gráficas mais sofisticadas, arquiteturas paralelas, alto poder de processamento e armazenamento. A maioria dos SOs atuais são voltados para suportar um elevado grau de interação com os usuários e são voltados para diversas plataformas, como *smartphones* e *tablets*. Além disso, tais SOs passaram a incorporar mecanismos automáticos de detecção e recuperação de erros.

2.3 Classificação dos SOs

De acordo com Machado e Maia (2013), os SOs podem ser classificados como: monoprogramáveis, multiprogramáveis e sistemas com múltiplos processadores, de modo que os dois últimos recebem as subclassificações mostradas na Figura 1.

[8] GUI (*Graphical User Interface*) é o tipo de interface que permite a interação entre usuário e computador a partir de elementos gráficos, como ícones.

[9] LAN (*Local Area Network*) é uma rede de computadores local, que consiste na interconexão de equipamentos em um mesmo espaço geográfico.

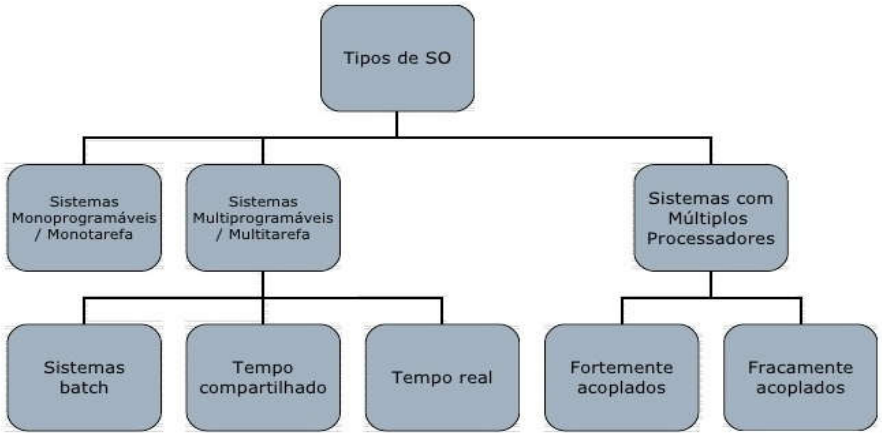


Figura 1: Tipos de Sistemas Operacionais (Fonte: Adaptada de Machado e Maia, 2013)

Os sistemas monoprogramáveis caracterizam-se por permitir a execução de apenas um programa por vez, ou seja, qualquer outra aplicação deve aguardar o término do programa atual para ser executada. Dessa forma, os recursos computacionais, como processador, memória e periféricos, ficam dedicados à execução do programa corrente. Embora seja um modelo simples de ser implementado, quando comparado a outros tipos de SOs, os recursos disponíveis são utilizados sem o intuito de obter a máxima eficiência dos mesmos. Por exemplo, quando o programa corrente aguarda por um evento qualquer, o processador fica ocioso, sem realizar qualquer tipo de processamento, impedindo que outra aplicação seja executada. Há também casos em que há um desperdício de memória, principalmente quando a memória requerida pelo programa corrente é menor do que a quantidade disponível (Silberschatz, 2015).

A necessidade de um SO que utilizasse os recursos computacionais disponíveis de maneira mais eficiente resultou na criação dos sistemas multiprogramáveis. Ao contrário dos monoprogramáveis, nos sistemas multiprogramáveis os recursos são compartilhados entre os diversos usuários e programas. Tais SOs permitem que, enquanto um determinado processo aguarda um evento qualquer, outros processos sejam executados de forma concorrente. A principal vantagem é a redução de custos, já que diversas aplicações compartilham recursos antes dedicados a um único programa. A desvantagem é a complexidade para garantir acesso concorrente e seguro aos diversos recursos compartilhados. É o que ocorre, por exemplo, no compartilhamento de periféricos de entrada e saída, como impressoras e discos, onde o SO deve

garantir acesso seguro e organizado, sem que um usuário ou aplicação interfiram no uso do recurso por outros.

Como mostrado na Figura 1, os SOs multiprogramáveis são classificados como: sistemas *batch*, sistemas de tempo compartilhado e sistemas de tempo real (Machado e Maia, 2013). Essa classificação é decorrente do processo de gerenciamento das informações:

- Sistemas *batch*: Nestes sistemas não existe interação do usuário com a aplicação, uma vez que os programas são organizados em filas e executados sequencialmente. Todas as entradas e saídas da aplicação são implementadas por algum tipo de memória secundária. Esse tipo de sistema é comumente utilizado em tarefas envolvendo cálculos numéricos, compilações, ordenações e *backups* (Machado e Maia, 2013) e foi utilizado nos primeiros sistemas multiprogramáveis.
- Sistema de tempo compartilhado (*time-sharing*): Esse modelo de sistema é baseado na divisão do tempo de uso do processador em intervalos de tempo, conhecidos como *time-slice*. Dessa forma, mesmo que um programa necessite de mais de um intervalo de tempo para atingir seu objetivo, no término do intervalo atual ele é interrompido e substituído por outro programa, enquanto aguarda por um novo intervalo. Para que o processo de troca entre programas em execução seja transparente ao usuário, o SO cria um ambiente de trabalho próprio para cada um deles (Machado e Maia, 2013). É importante considerar que, quando o SO realiza a troca de um programa residente no processador por outro, é necessário garantir que o programa removido, ao retornar, continue o processamento exatamente a partir da instrução em que parou, sem que o usuário perceba esse processo.
- Sistema de tempo real: Ao contrário do que acontece com os sistemas de tempo compartilhado, neste tipo de implementação, o programa corrente utiliza o processador o tempo que for necessário para concluir seu objetivo, ou até que outro programa com maior prioridade solicite sua retirada do processador. A prioridade dada a um programa não é responsabilidade do SO, mas sim do desenvolvedor da aplicação. Nesse modelo de sistema, limites rígidos são impostos ao tempo de processamento de um programa, de modo que, caso não sejam respeitados, esses processos podem apresentar erros irreparáveis. Esse tipo de sistema está presente em aplicações onde o tempo de processamento é fundamental para aplicação, como controle de tráfego aéreo e usinas nucleares (Machado e Maia, 2013).

Já os sistemas com múltiplos processadores apresentam dois ou mais processadores interligados, os quais trabalham em conjunto na execução de tarefas independentes ou no processamento simultâneo de uma mesma tarefa (Silberschatz, 2015). Segundo Machado e Maia (2013), este tipo de sistema deve respeitar os princípios básicos da multiprogramação, como: escalabilidade, disponibilidade e balanceamento de carga. Além disso, esses sistemas também compartilham a memória física (*shared memory*) e os dispositivos de entrada e saída.

2.3.1 Interrupções e exceções

O mecanismo fundamental para o correto funcionamento dos sistemas multiprogramáveis é chamado de interrupção, o qual permite o compartilhamento dos recursos do sistema computacional entre os diversos programas residentes. É através da interrupção que o SO sincroniza a execução de todas as suas rotinas, assim como as rotinas dos programas dos usuários, e também controla o uso de dispositivos de entrada e saída.

Durante a execução de um programa podem ocorrer eventos inesperados, os quais resultam em um desvio do seu fluxo normal de execução. Para controlar a ocorrência desses eventos, a unidade de controle do processador, ao final de cada instrução, verifica a ocorrência de algum tipo de interrupção. Caso algum programa tenha sido interrompido, um conjunto de informações sobre sua execução é salvo, de modo que, o programa pode, posteriormente, voltar a executar, a partir do ponto em que ele foi interrompido.

A Tabela 1 descreve as etapas realizadas pelo mecanismo de interrupção, onde é possível observar as tarefas via *hardware* e via *software* (Machado e Maia, 2013). É importante considerar que para cada tipo de interrupção existe uma rotina de tratamento associada.

Tabela 1: Etapas do mecanismo de Interrupção (Fonte: Machado e Maia, 2013)

Vias:	Etapas:
Via *Hardware*	1. Um sinal de interrupção é gerado pelo processador.
	2. Após o término da execução corrente, o pedido de interrupção é identificado pelo processador.
	3. Os conteúdos dos registradores PC (*Program Counter*[10]) e de status são salvos.
	4. O processador identifica a rotina de tratamento para aquela interrupção e carrega o PC com o endereço inicial dessa rotina.
Via *Software*	5. A rotina de tratamento salva o conteúdo dos demais registradores do processador na pilha de controle do programa.
	6. A rotina de tratamento é executada.
	7. Os registradores de uso geral, de *status* e o PC são restaurados, retomando à execução do programa interrompido.

Já as exceções, mecanismo também fundamental em sistemas multiprogramáveis, ao contrário do que acontece com as interrupções, são geradas por eventos previsíveis, resultantes da execução de uma instrução no próprio programa, definida pelo programador. De maneira semelhante às interrupções, sempre que uma exceção é gerada, a execução do programa é interrompida e o controle é desviado para uma rotina de tratamento da exceção, a qual permite tratar esse novo evento.

2.3.2 Modos de ativação e desativação do SO

Quando um computador é inicializado, alguns procedimentos são realizados antes da inicialização do SO. O processo de ativação do sistema computacional é inicializado pelo *software* chamado *boot loader*, que fica armazenado na memória ROM (*Read-Only*

[10] *Program Counter* (PC) é um registrador da CPU que indica qual é a posição atual na sequência de execução de um processo.

Memory)[11] da máquina. Em seguida, um programa, chamado *Power-On Self Test* (POST), o qual é responsável por verificar possíveis problemas no *hardware* do computador, é executado. Caso não sejam encontrados problemas, a próxima etapa consiste na verificação da presença de algum dispositivo de armazenamento secundário, no qual pode ser encontrado algum SO disponível para a inicialização. Caso este SO seja encontrado, um bloco específico do seu conteúdo, conhecido como *boot sector* (setor de boot), é carregado na memória principal, permitindo que os arquivos de inicialização do SO sejam executados, especificando procedimentos de customização e configuração do ambiente computacional (Machado e Maia, 2013).

A desativação do sistema, também conhecida como *shutdown*, finaliza ordenadamente as aplicações e os componentes do SO, garantindo a sua integridade. No entanto, alguns eventos externos podem fazer com que o SO seja finalizado inesperadamente, como a falta de energia, por exemplo.

2.3.3 Estrutura de um SO

Segundo Silberschatz (2015), um SO pode ser dividido em módulos, com características e funções definidas e cuidadosamente delimitadas. Exemplos destes módulos são:

- Gerência de processos: É o módulo do SO responsável pelo compartilhamento e gerenciamento do processador, desempenhando funções, como: criar e excluir programas, suspender e retomar a execução de programas, permitir sincronização e comunicação entre os diversos programas e *deadlocks*[12] (Silberschatz, 2015). Os principais conceitos e características de um processo serão descritos com maiores detalhes na Seção 2.4;
- Gerência de arquivos: É o módulo do SO que define os métodos de armazenamento de informações nos mais diversos tipos de dispositivos de armazenamento secundário, como fitas magnéticas, discos magnéticos e ópticos. Cada um desses dispositivos possui tecnologia e características próprias. Também é responsável por funções, como: criar e

[11] ROM é um tipo de memória que permite apenas leitura, ou seja, as suas informações são gravadas pelo fabricante um única vez.

[12] *Deadlocks* no contexto de sistemas operacionais refere-se a uma situação em que, dois ou mais processos, ficam impedidos de executar suas funções em razão do bloqueio de recursos compartilhados com os demais.

excluir arquivos, permitir a manipulação de arquivos e diretórios, mapear arquivos nos dispositivos de armazenamento secundário, oferecer funções de *backup*[13] e restauração de arquivos;

- Gerência de dispositivos de entrada e saída: Também chamada de gerência de I/O (*input/output*), esse módulo do SO é responsável por funções, como: permitir a comunicação com os dispositivos de I/O, simplificar o funcionamento desses componentes, apresentar uma interface geral de utilização do dispositivo, entre outras.

O módulo de gerência de memória principal será tratado com maiores detalhes na Seção 2.5, uma vez que é fundamental para este trabalho. No entanto, para compreender o gerenciamento da memória principal, a definição de um processo é um requisito básico e é apresentada na próxima seção.

2.4 Processos

Segundo Tanenbaum (2016), um processo é uma abstração de um programa em execução e deve ser considerado prioritário por qualquer SO. Processos não podem ser confundidos com programas. Segundo Silberschatz (2015), um programa é uma sequência de instruções codificadas, pelo usuário, usando uma linguagem de programação, com o objetivo de solucionar um problema específico. Já o processo, define a execução das instruções do programa. Portanto, para ser considerado um processo é necessário que o programa seja executado, alocando recursos de *hardware* e *software* do ambiente computacional em que se encontra. A estrutura de um processo é dividida em três partes: o contexto de *hardware*, o contexto de *software* e o espaço de endereçamento.

O contexto de *hardware* é responsável por armazenar as informações do fluxo de execução de um processo, como o conteúdo dos registradores gerais, dos registradores de uso específicos e do *Program Counter* (PC) do processador. São essas informações que permitem que um processo, ao ser interrompido, possa ser retomado a partir do ponto em que foi retirado de execução (Machado e Maia, 2013).

[13] *Backup* em informática caracteriza uma cópia de segurança de dados de um dispositivo de armazenamento para outro, permitindo a sua restauração caso necessário.

O contexto de *software* armazena as informações obtidas por um arquivo do SO, chamado de "arquivos de usuários", as quais podem ser divididas em três grupos: identificação, quotas e privilégios (Machado e Maia, 2013), cujas características são descritas a seguir:

- Identificação: Como o nome sugere, armazena os dados que identificam e diferenciam um processo de outro. Sempre que criado, um processo recebe uma identificação única, representada por números, chamada de *Process Identification* (PID). O usuário responsável pela criação do processo também recebe uma identificação única, denominada *User Identification* (UID);
- Quotas: Grupo em que são armazenadas as informações a respeito da capacidade máxima de recursos (limites) do sistema computacional que podem ser utilizados pelo processo, como, por exemplo, a capacidade máxima na memória principal que um processo pode utilizar. Segundo Tanenbaum (2016), limites mal definidos podem acarretar em problemas no funcionamento de um processo;
- Privilégios: define os direitos de ações de um processo com relação a ele mesmo, os demais processos e o SO. Os privilégios de um processo devem garantir que esse não ocasione problemas em outros processos e no próprio SO;

Todo processo ocupa uma área na memória principal do computador, onde ficam alocadas as instruções e os dados do programa, chamada de espaço de endereçamento. Essa área não pode ser ocupada ou sofrer interferência de outros processos. A distribuição dos processos na memória principal será assunto da Seção 2.5.1.

2.4.1 *Process Control Block* (PCB) / Bloco de Controle do Processo (BCP)

O bloco de controle do processo (BCP) é uma estrutura de dados responsável por armazenar todos os dados dos processos ativos. Tal estrutura é armazenada em uma área da memória principal de acesso exclusivo do SO, o qual é responsável pela definição do tamanho dessa área e do número máximo de processos simultâneos que podem ser suportados (Machado e Maia, 2013). Outras informações como PID, UID e tamanho ocupado em RAM (*Random Access Memory* – Memória de Acesso Aleatório) fazem parte do conteúdo do BCP.

2.5 Estrutura de Armazenamento

A função básica da estrutura de armazenamento de um computador é armazenar um determinado dado e permitir que ele seja recuperado quando necessário. Essa estrutura é organizada de maneira hierárquica e composta por dispositivos, como: registradores, memória cache, memória principal, discos sólidos e discos magnéticos (memória secundária). De acordo com Tanenbaum (2006), os principais parâmetros empregados na análise das características e consequente classificação hierárquica de um dispositivo de memória, são:

- Tempo de acesso: Indica o período de tempo decorrido entre o início de uma operação de leitura de dados em um dispositivo de memória e o momento em que esse dado é efetivamente transferido ao solicitante. Esse parâmetro também pode ser usado para medir o desempenho do dispositivo e depende, não apenas do modo como o sistema de memória é construído, mas também da velocidade de seus circuitos integrados (Tanenbaum, 2006);
- Capacidade de armazenamento: É a quantidade de dados suportados por um dispositivo de memória. As principais unidades de medidas usadas para esse parâmetro são *Kilobyte* (Kb), *Megabyte* (Mb), *Gigabyte* (Gb) e o *TeraByte* (Tb);
- Volatilidade: é a capacidade do dispositivo de manter ou não as informações. Memórias voláteis, como é o caso da memória RAM, são aquelas que removem os dados armazenados após serem utilizados, ou após o desligamento do computador. Já as memórias não voláteis mantêm os dados armazenados para uso posterior como, por exemplo, os discos sólidos (Tanenbaum, 2006);
- Custo: O custo monetário de um dispositivo de memória, além de depender de todos os parâmetros descritos anteriormente, depende fortemente da tecnologia de fabricação do dispositivo.

A Figura 2 apresenta a organização hierárquica da estrutura de armazenamento em forma de pirâmide, onde os dispositivos localizados mais próximos do topo apresentam maior desempenho, maior custo e menor capacidade de armazenamento, enquanto que os mais próximos da base possuem menor desempenho, menor custo e maior capacidade de armazenamento (Tanenbaum, 2006).

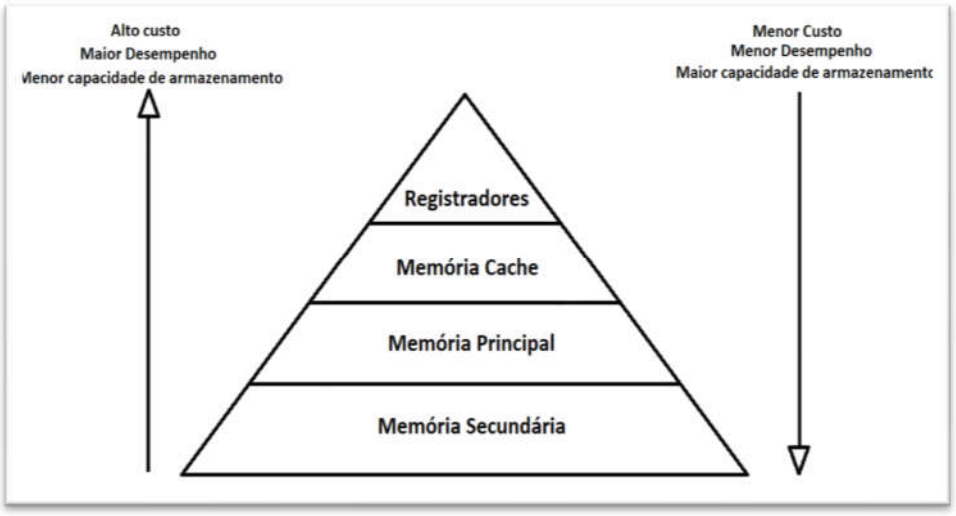


Figura 2: Pirâmide hierárquica dos dispositivos que constituem a estrutura de armazenamento (Fonte: Adaptada de Tanenbaum, 2006)

O trabalho do SO é coordenar como essas memórias serão utilizadas. A principal função de um gerenciador de memória principal é controlar as partes das memórias RAM que estão em uso ou não, alocar e desalocar espaços da memória aos processos quando necessário e gerenciar a troca entre memória principal e o disco quando a memória principal é insuficiente. Geralmente, os programas são armazenados em memórias secundárias, por serem não-voláteis, terem maior capacidade de armazenamento e baixo custo. No entanto, o processador sempre executa instruções localizadas na memória principal.

2.5.1 Gerenciamento de memória principal

A memória principal, também chamada de RAM, é um dispositivo de armazenamento volátil que, juntamente com os registradores e memória cache, podem ser acessados diretamente pelo processador. É através da memória RAM que o processador armazena e executa os programas do computador. Dados e instruções de um programa que não estão alocados nestes dispositivos, não podem ser executados pelo processador (Silberschatz, 2015).

A memória principal é implementada por uma tecnologia de semicondutores, denominada *Dynamic Random Access Memory* - DRAM (Memória de Acesso Aleatório Dinâmico) e é organizada por um vetor de palavras. Cada palavra possui seu próprio endereço na memória.

Existem dois tipos de memória principal: a memória lógica e a memória física. A memória lógica é aquela manipulada pelos programas, de modo que sempre que um programa necessita alocar um espaço na memória, esse espaço é alocado na memória lógica. Já a memória física possui um espaço de endereçamento físico, que é o conjunto formado por todos os endereços dos circuitos integrados que formam a memória. É nesta memória que os espaços alocados em memória lógica vão realmente residir. O processo de conversão (mapeamento) de endereços lógicos em endereços físicos é realizado por uma unidade de gerência de memória chamada MMU (*Memory Management Unit*).

Mesmo com a atual redução de custos e aumento da capacidade de armazenamento, a memória principal continua sendo um recurso computacional escasso, já que os programas atuais demandam de maior quantidade de memória disponível. Sendo assim, com o intuito de aumentar a disponibilidade deste recurso, o SO deve otimizar a utilização da memória de acordo com o número de usuários e aplicações do ambiente computacional. As principais funções da gerência de memória principal do SO são (Machado e Maia, 2013):

- Permitir acesso direto à memória principal.
- Criar uma abstração lógica da memória RAM.
- Oferecer técnicas de alocação de processos na memória RAM, com o intuito de manter este recurso com o maior número de processos residentes e, assim, otimizar o compartilhamento desse recurso. Tais técnicas também permitem controlar o desperdício de memória.
- Proteger o espaço de endereçamento dos processos.

2.5.2 Abstração da memória principal

Com o intuito de realizar um compartilhamento organizado e seguro da memória principal em um ambiente de múltiplos processos, o SO cria uma abstração lógica desta memória. Neste processo de abstração, cada posição física da memória do *hardware* é mapeada pelo SO, de modo que cada uma delas é correlacionada a uma posição lógica no ambiente criado pela abstração da memória RAM, chamada de endereçamento lógico, cujo tamanho é delimitado por um registrador-base e um registrador-limite (Tanenbaum, 2016). O endereço lógico, ou o conjunto deles, ocupado por um processo, durante sua alocação na memória, é chamado de espaço de endereçamento (Seção 2.4.3).

