**UNIVERSIDADE FEDERAL DE SANTA CATARINA**
**DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA**

**MATHEUS TEIXEIRA FERNANDES**

# Integração de um *Tracking* Magnético a um simulador de soldagem manual com Realidade Virtual utilizando *Unity3D*

**Florianópolis, 2016**
**Matheus Teixeira Fernandes**

# Integração de um *Tracking* Magnético a um simulador de soldagem manual com Realidade Virtual utilizando *Unity3D*

Trabalho de conclusão de curso com
o objetivo de conseguir o título de
Bacharel em Ciências da
Computação pela Universidade
Federal de Santa Catarina

**Orientação: Fabiano Garcia**
**Professor Responsável: Prof. Antônio Augusto Fröhlich**

**Universidade Federal de Santa Catarina**
**Florianópolis,**
**2º Semestre, 2016**

## Agradecimentos

À minha amada mulher, amiga e companheira, Evelyn, que sempre esteve ao meu lado me apoiando, incentivando e acreditando no meu potencial. Aos meus pais, Marcos e Maria, pela paciência. Ao meu irmão pelo apoio e paciência.

Aos meus colegas de trabalho, Alessandro e Bruno, pelo apoio e por não aceitarem um trabalho menos que excelente.

Ao meu orientador Fabiano, por aceitar o desafio de me orientar neste trabalho.

Trabalho de Conclusão de Curso apresentado sob o título de "Integração de um *Tracking* Magnético a um simulador de soldagem manual com Realidade Virtual utilizando *Unity3D*", defendido por Matheus Teixeira Fernandes no dia 11/11/16, em Florianópolis, Santa Catarina pela banca examinadora constituída por:

_____

Prof. Antônio Augusto Fröhlich, Doutor
Professor Responsável

_____

Fabiano Garcia
Orientador Externo

_____

Alessandro Vieira dos Reis

_____

Regis Henrique Gonçalves e Silva

## Resumo

A Realidade Virtual constitui-se em uma ferramenta que vem ganhando grande espaço na indústria de jogos eletrônicos e simuladores. Esta ferramenta traz muitas possibilidades, mas ainda não há muitas soluções com relação aos equipamentos que auxiliam na interação. A criação e integração destes equipamentos para tais aplicações apresenta grande importância para manter as vantagens da realidade virtual. Este trabalho teve por objetivo integrar um Rastreador (*tracking*) Magnético com um simulador de solda, desenvolvido em *Unity3D*, de forma a interagir com a pistola de solda e garantir uma resposta do movimento do usuário com os eventos na realidade virtual tornando-os mais naturais e imersivos.

Palavras-Chave: Solda; Simulação; Rastreador; Movimento; Engine; Virtual Reality; VR

**Abstract**

The Virtual Reality is a tool that are gaining space in games and simulation industry. This tool has many possibilities, but there is not so many solutions to devices that help in its interaction. The creation and integration of these devices for such applications is quite important to keep the virtual reality advantages. This study aimed to integrate a magnetic tracker with a welding simulator, developed in *Unity3D*, in order to interact with the welding gun and ensure that the user's movement responses to the events in virtual reality making them more natural and immersive.

*Key-Words*: Virtual Reality; *Unity3D*; Magnetic *Tracking*; *Integration*; *Welding*; *Simulator*

# LISTA DE FIGURAS

# LISTA DE TABELAS

## Sumário

# 1 Introdução

## 1.1 Contextualização

Entende-se por simuladores, máquinas que reproduzem o comportamento de um sistema sob determinadas condições permitindo que o usuário desse sistema possa praticar certa atividade antes de iniciar a mesma na vida real.

Segundo Baladez (2009), se tornam indispensáveis para a formação de pessoas que terão grande responsabilidade a seu cargo já que eventuais erros poriam em risco a sua vida e a vida de outras pessoas.

Segundo Macedo, Dickman e Andrade (2012), os simuladores podem ser divididos em dois grupos: não interativos onde o usuário não pode alterar os parâmetros de uma simulação e mostram e ilustram a evolução de algum evento e os interativos onde o usuário pode alterar vários parâmetros da simulação, explorando a situação apresentada e verificando as consequências de cada alteração realizada.

O computador é o recurso disponível atualmente para a utilização de simuladores com o uso de diversos e sofisticados recursos gráficos, matemáticos e lógicos para alcançar um modelo de sistema que ofereça ao mesmo tempo, flexibilidade, agilidade e confiabilidade para o projeto.

Para esse desenvolvimento é possível utilizar entre tantas ferramentas disponíveis, a *Unity3D* que é conhecida como um motor gráfico simples em sua utilização e destaca-se pelos seus projetos principalmente em *3D*, contando com uma boa imagem gráfica.

O software, *Unity3D,* foi criado pela empresa *Unity Technologies* que é uma empresa global, com sede em *San Francisco, CA,* Estados Unidos e conta com escritórios no Canadá, China, Colômbia, Dinamarca, Finlândia, Alemanha, Japão, Coreia, Lituânia, Cingapura, Suécia, Ucrânia e Reino Unido. (Unity, 2016).

Essa empresa é a criadora de uma plataforma de desenvolvimento de ponta a ponta, flexível e de alto desempenho, usada para criar experiências ricas, interativas *2D, 3D*, Realidade Virtual e Realidade Aumentada. (UNITY, 2016).

A *engine* gráfica *Unity* e o editor com todas as suas funcionalidades servem de base para o desenvolvimento de jogos ou aplicativos e publicá-los facilmente em várias plataformas como dispositivos móveis, computadores pessoais e sistemas incorporados. (Unity, 2016).

Também oferece soluções e serviços para a criação de jogos, aumentando a produtividade e a conexão com o público, incluindo *Unity Asset Store, Unity Cloud Build, Unity Analytics, Unity Ads, Unity Everyplay e Unity Certification*. (Unity, 2016)

A empresa *Unity Technologies* atende a mais de 5,5 milhões de desenvolvedores registrados, incluindo grandes editoras, estúdios independentes, estudantes e entusiastas de todo o mundo. (UNITY, 2016).

Os clientes da *Unity* incluem *Coca-Cola, Disney, Electronic Arts, LEGO, Microsoft, NASA, Nexon, Nickelodeon, Square Enix, Ubisoft, Obsidian, Insomniac e Warner Bros*. De grandes a pequenos estúdios até profissionais independentes. (UNITY, 2016).

Com a *Unity3D* é possível realizar entre outros projetos, os de realidade virtual que para Rodrigues e Porto (2013), pode ser considerada uma imersiva e interativa experiência que se baseia em imagens gráficas *3D* geradas por computador em tempo real, ou seja, é uma simulação do mundo real.

Essa situação de imersão mostra ao usuário que, quando imerso no ambiente virtual, pode ter a sensação de estar dentro desse ambiente e fazer parte dele. Essa sensação é captada pelos muitos dispositivos que transmitem a esse usuário a sensação de entrada no ambiente virtual, levando seus sentidos sensoriais e sua atenção para o que está acontecendo neste espaço e isolando-o do mundo exterior, permitindo assim que ele consiga explorar e não apenas observar.

É importante salientar que existe, além da realidade virtual utilizada neste trabalho, a realidade aumentada que segundo Garcia e Luz (2008):

> "A realidade aumentada integra informação virtual com o ambiente real. Através de um vídeo do ambiente real se extrai informações de uma imagem, como: movimento, distâncias e objetos. Com essas informações podemos adicionar suplementos virtuais ao vídeo, que são capazes de

> interagir e reagir com objetos do mundo real ou com os movimentos do usuário"

O simulador de solda abordado neste trabalho utiliza exclusivamente a tecnologia de realidade virtual, onde não é necessário nenhum objeto real, a não ser o *tracking* magnético para a pistola de solda. Desta maneira, é possível transportar de forma imersiva o aprendiz para qualquer lugar, tornando possível o treinamento em diferentes ambientes, como fábricas, oficinas, estaleiros, etc. Há também a possibilidade de oferecer peças para a soldagem de diversos tamanhos e formas. Já os simuladores que oferecem a realidade aumentada permitem o toque e manipulação das peças reais, contudo a simulação deverá respeitar as limitações físicas das peças e do ambiente.

## 1.2 Problemática

No contexto explicado anteriormente surge então o desafio de trazer uma boa imersão do usuário, garantindo qualidade e resultados positivos para as atividades relacionadas à soldagem.

Para garantir a real sensação de imersão em um determinado ambiente virtual, o rastreamento dos movimentos ou também conhecido como *tracking* pode ser utilizado e é um dos grandes suportes em um ambiente virtual. É através dele que o sistema de realidade virtual interpreta os comandos motores do usuário e responde a eles de forma adequada e com certa precisão na ação.

Entre muitos exemplos de simuladores que utilizam o *tracking* magnético, pode-se citar o cirúrgico com realidade virtual e o de solda que será abordado neste trabalho podendo inicialmente ser caracterizado como uma ferramenta dinâmica e eficaz ao aprendizado do usuário proporcionando assim uma real imersão dele em atividades que requerem prática para que possam ser realizadas no seu cotidiano na vida real.

Contudo, integrar um *tracker* em um projeto de simulador em andamento, utilizando uma tecnologia já definida, pode não ser uma tarefa simples. É preciso entender e ajustar as tecnologias envolvidas para que possam interagir entre sí. Neste projeto é imprescindível que a tecnologia a ser inserida, o *tracking* magnético, e a tecnologia já definida, o simulador de solda

desenvolvido na *Unity3D*, conversem de forma a entregar ao usuário uma experiência imersiva satisfatória.

Durante as pesquisas e abordagens realizadas para a execução deste projeto, descobriu-se a importância da utilização do *tracking* para diversas áreas, como simuladores para treinamento de cirugias na área médica ou diversos tipos de treinamento na área militar. A utilização destes dispositivos vem aumentando com o passar do tempo, porém poucas literaturas demonstraram o uso dele na área industrial no que diz respeito à soldagem de materiais.

E ainda, traz o desafio de se utilizar junto à realidade virtual que pode se tornar um bom aliado para o aperfeiçoamento das técnicas de soldagem.

Para o bom aprendizado da habilidade de soldagem utilizando um simulador virtual é imprescindível uma ferramenta com boa precisão e boa calibração.

## 1.3 Justificativa

A escolha do tema Integração de um *Tracking* Magnético a um simulador de soldagem manual com Realidade Virtual utilizando *Unity3D* apresenta relevância na área de realidade virtual e sistemas interativos no campo da Ciência da Computação; e se justifica em termos pelo seu valor social e econômico onde pode apresentar benefícios à indústria de simuladores focada em aplicações educacionais.

A utilização de simuladores traz benefícios como aprendizado, treinamento e aperfeiçoamento facilitando a realização das atividades em um ambiente imersivo e interativo muito próximo ao mundo real. Quanto a sua utilização, o soldador não terá contato com a fumaça, haverá a redução de custos e de problemas em preparar exercícios para o usuário, pois o mesmo não necessitará realizar corte de peça.

Este projeto poderá ainda, contribuir oferecendo uma nova forma de integrar um *tracking* magnético à um simulador, oferecendo uma nova perspectiva no aprendizado de novos soldadores e trazendo maior liberdade de ação e capacidade de aprendizado.

A justificativa do autor para a realização deste projeto também está relacionada com seu relacionamento a empresa *Simulógica*

e ao interesse acadêmico despertado pela oportunidade de aprendizagem no campo da realidade virtual e simuladores, que são inovadores e estão em extrema evidência.

### 1.4 Escopo

Neste trabalho será abordado a integração e calibração de um *tracking* magnético, modelo *MiniBird 500* da *Ascension Technology Corporation, que é composto por um aparelho físico (hardware) e um driver (software) para comunicação,* a um simulador de solda desenvolvido com a *Unity3D* versão 5.3.3, 64 *bits*.

O simulador de soldagem apresentava-se parcialmente desenvolvido no início deste projeto. O desenvolvimento do simulador continuou em paralelo durante todo o processo deste trabalho e não foi completamente concluido. Apesar de estar operacional, com efeitos gráficos, análise parcial da qualdiade da soldagem e simulação física parcialmente implementada.

Este trabalho limita-se a pesquisar e realizar um modelo de projeto que consiga integrar o *tracking* magnético ao simulador de solda, preservando a precisão, que é a proximidade entre os valores obtidos pela repetição do processo de mensuração, e acurácia, que é a proximidade da medida relativamente ao verdadeiro valor da variável, do equipamento e trazendo resultados positivos quanto a atividade de soldagem industrial. A integração deverá preservar todas as características do simulador, inclusive a realidade virtual.

Não faz parte do escopo deste trabalho fornecer um modelo definitivo com relação ao rastreamento do movimento para a solda industrial. Nem a garantia de funcionamento em outras plataformas ou ferramentas, apesar desta possibilidade. A apresentação, descrição e desenvolvimento do simulador de soldagem também não faz parte do escopo deste trabalho.

## 2 Objetivos

### 2.1 Objetivo Primário

O objetivo principal do trabalho descrito neste texto foi incorporar o *tracking* magnético *Mini Bird 500* da *Ascension*

*Technology Corporation*, à *Unity3D* criando um pequeno *framework*.

Este *framework* é, então, incorporado a um simulador de solda industrial com realidade virtual, que utiliza o processo *MIG/MAG*, passando a utilizar não mais *mouse* e teclado e sim a própria tocha de soldagem, permitindo assim, uma maior imersão do usuário com movimentos mais precisos e próximos à realidade e aproximando-se a seu uso de forma real.

## 2.2 Objetivos Secundários

- Levantamento bibliográfico com relação a simuladores, realidade virtual e *tracking* magnético. De modo a conduzir o trabalho e integrá-los.

- Comunicação da ferramenta *tracking* magnético *MiniBird 500* da *Ascension Technology Corporation* à *Unity3D*. Permitindo a inicialização correta do equipamento e o envio de informações do *tracking* magnético à *engine*.

- Interpretação dos dados recebidos do equipamento de t*racking* magnético; possibilitando a translação e rotação dos objetos virtuais na *Unity3D*.

- Integração com o simulador de solda industrial, permitindo a inicialização e comunicação com o aparelho de *tracking* a partir do simulador.

- Ajustes necessários para que, após a integração com o simulador, a tocha de solda virtual realize movimentos realistas acompanhando o *tracking* magnético.

# 3 Procedimento de Identificação de Referências

Com o propósito de fundamentar o presente TCC foi realizada uma revisão bibliográfica do tipo sistemática, onde busca definir uma estratégia e um método sistemático para realizar buscas e analisar resultados, permitindo a repetição por meio de ciclos contínuos até que os objetivos da revisão sejam alcançados. Este é um método científico para busca e análise de

artigos de uma determinada área que além da economia de tempo e recursos, os seus resultados permitem identificar lacunas na teoria que podem ser exploradas por outros pesquisadores, mas que não foram identificadas em estudos semelhantes devido à superficialidade e falta de rigor na revisão bibliográfica. Esta revisão então, será descrita nos itens a seguir.

## 3.1 Revisão Sistemática

> A revisão bibliográfica também conhecida como revisão literária tem como principal objetivo demonstrar o estágio atual da contribuição acadêmica em torno de um determinado assunto proporcionando uma visão ampla das pesquisas e contribuições anteriores, conduzindo ao ponto necessário para investigação e desenvolvimento de futuros estudos. MEDEIROS, REIS, BRAVIANO e GONÇALVES (2014).

Segundo Medeiros, Reis, Braviano e Gonçalves (2014), em todo projeto científico há a necessidade de uma pesquisa bibliográfica como fundamento para a criação de novos conhecimentos.

Ela comprova a relevância acadêmica do trabalho realizado. Pode-se também analisá-la como um trabalho comparativo que permite o progresso do pesquisador em relação ao seu tópico e a avaliação do tratamento dado por outros estudiosos ao mesmo assunto. (MEDEIROS, REIS, BRAVIANO e GONÇALVES, 2014).

Dentre os muitos objetivos da realização de uma revisão bibliográfica pode-se citar o levantamento dos pareceres de outros pesquisadores sobre o estudo a ser pesquisado, identificação de lacunas investigativas e utilização como material de estudo os trabalhos desenvolvidos por outros pesquisadores. (MEDEIROS, REIS, BRAVIANO e GONÇALVES, 2014).

Ainda para Medeiros, Reis, Braviano e Gonçalves (2014), a revisão bibliográfica contribui para o reconhecimento e fornece os devidos créditos à criação intelectual de outros autores e pode-se dizer que é uma questão de ética acadêmica.

Ela também abre um espaço para evidenciar que seu campo de conhecimento já está estabelecido, mas pode e deve

receber novas pesquisas. (MEDEIROS, REIS, BRAVIANO e GONÇALVES, 2014).

Através da revisão de literatura, é possível reportar e avaliar o conhecimento produzido em pesquisas anteriores, destacando alguns conceitos, procedimentos, resultados, discussões e conclusões relevantes para o trabalho. (MEDEIROS, REIS, BRAVIANO e GONÇALVES, 2014).

Um dos principais requisitos para se realizar uma boa revisão bibliográfica é a leitura, análise cuidadosa do material selecionado e sua classificação. (MEDEIROS, REIS, BRAVIANO e GONÇALVES, 2014).

Para Medeiros, Reis, Braviano e Gonçalves (2014), entre a coleta inicial dos dados e o relatório final do projeto (monografia, dissertação ou tese) há o processamento da informação e o pesquisador é quem deve decidir como realizá-lo.

No intuito de alcançar uma organização satisfatória é necessário que o pesquisador organize o material encontrado em diferentes subtópicos que correspondem à apresentação da sua literatura.

> Após a classificação desse material, a apresentação do mesmo pelo pesquisador pode assumir diferentes formatos como a revisão das fontes na qual é analisado as fontes pesquisadas, a revisão teórica, que analisa as principais teses e argumentações encontradas na literatura sobre o assunto, a revisão empírica, na qual o pesquisador procura explicar o assunto a partir das correntes filosóficas e teológicas defendidas pelos principais acadêmicos, a revisão temática, na qual o escritor divide a literatura de acordo com os temas mais relevantes e estabelece conexão com o tema principal do seu trabalho e finalmente a revisão histórica, onde o escritor busca reconstruir o desenvolvimento de um conceito ou abordagem sobre um tema específico. (MEDEIROS, REIS, BRAVIANO e GONÇALVES, 2014).

Cada revisão assumirá um formato distinto de acordo com sua função e objetivo.

Toda revisão bibliografica relacionada a um tópico de pesquisa deve apresentar elementos analíticos e temáticos a fim de que o leitor tenha uma compreensão não apenas da abrangência do que foi encontrado, mas também da maneira como o material foi processado e de como a literatura sobre um tema específico exige nova pesquisa sobre o mesmo tópico. (MEDEIROS, REIS, BRAVIANO e GONÇALVES, 2014).

> A partir de então, surge a necessidade de mensurar o valor científico de publicações utilizadas como referências bibliográficas dando origem a bibliometria que pode ser caracterizada como uma forma quantitativa de avaliar a relevância das publicações selecionadas e isso através de indicadores para nortear o processo de seleção do referencial bibliográfico que melhor se aproxime do interesse do assunto pesquisado. (MEDEIROS, REIS, BRAVIANO e GONÇALVES, 2014).

A bibliometria ainda fornece informações sobre os resultados do processo de investigação, o seu volume, evolução, visibilidade e estrutura, através de uma análise quantitativa de características bibliográficas de um conjunto de publicações (MEDEIROS, REIS, BRAVIANO e GONÇALVES, 2014).

### 3.2 Data e Bases consideradas

As bases utilizadas para a pesquisa com os termos *Unity3D* e Simuladores de Solda (*Welding Simulator*) foram: *Capes, Proquest, Oasisbr, ebrary, ieee, springer***.** A pesquisa foi realizada no dia 10 de agosto de 2016.

Já para a pesquisa com os termos *magnetic tracking*, *virtual reality* e *electromagnetic tracking*, a pesquisa foi realizada em 17 de agosto de 2016 com as bases: *Capes, Proquest, springer e ieee.*

Todas as pesquisas foram feitas utilizando a infra-estrutura da biblioteca universitária da *UFSC* (*Bu-UFSC*).

## 3.3 *Unity3D* e Simuladores de Solda

Buscou-se por *Unity3D* porque é a plataforma que foi utilizada para o desenvolvimento deste trabalho. O objetivo com este termo é descobrir se já existe algum trabalho semelhante sendo feito com este *software*. Ou se, pelo menos, há trabalhos que integrem ferramentas periféricas à plataforma *Unity3D*.

Para Simuladores de Solda, (ou *Welding Simulator*) buscou-se conhecer possíveis técnicas aplicada com realidade virtual e dispositivos periféricos para rastreamento do uso da tocha de soldagem.

### 3.3.1 Termos

Para cada base de dados considerada, procurou-se por títulos que tenham sido publicados entre 2012 e 2016, utilizando os seguintes termos:

#### a. *Unity3D*

Este termo retorna resultados tanto em português quanto em inglês. Com este termo os resultados não são muito numerosos e portanto não é preciso refinar utilizando outras palavras chaves. Para cada resultado retornado lê-se o título e caso seja relevante ao assunto (Simuladores, Integração com dispositivos externos, ou *Tracking* Magnético), reserva-se o título como importante para análise posterior.

Foram encontrados 332 títulos utilizando este termo e separamos 47 para análise posterior.

#### b. Simulador Solda ou Simulador Interativo

Com o termo Simulador Solda por vezes, em algumas bases, não consegue-se nenhum resultado significativo. Por este motivo utilizou-se também o termo Simulador Interativo, buscando encontrar algo que auxiliasse no desenvolvimento deste projeto. Contudo, os resultados encontrados não foram de muita valia ao assunto abordado.

Foram encontrados 283 títulos utilizando estes termos, mas todos eles estavam fora do assunto abordado e portanto nenhum foi selecionado para maiores análises.

**c. Welding Simulator**

Com o termo em inglês obteve-se melhores resultados. Os resultados variaram de simulação de solda não interativa (simulando os dados somente) à simulação interativa. Foram então, separados os títulos que enfatizavam a simulação interativa para maiores análises.

Foram encontrados 248 títulos utilizando esse termo e separados 22 para análise posterior.

### 3.3.2 Filtragem e Resultados

Dos 69 títulos separados para aprofundamento, foram analisados o resumo, a introdução e as conclusões de cada um deles; afim de filtrar e identificar aqueles que tem algo a contribuir ao escopo deste projeto. Para o termo *Unity3D* concentrou-se em títulos com realidade virtual ou com dispositivos periféricos utilizando a *Unity3D*, descartando aqueles que focavam na ferramenta em si ou em jogos/aplicativos interativos sem o uso periféricos. Já o termo *welding simulator* concentrou-se nos títulos que apresentavam simuladores em realidade virtual. O resultado foi o apresentado na Tabela 1.

**Tabela 1** – Revisão Bibliográfica com os termos *Unity3D* e *Welding Simulator*

| **Título** | **Ano** | **Palavras-chave** | **base** | **Termo** |
|---|---|---|---|---|
| Developing virtual reality applications with Unity | 2015 | consumer-level virtual-reality hardware, fully immersive vr experience, | ieee | *Unity3D* |

| | | | | |
|---|---|---|---|---|
| | | intuitive interface, virtual reality, unity game engine | | |
| First person movement control with palm normal and hand gesture interaction in virtual reality | 2015 | Avatar, first person movement control, gesture recognition, head - mounted display, leap motion new technological devices, motion control, oculus rift, simulation character, tracking, unity3d game engine, virtual reality | Ieee | *Unity3D* |
| Interactive virtual building walkthrough using Oculus Rift and Microsoft Kinect | 2015 | kinect rift compatibility, head tracking, object tracking, unity3d standard, virtual reality | Ieee | *Unity3D* |
| Uma ferramenta de interação | 2013 | Ambientes Virtuais, Dispositivos | capes | *Unity3D* |

| | | | | |
|---|---|---|---|---|
| 3D para ambientes virtuais de engenharia utilizando dispositivos móveis | | móveis, Intração 3D, RV | | |
| Application of precise indoor position tracking to immersive virtual reality with translational movement support | 2016 | tracking, unity3d, virtual reality, immersive, indoor position tracking | Springer | *Unity3D* |
| Leap Motion Development Essentials: leverage the power of Leap Motion to develop a fully interactive application | 2013 | leap motion, unity3d, interactive application | Ebrary | *Unity3D* |
| A real-time welding training system base on virtual reality | 2015 | weld, welding simulator track model, training, sensors, mathematical model, head-mounted display, gas | ieee | *Welding simulator* |

| | | | | |
|---|---|---|---|---|
| | | metal arc welding | | |
| Real-time simulation for a virtual reality-based MIG welding training system | 2012 | mig, simulation, virtual reality, welding, training | springer | *Welding Simulator* |

**Fonte**: Elaborada pelo autor.

### 3.4 *Tracking* Magnético e Realidade Virtual

A revisão foi feita para considerar os trabalhos já realizados que utilizaram algum *Tracking* Magnético com ênfase para imersão em Realidade Virtual. Encontrou-se muito material sobre o desenvolvimento de equipamentos que utilizam o campo eletromagnético para rastreamento de posição e rotação; e muito material sobre o *trackers* magnéticos para uso na área de medicina; mas pouco material recente foi encontrado com o foco em Realidade Virtual com ênfase na imersão para Realidade Virtual.

Quando os resultados recentes (de 2013 à 2016) não apresentavam-se muito relevantes à este trabalho, foi buscado então trabalhos de anos anteriores até encontrar artigos interessantes à este objeto.

Como para estes termos encontrou-se muitos resultados para a área de medicina, aplicou-se o filtro *Ciência da Computação* e/ou *Engenharia* quando possível.

Para este termo selecionou-se somente resultados em inglês, já que em português não houve resultado expressivo dentro dos filtros aplicados.

### 3.4.1 Termos

#### a. *Magnetic Tracking AND* Virtual Reality

Buscando por *Magnetic Tracking* com o operador 'E' e o termo *Virtual Reality* para buscar por títulos que contemplem os dois termos; onde fale sobre o assunto rastreador magnético dentro do assunto realidade virtual. Buscou-se também com a variação *Magnetic Tracker* AND *Virtual Reality*.