Segundo Silberschatz (2015), a correlação, do espaço de endereçamento de um processo no ambiente criado pela abstração com a sua localização física no *hardware*, pode ser feita durante as diferentes etapas do ciclo de vida de um processo. Ela pode ocorrer no momento da compilação, no momento do carregamento (conhecido como em tempo de carga) ou no momento da execução do mesmo. As diferentes técnicas de alocação de memória, descritas com maiores detalhes na próxima seção, são as responsáveis pela determinação do momento em que será realizada a correlação.

2.6 Técnicas de alocação de memória principal

Há uma enorme variedade de programas que podem ser executados por um computador, incluindo o próprio SO. No entanto, para serem executados, esses programas devem ser alocados na memória principal. Neste contexto, é necessário definir técnicas para alocar, remover e substituir processos na memória RAM, de forma organizada e segura, evitando a fragmentação[14] dos mesmos, uma vez que, em ambientes multiprogramáveis, as áreas de memória ocupadas por cada processo devem ser protegidas. As principais técnicas de alocação de processos na memória principal serão abordadas nos tópicos a seguir.

2.6.1 Alocação Contígua Simples

Nesta técnica, implementada em SOs monoprogramáveis como o MS-DOS, após o processo de abstração da memória principal (Seção 2.5.2), a mesma é dividida em dois endereços lógicos (também chamados de partição): o primeiro é destinado ao SO e o segundo é disponibilizado ao processo que será alocado. O tamanho do espaço de endereçamento do processo é definido por um registrador, como mostra a Figura 3, o qual também é responsável por proteger a partição destinada ao SO de acessos indevidos, intencionais ou não. Também é possível observar na figura, que o tamanho da partição destinada ao SO não é necessariamente igual ao tamanho da partição destinada ao processo.

[14] Fragmentar, segundo o dicionário Aurélio, significa reduzir ou fazer-se em fragmentos. Na gerência de memória principal é o termo usado para definir um desperdício do espaço de endereçamento de memória.

A principal vantagem dessa técnica de alocação de memória é a simplicidade de implementação e o código reduzido. Entretanto, sua desvantagem está no fato do SO dedicar os recursos computacionais, como a memória principal, exclusivamente à execução de um único processo corrente. Além disso, esses recursos são utilizados sem o intuito de obter a máxima eficiência dos mesmos, podendo resultar em um desperdício de memória, principalmente quando a memória requisitada pelo programa corrente é menor do que a quantidade disponível. Em contrapartida, o desenvolvedor do programa que será alocado deve preocupar-se apenas que seu *software* não ultrapasse a quantidade de memória disponível (Machado e Maia, 2013).

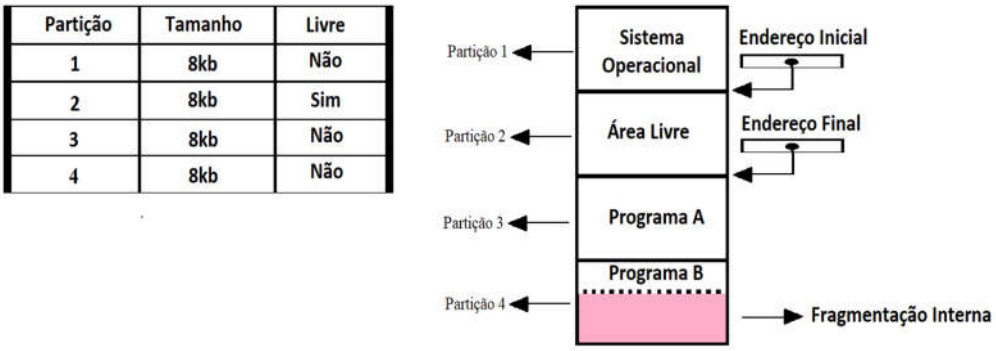
Figura 3: Representação esquemática da técnica de Alocação Contígua Simples (Fonte: Adaptada de Machado e Maia, 2013)

2.6.2 Alocação Particionada Estática

Nesta técnica, implementada em SOs multiprogramáveis, após o processo de abstração da memória principal (Seção 2.5.3), a mesma é dividida em endereços lógicos (partições) com tamanhos fixos, delimitados por registradores do endereço inicial e final. Uma representação esquemática pode ser vista na Figura 4. O tamanho das partições é definido na inicialização do SO, uma vez que o mesmo tem um conhecimento prévio da necessidade de memória (tamanho) de cada programa. Para alterar o tamanho das partições, o SO deve ser reinicializado com uma nova configuração. Um exemplo de SO que implementou essa técnica é o OS/MFT – *Multiprogramming with a Fixed Number of Talks*, da IBM (Machado e Maia, 2013). Uma das partições é destinada ao SO, enquanto as demais são disponibilizadas aos processos a serem alocados. É importante considerar que cada processo só pode ocupar uma partição, desde que o seu tamanho seja menor ou igual ao tamanho da mesma. Quando o tamanho do processo é

menor que o tamanho da partição, ocorre o que se chama de fragmentação interna, parâmetro que controla o desperdício de memória dentro da partição, isto é, considera a memória perdida dentro da área alocada para um processo. Portanto, quando há uma fragmentação interna, é válido que o espaço de endereçamento do processo é menor que o tamanho da partição. Para controlar o uso das partições, o SO cria uma tabela com o endereço inicial de cada partição, o tamanho da mesma e o seu status, que define se a partição está ocupada ou não.

Na Figura 4, é possível observar a partição ocupada pelo SO e pelo programa A, assim como a fragmentação interna da partição ocupada pelo programa B. A figura também ilustra um exemplo de tabela criada pelo SO que controla o uso das partições.



Partição	Tamanho	Livre
1	8kb	Não
2	8kb	Sim
3	8kb	Não
4	8kb	Não

Figura 4: Representação esquemática da técnica de Alocação Particionada Estática (Fonte: Adaptada de Machado e Maia, 2013)

2.6.3 Alocação Particionada Dinâmica

A técnica de alocação particionada dinâmica ou variável não usa o conceito de partições com tamanho fixo. Nela, o tamanho de cada partição só é definido após o processo de alocação do SO e dos processos correntes, uma vez que o espaço de endereçamento virtual é o responsável pela definição do tamanho da partição de cada processo. Dessa forma, o problema de fragmentação interna é solucionado, uma vez que não há desperdício de memória dentro da partição. Uma representação esquemática desta técnica é apresentada na Figura 5.

À medida que os processos alocados na memória são finalizados, suas partições deixam de existir, liberando espaço lógico da memória e permitindo a alocação de um novo processo, desde que o espaço de endereçamento do mesmo (partição) seja menor ou igual ao tamanho do espaço lógico disponível. Quando ocorre sobra de espaço lógico da memória, tem-se a chamada

fragmentação externa, parâmetro que controla o desperdício de memória entre as partições, ou seja, considera a memória perdida fora da área ocupada por um processo. A Figura 5 ilustra estes conceitos. Nela, existem duas áreas de memória livre (espaço lógico disponível) com tamanho de 5 kbytes, delimitado por registradores. Dois processos encontram-se na fila de execução e precisam ser alocados na memória: o processo D, com tamanho igual a 4 kbytes, e o processo E, de 6 kbytes. Sendo assim, o processo D poderá ser alocado, com fragmentação externa de 1 kbyte. Observe que a memória total livre no momento é de 5 kbytes, mas ela não é contígua, de modo que o processo E não será alocado e deverá aguardar a liberação de um espaço lógico livre com tamanho maior ou igual a 5 kbytes.

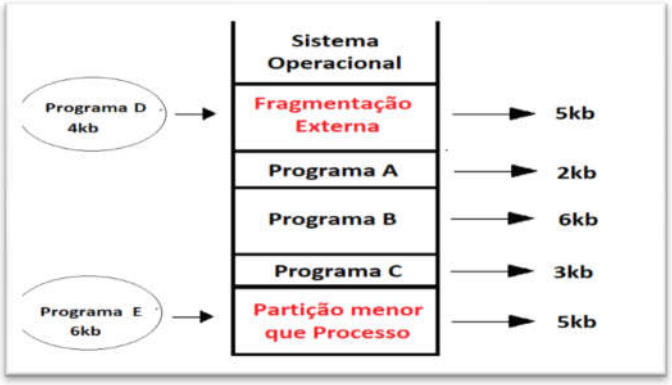


Figura 5: Representação esquemática da técnica de Alocação Particionada Dinâmica (Fonte: Adaptado de Machado e Maia, 2013)

O *Multiprogramming with a Variable Number of Talks* (OS/MVT), da IBM, é um exemplo de SO que implementou essa técnica (Machado e Maia, 2013).

2.6.4 Paginação

O problema de fragmentação externa apresentado pela técnica anterior é decorrente da necessidade de cada processo ocupar uma única área contígua na memória principal. A técnica de alocação por Paginação elimina essa necessidade e, consequentemente, soluciona o problema de fragmentação externa.

Nessa técnica o espaço de endereçamento lógico de um processo é dividido em páginas lógicas de tamanho fixo. Um processo pode utilizar uma ou mais páginas lógicas. Já o endereço lógico de um processo é dividido em duas partes: número de páginas e deslocamento dentro

dessas páginas. A memória física (*hardware*), por sua vez, é dividida em páginas físicas com tamanho fixo e idêntico ao tamanho da página lógica.

Em tempo de carga, um processo é carregado na memória principal, página a página. Cada página lógica de um processo ocupa exatamente uma página física na memória (*hardware*), no entanto, não é necessário que a área ocupada pelo processo seja contígua. Uma tabela, denominada tabela de páginas, é criada para gerenciar a ligação entre páginas lógicas e físicas, ou seja, é responsável por associar o endereço lógico, especificado em cada instrução do processo, com o endereço físico correspondente.

Como a unidade de alocação é a página e um processo sempre ocupa um número inteiro de páginas, tal técnica pode apresentar fragmentação interna. Sendo assim, o SO deve manter, através da gerência de memória, o controle das áreas ocupadas e livres na memória principal. Para facilitar este processo, uma memória *cache* interna ao *hardware* da memória principal, chamado MMU (*Memory Management Unit* – Unidade de Gerenciamento de Memória), mantém uma tabela com informações a respeito da posição das páginas livres na memória.

A Figura 6 ilustra o funcionamento da técnica de paginação, na qual o espaço de endereçamento do processo é dividido em páginas lógicas de tamanho fixo. Na figura, diferentes processos (X, Y e Z) estão alocados nestas páginas. É mostrado também um exemplo de tabela de Páginas, a qual associa as páginas lógicas ocupadas por cada um dos processos às páginas físicas, que se encontram na memória física.

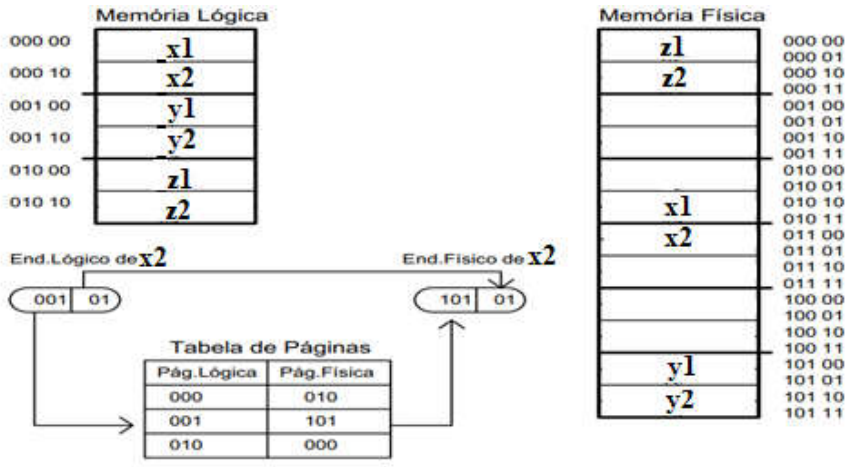


Figura 6: Representação esquemática da técnica de Paginação (Fonte: Adaptada de Oliveira, Carissimi e Toscani, 2001)

2.6.5 Segmentação

Ao contrário da técnica de paginação, a segmentação divide o espaço de endereçamento virtual em blocos de tamanhos diferentes, chamados segmentos, enquanto que na paginação o espaço de endereçamento virtual é dividido em páginas, com tamanhos fixos. Nessa técnica, o programa é tipicamente dividido em quatro segmentos: código, dados alocados estaticamente, dados alocados dinamicamente e pilha de execução. É possível orientar a gerência de memória para suportar diretamente o conceito de segmento. Dessa forma, uma posição da memória lógica passa a ser endereçada por um número de segmento e um deslocamento em relação ao início do mesmo. Durante o mapeamento dos segmentos em endereços físicos, no momento em que o processo é carregado (em tempo de carga), uma tabela de segmentos é criada, a qual informa, para cada segmento, qual o endereço da memória física, onde o mesmo foi colocado e qual o seu tamanho.

Como a área alocada para o segmento tem a quantidade exata de memória, não há fragmentação interna nesta técnica de gerenciamento de memória. No entanto, há ocorrência de fragmentação externa, uma vez que pode haver desperdício de memória após alocação de duas áreas contíguas de diferentes tamanhos. Uma das vantagens desta técnica está na facilidade de compartilhamento da memória, uma vez que cada segmento representa uma parte específica do programa, podendo ou não ser compartilhado. Uma consequência disso, é que uma sub-rotina do programa pode ser alterada, sem que o funcionamento do programa principal sofra com essas mudanças (Silberschatz, 2015). A Figura 7 ilustra o processo de segmentação.

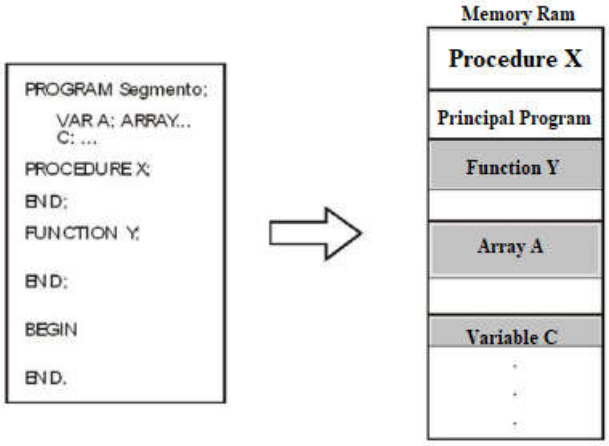


Figura 7: Representação esquemática da técnica de Segmentação (Fonte: Adaptada de Tanembaum, 2016)

2.6.6 Segmentação por paginação

Segundo Tanembaum (2016), a técnica de segmentação pode ser implementada de duas formas: a segmentação pura (apresentada anteriormente) e a segmentação com paginação. A segunda forma de implementação da segmentação surgiu com o objetivo de minimizar o desperdício da memória principal, uma vez que segmentos grandes poderiam ocupar desnecessariamente a RAM. De acordo com Machado e Maia (2013), a segmentação com paginação divide o espaço de endereçamento virtual em segmentos que, por sua vez, são divididos em páginas. Dessa forma, uma posição da memória lógica passa a ser endereçada por um número de segmento, um número de página e um deslocamento em relação ao início da página. A obtenção do endereço físico é feita a partir da combinação da tabela de páginas com a tabela de segmentos, as quais são criadas durante o processo de mapeamento.

Dentre as vantagens desta técnica, além de minimizar o desperdício da memória principal por fragmentação externa, destacam-se: modularidade, viabilização de memória RAM somente às partes do programa que realmente necessitam deste recurso, facilidade de programação, proteção e facilidade de compartilhamento da memória (Tanembaum, 2016).

3. REALIDADE VIRTUAL (RV)

Este capítulo é dedicado à apresentação de um referencial teórico que aborda os principais conceitos, definições e classificações das diferentes tecnologias de RV.

Na Seção 3.1, é apresentada uma introdução à RV, abordando definição, classificação e características dos diferentes tipos de sistemas. Na Seção 3.2 são apresentados os conceitos básicos e relevantes para a compreensão da tecnologia e seu funcionamento. Já a Seção 3.3 aborda os dispositivos não convencionais de I/O usados em sistemas de RV.

3.1 Introdução

Atualmente são encontrados vários tipos de *hardware* e *software* de baixo custo que possibilitam o desenvolvimento de aplicações em RV. Na educação, a RV é utilizada no desenvolvimento de novas metodologias de ensino-aprendizagem, uma vez que as aplicações baseadas nessa tecnologia, nas quais permitem a combinação do mundo físico e virtual, estimulam o processo de observação, exploração e construção do conteúdo estudado (Cardoso, 2007). A RV possibilita criar um modelo de visualização que permite diminuir a distância entre os modelos mentais do aluno e os conceitos teóricos abordados na disciplina estudada (Pears, 2007).

O termo RV foi criado pelo fundador da VPL Research Inc.[15], Jaron Lanier, no início dos anos 80, o qual passou a utilizar o termo para diferenciar as simulações tradicionais das simulações feitas em computadores envolvendo múltiplos usuários em um ambiente compartilhado (Araújo, 1996). Outras terminologias já foram utilizadas para a tecnologia, como Realidade Artificial e *Cyberspace* (Gibson, 1984).

A definição desta tecnologia é bastante abrangente e baseada nas experiências próprias dos desenvolvedores e pesquisadores do tema. Para Brooks (1999), a RV permite ao usuário uma experiência de imersão em um mundo virtual no qual ele possui o controle dinâmico do ponto de vista deste mundo. Segundo Sherman e Craig (2002), a RV é composta por um mundo virtual, interativo e tridimensional, que permite a imersão e resposta às ações dos usuários. Para

[15] VPL (*Virtual Programming Languages*) é uma empresa de tecnologia criada em 1984. Atualmente é uma das mais engajadas no desenvolvimento de tecnologias aplicadas à realidade virtual.

Kirner e Siscouto (2016) e Hancock (1995), a RV é, acima de tudo, uma interface avançada que permite ao usuário acessar aplicações executadas em computadores ou outros dispositivos, como *HMDs* (*Head-Mounted Display*) e sistemas de multiprojeção baseados no CAVE (*Cave Automatic Virtual Environment*). Também é permitida a movimentação do usuário no ambiente virtual, em tempo real. Em suma, trata-se de uma tecnologia de interface avançada entre um usuário e um sistema, cujo objetivo é recriar ao máximo a sensação de realidade para um indivíduo através de uma experiência imersiva, interativa e com envolvimento, proporcionada através de estímulos multissensoriais.

Aplicações práticas dessa tecnologia requerem *softwares* e *hardwares* que permitam que o usuário navegue e observe um mundo virtual em tempo real, com seis graus de liberdade (6DOF – Six Degrees of Freedom), além de ser necessário o uso do processo de *Rendering* (Seção 3.3.5). Esse mundo virtual é criado a partir de técnicas de computação gráfica e é frequentemente confundido com animação, como CAD[16] (*Computer Aided Design*) ou multimídia. Segundo Leston (1996), a RV diferencia-se dessas tecnologias por ser:

- Orientada aos usuários.
- Imersiva, pois apresenta forte sensação de presença no mundo virtual.
- Interativa, uma vez que o ambiente é influenciado e modificado em resposta às ações do usuário.
- Intuitiva, uma vez que o usuário pode manipular as interfaces computacionais apresentadas no ambiente com pouca ou nenhuma dificuldade. O conhecimento intuitivo do usuário sobre o mundo físico pode ser usado para facilitar o entendimento do mundo virtual.

A interação com sistemas de RV pode ser feita através de dispositivos variados, como os 3DUI (3D *User Interface*), *Microsoft Kinect*, *Wii Remote*, dispositivos 6DOF, câmeras e luvas de dados. No desenvolvimento de sistemas RV são utilizadas bibliotecas que fornecem os recursos necessários para operação dos componentes de interação, enquanto que a aplicação do desenvolvedor implementa o mundo virtual e sua renderização (Guimarães, Nunes, Rieder e Hounsell, 2015).

[16] CAD (*Computer aided design*) é o nome genérico dado a *softwares* utilizados pela Engenharia, Geografia, Arquitetura e Design para facilitar o projeto e desenhos técnicos.

3.2 Características dos sistemas de RV

Segundo Guimarães, Nunes, Rieder e Hounsell (2015), os sistemas de RV caracterizam-se pela coexistência integrada de três conceitos básicos: imersão, interação e sensação de presença, também chamada de envolvimento.

A imersão consiste no sentimento do usuário em fazer parte do ambiente virtual. A percepção visual, auditiva e tátil do usuário deve ser estimulada com o intuito de fortalecer esse sentimento. É uma das características primordiais de um bom sistema de RV e está associada aos dispositivos físicos que o usuário utiliza para ingressar na experiência em RV. Os sistemas de RV podem ser classificados em imersivos e semi-imersivos. Em sistemas imersivos, o usuário percebe apenas o mundo virtual e não possui contato com o mundo real. Esse tipo de sistema permite a visualização tridimensional e a movimentação do usuário no ambiente ao qual o mesmo se encontra. Nestes casos, é comum o uso de capacetes de RV. Recursos, como a texturização de objetos e a inserção de sons, podem ser incorporados para aumentar o realismo do ambiente virtual (Araújo, 1996). Em sistemas semi-imersivos o usuário recebe estímulos tanto do mundo real quanto do virtual, mas maior importância é dada aos estímulos vindos do mundo virtual. Neste caso é comum o uso de monitores.

Segundo Netto, Machado e Oliveira (2002), a interação está associada à capacidade de modificação do ambiente virtual em resposta às ações efetuadas pelos usuários. O computador deve detectar essas ações e modificar instantaneamente o mundo virtual. A interação pode ser feita por meio de dispositivos convencionais, como *mouse* e teclado, ou através de dispositivos não convencionais, como câmeras de captura de movimento.

Já conceito de sensação de presença, ou envolvimento, considera a percepção do usuário em estar presente no ambiente virtual e define o grau de envolvimento do usuário com a aplicação. Existem situações em que o envolvimento não está associado à imersão, como é o caso do uso de *avatares* que representam o usuário.

3.2.1 RV passiva, exploratória ou interativa

De acordo com Adams (1994), uma aplicação em RV pode proporcionar uma experiência de três formas diferentes: passiva, exploratória ou interativa.

1. Passiva: O foco desse tipo de experiência é o ambiente, o qual é explorado de forma automática e sem interferência do usuário. A rota e os pontos de observação são previamente determinados e controlados pelo *software* em utilização.
2. Exploratória: O caminho exploratório é dirigido pelo usuário, mas não há interação com o ambiente, ou seja, o usuário tem a liberdade de escolher sua rota e pontos de observação, mas não interage com os elementos presentes no mundo virtual. O foco dessa experiência se mantém no ambiente.
3. Interativa: Neste tipo de experiência, o usuário define sua rota dentro do ambiente virtual e pode interagir com elementos presentes nele. Pode realizar ações como visualizar, explorar, manipular e alterar o mundo virtual e seus componentes usando seus sentidos (Kirner e Siscouto, 2016).

Todas as interações do usuário com o ambiente devem ocorrer dentro de limites de tempo de resposta bem definidos, dando ao usuário, a impressão de que o mesmo atua no ambiente virtual em tempo real. A visualização deve ser feita de forma nítida, independente da tecnologia utilizada. Normalmente, é necessário que os displays representem as informações em pelo menos 30 quadros por segundo (*frames per second* – FPS), sem isso a imagem parecerá "piscando" (Guimarães, Nunes, Rieder e Hounsell, 2015).

A navegação em ambientes virtuais é controlada pelo posicionamento do observador no mundo real, o chamado *viewpoint* (ponto de observação). Dependendo do tipo de experiência utilizada, o posicionamento do observador passa a determinar a forma de interação com o ambiente.

3.3 Conceitos básicos de RV

O desenvolvimento e a concepção de sistemas de RV envolvem conceitos de diferentes áreas de estudo, como espaços 3D, estereoscopia, rastreadores, dispositivos acústicos, óptica, ferramentas de *hardware* e *software*, entre outros. Esta seção é dedicada em apresentar os mais relevantes desses conceitos.

3.3.1 Espaço 3D e ambiente virtual

Uma das principais características da RV é transportar o usuário para o ambiente virtual. O mundo virtual de sistemas de RV é 3D, mas os dispositivos utilizados na visualização deste mundo são bidimensionais (2D). No processo de visualização de uma imagem ou objetos 3D, é necessário projetar seu conteúdo sobre um plano 2D (Foley, 1990). A projeção de um elemento 3D é definida pela intersecção dos raios de projeção provenientes de um centro de projeção e que atingem pontos do objeto a ser projetado com o plano de projeção (Netto, Machado e Oliveira, 2002). Esse tipo de projeção é conhecido como geométrica planar e pode ser subdividido em duas classes: perspectiva e paralela, as quais podem ser diferenciadas pela distância entre o centro de projeção e o plano de projeção. Se essa distância for finita a classe é perspectiva, caso contrário, é paralela.

Na classe de projeção perspectiva, o centro de projeção é um ponto próprio no espaço 3D e utiliza até três pontos de fuga, os quais correspondem ao valor das coordenadas (x, y, z) do plano cartesiano[17] nos pontos em que tais eixos são interceptados pelo plano de projeção. Os cenários de imagens e animações são considerados bons exemplos deste tipo de classe (Foley, 1990).

Na classe de projeção paralela, o centro de projeção está posicionado no infinito e, consequentemente, seus raios de projeção são linhas paralelas. Por este motivo, uma direção de projeção é definida ao invés de um centro de projeção. Esse tipo de projeção é bastante utilizado na extração de medidas de objetos de cena, pois preserva a proporcionalidade das medidas, lados e ângulos (Foley, 1990).

Com relação ao ambiente virtual, os elementos usados para a sua construção tendem a buscar o realismo e apresentam atributos próprios, como cores, textura, posicionamento e iluminação. Comportamentos também podem ser atribuídos a esses elementos, que melhoram a interação do usuário com o mundo virtual e ajudam o sistema de RV a atingir seus objetivos (Kirner e Siscouto 2016). No entanto, há situações em que o ambiente virtual é construído com base em um modelo abstrato, sem referências ao mundo real. A escolha entre um modelo verossímil ou abstrato depende do objetivo do *software* em desenvolvimento.

[17] Plano cartesiano é um método criado pelo filósofo e matemático René Descartes. No caso bidimensional, corresponde a dois eixos perpendiculares pertencentes a um plano comum.

3.3.2 Estereoscopia e Displays em RV

Cada um dos olhos humanos, devido à sua localização na face, visualiza imagens ligeiramente diferentes de um mesmo objeto. A obtenção de duas imagens simultâneas, a partir de pontos de observações diferentes corresponde ao fenômeno chamado de Estereoscopia. Tais imagens são fundidas pelo cérebro humano em uma única representação. Nesse processo, são obtidas informações como profundidade, distância, posição e tamanho dos objetos presentes na cena, responsáveis pela visão 3D (Esteio, 2002).

Em geral, sistemas de RV fazem uso de imagens estereoscópicas, sejam elas ativas ou passivas, e implementam o movimento de paralaxe. Para isso, a correção da perspectiva do ambiente virtual de acordo com o ponto de vista do usuário é feita através de um ou mais dispositivos de rastreamento (Seção 3.2.3) de posição da cabeça (Guimarães, Nunes, Rieder e Hounsell, 2015), os quais serão descritos a seguir.

No âmbito computacional, a principal desvantagem da estereoscopia é a necessidade de mais capacidade de processamento do sistema de RV, já que duas imagens precisam ser processadas ao invés de uma. A vantagem é oferecer ao usuário uma visão mais próxima da visão humana e, portanto, com maior verossimilhança do mundo real.

3.3.3 Rastreadores

Em sistemas de RV, o controle do campo de visão do usuário, assim como a localização do mesmo no ambiente virtual são parâmetros de grande importância. É através das técnicas de rastreamento que o computador pode reconhecer a posição e os movimentos do usuário no espaço. Os rastreadores de posição informam ao computador a posição e a orientação do sensor que está monitorando. Os rastreadores utilizam um conjunto de seis coordenadas para identificar a posição e a orientação dos objetos rastreados. Essas coordenadas são conhecidas como DOF. Em sistemas 6DOF, Figura 8, existem seis possíveis direções diferentes de movimentação de um objeto, cada um deles controlado por um eixo específico: para frente ou para trás (eixo X), para cima ou para baixo (eixo Y), para esquerda ou para direita (eixo Z), rotação no eixo X (*Roll*), rotação no eixo Y (*Yaw*) e rotação no eixo Z (*Pitch*).

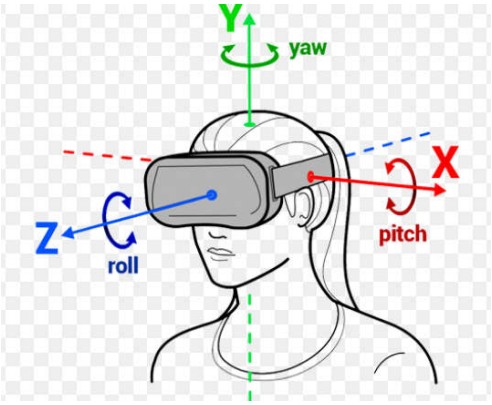


Figura 8: Coordenadas utilizadas por rastreadores de posição 6DOF (Fonte: Guimarães, Nunes, Rieder e Hounsell, 2015)

Existem rastreadores de posição construídos com diferentes tecnologias, como:

- Rastreadores eletromagnéticos: Esse tipo de rastreador possui um emissor de ondas eletromagnéticas que define o ponto de referência de coordenadas dos sensores monitorados dentro do mundo real.
- Rastreadores ultrassônicos: Utilizam ondas sonoras para realizar o rastreamento.
- Rastreadores mecânicos: São formados por um conjunto de componentes ligados entre si por articulações, as quais permitem rastrear precisamente a posição do objeto rastreado.
- Rastreadores ópticos: Utilizam sistemas de câmeras para filmar o ambiente e rastrear os objetos nele contidos.

Os rastreadores de posição em conjunto com HMDs são os mais utilizados em sistemas de RV. No entanto, existem outros tipos de rastreadores, responsáveis por detectar a posição e o movimento que são realizados por uma parte do corpo do usuário, como os rastreadores da cabeça, rastreadores dos olhos e rastreadores do corpo inteiro. A escolha de determinado tipo de rastreador depende diretamente do objetivo do sistema em RV utilizado.

3.3.4 Som tridimensional

Também conhecido como som binaural[18], o som 3D é um fator importante e auxilia na sensação de imersão do usuário no mundo virtual. As gravações de som 3D baseiam-se em um

[18] Binaural: literalmente "relacionado às duas orelhas", permite determinar a origem dos sons.

processo de manipulação auditiva, a qual permite que os sons sejam posicionados no espaço, controlando sua direção, distância e profundidade. Esses sons podem ser gerados por alto falantes ou fones de ouvido, dando ao usuário a sensação auditiva condizente com o ambiente virtual.

3.3.5 Rendering

O primeiro passo na criação de imagens pelo computador é descrever a cena 3D para o sistema. Para isso, é necessário criar um modelo tridimensional que permita que a cena possa ser visualizada por câmeras virtuais no sistema de RV (Albuquerque, 1999). O processo conhecido como *rendering* ou renderização, permite a construção de uma imagem bidimensional da cena 3D. Neste processo, parâmetros como a quantidade de faces poligonais a serem processadas, a quantidade e o tipo das fontes de iluminação presentes na cena, os cálculos de reflexão da luz, a radiosidade, a projeção de sombras e a quantidade de superfícies texturizadas devem ser levados em consideração (Elliott e Miller, 1995). Como resultado, os sólidos são projetados e as superfícies são tonalizadas (*shaded*) para a exibição em uma tela 2D.

3.3.6 Hardware e Software de RV

É necessário definir, desde a concepção do sistema RV, o conjunto de interfaces físicas necessárias ao funcionamento da aplicação em desenvolvimento (Kirner e Siscouto, 2016). Tais interfaces definem o conjunto de *hardware* comum a qualquer sistema de *software*, como processadores, dispositivos de armazenamento e *displays*, somados a outros dispositivos não convencionais de interação, específicos da área de RV, como *joysticks* e luvas de dados. O uso de dispositivos de interação não convencionais em RV às vezes é necessário, pois eles permitem ampliar a sensação de realidade transmitida pelo sistema ao usuário. Há uma diversidade de equipamentos responsáveis por estimular os diversos sentidos humanos, como visão, audição e tato.

Os componentes de *software* de um sistema de RV permitem a integração e tratamento dos dados da aplicação, cujo resultado é um mundo virtual. Sua criação envolve o uso de uma

vasta gama de tecnologias, *softwares* e linguagens de programação. Um mundo virtual é composto, basicamente, por objetos tridimensionais e regras de funcionamento. A criação de objetos envolve a utilização de ferramentas de modelagem, edição de imagem, som e vídeo. Já as regras de comportamento são definidas através de linguagens de programação que permitem associar funções aos objetos, tornando o mundo virtual mais interativo e próximo da realidade. S*oftwares* de cada parte do sistema, desde *firmwares*[19] até bibliotecas de desenvolvimento em RV, são responsáveis pela arquitetura de controle do mundo virtual.

O ambiente computacional usado para suportar sistemas de RV pode variar. É comum o uso de computadores pessoais, estações de trabalho com múltiplos processadores e até mesmo *smartphones*. A escolha do ambiente mais adequado depende de fatores como desempenho, capacidade de processamento gráfico e comunicação em redes. Contudo, independente do dispositivo escolhido, o ambiente deverá suportar a visualização e os sinais de I/O em tempo real, com uma taxa de atraso aceitável (Kirner e Siscouto, 2016).

3.4 Dispositivos visuais

Os dispositivos visuais e a qualidade da imagem gerada por eles influenciam na sensação de imersão do usuário em um sistema de RV. De acordo com Cadoz (1994), os olhos possuem 70% dos receptores sensitivos humanos, de modo que a maioria das informações recebidas pelo cérebro humano se encontra na forma de imagens.

Considerando os atrasos admissíveis para que o ser humano tenha a sensação de interação em tempo real, a qualidade visual do sistema de RV depende do número de quadros que podem ser processados e exibidos por segundo na renderização das imagens (Seção 3.3).

Os dispositivos visuais podem ser divididos em duas classes: a primeira composta por vídeo-capacetes (HMDs) e *head-coupled displays*, e a segunda, composta por monitores e sistemas de projeção.

[19] *Firmware* é o conjunto de instruções operacionais programadas diretamente no *hardware* de um equipamento eletrônico.

3.4.1 *Head-Mounted Display* (HMD)

Os capacetes de visualização HMDs são dispositivos de RV compostos por *displays* que ficam próximos aos olhos do usuário, o qual fica totalmente submerso no ambiente virtual, já que o sentido da visão é totalmente usado na experiência, ou seja, o usuário consegue visualizar apenas as imagens sintéticas produzidas pelo computador (Burdea e Coiffet, 2003). Segundo Guimarães, Nunes, Rieder e Hounsell (2015) é possível utilizar imagens com resolução em *full HD*[20] e som estéreo[21] neste tipo de dispositivo. Na prática, o *display* do próprio celular pode ser utilizado, mas há HMDs com *displays* embutidos. A figura 9 exibe um modelo de HMD desenvolvido pela empresa *Oculus*[22]. É importante considerar que alguns HMDs possuem rastreadores de posição que identificam o posicionamento do campo de visão do usuário, ou seja, permitem mapear para onde o usuário está olhando.

Figura 9: Exemplo de HMD desenvolvido pela empresa *Oculus* (Fonte: https://www.oled-info.com)

Atualmente são comercializados adaptadores, também chamados de *VR Mobile Phone 3D Glasses*, que transformam *smartphones* em HMDs. Esses dispositivos permitem encaixar o

[20] *Full HD* é a sigla de *Full High Definition* que significa Máxima Alta Definição. Define aparelhos com resolução de 1920 colunas de *pixels* e 1080 linhas de *pixels*.

[21] Som estéreo é um sistema de reprodução de áudio que utiliza os dois canais de sons monaurais sincronizados.

[22] Oculus VR é uma empresa americana de tecnologia fundada por Palmer Luckey, Brendan Iribe, Michael Antonov e Nate Mitchell em 2012, grande parte de seus produtos é desenvolvido para tecnologia de RV.

aparelho celular em suportes, deixando a visão do usuário próximo da tela, permitindo a observação da imagem do celular.

3.4.2 Monitores convencionais

Segundo Guimarães, Nunes, Rieder e Hounsell (2015), os monitores proporcionam a mais simples forma de visualização do ambiente virtual, pois é feita com a utilização de monitores convencionais. No processo de visualização de imagens em monitores são usadas técnicas de computação gráfica que tentam representar objetos tridimensionais, sua distância, profundidade, forma e cor. Um exemplo de aplicação em RV utilizando um monitor convencional é mostrado na Figura 10.

Figura 10: Exemplo de uso de monitor convencional em conjunto com HMD (Fonte: Desert West, 2018)

Óculos especiais, conhecidos como *shutterglasses*, também podem ser usados na visualização do ambiente virtual. Um exemplo desses óculos, da empresa japonesa JVC (*Japan Victor Company*), é apresentado na Figura 11. Neste tipo de dispositivo a visualização é estereoscópica (Seção 3.2.2), ou seja, as imagens são renderizadas duas vezes, uma para cada olho humano, com um pequeno deslocamento entre elas. Lentes especiais são utilizadas para que cada olho tenha percepção dessas imagens.

Figura 11: Exemplo de *shutterglasses* que pode ser utilizado na visualização do ambiente virtual (Fonte: *Japan Victor Company*, 2018)

3.4.3 Sistemas de projeções

Nesse tipo de visualização podem ser utilizados um conjunto de monitores convencionais ou projetores multimídia, resultando em grandes projeções do ambiente virtual. O tamanho da tela de projeção influencia diretamente na interface com o ambiente virtual. O uso de displays maiores permite que o usuário possa se locomover sobre as projeções e visualizar os objetos em tamanho real. A visão nesse tipo de sistema pode ser monoscópica ou estereoscópica (Burdea e Coiffet, 2003).

Um exemplo desse sistema de projeções é o CAVE (*Cave Automatic Virtual Environment*), um ambiente imersivo que é capaz de inserir todo o corpo do usuário no ambiente virtual. O espaço CAVE é organizado em formato de cubo, com imagens estereoscópicas projetadas em todos os lados, dando ao usuário a sensação de estar no centro do ambiente virtual e, consequentemente no centro dos elementos inseridos no mesmo (Burdea e Coiffet, 2003).

Uma das maiores vantagens desse tipo de ambiente é permitir que mais de um usuário esteja inserido simultaneamente no ambiente virtual. A quantidade de pessoas em um mesmo ambiente depende do tamanho do CAVE. No entanto, apenas um usuário pode realizar ações dentro do ambiente, enquanto os demais são agentes passivos e apenas observam as interações

sob o ponto de vista do controlador (Guimarães, Nunes, Rieder e Hounsell, 2015). A Figura 12 apresenta um ambiente CAVE utilizado pela empresa automotiva *Jaguar Land Rover*[23].

Figura 12: Ambiente imersivo CAVE da empresa Jaguar Land Rover, um exemplo de sistema de projeção (Fonte: www.vrcircle.com, 2018)

3.4.4 Dispositivos de interação

Interação é o processo de comunicação entre o usuário e o sistema. Os dispositivos de interação são responsáveis por captar e enviar os comandos dos usuários ao sistema de forma "não natural", ou seja, é necessário executar algum comando no dispositivo de interação para que a comunicação ocorra.

Há diversos dispositivos que podem ser utilizados para interação com o sistema RV, como os listados a seguir:

- Telas de toque (*Touch Screen*): São displays sensíveis ao toque que, através de um sistema de coordenadas cartesianas, identificam a posição tocada e captam a entrada da interação. A partir dessa tecnologia é possível selecionar *menus*, selecionar e arrastar objetos, definir zoom de visualização, entre outros.

[23] *Jaguar Land Rover* é uma multinacional automotiva britânica.

- *Joystick*: Dispositivo composto por botões e alavanca que, quando pressionados, executam tarefas dentro do ambiente virtual, como seleção de objetos e movimentação no ambiente.
- Comando de voz: Permite a interação com o sistema por meio de comando auditivos compostos de linguagens simples. Com essa tecnologia as mãos do usuário ficam livres para quaisquer outras atividades.
- Dispositivos hápticos: São dispositivos destinados ao uso geral ou específico (Guimarães, Nunes, Rieder e Hounsell, 2015), como os dispositivos que estimulam sensações táteis do usuário (Gradecki, 1995). Os dispositivos hápticos são essencialmente utilizados em aplicações nas quais a imersão depende de sensações que vão além da visão e da audição, de modo que também é necessária uma interação eletromecânica com o corpo do usuário. A utilização de tais dispositivos requer sistemas computacionais e dispositivos de entrada e saída específicos. É o caso dos exoesqueletos, nos quais o usuário se encaixa e todos os movimentos associados a ele são transmitidos ao mundo virtual.

4. TRABALHOS RELACIONADOS

Este capítulo apresenta alguns *softwares* simuladores genéricos de SOs, cujas características, vantagens e desvantagens foram analisadas durante a etapa de concepção do *SigemVR*. Uma vez que o *software SigemVR* tem como objetivo simular o funcionamento do módulo de gerência de memória principal em SOs, na revisão bibliográfica foram encontrados os seguintes simuladores de gerência da memória principal: SOsim (Maia, 2001), o SSOG (Kioki e Santiago, 2008), o TBC-SO/WEB (Reis, Parreira Jr e Costa, 2009) e o ESORV (Scamati, 2017).

4.1 SOsim

Simulador desenvolvido por Luiz Paulo Maia em linguagem de programação Pascal com utilização de paradigma de orientação a objetos, permite a apresentação de conceitos e mecanismos de um SO de forma simples e animada (Maia, 2001). Os módulos abordados nas simulações são: multiprogramação, processos e suas mudanças de estado, escalonamento e gerência de memória principal. Sua interface, bidimensional (2D), apresenta janelas separadas para cada um dos módulos citados.

Com relação ao módulo de gerência de memória, aborda apenas a técnica de alocação de processos por paginação. O simulador representa a memória principal como uma matriz bidimensional de cem posições, denominadas *frames* e cada processo pode ocupar no máximo cinco *frames*. A busca por páginas é configurável e pode ser feita através de paginação por demanda ou antecipada. Para substituição de páginas é utilizado o algoritmo FIFO (*First In First Out*). Sempre que um processo é alocado na memória principal, o seu espaço de endereçamento é representado por uma cor escolhida pelo usuário. Na Figura 13 é apresentada uma tela do simulador, na qual é mostrada a distribuição de processos na memória principal e o seu escalonamento.

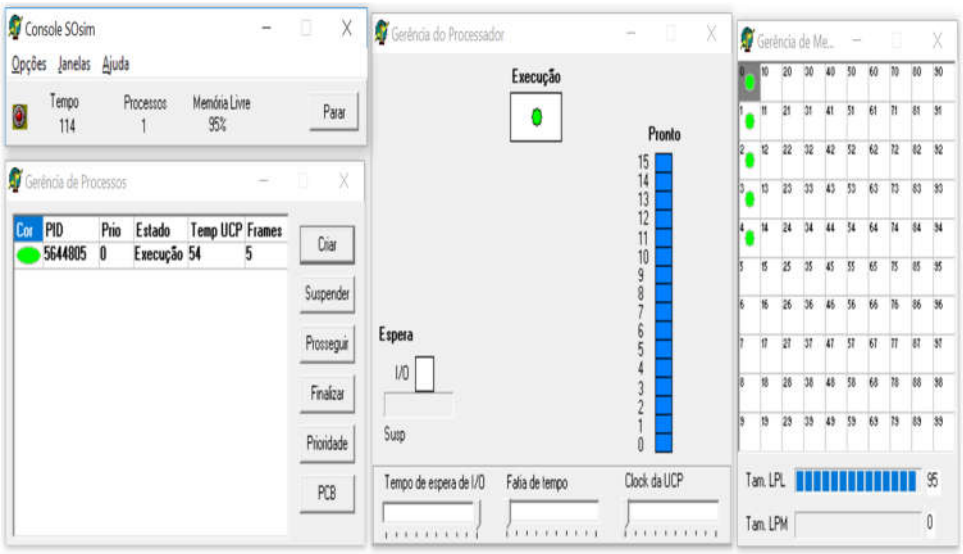


Figura 13: Interface do simulador SOsim (Fonte: Maia, 2001)

4.2 Simulador de Sistema Operacional Genérico (SSOG)

O Simulador de Sistema Operacional Genérico (SSOG) é um *software* didático que aborda alguns tópicos da disciplina de SO de forma ilustrativa e dinâmica por meio de uma interface gráfica simples e de fácil manuseio, inteiramente desenvolvido na linguagem de programação *JAVA*.