Foram encontrados 132 títulos utilizando esses termos e separou-se 12 títulos para análise posterior.

#### b. Electromagnetic *Tracking AND Virtual Reality*

Buscando também por *Electromagnetic Tracking*, também em associação com o termo *Virtual Reality*, para buscar por títulos que contemplem ambos os termos. Aplicou-se também a variação *Electromagnetic Tracker*.

Foram encontrados 123 títulos utilizando esses termos e separou-se 17 títulos para análise posterior

### 3.4.2 Filtragem e Resultados

Dos 29 títulos separados para aprofundamento, analisou-se o resumo, a introdução e as conclusões de cada um deles; afim de filtrar e identificar aqueles que tem algo a contribuir ao escopo deste projeto. Concentrou-se então nos títulos com foco no uso e aplicação do *Tracking* Magnético.

**Tabela 2** – Revisão Bibliográfica com os termos *Magnetic Tracking and Virtual Reality* e *Electromagnetic Tracking and Virtual Reality*

| **Título** | **ano** | **Palavras - chave** | **Base** | **termo** |
|---|---|---|---|---|
| Application of precise indoor | 2016 | Magnetic Tracking, Virtual | Springer | Magneti c Trackin |

| | | | | |
|---|---|---|---|---|
| position tracking to immersive virtual reality with translational movement support | | Reality, indoor, position, tracking, immer Sive | | g and Virtual Reality |
| A seamless solution for 3D real-time interaction: design and evaluation | 2014 | Real-time, 3d inter action, evalua tion, kinectic, glove, magnetic | springer | Magneti c Trackin g and Virtual Reality |
| Towards immersive virtual reality (iVR): a route to surgical expertise | 2015 | Immer sive virtual reality Haptic technolog ySurgical simulatio nsSurgi cal learning | springer | Magneti c Trackin g and Virtual Reality |
| A survey on human–computer interaction technologie s and techniques | 2016 | Human–computer interactio nGesture recognitio nRepresentationsR ecognitio | springer | Magneti c Trackin g and Virtual Reality |

| | | | | |
|---|---|---|---|---|
| | | nNatural interfaces | | |
| A survey of electromagnetic position tracker calibration techniques | 2000 | Electro magnetic position trackingTracker calibration Virtual reality | Springer | Electro magnetic Tracking and Virtual Reality |
| Calibration of electromagnetic tracking devices | 1999 | Electro magnetic tracker Tracker calibration Polynomial fitVirtual reality | Springer | Electro magnetic Tracking and Virtual Reality |
| Design of six degrees of freedom electromagnetic tracker for virtual reality system | 2012 | Coils, Magnetic fields, Electromagnetics, Magnetic cores, Magnetic field measurement, Virtual reality, Ferrites | Ieee | Electro magnetic Tracking and Virtual Reality |
| A framework | 2003 | Calibration, | ieee | Electro magneti |

| | | | | |
|---|---|---|---|---|
| for calibration of electromagnetic surgical navigation system | | Surgery, Navigation, Magnetic sensors, Distortion measurement, Magnetic field measurement, Biomedical optical imaging, Position measurement, Needles, Optical distortion | | c Tracking and Virtual Reality |
| Tracker-endoscope calibration for colonoscopy | 2008 | Tracker, endoscope, Calibration, Colonoscopy, Technique | proquest | Electromagnetic Tracking and Virtual Reality |

**Fonte**: Elaborada pelo autor

### 3.5 Conclusão Revisão bibliográfica

Conclui-se que apesar de não ser encontrado muitas obras relacionando os termos *tracker,* realidade virtual e *Unity3D*, foi possível encontrá-los em bons artigos e obras separadamente e assim, realizar o levantamento necessário para elaboração do

trabalho, com o embasamento suficiente para conciliá-los e conseguir, ao final deste projeto bons resultados com relação a utilização destes termos em conjunto.

# 4 Fundamentação

## 4.1 Realidade Virtual

Ambientes virtuais podem proporcionar o desenvolvimento de diversas atividades com grande potencial na simulação e treinamento de inúmeras áreas como medicina, segurança e até mesmo educação. Para Junior, Souza e Machado (2009), ambientes Virtuais proporcionam o desenvolvimento de diversas aplicações com potencial em simulação e treinamento em várias áreas.

> Existe uma extensa quantidade de definições sobre realidade virtual, de forma geral, elas fazem referência a uma imersiva e interativa experiência que se baseia em imagens gráficas 3D geradas por computador em tempo real, em outras palavras, é uma simulação de um mundo real [...] (RODRIGUES e PORTO, 2013, pg 99)

Em resumo pode-se afirmar que a realidade virtual é a simulação do mundo real ou até mesmo imaginário através do computador.

Segundo Araújo (1996 apud Rodrigues e Porto, 2013), o termo Realidade Virtual foi inventado no final da década de 1980 por Jaron Lanier.

Ainda para Araújo (1996 apud Rodrigues e Porto, 2013),

> Este cientista da computação e artista conseguiu afluir dois conceitos antagônicos em um novo conceito diferenciando as simulações tradicionais feitas por computador de simulações envolvendo múltiplos usuários em um ambiente compartilhado.

A Realidade Virtual é um reflexo da realidade física, na qual o indivíduo existe em três dimensões, tem a sensação do tempo real e a capacidade de interagir com o mundo ao seu redor. (RODRIGUES e PORTO, 2013)

Os equipamentos de Realidade Virtual simulam essas condições, chegando ao ponto em que o usuário pode sentir os objetos de um mundo virtual e fazer com que eles respondam, ou mudem, de acordo com suas ações (VON SCHWEBER, 1995 apud RODRIGUES e PORTO, 2013).

A realidade virtual conta com três ideias básicas interligadas: imersão, interação e envolvimento. (RODRIGUES e PORTO, 2013)

**Tabela 3** – Idéias básicas Interligadas da Realidade Virtual

| | **Objetivo** |
|---|---|
| Imersão | Mostrar ao usuário que quando imerso no ambiente virtual, ele pode ter a sensação de estar dentro do ambiente e fazer parte dele. A proporção dessa imersão é captada pelos dispositivos que transmitem ao usuário a sensação de entrada no ambiente virtual, levando seus sentidos sensoriais e sua atenção para o que está acontecendo neste espaço, isolando do mundo exterior e permitindo assim que possa explorar naturalmente os objetos ao invés de ser apenas um observador. |
| Interação | Pode ser descrita como a capacidade do computador de detectar as entradas do usuário e modificar em tempo real o mundo virtual e as ações executadas nele. Para parecer ainda mais realista, o ambiente virtual inclui objetos simulados e a inserção de sons ambientais e sons associados a objetos específicos. |

| | |
|---|---|
| Envolvimento | Envolve a estimulação de uma pessoa com determinada atividade podendo este ser ativo ou passivo. |

**Fonte**: Rodrigues e Porto (2013).

## 4.2 Aplicações de Realidade Virtual

O desenvolvimento de novas ferramentas de realidade virtual é contínuo e tende a se tornar cada vez mais sofisticado por conta dos *hardwares* gráficos que ficam melhores a cada ano.

Diversas áreas já passaram a utilizar a realidade virtual em suas atividades, tendo em vista a necessidade de oferecer sistemas mais realistas. Assim como setores da medicina, engenharia e aviação, vários outros já buscam formas de auxiliar a resolução de muitos problemas com essa tecnologia. (CAMPOS, 2010).

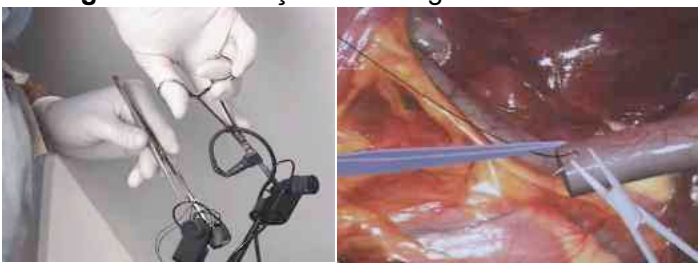
Segundo Campos(2010), a medicina é uma das áreas que mais vêm se beneficiando com os avanços tecnológicos nos últimos anos apresentados pela realidade virtual.

> Especialmente, em cirurgias minimamente invasivas, que envolve uma pequena câmera de vídeo e instrumentos específicos para conduzir a cirurgia. Acredita-se que neste cenário as técnicas de Realidade Virtual e Aumentada podem ter uma significativa parcela de contribuição. Costa e Ribeiro(2009) apud Campos (2010, Pg 7).

Dessa forma, projetos estão sendo desenvolvidos com o auxílio da realidade virtual para realização de cirurgias à distância. Por exemplo, num possível campo de batalha um médico poderia realizar uma cirurgia em um soldado usando técnicas de realidade virtual, como controlando os braços de um robô. (CAMPOS, 2010).

A área da medicina também tem sido uma das áreas de preferência dos desenvolvedores de realidade virtual como mostra a figura a seguir.

**Figura 1 -** Simulação de Cirurgia em Realidade Virtual.



**Fonte:** Rodrigues e Porto (2013).

Segundo Montero e Zanchet (2003) apud Rodrigues e Porto (2013), nos países desenvolvidos, a realidade virtual tem sido empregada no ensino de anatomia e na simulação de operações. As imagens dos moldes virtuais possibilitam avaliar os órgãos tridimensionalmente, além de permitir observar a estrutura interna do órgão entre outras vantagens.

> Devido à tridimensionalidade e a possibilidade de visualizar diferentes pontos de observação, a realidade virtual tem sido considerada uma ferramenta impar no que se refere à percepção, apreciação, avaliação e representação dos elementos arquitetônicos em escala e à simulação de texturas e efeitos luminosos nas superfícies e volumes. (RODRIGUES e PORTO, 2013).

A indústria militar, há anos tem interesse nessa área em decorrência de vários fatores.

> O primeiro, e mais óbvio, é a segurança de treinar um militar em ambiente simulado e controlado que numa situação real. Por um lado considerando a vida do piloto, por outro considerando o custo do equipamento, que geralmente chega a custar milhões de dólares. A Força Aérea, o Exército e a Marinha usam simuladores de voo para treinamento dos pilotos. As missões de treinamento podem incluir: voar em batalha, se restabelecer em caso de emergência ou coordenar a sustentação no ar com operações terrestres. (INFORMATIZAÇÃO MILITAR - QC6, 2001, on-line).

Ainda pode-se descrever um outro exemplo de utilização de realidade virtual em uma das maiores empresas brasileiras, a Petrobrás. De acordo com Costa e Ribeiro (2009, pg 59) apud Campos (2010):

> "É na sede da Petrobrás, empresa brasileira que possui 13 centros de realidade virtual espalhados por suas unidades, que se encontra o que há de mais moderno na área de exploração e produção de petróleo. É por meio dessa tecnologia, os geólogos e geofísicos analisam as propriedades do fundo do oceano, reconhecendo com precisão os pontos onde se deverá perfurar para chegar ao petróleo. Identificados os reservatórios, a realidade virtual também ajuda a aproveitar ao máximo a extração de cada um deles, o que ajuda a economizar tempo e dinheiro."

A Petrobrás é uma das empresas que mais investe na tecnologia e é através da realidade virtual que são feitos os treinamentos e simulações para se chegar aos poços de petróleo (CAMPOS, 2010).

## 4.3 Dispositivos de Realidade Virtual

Quanto ao desenvolvimento da tecnologia de realidade virtual pode-se citar a linguagem *VRML* (*Virtual Reality Modeling Languagem*) ou Linguagem para Modelagem em Realidade Virtual que "trata-se de uma linguagem independente de plataforma que permite a criação de cenários *3D*, por onde se pode passear, visualizar objetos por ângulos diferentes e interagir com eles". (CAMPOS, 2010, pg 4).

Segundo Campos (2010) Há vários dispositivos no mercado como o popular vídeo capacete ou *head-mounted display* (*HMD*) que é um dos dispositivos que mais isola o usuário do mundo, muito usado pela *NASA* e também vários outros projetos em desenvolvimento.

Esses dispositivos são intermediários entre o usuário e a imersão no mundo virtual e possibilitam a interação com o ambiente virtual. (CAMPOS, 2010)

Os dispositivos de entrada e saída de dados juntamente a realidade virtual tem como principal objetivo o estímulo da maior quantidade de sentidos possível e a captura com fidelidade dos movimentos do usuário. Alguns deles serão abordados a seguir:

### a) Dispositivos de saída de dados

O maior objetivo da tecnologia associada a realidade virtual é o isolamento dos sentidos, principalmente a visão. Na maioria das vezes, para acessar esse ambiente virtual é necessário o uso de bons equipamentos para conseguir isolar o usuário do mundo real. Dessa forma, cabe ao *hardware* de realidade virtual de saída de dados estimular tais sentidos. (CAMPOS, 2010)

#### a.1) Dispositivos Visuais

Os dispositivos visuais e a qualidade das imagens produzidas por eles são muito importantes para a percepção do nível de imersão de um sistema de realidade virtual. (CAMPOS, 2010)

Para Rodrigues e Porto (2013), existem duas categorias de dispositivos visuais e uma delas é composta pelos vídeo-capacetes (*HMDs*) e *head-coupled displays* (dispositivos que usam braços mecânicos para permanecerem posicionados na frente do usuário); a outra é composta pelos monitores de computador e sistemas de projeção.

A diferença entre as duas categorias é que, na primeira o dispositivo possui sensores para detectar os movimentos do usuário, enquanto que na segunda isso não ocorre. Com relação ao rastreamento, ele dependerá dos comandos do usuário via outro dispositivo de entrada. (RODRIGUES e PORTO, 2013)

#### a.2) Vídeo-capacetes (HMDs)

O vídeo-capacete (*Head-Mouted Display*, *HMD*) é um dos dispositivos de interface para realidade virtual e um dos mais

populares por se tratar do dispositivo de saída que melhor isola o usuário do mundo real. (CAMPOS, 2010).

Ele é composto basicamente por duas minúsculas telas de *TV* e um conjunto de lentes especiais como mostra a figura 2.

Essas lentes auxiliam no foco das imagens que estão a alguns milímetros dos olhos do usuário, ajudam também na ampliação do campo visual do vídeo. (CAMPOS, 2010).

Além disso, para Rodrigues e Porto (2013), os *HMDs* funcionam também como um dispositivo de entrada de dados, pois contam com sensores de rastreamento que medem a orientação e posição da cabeça, transmitindo os dados para o computador.

Os vídeo-capacetes são construídos, geralmente, usando três tipos de telas: os monitores de *TV* (*CRT*), os monitores de cristal liquido (*LCD*) – os mais utilizados atualmente -, e os de diodo emissores de luz orgânicos (*OLED*) (CAMPOS, 2010).

**Figura 2 -** Exemplo de Dispositivo de Realidade Virtual (Oculus Rift)


**Fonte**: Autoria própria

### *a.3) Head-Coupled Display* (BOOM)

É um *display* montado sobre um braço mecânico com um contrapeso. O formato do *BOOM* permite uma fácil transição entre a visualização do mundo virtual e a interação com monitores, teclados e outros dispositivos que fazem parte do controle da simulação (CAMPOS, 2010).

### a.4) Monitores e Sistemas de Projeção

Nos sistemas de realidade virtual baseados em monitores ou sistemas de projeção, o usuário precisa olhar constantemente para a tela ou monitor além de utilizar um dispositivo de entrada de dados para que possa controlar seus movimentos no ambiente virtual (CAMPOS, 2010).

Como exemplo pode-se citar os óculos obturadores (*shutter glasses*) que são utilizados para filtrar as duplas de imagens geradas pelo computador. Isso significa que o computador exibe, alternadamente, as imagens direita e esquerda sincronizadas com óculos que bloqueiam cada um dos olhos, permitindo ao usuário, visualizar uma imagem fora da tela (CAMPOS, 2010)

## b) Dispositivos Auditivos

O som *3D* tem como objetivo proporcionar a sensação de imersão. Da mesma forma que o ser humano possui visão estereoscópica, ele também possui audição estéreo que desempenha o mesmo papel, apesar de funcionar de forma diferente, do som 3D: enganar o cérebro. Tanto que, em um sistema de som 3D perfeito, não é possível diferenciar simulação de realidade. (RODRIGUES e PORTO, 2013)

Há diversas placas de som projetadas para funcionar juntamente com ferramentas que trabalham com ambientes em realidade virtual. Algumas delas permitem trabalhar simultaneamente com fontes variadas de som. (RODRIGUES e PORTO, 2013).

### c) Dispositivos Físicos

Este tipo de dispositivo busca a estimulação de sensações físicas, como o tato, a temperatura e a tensão muscular. A diferença entre estes e os visuais ou auditivos, é por conta da necessidade de uma sofisticada interação eletromecânica com o corpo do usuário. (RODRIGUES e PORTO, 2013).

A utilização de dispositivos físicos em realidade virtual envolve a usabilidade de sistemas computacionais potentes e dispositivos específicos de entrada e saída. (RODRIGUES e PORTO, 2013).

#### c.1) Reação Tátil

Também denominado *Feedback Tátil* é um tipo de sistemas que transmite sensações que atuam sobre a pele. (RODRIGUES e PORTO, 2013).

Ainda segundo Rodrigues e Porto (2013), esse *feedback* deve fornecer uma sensação de toque e permitir ao usuário distinguir diversas características como rugosidade, temperatura, geometria e características de atrito de superfície associadas ao objeto em que está tocando.

#### c.2 ) Reação de Força

São os sistemas que apresentam ao usuário as sensações de peso e pressão. Uma forma de criar um sistema deste seria uma espécie de braço mecânico encaixado no corpo do usuário, fazendo com determinados movimentos lhe permitisse sentir o peso de um objeto no ambiente virtual. (RODRIGUES e PORTO, 2013).

### d) Dispositivos de Entrada de Dados

Os dispositivos de saída inserem o usuário no mundo virtual, porém são os dispositivos de entrada que permitem os movimentos e interação do usuário. Sem um dispositivo de

entrada o usuário participaria apenas de forma passiva nesta interatividade. (RODRIGUES e PORTO, 2013)

### d.1) Dispositivos de Interação

> Os dispositivos de interação permitem que o usuário se movimente e manipule objetos no mundo virtual. Como exemplo pode-se citar a Luva de Dados que permite que o sistema de realidade virtual reconheça os movimentos das mãos do usuário que a utiliza e também os Sensores de Entrada Biológicos que processam atividades indiretamente, como sinais elétricos musculares e comando de voz. (RODRIGUES e PORTO, 2013)

Na realidade virtual os comandos de voz tendem a facilitar a execução de tarefas no ambiente virtual. Em contrapartida, os dispositivos que utilizam sinais elétricos musculares, detectam essas atividades por meio de eletrodos colocados sobre a pele. (RODRIGUES e PORTO, 2013).

### d.2) Dispositivos de Trajetória

Segundo Rodrigues e Porto (2013), este tipo de dispositivo é responsável pelo rastreamento da trajetória, conhecido também como *tracking*.

Esses dispositivos se baseiam na operação de diferentes posições em relação a um ponto de referência. (RODRIGUES e PORTO, 2013).

Essencialmente, existe uma fonte que emite o sinal (pode estar localizada no dispositivo de interação), um sensor que recebe este sinal, e uma caixa reguladora que processa o sinal e faz a comunicação com o computador. (RODRIGUES e PORTO, 2013)

Sobre a utilização deste sistema Rodrigues e Porto (2013) afirmam que

> Há o auxílio de pequenos sensores que são colocados sobre partes do corpo ou sobre o objeto e é feito assim na maioria das aplicações que usam detecção de trajetória, essa técnica é conhecida como *tracking*

> ativo. Já o *tracking* passivo utiliza sensores óticos ou câmeras para observar o objeto e determinar sua orientação e posição. Em oposição aos dispositivos que utilizam *tracking* ativo, os dispositivos de *tracking* passivo fazem uso de apenas um sensor para rastrear o objeto (RODRIGUES e PORTO, 2013).

### 4.3.1 *Tracking* Magnético

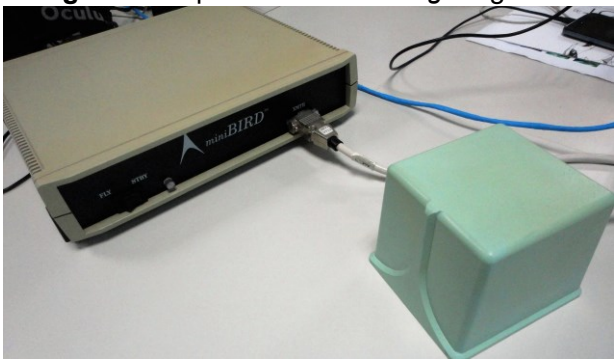
O rastreamento dos movimentos de um usuário é um dos grandes suportes para garantir a real sensação de imersão em um ambiente virtual.

É através dele que o sistema de realidade virtual interpreta os comandos do usuário e responde a eles de forma adequada.

Os chamados *tracking devices*, são os dispositivos com o objetivo de determinar a posição ou a orientação de uma parte do corpo do usuário. Quanto maior a precisão, velocidade e área de ação destes equipamentos, maior será a quantidade de dados disponíveis à interpretação pelo sistema. JUNIOR, SOUZA e MACHADO (2009).

A seguir, na figura 3, é possível visualizar um modelo de *tracker.*

**Figura 3 –** Aparelho de *Tracking* magnético




**Fonte:** Autoria Própria

Para Junior, Souza e Machado (2009), os sistemas de rastreamento precisam atender a requisitos mínimos para que não comprometam o desempenho das aplicações de realidade virtual.

Esses requisitos estão relacionados ao comportamento do dispositivo com relação às diversas interações do usuário e ao estado do sistema quando não há interação.

Para Junior, Souza e Machado (2009), alguns dos principais requisitos que deveriam estar presentes em um sistema de rastreamento ideal são:

**Tabela 4** – Requisitos para Sistemas de Rastreamento Ideal

| | |
|---|---|
| Accuracy (Precisão/Exatidão) | Requer sistemas com alta precisão já que o conteúdo do cenário virtual deve ser mapeado na cena real. Os sistemas ideais deveriam oferecer uma precisão de aproximadamente 1mm para a posição dos objetos e erros de orientação menores que 0.1 graus. |
| Jitter (Perturbação) | Quando não houver movimentação, as informações de rastreamento devem ser constantes. |
| Robustness (Robustez) | Pequenos movimentos devem sempre implicar em pequenas modificações na cena. Movimentos fora do espaço de atuação da aplicação devem ser descartados. Operações de rastreamento devem ser contínuas ao longo do tempo, de forma que não hajam perdas significativas da trajetória rastreada. |
| Mobility (Mobilidade) | Os usuários devem possuir uma mobilidade irrestrita dentro do espaço de atuação |

| | |
|---|---|
| | do sistema. Um sistema ideal seria um em que não se utiliza cabos conectados aos dispositivos, e as partes que compõem o sistema móvel fossem leves. Com o objetivo de oferecer mobilidade total, nenhuma das partes do dispositivo deveria estar atrelada a algum tipo de plataforma. |
| Prediction (Predição) | Uma vez que a renderização de uma nova posição requer algum tempo, a predição de posições futuras se faz necessário. Através desta predição evitamos a inserção de atrasos no rastreamento e apresentação dos objetos, principalmente em momentos em que há uma rápida movimentação do objeto rastreado. |

**Fonte**: Junior, Souza e Machado (2009).

Com relação aos mecanismos de rastreamento, existem inúmeras técnicas e algoritmos que descrevem formas de rastreamento de objetos reais. A diferença entre elas está nos meios físicos utilizados para realizar o rastreamento.

> Em geral, sensores mecânicos, inerciais, acústicos, magnéticos, óticos e de frequência de rádio são utilizados. Cada técnica possui suas vantagens e limitações. As limitações estão relacionadas a fatores físicos, como o seu campo de atuação, processamento dos sinais eletrônicos e design. (JUNIOR, SOUZA e MACHADO, 2009, pg 3)

Ainda segundo Junior, Souza e Machado (2009), é possível classificar os tipos de rastreadores com relação aos sensores que os dispositivos utilizam para fazer o rastreamento.

A descrição de alguns tipos de sensores pode ser observada a seguir:

**Tabela 5** – Tipos de sensores

| | |
|---|---|
| Sensores mecânicos | Envolvem normalmente alguma forma de ligação física direta a entidade rastreada e o ambiente. Para Junior, Souza e Machado(2009), as construções mais típicas envolvem uma série articulada de duas ou mais partes mecânicas interligadas e transdutores eletromecânicos, como por exemplo, potenciômetros. Utilizando um conhecimento sobre as partes mecânicas rígidas e as medições em tempo real dos transdutores é possível estimar a posição da entidade rastreada em relação ao ambiente à medida que se movimentam |
| Sensores Inerciais | Estes oferecem baixa latência e conseguem mensurar a posição a taxas relativamente altas (milhares de amostras por segundo), além de conseguirem estimar, através da velocidade e aceleração, a posição de uma cabeça ou de uma mão 40 ou 50 ms no futuro. Bons sensores inerciais também oferecem pouco ruído. (JUNIOR, SOUZA e MACHADO, 2009) |
| Sistemas Acústicos | Utilizam-se da transmissão por ondas sonoras. Todos os |

| | |
|---|---|
| | sensores acústicos comerciais funcionam através da medição da duração de um breve pulso ultrassônico. (JUNIOR, SOUZA e MACHADO, 2009) |

**Fonte**: Junior, Souza e Machado (2009).

Existem muitos outros sensores que podem ser utilizados para o desenvolvimento de dispositivos de rastreamento espacial. Em uma lista destes sensores podería-se incluir sensores mecânicos, inerciais e acústicos e também ópticos, rádio, microondas além dos sensores magnéticos. (JUNIOR, SOUZA e MACHADO, 2009).