Segundo Kioki e Santiago (2008), desenvolvedores do *software*, o mesmo foi desenvolvido com o intuito de cumprir os seguintes objetivos:

- Ser de utilização simples e interativa, com foco nos alunos.
- Apresentar de maneira objetiva os conceitos de gerência de memória principal e virtual, gerência de processador e escalonamento de processos.
- Utilizar uma linguagem de programação multiplataforma.

O SSOG é dividido em dois módulos com as seguintes funcionalidades: gerência de processador e gerência de memória principal. Todo o módulo de gerência de memória baseia-se na técnica de alocação da memória virtual por paginação. A memória principal é representada por uma matriz bidimensional (2D) de cem posições, denominadas *frames*, onde os vinte primeiros *frames* são reservados ao SO, enquanto que as demais posições são reservadas à

alocação de processos. A interface 2D apresenta informações como: a quantidade de memória livre e ocupada, a quantidade de processos alocados na memória, a ordem de execução do processo, seu identificador (PID), seu tipo, prioridade e o tempo de processamento em milissegundos. Embora seu objetivo seja simular a técnica de Memória Virtual por Paginação, não há a representação do dispositivo de armazenamento secundário, dando a impressão que todos os *frames* são alocados somente na memória principal.

A Figura 14 apresenta a matriz representativa da memória principal e o BCP com informações sobre a quantidade de memória usada e livre, assim como a quantidade de processos alocados. É possível perceber que as vinte primeiras posições da matriz possuem fundo de cor diferente das demais, indicando que estão sendo utilizadas, possivelmente pelo SO. Ao lado da matriz é apresentada uma tabela com as informações a respeito dos processos a serem alocados na memória.

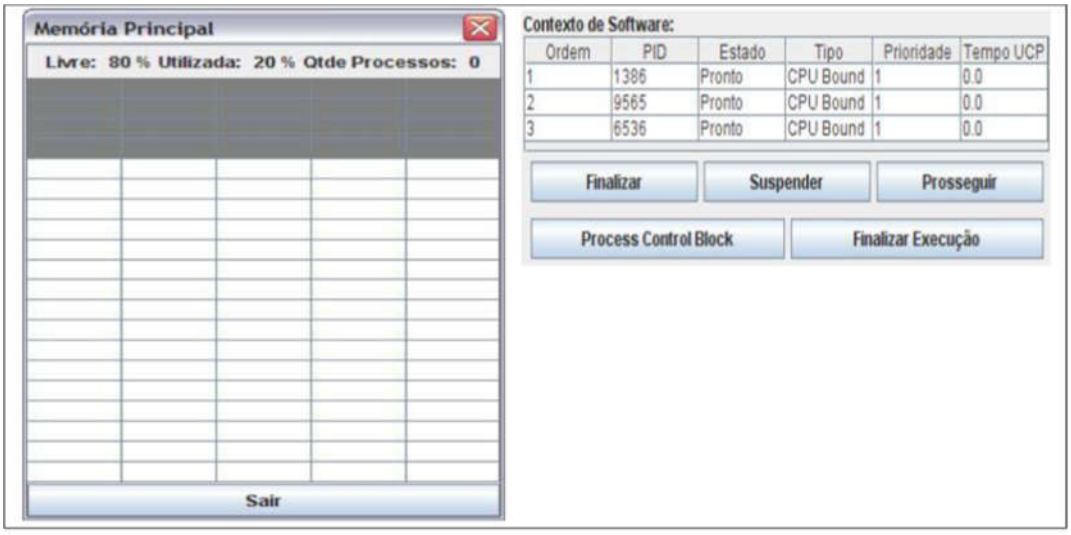


Figura 14: Interface do simulador SSOG (Fonte: Kioki e Santiago, 2008)

4.3 TBC-SO/WEB

O TBC-SO/WEB é um *software* Educacional para o ensino de políticas de escalonamento de processos e alocação de memória em SOs. Ele é um simulador de ensino-

aprendizagem via *web*[24], que visa contribuir nas aulas presenciais e/ou no ensino a distância de tópicos presentes na disciplina de SO (Reis, Parreira Jr e Costa, 2009). Ele apresenta as seguintes características:

- Interface gráfica simples.
- Simulação objetiva dos fenômenos abordados.
- Janelas e telas de ajuda explicativas para ilustrar o funcionamento do *software*.
- Animações gráficas que auxiliam no entendimento dos conceitos abordados.

No módulo de gerência de memória, apresenta em sua interface uma introdução textual explicativa da simulação, um algoritmo escrito em Portugol[25] para exemplificar a técnica de alocação de memória principal e um conjunto de quadrados que representam as posições de memória principal (*frames*).

A Figura 15 apresenta a interface do *software* TBC-SO/WEB, que distribui os processos em sua matriz representativa, sem controle e informações de alocação ou fragmentação. Em seu manual há referência à técnica de *swapping*, mas não há como identificar o funcionamento dessa técnica durante o uso do *software*.

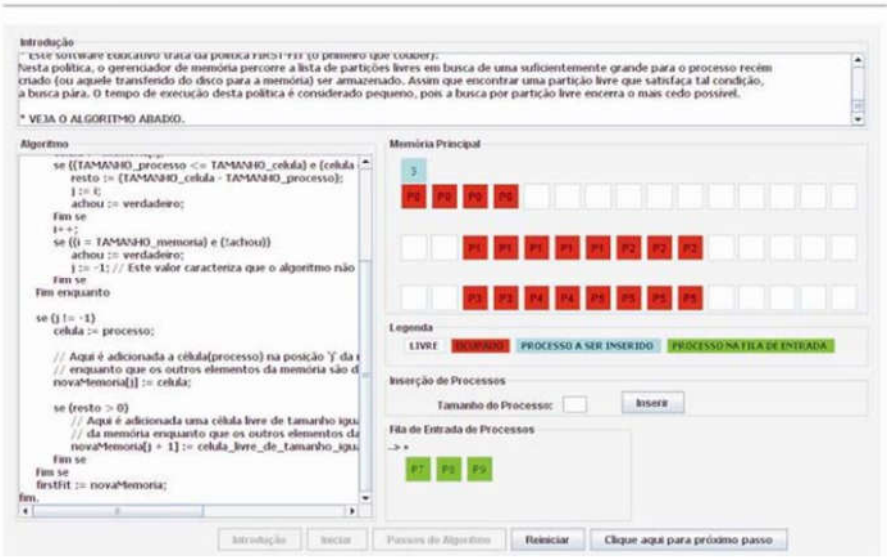


Figura 15: Interface do simulador TBC-SO/WEB (Fonte: Reis, Parreira Jr e Costa, 2009)

[24] Software *web* são *softwares* projetados para utilização a partir de navegadores com comunicação via Internet.

[25] Portugol, também conhecido como português estruturado, é um pseudocódigo escrito em português.

4.4 Simulador ESORV

O Simulador para o Ensino de SOs com RV (ESORV) é um *software* desenvolvido a partir dos princípios dos escalonadores de processos. Com suporte à tecnologia de RV, o ESORV permite que o usuário sinta-se imerso "dentro de um computador", através de um ambiente imersivo e tridimensional que simula as técnicas de escalonamento de processos (Scamati, 2017). Desenvolvido com a *Engine* Unity, a interação com o ambiente virtual pode ser realizada através de *Kinect*, *Wii Remote*, o *Keyboard* ou *Mouse*. Implementa as principais técnicas de escalonamento como o FCFS (*First in, First out*), RR (*Round-robin*) e PR (por Prioridade). Em sua interface é possível visualizar os estados de um processo (pronto, em execução e em espera) e também o Bloco de Controle do Processo, com informações como PID (*Process Identification*), Tempo de execução e Limites de Memória. A Figura 16 exibe o ambiente virtual do ESORV.

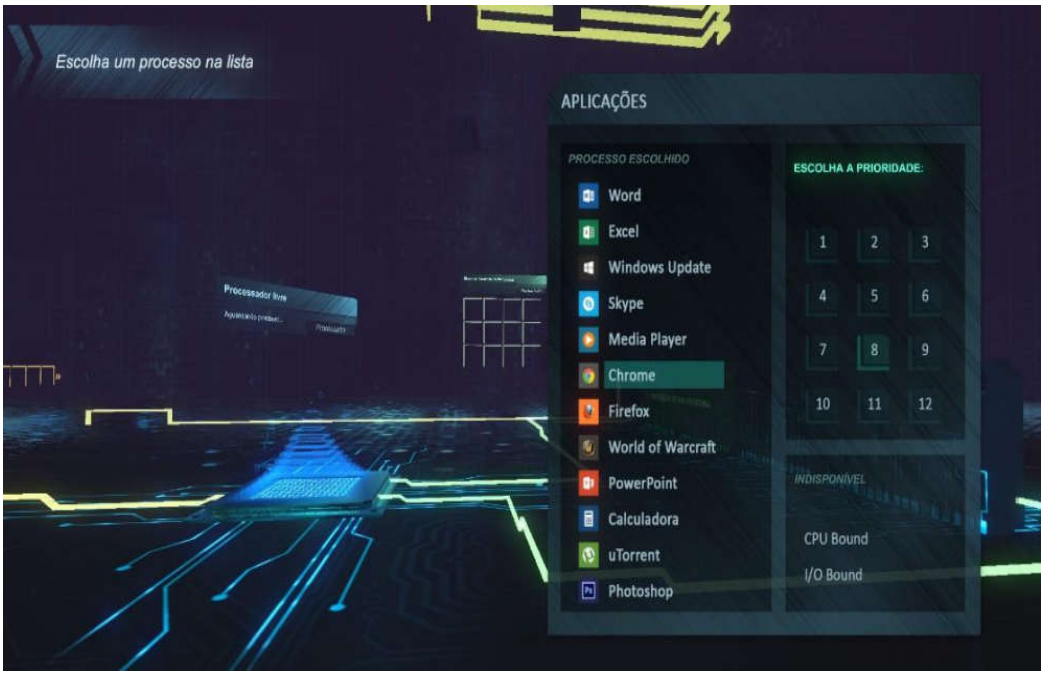


Figura 16: Ambiente virtual do Simulador ESORV (Fonte: Scamati, 2017)

4.5 Estudo Comparado

Uma vez que o *software SigemVR* tem como objetivo simular, em um ambiente 3D, o funcionamento das principais técnicas de alocação de processos na memória principal (contígua simples, particionada estática, particionada dinâmica e paginação), a Tabela 2 mostra qual(is) das técnicas listadas é(são) abordada(s) nos *softwares* já implementados e citados anteriormente, além de informar se a interface utilizada é 3D ou não. Dos softwares listados, nenhum deles tem como foco principal a simulação do módulo de gerência de memória principal, uma vez que só abordam a técnica de paginação. Com exceção do ESORV, todos os simuladores possuem interface bidimensional.

Tabela 2: Técnicas de alocação da memória principal abordadas nos simuladores analisados.

Simuladores	Simulação de gerência de memória principal – Técnicas e Características				
	Alocação Contígua Simples	Alocação Particionada Estática	Alocação Particionada Dinâmica	Alocação por Paginação	Interface 3D
SOsim	Não	Não	Não	Sim	Não
SSOG	Não	Não	Não	Sim	Não
TB-SO/WEB	Não	Não	Não	Sim	Não
ESORV	Não	Não	Não	Não	Sim

5. METODOLOGIA

A principal questão que norteia este trabalho é: "Como um simulador, com suporte à tecnologia de RV, pode auxiliar o processo de ensino-aprendizagem de tópicos abstratos da disciplina de SO?". Com base nesta questão, foi proposto explorar teorias e tecnologias de desenvolvimento de *softwares* em RV na concepção e desenvolvimento de um simulador, chamado de *SigemVR*, que pode ser usado como ferramenta auxiliar no ensino de gerência de memória principal da disciplina de SO.

Para desenvolvimento do Simulador de Gerência de Memória Principal em RV (*SigemVR*) foi adotado o método de análise e projeto apresentado por Larman (2001), o modelo incremental de desenvolvimento de *software*. Esse modelo apresenta-se como uma opção concisa e eficiente no processo de desenvolvimento de *software*, pois desde o início leva sistemicamente à produção de uma arquitetura com possibilidades de incluir novos requisitos e modificar os já existentes de um modo ordenado (Wazlawick, 2011). Tal método é dividido em quatro etapas: a análise, o projeto, a implementação e os testes. As atividades desenvolvidas em cada uma destas etapas estão representadas no fluxograma da Figura 17, onde cada atividade é representada por um número.

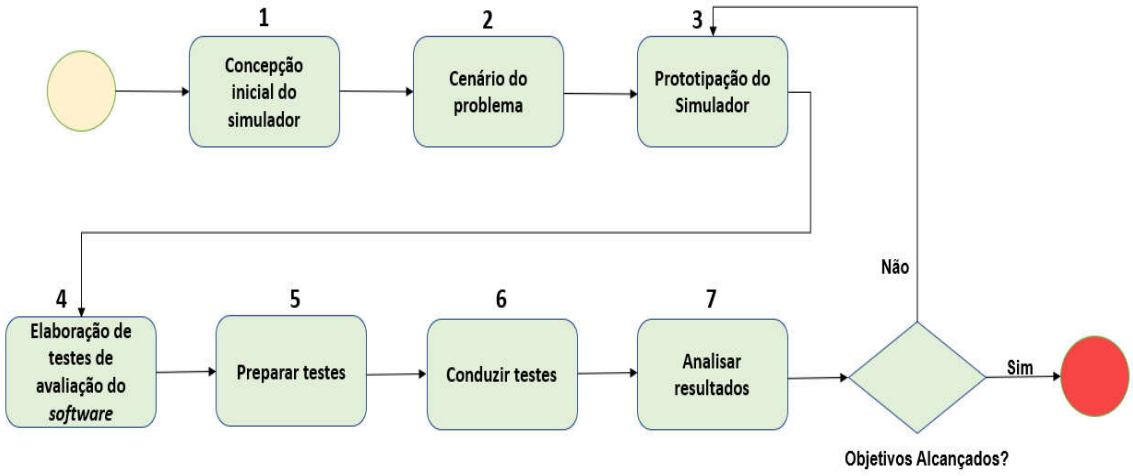


Figura 17: Fluxograma de atividades do processo de desenvolvimento do *software* SigemVR

A etapa de análise buscou definir o objetivo inicial para concepção do simulador e identificar os requisitos funcionais e não-funcionais. No presente trabalho, foi dividida em duas atividades, 1 e 2 (Figura 17). Na atividade 1, que consiste na concepção do SigemVR, foi feita

uma revisão bibliográfica do tema abordado no trabalho, o qual explora conceitos de SOs e tecnologias de RV, com ênfase dada ao módulo de gerência de memória principal e ao desenvolvimento de sistemas de RV. Uma vez definido o objetivo principal do trabalho, simular o funcionamento do módulo de gerência de memória principal, foi feita uma busca por simuladores já implementados e que têm o mesmo objetivo. As principais características, diferenças, vantagens e desvantagens de cada um dos simuladores foram analisadas.

Ainda na etapa de análise, na atividade 2 foi feita uma avaliação dos cenários específicos para a pesquisa e experimentação, permitindo o levantamento dos principais problemas e dificuldades encontrados. Neste sentido, a necessidade de simuladores, que podem ser usados como ferramentas de ensino, é uma realidade da grande maioria dos professores da disciplina de SO, uma vez que ela aborda conceitos difíceis de serem assimilados pelos alunos. Um destes conceitos, é o módulo de gerência de memória principal, o qual considera técnicas de alocação de processos na memória principal. Nesta atividade, um questionário (Apêndice I) foi aplicado a um conjunto de professores da disciplina de SO, de diferentes instituições, com o intuito de avaliar as dificuldades encontradas pelos mesmos nesta disciplina. Nesse cenário foi concebida a ideia de desenvolvimento do *SigemVR*.

A atividade 3 engloba as etapas de projeto e implementação. Na etapa de projeto, foram definidas a arquitetura do *SigemVR* e as ferramentas utilizadas no desenvolvimento, como: a *Engine Unity* e a linguagem de programação C# (Tópicos abordados com maiores detalhes na Seção 6.3.2 e 6.3.3). Na implementação o *software* foi de fato concebido. A partir de características específicas das linguagens de programação, ferramentas, *frameworks* e estruturas de dados, o *software* passou a se apresentar como um produto palpável.

A etapa de testes foi dividida nas atividades 4, 5 e 6. Na atividade 4, foram criados os questionários avaliativos dos conhecimentos adquiridos em gerência de memória principal e dos recursos tecnológicos apresentados pelo simulador. Esses questionários podem ser vistos nos Apêndices II e III e foram usados para avaliar a eficácia do simulador como ferramenta de ensino e os recursos tecnológicos apresentados durante a experiência de uso do *software*. Na atividade 5, foram definidos o local, o número de alunos e os recursos disponíveis para realização dos testes elaborados anteriormente. Na atividade 6, que finaliza a etapa de testes, cento e vinte estudantes do curso de informática, tanto do nível superior quanto do nível técnico, do Instituto Federal de Mato Grosso do Sul (IFMS) – Campus Corumbá foram divididos em três grupos iguais. Cada grupo participou de uma aula, na qual foram considerados três métodos

de ensino distintos: a) Método de ensino tradicional, b) Método de ensino auxiliado pelo simulador executado em *desktop* e c) Método de ensino auxiliado pelo simulador executado em HMD. Ao final desta atividade, os estudantes responderam aos questionários elaborados na atividade 4.

Na atividade 7, os resultados obtidos para cada um dos métodos utilizados na atividade anterior foram analisados e comparados. Para isso, foram utilizados os métodos de análise de variância: ANOVA e o teste de *Tukey*, abordados com maiores detalhes na Seção 7.3.2. Tais métodos permitiram avaliar a eficácia do uso do simulador *SigemVR* como recurso auxiliar de ensino.

6. CONCEPÇÃO E DESENVOLVIMENTO DO SIMULADOR DE GERÊNCIA DE MEMÓRIA PRINCIPAL EM REALIDADE VIRTUAL – SIGEMVR

O presente capítulo é dedicado ao detalhamento da aplicação da metodologia descrita no capítulo anterior na concepção e desenvolvimento do simulador *SigemVR*. A Figura 18 exibe as etapas realizadas no desenvolvimento do *SigemVR*, bem como as atividades realizadas dentro de cada uma dessas etapas.

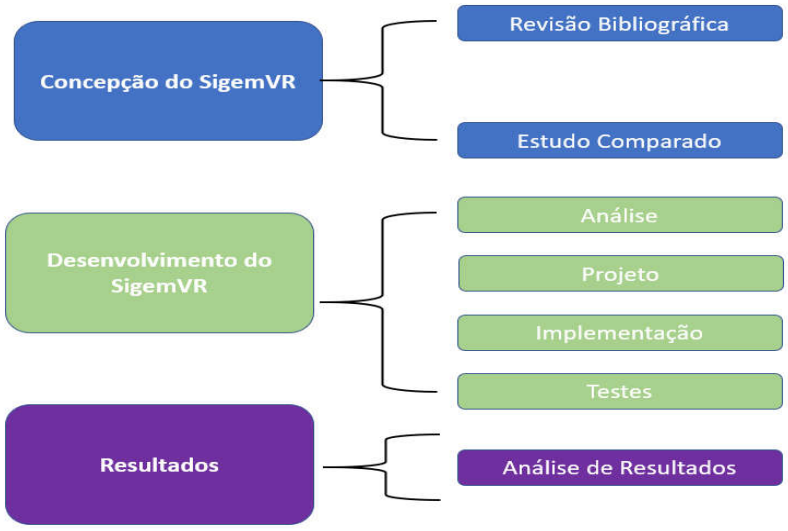


Figura 18: Etapas do desenvolvimento do SigemVR

O processo de concepção do SigemVR foi detalhado nos capítulos 2, 3 e 4. Neste capítulo, a Seção 6.1 é dedicada à etapa de análise. Nas Seções 6.2 e 6.3 são descritas, respectivamente, as etapas de projeto e implementação, enquanto que a Seção 6.4 apresenta o detalhamento dos testes aplicados aos usuários do simulador. A análise dos resultados é apresentada no Capítulo 7.

6.1 Análise

Na etapa de análise do processo de desenvolvimento do *SigemVR,* foi necessário identificar os métodos de ensino utilizados por um grupo de professores da disciplina de SO, de diferentes instituições, bem como o perfil dos docentes envolvidos. Para tal objetivo, foi definido como método, para coleta de dados, um questionário (Apêndice I), aplicado ao

conjunto de professores. As informações coletadas auxiliaram na investigação do contexto e da motivação para a concepção do SigemVR. Os dados obtidos nesta fase foram divididos em três partes: a) Perfil dos Entrevistados, apresentados na sub-seção 6.1.1; b) Conhecimento sobre RV, apresentados na sub-seção 6.1.2; c) Enunciado do problema e identificação de requisitos, apresentados na sub-seção 6.1.3.

6.1.1 Perfil dos entrevistados

Responderam a um questionário dez professores, tanto do nível técnico quanto do superior, ministrantes da disciplina de SO em diferentes instituições de ensino. A Figura 19 apresenta os dados referentes à formação acadêmica dos entrevistados. Embora seja possível verificar uma variedade de cursos de graduação, todos afirmaram ter feito a disciplina de SO durante o processo de formação.

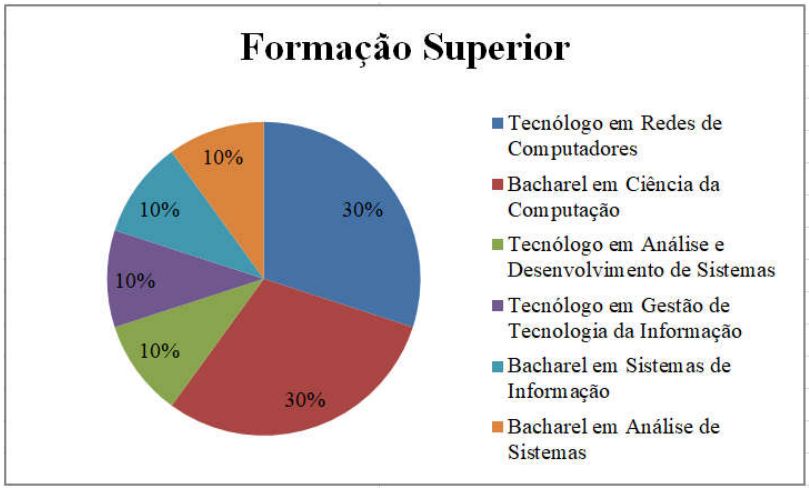


Figura 19: Formação acadêmica dos professores entrevistados

Os docentes entrevistados pertencem a quatro instituições de ensino, sendo duas públicas, Instituto Federal de Mato Grosso do Sul (IFMS) e Instituto Federal do Pará (IFPA), e duas privadas, Serviço Nacional de Aprendizagem Comercial do Estado do Mato Grosso do Sul (Senac/MS) e Universidade para o Desenvolvimento do Estado e da Região do Pantanal (Uniderp Anhanguera), como mostra a Figura 20.

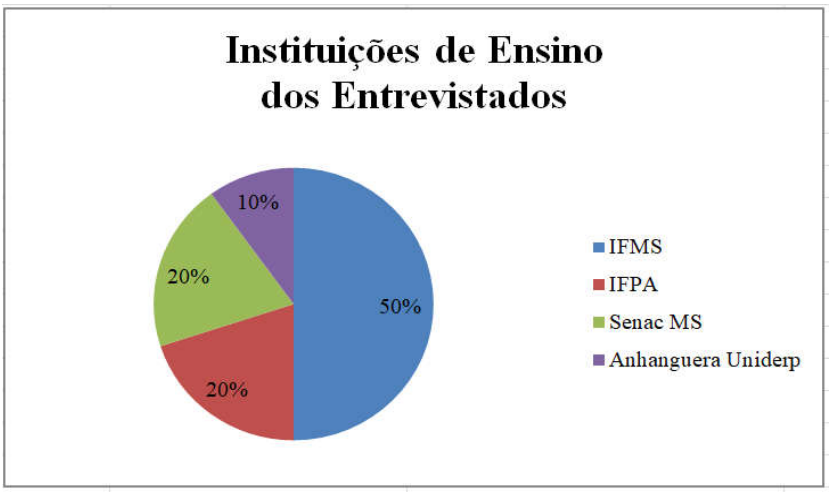


Figura 20: Instituição de ensino dos professores entrevistados

Todos os participantes afirmaram possuir formação acadêmica complementar à graduação, uma vez que sete deles são especialistas e três são mestres na área de informática. A Figura 21 ilustra o tempo, em anos, que os docentes lecionam a disciplina de SO.

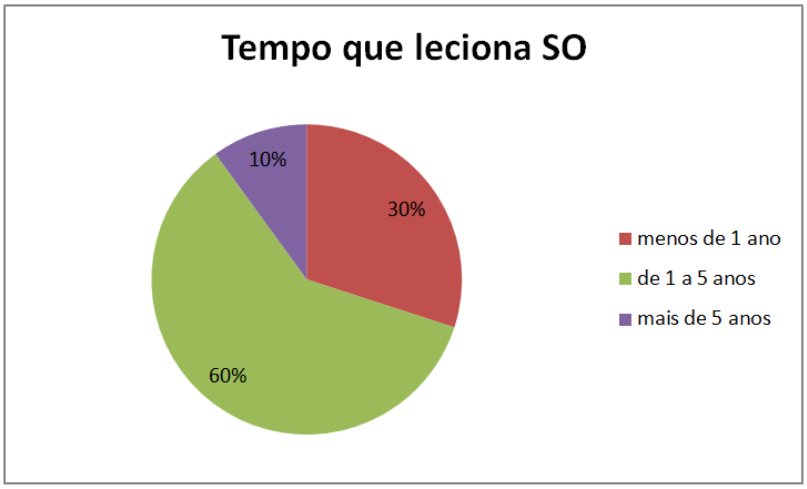


Figura 21: Tempo que os professores entrevistados lecionam a disciplina de Sistemas Operacionais

A Figura 22 apresenta os níveis de ensino aos quais os docentes atuam diretamente no ensino da disciplina de SOs. É importante considerar que um mesmo professor pode lecionar em mais de um nível de ensino.

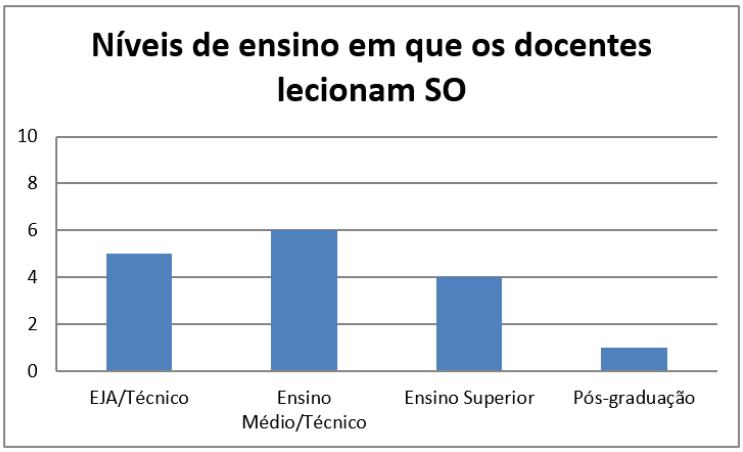


Figura 22: Níveis de ensino em que os docentes entrevistados lecionam a disciplina de Sistemas Operacionais

Quanto aos recursos didáticos utilizados no ensino da disciplina de SOs, todos relataram o uso de projetores multimídia e lousa, ou seja, utilizam um método tradicional de ensino. No entanto, seis dos entrevistados utilizam recursos adicionais ao método tradicional, como: todos utilizam material didático impresso e três utilizam aulas práticas em laboratórios de informática, dos quais dois utilizam SOs de código aberto, como o MINIX e simuladores, enquanto um participante faz uso de técnicas de desenvolvimento de rotinas em sala de aula. Com relação às dificuldades encontradas na execução dessas aulas práticas, o fator mais mencionado foi a falta de material didático que aborde, na prática, alguns conceitos da disciplina, como mostra a Figura 23.

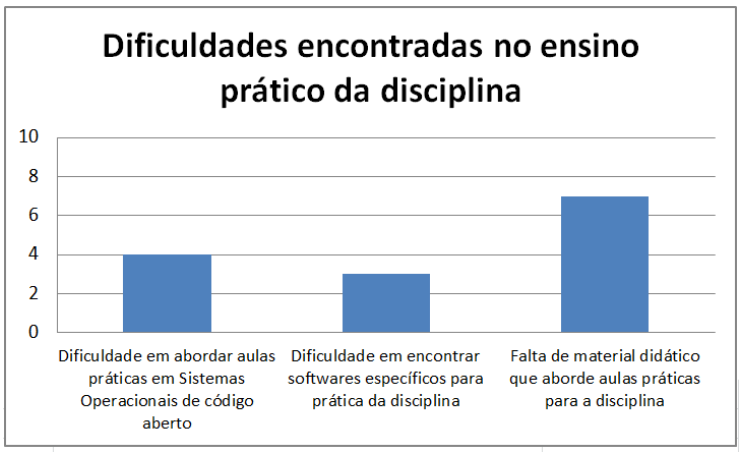


Figura 23: Dificuldades encontradas no ensino prático da disciplina de Sistemas Operacionais

Dentre os simuladores já implementados (apresentados nas Seções 4.1, 4.2, 4.3 e 4.4), sete entrevistados relataram conhecer o simulador SOsim (Maia, 2001). Apenas dois deles utilizam ou já utilizaram esse *software* em sala de aula. Nenhum dos entrevistados alegou conhecer os simuladores SSOG, TB-SO/Web e ESORV.

6.1.2 Uso de RV

O SigemVR é um simulador com suporte à tecnologia de RV. Nesse sentido, quatro questões aplicadas aos professores (Apêndice I) foram direcionadas com o intuito de identificar o grau de conhecimento dos mesmos sobre o tema. Dos dez entrevistados, seis afirmaram que já utilizaram algum *software* em RV, dos quais quatro fizeram uso de capacetes de RV (HMD) durante a experiência, enquanto dois utilizaram computadores convencionais. A maioria dos entrevistados mencionaram a interface e a usabilidade como principais vantagens dos sistemas em RV, enquanto que o custo e o desempenho são relatados como desvantagens nesses sistemas. Esses dados são mostrados na Figura 24.

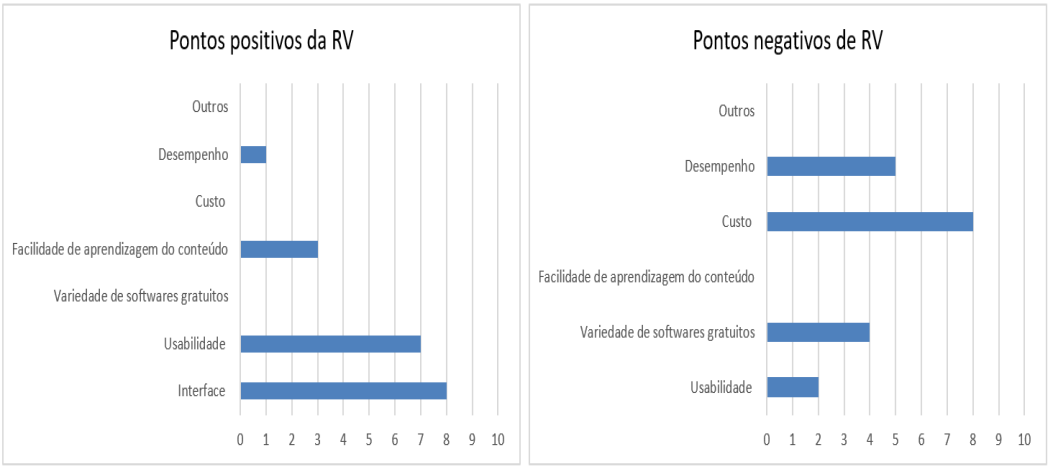


Figura 24: Vantagens e desvantagens dos sistemas de RV identificados pelos docentes entrevistados. Os números apresentados representam o número de professores entrevistados.

6.1.3 Enunciado do problema e identificação de requisitos

Os resultados anteriores mostraram que todos os professores entrevistados preferencialmente lecionam a disciplina de SO utilizando métodos tradicionais de ensino, ou seja, aulas teóricas com uso de lousa e projetor multimídia. No entanto, todos reconheceram que o uso de simuladores favorece a aplicação prática dos conceitos, tornando o processo ensino-aprendizagem mais fácil e prazeroso. Neste contexto, dentre os simuladores já existentes (Capítulo 4), nenhum deles aborda as diferentes técnicas de alocação de processos na memória principal (tratadas nas Seções 2.6.1 a 2.6.6), assunto de extrema importância no módulo de gerência de memória em SOs. Nesse cenário foi concebida a ideia de desenvolvimento do *SigemVR.*

Com relação aos requisitos de um sistema, os mesmos podem ser classificados em duas grandes categorias: a) Os requisitos funcionais, que correspondem à listagem do que o sistema deve fazer; b) Os requisitos não-funcionais, que são as restrições colocadas sobre como o sistema deve realizar seus requisitos funcionais. Na concepção do *SigemVR* e a partir das necessidades levantadas pelos professores submetidos ao questionário de análise do *software* foram identificados os seguintes conjuntos de requisitos:

a) Requisitos funcionais:

- O sistema deve ter um Menu Inicial com as opções de técnicas de alocação de memória disponíveis para simulação. O usuário deve ter a liberdade de escolher qualquer uma das opções, basta selecionar a técnica desejada.
- O SigemVR deve suportar a simulação das seguintes técnicas de alocação de memória: alocação contígua simples, alocação particionada estática, alocação particionada dinâmica e alocação por paginação.
- Cada simulação deve possuir cinco opções de processos que o usuário pode alocar na memória, incluindo o próprio SO. Há duas regras para alocar um novo processo em RAM: (1) O SO deve ser inicializado; (2) Para que um processo seja alocado, é necessário que haja memória disponível e compatível à sua necessidade. Caso contrário, uma mensagem de áudio deve informar tal restrição ao usuário, permitindo uma experiência multissensorial.
- A BCP, deve apresentar as principais informações pertinentes ao processo criado, caso este seja alocado na memória.

- A memória é representada graficamente por um conjunto de elementos instanciados na forma de cubos tridimensionais. Cada cubo deve representar uma partição, ou *frame,* da memória. Quando o usuário alocar um processo na memória, a partição ou *frame* destinado a ele deve alterar sua cor.
- O usuário poderá finalizar qualquer um dos processos alocados na memória. Quando isso acontecer, a memória destinada ao processo encerrado ficará disponível para os demais processos.
- Caso o SO seja finalizado, todos os demais processos na memória RAM serão também encerrados.

b) Requisitos não-funcionais:

- Cada técnica de alocação deverá ser apresentada isoladamente.
- O sistema deverá incluir áudios ou outras mídias que facilitem o entendimento da simulação, de modo que a experiência seja multissensorial.
- O ambiente virtual deverá proporcionar ao usuário a sensação de "estar dentro de um computador", garantindo a imersão.
- A execução do simulador deve suportar as plataformas *desktop* e *HMD* e, independentemente da plataforma escolhida, o sistema deve ser intuitivo, utilizando *joystick* ou mouse como dispositivo de interação.

6.2 Projeto

A etapa de projeto define todo o planejamento anterior ao desenvolvimento do simulador. É nesta fase que são definidos a arquitetura do simulador e a expansão dos casos de uso do mesmo. A Figura 25 mostra a arquitetura criada para o projeto do *SigemVR.* Com base na arquitetura definida, o *SigemVR* foi desenvolvido utilizando o motor de jogos (*engine) Unity,* em conjunto com a linguagem de programação C# (*Sharp*), usada na edição de *scripts*. A partir dessas ferramentas foram definidos o ambiente virtual e os equipamentos de interação, que pode ser via *mouse*, *joystick* ou *gaze*. O ambiente virtual possui elementos gráficos que representam os principais *hardwares* de um computador, como CPU, memória RAM e HD. O pacote de desenvolvimento *Unity* permite criar aplicações para as plataformas HMD e *Desktop*. Na

presente versão do simulador, as técnicas de alocação de memória suportadas são: contígua simples, particionada estática, particionada dinâmica e alocação por paginação, mas outras técnicas podem ser incluídas como, por exemplo segmentação (Seção 2.6.5) e segmentação por paginação (Seção 2.6.6), as quais serão realizadas em trabalhos futuros.

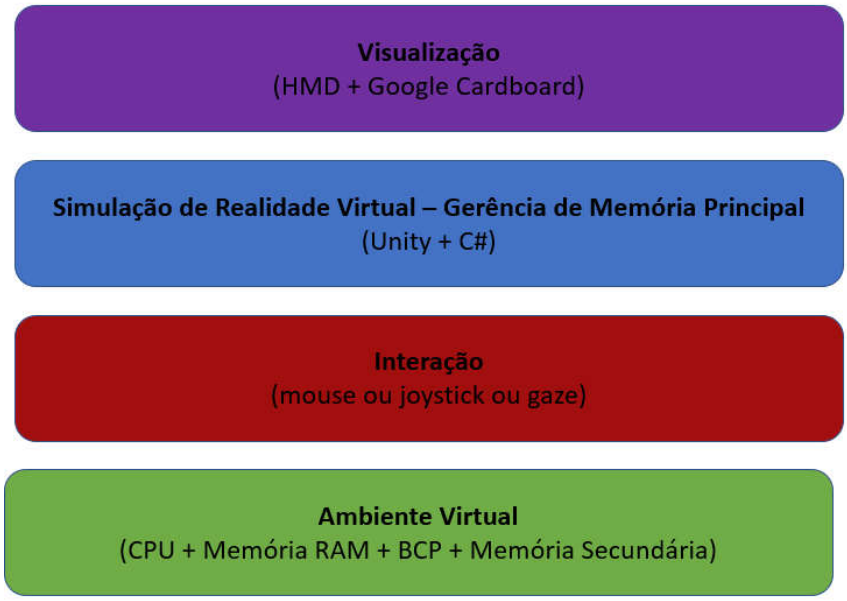


Figura 25: Arquitetura do SigemVR

Após a definição da arquitetura do simulador, foi descrito "o quê" acontece entre o usuário e o sistema, sem, entretanto, informar "como" essa interação ocorre. É o que se chama de diagrama de casos de uso, no qual são descritas as essências das operações realizadas, independente da tecnologia ou interface. O diagrama de Casos de Uso do *SigemVR* apresenta seis casos de uso, detalhados a seguir:

1. Criar novos processos: O usuário pode criar diferentes opções de processos durante a simulação. Para isso, é necessário que o SO seja inicializado e que exista memória principal disponível e compatível com a necessidade desse processo;
2. Finalizar processos: O usuário pode finalizar os processos alocados na memória principal, de acordo com a sua vontade e, quando isso ocorrer, a memória dedicada a esses processos devem ser desocupadas e disponibilizadas a novos processos;
3. Navegar no ambiente: O usuário pode navegar no ambiente virtual de maneira exploratória, visualizando os principais *hardwares* de um sistema computacional;

4. Visualizar a BCP: Durante a simulação, o usuário deve visualizar a BCP com informações sobre os processos residentes na memória principal;
5. Solicitar Ajuda: O usuário pode solicitar ajuda sobre o funcionamento do simulador e, quando isso for feito, um vídeo explicativo deverá ser apresentado;
6. Concluir a simulação: o usuário pode finalizar a simulação, retornando ao *menu* principal do simulador.

A partir dos casos de uso descritos acima, o diagrama UML[26] (*Unified Modeling Language*) foi definido e é mostrado na Figura 26.

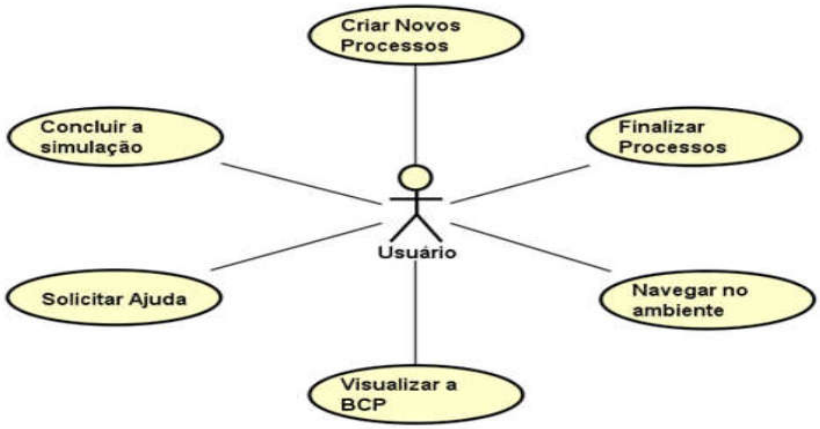


Figura 26: Diagrama de casos de uso do SigemVR

6.3 Implementação

Na fase de implementação o *software* é concebido, ou seja, ele é construído de acordo com as análises e definições anteriores. A seção 6.3.1 apresenta um fluxograma detalhado do funcionamento do SigemVR. O motor de jogos *Unity*, ferramenta utilizada no desenvolvimento do *SigemVR*, é detalhado na Seção 6.3.2. As Seções 6.3.3, 6.3.4 e 6.3.5 são dedicadas, respectivamente, à linguagem de programação C#, à construção do ambiente virtual e aos equipamentos de interação com o sistema.

[26] UML (Linguagem de modelagem unificada) é uma linguagem utilizada na elaboração da estrutura de projetos de *softwares*.

6.3.1 Fluxograma de funcionamento do SigemVR

A Figura 27 exibe o fluxograma de funcionamento do simulador *SigemVR*. Quando o *SigemVR* é inicializado é possível escolher dentre as quatro técnicas simuladas ou a opção de sair do simulador (Figura 28). Ao selecionar uma das técnicas de alocação de memória simulada é possível interagir com o simulador através dos casos de uso definidos anteriormente (Figura 26).