> Os rastreadores magnéticos são dispositivos que utilizam um conjunto de bobinas em um transmissor para gerar campos magnéticos que terão seu tamanho e direção calculados pelos sensores, os quais também apresentam bobinas ao longo de três eixos ortogonais. Com a passagem de corrente elétrica pelas bobinas do emissor, um campo magnético é criado e uma corrente elétrica é gerada no sensor com intensidade proporcional ao campo e inversamente proporcional à distância, permitindo o cálculo da posição e orientação do sensor em relação ao emissor. (JUNIOR, SOUZA e MACHADO, 2009, pg 3)

Segundo Junior, Souza e Machado (2009), os rastreadores magnéticos têm sido a tecnologia mais utilizada para rastreamento em ambientes virtuais.

O conceito por trás do rastreamento por campos magnéticos é de que se um fio é submetido à um campo magnético então uma corrente elétrica é gerada neste fio (JUNIOR, SOUZA e MACHADO, 2009).

Para Pinho (2009), a intensidade desta corrente é proporcional à intensidade do campo magnético e inversamente proporcional ao alinhamento entre o campo e o fio.

Nos rastreadores disponíveis no mercado, os emissores são peças fixas e os receptores ficam presos ao ponto a ser rastreado. (JUNIOR, SOUZA e MACHADO, 2009)

> Do ponto de vista tecnológico, entretanto, o oposto é totalmente viável. Os maiores problemas apresentados pelos rastreadores magnéticos são o tempo que gasto para o cálculo de uma nova posição e as interferências causadas por objetos de ferro e por outras fontes de campos magnéticos próximas ao emissor ou ao receptor, como monitores, e caixas com alto-falantes. (PINHO, 2009)

*Motion Tracking* pode ser considerado como um processo para se obter as coordenadas de objetos em movimento e em tempo real.

Em diversos casos a posição e a orientação destes objetos são recuperadas compondo sistemas de interação com seis graus de liberdade. (JUNIOR, SOUZA e MACHADO, 2009)

Ainda segundo os autores Junior, Souza e Machado (2009), os sistemas de rastreamento devem rastrear todos os objetos desejados sem causar atrasos na geração destas cenas.

Algumas das principais áreas onde tais sistemas podem ser utilizados são: controle de visualização, navegação, rastreamento de instrumentos, seleção e manipulação de objetos e animação de avatar (JUNIOR, SOUZA e MACHADO, 2009).

Ambientes virtuais conseguem apresentar o desenvolvimento de inúmeras aplicações com alto potencial de simulação e observação de forma a cobrir diversas áreas do conhecimento. A interatividade é um dos fatores decisivos para a construção de tais ambientes (JUNIOR, SOUZA e MACHADO, 2009).

Ainda segundo Junior, Souza e Machado (2009), uma ferramenta que se proponha a oferecer um modo automatizado para a construção deste tipo de ambiente deve oferecer meios para que formas variadas de interação possam ser integradas.

Uma das ferramentas que se destaca por sua versatilidade, fácil utilização e com grande potencial para utilização deste recurso em projetos chama-se *Unity3D* e será vista a seguir:

## 4.4 *Unity3D*

Segundo Signori e Rigo (2012), a *Unity3D* é uma ferramenta de desenvolvimento de jogos multiplataforma com suporte a *Microsoft Windows, Mac OS, Web, Wii* e *Iphone*.

Pode-se afirmar que O *Unity Engine* é um motor gráfico muito simples em sua utilização e que se destaca pelos projetos principalmente em *3D*, contando com uma boa imagem gráfica (SIGNORI e RIGO, 2012).

Ela oferece um sistema de script compreensivo e flexível, possibilitando o desenvolvimento de todas as partes de um projeto sem precisar utilizar especificamente a linguagem de programação C/C++ (SIGNORI e RIGO, 2012).

A *Unity3D* está disponível em duas versões: uma versão gratuita que contém diversas funcionalidades, porém algumas ferramentas são restritas e uma versão paga, que engloba todas as funcionalidades da versão já criada (SIGNORI e RIGO, 2012).

Ainda para Signori e Rigo (2012), a *Unity* conta com diversos tutoriais e exemplos de funcionalidades e ferramentas, além de uma documentação completa.

Conta ainda com diversos fóruns e *blogs* que possibilitam aos desenvolvedores da ferramenta, o compartilhamento de diversas experiências e informações (SIGNORI e RIGO, 2012).

A *Unity3D* possui uma interface simples e completa de utilização, com janelas conhecidas como *Views* onde cada uma delas terá uma funcionalidade específica (SIGNORI e RIGO, 2012).

> A *view* denominada *Project View*, é a responsável pela organização dos arquivos que fazem parte do projeto, tais como modelos, texturas e efeitos sonoros.
>
> Já a *HierarchyView*, possibilita a ordenação e visualização de todos os objetos que fazem parte do jogo.
>
> A *Game View*, é uma *view* que permite a visualização final do projeto de forma que nele possa ser analisado todos os elementos presentes no jogo (SIGNORI e RIGO, 2012).

A *Unity3D* trabalha com os conceitos de orientação a objetos (classes, objetos, métodos, etc.) e permite que diversos

atributos sejam adicionados ou removidos dos objetos que compõe o cenário do jogo.

Segundo Signori e Rigo (2012), cada um destes atributos são implementados por um componente, ou seja, via herança, um objeto de uma classe herda atributos e comportamentos de um componente.

Segundo Signori e Rigo (2012), a *Unity3D* possui alguns componentes importantes e que facilitam o desenvolvimento de um projeto e são eles:

- PREFABS

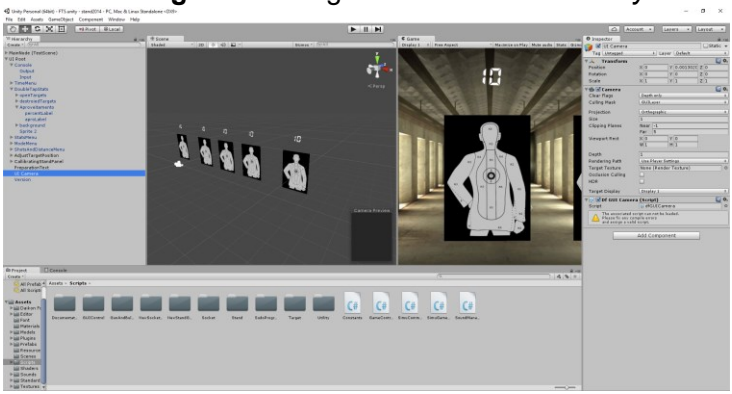
A *Unity3D* oferece modelos pré-definidos, que definem um elemento através de um grupo de vários componentes. Este conceito é conhecido como *Prefab*.

- ASSETS

Os Assets podem ser modelos *3D*, *scripts*, efeitos de som e texturas, ou seja, diferentes tipos de recursos que podem ser vinculados aos objetos. Os Assets são elaborados em ferramentas externas e posteriormente importados para a *Unity 3D*. Esta ferramenta oferece uma forma completa para importação destes Assets, necessitando apenas arrastar para uma pasta do projeto, onde a importação é feita de maneira automática.

A seguir, é apresentado uma figura que representa a interface da *Unity3D*.

**Figura 4 -** Imagem da Interface da *Unity3D*



**Fonte:** Autoria Própria.

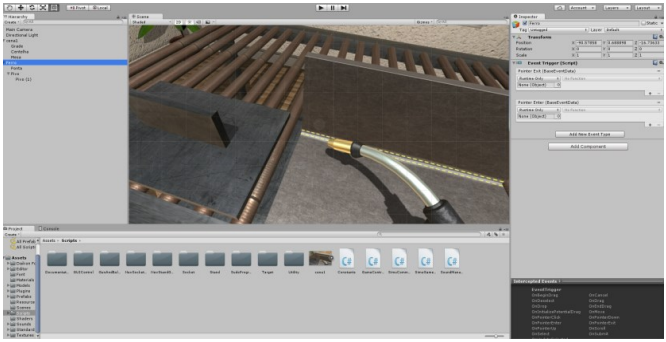
A *Unity3D* permite que os *scripts* sejam implementados em três linguagens, à escolha do desenvolvedor, podendo optar por *C#, Javascript ou Boo* (SIGNORI e RIGO, 2012). Porém é importante salientar que

> Existem várias tecnologias para suportar o desenvolvimento de sistemas de realidade Virtual, entre elas linguagens de programação como *VRML* e *X3D*, bibliotecas gráficas como *OpenGL* e *Java 3D* além das *engine* de jogos como *Unity 3D,* que vêm se destacando devido à alta produtividade, às diversas plataformas de desenvolvimento e à possibilidade de criação e reutilização de pacotes de customização para diversos tipos de soluções (SILVA, Alexandre Carvalho et al., 2013, pg 2)

Segundo Cardozo (2015), os números das últimas *Global Game Jam*[1] 2014 e 2015 mostraram que pelo menos 50% dos jogos utilizaram o *Unity*. Esses números refletem no Brasil, pois a *Unity* é disparado a *engine* mais utilizada.

A seguir, é possível visualizar um exemplo de aplicação realizada em *Unity3D* com a utilização da realidade virtual.

**Figura 5 -** Exemplo de Aplicação na *Unity3D* que Utiliza Realidade Virtual



**Fonte:** Autoria Própria

[1] É um evento internacional onde participantes se reúnem para desenvolver ideias, formar pequenos grupos e criar novos jogos que sejam criativos e inovadores para então, apresentá-los aos seus colegas da comunidade global, tudo em um período de tempo de 48 horas.

Para SILVA, Alexandre Carvalho et.al. (2013), é possível citar alguns trabalhos relacionados ao desenvolvimento de aplicações baseadas em técnicas de realidade virtual que utilizaram a *engine Unity* para o desenvolvimento de sistemas voltados a treinamento e simulação no contexto de *Serious Games*.

- *Designing a Game for Occupational Health and Safety in the Construction Industry*

A pesquisa enfatiza a utilização de um ambiente virtual para treinamento de trabalhadores da construção civil, tendo como intuito a redução de lesões e riscos de acidentes de trabalho.

O sistema destina-se a ajudar o usuário a aprender sobre os riscos no local de trabalho e manter seu conhecimento sobre os procedimentos de segurança auxiliando no processo real. (SILVA, Alexandre Carvalho et al., 2013)

- *An Easy to Author Dialogue Management System for Serious Game*

O trabalho descreve uma solução para ambientes virtuais que possuam atividades de gestão de diálogo em personagens virtuais (*avatar*) empregados em *Serious Games*, tendo como objetivo melhorar a experiência de aprendizagem, aumentando a sensação de imersão e envolvimento do usuário.

Para validar a ferramenta, os autores desenvolveram um ambiente simples onde o usuário pode interagir com representação virtual de um artista a fim de adquirir conhecimento sobre sua vida e suas obras de arte. (SILVA, Alexandre Carvalho et al., 2013)

- *Emergency Medicine Training with Gesture Driven Interactive 3D Simulations*

O trabalho proposto apresenta um sistema protótipo utilizado para simulação e treinamento de equipe médica em medicina de emergência. A pesquisa relata que o uso de simulações imersivas na formação médica é de extrema utilidade para enfrentar cenários de emergência que vão desde o habitual ao extremo, sem colocar os participantes da simulação em risco. O protótipo conta com cenários *3D* interativos e uma *interface* natural baseada em gestos. (SILVA, Alexandre Carvalho et al., 2013)

Ainda para Silva, Alexandre Carvalho et al. (2013) a *engine Unity* correspondeu com eficiência e eficácia nos requisitos envolvidos na construção de sistemas de realidade

virtual voltados ao controle e treinamento, se mostrando razoavelmente flexível às adequações e possibilitando a criação de sistemas de *interface* gráfica *3D* em diferentes plataformas de desenvolvimento.

Um outro exemplo que pode ser citado com relação a desenvolvimento utilizando a *Unity3D* é um jogo, voltado ao desenvolvimento de Portadores da Síndrome de Down denominado *Salve os Mini Downs* onde, segundo o próprio tema,

> é um jogo para desenvolvimento de características cognitivas em portadores de síndrome de *down* e que pode apresentar uma nova forma de auxílio na educação de portadores da síndrome no reconhecimento de cores com utilização de sons e imagens interagindo com o usuário de forma lúdica SILVA, MACHADO e SOARES (2013).

A figura 6 apresenta o jogo ainda em desenvolvimento na *Unity3D*.

**Figura 6 –** Jogo Salve os Mini Downs em Desenvolvimento na *Unity3D*

**Fonte:** Silva, Machado e Soares (2013).

## 4.5 Simuladores

Segundo Reis (2016, pg 49), “O realismo dos simuladores demanda o alinhamento entre dois fatores: os comandos motores emitidos pelo usuario e as consequencias de tais comandos no ambiente virtual, em termos que geram *feedback* sensorial e motor ao usuario.”

O simulador permite que o usuario emita comandos usando todo seu corpo e em outras palavras, o artefato utilizado reproduz, como por exemplo em um simulador de motocicleta, os pedais, guidão, alavancas, botões nos painéis, etc. (REIS, 2016)

Tais comandos corporais afetam a realidade virtual na qual o usuário esta imerso atraves de estimulação audiovisual. (REIS, 2016).

### 4.5.1 Definição de Simulador

Entende-se neste trabalho como simulador uma máquina que reproduz o comportamento de um sistema sob determinadas condições permitindo que o usuário desse sistema pratique antes de iniciar a mesma tarefa na vida real.

> Os simuladores costumam combinar o mecânico ou eletrônico com parte virtual que simula a realidade e podem ser usados no meio profissional ou como um meio de lazer ou entretenimento. Tornam-se indispensáveis para a formação de pessoas que terão uma grande responsabilidade a seu cargo, uma vez que os seus eventuais erros poriam em risco a sua vida e de terceiros. (BALADEZ, 2009)

Ainda segundo Baladez (2009), o computador como peça chave para construir um simulador permite o uso de diversos e sofisticados recursos gráficos, matemáticos e lógicos para conseguir chegar a um modelo de sistema que proporcione ao mesmo tempo, flexibilidade, agilidade e confiabilidade para o projeto.

A seguir, na figura 7, é possível visualizar um exemplo de simulador que traz essas características.

**Figura 7 –** Testador Utilizando um Simulador

**Fonte:** Reis (2016).

A simulação permite a análise e o teste de diferentes alternativas de funcionamento de um sistema, com variações de diversos parâmetros, formas de controle, prioridade de eventos, diversos tipos de equipamentos e velocidades, ciclos de trabalho e *layouts* ( BALADEZ, 2009).

Variações aleatórias em processos e ciclos de trabalho podem ser incorporadas ao modelo fazendo com que se aproxime ao máximo do funcionamento real dos sistemas.

Para Baladez (2009), a utilização de simuladores pode resolver alguns problemas de desenvolvimento como gargalos e limitações de capacidade que podem ser revelados ainda durante a fase de concepção e projeto.

Soluções alternativas para estes problemas podem ser geradas, testadas e certificadas, garantindo o desempenho muito antes da implementação física ( BALADEZ, 2009).

Hoje em dia, a utilização de simuladores é crescente. Para Baladez (2009), há simuladores para treinar vários tipos de habilidades não combativas.

> “O simulador é um grande aliado da educação e do treinamento, portanto pode ser usado para desenvolver diversos tipos de conhecimentos e habilidades, tais como a

> educação em língua e cultura estrangeiras." (BALADEZ, 2009, pg 9).

Em resumo, o uso de simulador pode ser apresentado em diversas áreas e com sua utilização é possível abranger diversas atividades que poderiam ser alcançadas no cotidiano porém com maiores dificuldades.

### 4.5.2 Vantagens de Uso

Graças ao simulador é possível treinar até que se adquira a experiência e destreza necessárias para desempenhar profissionalmente a sua função. A simulação é usada tanto para o treinamento de aprendizes como para operadores experientes proporcionando-lhes constante revisão de processos e evitando assim situações arriscadas.

> "O treino repetitivo torna o jogador habilidoso nesse aspecto, que é basicamente um treino de seus reflexos e de sua capacidade de tomar as decisões corretas. O simulador permite variações de ambientes, situações e níveis de dificuldade e repetições que não seriam possíveis em uma situação real." (BALADEZ, 2009, pg 8)

Pode-se afirmar que o uso da simulação reduz riscos de novas implementações e dá suporte a decisões complexas que por vezes envolve investimentos elevados.

Para Schitcoski apud Baladez (2009, pg 21):

> "O Orçamento de 2003 para a Seção de Defesa Nacional dos Estados Unidos especificou uma verba de US$ 10 bilhões para treinamento. Para o Pentágono, são destinados anualmente US$ 4 bilhões para equipamentos de simulação e jogos de guerra."

Segundo Baladez (2009), muitas vezes um simulador pode alcançar um valor elevado para o seu desenvolvimento, e mesmo alguns valores elevados podem ser compensadores quando experimentar em uma situação real requer um custo ainda mais elevado.

Para Rodrigues e Porto (2013), a todo o momento aplicações novas surgem, devido à demanda e capacidade

criativa das pessoas através da realidade virtual, a interação homem-maquina mudou.

Devido ao avanço tecnológico de *hardwares* e *softwares*, a utilização de recursos de realidade virtual vem propiciando as empresas maior desempenho e menores custos. (RODRIGUES e PORTO, 2013)

### 4.5.3 Aplicações de Simuladores

A seguir algumas principais áreas de aplicações:

#### a) Educação e Treinamento

Para aprender a fazer é preciso combinar interface com mundo real, e isso pode ser estruturado através de um simulador. Com a simulação, há uma integração do aprendizado à vida fazendo com que o aluno possa de fato encarar o conteúdo de aprendizado formal como uma quebra em sua rotina diária. Há muitos estudos em áreas diversificadas que mostram a interação de aluno e simulador trazendo resultados positivos e significativos ao aprendizado.

> Quanto mais relevante para a vida do estudante for o conhecimento adquirido, mais capacidades podem ser desenvolvidas de forma efetiva no processo de aprendizagem. Cada estudante faz uma filtragem dos conteúdos que têm significado ou não para si próprio (VARGA, Cássia Regina Rodrigues et al., 2009)

A prática torna-se importante tanto para aprimorar o conhecimento quanto para executar a tarefa corretamente.

Para Sharif e Masoumi (2005), os problemas na prática dos alunos são fatores frequentes na educação em enfermagem, e estão relacionados à ansiedade durante a supervisão e a avaliação.

As simulações computacionais contribuem para a qualidade dos cuidados oferecidos aos pacientes.

> “Há programas educativos e jogos online que permitem o treinamento de procedimentos clínicos em situações de simulação, sendo frequente o desenvolvimento de objetos

> digitais para esse fim.” (Schatkoski et al., 2007; Melo, Damasceno, 2006; Barbosa, Marin, 2000 apud Teixeira e Felix, 2011, pg 12).

Pode-se também destacar, com relação a área de treinamentos e educação, o uso de simuladores de trânsito para treinamento de novos motoristas.

Para Ribeiro (2006), o desenvolvimento de um cenário virtual, traz benefícios, como recriar cenários de difícil acesso (temporários e perigosos), criar um espaço virtual com a ocorrência de problemas e situações contextualizadas.

Na educação do trânsito, por exemplo, a possibilidade de criar ambientes com esses recursos favorece a aprendizagem, a interação sujeito-ambiente, como, por exemplo, simular situações perigosas e, então, verificar a reação do sujeito – isso em um ambiente controlado e sem riscos. (RIBEIRO, 2006).

De acordo com Hancock (2009) apud Reis (2016, pg 48), “um simulador veicular consiste em qualquer dispositivo capaz de reproduzir as condicões de uso de um veículo em um cenário virtual, o que inclui uma interação ambiental do usuário com um artefato físico.”

Segundo Reis (2016), destaca-se a “necessidade de projetar cenários ricos em problemas de trânsito para treino do aluno, e com uma variedade satisfatória dos mesmos (diferentes tipos de vias, cenários urbanos e rurais, variações de condições climática, dia e noite, etc.)” (REIS, 2016, pg 102).

Para Liu (2011) apud Reis (2016, pg 49), “simuladores prezam pela naturalidade de uso uma vez que os comportamentos necessários para isso devem ser semelhantes ou mesmo idênticos aos usados no veículo que imitam.”

Para LUCAS, Felipe Rabay et al. (2013), Os simuladores de direção oferecem a possibilidade de condução de experimentos, apresentado um realismo natural, com a vantagem do controle das diversas variáveis do estudo, sem risco aos usuários.

### b) Medicina

Um dos ambientes em que mais se utiliza simulação para ensino-aprendizagem é a medicina.

> “O processo de aprendizagem por meio de situações simuladas tem se mostrado um método útil e efetivo para avaliar desempenhos e habilidades clínicas, pois permite controle de fatores externos, padronização dos problemas apresentados pelos pacientes e *feedback* positivo para os alunos, aumentando o autoconhecimento e a confiança destes.” VARGA, Cássia Regina Rodrigues et al. (2009, pg 4)

Dá oportunidade, ainda, para que a aprendizagem clínica seja centrada no paciente, garantindo melhor relacionamento interpessoal, resolução de problemas e análise e síntese das informações clínicas, mesmo sem a utilização de pacientes reais.

Há algumas simulações que utilizam manequins, facilitando a aquisição de novas habilidades com relação aos cuidados dos pacientes, proporcionando a imersão dos estudantes em ambientes interativos seguros para o desenvolvimento de processo na enfermagem.

Para Gomes e Germano (2007 apud Teixeira e Felix, 2011), considerando que a segurança do paciente é imprescindível, inicialmente, os acadêmicos participam de aulas teóricas e práticas, com simulações no laboratório de enfermagem; posteriormente, desenvolvem os primeiros cuidados em instituições de saúde.

O laboratório de enfermagem é um recurso com estrutura para a aprendizagem, que dispõe de equipamentos e materiais simuladores para o desenvolvimento de habilidades profissionais, tais como: avaliação do paciente, desempenho psicomotor, pensamento crítico para a solução de problemas e colaboração Interdisciplinar (ROTHGEB, 2008 ; COUTINHO e FRIEDLANDER, 2004 apud Teixeira e Felix, 2011).

Para Barbosa e Marin apud Teixeira e Felix (2011, pg 3)

> “No Brasil, as pesquisas sobre simulação estão na fase inicial; mas os docentes e os discentes têm demonstrado atitudes favoráveis ao uso dessa estratégia para o ensino de procedimentos e técnicas de enfermagem. “

### c) Segurança

O uso de simuladores é conhecido e utilizado há tempos pela sociedade. Um dos jogos que foi considerado há muito tempo como um simulador e também um dos mais conhecidos e antigos é o Xadrez. Atualmente ele é utilizado para o entretenimento mas no passado foi peça importante para o treinamento de comandantes do exército antigo, sendo assim, uma simulação de combate.

Segundo Baladez (2009, pg 3), “era um simulador de batalhas que enfatizava a estratégia”. No xadrez, o tabuleiro é uma representação do campo de batalha e suas peças representam as forças beligerantes envolvidas em uma batalha medieval.

> “Os simuladores sempre tiveram papel importantíssimo, muito antes de serem desenvolvidos em computadores, fato inconteste que creio estar suficientemente compreendido, mas que merece o exemplo dos simuladores de veículos. Estes simuladores são ferramentas essenciais no treinamento civil e militar.” (BALADEZ, 2009, pg 6)

Para Baladez (2009), A reprodução dos controles dos aviões auxilia no treinamento de pilotos, tornando possível a prática sem riscos. Os simuladores de vôo também tem grande aceitação pelo público em geral, e tem sua eficiência consagrada pela História.

> As guerras modernas tiveram novas exigências. Na Primeira Guerra Mundial, um grande número de pilotos deveria ser treinado em pouco tempo. Havia um grande problema nisso, pois o número de acidentes era grande. Perdia-se o piloto, o avião e todo o tempo – que não fora suficiente – de treinamento dado a este piloto. Simuladores mecânicos foram então criados para oferecer treinamento prático para os pilotos, o que fazia com que eles tivessem mais segurança e alguma experiência quando pilotassem aviões reais (BALADEZ, 2009).

Alguns jogos oferecem grande sofisticação gráfica, os cenários podem conter muitos detalhes. Mas em jogos com o intuito de treinamento militar é necessário sempre ter o essencial, o excesso pode ser inútil ou até mesmo prejudicial (BALADEZ, 2009).

Há sombras, árvores, locais escuros, pontos estratégicos onde um inimigo pode se situar. Pode-se encontrar também civis, barris com material inflamável etc. A tensão se amplifica em momentos de combate, e é imprescindível que um soldado consiga diferenciar seus alvos, não atirar em civis ou em seus próprios parceiros (BALADEZ, 2009).

## 4.6 Processo de Soldagem

A soldagem é considerada um método de união, porém, muitos processos de soldagem ou variantes são usados para a colocação de material sobre uma superfície, visando a recuperação de peças com desgastes ou formação de um revestimento com características especiais (MODENESI e MARQUES, 2006).

Ainda segundo Modenesi e Marques (2006), existem mais algumas definições de soldagem:

a. Processo de Junção de Metais por Fusão, lembrando que não só metais são soldáveis e que é possível soldar metais sem fusão.
b. Operação que visa obter a união de duas ou mais peças, assegurando, na junta soldada, a continuidade de propriedades físicas, químicas e metalúrgicas.
c. Operação que visa obter a coalescência localizada produzida pelo aquecimento até uma temperatura adequada, com ou sem a aplicação de pressão e de metal de adição.
d. Processo de união de materiais baseado no estabelecimento, na região de contato entre os materiais sendo unidos, de forças de ligação química de natureza similar às atuantes no interior dos próprios materiais.

Segundo Paes (2015), a crescente demanda por processos de soldagem mais produtivos tem incentivado o desenvolvimento de sistemas mecanizados, e até mesmo automáticos.

No entanto, uma significativa parcela das aplicações industriais no setor ainda faz uso do operador para a execução dos procedimentos.

Além de todos os problemas relacionados à saúde e à produtividade, a qualidade do produto final muitas vezes é prejudicada e um exemplo disso relaciona-se com a soldagem *MIG/MAG* convencional semi-automática. (PAES, 2015)

Para PAES (2015), as inevitáveis variações da Distância Bico de Contato-Peça (*DBCP*), advindas da dificuldade de acesso em geometrias complexas e soldagem fora de posição, acarretam alterações no processo e na solda.