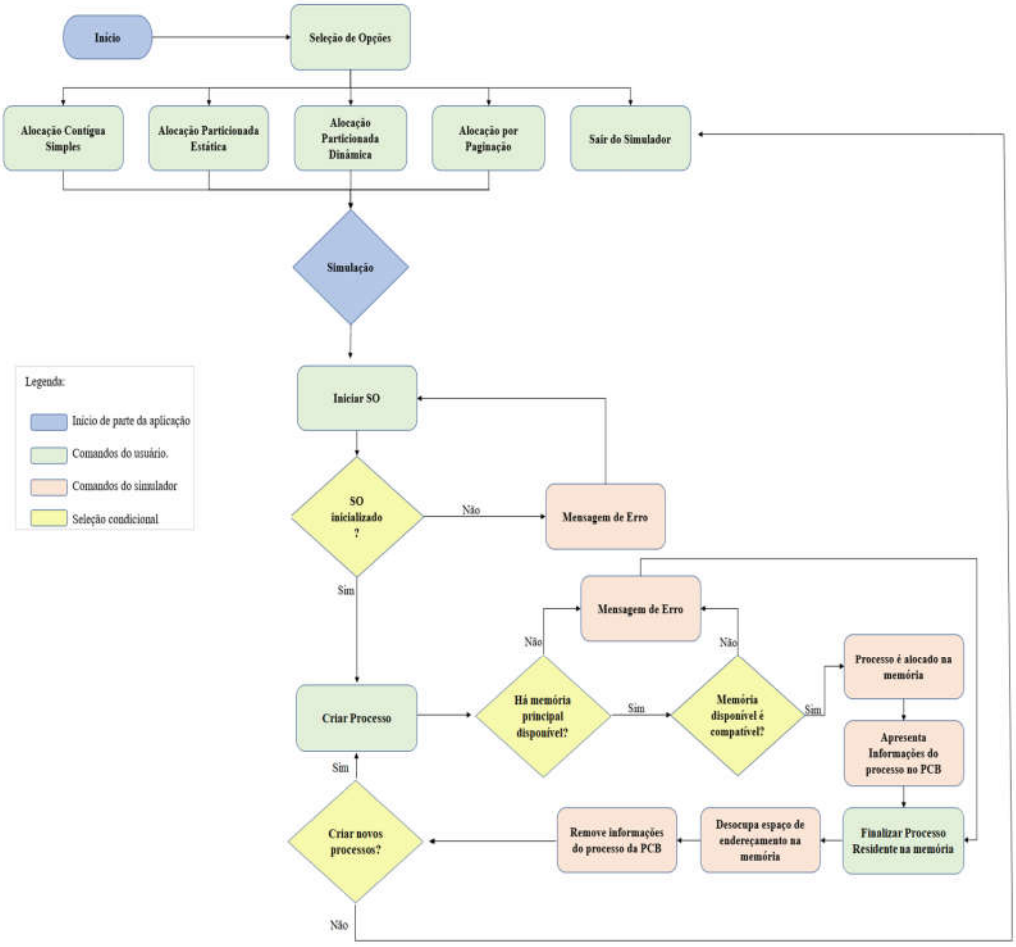


Figura 27: Fluxograma de funcionamento do simulador SigemVR

A Figura 28 mostra o menu inicial do SigemVR, no qual são apresentadas ao usuário as opções das diferentes técnicas simuladas, assim como a opção de sair da aplicação.

Figura 28: Menu inicial do Simulador SigemVR

Através de um comando (INPUT[27]), enviado via *joystick* ou *mouse*, o usuário seleciona uma das opções apresentadas no menu inicial. Em todas as técnicas simuladas as regras para alocar novos processos na memória principal são as mesmas: a) o SO deve ser inicializado; b) deve existir memória disponível e compatível com a necessidade do novo processo. Caso o um desses pré-requisitos não seja atendido, uma mensagem de áudio é executada, informando ao usuário a restrição. Nos casos em que não há memória livre ou compatível com o processo, o usuário tem a opção de finalizar os processos já residentes na memória principal. Cada técnica possui métodos próprios para alocar processos na memória, os quais foram discutidos na Seção 2.6.

6.3.2 *Unity*

O alto custo de produção de simulações computacionais foi contexto do surgimento dos motores de jogos em 1999, possibilitando um desenvolvimento de ambientes virtuais em alto nível, possibilitando mais tempo para que os desenvolvedores se preocupem com recursos como animação, comportamento e narrativa da aplicação (Lewis e Jacobson, 2002).

[27] *INPUT* é o temo usado na computação para indicar entrada de dados.

O *Unity* é uma plataforma de desenvolvimento de jogos, também chamada de motor de jogos, criada e distribuída pela empresa *Unity Technologies*. Com esse *software*, é possível construir jogos para dispositivos móveis e *desktops,* com suporte à tecnologia de RV. Segundo a SourceDNA[28] (2016), em 2016, foram registrados mais de cinco bilhões de *downloads* de jogos construídos com o *Unity* e, no mesmo ano, cerca de 34% dos jogos para dispositivos móveis presentes no mercado foram desenvolvidos a partir dessa plataforma. Essa plataforma oferece um ambiente gráfico de desenvolvimento e um editor de *scripts*, de modo que podem ser criados projetos para uma grande variedade de mídias ou dispositivos, como consoles, *web* ou HMDs.

A Figura 29 exibe a execução do SigemVR através do uso da biblioteca *Google CardBoard*. Essa biblioteca, desenvolvida pelo *Google*, permite configurar um visualizador de RV, baseado em estereoscopia, diretamente em um *smartphone*, além de fornecer um conjunto de bibliotecas de programação que permitem a implementação de novos recursos de RV.

Figura 29: SigemVR em execução através do uso da biblioteca *Google Cardboard*

[28] SourceDNA é uma empresa norte-americana de segurança responsável por oferecer soluções de assistência a desenvolvedores de *software*.

6.3.3 Linguagem de programação C#

O C#, também chamado de *C Sharp*, é uma linguagem de programação desenvolvida pela *Microsoft* como parte da plataforma .Net[29]. O desenvolvimento de sistemas no *Unity*, associado a esta linguagem, permite criar *scripts* que automatizam a execução de tarefas no ambiente virtual, além de permitir associar comportamentos aos elementos presentes no mundo virtual. Os *scripts* também fornecem recursos ao *software* em desenvolvimento, como recursos de interface gráfica e interação. A Tabela 3 apresenta, em ordem alfabética, alguns dos principais recursos da linguagem C# utilizados no desenvolvimento do *SigemVR*, assim como suas respectivas descrições.

Tabela 3: Recursos da linguagem C# utilizados no desenvolvimento do SigemVR

AudioSource	É um tipo de objeto de jogo que permite adicionar sons no ambiente virtual 3D. Através desse componente foi possível adicionar mensagens de áudio informativas durante a simulação.
BoxCollider	Recurso que permite associar um "colisor" a um componente no ambiente virtual. Usado nos elementos de interação, como nos botões e na representação gráfica da memória RAM.
EventTrigger	É usado para associar funções que são chamadas para cada evento identificado pelo *EventSystem* do sistema. Com esse recurso, foi possível criar eventos programados para as ações do usuário no ambiente virtual.
GameObject	Permite instanciar um elemento da cena como um objeto de jogo, no qual podem ser feitas alterações nos seus principais componentes, como: escala, forma, cor, entre outros.
GvrEditorEmulator	É um *script* associado à câmera principal da cena, fornecido no conjunto de recursos do *Google VR SDK*, que permite emular os movimentos da cabeça do usuário e a entrada (*INPUT*) de suas interações.
GvrPointerPhysics Raycaster	É um *script* associado à câmera principal da cena, fornecido no conjunto de recursos do *Google VR SDK*. Esse recurso associa um campo de visão ao qual o usuário pode interagir através do "olhar" com os elementos em cena.
GvrReticlePointer	É um *script* associado à câmera principal da cena, fornecido no conjunto de recursos do *Google VR SDK*. Ele cria um ponto gráfico no campo de observação do usuário, informando qual elemento ele está observando.
PointerClick	É o tipo de evento que associa um clique a um determinado comportamento de um elemento na cena, definido por *script*.
Unity.Engine.UI	É uma biblioteca da linguagem que permite criar interfaces gráficas de interação com o usuário, como botões, imagens e textos.

[29] .Net é uma iniciativa da empresa *Microsoft*, que visa obter uma plataforma única para desenvolvimento e execução de sistemas e aplicações.

6.3.4 Ambiente Virtual

O ambiente virtual do *SigemVR* simula o interior de um computador. Há representações gráficas do processador, memória RAM e disco rígido. A Figura 30 ilustra os componentes de *hardware* do ambiente.

Figura 30: Ambiente Virtual do SigemVR, onde podem ser vistos componentes de *hardware* como memória RAM e CPU

Um BCP virtual apresenta as seguintes informações dos processos alocados na memória principal:

- ID_Processo: identificador único de um processo.
- NomeProcesso: apresenta o nome do programa em execução.
- TamanhoProcesso: indica a quantidade de memória principal necessária para execução do processo.
- PosiçãoOcupada / PosicaoRAM: usada para identificar as partições ou *frames* alocados para o processo vigente.
- PosicaoHD: apresentado somente na técnica de alocação por paginação, indica graficamente os *frames* ocupados pelo processo na memória secundária (HD).
- tamanhoMemoria / NumeroFrames: indica o tamanho das partições ou *frames* disponíveis para alocação do novo processo.
- Fragmentação: indica a subtração do tamanho da memória ou do *frame* com o necessário para alocação do novo processo (tamanhoProcesso), caso a memória disponível seja compatível com a necessidade do novo processo.
- FinalizarProcesso: botão usado para finalizar um processo já alocado na memória principal. Caso esse processo finalizado seja o SO, todos os demais processos residentes na memória principal também são encerrados.

A Figura 31 apresenta o BCP presente no ambiente virtual durante a execução do simulador em um *desktop*.

Figura 31: Bloco de Controle do Processo (BCP) presente no ambiente virtual do SigemVR

6.3.5 Equipamentos de interação

No simulador SigemVR, a interação do usuário com os elementos em cena depende de dois componentes: o olhar e o *Input*. Para interagir com um elemento do ambiente, o usuário precisa olhar na direção dele e usar qualquer tipo de entrada de dados com o sistema, que pode ser um clique através do mouse ou *joystick,* ou até mesmo um toque no *display* do aparelho quando *smartphones* são utilizados. Isso é possível, pois o pacote de ferramentas *Google VR SDK for Unity with Android*, oferece um conjunto de *scrips* e *prefabs* que permitem a visualização e a interação quando a tecnologia *CardBoard* é usada.

Para incluir suporte de *input* no SigemVR, os seguintes *prefabs*[30] e *scripts* foram utilizados:

- *GvrControllerMain*: Esse *prefab* é responsável por gerenciar e incluir componentes de *input* em cada ponto de interação na cena;
- *GvrControllerPointer*: Um ponto gráfico na tela anima-se ao "olhar" para um item de interação;
- *GvrEventSystem*: Inclui como sistema de eventos do *Unity* o componente *GvrPointer*;

[30] *Prefabs* são objetos previamente criados e reutilizáveis.

- *GvrEditorEmulator*: É o controlador da câmera durante a execução do *software*. Permite simular o movimento da cabeça do usuário e adequar a visualização do ambiente de acordo com esse ponto de vista;
- *GvrPointerPhysicsRaycaster*: Define como campo de interação o raio do *GvrPointer*. Dessa forma, o usuário olha para o elemento que deseja interagir e confirma seu uso com um clique;

7. TESTES E RESULTADOS

Uma vez concebido o produto de *software,* é necessário validar se ele atende às necessidades do usuário final (Wazlawick, 2011). Para isso, o simulador *SigemVR* foi submetido a testes funcionais, na forma de questionários, com cento e vinte (120) estudantes do curso superior em Análise e Desenvolvimento de Sistemas e do curso técnico em Informática de nível médio, do Instituto Federal de Mato Grosso do Sul – IFMS.

A Seção 7.1 apresenta os tipos de testes aplicados e a abordagem usada em sua execução. A Seção 7.2 é usada para demonstrar o cenário e os participantes dos testes. A Seção 7.3 apresenta os resultados obtidos a partir dos testes realizados, enquanto a Seção 7.4 é dedicada às discussões dos resultados obtidos.

7.1 Tipos de testes

Foram aplicados dois tipos de testes (questionários): a) Questionário de conhecimentos em SO (Apêndice II) e b) Questionário de recursos tecnológicos do *SigemVR* (Apêndice III). O teste de conhecimentos em SO tem o objetivo de identificar a eficiência do uso do simulador no auxílio do processo ensino-aprendizagem da disciplina de SO. Já o teste de recursos tecnológicos é usado para avaliar a interface, usabilidade e demais recursos tecnológicos do SigemVR.

7.1.1 Questionário de conhecimento em SO

Esse teste foi aplicado em três grupos de estudantes que foram divididos em três diferentes abordagens de ensino, como apresentado na Tabela 4. Os estudantes responderam ao teste logo após a execução da aula, realizada de acordo com a abordagem de ensino do grupo considerado. Em anexo ao questionário de conhecimentos em SO, foi aplicado um pré-teste que visou identificar o perfil do estudante participante. Após a aplicação dos testes, os resultados obtidos foram usados para comparar a eficácia de cada um dos métodos de ensino considerados.

Tabela 4: Abordagens de ensino consideradas na aplicação do questionário de conhecimentos em SO

Abordagem de Ensino:	**Objetivo:**	**Número de participantes:**
1. Método tradicional de ensino	Apresentar as principais técnicas de alocação de memória principal com uso de quadro negro e projetor multimídia e criar uma base de resposta ao questionário de Conhecimento em SO (apêndice II) para comparação com demais métodos.	40 estudantes
2. Método de ensino com auxílio do *SigemVR* em *desktop*	Apresentar as principais técnicas de alocação de memória principal com uso do simulador SigemVR executado em *desktop* e criar uma base de resposta ao questionário de Conhecimento em SO (Apêndice II) para comparação com demais métodos.	40 estudantes
3. Método de ensino com auxílio do SigemVR em *HMD*	Apresentar as principais técnicas de alocação de memória principal com uso do simulador SigemVR executado em HMD e criar uma base de resposta ao questionário de Conhecimento em SO (Apêndice II) para comparação com demais métodos.	40 estudantes

Os principais tópicos avaliados no teste de conhecimento em SO estão relacionados diretamente ao módulo de gerência de memória principal da disciplina. A Tabela 5 apresenta os tópicos abordados.

Tabela 5: Tópicos avaliados no teste de conhecimentos em SO

Tópico do conteúdo:	Objetivo:
Processo	Avaliar a definição de processo em SO e sua implementação.
Espaço de endereçamento	Identificar o controle dos recursos de *hardware* destinados a um processo e avaliar as definições de partições ou *frames* utilizados nas técnicas de alocação de memória principal.
Alocação de processos em memória	Definir a distribuição de processos na memória RAM de acordo com a técnica simulada.
Fragmentação de memória	Compreender como as técnicas de alocação de memória controlam o acesso a este recurso na tentativa de evitar desperdícios.
Técnicas de alocação	Identificar as características e as regras utilizadas nas diferentes técnicas de alocação de memória simuladas.
Inicialização de processos	Definir as regras para a inicialização e a alocação de processos na memória.

7.1.2 Questionário de recursos tecnológicos do SigemVR

O teste de recursos tecnológicos foi aplicado em dois grupos de estudantes, divididos de acordo com a plataforma em que o simulador *SigemVR* foi executado durante a aula. A Tabela 6 apresenta os principais tópicos avaliados neste teste. A escala *likert*[31] (Likert, 1932) foi utilizada na avaliação dos dados obtidos.

[31] Escala *likert* é um tipo de escala de resposta psicométrica usada habitualmente em questionários e, é também, uma das mais usadas em pesquisas de opinião.

Tabela 6: Tópicos avaliados no teste de recursos tecnológicos do SigemVR

Tópico do conteúdo	Objetivo
Interface gráfica	Avaliar o ambiente virtual, como o conforto e a facilidade de visualização de seus elementos.
Usabilidade	Definir as melhores formas de interação com o simulador, a partir da plataforma de execução utilizada.
Intuição	Avaliar se o simulador é intuitivo para o usuário.
Tempo de resposta	Verificar a performance do simulador durante a simulação, independentemente da plataforma utilizada para sua execução.
Mensagens de áudio	Avaliar o quão explicativas e multissensoriais são as mensagens de áudio presentes na simulação.
Representações gráficas	Avaliar a verossimilhança dos elementos gráficos presentes no ambiente virtual com o mundo real.
Desconforto em RV	Identificar possíveis desconfortos na utilização do simulador.
Imersão	Avaliar o grau de sentimento do usuário em "estar dentro de um computador".
Contribuição no ensino	Verificar se o simulador facilita o entendimento do conteúdo estudado.

7.2 Cenário, Participantes e Metodologia Aplicadas

Os testes foram aplicados em quatro turmas do IFMS - Campus Corumbá, sendo três do curso técnico de nível médio em Informática e uma do curso superior de Tecnologia em Análise e Desenvolvimento de Sistemas, totalizando 120 participantes. A Tabela 7 apresenta

as informações do *setup* dos testes, o cenário de sua execução, os professores responsáveis em cada uma das abordagens de ensino e os recursos tecnológicos utilizados.

Tabela 7: Cenários, professores responsáveis, metodologia e recursos tecnológicos usados durante a realização dos testes

<table>
<tr><th colspan="2">Localidade</th></tr>
<tr><td colspan="2">Local: IFMS – Campus Corumbá
Endereço: Rua Pedro de Medeiros, s/n – Bairro Popular Velha. CEP: 79.310-110
Corumbá – MS. Telefone: (67) 3234-9100</td></tr>
<tr><th colspan="2">Participantes</th></tr>
<tr><td colspan="2">1. Técnico em Informática, turma 1023. Quantidade de alunos: 28
2. Técnico em Informática, turma 1024. Quantidade de alunos: 36
3. Técnico em Informática, turma 2023. Quantidade de alunos: 40
4. Superior em Análise e Desenvolvimento de Sistemas, turma 1224. Quantidade de alunos: 16</td></tr>
<tr><th colspan="2">Aplicadores e Metodologia</th></tr>
<tr><td>1. Professor Luiz Felipe dos S. Freitas
Formação: Tecnólogo em Redes de</td><td>Metodologia: Aula de 45 minutos com uso de projetor multimídia e lousa sobre os principais conceitos aplicados no questionário de Conhecimentos em SO.</td></tr>
<tr><td>2. Professora Alessandra Carla Mendes
Formação: Doutora em Física</td><td>Metodologia: Aula de 45 minutos com uso do SigemVR executado em desktop e HMD para posterior aplicação do questionário de conhecimentos em SO.</td></tr>
<tr><th colspan="2">Recursos de Hardware</th></tr>
<tr><td>1. Laboratório de dispositivos móveis IFMS – Campus Corumbá:
40 Computadores HP – Intel Core i5 CPU 7100, 3.9Ghz
4GB de memória RAM, HD 500GB e Monitor Led 18,5 polegadas</td><td>2. Laboratório de dispositivos móveis IFMS – Campus Corumbá:
4 smartphones Android 6.0;</td></tr>
</table>

As Figuras 32, 33 e 34 apresentam as fotos do momento da realização do teste de conhecimentos em SOs em cada um dos grupos representativos dos diferentes métodos de abordagem de ensino, apresentados na Tabela 4. A Figura 31 mostra uma foto do grupo no qual não foi apresentado o simulador *SigemVR*. Já a Figura 32 mostra uma foto do grupo no qual o

simulador foi executado em *desktop*. Na Figura 33, é mostrada a foto de um aluno do grupo em que o simulador foi executado em um HMD, no momento em que o simulador foi apresentado após a aula teórica.

Figura 32: Aplicação do questionário de conhecimento em SO. Grupo de estudantes em que não foi apresentado o simulador

Figura 33: Aplicação do questionário de conhecimentos em SO. Grupo de estudantes em que o SigemVR foi executado em *desktops*

Figura 34: Foto de um aluno representante do grupo em que o SigemVR foi executado em um HMD

7.3 Resultados obtidos

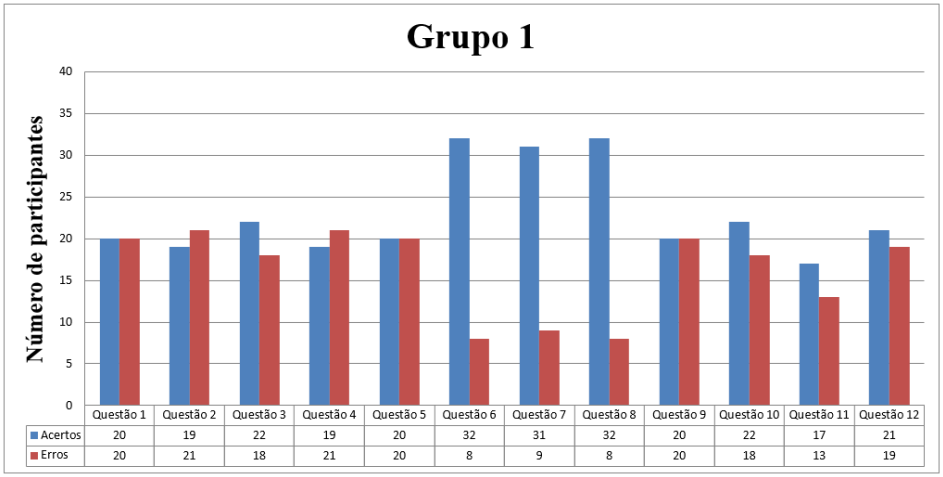
Esta seção apresenta os resultados obtidos nos testes do *software SigemVR*. A sub-seção 7.3.1 é dedicada à análise do perfil dos participantes. Na sub-seção 7.3.2 é feita uma comparação entre os resultados obtidos no questionário de conhecimentos em SO, considerando três diferentes métodos de ensino do módulo de gerência de memória principal em SO: 1. Método tradicional de ensino; 2. Método de ensino com auxílio do simulador *SigemVR* executado em *desktop*; 3. Método de ensino com auxílio do simulador *SigemVR* executado em HMD. Na sub-seção 7.3.3 são apresentados os resultados obtidos no questionário de recursos tecnológicos aplicado em estudantes que utilizaram as abordagens 2 e 3 de ensino. É importante considerar que na avaliação destes dados foi considerada a escala *Likert*.

7.3.1 Perfil dos participantes

Participaram dos testes 120 estudantes, com idade entre 15 e 53 anos, do Ensino Médio Técnico em Informática e do Curso Superior em Análise e Desenvolvimento de Sistemas do IFMS – Campus Corumbá. Desses, apenas 40 afirmaram terem cursado a disciplina de SO até o período de aplicação dos testes. Dentre os que cursaram a disciplina, nenhum estudante afirma ter usado algum *software* de auxílio. Um conjunto de 45 estudantes afirmou possuir experiência anterior com algum *software* com suporte à tecnologia de RV, dos quais apenas um entrevistado executou a aplicação em um *desktop*, enquanto os demais relataram o uso de *smartphones,* em conjunto com um dispositivo HMD, na experiência.

7.3.2 Comparação entre os métodos de ensino

O teste de conhecimento em SO (Apêndice II) foi realizado em três grupos distintos de 40 estudantes cada, de modo que em cada grupo foi usada uma abordagem de ensino diferente [Método Tradicional de Ensino (Grupo 1), Método de Ensino com auxílio do *SigemVR* em *desktop* (Grupo 2), Método de Ensino com auxílio do *SigemVR* em *HMD (Grupo 3)]*. As Figuras 35, 36 e 37 apresentam o quantitativo de acertos, em cada uma das questões consideradas no teste, nas diferentes abordagens de ensino. É possível verificar um maior quantitativo de acertos nas abordagens auxiliadas pelo simulador *SigemVR*.



	Questão 1	Questão 2	Questão 3	Questão 4	Questão 5	Questão 6	Questão 7	Questão 8	Questão 9	Questão 10	Questão 11	Questão 12
Acertos	20	19	22	19	20	32	31	32	20	22	17	21
Erros	20	21	18	21	20	8	9	8	20	18	13	19

Figura 35: Quantitativo de acertos e erros, em cada uma das questões consideradas no teste de conhecimentos em SO, dos alunos do Grupo 1

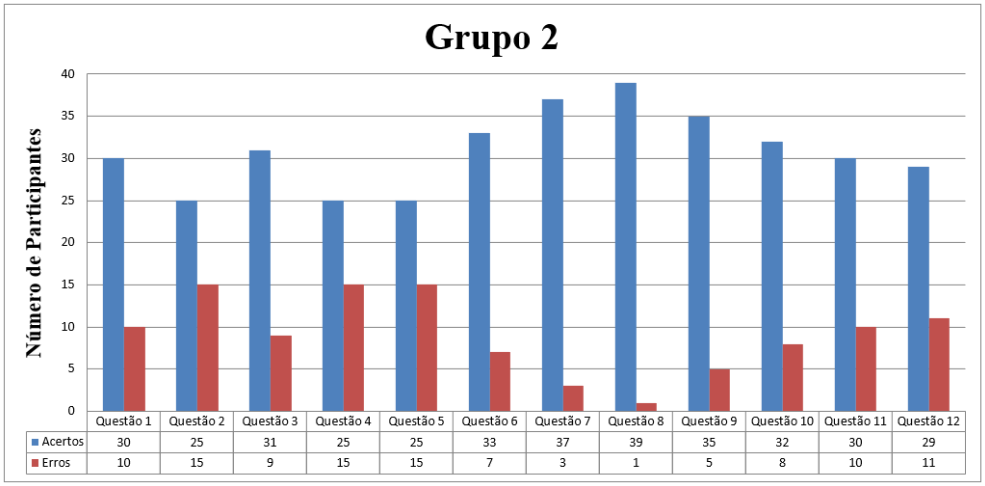


Figura 36: Quantitativo de acertos e erros, em cada uma das questões consideradas no teste de conhecimento em SO, dos alunos do Grupo 2

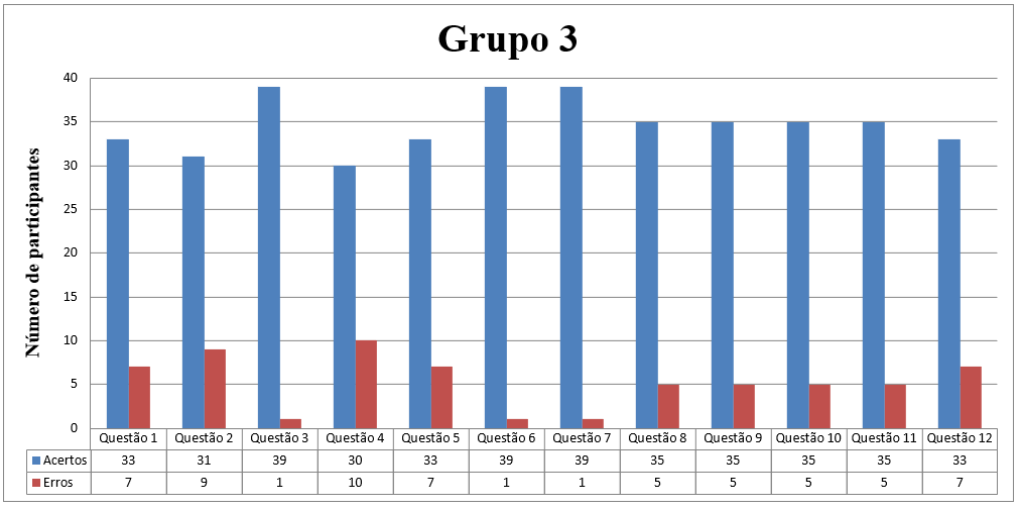


Figura 37: Quantitativo de acertos e erros, em cada uma das questões consideradas no teste de conhecimento em SO, dos alunos do Grupo 3

Para avaliar estatisticamente se o uso do simulador exerce alguma influência na variação do número de acertos dos alunos, foi adotado o modelo de Análise de Variância ANOVA, o qual é usado para comparar a distribuição de três ou mais grupos em amostras independentes. Um dos objetivos deste modelo é realizar um teste estatístico, que além de considerar a média do número de acertos entre os grupos, também leva em conta a variação do número de acertos dentro do mesmo grupo, com o intuito de verificar se existe uma diferença significativa entre essas médias. Em outras palavras, a análise de variância é utilizada para verificar se as diferenças amostrais são reais (causadas por diferenças significativas entre os grupos) ou casuais (decorrentes da aleatoriedade amostral). Essa análise considera que o acaso só produz

pequenos desvios, uma vez que as grandes diferenças nas médias são geradas por causas reais, que, no presente experimento, corresponde ao uso ou não do simulador. O teste pode ser aplicado uma vez que os principais pressupostos da análise são satisfeitos: as amostras são independentes (os dados são coletados aleatoriamente dentro do espaço amostral) e a variância entre os grupos é a mesma (cada aluno contribui de forma igual na soma dos quadrados). Para a aplicação do teste, a ANOVA define as seguintes hipóteses:

- H_0 (hipótese nula): evidencia a não existência de variâncias entre as diferentes abordagens de ensino avaliadas, ou seja, os grupos apresentam médias iguais.
- H_1 (hipótese alternativa): apresenta variância entre os grupos das diferentes abordagens de ensino avaliadas, ou seja, há pelo menos um grupo com alunos com desempenho diferente.

As informações geradas na análise de variância são mostradas na Tabela 8.

Tabela 8: Informações consideradas na análise de variância ANOVA

Fonte de variação	SQ	*gl*	MQ	F
Entre grupos	$SQE = \sum_{j=1}^{j=k} n_j (\overline{x_j} - \overline{\overline{x}})^2$ $\overline{x_j} = \frac{\sum_{i=1}^{n_j} x_i}{n_j} \quad e \quad \overline{\overline{x}} = \frac{\sum_{j=1}^{k} \overline{x_j}}{k}$	k-1	$MQE = \frac{SQE}{k-1}$	$F = \frac{MQE}{MQD}$
Dentro dos grupos	SQD = SQT - SQE	n-k	$MQD = \frac{SQD}{n-k}$	
Total	$\sum_{j=1}^{k} \sum_{l=1}^{n_j} (x_j^l - \overline{x_j})^2$	n-1	-	

O índice *j* indica qual o grupo, n_j = número de elementos do grupo *j,* k = número de grupos e *n* = soma do número de elementos de todas as amostras. No presente experimento*, j* pode ser igual a 1, 2 ou 3 (pois os alunos foram divididos em três grupos), $n_j = 40$ (cada grupo apresenta 40 estudantes)*,* k = 3 e *n* = 120 (número total de alunos). Na tabela, os graus de liberdade (*gl*) dependem do número total de dados e do número de grupos. São considerados dois estimadores de variância:

- SQE: representa a dispersão entre os grupos, uma vez que considera a soma dos quadrados do desvio da média de cada grupo ($\overline{x_1}, \overline{x_2}\, e\, \overline{x_3}$) em relação à média global ($\overline{\overline{x}}$). A média do quadrado entre os grupos (MQE) nada mais é do que o valor do SQE dividido pelo número de graus de liberdade (k-1 = 2).
- SQD: representa a dispersão dentro dos grupos, já que considera a soma dos quadrados do desvio do número de acertos de cada aluno de um determinado grupo ($x_j^l, com\ j = 1,2,3\, e\, l = 1,2,3...40$) em relação à média do grupo ($\overline{x_1}, \overline{x_2}\, e\, \overline{x_3}$). A média do quadrado dentro dos grupos (MQD) nada mais é do que o valor do SQD dividido pelo número de graus de liberdade (n-k = 117).

O fator F, usado como parâmetro para rejeitar ou não a hipótese nula definida pelo método, é dado pela razão entre MQE e MQD.

Para aplicação da análise de variância os dados das Figuras 33, 34 e 35, foram organizados na Tabela 9. As informações geradas pela ANOVA estão resumidas na Tabela 10.

Tabela 9: Número de acertos no teste de conhecimento em SO de cada um dos grupos de 40 alunos

Grupo 1	Número de acertos	7	10	3	8	11	3	10	8	8	2	1	4	7	12	8	12	5	2	12	4	7	5	10	2	3	4	10	4	10	7	3	6	8	10	12	4	5	9	7	12
	Aluno	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
Grupo 2	Número de acertos	11	10	8	10	12	9	12	8	11	11	8	9	11	12	11	4	12	11	9	8	10	11	11	5	3	11	10	9	11	12	8	9	9	12	11	12	8	9	6	6
	Aluno	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
Grupo 3	Número de acertos	12	10	10	12	9	9	10	9	10	7	8	9	9	8	10	8	12	11	11	11	12	12	12	12	12	11	11	11	8	10	8	12	11	11	10	12	12	12	11	12
	Aluno	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40

A Tabela 10 apresenta a análise de variância obtida a partir dos dados apresentados na tabela 9.

Tabela 10: Análise de variância dos dados apresentados na Tabela 9

Fonte de variação	SQ	*gl*	MQ	F
Entre grupos	262,4667	2	131,2333	21,68304
Dentro dos grupos	708,125	117	6,05235	
Total	970,5917	119	-	

Na tabela anterior, F ~ 21,68 e, de acordo com a estatística F, tal valor deve ser comparado ao valor de F_c (F crítico), o qual pode ser encontrado na tabela de distribuição F de Fisher-Snedecor com *k*-1 e *n-k* graus de liberdade, onde *k* é o número de grupos e *n* é o número de observações (Vieira, 2016). Para o presente experimento, com *k* = 3 e n = 120, F_c = 3,073. Como F > F_c, a hipótese nula (H_0) pode ser rejeitada, de modo que pode-se concluir que existe pelo menos uma média diferente das demais, mas não indica entre quais grupos a diferença é significativa. As médias do número de acertos dos grupos, assim como a média global ($\bar{\bar{x}}$), estão apresentadas na Tabela 11.

Tabela 11: Média do número de acertos no teste de conhecimentos em SO para cada um dos grupos considerados ($\overline{x_j}$, com *j* = 1, 2 e 3).

Grupos:	**Grupo 1**	**Grupo 2**	**Grupo 3**	$\bar{\bar{x}}$
Média de acertos ($\overline{x_j}$)	6,8750	9,5000	10,4250	8,9333

Para verificar qual(is) abordagens de ensino apresenta(m) alunos com desempenho significativamente diferentes, foi utilizado um teste de comparação múltipla, complementar ao ANOVA, conhecido como Teste de *Tukey*, o qual é aplicado para comparar médias, duas a duas. De acordo com o teste, duas médias são estatisticamente diferentes ao nível de significância de 5% toda vez que o valor da diferença entre elas for igual ou maior do que a *honestly significant difference* (HSD), ou seja, a diferença honestamente significante. Para obter o valor da *HSD* é preciso calcular:

$$HSD = q\sqrt{\frac{MQD}{r}}$$

Nessa fórmula:

- *q* é denominado amplitude estudentizada, encontrado em uma tabela (Vieira, 20175) que depende do número de grupos na análise de variância e do número de graus de liberdade da variação dentro dos grupos da ANOVA (Tabela 8).
- *MQD* é o quadrado médio dentro do grupo da análise de variância (Tabela 8).
- *r* é o número de repetições de cada um dos grupos.

Considerando um nível de significância de 5%, tem-se que:

$$HSD = 3{,}3573\sqrt{\frac{6{,}05235}{10}} \cong 2{,}6117$$

onde q = 3,3573 é o valor dado na tabela de amplitude estudentizada ao nível de significância de 5% associado a k = 3 grupos e 117 graus de liberdade. Os resultados obtidos pelo teste de *Tukey* estão apresentados na Tabela 12, que mostra o valor da *HSD* e o valor do módulo da diferença entre as médias de acerto para cada par de grupos ($\left|\overline{x_i} - \overline{x_j}\right|$, onde i e j são índices que representam os grupos). Pela tabela é possível observar que $\left|\overline{x_1} - \overline{x_2}\right|$ e $\left|\overline{x_1} - \overline{x_3}\right| >$ *HSD*, de modo que é possível concluir que o desempenho médio dos alunos que fizeram uso do simulador *SigemVR* (Grupo 2 e 3) é significativamente diferente dos alunos que não fizeram o uso do mesmo (Grupo 1). No entanto, quando a comparação é feita levando-se em consideração a execução do simulador em *desktop* (Grupo 2) ou *HMD* (Grupo 3), a diferença entre as médias de acerto no teste de conhecimento em SO deixa de ser significativa, uma vez que $\left|\overline{x_2} - \overline{x_3}\right| <$ *HSD.*

Tabela 12: Resultados obtidos pelo Teste de *Tukey* para o valor da HSD e da diferença entre as médias de acerto para cada par de grupos ($\left|\overline{x_i} - \overline{x_j}\right|$, onde *i* e *j* são índices que representam os grupos 1, 2 ou 3).

HSD	Comparação entre pares de grupos	$\left\|\overline{x_i} - \overline{x_j}\right\|$
2,6117	Grupo 1 e 2	2,625
	Grupo 1 e 3	3,55
	Grupo 2 e 3	0,925

7.3.3 Resultados do questionário de recursos tecnológicos do SigemVR

Além do teste de conhecimento em SO, os grupos de alunos que fizeram uso do simulador *SigemVR* [Grupo 2 (Método de Ensino com auxílio do *SigemVR* em *desktop*) e Grupo 3 (Método de Ensino com auxílio do *SigemVR* em HMD)] foram submetidos a um teste de recursos tecnológicos (Apêndice III), um questionário composto por doze questões que visa avaliar, a partir da escala *Likert*, recursos como a interface, usabilidade e performance do simulador. Questionários baseados na escala *Likert* especificam o nível de concordância do entrevistado com uma determinada afirmação. Seu formato típico de alternativas utiliza cinco

opções de respostas: 1. Concordo fortemente; 2. Concordo; 3. Nem concordo nem discordo; 4. Discordo; 5. Discordo fortemente. A Figura 37 apresenta o resultado do teste. No questionário aplicado havia os seguintes campos abertos para preenchimento do participante: 1. Aspectos identificados como positivos; 2. Aspectos identificados como negativos; 3. Caso tenha sentido algum desconforto na experiência, pode descrevê-lo?; 4. Sugestões para a aplicação. No entanto, nenhum desses campos foram preenchidos pelos participantes.

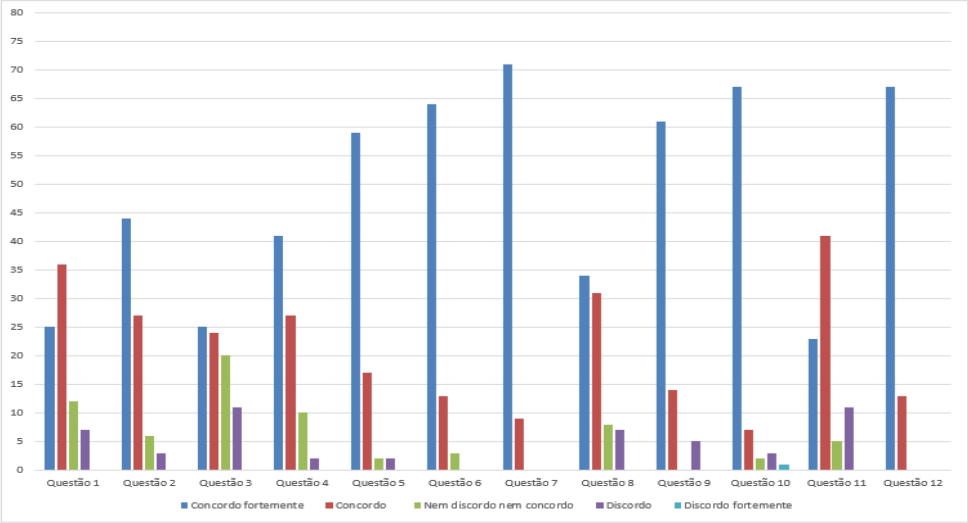


Figura 38: Resultado do teste de recursos tecnológicos do SigemVR realizado pelos grupos de alunos que fizeram uso do simulador (Grupo 2 e 3)

7.4 Discussão dos resultados

Os resultados dos testes de conhecimento em SO, mais especificamente sobre as diferentes técnicas de alocação de processos na memória RAM, revelaram que o simulador pode ser considerado uma ferramenta auxiliar no ensino do módulo de gerência de memória principal na disciplina de SOs desenvolvida no curso Técnico em Informática e no curso Superior em Análise e Desenvolvimento de Sistemas do IFMS – Campus Corumbá, uma vez que, em um espaço amostral de 120 alunos dividido igualmente em três grupos, a média de acertos foi maior nos grupos que fizeram uso do *software*, como mostraram os resultados dos testes estatísticos ANOVA e Teste de *Tukey* (Seção 7.3.2).

Já os resultados do teste de recursos tecnológicos mostraram que a grande maioria dos alunos concordou com a facilidade de uso, entendimento da aplicação e na facilidade da

compreensão das diferentes técnicas de alocação de processos na memória RAM, uma vez que as mensagens de áudio e vídeo foram classificadas pelos mesmos como recursos úteis e que facilitaram o aprendizado. Já a interface, apesar de ser classificada pelos alunos como agradável, compreensível, intuitiva e não causar desconforto, requer melhorias no sentido de aumentar a sensação do usuário em estar dentro do computador.

8. CONCLUSÕES E TRABALHOS FUTUROS

O uso da tecnologia como recurso auxiliar no processo de ensino-aprendizagem é crescente. A disciplina de SO possui conceitos abstratos que, se bem compreendidos, fornecem conhecimentos básicos para as demais áreas da computação. Contudo, na grande maioria dos casos, essa disciplina é abordada de maneira teórica através de recursos como quadro negro e projetor multimídia, dificultando a capacidade de compreensão dos alunos. Neste sentido, é importante adicionar às aulas de SOs ferramentas tecnológicas que possibilitem simular, de maneira genérica, prática e didática, diferentes tópicos abordados em sala de aula. Esta pesquisa desenvolveu uma aplicação em RV que simula as principais técnicas de gerência de memória principal em SOs, chamada de *SigemVR*. Essa aplicação oferece uma experiência tridimensional, imersiva e multissensorial através de um ambiente que dá ao usuário a sensação de estar dentro de um computador. A cada simulação, o usuário pode interagir com o *software*, de modo que ele pode criar novos processos, visualizar graficamente a alocação desses processos na memória principal (com base nos principais algoritmos utilizados por SOs modernos) e receber, em tempo real, mensagens de áudio em resposta às suas ações.

Após a concepção e desenvolvimento do *software*, um teste de conhecimento em SO foi realizado em três grupos distintos de 40 estudantes cada, de modo que em cada grupo foi usada uma abordagem de ensino diferente na apresentação das diferentes técnicas de alocação de processos na memória principal: Método Tradicional de Ensino (lousa e projetor multimídia) e Método de Ensino com auxílio do *SigemVR*, executado em *desktop e* em HMD. Em um espaço amostral de 120 alunos dos cursos Técnico em Informática e Superior em Análise e Desenvolvimento de Sistemas do IFMS – Campus Corumbá, os resultados revelaram que o simulador pode ser considerado uma ferramenta auxiliar no ensino do módulo de gerência de memória principal na disciplina de SO desenvolvida na instituição considerada, uma vez que a média de acertos foi maior nos grupos de alunos que fizeram uso do *software*, como mostraram os resultados dos testes estatísticos ANOVA e Teste de *Tukey*. Os mesmos grupos de alunos foram submetidos a um teste de avaliação dos recursos tecnológicos do simulador, cujos resultados apontaram considerações positivas quanto à usabilidade e performance do mesmo mas, em contrapartida, revelaram a necessidade de melhorias na sua interface.

8.1 Trabalhos futuros

Essa dissertação apresentou todo o processo de concepção e desenvolvimento de um simulador da gerência de memória em SO com suporte à tecnologia de RV que pode ser usado como ferramenta didática e complementar às aulas teóricas da disciplina de SOs. Neste contexto, os seguintes tópicos são sugeridos como trabalhos futuros:

- Adicionar novas técnicas de alocação de memória principal, como a técnica de segmentação.
- Realizar avaliações qualitativas do uso do simulador, com intuito de verificar o nível motivacional da sua utilização em detrimento do uso de simuladores 2D.
- Melhorar o ambiente virtual, inserindo novos recursos gráficos e animações.
- Desenvolver e integrar novos módulos de simulação ao *software*, como os módulos de gerência de processos e sistemas de arquivos.

REFERÊNCIAS

ADAMS, L. Visualização e Realidade Virtual. 2. ed. São Paulo: Makron Books, 1994.

ALBUQUERQUE, A. L. P. Cenários Virtuais com um Estudo de Sincronismo de Câmera. 1999. Dissertação (Mestrado em Ciência da Computação), Departamento de Informática, Pontífice Universidade Católica do Rio de Janeiro, Rio de Janeiro.