Em determinadas situações, tal fato é plenamente aceitável. Em alguns sensores de segmento de junta, não só é aceitável, mas também necessária. Porém, em aplicações de responsabilidade, as avaliações são mais rigorosas e os defeitos oriundos destas alterações são decisivos na etapa de qualificação. (PAES, 2015).

### 4.6.1 Tipos de Soldagem

#### a) Soldagem por pressão ou deformação

Este primeiro grupo inclui os processos de soldagem por ultrassom, por fricção, por forjamento, por resistência elétrica, por difusão, por explosão, entre outros (MODENESI e MARQUES, 2006).

Diversos destes processos, como por exemplo, os processos de soldagem por resistência, apresentam características intermediárias entre os processos de soldagem por fusão e por deformação (MODENESI e MARQUES, 2006).
A seguir é caracterizado alguns desses processos:

#### a.1) Soldagem por resistência

A soldagem por resistência (*Resistance Welding, RW*) compreende um grupo de processos de soldagem nos quais o calor necessário à formação da junta soldada é obtido pela resistência à passagem da corrente elétrica através das peças sendo soldadas. (MODENESI e MARQUES, 2006).

### a.2) Soldagem por centelhamento

A soldagem por centelhamento *(Flash Welding, FW*) é muitas vezes classificado como um processo por resistência pois apresenta diversas características e aplicações similares à soldagem de topo por resistência (*UW*) (MODENESI e MARQUES, 2006).

Nesta soldagem, as peças a serem soldadas são aproximadas sem, contudo, as suas superfícies entrarem em contato.(MODENESI e MARQUES, 2006).

### a.3) Soldagem por alta frequência

Na soldagem por alta frequência *(High Frequency Induction Welding, HFIW*), são utilizadas bobinas por onde passa uma corrente de alta frequência que causa o aparecimento de correntes induzidas na região da junta das peças que estão sendo soldadas. (MODENESI e MARQUES, 2006).

Ainda para Modenesi e Marques (2006),

> estas corrente aquecem a junta por efeito *Joule* o que facilita a deformação localizada e a formação da solda com a aplicação de pressão. Desta forma, este processo apresenta grande semelhança com a soldagem *RW*, sendo considerado, por diversos autores, como um processo de soldagem por resistência (MODENESI e MARQUES, 2006).

### a.4) Soldagem por fricção

Segundo Modenesi e Marques (2006), a soldagem de fricção (*Friction Welding, FW*) é um processo que utiliza energia

mecânica, em geral associada com a rotação de uma peça, para a geração de calor na região da junta a ser soldada.

### a.5) Soldagem por difusão

A soldagem por difusão (*Diffusion Welding, DFW*) é um processo de união no estado sólido que produz a solda pela aplicação de pressão a elevada temperatura sem a deformação macroscópica das peças.

Para Modenesi e Marques (2006), a soldagem por difusão é um processo especializado de soldagem de aplicação restrita quando deseja-se:

(a) evitar problemas metalúrgicos associados com a soldagem por fusão,

(b) fabricar componentes de dimensões e forma próximas das desejadas no produto final (*net shape*), e

(c) produzir peças espessas com propriedades uniformes ao longo da espessura.

> O processo só é economicamente viável quando materiais especiais e de elevado custo são utilizados ou quando existe uma grande exigência quanto às dimensões da peça soldada, tendo suas aplicações sido, até o presente, limitadas, em geral, às indústrias eletrônica e aeroespacial (MODENESI e MARQUES, 2006).

### a.6) Soldagem por explosão

Para Modenesi e Marques (2006), a soldagem por explosão (*Explosive Welding, EXW)* é um processo que utiliza a energia de detonação de um explosivo para promover a união de peças metálicas.

Uma das peças é lançada ao encontro da outra pela explosão e, durante a colisão, desenvolve-se uma intensa deformação plástica superficial capaz de remover as contaminações superficiais e promover a união das peças (MODENESI e MARQUES, 2006).

### a.7) Soldagem por laminação

Este é um processo de união usado para a produção de chapas bimetálicas através da laminação conjunta (colaminação) de chapas de metais diferentes, em geral, à temperatura ambiente ou a temperaturas próximas desta.(MODENESI e MARQUES, 2006).

### a.8) Soldagem a frio

Segundo Modenesi e Marques (2006), a soldagem a frio (*Cold Welding, CW*) é realizada pela aplicação de uma forte deformação localizada, à temperatura ambiente, das peças a serem unidas. Este processo é aplicável para metais de elevada dutilidade, como o alumínio e cobre, tendo, como aplicação típica, a união de condutores de eletricidade.

### a.9) Soldagem por ultrassom

A soldagem por ultrassom *(Ultrasonic Welding, USW*) produz a união das peças pela aplicação localizada de energia vibracional de alta frequência (ultrassom), enquanto as peças são mantidas sob pressão.(MODENESI e MARQUES, 2006).

## b) Soldagem por fusão

Segundo Modenesi e Marques (2006), existe um grande número de processos por fusão que podem ser separados em subgrupos de acordo com o tipo de fonte de energia usada para fundir as peças.

Dentre estes, os processos de soldagem a arco onde a fonte de energia é o arco elétrico são os de maior importância industrial atualmente. (MODENESI e MARQUES, 2006)

Ainda para Modenesi e Marques (2006), devido à tendência de reação do material fundido com os gases da atmosfera, a maioria dos processos de soldagem por fusão utiliza algum meio de proteção para minimizar estas reações.

A tabela a seguir mostra os principais processos de soldagem por fusão e suas principais características:

**Tabela 6** – Principais Processos de Soldagem por Fusão e suas Principais Características

| PROCESSO | FONTES DE CALOR | TIPO DE CORRENTE E POLARIDADE | AGENTE PROTETOR OU DE CORTE | OUTRAS CARACTERÍSTICAS | APLICAÇÕES |
|---|---|---|---|---|---|
| Soldagem por eletro-escória | Aquecimento por resistência da escória líquida | Contínua ou alternada | Escória | Automática/Mecanizada. Junta na vertical. Arame alimentado mecanicamente na poça de fusão. Não existe arco | Soldagem de aços carbono, baixa e alta liga, espessura ≥ 50 mm. Soldagem de peças de grande espessura, eixos, etc. |
| Soldagem ao Arco Submerso | Arco elétrico | Contínua ou alternada. Eletrodo + | Escória e gases gerados | Automática/mecaniz. ou semiautomática. O arco arde sob uma camada de fluxo granular | Soldagem de aços carbono, baixa e alta liga. Espessura ≥ 10 mm. Posição plana ou horizontal de peças estruturais, tanques, |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | vasos de pressão, etc. |
| Soldagem com Eletrodos Revestidos | Arco elétrico | Contínua ou alternada. Eletrodo + ou - | Escória e gases gerados | Manual. Vareta metálica recoberta por camada de fluxo | Soldagem de quase todos os metais, exceto cobre puro, metais preciosos, reativos e de baixo ponto de fusão. Usado na soldagem em geral. |
| Soldagem com Arame Tubular | Arco elétrico | Contínua. Eletrodo + | Escória e gases gerados ou fornecidos por fonte externa. Em geral o CO2 | O fluxo está contido dentro de um arame tubular de pequeno diâmetro. Automático ou semiautomático | Soldagem de aços carbono com espessura ≥ 1 mm. Soldagem de chapas |
| Soldagem MIG/MAG | Arco elétrico | Contínua. Eletrodo + | Argônio ou Hélio, Argônio + O2, Argônio + CO2, CO2 | Automática/mecaniz. ou semiautomática. O arame é sólido | Soldagem de aços carbono, baixa e alta liga, não ferrosos, com espessura ≥ 1 mm. Soldage |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | m de tubos, chapas, etc. Qualquer posição |
| Soldagem a Plasma | Arco elétrico | Contínua. Eletrodo - | Argônio, Hélio ou Argônio + Hidrogênio | Manual ou automática. O arame é adicionado separadamente. Eletrodo não consumível de tungstênio. O arco é constrito por um bocal | Todos os metais importantes em engenharia, exceto Zn, Be e suas ligas, com espessura de até 1,5 mm. Passes de raiz |
| Soldagem TIG | Arco elétrico | Contínua ou alternada. Eletrodo - | Argônio, Hélio ou misturas destes | Manual ou automática. Eletrodo não consumível de tungstênio. O arame é adicionado separadamente. | Soldagem de todos os metais, exceto Zn, Be e suas ligas, espessura entre 1 e 6 mm. Soldagem de não ferrosos e aços inox. Passe de raiz de soldas em tubulações |
| Soldage | Feixe | Contínua. | Vácuo | Soldagem | Soldage |

| | | | | | |
|---|---|---|---|---|---|
| m por Feixe Eletrôni co | eletrônic o | Alta Tensão. Peça + | (»10-4mm Hg) | automática. Não há transferência de metal. Feixe de elétrons focalizado em um pequeno ponto. | m de todos os metais, exceto nos casos de evolução de gases ou vaporizaç ão excessiva , a partir de 25 mm de espessur a. Indústria nuclear e aeroespa cial. |
| Soldage m a Laser | Feixe de luz | | Argônio ou Hélio | Como acima | Como acima. Corte de materiais não Metálicos |
| Soldage m a Gás | Chama oxiacetil ênica | | Gás (CO, H2, CO2, H2O) | Manual. Arame adicionado Separadamente | Soldage m manual de aço carbono, Cu, Al, Zn, Pb e bronze. Soldage m de chapas finas e tubos de pequeno diâmetro |

**Fonte**: Modenesi e Marques (2006).

Para este trabalho, a solda do tipo *Mig Mag* foi utilizada. A seguir, alguns conceitos importantes sobre este tipo de solda.

**c) Soldagem GMAW (MIG/MAG)**

É um processo de soldagem a arco que produz a união dos metais pelo aquecimento destes com um arco elétrico estabelecido entre um eletrodo metálico contínuo (e consumível) e a peça. (MODENESI, MARQUES E SANTOS, 2012).

Também denominado *GMAW* (*gas metal arc welding*), ou seja Soldagem a Arco com Proteção Gasosa e Eletrodo Metálico.

*MIG* e *MAG* são as abreviaturas de *Metal Inert Gas* e *Metal Active Gas*, como este processo ficou conhecido antes de receber uma sigla internacional.

> *MIG* se refere ao processo cuja proteção é realizada com gases inertes e é utilizado na soldagem de materias não ferrosos como alumínio e suas ligas, cobre, níquel, entre outras; enquanto *MAG*, que é o processo usual na soldagem de materiais ferrosos como aços ao carbono e aços baixa liga entre outros, se refere ao processo com proteção por gases ativos. (REVISTA DA SOLDAGEM, Ano I, N°4)

Ainda sobre este processo, tem uma característica de auto ajuste do arco e, através de diferentes tipos de transferência metálica, permite a soldagem com uma grande variedade de parâmetros e de técnicas de soldagem. (REVISTA DA SOLDAGEM, Ano I, N°4).

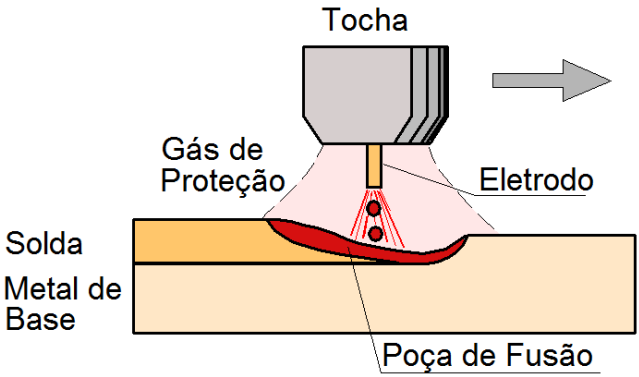
> O eletrodo - que é o próprio arame sólido ou tubular com alma metálica - é alimentado com uma velocidade constante e ajustável podendo soldar com correntes que variam entre 50A e 600A e tensões entre 13V e 38V, ou seja é possível soldar com praticamente qualquer parâmetro, mas segundo transferências metálicas diferentes, sendo que algumas podem não ser exatamente as recomendáveis para a aplicação específica (REVISTA DA SOLDAGEM, Ano I, N°4)

A soldagem *MIG/MAG* é bastante versátil quanto às ligas soldáveis e espessuras de material, podendo ser usada em todas as posições. Possui uma gama de aplicações na soldagem de não ferrosos e aços inoxidáveis. (PEIXOTO, 2012).

Como num processo semiautomático, sua produtividade é bastante elevada. Quando o material é de boa soldabilidade, o processo *MIG/MAG* é sempre uma alternativa viável, vantajosa com relação à soldagem com eletrodos revestidos. (PEIXOTO, 2012).

Um desenho esquemático deste tipo de soldagem pode ser visualizado na figura 8.

**Figura 8 -** Soldagem GMAW (Esquemática)




**Fonte:** Modenesi e Marques (2012).

Este processo é normalmente operado de forma semiautomática, podendo ser, também, mecanizado ou automatizado MODENESI e MARQUES (2012).

Ainda segundo Modenesi e Marques (2012), é o processo de soldagem a arco mais usado com robôs industriais.

> Trabalha com um (ou mais) arame(s) contínuo(s), o que permite um alto fator de ocupação, com elevadas densidades de corrente no eletrodo (elevada taxa de deposição) e, assim, tende a apresentar uma elevada produtividade. (MODENESI e MARQUES, 2012)

A tabela a seguir mostra as relativas vantagens, limitações e principais aplicações do processo.

**Tabela 7 –** Vantagens, Limitações e Principais Aplicações do Processo de Soldagem

| Vantagens e Limitações | Aplicações |
|---|---|
| Processo com eletrodo contínuo. | Soldagem de ligas ferrosas e não ferrosas. |
| Permite soldagem em qualquer posição. | Soldagem de carrocerias e estruturas de veículos. |
| Elevada taxa de deposição de metal. | Soldagem de tubulações, etc. |
| Elevada penetração. | |
| Pode, em princípio, soldar diferentes ligas metálicas. | |
| Exige pouca limpeza após soldagem. | |
| Processo exige, em geral, menos habilidade do soldador que a soldagem SMAW. | |
| Processo de ajuste mais difícil e sensível que o processo SMAW | |
| Equipamento relativamente caro e complexo. | |
| Pode apresentar dificuldade para soldar juntas de acesso restrito. | |
| Proteção do arco é sensível a correntes de ar. | |
| Pode gerar elevada quantidade de respingos. | |

**Fonte**: Modenesi, Marques e Santos (2012).

> "A proteção do arco e poça de fusão é obtida por um gás ou mistura de gases. Se este gás é inerte, o processo é também chamado de MIG (Metal Inert Gas). Se o gás for ativo, o processo é chamado de MAG (Metal Active Gas)" (MODENESI, MARQUES E SANTOS, 2012, pg 17).

O processo é normalmente operado de forma semiautomática e apresenta alta produtividade. Para Modenesi, Marques e Santos (2012),

> a transferência de metal através do arco se dá, basicamente, por três mecanismos: aerosol (*spray*), globular e curto-circuito, dependendo de parâmetros operacionais, tais como o nível de corrente, sua polaridade, diâmetro e composição do eletrodo, composição do gás de proteção e comprimento do eletrodo.

Ainda pode se mencionar os processos híbridos de soldagem que segundo Modenesi e Marques (2006), é a combinação de mais de um processo de soldagem (em geral, um processo a arco e algum outro) permitindo a obtenção de um nosso processo que pode apresentar vantagens sobre cada um dos processos iniciais.

Os processos híbridos mais conhecidos são os que envolvem o uso conjunto da soldagem *GMAW* e a soldagem *laser* ou a plasma (MODENESI e MARQUES, 2006).

O uso conjunto dos processos afeta o funcionamento de cada um (por exemplo, a interação do *laser* com o material gera um plasma que pode estabilizar o arco e a poça de fusão gerada pelo arco facilita a penetração do *laser* no material) e o formato final do cordão de solda. (MODENESI e MARQUES, 2006).

## 5 Procedimentos Metodológicos

Este trabalho tem como intuito, testar um *tracking* magnético integrando-o à um simulador de solda industrial com auxílio da ferramenta *Unity3D* e realidade virtual utilizando o método de pesquisa em livros e artigos relacionados ao assunto.

Com relação à codificação do *framework*, optou-se pela utilização da linguagem *C#* com auxílio do *Visual Studio* como

ambiente de desenvolvimento. Foram realizados testes de observação para atestar o funcionamento do projeto e houve então, alguns critérios abordados relacionados à utilização como precisão e erro acumulado onde se analisou e verificou a variação entre acertos e erros enquanto manipula-se a ferramenta, existência de acúmulo de erros durante execução e descalibre da ferramenta no mundo virtual.

## 5.1 Instrumental utilizado

Para elaboração do projeto utilizamos os seguintes instrumentos:

### 5.1.1 *Unity3D* versão 5.3.3

A *Unity3D* é um motor gráfico considerado de fácil utilização e que destaca-se principalmente por seus projetos *3D*. Ela oferece um sistema de script compreensivo e flexível, possibilitando a prototipação e desenvolvimento rápido e robusto de todas as partes de um projeto sem, necessariamente, conhecer programação de baixo nível ou múltiplas bibliotecas de desenvolvimento e plataformas.

A *Unity3D* está disponível em duas versões: uma versão gratuita que contém diversas funcionalidades, porém algumas ferramentas são restritas e uma versão paga, que engloba todas as funcionalidades da versão já criada (SIGNORI e RIGO, 2012).

Esta ferramenta ainda conta com muitos tutoriais e exemplos de suas funcionalidades e ferramentas, além de uma documentação completa.

Também é possível encontrar diversos *fóruns* e *blogs* que auxiliam os desenvolvedores da ferramenta no compartilhamento de diversas experiências e informações.

O *download* desta ferramenta pode ser realizado gratuitamente através do seu site oficial[2].

[2] https://unity3d.com/pt/get-unity/download

### 5.1.2 Oculus Rift

**Figura 9 –** Oculus Rift, DK2 (Development kit 2)


**Fonte:** Autoria Própria

O *Oculus Rift,* como mostra a figura 9, é um óculos de realidade virtual muito popular atualmente. É um sistema visual do tipo *Head-mounted display* desenvolvido pela empresa *Oculus VR*. A ideia do capacete foi lançada no *Kickstarter*, uma plataforma de *crowdfunding*, pedindo 200 mil dólares para que o projeto saísse do papel. A aceitação foi muito positiva e cerca de um mês depois, o projeto arrecadou mais de dois milhões de dólares.

O primeiro kit financiado pelo site colaborativo, *Kickstarter,* tinha tela de LCD de 7 polegadas com resolução de 640×800 por olho. A nova geração do *Development Kit* (DK2), lançada em 2014 (que usamos nesse projeto), trouxe algumas melhorias, como o uso de um *display* de *OLED* de baixa persistência com o dobro da qualidade (são 960×1080 pixels por olho) e um acessório extra para aprimorar o rastreamento de posição.

O potencial do *Rift* vai além do entretenimento e já está sendo explorado por outros setores como educação, saúde, mercado imobiliário entre outros. O *Oculus Rift* possui integração com as engines *Unity3D* e *Unreal Engine* sendo hoje um dos principais produtos do seguimento. É possível realizar o

*download* do *sdk* e *runtime* do *Oculus Rift* através do site oficial[3] da empresa.

### 5.1.3 *Tracking* Magnético Modelo *miniBird*-500 da Ascension Technology

O modelo do *tracking* magnético utilizado para este trabalho foi o *MiniBird*-500 da *Ascension Technology,* mostrado na figura 10.

**Figura 10 –** Aparelho *MiniBird*-500 da Ascension Technology


**Fonte**: Autoria Própria

Algumas características importantes para a utilização deste modelo são as medidas de orientação e posição com boa precisão, gama irrestrita de movimentos e oferecimento de 6 graus de liberdade, ou *6DOF,* que pode ser definido como a liberdade de movimento de um corpo rígido no espaço tridimensional onde este corpo é livre para mudar de posição por meio de rotação em torno de três eixos perpendiculares.

Uma vez que aplicações de realidade virtual e aumentada exigem geração em tempo real de cenas tridimensionais, sistemas de rastreamento devem rastrear todos os objetos desejados sem causar atrasos na geração destas cenas.

Outros benefícios quanto a utilização deste modelo são mini sensores que permitem medições internas, medição simultânea de múltiplos sensores e *software* de interface. É

[3] https://developer.oculus.com/downloads/

possível realizar o *download* da biblioteca do *MiniBird* e da sua documentação no *link*[4]

## 6 Histórico do Desenvolvimento

### 6.1 Início

Ao iniciar este trabalho, no começo de março de 2016, o desenvolvimento do Simulador, utilizado aqui, já estava em desenvolvimento desde janeiro de 2016. Era possível controlar, com o *mouse* e teclado, a pistola de solda e fazer pequenos caminhos, ainda experimentais, de soldagem. Já sendo possível utilizar um óculos de realidade virtual, a imersão era significativa. Contudo, utilizar teclado e *mouse* estava longe do ideal. Se fazia necessário uma maneira de controlar a pistola de solda virtual o mais próximo da realidade possível. Tornara-se necessário um aparelho capaz de rastrear os movimentos da pistola de solda com precisão e acurácia.

Na figura 11, é possivel visualizar o estado inicial do desenvolvimento.

**Figura 11 –** Estado inicial do simulador de solda


**Fonte:** Autoria Própria

Com relação aos controles de movimentos e de realidade virtual, existe uma grande necessidade de sensores para monitorizar o movimento de um objeto. (SOSNICKI, 2010)

[4] https://www.dropbox.com/s/wzr7a93rgwal3rp/bird.zip?dl=0

Ainda para SOSNICKI (2010), estes sensores são caracterizados por sua performance como faixas, precisão, desvios, e suscetibilidade ao ambiente.

O rastreamento do movimento de um usuário é um dos grandes suportes para garantir a real sensação de imersão em um ambiente virtual e é através dele que o sistema de realidade virtual interpreta os comandos do usuário e responde a eles de forma adequada.

Os chamados *tracking devices*, são os dispositivos que tem como objetivo, determinar a posição ou a orientação de uma parte do corpo do usuário e quanto maior a precisão, velocidade e área de ação destes equipamentos, maior será a quantidade de dados disponíveis à interpretação pelo sistema.

Segundo Junior, Souza e Machado (2009),

> Um dos principais requisitos que deve estar presente em um sistema de rastreamento é a acurácia que, ainda segundo os autores, os sistemas ideais deveriam oferecer uma precisão de aproximadamente 1mm para a posição dos objetos e erros de orientação menores que 0.1 graus.

Para Peters e Cleary (2008), a escolha do sistema de *tracking* é dependente da aplicação e requer uma compreensão do volume de trabalho desejado e requisitos de precisão.

Para este trabalho, a precisão com relação ao rastreamento do movimento foi uma das características fundamentais e para isso foi necessário a utilização de um modelo de *tracking que* oferecesse entre outras vantagens, medidas de orientação e posição com boa precisão, gama irrestrita de movimentos e oferecimento de 6 graus de liberdade ou *6DOF*.

**Tabela 8** – Cronograma das Atividades

| **Cronograma das atividades (2016)** | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Acões | Mar | Abr | Mai | Jun | Jul | Ago | Set | Out | Nov | Dez |
| a | Definição do Tema | ■ | | | | | | | | | |
| b | Acordo com Orientador | | ■ | | | | | | | | |
| c | Definição dos Objetivos | | ■ | ■ | | | | | | | |
| d | Pesquisa bibliográfica | ■ | ■ | | | | ■ | | | | |
| e | Desenvolvimento | | ■ | ■ | ■ | ■ | ■ | ■ | ■ | | |
| f | Testes | | | | | | ■ | ■ | | | |
| g | Entrega Parcial do Relatório | | | | | ■ | | | | | |
| h | Entrega Final do Relatório | | | | | | | | | ■ | |

**Fonte:** Elaborada pelo autor.

### a. Definição do tema

No ano de 2016 a realidade virtual ficou em evidência. E o mercado indica que haverá ainda mais pesquisa e desenvolvimento acerca do tema. Realidade virtual é sobre imersão, para poder acreditar no que pode-se ver e nas interações feitas. No final de março de 2016 o simulador de solda apresentava uma deficiência na imersão da realidade virtual. O tema deste trabalho Integração de um *Tracking* Magnético à um

simulador de soldagem manual com Realidade Virtual utilizando *Unity3D*, tem o intuito de suprir a deficiência deste, e talvez de outros, simuladores que utilize alguma ferramenta de precisão em conjunto com a realidade virtual.

**b. Acordo com Orientador**

Início de Abril o orientador Fabiano Garcia e o professor responsável Antônio Augusto Fröhlich, aceitaram o desafio de guiar e orientar este trabalho.

**c. Definição dos Objetivos**

Em conjunto com o orientador foi definido uma série de objetivos à alcançar. Estes objetivos eram muito abrangentes e, após as pesquisas bibliográficas iniciais, e com o desenvolvimento em andamento, os objetivos foram revistos. Em meados de maio os objetivos foram redefinidos para serem mais específicos; são os apresentados aqui neste trabalho.

**d. Pesquisa Bibliográfica**

Para a própria concepção do tema e verificação da viabilidade foi feito levantamento de vários artigos, livros, etc. A pesquisa inicial levou em conta a definição e conceitos de realidade virtual, tecnicas de soldagem, as ferramentas utilizadas, tipos de *tracking*, etc.

A segunda parte da pesquisa, foi focado no próprio *tracking* magnético e na ferramenta *Unity3D*; com o intuito de conceber melhor a integração entre o aparelho de *tracking* e o simulador de solda.

**e. Desenvolvimento**

O desenvolvimento foi iniciado com a aprovação do projeto e o que abrange este trabalho, foi encerrado no início de outubro.

### f. Testes

Os testes foram todos internos e baseados em observação e ocorreram periodicamente durante todo projeto. Os testes de validação, contudo, com a liderança do projeto, foi feito no final, quando foi realizado a integração com o simulador de solda.

### g. Entrega Parcial do Relatório

A entrega parcial do relatório deste trabalho foi feita no final de julho. Neste primeiro relatório foi entregue a fundamentação e os conceitos utilizados neste trabalho.

### h. Entrega Final do relatório

Para a ultima entrega, foi modificado um pouco a entrega parcial anterior para deixá-la mais robusta e foi incluído o desenvolvimento e conclusão do trabalho.

## 6.2 Tracker

Por questão de disponibilidade e também por cumprir os requisitos, foi determinado pela liderança do projeto que o *tracker* utilizado seria o *MiniBird 500* da *Ascension Technology.*

O *MiniBird* é um dispositivo de medição de seis graus de liberdade (6DOF), utilizado para medir a posição e orientação com um pequeno sensor, respeitando o transmissor.

> O sensor é capaz de produzir de 30 a 144 medições por segundo de sua posição e orientação quando localizado dentro de ± 30 polegadas de seu transmissor e determina a posição e orientação através da transmissão de um campo magnético pulsante DC que é medido por um sensor. (ASCENSION TECHNOLOGY CORPORATION, 2001)

A partir das características do campo magnético medido, o sensor calcula a sua posição e orientação e torna esta informação disponível. (ASCENSION TECHNOLOGY CORPORATION, 2001)

O *MiniBird* ainda pode ser configurado para atender às necessidades de muitas aplicações diferentes a partir de uma unidade autônoma que consiste de um único transmissor e sensor para mais complexa configurações incluindo várias combinações de emissores e sensores. (ASCENSION TECHNOLOGY CORPORATION, 2001)

Algumas das características mais importantes para a utilização deste modelo neste trabalho foram as medidas de orientação e posição com boa precisão, gama irrestrita de movimentos e oferecimento de 6 graus de liberdade (6DOF) com alguns outros benefícios como mini sensores que permitem medições internas, medição simultânea de múltiplos sensores e *software* de interface.

Este modelo apresenta uma *interface RS-232,* mostrado na figura 12, que é um padrão de protocolo para troca serial de dados binários, comumente usado nas portas seriais dos computadores e que, por ser antiga, necessitou da utilização de um adaptador *USB.*

**Figura 12 –** Traseira do *MiniBird*-500 da Ascension Technology Corporation


**Fonte**: Autoria Própria

## 6.3 Controle e Adaptação

Neste estágio inicial o controle do simulador era bastante rudimentar. Movendo o *mouse* sobre a mesa de solda, com o botão esquerdo do *mouse* pressionado, controlava-se a posição

da tocha de soldagem na mesa. A roda do meio do *mouse* controlava a altura da pistola de soldagem e, por fim, segurando *shift* no teclado controlava-se a inclinação da tocha de soldagem. A soldagem começava com a aproximação da ponteira da tocha de soldagem na mesa. Com isso o simulador conseguia capturar diversas características como inclinação, velocidade e altura necessárias para analisar se a soldagem estava adequada ou não. Contudo, por ser controlado por *mouse* e teclado não refletia as habilidades do usuário. Para um melhor acompanhamento e treinamento, tornou-se imprescindível o *tracker.*

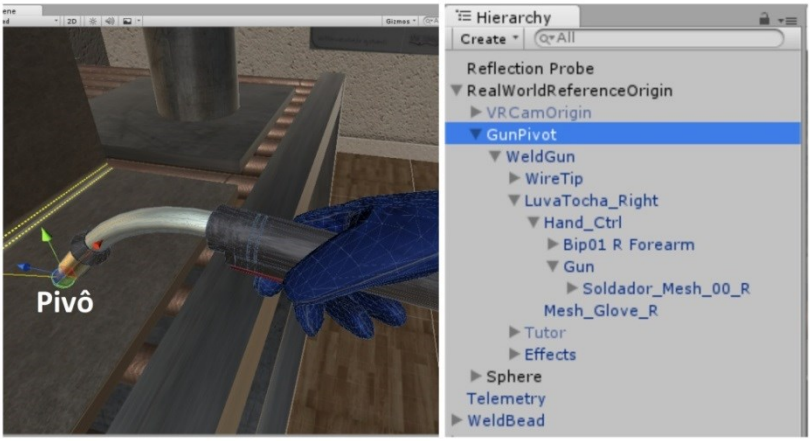
Com o *tracker* integrado, é esperado que o simulador capture, bem próximo à realidade, essas caraterísticas do usuário e indique se este fez uma boa soldagem ou não, bem como seu progresso de treino. O *tracker*, no simulador, irá controlar a tocha de soldagem, substituindo o controle rudimentar do teclado e *mouse.*

Para o bom andamento de ambos os projeto, tanto o simulador, quanto a integração do *tracking* magnético com a *Unity3D,* decidiu-se por trabalhar separadamente até o momento em que ambos estivessem maduros o suficiente para fazer a integração - Isso ocorreu, com sucesso, em agosto de 2016.

Ficou combinado que o *tracking* iria, na integração, controlar a translação e rotação de um GameObject "pai" da tocha de soldagem. O "pai" seria o *pivô* da tocha de soldagem. Ou seja, todo o ferro acompanharia os movimentos ditados pelo "pai", como se este fosse a origem dos "filhos". Se o pai for movido, os filhos também serão, se for rotacionado os filhos também serão rotacionados de acordo.

Abaixo, segue a figura que representa a utilização do pivô deste trabalho.

**Figura 13 –** Ilustração do Pivô




**Fonte**: Autoria Própria

## 6.4 Documentação e Biblioteca do *MiniBird*

Depois das definições iniciais foi dado, em meados de março, início à um novo projeto, com a intenção de solucionar a integração do *tracking* magnético com a *Unity3D* e, posteriormente, integrar ao simulador de solda. O primeiro passo foi obter todas as documentações e códigos disponíveis para o *tracker MiniBird 500* da *Ascension Technology*. Reuniu-se então, alguns tutoriais oficiais para fazer *upgrade* de *driver*, além de documentos sobre o uso de alguns *softwares* de teste e demonstrativos. A documentação mais importante reunida, no entanto, foi o guia de instalação e uso do aparelho e a documentação da *API* do *driver*.

A primeira informação à chamar a atenção em toda a documentação é a idade avançada do aparelho, que data de 2001. Com isto, foi necessário checar a compatibilidade com os novos sistemas e máquinas. Rodou-se um programa teste na máquina de Processador *Intel(R) Core(™) i7-4770*, *CPU* de 3.40 GHz, memória de 8 GB com sistema de 64 *bits* e placa de vídeo *GeForce GTX* 750 de 1024 MB com o sistema operacional *Windows 8.1 Pro* de *64 bits* , e o *software* rodou sem problema algum.

Esse aparelho vem com um arquivo *DLL* para controlar e obter informações do *tracker*. Foi utilizado esse arquivo e sua documentação para desenvolver o *framework* que possibilitaria a integração do *tracker* com o ambiente *Unity3D*.

*DLL* é a sigla para *Dynamic Link Library* e se trata de uma biblioteca dinâmica que contém dados que podem ser acessados por mais de um programa instalado no computador, inclusive simultaneamente.

Elas são compostas por sub-rotinas armazenadas em disco, podendo ser carregadas na memória e executadas quando um aplicativo realiza o seu acesso. Uma *DLL* pode conter código, dados ou recursos (ícones, fontes, cursores, entre outros).

Desta forma, existe *DLLs* para as mais variadas funções, como efetuar o gerenciamento de memória para um aplicativo ou abrir uma janela de mensagem no sistema. Um aplicativo pode conter vários desses arquivos, fazendo com que as suas funções fiquem modularizadas no Sistema Operacional. É possível então, por exemplo, realizar uma atualização de um programa no *Windows* em vez de ser necessário remover e instalar novamente a aplicação sempre que houver uma modificação por parte do desenvolvedor.

Além disso, quando vários aplicativos utilizam uma mesma biblioteca de funções, o emprego de *DLLs* reduz a duplicação do código carregado pelo computador. Isso influencia no desempenho tanto do *software* em execução em primeiro plano quanto de outras aplicações abertas no computador, fazendo com que tudo fique mais rápido. Uma das vantagens de usar *DLL* está na economia de recursos. Quando vários programas utilizam uma mesma biblioteca de funções, o emprego de *DLLs* faz com que a duplicação do código carregado no disco e na memória física seja reduzida, melhorando tanto o desempenho dos aplicativos ativos (em primeiro plano) quanto das aplicações rodando em plano de fundo. Outra vantagem é que o uso de *DLLs* deixa os programas modulares, facilitando o desenvolvimento de aplicações que necessitem de versões em vários idiomas, por exemplo. Apenas a modificação da *DLL* já é capaz de fazer a correção, e todos os programas associados a ela são beneficiados com a atualização.

## 6.5 Inicialização do *MiniBird*

Da documentação deste aparelho específico *MiniBird – 500 da Ascension Technology Corporation* foi identificado diversos modos de iniciar, configurar, e obter os dados do aparelho. Tendo a aplicação do simulador de solda em mente, foi selecionado um conjunto básico de funções que atendem às necessidades.

O primeiro passo a fazer é "acordar" o aparelho. Essa é uma função que irá fazer o *setup* inicial do aparelho. Como o aparelho funcionará com uma *interface RS232*, utiliza-se a função:

```
BOOL birdRS232WakeUp(int nGroupID,
                     BOOL bStandAlone,
                     int nNumDevices,
                     WORD *pwComport,
                     DWORD dwBaudRate,
                     DWORD dwReadTimeout,
                     DWORD dwWriteTimeout,
                     int nGroupMode);
```

O parâmetro *nGroupID* é usado somente quando mais de um aparelho é utilizado ao mesmo tempo, formando grupos. Como utilizou-se somente um aparelho por simulador, essa opção pode ser '0'. Pelo mesmo motivo a opção *bStandAlone* será *'true'* e *nNumDevices* será '0'. O parâmetro *pwComport* é um *array* com os endereços das portas onde cada aparelho está conectado. Como foi utilizado um único aparelho, será um *array* com um único elemento indicando o número de sua porta. *dwBaudRate* é a taxa de transmissão a ser usada e *dwReadTimeout* e *dwWriteTimeout* são o tempo máximo de leitura e escrita respectivamente. Todos os tempos são definidos em milisegundos. Esses tempos devem ser balanceados de forma que a taxa de transmissão seja boa o suficiente para enviar e receber dados novos sem que a *CPU* fique sobrecarregada; e que tenha tempo o suficiente para escrever e ler do aparelho mas que não ocupe a *CPU* tentando escrever ou

ler do aparelho por muito tempo. A configuração de tempo definida para este trabalho foi: '115200', '2000' e '2000' respectivamente.

Todas as funções do *driver*, com exceção dos métodos de acesso aos erros, retornam *TRUE*, se foi executado corretamente, ou *FALSE* caso algum erro tenha ocorrido. Em caso de falha, pode-se obter informações do erro atráves das funções *birdGetErrorMessage* e *birdGetErrorCode*, que retornam uma *string* e um inteiro respectivamente. Estas são funções importantes para debugar e entender os problemas no desenvolvimento.

Depois de "acordar" o *tracker*, é preciso obter as informações básicas do sistema do aparelho. Estas informações estão estruturadas numa *struct* do tipo *BIRDSYSTEMCONFIG*. Essa informação deve ser obtida através da função *birdGetSystemConfig*. Caso alguma configuração especial seja necessária, utiliza-se *birdSetSystemConfig* com *BIRDSYSTEMCONFIG* modificado como parâmetro. Neste trabalho não foi preciso nenhuma configuração especial.

```
BOOL birdGetSystemConfig(int nGroupID,
                         BIRDSYSTEMCONFIG *psyscfg,
                         BOOL bGetDriverCopy);
```

Para finalizar a inicialização do *bird* foi preciso pegar as informações do aparelho. As informações são compiladas na *struct BIRDDEVICECONFIG*. E podem ser obtidas através da função *birdGetFastDeviceConfig*:

```
BOOL birdGetFastDeviceConfig(int nGroupID,
                             int nDeviceNum,
                             BIRDDEVICECONFIG *pdevcfg);
```

Uma estrutura é passada como referência e é preenchida com as informações pertinentes pela função. O valor do *id* do grupo (*nGroupID*) é aquele passado na função *WakeUp*

executada anteriormente. O valor de índice do aparelho (*nDeviceNum*), nesse caso será '0', porque há somente um aparelho sendo observado. Utilizou-se as configurações padrão, exceto pelo que diz respeito à captura da posição e translação. Fez-se necessário adquirir os dados de posição e rotação utilizando quaternions. Por isso, foi modificado o campo *'byDataFormat'* da struct preenchida pela função anterior e setar novamente utilizando a função *birdSetFastDeviceConfig*:

```
BOOL birdSetFastDeviceConfig(int nGroupID,
                             int nDeviceNum,
                             BIRDDEVICECONFIG *pdevcfg);
```

Depois de todos esses passos tem-se o dispositivo pronto para comunicação. O aparelho é acordado, adquiriu-se as informações do sistema de *trackers*, as informações do *tracker* específico e foram feitas as customizações necessárias. Na figura 14 há um exemplo de código. E as tabelas 9 e 10 mostram os detalhes das funções e estruturas utilizadas.

**Tabela 9 –** Descrição e Assinatura das Funções Utilizadas para Inicialização do *MiniBird*-500

| Função | Parâmetros | Descrição |
|---|---|---|
| BOOL birdRS232WakeUp | int nGroupID, BOOL bStandAlone, int nNumDevices, WORD *pwComport, DWORD dwBaudRate, DWORD dwReadTimeout, DWORD dwWriteTimeout, int nGroupMode | Acorda um grupo de aparelho *bird* no modo RS232. Nota: A variável 'pwComport' pode ser deletada depois que a função retorna |

| | | |
|---|---|---|
| BOOL birdGetSystemConfig | int nGroupID, BIRDSYSTEMCONFIG *psyscfg, BOOL bGetDriverCopy | Pega a configuração do sistema de um grupo de aparelho *bird* identificados por nGroupID na função WakeUp. A informação é retornada na variável psyscfg. Quando bGetDriverCopy é falso, os dados são lidos do hardware. Quando *true*, a informação obtida é mais rápida; vem da memória, carregada quando WakeUp foi chamado. |
| BOOL birdGetFastDeviceConfig<br>E<br>BOOL birdSetFastDeviceConfig | int nGroupID, int nDeviceNum, BIRDDEVICECONFIG *pdevcfg | Pega ou seta individualmente as informações do aparelho no grupo identificado por nGroupID na função WakeUp; e pelo identificador individual nDevice. A informação é retornada ou setada pela variável pdevcfg. Nestes métodos os parâmetros "byReportRate byHemisphere |

| | | |
|---|---|---|
| | | wAlphaMin[]<br>wAlphaMax[]<br>wVM[]<br>anglesReferenceFrame<br>anglesAngleAlign"<br>Não são suportados. |

**Fonte:** Elaborada pelo Autor

**Tabela 10** – Descrição das Estruturas Utilizadas

| Estrutura | Campos | Descrição |
|---|---|---|
| BIRDSYSTEMCONFIG | BYTE bySystemStatus | Status atual do sistema de acordo com a enumeração BIRD_SYSTEM_STATUS |
| | BYTE byError | Erro detectado |
| | BYTE byNumDevices | Numero de aparelhos no sistema |
| | BYTE byNumServers | Numero de servidores no sistema (quando não está no modo "StandAlone") |
| | BYTE byXmtrNum | Numero de transmissão de acordo com a enumeração TRANSMITTER_NUMBER |
| | WORD wXtalSpeed | Velocidade do crystal em MHz |

| | | |
|---|---|---|
| | Double dMeasurementRate | Taxa de Medição em frames por segundo |
| | BYTE byChassisNum | Numero do chassi |
| | BYTE byNumChassisDevices | Numero de aparelhos dentro deste chassi |
| | BYTE byFirstDeviceNum | Numero do primeiro aparelho neste chassi |
| | WORD wSoftwareRev | Versão do *software* |
| | BYTE byFlockStatus[] | Status de todos os aparelhos no sistema, indexados pelo numero identificador do aparelho de acordo com a enumeração BIRD_FLOCK_STATUS |
| BIRDDEVICECONFIG | BYTE byStatus | Status do aparelho de acordo com a enumeração BIRD_DEVICE_STATUS. |
| | BYTE byID | Identificador do aparelho. Identifica o tipo do aparelho de acordo com a enumeração BIRD_DEVICE_ID |
| | WORD | Versão do software |

| | | |
|---|---|---|
| | wSoftwareRev | |
| | BYTE byError | Código de erro reportado |
| | BYTE bySetupsetup | Enumeração de acordo com BIRD_DEVICE_SETUP com informações sobre o setup do aparelho. |
| | BYTE byDataFormat | Formato dos dados de acordo com a enumeração BIRD_DATA_FORMATS. Essa informação é utilizada para informar o formato dos dados como por exemplo se os ângulos serão em *quaternion* ou *euler*, ou se deve pegar uma matriz ao invés de dados individuais etc. |
| | BYTE byReportRate | Taxa de obtenção de informação em unidades de frame |
| | WORD wScaling | Escala total de medição em polegadas |
| | BYTE byHemisphere | Semi-esfera de operação de acordo com a enumeração BIRD_HEMISPHERE_CODES |

| | | |
|---|---|---|
| | BYTE byDeviceNum | Número do aparelho |
| | BYTE byXmtrType | Tipo de transmissão de acordo com a enumeração BIRD_TRANSMITTER_TYPE |
| | WORD wAlphaMin[7] | Filtros de acordo com a enumeração ALPHA_MIN_FILTER_TABLE |
| | WORD wAlphaMax[7] | Filtros de acordo com a enumeração ALPHA_MAX_FILTER_TABLE |
| | WORD wVM[7] | Filtros de acordo com a enumeração Vm_FILTER_TABLE |
| | BIRDANGLES anglesReferenceFrame | Frame de referência para as leituras do aparelho |
| | BIRDANGLES anglesAngleAlign | Alinhamento das leituras do aparelho |

**Fonte**: Elabora pelo autor

**Figura 14 –** Exemplo de Código para Inicialização do *MiniBird*

```
public void StartBird()
{
    if (started) return;
    int groupId = 0;
    bool standAlone = true;
    int numDevices = 0;
    ushort[] COM_port = new ushort[] { 3 };
    uint boudRate = 115200;
    uint readTimeout = 2000;
    uint writeTimeout = 2000;
    int groupMode = (int)GroupModeSettings.NUM_GROUP_MODE_SETTINGS;
    if (!StartBird(groupId, standAlone, numDevices, COM_port,
        boudRate, readTimeout, writeTimeout, groupMode))
    {
        throw new Exception("Could not wakeup bird");
    }
    started = true;
}

private bool StartBird(int groupId, bool standAlone, int numDevices, ushort[] COM_port,
    uint boudRate, uint readTimeout, uint writeTimeout, int groupMode)
{
    if (birdRS232WakeUp(groupId, standAlone, numDevices, COM_port,
        boudRate, readTimeout, writeTimeout, groupMode))
    {
        BIRDSYSTEMCONFIG sysconfig = new BIRDSYSTEMCONFIG();
        BIRDDEVICECONFIG devconfig = new BIRDDEVICECONFIG();
        if (birdGetSystemConfig(groupId, ref sysconfig, true) &&
            birdGetFastDeviceConfig(groupId, numDevices, ref devconfig))
        {
            devconfig.byDataFormat = (byte)BirdDataFormat.BDF_POSITIONQUATERNION;
            if (birdSetFastDeviceConfig(groupId, numDevices, ref devconfig))
            {
                return true;
            }
        }
    }

    return false;
}
```

**Fonte**: Elaborada pelo autor.

## 6.6 Leitura do *MiniBird*

Com o aparelho preparado, utilizou-se então a função *birdStartFrameStream* para iniciar o processo de obter informações de posição e rotação regularmente do aparelho. Quando não houver mais necessidade do fluxo constante de informação do dispositivo utiliza-se sua contraparte *birdStopFrameStream.*

A comunicação é estabelecida e o aparelho pronto para enviar informações. A informação será compilada na *struct BIRDREADING*. Onde terá disponível para leitura a posição e o

ângulo do *tracker* (o formato e as informações disponíveis foram ajustados na etapa de configuração do aparelho).

Para pegar as informações mais recentes do *tracker*, foi pego o último *frame* disponível.

Um *frame* é um momento, um pedaço, no tempo de vida do rastreamento. E esse momento descreve as condições no sistema.

Foi utilizado a função *birdGetMostRecentFrame*, onde obtém-se o *frame* mais recente em uma *struct BIRDFRAME*.

```
BOOL birdGetMostRecentFrame(int nGroupID,
                                    BIRDFRAME *pframe)
```

Dentro do frame há um *array* de *struct BIRDREADING*, com as informações pertinentes. Como foi utilizado somente um aparelho, o objeto de interesse estará na posição '0' deste *array*. O *driver* oferece ainda um método para verificar se há um novo *frame* disponível para pegar, *birdFrameReady*:

```
BOOL birdFrameReady(int nGroupID);
```

Por último, quando não precisar mais das informações do *tracker*, precisará então finalizar sua operação utilizando o método *birdShutDown*:

```
void birdShutDown(int nGroupID);
```

Na figura 15 há um exemplo de como obter e utilizar o *frame.* E as tabelas 11 e 12 mostram os detalhes das funções e da estruturas utilizadas.

**Tabela 11** – Descrição e Assinatura das Fuções de Leitura e Encerramento do *MiniBird*-500

| Função | Parâmetros | Descrição |
|---|---|---|
| BOOL birdStartFrameStream E BOOL birdStopFrameStream | Int nGroupID | Começa ou para o *streaming* de frames de dados. nGroupID é o grupo indicado na função WakeUp |
| BOOL birdFrameReady | int nGroupID | Retorna TRUE se o frame de dados do grupo identificado por nGroupID estiver pronto, retorna FALSE caso contrário. |
| BOOL birdGetMostRecentFrame | int nGroupID, BIRDFRAME *pframe | Pega o *frame* mais recente disponível. O valor é passado através da variável pframe. |
| void birdShutDown | int nGroupID | Desconecta virtualmente o aparelho, limpando a memória e desfazendo as configurações iniciais. |

**Fonte**: Elaborada pelo autor

**Tabela 12 –** Descrição das Estruturas Utilizadas

| Estrutura | Campos | Descrição |
|---|---|---|
| BIRDFRAME | DWORD dwTime | Momento no qual a leitura foi feita, em milisegundos |
| | BIRDREADING reading[] | Leitura de cada bird. |
| BIRDREADING | BIRDPOSITION position | Posição do *tracker* |
| | BIRDANGLES angles | Orientação do tracker em angulos (Euler) |
| | BIRDMATRIX matriz | Orientação do tracker em forma de matriz |
| | BIRDQUATERNION quaternion | Orientação do tracker em quaternion |
| | WORD wButtons | Estado dos botões |

**Fonte**: Elaborada pelo autor

**Figura 15 –** Exemplo de Código de Obtenção das Informações do *MiniBird*

```
private string SerializeBirdReading(BIRDREADING birdReading)
{
    string value = "";
    float lengthFactor = 91.44f;
    value += (birdReading.position.nX * lengthFactor / 32767.0f) + ";";
    value += (birdReading.position.nY * lengthFactor / 32767.0f) + ";";
    value += (birdReading.position.nZ * lengthFactor / 32767.0f) + ";";
    value += (birdReading.quaternion.nQ0 / 32767.0f) + ";";
    value += (birdReading.quaternion.nQ1 / 32767.0f) + ";";
    value += (birdReading.quaternion.nQ2 / 32767.0f) + ";";
    value += (birdReading.quaternion.nQ3 / 32767.0f);
    return value;
}

public string GetSerializedBirdFrame()
{
    if (!streaming)
    {
        throw new Exception("Streaming was not started");
    }
    int groupId = 0;
    if (birdFrameReady(groupId))
    {
        BIRDFRAME birdFrame = new BIRDFRAME();
        birdGetMostRecentFrame(groupId, ref birdFrame);
        BIRDREADING birdReading = birdFrame.readings[0];
        lastData = SerializeBirdReading(birdReading);
    }

    return lastData;
}
```

**Fonte**: Elaborada pelo próprio autor

## 6.7 Conflito entre módulo 32 *bits* e 64 *bits*.

A partir do entendimento das funções disponíveis, das necessidades do projeto e dos testes práticos, foi dado início à integração do *bird* com o *Unity3D*.

A primeira dificuldade encontrada foi as versões dos *softwares*. O projeto do simulador de solda foi desenvolvido na versão da *Unity3D* de 64 *bits*. Contudo, o *driver* do *bird* só tem disponível a versão de 32 *bits*. E, no sistema operacional *Windows*, um processo de 64 *bits* não pode, diretamente, carregar uma biblioteca de 32 *bits*.

Uma das principais vantagens da tecnologia de 64 *bits* é a sua capacidade de trabalhar com até 8Tb de memória, contra o máximo de 2Gb para processos de 32 *bits*.

Segundo Becker (2007), a tecnologia de 64 *bits* permite que a maioria dos dados de processamento possam ser armazenados na memória, sem qualquer necessidade de armazenamento temporário em disco.

Isso pode aumentar consideravelmente o desempenho e abrir novos cenários de processamento de dados.

Há, portanto, boas razões para migrar atuais produtos de *software* de 32 *bits* para uma plataforma de 64 *bits*.

Muitas aplicações em *C* ou *C++* podem ser migradas com facilidade para a plataforma de 64 *bits*, no entanto alguns *software* baseado em módulo podem causar mais problemas. (BECKER, 2007)

Uma questão importante com relação a migração diz respeito aos componentes do *software* de 32 *bits* que não podem ser migrados, pois perdeu-se o código fonte ou uma das dependências não pode ser migrada. (BECKER, 2007)

> O *software* de 32 *bits* ainda é suportado numa plataforma de 64 *bits* como processos de 32 *bits* pode ser executado dentro do *Windows* dedicado (WOW64) que faz parte de todos os sistemas operacionais *Windows* de 64 *bits*. No entanto, um processo de 64 *bits* não pode carregar um módulo de 32 *bits* em seu espaço de processamento e vice e versa. (BECKER, 2007)

A única forma de acontecer a comunicação entre os módulos de 32 *bits* e 64 *bits* é através da comunicação entre processos (*IPC*). Em outras palavras, os processos de 32 *bits* e de 64 *bits* pode trocar dados usando técnicas de *IPC*, tais como *Out-of-process COM*, *sockets*, mensagens do *windows*, ou arquivos mapeados em memória. (BECKER, 2007)

Segundo Becker, 2007, um produto de *software* de 32 *bits* contém o módulo de vigilância principal que chama a *DLL WeatherStationControl*. Tanto o módulo principal e da *DLL* são 32 *bits* e o process então poderá ser executado em ambas as plataformas de 32 *bits* e de 64 *bits*, dentro do *WOW64*.

Concluindo, se o módulo principal e *DLL* forem migrados para a plataforma de 64 *bits*, eles poderão então, ser executados

em 64 *bits*. No entanto, se apenas o módulo principal for migrado para 64 *bits*, ele não será capaz de carregar a *DLL* de 32 *bits*. (BECKER, 2007).

E como foi necessário que o *bird*, com uma biblioteca compilada em 32 *bits*, interajisse com a *Unity3D* versão 64 *bits*, foi preciso criar um novo processo, de 32 *bits*, e abrir um canal de comunicação entre os dois processos.

## 6.8 Comunicação Entre Processos

O sistema operacional *Windows* fornece mecanismos para facilitar a comunicação e partilha de dados entre aplicações. As atividades habilitadas para estes mecanismos são chamados de comunicação entre processos (*IPC*). (MICROSOFT, 2010),

Segundo a Microsoft (2010), os seguintes mecanismos de *IPC* são suportados pelo Windows:

**Tabela 13 –** Mecanismos de *IPC* (Inter-process communication)

| | |
|---|---|
| Clipboard | Funciona como um repositório central para compartilhamento de dados entre aplicativos. |
| COM | Um componente de *software* que usa *COM* pode se comunicar com uma ampla variedade de outros componentes, mesmo com aqueles que ainda não tenham sido escritos. |
| Data Copy | Permite que um aplicativo envie informações para outro aplicativo usando a mensagem *WM_COPYDATA*. Este método requer a cooperação entre o aplicativo de envio e o aplicativo de recebimento. |
| DDE | É um protocolo que permite que os aplicativos troquem dados em uma variedade de formatos. Os aplicativos podem usar *DDE* para trocas de dados de uma só vez ou para a troca permanente onde poderão se atualizar sempre que existirem novos dados. |

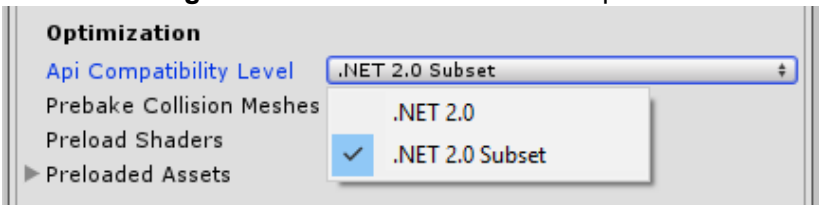
| | |
|---|---|
| File Mapping | Permite tratar o conteúdo de um arquivo como se fosse um bloco de memória no espaço de endereço do processo. |
| Mailslot | fornecer um único caminho para a comunicação. Qualquer processo que cria um processador de mensagens é um servidor de processador de mensagens. |
| Pipes | Existem dois tipos de *pipes* para a comunicação de duas vias: *Pipes* anônimo que permite que processos relacionados possam transferir informações entre si e pipes nomeados que são usados para transferir dados entre os processos que não estão relacionados entre si ou em diferentes computadores. |
| RPC | Permite que os aplicativos chamem funções remotamente e pode operar processos em um único computador ou em diferentes computadores em uma rede. |
| Windows Sockets | É uma interface independente de protocolo e tira proveito das capacidades de comunicação dos protocolos subjacentes. |

**Fonte**: Microsoft (2010).

Como mostra a tabela 13, o protocolo *Pipe* é adequado para o propósito dessa aplicação. É necessário observar, em um processo, um fluxo de dados oferecido por outro processo.

O *Pipe* é bem suportado em *C#* e, consequentemente, é suportado na *Unity3D* também. Contudo, é preciso habilitar o *framework Mono 2.0* completo para obter acesso ao recurso *pipe* (*System.IO.Pipe*). A *Unity*, por padrão, remove vários pacotes do *framework* afim de reduzir o tamanho final do *build*. Neste projeto, para suportar o *Pipe*, foi necessário habilitar o *framework* completo ajustando a configuração em *Player Settings* como mostra a figura 16.

**Figura 16 –** Habilitar .NET 2.0 Completo




**Fonte**: Elaborado pelo autor

O *Pipe* funciona com um servidor e um cliente. Neste projeto, o *pipe* precisa ser nomeado (*Named Pipe*) porque os processos são distintos e não relacionados. Um *pipe* oferece um fluxo constante de *bytes* de um processo ao outro e pode ser unidirecional ou bidirecional.

Aqui, o fluxo irá fluir unidirecionalmente do servidor para o cliente em forma de *strings*. O cliente irá capturar a *string* e interpretá-la. Como a figura 17 ilustra.

**Figura 17 –** Exemplo de código para Comunicação *Pipe*

```
1  /// SERVER, running on thread.
2  public void OpenPipeServer(string named)
3  {
4      NamedPipeServerStream pipeServer =
5          new NamedPipeServerStream(named, PipeDirection.Out);
6      pipeServer.WaitForConnection();
7      StreamWriter ww = new StreamWriter(pipeServer);
8      connected = true;
9      while (connected)
10     {
11         ww.Write(GetSerializedBirdFrame());
12         Thread.Sleep(100);
13     }
14     pipeServer.Close();
15 }
16
17 /// CLIENT, running on thread
18 public void ConnectPipeToServer(string named)
19 {
20     NamedPipeClientStream pipeClient = new NamedPipeClientStream(".", named, PipeDirection.In);
21     pipeClient.Connect();
22     StreamReader reader = new StreamReader(pipeClient);
23     connected = true;
24     while (connected)
25     {
26         var frame = GetDeserializedBirdFrame(reader.ReadLine());
27         Update(frame);
28     }
29 }
```

**Fonte**: Elaborado pelo Autor

## 6.9 *Middleware*

Foi iniciado, para suportar o *bird* utilizando esta técnica, um novo projeto, chamado aqui de *Middleware*, por ser o meio de comunicação entre a aplicação principal e o *driver* do aparelho. Esse projeto será um *software* que rodará em forma de *console*, ou seja, sem interface gráfica. Este *software* será iniciado pelo simulador de solda através da *Unity3D*. Portanto não é necessário nenhuma interface gráfica para comunicar com o usuário.

Esta aplicação irá iniciar, configurar e obter as informações do aparelho. Ou seja, irá carregar a biblioteca do *bird*, executar todos os passos necessários para inicializar o aparelho e requisitar os *frames* para obter as informações e poder repassar para o processo interessado.

O *Middleware*, deverá fazer a inicialização do *bird* assim que o processo for instanciado. Este será o servidor *pipe*, servindo os dados do *tracker* para a aplicação através de um 'fluxo de *string'*. E, quando o processo for encerrado, deverá fazer a finalização do *bird* também.

Assim, a funcionalidade do *Middleware* resume-se a gerenciar o *bird*, iniciar o servidor *pipe* e iniciar o fluxo de informação do *tracker* ao cliente *pipe*: o simulador de solda.

O *Middleware* ainda ficará responsável por dar o tratamento final e formatar os dados obtidos do *tracker*. Será então, convertido a posição para centímetros e o ângulo para quaternion. Da documentação e exemplos, retira-se a seguinte conversão:

Posição

```
x = birdReading.position.nX * lengthFactor / 32767.0f;
y = birdReading.position.nY * lengthFactor / 32767.0f;
z = birdReading.position.nZ * lengthFactor / 32767.0f;
```

Onde *lengthFactor* pode ser 36 para polegadas ou 91.44 para centímetros. Foi utilizados centímetros.

Quaternion

```
w = birdReading.quaternion.nQ0 / 32767.0f;
x = birdReading.quaternion.nQ1 / 32767.0f;
y = birdReading.quaternion.nQ2 / 32767.0f;
z = birdReading.quaternion.nQ3 / 32767.0f;
```

Será responsabilidade do Simulador de Solda inicializar o *Middleware*, conectar-se, interpretar as informações e encerrar o *Middleware* quando for o momento.

## 6.10 *Unity3D*

A *Unity3D* é orientada a Componentes chamados *Monobehaviour*. Esses componentes são anexados à um *GameObject* através do editor. Através desse componente, anexado a um objeto na cena, tem-se acesso e controle a diversas propriedades como a propriedade *Transform* que, entre outras coisas, controla a rotação e posição do objeto na cena.

É criado um *script* chamado *Tracker*, estendendo *Monobehaviour*, que irá conectar-se ao servidor *pipe* e irá interpretar as informações e aplicá-las ao objeto associado.

Para isso utiliza-se o método *Update*. Na *Unity3D* o método *update*, implementado em uma classe derivada de *Monobehaviour*, é chamado todos os *frames*.

Através do cliente *pipe*, lêem-se uma linha do fluxo de *strings* enviado pelo servidor. O servidor envia as informações em uma única linha, com as propriedades separadas por ponto e vírgula: "x;y;z;qw;qx;qy;qz". Onde x,y,z formam um vetor e qw,qx,qy,qz formam um quaternion. A figura 18 exemplifica a implementação.

**Figura 18 –** Exemplo de código para Utilização dos Dados de *Tracking* Magnético na *Unity3D*

```
StreamReader reader;
Vector3 framePosition;
Quaternion frameRotation;

void DeserializeFrame(string frame)
{
    string[] info = frame.Split(new string[] { ";" },
        StringSplitOptions.RemoveEmptyEntries);

    framePosition = new Vector3(float.Parse(info[0]),
        float.Parse(info[1]), float.Parse(info[2]));

    frameRotation = new Quaternion(float.Parse(info[3]),
        float.Parse(info[4]), float.Parse(info[5]), float.Parse(info[6]));
}

// Running on Thread
void GetFrame()
{
    Monitor.Enter(reader);
    DeserializeFrame(reader.ReadLine());
    Monitor.Exit(reader);
    Thread.Sleep(100);
}

void Update()
{
    if (Monitor.TryEnter(reader, 0)) // non blocking
    {
        transform.position = framePosition;
        transform.rotation = frameRotation;
        Monitor.Exit(reader);
    }
}
```

**Fonte**: Elaborado pelo Autor

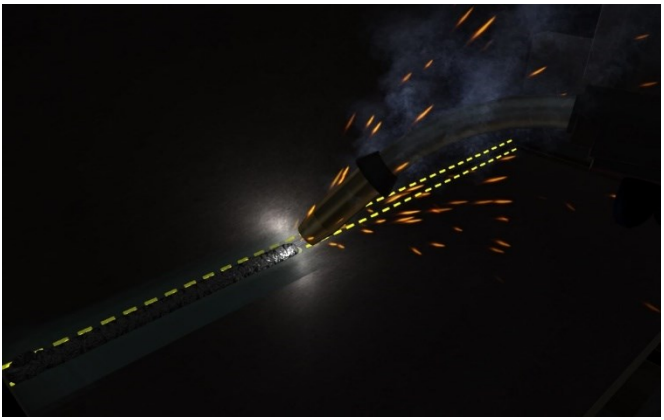
## 6.11 Integração com Simulador

Foi importado os *scripts* da integração dentro do projeto do Simualdor. O *script Tracker* foi anexado ao pivô do objeto da pistola de solda, conforme planejado anteriormente.

O resultado da integração do *tracker* com a *Unity3D* foi bem sucedido. Os movimentos e rotação correspondiam ao esperado. Quando a pistola real era movida ou rotacionada, a pistola virtual correspondia à interação, movendo-se e rotacionando de acordo. Isso porque a escala do simulador foi considerada em centímetros, assim como as informações do *tracker*.

O simulador também estava mais robusto e melhor preparado. Gráficos e medições haviam sido aprimorados. A figura 19 mostra o simulador aperfeiçoado em uso.

**Figura 19** – Imagem do Simulador Aperfeiçoado em Uso



**Fonte:** Autoria Própria.

O tracker foi incorporado à uma pistola de soldagem, como mostra a figura 20. E o pivô, que anteriormente correspondia à ponta da pistola de solda virtual no simulador, foi movido de forma a corresponder aproximadamente ao local do posicionamento do tracker dentro da pistola real.

**Figura 20** – Imagem do *Tracker Incorporado a Pistola de Soldagem*

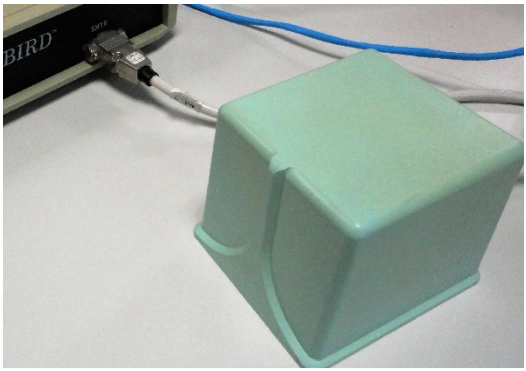

***Fonte**: Autoria Própria.*

A integração do simulador de solda, do O*culus Rift* e do *tracking* magnético ocorreu sem erros. Foi possível utilizar os equipamentos com o simulador sem nenhum problema inesperado. Contudo, foi preciso assegurar que o *Oculus Rift* estivesse na posição e orientação inicial desejada (na direção da mesa de solda virtual) para assegurar que o usuário pudesse observar a tocha de soldagem. Algumas vezes, dependendo da inicialização do simulador, a orientação do usuário com relação o cenário ficava incorreta. Para corrigir este problema e sempre que necessário corrigir esta rotação, foi criado um simples script que utiliza uma função de reset da própria *Unity3D*, que reseta a rotação de capacetes virtuais, chamado *Recenter:*

```
UnityEngine.VR.InputTracking.Recenter()
```

O *tracker* envia a posição e rotação em valores absolutos a partir de sua origem. A origem é o centro do aparelho que gera o campo magnético. Essa peça pode ser observada na figura 21.

**Figura 21** – Imagem da Peça




***Fonte**: Autoria Própria.*

Então, quando o *tracker* informa que o valor da posição *X* é 10 cm. O correspondente na *Unity3D* é 10 unidades (consideradas centímetros nesse projeto) a partir da origem da cena. E esse posicionamento, na cena do simulador, não correspondia à proximidade da mesa de solda.

Por esse motivo, foi necessário criar um nível de hierarquia para o pivô utilizado. Esse novo nível, chamado de *offset* representa a origem do aparelho, e portanto a origem da pistola de solda virtual. O *offset* é posicionado em um local na mesa de solda virtual, de forma que os movimentos feitos fiquem na proximidade da mesa, no objeto de interesse.

À essa altura a noção de imersão almejada, por observação, estava bastante satisfatória. Já era possível utilizar a tocha de soldagem na mesa virtual e treinar as técnicas de soldagem.

Entretanto, a sensação tátil ainda quebrava um pouco a imersão total possível. A mesa real e a mesa virtual não correspondiam, nem a posição da solda real e virtual com relação à posição da mesa. Para que essas correspondências fossem alcançadas, seria necessário uma calibração de equiparação, o que foge do escopo deste trabalho.

Não foi observado erro significativo quanto a precisão e acurácia no deslocamento ou na rotação da pistola de solda.

Ao final desta integração, era possível utilizar o simulador com realidade virtual e pistola de solda com *tracking* magnético com boa precisão e imersão.

## 7. Conclusão

Ao fim deste trabalho pode-se concluir que a integração entre um *tracking* magnético e um simulador de solda com realidade virtual é possível e traz benefícios como a imersão, maior liberdade de controle e maior proximidade com a realidade.

A realidade virtual vem se destacando como um importante aliado em diversas áreas que junto a simuladores, podem oferecer novas experiências principalmente na área da educação e segurança. Como exemplo é possível citar os simuladores de medicina, simuladores de trânsito e também os industriais como os de solda. Unindo a realidade virtual, simulador de solda e periféricos como o *tracking* magnético,

pode-se envolver o usuário em uma nova experiência imersiva e realista, oferecendo benefícios significativos e resultados positivos para o aprendizado.

O *tracking* magnético, mini bird 500, é um equipamento antigo, com tecnologia e documentação datada de 2001. Já a engine *Unity3D*, é uma tecnologia nova e ativa. A *Unity3D* é uma ferramenta amplamente utilizada no desenvolvimento de jogos e aplicações interativas, *serious games* e simuladores. Foi também a ferramenta utilizada para implementar o simulador de solda usado neste trabalho e o desafio foi integrar ambas as tecnologias.

O empecilho encontrado e não previsto foi o fato de que as *DLL's* do mini bird foram compiladas em 32 *bits*, já a versão da *Unity3D* é a de 64 *bits*. Não seria possível mudar nem um dos softwares estabelecidos, contudo seria necessário fazê-los comunicar entre si. A solução adotada foi abrir uma comunicação Pipe entre eles.

Desta forma, o processo responsável pelo mini bird, compilado em 32 *bits*, inicializa o dispositivo corretamente e abre um servidor Pipe para enviar constantemente as informações do tracker. Enquanto a *Unity3D* conecta-se ao servidor, recebe e interpreta as informações.

O que inicialmente parecia um problema, resolveu-se de maneira interessante para projetos futuros. A abordagem escolhida possibilita que o software responsável pelo mini bird 500 não precise ser re-elaborado nem re-compilado para utilizá-lo em outros projetos. Basta que o projeto conecte-se ao processo e interprete os dados. É possível ainda a troca do processo responsável pelo mini bird para, por exemplo, suportar outros equipamentos. Basta seguir o protocolo estabelicido de comunicação.

Contudo, como foi mostrado no trabalho, a inicialização do equipamento pode ser complexa, com muitos parâmetros e customizações. Para este trabalho estas customizações foram desnecessárias, mas outro projeto pode fazer bom uso delas. Por esse motivo, é sugerido que em trabalhos futuros seja elaborado um protocolo de comunicação mais robusto, permitindo que o cliente possa enviar informações para configuração do aparelho.

Com os testes bem sucedidos em um projeto em branco na *Unity3D*, fizemos a integração com o simulador de solda.

Como o simulador considerava sua unidade de medida centímetros e a rotação de um objeto na *unity3d* é em quaternion, nenhum ajuste precisou ser feito com relação à esses dados. Caso contrário, conversões precisariam ser feitas para a correta correspondência entre o tracker e o objeto virtual no simulador de solda.

Apesar da unidade de medida ser a mesma, a correspondência entre a posição do tracker e a posição da pistola de solda no mundo virtual pareciam não corresponder. A pistola de solda, após a integração, ficava em um local muito longe da mesa de solda. O que foi observado é que a posição enviada pelo tracker é absoluta com relação à sua origem. E esse valor, no simulador não era próximo à mesa. A solução foi fazer um deslocamento, utilizando o sistema de hierarquia de objetos da *unity3d*, para deixar a pistola mais próxima à mesa de soldagem.

Contudo essa solução não prevê a mudança de ambiente. Se a origem do aparelho de *tracking* for deslocada, a posição da pistola de solda no simulador será diferente; o ajuste de deslocamento terá que ser outro. Para resolver esse problema, ou mantém-se a origem do aparelho sempre fixa com relação a um ponto de utilização do tracker, ou é elaborado um sistema de calibração.

Após a integração observou-se um ganho na imersão e no treinamento de soldagem utilizando o simulador, caracterizado na tabela 14:

**Tabela 14 –** Comparação de Utilização e Não Utilização do *Tracker* Magnético

| **Simulador com Tracker Magnético** | **Simulador utilizando teclado e *mouse*** |
|---|---|
| Maior imersão | Baixa imersão |
| Controle livre e natural | Controle restrito e não natural |
| Utilização da ferramenta real de trabalho (tocha de soldagem adaptada ao tracker magnético) | Utilização de equipamento comum ao uso de computadores pessoais, mas não para soldagem |
| Treinamento bem próximo ao real | Treinamento longe do real |
| Acertos e erros correspondem à habilidade de soldagem | Acertos e erros correspondem à habilidade de uso de computadores |

**Fonte:** Elaborada pelo Autor

O simulador utilizado neste trabalho foi desenvolvido em paralelo e de forma independente. No momento da integração o simulador estava bastante maduro, apresentando simulação básica de soldagem e conseguindo avaliar de forma consistente a soldagem, informando erros e acertos sobre algumas variáveis de soldagem, como velocidade e ângulo de avanço.

Contudo, o simulador ainda necessita de diversos ajustes e funcionalidades para oferecer uma simulação mais consistente e coerente com a realdiade, e um aprendizado que satisfaça as necessidades do mercado.

Como software educacional o simulador precisará oferecer exercícios e uma forma de avaliação. Exercícios que contemplem do básico da soldagem à exercícios mais avançados utilizando todo o potencial da realidade virtual, apresentando formas geométricas complexas ou ambientes inusitados. Além disso seria interessante um tutor virtual. Uma maneira de acompanhar o usuário, mostrar seus acertos e erros, bem como meios de aperfeiçoar as técnicas para melhorar continuamente.

Para aumentar a imersão e aprendizado, seria interessante incorporar ao simulador um hardware capaz de

simular o equipamento real de soldagem. Podendo, por exemplo, ajustar no equipamento a voltagem ideal para o exercício e essa escolha impactar no simulador.

Como podemos ver, além dos pontos levantados, ainda há muitos aspectos à melhorar no simulador. A integração com o *tracking* magnético foi um passo importante para a melhoria e viabilidade do simulador. Entretanto ainda há muito o que fazer para que seja uma ferramenta segura de aprendizado eaperfeiçoamento de técnicas de soldagem.

## 7.1 Sugestões para Trabalhos Futuros

Como sugestão de trabalhos futuros, ao que tange o *tracking* magnético, apresentado neste trabalho, pode-se citar:

a. *Feedback* tátil. Para uma imersão e aprendizado mais completos, é interessante que o usuário possa sentir a resposta de todo o ambiente ao seu redor. Isso inclui o que toca e o que sente no momento da solda. A resistência que a mesa oferece ao tocá-la, tanto com a tocha de soldagem quanto com as mãos, etc.
b. Integração do *tracker* com outras ferramentas, como por exemplo *Unreal engine*, ou *Ogre engine*. Isso estenderia as possibilidades de desenvolvimento de novos simuladores e ferramentas. Com este trabalho criamos um processo que utiliza *Pipe* para servir informações do *tracker* ao cliente. Seria possível, portanto, trocar o cliente e utilizar outras ferramentas e *frameworks*, integrando com algum outro simulador.
c. Estender o que é oferecido como comunicação entre o *tracker* e a aplicação e padronizar para poder acolher o maior número possível de *trackers*, possibilitando a troca do mesmo com maior facilidade.
d. Desenvolver um botão para acionar a pistola de solda, incrementando a imersão. Seria necessário desenvolver

um *hardware* e a interface de comunicação entre este e o simulador.

Há várias possibilidades de trabalhos futuros para melhoraria e aperfeiçoamento de simuladores de solda. Algumas sugestões são:

a. Análise e elaboração de exercícios adequados ao simulador de soldagem virtual
b. Elaboração de *design* de interface com o usuário com foco no ensino e acompanhamento, formas de mostrar erros e acertos e acompanhar o usuário durante a soldagem.
c. Desenvolvimento de *hardware* que simule o equipamento utilizado, permitindo que o usuário faça os ajustes necessários no simulador diretamente de um equipamento, aproximando ainda mais de uma experiência real.

**Referências**

ASCENSION TECHNOLOGY CORPORATION. **MiniBird: Installation and Operation Guide.**
USA. 2001.

BADI, Haitham et al. **A survey on human–computer interaction technologies and techniques.**
Iraque, 2016.

BALADEZ, Fabio. **O Passado, o presente e o futuro dos simuladores**.
São Caetano do Sul: FATEC, 2009.

BALBINOT, Amanda B.; TIMM, Maria Isabel; ZARO, Milton Antonio. **Aplicação de Jogos e Simuladores como Instrumento para Educação e Segurança no Trânsito**.
Porto Alegre: UFRGS. 2009.

BECKER, Mike. **Accessing 32-bit DLLs from 64-bit code.**
Alemanha. 2007.

CAMPOS, Itamar Albertino de. **Realidade Virtual e Aumentada - Conceitos, Tecnologia e Aplicações**. Cuiabá: Instituto Cuiabano de Educação. 2010.

CARDOZO, Daniel. **Unity e UDK inseridos na Realidade Aumentada**. 2015
Disponível em: <http://danielcardozo.com.br/2015/04/02/artigo-unity-udk-inseridos-realidade-aumentada/>
Acesso em: 10 mar. 2016.

CHAMBERS, Terrence L. et al. **Real-time simulation for a virtual reality-based MIG welding training system.**
USA, 2012.

DARGAR, Saurabh et al. **Towards immersive virtual reality (iVR): a route to surgical expertise.**

USA, 2015.

DUSTY, Sargent. **Tracker-endoscope calibration for colonoscopy.**
California, 2008.

GARCIA, Luis Fernando Fortes; LUZ, Marlon. **Realidade Aumentada em Dispositivos Móveis**.
Porto Alegre, 2008.

HERNOUX, Franck; CHRISTMANN, Olivier. **A seamless solution for 3D real-time interaction: design and evaluation.**
França, 2014

INFORMATIZAÇÃO MILITAR – QC6.
Disponível em: <http://csufes20092.pbworks.com/w/page/5335740/Informatiza%C3%A7%C3%A3o%20Militar%20-%20QC6>
Acesso em: 10 mar. 2016.

JASON, Jerald et al. **Developing Virtual Reality Applications with Unity.**
USA, 2015.

JUNIOR, Antonio Deusany de Carvalho; SOUZA, Daniel Faustino Lacerda de; MACHADO, Liliane dos Santos. **Integração de Rastreadores Magnéticos a um Framework para Desenvolvimento de Sistemas de Realidade Virtual**. Paraíba: Universidade Federal da Paraíba. 2009.

KHUNDAM, Chaowanan. **First person movement control with palm normal and hand gesture interaction in virtual reality.**
Tailandia, 2015.

KINDRATENKO, Volodymyr. **A survey of electromagnetic position tracker calibration techniques.**
USA, 2000.

KINDRATENKO, Volodymyr. **Calibration of electromagnetic tracking devices.**
USA, 1999.

LUCAS, Felipe Rabay et al. **Uso de Simuladores de Direção Aplicada ao Projeto
de Segurança Viária**. São Paulo: USP. 2013.

MACEDO, Josué Antunes de; DICKMAN, Adriana Gomes; ANDRADE, Isabela Silva Faleiro de. **Simulações Computacionais como Ferramentas para o Ensino de Conceitos Básicos de Eletricidade**.
Minas Gerais, 2012.

MEDEIROS, Daniel Pires de Sá. **Uma ferramenta de interação 3D para ambientes virtuais de engenharia utilizando dispositivos móveis.**
PUC Rio de Janeiro, 2013.

MEDEIROS, Ivan Luiz de; REIS, Alessandro Vieira dos; BRAVIANO Gilson; GONÇALVES, Berenice Santos. **Revisão Sistemática e Bibliometria facilitadas por um Canvas para visualização de informação**.
UFSC, 2014.

MICROSOFT. Developer Network. **Interprocess Communications.**
USA. 2010.

MODENESI, Paulo J.; MARQUES, Paulo Villani. **Soldagem I - Introdução aos Processos de Soldagem.** Minas Gerais: Belo Horizonte, 2006

MODENESI, Paulo J.; MARQUES, Paulo Villani; SANTOS, Dagoberto B. **Introdução à Metalurgia da Soldagem.** Minas Gerais: Belo Horizonte, 2012.

PAES, Luiz Eduardo dos Santos et al. Avaliação dos Efeitos Decorrentes do Aumento da Extensão Sólida de Arame-Eletrodo

(Stick out) – **Processo MIG/MAG Convencional com Transferencia por Vôo Livre.**
XLI CONSOLDA – Congresso Nacional De Soldagem
Salvador - BA, 2015

PEIXOTO, Arildomá Lobato. **Soldagem**.
IFSC. Pará, 2012.

PETERS, Terry; CLEARY, Kevin. **Image-Guided Interventions: Technology and Applications.**
Capítulo 2. 1.ed. Springer U. Springer-Verlag US. 2008.

PINHO, Marcio Sarroglia. **Realidade Virtual. Tecnologias de Rastreamento e Captura de Movimento**. Rio Grande do Sul: PUCRS, 2009
Disponível em <http://www.inf.pucrs.br/~pinho/CGII/PDFs/Aula4-Rastreamento.pdf>
Acesso em: 03 mar. 2016.

REIS, Alessandro Vieira dos. **Interfaces Tangíveis em Simuladores Veiculares**: Componentes para um Protocolo de Avaliação de Usabilidade.
UFSC. 2016.

REVISTA DA SOLDAGEM. **A Soldagem GMAW (ou MIG-MAG)**.
Associação Brasileira de Soldagem. Ano I, N° 4.

RODRIGUES, Gessica Palhares; PORTO, Cristiane de Magalhães. **Realidade Virtual: Conceito, Evolução, Dispositivos, Aplicações**. Aracaju. 2013.

SANTOS, Luiz Fernando Amaral dos. **Apostila Metodologia da Pesquisa Cientifica II**.
Itapeva, 2006
http://www.socrates.cnt.br/apostmetoditapeva.pdf

SANTOS, Valdeci. **O que é e como fazer revisão da literatura na pesquisa teológica**.
São Paulo, 2012.

SERVSOLDA. **Processo MIG/MAG - Parâmetros de Soldagem.**
Brasília - DF. 2014

SHIN, Jongkyu et al. **Application of precise indoor position tracking to immersive virtual reality with translational movement support.**
USA, 2016

SIGNORI, Gláuber; RIGO, Wanderson. **Introdução ao desenvolvimento de jogos com a Engine Unity 3D**. Passo Fundo: IMED, 2012.

SILVA, Alexandre Carvalho et al. **Uso da Engine de Jogos Unity3D para Sistemas de Realidade Virtual Aplicado a Monitoramento e Controle de Subestações de Energia Elétrica**. Minas Gerais: Belo Horizonte. 2013.

SILVA, Evelyn Macedo da; MACHADO, Leandro; SOARES, Thayson Bonifácio. **Salve os Mini Downs: Jogo para Desenvolvimento de Características Cognitivas em Portadores de Síndrome de Down.**
IESGF, 2013

SOSNICKI, Olivier et al. **AC magnetic field detection system applied to motion tracking.**
França. 2010.

SPIEGELMOCK, Mischa. **Leap Motion Development Essentials: leverage the power of Leap Motion to develop a fully interactive application.**
Ucrania, 2013

TEIXEIRA, Ilka Nicélia D'Aquino Oliveira; FELIX,Jorge Vinícius Cestari. **Simulação como estratégia de ensino em enfermagem: revisão de literatura**. Paraná: UFPR. 2011.

UNITY. **O software líder global da indústria de jogos.**

São Francisco. EUA. 2016.

VARGA, Cassia Regina Rodrigues et al. **Relato de Experiência: o Uso de Simulações
no Processo de Ensino-aprendizagem em Medicina**. São Paulo: Universidade Federal de São Carlos. 2009.

WOODARD, Will; SUKITTANON, Somsak. **Interactive virtual building walkthrough using Oculus Rift and Microsoft Kinect.** USA, 2015.

WU, Xiaohui; TAYLOR, R.H. **A framework for calibration of electromagnetic surgical navigation system.**
USA, 2003.

XIE, Benkai; ZHOU, Qiang; YU, Liang. **A real-time welding training system base on virtual reality.**
USA, 2015.

YIN, Juan; WANG, Xuemin; YI, Zhang. **Design of six degrees of freedom electromagnetic tracker for virtual reality system.**
China, 2012.

## Apêndice A – Código da integração

### Unity3D, PipeClient.cs

```
using System.Security.Principal;
using System.Diagnostics;
using System.Threading;
using System.IO.Pipes;
using System.Text;
using System.IO;
using System;


public class PipeClient
{
	public string validateServerString;
	public string namedPipeName;
	public string serverPath;

	private System.Diagnostics.Process server;
	private NamedPipeClientStream pipeClient;
	private static int numClients = 1;
	private Quaternion frameRotation;
	private bool connected = false;
	private Vector3 framePosition;
	private Thread getFrameThread;
	private StreamReader reader;


	public void Start()
	{
		UnityEngine.Debug.Log("Starting...");

		server = System.Diagnostics.Process.Start(serverPath);
		pipeClient = new NamedPipeClientStream(".",
namedPipeName, PipeDirection.In);

		UnityEngine.Debug.Log("Connecting to server..." +
namedPipeName);
		pipeClient.Connect();

		reader = new StreamReader(pipeClient);
		// Validate the server's signature string
```

```
                if (reader.ReadString() == validateServerString)
                {
                        connected = true;
                        getFrameThread = new Thread(GetFrame);
                        getFrameThread.Start();
                }
                else
                {
                        UnityEngine.Debug.Log("Server could not be
verified.");
                        connected = false;
                        enabled = false;
                }
        }

        void GetFrame()
        {
                while (connected)
                {
                        Monitor.Enter(reader);
                        TransformObject(reader.ReadLine());
                        Monitor.Exit(reader);
                        Thread.Sleep(100);
                }
        }

        void Update()
  {
     if (Monitor.TryEnter(reader,0)) // non blocking
     {
        transform.position = framePosition;
                transform.rotation = frameRotation;
                Monitor.Exit(reader);
     }
  }

  void TransformObject(string serializedValue)
  {
        string [] values = serializedValue.Split(new string[]{";"},
System.StringSplitOptions.RemoveEmptyEntries);
        framePosition = new Vector3(float.parse(values[0]),
float.parse(values[1]), float.parse(values[2])); // x,y,z
        frameRotation = new Quaternion(float.parse(values[3]),
float.parse(values[4]),float.parse(values[5]), float.parse(values[6])); // x,y,z,w
```

```
    }


    void OnDestroy()
    {
            connected = false;
        pipeClient.Close();
        server.Close();
        getFrameThread.Join();
    }

}
```

### Middleware, PipeServer.cs

```
using System;
using System.IO;
using System.IO.Pipes;
using System.Text;
using System.Threading;

public class PipeServer
{
    public static bool connected = false;

    private bool streaming;
    private string lastData;

    public static void Main()
    {
        Thread server = new Thread[numThreads];

        Console.WriteLine("\n*** Named pipe server stream for bird
communication***\n");
        Console.WriteLine("Waiting for client connect...\n");
        PipeServer pipeServer = new PipeServer();
        int i = 1;
        while (i > 0 && server != null)
        {
            if (server.Join(250))
            {
                Console.WriteLine("Server thread[{0}] finished.",
server.ManagedThreadId);
```

```
            server = null;
            i--;
        }
    }
    server = new Thread(pipeServer.ServerMiniBirdThread);
    server.Start();
}

private PipeServer()
{
    StartBird();
}


private StartBird() {
    if (started) return;
    int groupId = 0;
    bool standAlone = true;
    int numDevices = 0;
    ushort[] COM_port = new ushort[]{3};
    uint boudRate = 115200;
    uint readTimeout = 2000;
    uint writeTimeout = 2000;
    int groupMode =
(int)GroupModeSettings.NUM_GROUP_MODE_SETTINGS;
    if (!StartBird(groupId, standAlone, numDevices, COM_port, boudRate,
readTimeout, writeTimeout, groupMode)) {
        throw new Exception("Could not wakeup bird");
    }
    started = true;
}

private bool StartBird(int groupId, bool standAlone, int numDevices,
ushort[] COM_port
    , uint boudRate, uint readTimeout, uint writeTimeout, int groupMode){
    if (birdRS232WakeUp(groupId, standAlone, numDevices, COM_port,
boudRate, readTimeout, writeTimeout, groupMode)) {
        BIRDSYSTEMCONFIG sysconfig = new BIRDSYSTEMCONFIG();
        BIRDDEVICECONFIG devconfig = new BIRDDEVICECONFIG();
        if (birdGetSystemConfig(groupId, ref sysconfig, true) &&
            birdGetFastDeviceConfig(groupId, numDevices, ref devconfig)) {
            devconfig.byDataFormat =
(byte)BirdDataFormat.BDF_POSITIONQUATERNION;
            if (birdSetFastDeviceConfig(groupId, numDevices, ref devconfig)){
```

```
                streaming = birdStartFrameStream(groupId)
                return streaming;
            }
        }
    }

    return false;
}

private string SerializeBirdReading(BIRDREADING birdReading) {
    string value = "";
    float lengthFactor = 91.44f;

    value += (birdReading.position.nX * lengthFactor / 32767.0f) + ";";
    value += (birdReading.position.nY * lengthFactor / 32767.0f) + ";";
    value += (birdReading.position.nZ * lengthFactor / 32767.0f) + ";";
    value += (birdReading.quaternion.nQ0 / 32767.0f) + ";";
    value += (birdReading.quaternion.nQ1 / 32767.0f) + ";";
    value += (birdReading.quaternion.nQ2 / 32767.0f) + ";";
    value += (birdReading.quaternion.nQ3 / 32767.0f);

    return value;
}

private string GetSerializedBirdFrame() {
    if (!streaming) throw new Exception("Streaming was not started");
    int groupId = 0;

    if (birdFrameReady(0)) {
        BIRDFRAME birdFrame = new BIRDFRAME();
        birdGetMostRecentFrame(groupId, ref birdFrame);
        BIRDREADING birdReading = birdFrame.readings[0];
        lastData = SerializeBirdReading(birdReading);
    }

    return lastData;
}

public void ServerMiniBirdThread(object data)
{
    NamedPipeServerStream pipeServer =
        new                    NamedPipeServerStream("weldingSimulator",
PipeDirection.Out);
```

```
        int threadId = Thread.CurrentThread.ManagedThreadId;
        pipeServer.WaitForConnection();

        Console.WriteLine("Client connected on thread[{0}].", threadId);
        StreamWriter ww = new StreamWriter(pipeServer);
        bool connected = true;

        try
        {
            ww.Write("I am the one true server!");
            Thread.Sleep(100);
            while(connected)
            {
                ss.WriteString(GetSerializedBirdFrame());
                Thread.Sleep(100);
            }
        }

        // Catch the IOException that is raised if the pipe is broken
        // or disconnected.
        catch (IOException e)
        {
            Console.WriteLine("ERROR: {0}", e.Message);
        }

        pipeServer.Close();
    }
}
```

### Middleware, BirdWrapper.cs

```
using System;
using System.Runtime.InteropServices;

namespace BirdServer {
        public static class BirdWrapper {
                public enum BirdDataFormat {
                        // Bird data formats
                        BDF_NOBIRDDATA = 0,   // no data (NOTE:
RS232 and ISA modes have no way of specifying this format)
                        BDF_POSITION = 1,   // position only
                        BDF_ANGLES = 2,   // angles only
```

```
                BDF_MATRIX = 3,   // matrix only
                BDF_POSITIONANGLES = 4,   // position and
angles
                BDF_POSITIONMATRIX = 5,   // position and
matrix
                BDF_QUATERNION = 7,   // quaternion only
                BDF_POSITIONQUATERNION = 8    // position
and quaternion
        }

        [StructLayout(LayoutKind.Sequential, Pack = 1)]
        public struct BIRDPOSITION {
                public short nX;          // x-coordinate

                public short nY;          // y-coordinate

                public short nZ;          // z-coordinate
        }

        // Bird angles structure
        [StructLayout(LayoutKind.Sequential, Pack = 1)]
        public struct BIRDANGLES {
                public short nAzimuth;    // azimuth angle

                public short nElevation;    // elevation angle

                public short nRoll;       // roll angle
        }


        [StructLayout(LayoutKind.Sequential, Pack = 1)]
        public struct BIRDMATRIX {
                [MarshalAs(UnmanagedType.ByValArray,
SizeConst = 9)]
                public short[,] n;
        }

        [StructLayout(LayoutKind.Sequential, Pack = 1)]
        public struct BIRDQUATERNION {
                public short nQ0;       // q0
                public short nQ1;       // q1
                public short nQ2;       // q2
                public short nQ3;       // q3
        }
```

```
		[StructLayout(LayoutKind.Sequential, Pack = 0)]
		public struct BIRDREADING {
			public BIRDPOSITION position;    // position of
receiver
			public BIRDANGLES angles;        // orientation of
receiver, as angles
			public BIRDMATRIX matrix;        // orientation of
receiver, as matrix
			public BIRDQUATERNION quaternion; //
orientation of receiver, as quaternion
			public ushort wButtons;    // button states

		}

		[StructLayout(LayoutKind.Sequential, Pack = 0)]
		public struct BIRDFRAME {
			public uint dwTime;        // time at which readings
were taken, in msecs
			[MarshalAs(UnmanagedType.ByValArray,
SizeConst = 127)]
			public BIRDREADING[] readings; // reading from
each bird
		}

		[StructLayout(LayoutKind.Sequential, Pack = 0)]
		public struct BIRDSYSTEMCONFIG {
			public byte bySystemStatus;
			public byte byError;
			public byte byNumDevices;
			public byte byNumServers;
			public byte byXmtrNum;
			public ushort wXtalSpeed;
			public double dMeasurementRate;
			public byte byChassisNum;
			public byte byNumChassisDevices;
			public byte byFirstDeviceNum;
			public ushort wSoftwareRev;

			[MarshalAs(UnmanagedType.ByValArray,
SizeConst = 127)]
			public byte[] byFlockStatus;
		}
```

```
                [StructLayout(LayoutKind.Sequential, Pack = 0)]
                public struct BIRDDEVICECONFIG {
                        public byte byStatus;           // device status (see
bird device status bits, above)
                        public byte byID;               // device ID code (see
bird device ID's, above)
                        public ushort wSoftwareRev;     // software
revision of devic
                        public byte byError;            // error code flagged
by device
                        public byte bySetup;            // setup information
(see bird device setup bits, above)
                        public byte byDataFormat;       // data format (see
bird data formats, above)
                        public byte byReportRate;       // rate of data
reporting, in units of frames
                        public ushort wScaling;         // full scale
measurement, in inches
                        public byte byHemisphere;       // hemisphere of
operation (see bird hemisphere codes, above)
                        public byte byDeviceNum;        // bird number
                        public byte byXmtrType;         // transmitter type
(see bird transmitter type bits, above)

                        [MarshalAs(UnmanagedType.ByValArray,
SizeConst = 7)]
                        public ushort[] wAlphaMin;      // filter constants
(see Birdnet3 Protocol pp.26-27 for values)

                        [MarshalAs(UnmanagedType.ByValArray,
SizeConst = 7)]
                        public ushort[] wAlphaMax;      // filter constants
(see Birdnet3 Protocol pp.26-27 for values)

                        [MarshalAs(UnmanagedType.ByValArray,
SizeConst = 7)]
                        public ushort[] wVM;            // filter constants
(see Birdnet3 Protocol pp.26-27 for values)

                        BIRDANGLES anglesReferenceFrame;    //
reference frame of bird readings
                        BIRDANGLES anglesAngleAlign;        // alignment
of bird readings
                }
```

```
        public enum GroupModeSettings {
            //    GMS_DEFAULT,                  // driver will
determine whether or not to use RS232 group mode
            GMS_GROUP_MODE_NEVER,               //
RS232 group mode will never be used
            GMS_GROUP_MODE_ALWAYS,              //
RS232 group mode will always be used
            NUM_GROUP_MODE_SETTINGS
        };

        [DllImport(@"Bird.dll", CallingConvention =
CallingConvention.Cdecl)]
        public static extern bool birdRS232GroupModeEnabled(int
GroupID);

        [DllImport(@"Bird.dll", CallingConvention =
CallingConvention.Cdecl)]
        public static extern bool birdShutDown(int GroupID);

        [DllImport(@"Bird.dll", CallingConvention =
CallingConvention.Cdecl)]
        public static extern bool birdRS232WakeUp(int nGroupID,
Boolean bStandAlone, int nNumDevices,
                                                  ushort[]
pwComport, uint dwBaudRate, uint dwReadTimeout,
                                                  uint
dwWriteTimeout, int nGroupMode);

        [DllImport(@"Bird.dll", CallingConvention =
CallingConvention.Cdecl)]
        public static extern bool birdGetSystemConfig(int
nGroupID, ref BIRDSYSTEMCONFIG psyscfg, bool bGetDriverCopy);

        [DllImport(@"Bird.dll", CallingConvention =
CallingConvention.Cdecl)]
        public static extern bool birdSetSystemConfig(int
nGroupID, ref BIRDSYSTEMCONFIG psyscfg);

        [DllImport(@"Bird.dll", CallingConvention =
CallingConvention.Cdecl)]
        public static extern bool birdGetFastDeviceConfig(int
nGroupID, int nDeviceNum, ref BIRDDEVICECONFIG pdevcfg);

        [DllImport(@"Bird.dll", CallingConvention =
CallingConvention.Cdecl)]
```

```
        public static extern bool birdSetFastDeviceConfig(int
nGroupID, int nDeviceNum, ref BIRDDEVICECONFIG pdevcfg);

        [DllImport(@"Bird.dll", CallingConvention =
CallingConvention.Cdecl)]
        public static extern bool birdGetDeviceConfig(int
nGroupID, int nDeviceNum, ref BIRDDEVICECONFIG pdevcfg, bool
bGetDriverCopy = false);

        [DllImport(@"Bird.dll", CallingConvention =
CallingConvention.Cdecl)]
        public static extern bool birdSetDeviceConfig(int nGroupID,
int nDeviceNum, ref BIRDDEVICECONFIG pdevcfg);

        [DllImport(@"Bird.dll", CallingConvention =
CallingConvention.Cdecl)]
        public static extern bool birdStartReading(int nGroupID, int
nDeviceNum);

        [DllImport(@"Bird.dll", CallingConvention =
CallingConvention.Cdecl)]
        public static extern bool birdReadingReady(int nGroupID,
int nDeviceNum);

        [DllImport(@"Bird.dll", CallingConvention =
CallingConvention.Cdecl)]
        public static extern bool birdGetReading(int nGroupID, int
nDeviceNum, ref BIRDREADING preading);

        [DllImport(@"Bird.dll", CallingConvention =
CallingConvention.Cdecl)]
        public static extern bool birdStartFrameStream(int
nGroupID);

        [DllImport(@"Bird.dll", CallingConvention =
CallingConvention.Cdecl)]
        public static extern bool birdStartSingleFrame(int
nGroupID);

        [DllImport(@"Bird.dll", CallingConvention =
CallingConvention.Cdecl)]
        public static extern bool birdGetMostRecentFrame(int
nGroupID, ref BIRDFRAME pframe);
```

```
                [DllImport(@"Bird.dll", CallingConvention =
CallingConvention.Cdecl)]
                public static extern bool birdGetFrame(int nGroupID, ref
BIRDFRAME pframe);

                [DllImport(@"Bird.dll", CallingConvention =
CallingConvention.Cdecl)]
                public static extern bool birdFrameReady(int nGroupID);

                [DllImport(@"Bird.dll", CallingConvention =
CallingConvention.Cdecl)]
                public static extern string birdGetErrorMessage();

                [DllImport(@"Bird.dll", CallingConvention =
CallingConvention.Cdecl)]
                public static extern bool birdStopFrameStream(int
nGroupID);
        }
}
```

## Apendice B – Artigo

# Integração de um *Tracking* Magnético a um Simulador de Soldagem Manual com Realidade Virtual Utilizando *Unity3D*

**Matheus Teixeira Fernandes**[1]

[1] Curso de Bacharelado em Ciências da Computação. Universidade Federal de Santa Catarina (UFSC). CEP 88040-900 – Florianópolis – SC – Brazil

matheus.tf@grad.ufsc.br

***Abstract.*** *This work has as main subject a magnetic tracking integration to a manual welding simulator, that together to virtual reality, can bring a whole new experience to learning and welding practice. With this integration, it's possible that the user experience become more immersive and close to reality. For the project development it was used Unity3D software that brings support for the requisites demanded for a good final result. At the end of this paper is possible to visualize a virtual welding model with good precision, being close to reality.*

***Resumo.*** *Este trabalho tem como foco principal a integração de um tracking magnético a um simulador de solda manual que junto a realidade virtual, podem trazer uma nova experiência com relação a atividades de aprendizado e prática de soldagem. Com isso, é possível que essas atividades exercidas pelo usuário tornem-se mais imersivas e próximas da realidade. Para o desenvolvimento do projeto utilizou-se a ferramenta Unity3D que traz um grande suporte aos requisitos exigidos para um resultado final satisfatório. Ao final deste trabalho é possível visualizar um modelo de solda virtual com boa precisão, sendo muito próximo ao modelo de solda real e convencional.*

## 1. Introdução

Pode-se afirmar que simuladores são máquinas que reproduzem o comportamento de um sistema sob determinadas condições, permitndo que o usuário deste sistema possa praticar certas atividades antes de iniciá-las na vida real. O computador atualmente é o melhor recurso disponível para a utilização destes simuladores incluindo o uso de diversos e sofisticados recursos gráficos, matemáticos e lógicos para alcançar um modelo de sistema que ofereça flexibilidade, agilidade e confiabilidade

Com este trabalho, foi possível integrar um modelo de simulador a um *tracking* magnético, focando assim, na imersão do usuário em sua atividade proposta e na maior proximidade possível do mundo real.

Para o desenvolvimento deste projeto, utilizou-se então a ferramenta denominada *Unity3D* que destaca-se por inúmeros projetos, principalmente em *3D* e realidade virtual, contando com boa imagem gráfica e com todos os recursos necessários disponíveis para a execução do trabalho.

O simulador de solda abordado aqui, utilizou realidade virtual e assim, junto ao *tracking* magnético para a pistola de solda, não foi necessário a utilização de nenhum objeto real. Desta forma, é possível transportar de forma imersiva e realista o usuário para qualquer lugar, tornando possível o treinamento em diferentes ambientes e possibilitando oferecer peças para a soldagem de diversos tamanhos e formas.

## 2. Realidade Virtual

Pode-se afirmar que realidade virtual trata-se uma simulação do mundo real através do computador. Esses ambientes virtuais podem proporcionar o desenvolvimento de inúmeras atividades com potencial para simulação e treinamento em áreas como segurança, trânsito, educação e até mesmo na medicina. Nesta

área ela traz até mesmo a possibilidade de realizar cirurgias a longa distância como afirma Campos (2010) quando menciona o uso desse avanço tecnológico em cirurgias minimamente invasivas onde uma pequena câmera de video e instrumentos especificos utilizados conduzem a cirurgia. A realidade virtual é um reflexo da realidade física e proporciona a sensação de que o usuário está imerso em três dimensões com sensação em tempo real e com capacidade de interagir com o meio ao seu redor. (RODRIGUES e PORTO, 2013)

A realidade virtual conta com inúmeros dispositivos porém, para este trabalho, foi utilizado o *Head-Mouted Display (HMD)* que é um dispositivo de interface composto por duas minúsculas telas de TV e um conjunto de lentes especiais, sendo um dos dispositivos mais populares e que melhor isola o usuário do mundo real (CAMPOS, 2010).

## 3. Tracking Magnético

O rastreamento dos movimentos de um usuário é um dos grandes suportes para garantir a real sensação de imersão no ambiente virtual e será através dele que os comandos do usuário serão interpretados e a resposta então, virá de form adequada. Para os autores Junior, Souza e Machado (2009), este dispositivo determinam a posição ou orientação e quanto maior a precisão, velocidade e área de ação do equipamento, maior será a quantidade de dados disponíveis a interpretação pelo sistema.

Para este projeto foi utilizado o modelo *MiniBird*-500 da Ascension Technology, conforme mostra a figura 1, que dentre suas muitas características positivas, estão as medidas de orientação e posição com boa precisão, gama irrestrita de movimentos, oferecimento de 6DOF (6 graus de liberdade) e utilização de mini sensores que permitem medições internas simultâneas de multiplos sensores e software de interface.


**Figura 1, Aparelho *MiniBird*-500**

É importante salientar que, uma vez que aplicações de realidade virtual exigem geração em tempo real de cenas em três dimensões, os sistemas de rastreamento devem rastrear todos os objetos desejados sem causar atraso na geração das cenas.

A partir do entendimento das necessidades deste projeto e de alguns testes práticos, foi iniciado a integração do *MiniBird* com a *Unity3D*, que será abordada mais a frente e então surgiu a primeira dificuldade deste projeto que será visto a seguir.

## 3.1 Conflito entre Módulos

A primeira dificuldade encontrada foi as versões dos *softwares*. O projeto do simulador de solda foi desenvolvido na versão da *Unity3D* de 64 *bits*. Contudo, o *driver* do *bird* só tinha disponível a versão de 32 *bits*. E, no sistema operacional *Windows*, um processo de 64 *bits* não poderia, diretamente, carregar uma biblioteca de 32 *bits*.

Uma das principais vantagens da tecnologia de 64 *bits* é a sua capacidade de trabalhar com até 8Tb de memória, contra o máximo de 2Gb para processos de 32 *bits*. Segundo Becker (2007), a tecnologia de 64 *bits* permite que a maioria dos dados de processamento possam ser armazenados na memória, sem

qualquer necessidade de armazenamento temporário em disco e isso poderia aumentar consideravelmente o desempenho e abrir novos cenários de processamento de dados. Há, portanto, boas razões para migrar atuais produtos de *software* de 32 *bits* para uma plataforma de 64 *bits*. Muitas aplicações em *C* ou *C++* podem ser migradas com facilidade para a plataforma de 64 *bits*, no entanto alguns *softwares* baseados em módulo podem causar mais problemas. (BECKER, 2007).

A única forma de realizar a comunicação entre os módulos de 32 *bits* e 64 *bits* seria através da comunicação entre processos (*IPC*). Segundo Becker, 2007, um produto de *software* de 32 *bits* contém o módulo de vigilância principal que chama a *DLL WeatherStationControl*. Tanto o módulo principal e da *DLL* são 32 *bits* e o processo então poderá ser executado em ambas as plataformas de 32 *bits* e de 64 *bits*, dentro do *WOW64*.

Concluindo, se o módulo principal e *DLL* forem migrados para a plataforma de 64 *bits*, eles poderão então, ser executados em 64 *bits*. No entanto, se apenas o módulo principal for migrado para 64 *bits*, ele não será capaz de carregar a *DLL* de 32 *bits*. (BECKER, 2007).

E como foi necessário que o *MiniBird*, com uma biblioteca compilada em 32 *bits*, interagisse com a *Unity3D* versão 64 *bits*, foi preciso criar um novo processo, de 32 *bits*, para abrir um canal de comunicação entre os dois processos.

## 4. Oculus Rift

O *Oculus Rift*, visto na figura 2, tem um grande potencial, que vai além do entretenimento e já é exlorado por diversos setores como a educação, saúde, mercado imbiliário entre outros. Ele possui integração com as *engines Unreal Engine* e *Unity3D* que foi utilizada neste projeto e será abordada mais a frente.

Este dispositivo foi utilizado para o projeto sendo que o utilizado, é de uma versão lançada em 2014 e que utiliza um *display* de *OLED* de baixa persistencia e com dobro de qualidade, contando com 960x1080 *pixels* por tela e um acessório extra para aprimorar o rastreamento de posições.


**Figura 2, Oculus Rift, DK2(Development Kit 2)**

## 5. Unity3D

Para o desenvolvimento deste trabalho a *Unity3D*, apresentada na figura 3, foi escolhida por ser uma ferramenta de fácil utilização e que destaca-se em inúmeros projetos com utilização *3D* e até mesmo realidade virtual, além de ser compatível com o *Oculus Rift*, visto anteriormente. Foi utilizada então a versão 5.3.3 deste motor gráfico que possibilitou oferecer um sistema de *script* compreensivo e flexivel possibilitando a prototipação e desenvolvimeno rápido e robusto de todas as partes do projeto.

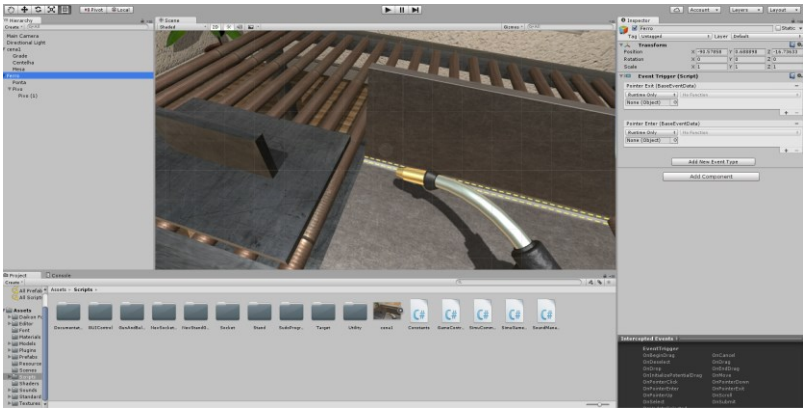


**Figura 3, Projeto sendo Desenvolvido na *Unity3D***

É possível citar alguns trabalhos relacionados ao desenvolvimento de aplicações baseadas em técnicas de realidade virtual que utilizaram a *Unity3D* para o desenvolvimento de sistemas voltados a treinamento e simulação no contexto de *Serious Games*.

- *Designing a Game for Occupational Health and Safety in the Construction Industry:* A pesquisa enfatiza a utilização de um ambiente virtual para treinamento de trabalhadores da construção civil, tendo como intuito a redução de lesões e riscos de acidentes de trabalho. O sistema destina-se a ajudar o usuário a aprender sobre os riscos no local de trabalho e manter seu conhecimento sobre os procedimentos de segurança auxiliando no processo real. (SILVA, Alexandre Carvalho et al., 2013)

- *An Easy to Author Dialogue Management System for Serious Game:* O trabalho descreve uma solução para ambientes virtuais que possuam atividades de gestão de diálogo em personagens virtuais (*avatar*) empregados em *Serious Games*, tendo como objetivo melhorar a experiência de aprendizagem, aumentando a sensação

de imersão e envolvimento do usuário. Para validar a ferramenta, os autores desenvolveram um ambiente simples onde o usuário pode interagir com representação virtual de um artista a fim de adquirir conhecimento sobre sua vida e suas obras de arte. (SILVA, Alexandre Carvalho et al., 2013)

- *Emergency Medicine Training with Gesture Driven Interactive 3D Simulations:* O trabalho proposto apresenta um sistema protótipo utilizado para simulação e treinamento de equipe médica em medicina de emergência. A pesquisa relata que o uso de simulações imersivas na formação médica é de extrema utilidade para enfrentar cenários de emergência que vão desde o habitual ao extremo, sem colocar os participantes da simulação em risco. O protótipo conta com cenários *3D* interativos e uma *interface* natural baseada em gestos. (SILVA, Alexandre Carvalho et al., 2013).

Ainda para Silva, Alexandre Carvalho et al. (2013) a *engine Unity* correspondeu com eficiência e eficácia nos requisitos envolvidos na construção de sistemas de realidade virtual voltados ao controle e treinamento, se mostrando razoavelmente flexível às adequações e possibilitando a criação de sistemas de *interface* gráfica *3D* em diferentes plataformas de desenvolvimento.

## 6. Simulador

O simulador permite que o usuario emita comandos usando todo seu corpo e em outras palavras, o artefato utilizado reproduz, como por exemplo em um simulador de motocicleta, os pedais, guidão, alavancas, botões nos painéis, etc. (REIS, 2016), como mostra o exemplo na figura 4.

**Figura 4, Simulador de Trânsito - Reis (2016).**

A simulação permite a análise e o teste de diferentes alternativas de funcionamento de um sistema, com variações de diversos parâmetros, formas de controle, prioridade de eventos, diversos tipos de equipamentos e velocidades, ciclos de trabalho e *layouts* ( BALADEZ, 2009).

Com o uso dessa tecnologia, é possivel treinar até que se esteja apto a realizar a atividade no mundo real, adquirindo experiência e destrezas necessárias para desempenhar a função sem correr riscos à sua saúde e de terceiros.

Pode-se afirmar que o uso da simulação reduz riscos de novas implementações e dá suporte a decisões complexas que por vezes envolve investimentos elevados. Devido aos avanços em relação a *hardware* e *software*, a utilização de recursos de realidade virtual proporciona as empresas, maior desempenho e menores custos (RODRIGUES e PORTO, 2013).

Quanto a sua aplicação, os simuladores são aplicados em diversas áreas como educação, medicina e segurança. Com relação a este trabalho, no início do ano de 2016, pode-se constatar que o simulador de solda apresentava uma deficiencia na imersão da realidade virtual e a proposta então foi de tentar

suprir essa deficiência e talvez até mesmo de outros simuladores que utilizem a ferramenta de precisão em conjunto com a realidade virtual.

## 7. Histórico

Ao iniciar este projeto em março de 2016, o desenvolvimento do Simulador, utilizado aqui, já estava em andamento desde janeiro de 2016. Era possível controlar, com o *mouse* e teclado, a pistola de solda e fazer pequenos caminhos, ainda experimentais, de soldagem. Já sendo possível utilizar um óculos de realidade virtual, a imersão era significativa. Contudo, utilizar teclado e *mouse* estava longe do ideal. Se fazia necessário uma maneira de controlar a pistola de solda virtual o mais próximo da realidade possível. Tornara-se necessário um aparelho capaz de rastrear os movimentos da pistola de solda com precisão e acurácia. Na figura 5, é possivel visualizar o estado inicial do desenvolvimento.


**Figura 5, Estado Inicial do Simulador**

### 7.1. Controle e Adaptação

Neste estágio inicial o controle do simulador era bastante rudimentar. Movendo o *mouse* sobre a mesa de solda, com o

botão esquerdo do *mouse* pressionado, controlava-se a posição da tocha de soldagem na mesa. A roda do meio do *mouse* controlava a altura da pistola de soldagem e, por fim, segurando *shift* no teclado controlava-se a inclinação da tocha de soldagem. A soldagem começava com a aproximação da ponteira da tocha de soldagem na mesa. Com isso o simulador conseguia capturar diversas características como inclinação, velocidade e altura necessárias para analisar se a soldagem estava adequada ou não. Contudo, por ser controlado por *mouse* e teclado não refletia as habilidades do usuário. Para um melhor acompanhamento e treinamento, tornou-se imprescindível o *tracker.*

Com o *tracker* integrado, o esperado era que o simulador capturasse, bem próximo à realidade, essas caraterísticas do usuário e indicasse se este fez uma boa soldagem ou não, bem como seu progresso de treino. O *tracker*, no simulador, iria controlar a tocha de soldagem, substituindo o controle rudimentar do teclado e *mouse.*

Para o bom andamento de ambos os projeto, tanto o simulador, quanto a integração do *tracking* magnético com a *Unity3D,* decidiu-se por trabalhar separadamente até o momento em que ambos estivessem maduros o suficiente para fazer a integração - Isso ocorreu, com sucesso, em agosto de 2016.

Ficou combinado que o *tracking* iria, na integração, controlar a translação e rotação de um *GameObject* "pai" da tocha de soldagem. O "pai" seria o *pivô* da tocha de soldagem. Ou seja, todo o ferro acompanharia os movimentos ditados pelo "pai", como se este fosse a origem dos "filhos". Se o pai fosse movido, os filhos também seriam, se fosse rotacionado os filhos também seriam rotacionados. A figura 6 representa a utilização do pivô deste trabalho.

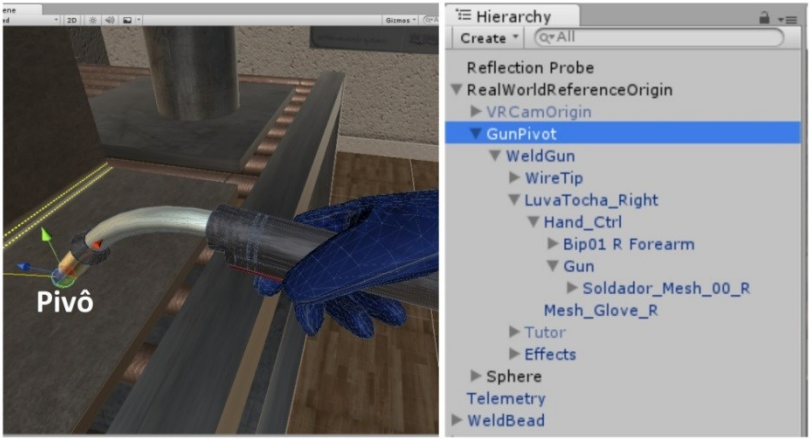



**Figura 6, Ilustração do Pivô**

## 7.2. Documentação e Biblioteca do *MiniBird*

Depois das definições iniciais foi dado, em meados de março, início à um novo projeto, com a intenção de solucionar a integração do *tracking* magnético com a *Unity3D* e, posteriormente, integrar ao simulador de solda. O primeiro passo foi obter todas as documentações e códigos disponíveis para o *tracker MiniBird 500* da *Ascension Technology*. Reuniu-se então, alguns tutoriais oficiais para fazer *upgrade* de *driver*, além de documentos sobre o uso de alguns *softwares* de teste e demonstrativos. A documentação mais importante reunida, no entanto, foi o guia de instalação e uso do aparelho e a documentação da *API* do *driver*.

A primeira informação à chamar a atenção em toda a documentação foi a idade avançada do aparelho, que datava de 2001. Com isto, foi necessário checar a compatibilidade com os novos sistemas e máquinas. Rodou-se um programa teste na máquina de Processador *Intel(R) Core(™) i7-4770*, *CPU* de 3.40 GHz, memória de 8 GB com sistema de 64 *bits* e placa de vídeo *GeForce GTX* 750 de 1024 MB com o sistema

operacional *Windows 8.1 Pro* de *64 bits* , e o *software* rodou sem problema algum.

Esse aparelho vem com um arquivo *DLL* para controlar e obter informações do *tracker*. Foi utilizado esse arquivo e sua documentação para desenvolver o *framework* que possibilitaria a integração do *tracker* com o ambiente *Unity3D*.

### 7.3. Inicialização do *MiniBird*

Da documentação deste aparelho específico *MiniBird - 500 da Ascension Technology Corporation* foi identificado diversos modos de iniciar, configurar, e obter os dados do aparelho. Tendo a aplicação do simulador de solda em mente, foi selecionado um conjunto básico de funções que atendiam às necessidades sendo que o primeiro passo a fazer foi "acordar" o aparelho.

Depois de "acordar" o *tracker*, é preciso obter as informações básicas do sistema do aparelho. Estas informações estão estruturadas numa *struct* do tipo *BIRDSYSTEMCONFIG*. Essa informação deve ser obtida através da função *birdGetSystemConfig*. Caso alguma configuração especial seja necessária, utiliza-se *birdSetSystemConfig* com *BIRDSYSTEMCONFIG* modificado como parâmetro. Neste trabalho não foi preciso nenhuma configuração especial.

Para finalizar a inicialização do *bird* foi preciso pegar as informações do aparelho. As informações são compiladas na *struct BIRDDEVICECONFIG*. Então, depois desses passos tem-se o dispositivo pronto para comunicação. O aparelho é acordado, adquiriu-se as informações do sistema de *trackers*, as informações do *tracker* específico e foram feitas as customizações necessárias.

### 7.4. Leitura do *MiniBird*

Com o aparelho preparado, utilizou-se então a função *birdStartFrameStream* para iniciar o processo de obter

informações de posição e rotação regularmente do aparelho. Quando não houvesse mais necessidade do fluxo constante de informação do dispositivo utilizava-se sua contraparte *birdStopFrameStream.*

A comunicação foi estabelecida e o aparelho pronto para enviar informações. A informação seria compilada na *struct BIRDREADING*. Onde teria disponível para leitura a posição e o ângulo do *tracker* (o formato e as informações disponíveis foram ajustados na etapa de configuração do aparelho).

Para pegar as informações mais recentes do *tracker*, foi pego o último *frame* disponível sendo que um *frame* é um momento, um pedaço, no tempo de vida do rastreamento. E esse momento descreve as condições no sistema. Foi utilizado a função *birdGetMostRecentFrame*, onde obtém-se o *frame* mais recente em uma *struct BIRDFRAME.*

Dentro do frame há um *array* de *struct BIRDREADING*, com as informações pertinentes. Como foi utilizado somente um aparelho, o objeto de interesse ficou na posição '0' deste *array*. O *driver* oferece ainda um método para verificar se há um novo *frame* disponível para pegar, *birdFrameReady*.

Por último, quando não precisasse mais das informações do *tracker*, era necessário finalizar sua operação utilizando o método *birdShutDown*.

### 7.5. Middleware

Foi iniciado, para suportar o *bird* utilizando esta técnica, um novo projeto, chamado aqui de *Middleware*, por ser o meio de comunicação entre a aplicação principal e o *driver* do aparelho. Esse projeto seria um *software* que rodaria em forma de *console*, ou seja, sem interface gráfica. Este *software* seria iniciado pelo simulador de solda através da *Unity3D*. Portanto

não seria necessário nenhuma interface gráfica para comunicar com o usuário.

Esta aplicação iria iniciar, configurar e obter as informações do aparelho. Ou seja, carregaria a biblioteca do *bird*, executaria todos os passos necessários para inicializar o aparelho e requisitaria os *frames* para obter as informações e poder repassar para o processo interessado.

O *Middleware*, deveria fazer a inicialização do *bird* assim que o processo fosse instanciado. Este seria o servidor *pipe*, servindo os dados do *tracker* para a aplicação através de um 'fluxo de *string'*. E, quando o processo fosse encerrado, deveria fazer a finalização do *bird* também.

Assim, a funcionalidade do *Middleware* resumiu-se a gerenciar o *bird*, iniciar o servidor *pipe* e iniciar o fluxo de informação do *tracker* ao cliente *pipe*: o simulador de solda. O *Middleware* ainda ficaria responsável por dar o tratamento final e formatar os dados obtidos do *tracker*. Seria então, convertido a posição para centímetros e o ângulo para quaternion.

Seria responsabilidade do Simulador de Solda inicializar o *Middleware*, conectar-se, interpretar as informações e encerrar o *Middleware* quando for o momento.

### 7.6. Integração com Simulador

Foi importado os *scripts* da integração dentro do projeto do Simualdor. O *script Tracker* foi anexado ao pivô do objeto da pistola de solda, conforme planejado. O resultado da integração do *tracker* com a *Unity3D* foi bem sucedido. Os movimentos e rotação correspondiam ao esperado. Quando a pistola real era movida ou rotacionada, a pistola virtual correspondia à interação, movendo-se e rotacionando de acordo. Isso porque a escala do simulador foi considerada em centímetros, assim como as informações do *tracker*. O simulador também estava mais robusto e melhor preparado. Gráficos e medições haviam

sido aprimorados. A figura 7 mostra o simulador aperfeiçoado em uso.

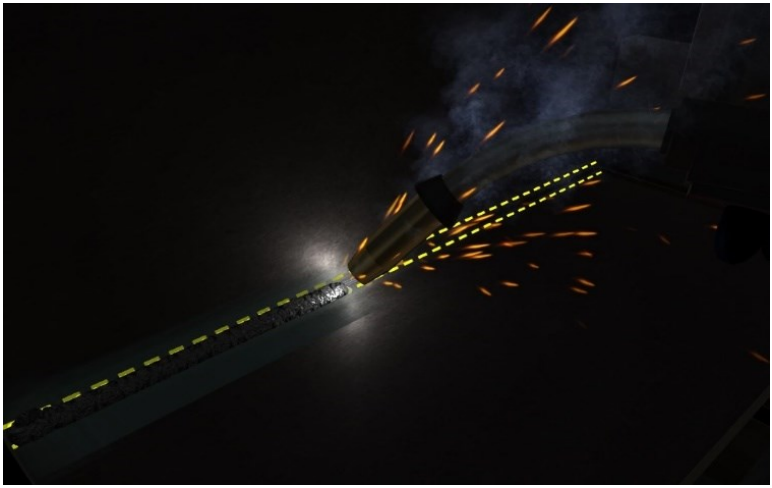


**Figura 7, Simulador Aperfeiçoado em Uso**

O tracker foi incorporado à uma pistola de soldagem, como mostra a figura 8 e o pivô, que anteriormente correspondia à ponta da pistola de solda virtual no simulador, foi movido de forma a corresponder aproximadamente ao local do posicionamento do tracker dentro da pistola real.


**Figura 8 – Tracker Incorporado a Pistola de Solda**

A integração do simulador de solda, do O*culus Rift* e do *tracking* magnético ocorreu sem erros. Foi possível utilizar os equipamentos com o simulador sem nenhum problema

inesperado. Contudo, foi preciso assegurar que o *Oculus Rift* estivesse na posição e orientação inicial desejada (na direção da mesa de solda virtual) para assegurar que o usuário pudesse observar a tocha de soldagem. Algumas vezes, dependendo da inicialização do simulador, a orientação do usuário com relação ao cenário ficava incorreta. Para corrigir este problema e sempre que necessário corrigir esta rotação, foi criado um simples script que utiliza uma função de *reset* da própria *Unity3D*, que reseta a rotação de capacetes virtuais, chamado *Recenter.* O *tracker* então, envia a posição e rotação em valores absolutos a partir de sua origem. A origem é o centro do aparelho que gera o campo magnético.

Quando o *tracker* informa que o valor da posição *X* é 10 cm. O correspondente na *Unity3D* é 10 unidades (consideradas centímetros nesse projeto) a partir da origem da cena. E esse posicionamento, na cena do simulador, não correspondia à proximidade da mesa de solda. Por esse motivo, foi necessário criar um nível de hierarquia para o pivô utilizado. Esse novo nível, chamado de *offset* representa a origem do aparelho, e portanto a origem da pistola de solda virtual. O *offset* é posicionado em um local na mesa de solda virtual, de forma que os movimentos feitos fiquem na proximidade da mesa, no objeto de interesse.

À essa altura a noção de imersão almejada, por observação, estava bastante satisfatória. Já era possível utilizar a tocha de soldagem na mesa virtual e treinar as técnicas de soldagem.

Entretanto, a sensação tátil ainda quebrava um pouco a imersão total possível. A mesa real e a mesa virtual não correspondiam, nem a posição da solda real e virtual com relação à posição da mesa. Para que essas correspondências fossem alcançadas, seria necessário uma calibração de equiparação, o que foge do escopo deste trabalho.

Não foi observado erro significativo quanto a precisão e acurácia no deslocamento ou na rotação da pistola de solda.

Ao final desta integração, era possível utilizar o simulador com realidade virtual e pistola de solda com *tracking* magnético com boa precisão e imersão.

## 8. Testes

Os testes foram todos internos e baseados em observação e ocorreram periodicamente durante todo projeto. Os testes de validação, contudo, com a liderança do projeto, foi feito no final, quando foi realizado a integração com o simulador de solda.

## 9. Conclusão

Ao fim deste trabalho pode-se concluir que a integração entre um *tracking* magnético e um simulador de solda com realidade virtual é possível e traz benefícios como a imersão, maior liberdade de controle e maior proximidade com a realidade.

É importante salientar que a realidade virtual vem se destacando como um importante aliado em diversas áreas que junto a simuladores, podem oferecer novas experiências principalmente na área da saúde, educação e segurança. Unindo essas duas tecnologias a periféricos como o *tracking* magnético, pode-se envolver o usuário em uma nova experiência imersiva e realista, oferecendo benefícios significativos e resultados positivos para o aprendizado.

O simulador utilizado neste trabalho foi desenvolvido em paralelo e de forma independente. No momento da integração o simulador estava bastante maduro, apresentando simulação básica de soldagem e conseguindo avaliar de forma consistente a soldagem, informando erros e acertos sobre algumas variáveis de soldagem, como velocidade e ângulo de avanço.

Contudo, o simulador ainda necessitava de diversos ajustes e funcionalidades para oferecer uma simulação mais consistente e coerente com a realdiade, e um aprendizado que fosse satisfatório as necessidades do mercado.

Como *software* educacional o simulador precisaria oferecer exercícios e uma forma de avaliação. Exercícios que contemplassem do básico da soldagem à exercícios mais avançados utilizando todo o potencial da realidade virtual, apresentando formas geométricas complexas ou ambientes inusitados. Além disso seria interessante um tutor virtual. Uma maneira de acompanhar o usuário, mostrar seus acertos e erros, bem como meios de aperfeiçoar as técnicas para melhorar continuamente.

Para aumentar a imersão e aprendizado, seria interessante incorporar ao simulador um *hardware* capaz de simular o equipamento real de soldagem. Podendo, por exemplo, ajustar no equipamento a voltagem ideal para o exercício e essa escolha impactar no simulador.

Como pode-se observar, além dos pontos levantados, ainda há muitos aspectos à melhorar no simulador. A integração com o *tracking* magnético foi um passo importante para a melhoria e viabilidade do simulador. Entretanto ainda há muito o que fazer para que seja uma ferramenta segura de aprendizado eaperfeiçoamento de técnicas de soldagem.

## References

CAMPOS, Itamar Albertino de. Realidade Virtual e Aumentada - Conceitos, Tecnologia e Aplicações. Cuiabá. 2010.

RODRIGUES, Gessica Palhares; PORTO, Cristiane de Magalhães. Realidade Virtual: Conceito, Evolução, Dispositivos, Aplicações. Aracaju. 2013.

JUNIOR, Antonio Deusany de Carvalho; SOUZA, Daniel Faustino Lacerda de; MACHADO, Liliane dos Santos. Integração de Rastreadores Magnéticos a um Framework para Desenvolvimento de Sistemas de Realidade Virtual. Paraíba. 2009.

BECKER, Mike. Accessing 32-bit DLLs from 64-bit code. Alemanha. 2007.

SILVA, Alexandre Carvalho et al. Uso da Engine de Jogos Unity3D para Sistemas de Realidade Virtual Aplicado a Monitoramento e Controle de Subestações de Energia Elétrica. Minas Gerais. 2013.

REIS, Alessandro Vieira dos. Interfaces Tangíveis em Simuladores Veiculares: Componentes para um Protocolo de Avaliação de Usabilidade. UFSC. 2016.

BALADEZ, Fabio. O Passado, o presente e o futuro dos simuladores. São Caetano do Sul. 2009.