ANOVA, "Análise de Variância (ANOVA)". Disponível em: https://pt.khanacademy.org/math/statistics-probability/analysis-of-variance-anova-library. Acesso em 06/02/2018.

ARAÚJO, R. B. Especificação e Análise de um Sistema Distribuído de Realidade Virtual. 1996. Tese (Doutorado), Escola Politécnica da Universidade de São Paulo, São Paulo.

BROOKS, F. P. J. *What's real about virtual reality?* 1.ed. *Computer Graphics and Applications*, IEEE, 19(6):16-27, 1999.

BURDEA, G. *Force and Touch Feedback for Virtual Reality*. 2. ed. New York: John Wiley & Son's, 1996.

BURDEA, G.; COIFFET, P. *Virtual Reality Technology*. 2.ed. New York: John Wiley & Son's, 2003.

CADOZ, C. A realidade virtual. Flammarion: Editora BBCC, 1994.

CARDOSO, A. Tecnologias para o desenvolvimento de sistemas de realidade virtual e aumentada. Recife: Editora Universitária da UFPE, 2007.

ELLIOTT, S. D.; MILLER, P. L. 3D Studio 4.0 Técnicas Avançadas. São Paulo: Berkeley, 1995.

FOLEY, J. D. et. al. *Computer Graphics, Principles and Practice*. 2. ed. New York: Addison Wesley, 1990.

Google, "*Quickstart for Google VR SDK for Unity with Android*". Disponível em: https://developers.google.com/vr/develop/unity/get-started-android. Acesso em: 28/12/2017.

GIBSON, W. *Neuromancer*. New York: ACE Books, 1984.

GRADECKI, J. *The Virtual Reality Construction Kit*. 2. ed. New York: John Wiley & Son's, 1995.

GUIMARÃES, M. P.; NUNES, E. P. S.; RIEDER, R.; HOUNSELL, M. S. Tendências e Técnicas em Realidade Virtual e Aumentada. V. 5. Porto Alegre: SBC, 2015.

HANCOCK, D. *Viewpoint: Virtual Reality in Search of Middle Ground*. IEEE Spectrum, 32(1): 68, 1995.

JACOBSON, L. Realidade Virtual em casa. 1. ed. Rio de Janeiro: Berkeley, 1994.

KIOKI, E. Y.; SANTIAGO, P. P. S. A. C. 2008. Um simulador didático como ferramenta de apoio ao ensino da disciplina de Sistemas Operacionais. Revista INICIA, P41-48.

KIRNER, C.; SISCOUTTO, R. Realidade Virtual e Aumentada: Conceitos, Projeto e Aplicações. Gramados: *Symposium on Virtual and Augmented Reality*, 2016.

KIRNER, T.; KIRNER, C.; KAWAMOTO, A. L. S.; WAZLAWICK, R. S. *Development of a Collaborative Virtual Environment for Education Applications*. 2001, *Proceedings of the ACM WEB3D International Conference*. Pederborn, Germany, p. 61-68.

LARMAN, C. Utilizando UML e Padrões. 3ed. São Paulo: Bookman, 2007.

LESTON, J. *Virtual Reality: the it Perspctive. Computer Bulletin*, pp. 12-13, 1996.

LEWIS, M.; JACOBSON, J. *Games engines in scientific research*. Commun. ACM, 45(1): 27-31, 2002.

LIKERT, R. *A Technique for the Measurement of Attitudes*. *Archives of Psychollogy*, 140: pp. 1-155, 1932.

MACHADO, L. S. Conceitos Básicos de Realidade Virtual. 1995. Monografia, INPE – 5975-PUD/025, Instituto Nacional de Pesquisas Espaciais, São José dos Campos / SP. Disponível on-line em: http://www.lsi.usp.br/~liliane/conceitosrv.html

MACHADO, F. B.; MAIA, L. P. Arquitetura de Sistemas Operacionais. 5ed. São Paulo: LTC, 2013.

MAIA, L. P. Sosim: Simulador para o ensino de sistemas operacionais. 2001. Dissertação (Mestrado em Informática), Universidade Federal do Rio de Janeiro, Rio de Janeiro.

MARTINS, F. M.; GUIMARÃES, M. P. Desafios para o uso de Realidade Virtual e Aumentada de maneira efetiva no ensino. Workshop de Desafios da Computação Aplicada à Educação, 2012.

MARTINS, F. M.; OLIVIEIRA, A. J. G.; GUIMARÃES, M. P. Implementação de um laboratório de realidade virtual de baixo custo: estudo de caso de montagem de um laboratório para o ensino de matemática. Revista Brasileira de Computação Aplicada, Passo Fundo, v. 5, n.1, p. 98-112, 2013.

MARTINS, V. F.; NIEDERMEYER, A.; GUIMARÃES, M. P. *Educational game for Aid in Literacy Process*. IEEE, 2016.

MENDES, F.; LIMA, A. S.; MARTINS, V. F.; GUIMARÃES, M.P. Desenvolvimento de Aplicações para *Oculus Rift*: Integração do *Oculus Rift* com o *Street View*.

MORIE, J. F. *Inspiring the future: Merging Mass Communication, Art, Entertainment and Virtual Environment. Computer Graphics*, 1994.

NASCIMENTO, J. K. F. Informática Aplicada à Educação. 84p, 2009. ISBN: 978-85-230-0981-6.

NETTO, A. V.; MACHADO, L. S.; OLIVEIRA, M. C. F. Realidade Virtual Fundamentos e Aplicações. 1. ed. Florianópolis: Visual Books, 2002;

OLIVEIRA, R. S.; CARISSIMI, A. S.; TOSCANI, S. S. Sistemas Operacionais. Revista de Informática Teórica e Aplicada, vol III, número 3, dezembro de 2001.

PANTELIDIS, V.; VINCIGUERRA, D. *What is Virtual Reality?* Versão eletrônica, 1.1. Retirado em 02 de setembro de 2018 de http://http://vr.coe.ecu.edu/pu.htm

PEARS, A. *A Survey of Literature on the Teaching of Introductory Programming in Innovationand Technology in Computer Science Education*. ACM, 204-223. 2007.

REIS, F. P., PARREIRA JÚNIOR, P. A., XAVIER COSTA, H. A. (2009). "Tbc-so/web: Um software educacional para o ensino de políticas de escalonametno de processos e de alocação de memória em sistemas operacionais." Anais do Simpósio Brasileiro de Informática na Educação, volume 1.

SCAMATI, V. Um simulador para o Ensino de Sistemas Operacionais com a tecnologia da Realidade Virtual (ESORV). 2017. Dissertação (Mestrado em Ciência da Computação) - Faculdade Campo Limpo Paulista, Campo Limpo Paulista.

SHERMAN, W. R.; CRAIG, A.B. *Understanding virtual reality: Interface, Application and design.* 1ed. *Morgan Kaufman Series in Computer Graphics*, 2002.

SILBERSCHATZ, A. Fundamentos de Sistemas Operacionais: Princípios Básicos. 1. ed. São Paulo: Editora LTC, 2015.

SourceDNA, "Unity Use for Developers". Disponível em: https://www.crunchbase.com. Acesso em: 01/03/2018.

TAJRA, S. F. Informática na educação: novas ferramentas para o professor da atualidade. 2 ed. São Paulo: Érica, 2000.

TANEMBAUM, A. Organização Estruturada de Computadores. 2. ed. São Paulo: Pearson - Addison Wesley, 2006.

TANEMBAUM, A. Sistemas Operacionais Modernos. 4. ed. São Paulo: Pearson Prentice Hall, 2016.

Unity, "*Unity Engine*". Disponível em https://unity3d.com/pt/unity. Acesso em 06/02/2018.

VIEIRA, S. Análise de Variância (ANOVA). 2º edição. São Paulo: Atlas, 2016.

VIEIRA, S. Estatística Básica. 1º edição. São Paulo: Cengage, 2015.

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

WAZLAWICK, R. S. Análise e Projeto de Sistemas de Informação Orientados a Objetos. 1. ed. Rio de Janeiro: Elsevier Editora, 2004.

ZORZO, A. F.; NUNES, D.; MATOS, E.; STEINMACHER, I.; LEITE, J.; ARAUJO, R. M.; CORREIA, R.; MARTINS, S. "Referenciais de Formação para os Cursos de Graduação em Computação". Sociedade Brasileira de Computação (SBC). 153p, 2017. ISBN 978-85-7669-424-3

Apêndice I – Questionário de análise para desenvolvimento do Simulador SigemVR (elaborado pelo autor)

Cidade:	**Instituição de Ensino:**
Nome:	**Data:**
Formação:	
Formação complementar: ( ) Pós-Graduação ( ) Mestrado ( ) Doutorado _____	
Sobre a disciplina de Sistemas Operacionais:	
1. **Você cursou a disciplina de Sistemas Operacionais durante a graduação?** **Assinale com X apenas uma alternativa** ( ) Sim ( ) Não	
2. **Se sim, assinale os conteúdos que foram abordados durante a disciplina:** Assinale com X uma ou mais alternativas ( ) Fundamentos ( ) Gerência de Processos ( ) Gerência de Memória Principal ( ) Sistema de arquivos ( ) Gerência de Dispositivos ( ) Sistemas com múltiplos processadores ( ) Outros Especifique outros:_____	
3. **Você participou de aulas práticas dessa disciplina durante sua formação?** **Assinale com X apenas uma alternativa** **( ) Sim ( ) Não**	
4. **Há quanto tempo você leciona essa disciplina?** **Assinale com X apenas uma alternativa** **( ) Menos de 1 ano** **( ) de 1 a 5 anos**	

( ) 5 anos ou mais

5. **Em quais níveis de ensino você lecionou a disciplina?**

Assinale com X uma ou mais alternativas.

( ) EJA/Técnico

( ) Ensino Médio/Técnico

( ) Ensino Superior

( ) Pós-Graduação

6. **Quais recursos pedagógicos você já utilizou ou utiliza em suas aulas?**

Assinale com X uma ou mais alternativas.

( ) Projetor multimídia

( ) Quadro/Lousa

( ) Laboratório de informática

( ) Material Impresso

( ) Outros

Especifique outros: _____

7. **Você utiliza recursos que práticos em sala de aula?**

Assinale com X apenas uma resposta.

( ) Sim ( ) Não

7. a) Se a resposta da questão 7 for sim, qual (s) do(s) método(s) apresentado(s) a seguir você utilizou em suas aulas práticas?

Assinale com X uma ou mais alternativas.

( ) Sistemas Operacionais de código aberto

( ) Simuladores

( ) Desenvolvimento de rotinas

( ) Outros

Especifique outros: _____

7. a) Se a resposta da questão 7 for não, qual(s) dificuldade(s) encontrada(s) para realizar o ensino prático da disciplina?

Assinale com X uma ou mais alternativas.

( ) Dificuldade em abordar prática em Sistemas Operacionais de código aberto

( ) Dificuldade em encontrar softwares específicos para prática da disciplina

( ) Falta de material didático que aborde aulas práticas para a disciplina

( ) Outros.

Especifique outros:_____

8. **Você conhece ou utiliza algum dos softwares simuladores de Sistemas Operacionais apresentados a seguir?**

Assinale com X uma ou mais alternativas

( ) SOsim

( ) SSOG (Simulador de Sistema Operacional Genérico)

( ) TB-SO/Web

( ) Outro.

Qual(s)? _____

9. **Caso já tenha utilizado algum dos *softwares* apresentados, o que você classifica como positivo nessa experiência?**

Assinale com X uma ou mais alternativas

( ) Facilidade em adquirir o software

( ) Interface

( ) Facilidade de Uso

( ) Facilidade em assimilar com conteúdo teórico

( ) Aborda diversos conteúdos disciplinares

( ) Outros.

Qual(s)? _____

10. **Caso já tenha utilizado algum dos *softwares* apresentados, o que você classifica como negativo nessa experiência?**

Assinale com X uma ou mais alternativas

( ) Facilidade em adquirir o software

( ) Interface

( ) Facilidade de Uso

( ) Facilidade em assimilar com conteúdo teórico

( ) Não aborda diversos conteúdos disciplinares

( ) Outros.

( ) Especifique outros: _____

Sobre a tecnologia de Realidade Virtual:

1. **Você já utilizou algum software de realidade virtual?**

Assinale com X uma alternativa

( ) Sim ( ) Não

Qual(s)? _____

2. **Se a resposta anterior for sim, em que tipo de dispositivo você utilizou esse software?**

Assinale com X uma ou mais alternativas

( ) HaedSet

( ) Computador

( ) HMD (Head Mounted Display)

( ) Outro(s)

Especifique outros: _____

3. **Quais pontos você considera positivo no uso da tecnologia de realidade virtual?**

Assinale com X uma ou mais alternativas

( ) Interface

( ) Usabilidade

( ) Variedade de softwares gratuitos

( ) Facilidade de aprendizagem do conteúdo

( ) Custo

( ) Desempenho

( ) Outros.

Especifique outros:_____

4. **Quais pontos você considera negativos no uso da tecnologia de realidade virtual?**

Assinale com X uma ou mais alternativas

( ) Interface

( ) Usabilidade

( ) Variedade de softwares gratuitos

( ) Facilidade de aprendizagem do conteúdo

( ) Custo

( ) Desempenho

( ) Outros.

Especifique outros:_____

Sobre gerenciamento de memória principal em Sistemas Operacionais:

1. **Quais técnicas de gerenciamento de memória são abordadas durante as suas aulas?**

Assinale com X uma ou mais alternativas

( )Alocação Contígua Simples

( ) Alocação Particionada Estática

( ) Alocação Particionada Dinâmica

( ) Paginação

( ) Segmentação

( ) Segmentação com paginação

( ) Outros.

Especifique outros:_____

2. **Quais dos conteúdos a seguir, você considera fundamental para o ensino de Gerenciamento de Memória Principal em Sistemas Operacionais?**

Assinale com X uma ou mais alternativas

( )Alocação Contígua Simples

( ) Alocação Particionada Estática

( ) Alocação Particionada Dinâmica

( ) Paginação

( ) Segmentação

( ) Segmentação com paginação

3. **Das opções assinaladas acima, quais você atribui menor dificuldade no processo de ensino-aprendizagem?**

Assinale com X uma ou mais alternativas

( )Alocação Contígua Simples

( ) Alocação Particionada Estática

( ) Alocação Particionada Dinâmica

( ) Paginação

( ) Segmentação

( ) Segmentação com paginação

4. **Das opções assinaladas acima quais você atribui maior dificuldade no processo de ensino-aprendizagem?**

Assinale com X uma ou mais respostas

( )Alocação Contígua Simples

( ) Alocação Particionada Estática

( ) Alocação Particionada Dinâmica

( ) Paginação

( ) Segmentação

( ) Segmentação com paginação

Apêndice II – Questionário: Teste de Conhecimento em SO (elaborado pelo autor)

Teste de usuário – SigemVR

Pré-teste		
Nome:		
Curso:		Turma:
Idade:	Sexo: ( ) Masc ( )Fem	Data:
Já cursou a disciplina de Sistemas Operacionais? ( ) Sim ( ) Não		
Se a resposta anterior for sim, seu professor utilizou algum software no decorrer da disciplina? ( ) Não ( ) Sim. Qual? _____		
Já utilizou *softwares* com tecnologia de Realidade Virtual? ( ) Sim ( ) Não		
Se a resposta anterior for sim, qual o dispositivo usado na experiência RV? Desktop ou computador pessoal ( ) HMDs ou smartphones ( ) Outros ( ) _____		
Abordagem usada no teste: Sala de aula ( ) Desktop ( ) HMD ( )		

<u>Questões sobre o aprendizado de Gerência de Memória Principal em Sistemas Operacionais:</u>

1. O que são processos? ( ) São módulos de execução do sistema operacional. ( ) São programas em execução. ( ) São arquivos com dados de usuários e sistema. ( ) São espaços de endereçamento de programas na memória RAM.
2. O que é um espaço de endereçamento de um processo? ( ) São endereços usados para mapear os arquivos na memória secundária. ( ) São endereços lógicos da memória física secundária, geralmente HD, disponibilizadas aos processos. ( ) São endereços lógicos da memória física RAM disponibilizadas aos processos. ( ) São endereços físicos das instruções executadas pelo processador.
3. Todo novo processo possui um espaço de endereçamento na memória RAM?

( ) Sim. ( ) Não.
4. Em sistemas operacionais, o que é alocar memória a um processo? ( ) Significa direcionar (disponibilizar) a um determinado processo um espaço de endereçamento. ( ) Significa permitir que um processo seja executado pelo processador. ( ) Alocação define a organização dos arquivos de um processo no disco. ( ) Indica a mudança de estado de um processo de pronto para executando.
5. Em sistemas operacionais, o que é fragmentação de memória? ( ) Indica o desperdício de memória RAM. ( ) Significa permitir que um processo seja executado pelo processador. ( ) Indica a mudança de estado de um processo de pronto para executando. ( ) Indica o desperdício
6. O sistema operacional ocupa espaço em memória? ( ) Sim. ( ) Não.
7. É possível criar um novo processo antes que o sistema operacional seja inicializado? ( ) Sim. ( ) Não.
8. Quando um processo é finalizado, ele continua ocupando o espaço de endereçamento destinado a ele em RAM? ( ) Sim. ( ) Não.
9. “Essa técnica de alocação de memória caracteriza-se por: dividir a memória RAM em duas partes. Uma parte para o sistema operacional e a segunda, composta pela sobra da memória deixada pelo SO, é disponibilizada ao processo.” Essa definição corresponde a qual técnica de alocação de memória principal? ( ) Alocação contígua simples. ( ) Alocação particionada estática. ( ) Alocação particionada dinâmica. ( ) Alocação por paginação.
10. “Nessa técnica não há o conceito de partições de tamanho fixo. Cada processo é alocado de acordo com a sua necessidade e de maneira contínua, tornando aquela área sua partição.” Essa definição corresponde a qual técnica de alocação de memória principal? ( ) Alocação contígua simples. ( ) Alocação particionada estática. ( ) Alocação particionada dinâmica. ( ) Alocação por paginação.

11. “Essa técnica divide a memória principal em pedaços de tamanho fixo, denominados partições. O tamanho da partição é definido na inicialização do sistema operacional. A necessidade de memória do processo deve ser igual ou menor ao tamanho da partição disponível, para que ele possa ser alocado.” Essa definição corresponde a qual técnica de alocação de memória principal?

( ) Alocação contígua simples.

( ) Alocação particionada estática.

( ) Alocação particionada dinâmica.

( ) Alocação por paginação.

12. “Essa técnica permite alocar processos na memória principal de maneira não contígua, através de endereços chamados de *frames.* Esses frames são distribuídos na memória RAM e em parte da memória secundária.” Essa definição corresponde a qual técnica de alocação de memória principal?

( ) Alocação contígua simples.

( ) Alocação particionada estática.

( ) Alocação particionada dinâmica.

( ) Alocação por paginação.

Apêndice III – Questionário: Teste de recursos tecnológicos do SigemVR

(elaborado pelo autor)

Teste de usuário – SigemVR

Nome:	
Curso:	
Data:	**Turma:**
Abordagem: ( ) Desktop ( ) HMD ( ) Sala de aula	

Questões sobre Realidade Virtual e uso do software SigemVR

Experiência de utilização do SigemVR
1. A interface gráfica do simulador SigemVR é agradável e compreensível. ( ) Discordo Fortemente ( ) Discordo ( ) Nem discordo nem concordo ( ) Concordo ( ) Concordo Fortemente
2. É fácil entender o que temos que fazer dentro da aplicação. ( ) Discordo Fortemente ( ) Discordo ( ) Nem discordo nem concordo ( ) Concordo ( ) Concordo Fortemente
3. A interface é muito intuitiva. ( ) Discordo Fortemente ( ) Discordo ( ) Nem discordo nem concordo ( ) Concordo ( ) Concordo Fortemente
4. Foi fácil aprender a usar a aplicação. ( ) Discordo Fortemente ( ) Discordo ( ) Nem discordo nem concordo ( ) Concordo

( ) Concordo Fortemente
5. O tempo de resposta às ações dentro do ambiente é aceitável. ( ) Discordo Fortemente ( ) Discordo ( ) Nem discordo nem concordo ( ) Concordo ( ) Concordo Fortemente
6. As mensagens de áudio funcionam no momento certo e são explicativas. ( ) Discordo Fortemente ( ) Discordo ( ) Nem discordo nem concordo ( ) Concordo ( ) Concordo Fortemente
7. Quando um processo é criado, eu consigo visualizar a distribuição gráfica dele na memória RAM. ( ) Discordo Fortemente ( ) Discordo ( ) Nem discordo nem concordo ( ) Concordo ( ) Concordo Fortemente
8. É fácil clicar nos elementos de interação disponíveis no ambiente. ( ) Discordo Fortemente ( ) Discordo ( ) Nem discordo nem concordo ( ) Concordo ( ) Concordo Fortemente
9. O vídeo explicativo é de fácil compreensão e explicativo. ( ) Discordo Fortemente ( ) Discordo ( ) Nem discordo nem concordo ( ) Concordo ( ) Concordo Fortemente
10. Não tive nenhum desconforto ao executar a aplicação. ( ) Discordo Fortemente ( ) Discordo ( ) Nem discordo nem concordo ( ) Concordo ( ) Concordo Fortemente

11. A aplicação permite que o usuário sinta-se dentro de um computador.

( ) Discordo Fortemente

( ) Discordo

( ) Nem discordo nem concordo

( ) Concordo

( ) Concordo Fortemente

12. A aplicação contribui na compreensão das técnicas de alocação de memória RAM em Sistemas Operacionais.

( ) Discordo Fortemente

( ) Discordo

( ) Nem discordo nem concordo

( ) Concordo

( ) Concordo Fortemente

Considerações sobre a aplicação

Aspectos identificados como positivos:

Aspectos identificados como negativos:

Caso tenha sentido algum desconforto na experiência, pode descrevê-lo?

Sugestões para a aplicação: