U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Desenvolvimento de processos de interação entre tecnologia BIM e equipamentos de Realidade Virtual e sua aplicabilidade

FÁBIO ALEXANDRE MATOSEIRO DINIS

Dissertação submetida para satisfação parcial dos requisitos do grau de

MESTRE EM ENGENHARIA CIVIL — ESPECIALIZAÇÃO EM CONSTRUÇÕES

Orientador: Professor Doutor João Pedro da Silva Poças Martins

JULHO DE 2016

MESTRADO INTEGRADO EM ENGENHARIA CIVIL 2015/2016

DEPARTAMENTO DE ENGENHARIA CIVIL

Tel. +351-22-508 1901

Fax +351-22-508 1446

✉ miec@fe.up.pt

Editado por

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Rua Dr. Roberto Frias

4200-465 PORTO

Portugal

Tel. +351-22-508 1400

Fax +351-22-508 1440

✉ feup@fe.up.pt

🌍 http://www.fe.up.pt

À MEMÓRIA DO MEU AVÔ PATERNO

AGRADECIMENTOS

A concretização deste trabalho, muito além do esforço individual, contou com o apoio de algumas pessoas, sem as quais seria impossível atingir as metas delineadas. É com sincera gratidão que lhes dedico este espaço, na esperança de poder refletir com a devida justiça o reconhecimento que lhes devo.

Ao meu orientador, João Pedro da Silva Poças Martins, por todo o incentivo e confiança que me transmitiu desde o primeiro momento em que lhe apresentei a ideia base deste trabalho. Pela disponibilidade irrepreensível, compreensão e sensibilidade demonstradas ao abordar uma temática de dissertação inovadora, estando sempre ciente das dificuldades e do risco em torno do tema que lhe propus. O meu franco agradecimento pela oportunidade.

Ao Tiago Tavares, amigo, companheiro de trabalho, muitas vezes professor e futuro engenheiro informático desta mesma casa. Pela prova de amizade, pelas noites intermináveis de lições de programação, pelo carácter que demonstrou ao acompanhar esta viagem pelo conhecimento. Reconheço sem embaraço que muito do mérito deste trabalho se deve à paciência, disponibilidade e esforço que em conjunto investimos. Mais do que um colega engenheiro, um verdadeiro amigo.

Ao Laboratório de Instrumentação e Medição da FEUP e em especial à investigadora Teresa Restivo pelo equipamento que me confiou, fundamental para o desenvolvimento deste projeto.

Ao Leonardo Morais pelos incontáveis testes de Realidade Virtual que o fiz avaliar. Obrigado por todo o entusiasmo e confiança transmitidos.

A todos os meus amigos que durante esta jornada me confortaram, encorajaram e ouviram, em especial ao Miguel Campos, Diana Cardoso e Sara Cardoso.

Por último, aos meus pais e avó materna por confiarem nos meus ideais, alertando-me sempre para as dificuldades e disciplina necessárias para atingir os meus sonhos. Pelos valores transmitidos desde muito cedo. Pela oportunidade de me tornar um engenheiro. Não existe agradecimento suficientemente justo que possa ser descrito.

RESUMO

A metodologia *Building Information Modeling* (BIM) representa uma evolução face ao método estabelecido na indústria AEC para coordenar e desenvolver todos os processos relativos à construção. Todavia, a implementação do BIM não é adotada transversalmente por todos os intervenientes no ramo da Arquitetura e Engenharia Civil. Ainda existem barreiras, seja por razões economicistas, resistência à mudança, etc., que dificultam a implementação desta nova metodologia de trabalho.

A Realidade Virtual (RV) poderá ser uma ferramenta relevante no que à interação diz respeito. Desta forma, através da potencialidade de imersão total num projeto e recorrendo de uma interface de uso natural, é possível, em primeira análise, colmatar a complexidade ou falta de experiência com BIM. Ou seja, a curva de aprendizagem poderia ser otimizada.

Durante a dissertação desenvolveram-se aplicações informáticas que permitem a um utilizador tirar partido das vantagens da Realidade Virtual como meio de comunicação, interação e simultaneamente exercer modificações no modelo BIM.

Primeiramente recorreu-se a um motor de jogo para importação de um modelo BIM. Neste programa iniciou-se a prova de conceito, onde alguns comandos do Unity estabelecem relação com operações executadas através de equipamento de RV. Neste sentido, foram selecionados e programados alguns comandos de interação, possibilitando o uso de RV para alteração em tempo real da geometria de alguns elementos de um projeto BIM.

Posteriormente desenvolveu.se um caso de estudo, onde através de uma experiência prática e preenchimento de um inquérito, se avaliou o grau de naturalidade de execução de cada um dos comandos produzidos. Através desta análise qualitativa pretendeu-se, embora de uma forma pouco representativa, avaliar a interface desenvolvida de um ponto de vista distinto de quem programa ou produz a plataforma.

PALAVRAS CHAVE: interação, Realidade Virtual, BIM, motor de jogo, tecnologia da informação.

ABSTRACT

The Building Information Modeling (BIM) methodology represents an upgrade of the established method to manage all the construction processes related to the AEC industry. However, this new methodology is far from being accepted by all the participants in the Architecture or Civil Engineering fields. There are still some issues, either for economic reasons, resistance to change, etc., that make it difficult for the implementation of this new way of doing things.

Virtual Reality (VR) could be a relevant tool, concerning its interaction capabilities. Therefore, the capacity of full immersion in a project, plus a natural user interface, could help to overcome the initial complexity or lack of experience with BIM. That is, the learning curve would be improved.

During this dissertation, computational applications were developed that enable the user to take part of the capabilities of VR as a mean of communication, interaction and as a way of making changes in the BIM model.

Initially a game engine was used to import a BIM project: in this program, the concept was initialized by linking some commands of Unity to VR equipment operations. Therefore, some functions were programmed to allow the use of VR in real time in order to handle the geometry of some BIM elements.

Then a case study was made, being it evaluated through a practical experience and filling out of a survey related to the degree of "naturalness" of execution of each of the produced commands. Although unrepresentative, the analysis of this case study allowed the analysis of the interface from a distinct point of view compared with whom produced the platform.

KEYWORDS: interaction, Virtual Reality, BIM, game engine, information technology

ÍNDICE GERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

SÍMBOLOS E ABREVIATURAS

AEC – Arquitetura, Engenharia e Construção

API - Application Programming Interface

BIM – Building Information Modeling

CIR – Centro Instantâneo de Rotação

FBX – Filmbox

FEUP – Faculdade de Engenharia da Universidade do Porto

GDC - Game Developers Conference

HMD – Head-Mounted Display HTML

MEP – Mechanical, Electrical and Plumbing

MWC - Mobile World Congress

OBJ – Object Files

RA – Realidade Aumentada

RV – Realidade Virtual

VCUHS - Virginia Commonwealth University Health System

1
INTRODUÇÃO

1.1. EQUADRAMENTO GERAL

Nesta dissertação propõe-se a avaliação do uso da realidade virtual (RV) como complemento à interação com *software* comercial BIM (*Building Information Modeling*). A RV, conceito aprodundado no Capítulo 2 (maior detalhe em 2.2), poderá atuar como ferramenta facilitadora da interação entre utilizador e máquina. Deste modo, será avaliada a sua viabilidade em operações como visualização, edição de modelos 3D, diminuição do tempo de familiarização do utilizador com a mecânica do trabalho, o potencial do seu uso como ferramenta facilitadora do diálogo entre os diferentes intervenientes num projeto, entre outros.

Pretende-se ainda analisar a potencialidade de RV associada a uma interface de uso natural (NUI - *natural user interface*). Assim, o processo pelo qual a ideia surge na consciência do utilizador e consequentemente se vai materializando numa série de operações e comandos até à solução modulada, poderá ser otimizado.

Neste documento são considerados aspetos que limitam a interação com a "máquina informática" e as alternativas que a RV poderá oferecer para que estes não se interponham no processo criativo e racional.

1.2. ÂMBITOS E OBJETIVOS

1.2.1. OS PRINCIPAIS ENTRAVES NA GENERALIZAÇÃO E APLICAÇÃO DA METODOLOGIA BIM

Os programas comerciais associados à tecnologia BIM, apesar da sua constante atualização, são por vezes rivalizados por outro tipo de soluções informáticas tecnologicamente menos evoluídas e com mais anos de mercado. Tal fato poderá justificar-se pela maior simplicidade e rapidez na execução de determinadas tarefas por estes programas, tornando-os mais apelativos para as empresas que ainda não recorrem ao BIM.

As soluções BIM estão muitas vezes associadas a maior detalhe e informação, levando a que mais tempo e experiência sejam necessários para o desenvolvimento das várias fases do projeto. Neste ponto as soluções como o CAD podem ser preferíveis para utilizadores que ainda não fizeram a transição para o BIM. Sendo que, muitas das vezes implicam menor detalhe ou mais tempo gasto na realização de tarefas que, de outra forma, seriam automáticas no caso de um modelo BIM. [1]

O *know-how* e o processo de familiarização com os programas informáticos comerciais dedicados ao BIM é um dos principais obstáculos no percurso pela sua implementação.

Não menos relevante, o desempenho das máquinas, responsáveis pelo processamento da informação, poderão determinar a morosidade de execução de tarefas, dependendo da dimensão do projeto.

A atualização das empresas ao nível do pessoal e do equipamento informático acarreta custos muitas vezes incomportáveis com a capacidade financeira das mesmas. Neste sentido, a falta de capacidade monetária para comportar este ponto gera uma barreira para a generalização do BIM entre os intervenientes na indústria da construção. [2]

> *Um dos responsáveis pela intervenção referiu numa reunião: "O software disponível no mercado para gerar modelos BIM tem custos muito elevados, será ainda necessário melhorar o hardware de muitas máquinas para correr o programa. É ainda preciso apostar na formação do pessoal, e todos estes custos são uma barreira à implementação desta tecnologia. Temos de avaliar sempre em função da dimensão e complexidade dos projetos, assim como dos orçamentos disponíveis."* [2]

1.2.2. REALIDADE VIRTUAL E BIM – OBJETIVOS E MUDANÇAS NO PARADIGMA

Neste trabalho é proposta uma solução baseada em RV para dar resposta aos seguintes desafios:

1. Falta de experiência e incentivo por parte do pessoal ativo nas empresas relativamente à transição para a implementação da metodologia BIM e sua operabilidade;
2. Diminuição do tempo de adaptação e familiarização com uma nova ferramenta informática;
3. Acessibilidade da tecnologia a novos públicos, desde utilizadores de nível profissional com vasta experiência em modelação, a estudantes, docentes, utilizadores de "um BIM mais teórico", a trabalhadores da indústria da construção sem qualquer contato anterior com este tipo de tecnologia, etc.;
4. Restrições associadas à visualização limitada pelo confinamento de ecrãs 2D;

1.2.3. POTENCAIS INTERESSADOS NA UTILIZAÇÃO DE RV NA CONSTRUÇÃO

A generalização do uso de aplicações e equipamento que recorre a RV ilustra a aposta progressiva neste tipo de tecnologias. O interesse por parte das empresas e instituições no desenvolvimento de ferramentas capazes de proporcionar experiências em ambiente virtual reflete o papel que esta tecnologia poderá exercer nos modos de interação, nos mecanismos e lógicas de trabalho. Enquanto ferramenta com potencial de aplicação em distintas áreas do conhecimento e da indústria, e dada a relativa facilidade de manipulação dos seus equipamentos, o uso de RV poderá expandir-se por todo o espectro da hierarquia do trabalho.

O recurso a uma *Natural User Interface* (NUI) promove uma redução no tempo de adaptação a uma nova ferramenta, neste caso a *hardware* de RV. Com efeito, os utilizadores de equipamentos com este tipo de interface adquirem em pouco tempo destreza e controlo que só seriam comuns caso possuíssem experiência prévia de trabalho com o sistema. Tal como será descrito em detalhe no Capítulo 4, os utilizadores de perfis distintos, com ou sem experiência na metodologia BIM, poderão encontrar vantagens ao interagir através de uma interface de operabilidade natural.

Importa fazer a ressalva que a componente de modelação é apenas uma vertente, mais prática, da metodologia BIM. Aspetos relacionados com o controlo e acompanhamento do processo construtivo, gestão de instalações, sustentabilidade na construção (vertente energética), fazem parte do vasto espetro que o BIM consegue abranger.

A associação de equipamentos de RV com diferentes tipos de sensores ou outros dispositivos tecnológicos, como o caso dos *drones*, entre outros, poderá expandir o campo das aplicações, assim como a gama de interessados em adquirir novas metodologias de trabalho.

1.3. ESTRUTURA DA DISSERTAÇÃO

O presente trabalho é composto por cinco capítulos, realizando-se a proposta do tema de dissertação, enquadramento e o seu âmbito de desenvolvimento neste primeiro capítulo. É feita, de igual modo, a introdução e definidos os objetivos da dissertação.

No segundo capítulo é descrito com pormenor o contexto de aplicação da RV. A aplicabilidade deste tipo de tecnologia é retratada e contextualizada em diferentes campos. Apresenta-se uma breve evolução cronológica da RV, assim como os marcos e referências de maior relevo para o desenvolvimento desta tecnologia. Na fase final do Capítulo 2 são descritos variados tipos de equipamentos de RV como conceitos relacionados com desenvolvimento de *software: scripting*, motores de jogo, entre outros...

A apresentação das tecnologias disponíveis para a exploração da interface que se pretende criar neste trabalho e as descrições essencialmente técnicas inerentes ao funcionamento dos comandos criados estarão contempladas no Capítulo 3.

Foi produzido um caso de estudo formado por uma componente prática de exploração de um modelo BIM e um inquérito para avaliação do grau de adequabilidade das funcionalidades produzidas para RV. A estrutura deste caso de estudo e a análise de resultados obtidos é feita no capítulo quarto.

Por último, no capítulo, apresenta-se uma conclusão englobando todo o trabalho produzido. Tecem-se algumas expectativas sobre o futuro da RV e possíveis aplicações de um modelo mais complexo baseado na prova de conceito explorada neste trabalho.

2

UM MÉTODO ALTERNATIVO PARA MANIPULAÇÃO DE MODELOS BIM

2.1. POSSÍVEIS APLICAÇÕES DA REALIDADE VIRTUAL NA ENGENHARIA CIVIL, ARQUITETURA E INDÚSTRIA DA CONSTRUÇÃO

A RV na indústria da construção poderá abranger as aplicações em projeto, como auxílio no trabalho colaborativo, fornecendo alternativas vantajosas para a visualização, contribuindo para a melhoria e agilização dos processos. [3]

Como tecnologia catalisadora de novos métodos de trabalho, a RV não se encontra circunscrita a um número limitado de aplicações. A oferta de sensores, *software* e capacidade dos equipamentos atuais é suficiente para afirmar que a RV se poderia aplicar "a todo o tipo de objetivos". Quer isto dizer que a RV não se encontra prescrita para determinado uso numa única área do conhecimento. De modo evidente, a razoabilidade dessa mesma aplicação terá que ser sempre avaliada, no entanto recai sobre a imaginação o potencial que a RV poderá exercer sobre novas metodologias e renovação de processos de trabalho.

2.1.1. FERRAMENTA DE COMUNICAÇÃO

Na comunicação de ideias e melhoria da perceção espacial, a RV poderá assumir um papel determinante, atuando como mediador na transmissão de informação entre diferentes intervenientes do processo construtivo, eliminando dúvidas ou dificuldades. Esta tecnologia poderá auxiliar os projetistas em processos de decisão em fases muito iniciais do projeto.

Rendering, consiste no processo de produção de uma imagem a partir de dados armazenados num computador. *Render* corresponde à imagem processada (termo *render* utilizado independentemente da linguagem escrita). Um *render* poderá induzir o observador em erros por se encontrar confinado ao espaço bidimensional de um ecrã ou projeção. De facto, a interpretação de uma imagem plana, imprimida ou projetada, é sempre recriada pela capacidade de quem a está a visualizar e a interpretar a 3D. Ainda que se simule a terceira dimensão do espaço (imagem plana, recorrendo à perspetiva ou ao 3D), a imersão espacial é inexistente, comprometendo a perceção do espaço. [4]

Estes fatores considerados a jusante poderão traduzir custos acrescidos, o que demonstra uma necessidade de implementar novas estratégias com capacidade de otimização e prevenção de erros.

Recorrendo à RV através de HMD's conectados a um *software* de modelação, é possível imergir o observador no projeto modelado. Desta forma, a perceção da informação assemelha-se ao que aconteceria no local de construção. O ponto de vista passa de um observador "externo" à cena (caso do equipamento de "visualização plana" como ecrãs, projetores, etc.), para um participante do próprio modelo. Assim, determinadas questões que só se levantariam em obra surgem a montante, onde as alterações acarretam menores custos.

Como exemplo das mais valias para a representação do espaço surge o caso do projeto de renovação de um hospital a decorrer no VCUHS (Virginia Commonwealth University Health System). A obra irá incidir sobre uma área de aproximadamente 7900 m^2, situado numa zona povoada da cidade. O funcionamento deste hospital contempla 24 horas diárias, pelo que foi necessário optar por uma alternativa aquando médicos, enfermeiros e restantes profissionais foram chamados a pronunciar-se sobre o projeto de renovação.

Num caso habitual, a empresa responsável pelo projeto, levaria a cabo a construção de algumas maquetes à escala real que representassem determinados espaços relevantes do empreendimento. Nesse sentido, seria mandatário transportar um número restrito de pessoas para avaliarem o protótipo, tendo necessariamente que abandonar o seu local de trabalhado para uma deslocação ao lugar onde este estivesse instalado.

Perante a disponibilidade limitada dos profissionais em serviço e as condições espaciais do local, optou-se por recorrer à RV como solução. Assim, para obter um parecer por parte destes profissionais sobre o seu futuro espaço de trabalho foi utilizado um modelo BIM e explorado em conjunto com um HMD. Deste modo, tornou-se possível percorrer o edifício o modelo sobre uma perspetiva única.

A partir desta solução, foi viável explorar o modelo recorrendo ao *walkthrough*, isto é, o observador imergido num cenário virtual correspondente ao futuro edifício é capaz de se descolar neste através de simples comandos de movimento. Com a portabilidade deste sistema as sessões de avaliação puderam decorrer durante pausas no serviço, em gabinetes de escritório e reuniões. O custo desta alternativa representou apenas 15% do que seria esperado caso se tivesse optado pela realização de um modelo físico *(mockup)*. Ao longo de um mês foram sugeridas 35 alterações ao projeto, quer em salas de operação como de recobro, sobre questões relacionadas com a arrumação existente ser insuficiente, alterações no equipamento fixado a algumas paredes, entre outros.

FIG.2.1 - EXEMPLO DA APLICAÇÃO DE RV AO PROJETO DE RENOVAÇÃO DE UM HOSPITAL DO VIRGINIA COMMONWEALTH UNIVERSITY HEALTH SYSTEM (VCUHS) NOS ESTADOS UNIDOSDA AMÉRICA [5]

2.1.2. INVESTIGAÇÃO E ENSINO

Um modelo tridimensional imersivo poderá desempenhar um papel relevante na evolução da perceção visual dos espaços e/ou elementos, existentes ou não. A sensação de imersão num ambiente virtual distingue-se claramente da realidade transmitida por um ecrã ou vários em associação. No caso do ensino por exemplo, a apresentação de conteúdos que outrora seriam de difícil interpretação em formato 2D poderão ganhar, no que concerne a inteligibilidade, se apresentados recorrendo a tecnologia de RV.

A introdução de RV nas escolas permite aos estudantes considerar este tipo de tecnologia na sua vida profissional. A interação visual pode influenciar a docência na engenharia civil. Apenas uma pequena parte da informação transmitida nas escolas de engenharia é visual, sendo o contato dos estudantes com lições e material escrito francamente superior ao volume de informação visual à sua disposição. No entanto, a maioria dos alunos retém a informação de forma visual (os chamados *visual learners*), conferindo ao modelo virtual um papel de relevo como ferramenta didática.[6]

O modelo virtual poderá ser útil não só lições presenciais, mas também para aprendizagem à distância. Deste modo, a RV proporciona novas alternativas para um ensino técnico, de uma forma impraticável recorrendo a metodologias tradicionais. [6]

Exemplos de com alguma complexidade visual e apresentação contida numa só imagem são raros e difíceis de concretizar. Na generalidade dos casos são precisos conjuntos de plantas, cortes e pormenores construtivos para possibilitar a plena compreensão de alguns elementos construtivos.

A Fig.2.2 poderá ser de simples interpretação para um observador com alguma formação ou experiência no setor da AEC. No entanto, para um estudante ou observador sem formação na área a inteligibilidade da figura poderá ficar comprometida. Uma visualização mais interativa e representativa do elemento, da sua configuração no real, poderá auxiliar o processo de compreensão da escala e características do pormenor desenhado.

A RV aplicada ao ensino, na área da Engenharia Civil e Arquitetura, poderá atuar como complemento, influenciando positivamente a apreensão de conhecimentos a partir da imersão e interatividade proporcionadas por este tipo de tecnologia.

Em casos como o VizLab no Technion – Israel Institute of Technology, é dada a oportunidade aos alunos de controlarem a imagem ou modelo a partir de sensores, câmaras e controlos sem fios, viabilizando uma viagem virtual pelo ambiente projetado (Portman, Natapov, and Fisher-Gewirtzman 2015). Segundo Yehuda Kalay, professor e diretor da Faculdade de Arquitetura e Planeamento Urbano no Technion, Israel Institute of Technology a razão pela qual as pessoas respondem mais emotivamente a filmes no cinema que em suas próprias casas reside no fato de o grande ecrã proporcionar uma experiência imersiva. Outro aspeto a ter em conta é a experiência coletiva e social entre vários observadores que intensifica a experiência emocional.

Nesta ordem de ideias, o contributo dos laboratórios para a visualização é considerado uma mais valia como ferramenta para o planeamento urbano, potenciando a discussão, debate e tomada de decisões sobre design e planeamento de paisagens urbanas e naturais.[7]

No ITaP's Envision Center (Information Technology at Purdue Envision Center) (Fig.2.3) decorrem estudos e simulações que visam dar aos estudantes mais e melhores ferramentas para inspeção de pontes. Neste laboratório são feitas simulações que permitem incorporar condições climáticas, ruídos e também ferramentas que seriam utilizadas no local como lanternas, lupas e

vassouras. O trabalho desenvolvido pelo Envision Center permite auxiliar a investigação e o ensino, representando graficamente informação e dados com especial foco em simulações virtuais, interação humano-computador, visualização e análise de dados e criação multimédia.

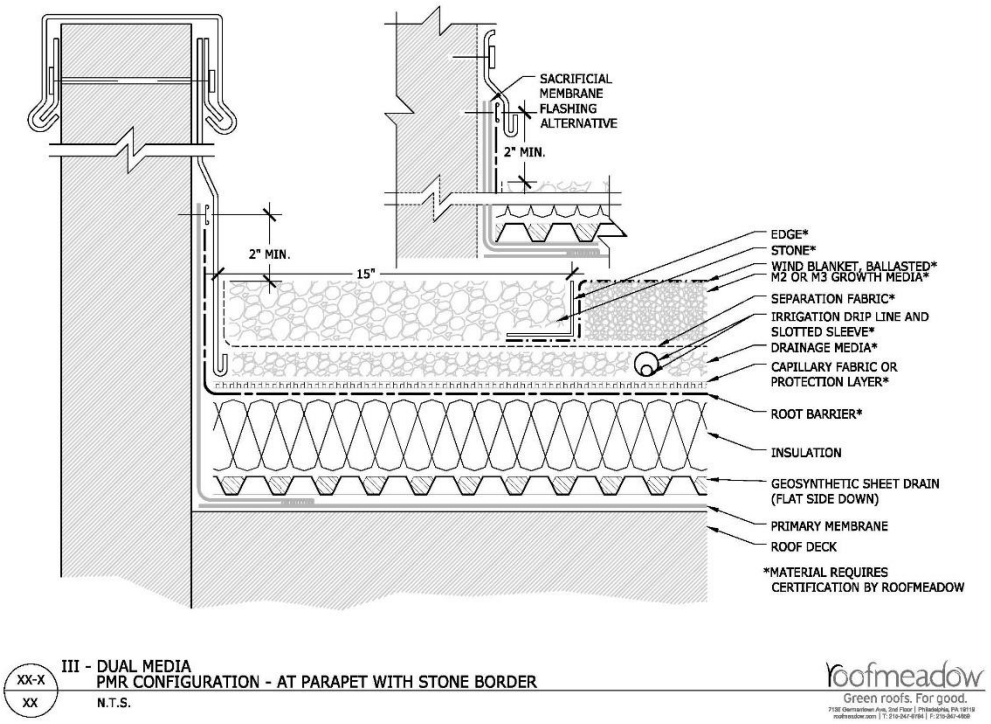


FIG.2.2– PORMENOR CONSTRUTIVO DE UMA PLATIBANDA E TERRAÇO [8]

Este ambiente 3D e imersivo criado no laboratório, está também disponível para integração em aparelhos portáteis como *HMD's*, equipamentos outrora proibitivos, mas atualmente disponíveis a preços mais acessíveis. [7].

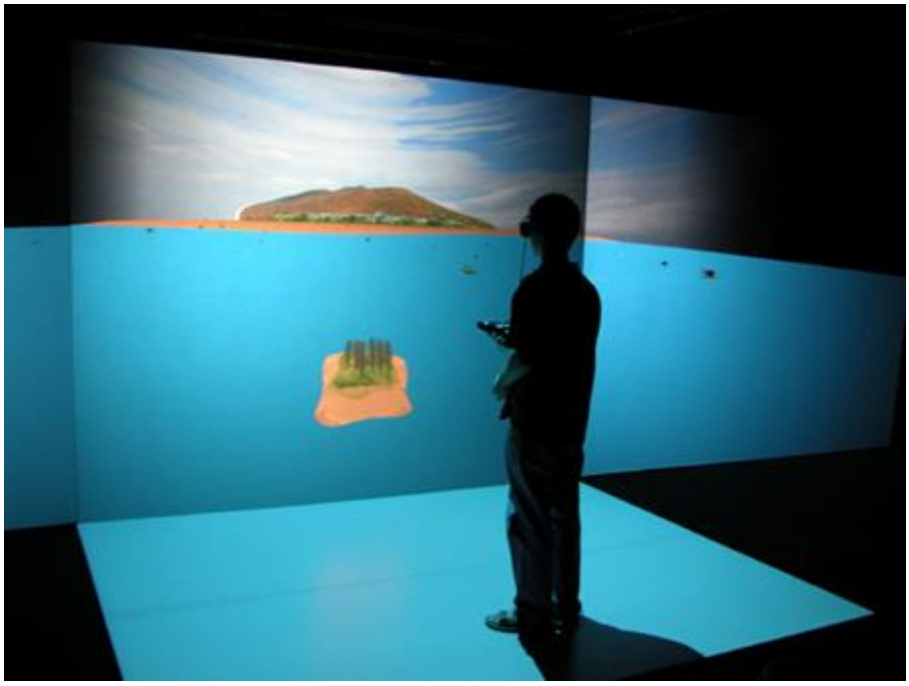


FIG.2.3 - ITAP'S ENVISION CENTER (INFORMATION TECHNOLOGY AT PURDUE ENVISION CENTER) [7]

2.1.3. SEGURANÇA EM OBRA

A segurança dos trabalhadores é vital para o bom decorrer das operações durante todo o ciclo de vida do processo construtivo. Sensibilizar os intervenientes que atuam no cenário de obra para as medidas de segurança e comportamentos que deverão assumir perante diversas situações poderá ganhar uma nova abordagem recorrendo à RV.

Para tal, através de modelos BIM e importando-os para motores de jogo, pode recriar-se cenários reais onde terão decorrido acidentes de trabalho. Os operários presentes na obra terão oportunidade de percorrer os vários modelos, imergindo no cenário. Através do motor físico (acessível pelo motor de jogo) é possível recriar colisões e interações com os objetos simulando consequências de determinados comportamentos. [4]

A apresentação visual e a interação com a causa de um acidente real através desta tecnologia é suscetível de ser melhor apreendida pelo observador face à sua explicação por métodos tradicionais (oralmente, projeção de imagens, etc.).

2.1.4. AUXILIAR À MODELAÇÃO 3D

Na vida real o modo como construímos algo a partir de ferramentas ou com as próprias mãos é, em muitos casos, um processo intuitivo. Não é absolutamente necessário que uma fonte exterior nos informe por exemplo, como moldar um determinado material ou unir duas peças. O processo decorre de forma natural e é precisamente sobre esse processo que a RV poderá intervir. Assim sendo, o modo como criamos algo no mundo real terá semelhanças ao método como modelamos em 3D. Com o auxílio de um HMD e controlos específicos, como os Razer Hydra, é possível encontrar algumas soluções que atualmente se dedicam a esta nova forma de interação com a modelação 3D.

O VRClay é uma aplicação que permite ao utilizador modelar objetos em pleno ar. O interface é feito em ambiente de RV e compatível com o Oculus Rift DK1 ou DK2 (*Development Kit* 1 ou *Development Kit* 2) e os controlos Razer Hydra (Fig.2.4).

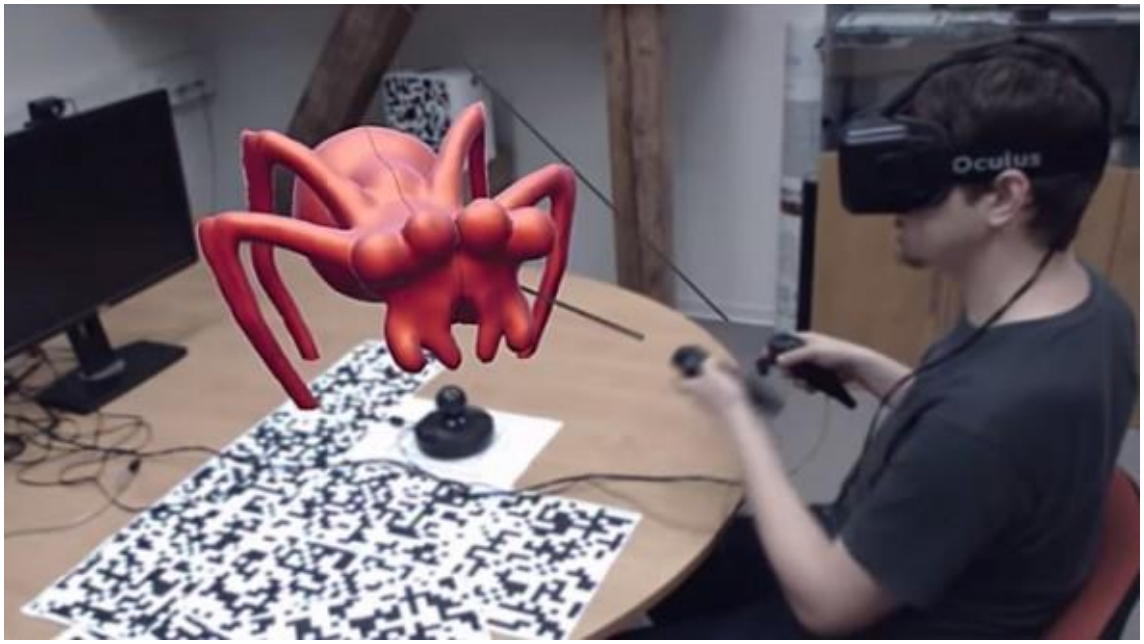


FIG.2.4 - MODELAÇÃO DE UM OBJETO RECORRENDO AO SISTEMA VRCLAY. NA FIGURA É POSSÍVEL RECONHECER O HMD OCULUS RIFT DK2 E OS CONTROLOS RAZER HYDRA [9]

Outro exemplo de um programa atualmente a ser desenvolvido para a modelação através de RV é o MakeVR da Sixense. A sua compatibilidade estende-se a qualquer tipo de HMD e como controlos recorre, de igual modo, aos Razer Hydra ou a dois controladores STEM (Fig. 2.5). Esta aplicação tem a particularidade de se poderem exportar os modelos produzidos para motores de jogo como o Unity e Unreal 4 e capacidade para operar com 5 utilizadores em simultâneo. [10]

A grande vantagem de uma interface deste tipo consiste na manipulação de objetos "com as mãos" como no dia a dia, sem recorrer a menus ou outro tipo de operações mais comuns nos programas de modelação. A figura 2.6 ilustra a utilização do MakeVR no caso prático de modelação do cenário para a ópera Turandot, podendo o resultado ser comparado com o cenário real representado na Fig.2.7.

FIG.2.5 - CONTROLADOR RAZER HYDRA (ESQUERDA) E STEM (DIREITA) [10]

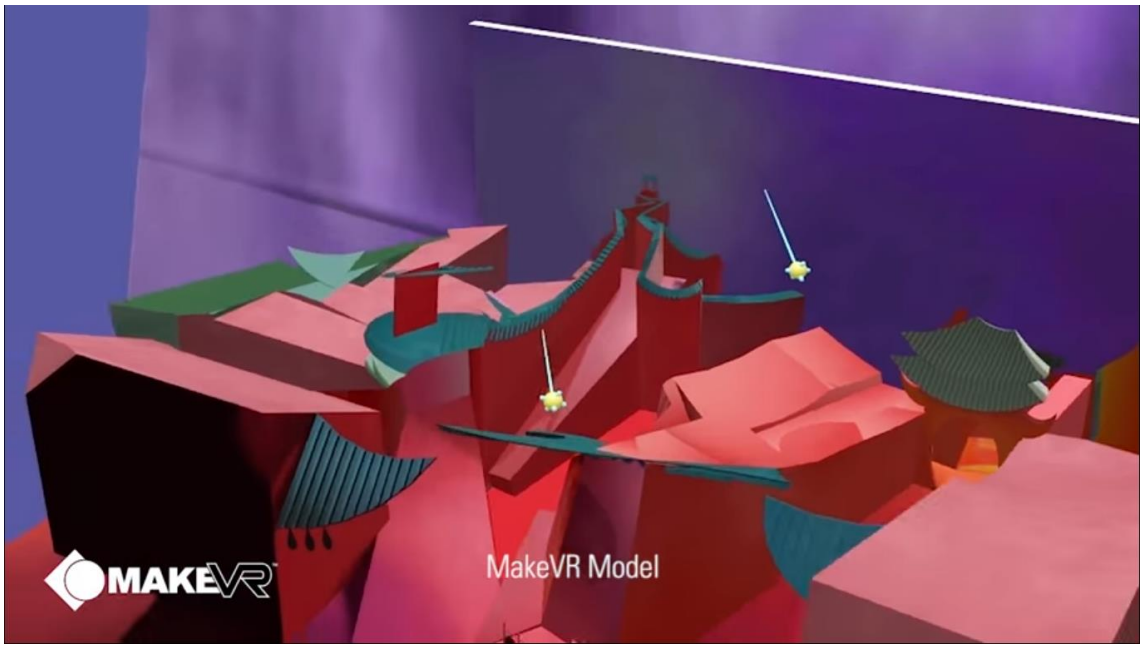


FIG.2.6- MODELO EM PRODUÇÃO ATRAVÉS ATRAVÉS DE MAKEVR (CENÁRIO PARA TURANDOT – DAVID HOCKNEY) [11]

FIG.2.7 – CENÁRIO (REAL) PARA A ÓPERA TURANDOT DA AUTORIA DE DAVID HOCKNEY [11]

2.1.5. FISCALIZAÇÃO E ACOMPANHAMENTO DO PROJETO E DO PROCESSO CONSTRUTIVO

Recorrendo à RV para percorrer um modelo BIM poderá presumir-se a capacidade de acompanhar o que de fato ocorre durante a construção e verificá-lo no modelo de um modo imersivo. Determinados aspetos poderão ser mais facilmente identificados caso se esteja a habitar um espaço virtual, comparativamente à observação de plantas, cortes, alçados e *renders 3D, etc*. Uma combinação com eventual potencial para a fiscalização poderá ter como base a interligação de RV com percursos efetuados por *drones*, transmitido a informação em formato de vídeo através de uma emissão online. Mais concretamente, o percorrer de um edifício por *drones* dotados de câmaras, ligados à internet e aos Oculus Rift (num computador remoto). Deste modo, controlado o percurso do *drone* e visualizando-o em tempo real, possibilitar-se-ia um acompanhamento de obra em tempo real, assim como a comparação com modelo BIM à distância.

FIG.2.8 – COMBINAÇÃO DE CONTROLO DE UM *DRONE* E VISUALIZAÇÃO DO PERCURSO ATRAVÉS DE UM HMD [12]

2.2. O CONCEITO DE RV

> *Realidade virtual consiste em reproduzir uma experiencia sintética representando um contexto de simulação virtual ou ilusório ao utilizador.* [13]

Ou ainda…

> *Realidade virtual pode ser definida como a apresentação de alargado campo visual de informação multissensorial gerada por computador que monitoriza um utilizador em tempo real.* [14]

RV é uma representação de um ambiente real gerado por computadores. Aquilo a que nos referimos como realidade consiste num conjunto de processos tendo como base os sentidos do ser humano e o fluxo de informação que estes são capazes de percecionar, conduzir e fazer interpretar pelo cérebro. Neste sentido, a simulação de um ambiente virtual, replicando informaticamente os elementos do real e induzindo sensorialmente no utilizador a noção de imersão numa atmosfera artificial, reflete o que sumariamente se interpreta como realidade virtual**.**

As origens da realidade virtual não podem ser atribuídas a um evento circunscrito no espaço e no tempo, todavia a um conjunto de necessidades e intenções dispersas, associadas a um acumular de pesquisas e ideias. Desta forma, foi através de um conjunto de iniciativas dispersas ao longo dos últimos 60 anos, suportadas por progressos graduais na investigação e na prototipagem, que se moldou o conceito atual de RV**.** De seguida, apresentam-se alguns marcos de relevo na evolução desta tecnologia.

1838 - Estereoscópios vitorianos - primeiros aparelhos que podiam gerar imagens 3D. Atribui-se a criação do primeiro tipo de estereoscópio a Sir Charles Wheatstone

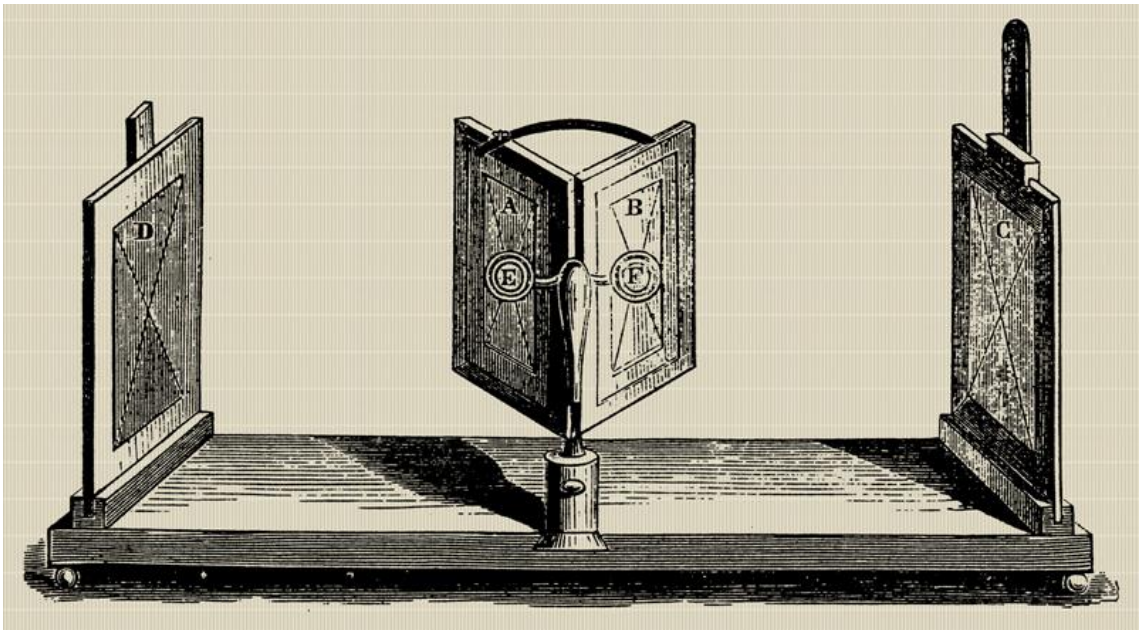


FIG.2.9 – ESQUEMA DO ESTEREOSCÓPIO INVENTADO POR SIR CHARLES WHEATSTONE [15]

1962 - Morton Heiling - primeiro sistema VR, "Sensorama" . Consistia numa cabine ao estilo *arcade*, com projeção de imagem a cores, assento vibrante, som estéreo, produção de aromas e sensação de vento (Fig.2.10).

FIG.2.10 – SENSORAMA – EQUIPAMENTO DESENVOLVIDO POR MORTON HEILING [13]

1963 - Jaron Lanier - construção de um head mounted display (HMD). Encontra-se exposto no museu de história dos computadores em Silicon Valley Foi ainda cofundador da VPL Research,

uma das primeiras empresas a comercializar material de RV como a "Data Glove" (Fig. 2.11) e o *EyePhone* (Fig. 2.12*).*

FIG.2.11 - EQUIPAMENTO DATA GLOVE [13]

FIG.2.12 - EYEPHONE DESENVOLVIDO PELA EMPRESA DE RV VPL NA DÉCADA DE 60 [16]

1965 - Ivan Sutherland - *The Ultimate Display* - Citado pelo próprio como "uma sala onde o computador pode controlar a existência de matéria". O protótipo do que é considerado um dos primeiros HMD's a reunir RV e RA (realidade aumentada), "Sword of Damocles" (Fig.2.13), surgiu em1968.

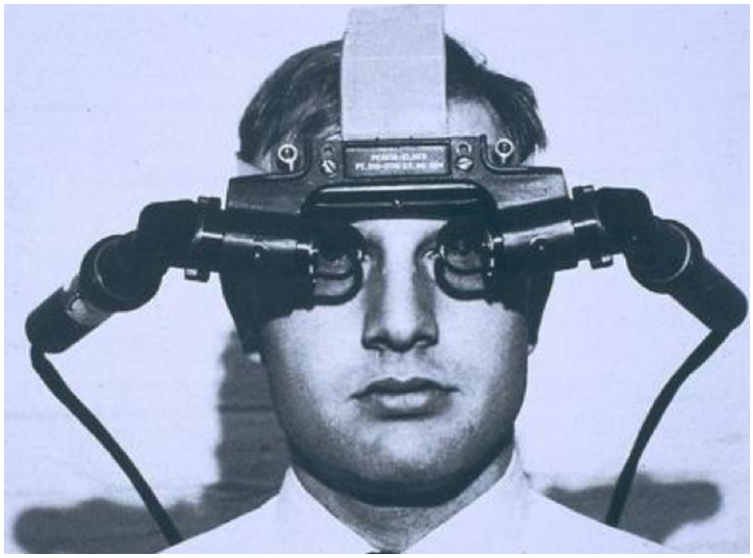


FIG.2.13 CONSIDERADO O PRIMEIRO HMD A CONJUGAR REALIDADE VIRTUAL E AUMENTADA, "SWORD OF DAMOCLES" [13]

Final da década de 60 - Myron Krueger apresenta um conceito que intitula como realidade artificial - exploração da interação humano-máquina. Em 1969 cria juntamente com Dan Sandin, Jerry Erdman e Richard Venezky o "GlowFlow". Este projeto tinha como fundamento a resposta por parte do ambiente virtual a nível visual e auditivo a todo o movimento realizado pelo utilizador.

1982 - 86 - Thomas Furness III desenvolve o "Visually Coupled Airborne Systems Simulator" conhecido como o capacete do Darth Vader. Foi ainda responsável pelo simulador de voo na base aérea Wright Patterson que mais tarde se tornaria no programa "Super Cockpit".

1982 - Atari Sunnyvale Research Laboratory - exploração do futuro do entretenimento digital. Local de trabalho e investigação de muitos dos pioneiros da realidade virtual como Brenda Laurel, Tom Zimmerman e Jaron Lanier.

1989 – Lançamento da Nintendo Power Glove (Fig. 2.14).

FIG.2.14 EQUIPAMENTO DE RV NINTENDO POWER GLOVE,
ALTERNATIVA À INTERAÇÃO FEITA COM *JOYSTICK* OU TECLADO [23]

Anos 90 – O crescimento do interesse pela realidade virtual gera a criação de um novo mercado de livros especializados na temática, revistas e boletins informativos. Durante esta década surgem referências como a Cyber Journal (criada por Ben Delaney, dedicada a negócios dentro da área da RV), MIT Presence (virtual environment research), PCVR criada por Joseph Gradeki da Universidade de Wyoming e sua esposa, uma publicação bimensal onde se apresentaram guias de como fazer sistemas caseiros de RV.

1991 – Daniel Sandin, Cruz-Neira e outros colaboradores desenvolvem no EVL (Electronics Visualization Laboratory) o conceito de CAVE (*Cave Automatic Virtual Environment)*, sendo este apresentado em 1992 na conferência SIGGRAPH.

1993 – Char Davies desafia as aplicações correntes da RV. Com a instalação "Osmose", Davies proporciona ao utilizador, através de um HMD e um fato de monitorização de movimentos, uma experiência imersiva pouco usual. O utilizador teria que recorrer à respiração e mudanças no seu ponto de equilíbrio para se poder movimentar no mundo virtual desta instalação, onde o grafismo também viria a marcar uma distinção com o realismo pretendido noutras aplicações de RV.

2012 – Lançamento da primeira versão do sensor de deteção de movimentos gestuais The Leap (atual Leap Motion, ver Fig.2.15.).

FIG.2.15 - SISTEMA DE DETEÇÃO DE MOVIMENTOS GESTUAIS THE LEAP, ATUALMENTE DESIGNADO COMO LEAP MOTION [18]

2012 - Palmer Luckey – Abertura da campanha online para angariação de fundos (Kickstarter) para o projeto "The Rift".

2014 - Aquisição da Oculus VR pelo Facebook – negócio que gerou 2 mil milhões de euros.

2016 – Possivelmente o ano que marcará a maior disponibilidade comercial de vários tipos de HMD's. Mudança de paradigma na indústria do entretenimento? [14] [19]

Existem um número vasto de referências no mundo da RV, sejam investigadores e diretores de laboratórios responsáveis pelo avanço da tecnologia, fundadores de empresas a pioneiros na aplicação da RV a campos nunca antes considerados. Da recente história da RV importa referir Stephen Ellis – Responsável pelo NASA Ames Research Center's Advanced Displays and Spatial Perception Laboratory; Scott Fisher –Fundador do NASA Ames Research Center's Virtual Workstation Project e cofundador da VR Company Telepresence Research; Tom Zimmerman – Cofundador da VPL, inventor da DataGlove; Mark Bolas – Diretor do Laboratório de Mixed Reality da Universidade da Califórnia do Sul e fundador da companhia de *hardware* de realidade virtual Fakespace; Skip Rizzo – recorreu à RV para desenvolver métodos para a reabilitação cognitiva, motora e de desordem de stress pós traumático. Estes são apenas alguns dos nomes responsáveis pelo progresso e evolução da RV.

Como noutros desenvolvimentos tecnológicos, a realidade virtual apresenta forte ligação à indústria do entretenimento. Este facto poderá ser reflexo de uma vontade de tornar esta tecnologia acessível para a população por valores comportáveis, contribuindo para a sua rápida disseminação e aceitação.

Empresas como a Sony, Samsung, Microsoft, Google, Facebook, Valve, entre outros, revelaram ao longo dos últimos anos a sua visão relativamente a equipamentos para suporte de RV. Brevemente encontraremos disponível no mercado um leque de diferentes opções de *hardware* de RV a preços competitivos.

2.3. REALIDADE AUMENTADA (VANTAGENS E DESVANTAGENS VS RV)

Realidade Aumentada (RA) refere.se à tecnologia que oferece, em tempo real, uma visão sobre o espaço envolvente do utilizador alterado ou completado com informação gerada por computador. Quando um utilizador examina o ambiente à sua volta através de equipamentos de RA, estará a ver informação sobreposta aos objetos em seu redor. [20]

Refere-se nesta dissertação a realidade aumentada como a sobreposição de informação sobre a realidade percecionada pelo utilizador.

As aplicações de realidade aumentada, ou RA, apresentam-se como uma solução cada vez mais refinada, ocupando progressivamente o seu lugar no quotidiano da população em geral. [22]

2.3.1. VANTAGENS DA UTILIZAÇÃO DE RA PARA A INDÚSTRIA DA CONSTRUÇÃO

A RA pode ser usada como auxiliar para a compreensão da informação. Com efeito, apresenta-se como uma mais valia para a visualização de vídeos em tempo real através da sobreposição a modelos 3D. Devido à constante maturação da tecnologia ao longo do tempo, começa a ser verificado que a RA demonstra viabilidade na combinação de realidade virtual sobreposta à do "mundo real".

Uma conexão com a rede e recorrendo a aplicações de RA, viabiliza o acesso a informação para a construção, análise de dados, resultados, etc. A partilha deste tipo de dados é passível de se realizar em qualquer local desde que para isso se disponha de acesso à internet. Neste sentido, basta que os vários utilizadores compartilhem um mesmo tipo de dispositivos para que possam aceder a uma vasta gama de informação. A capacidade de combinar informação planeada com a informação obtida do real, permite fornecer *in situ* uma melhor compreensão do elemento projetado. Desta forma é gerado um contributo positivo para um cumprimento mais estreito das intenções do projeto.

A RA apresenta determinadas particularidades como a questão da necessidade de recorrer a marcadores. A sua função recai na deteção das posições exatas para colocação do elemento virtual em contexto. Métodos alternativos fornecem outras opções face a este procedimento. Por exemplo, através da relação entre imagens captadas por câmaras e um sistema de coordenadas é possível a aplicação de RA nos campos da engenharia civil e indústria da construção. Assim se ultrapassam questões como a morosa aplicação de marcadores em ambientes mais complexos como os que surgem numa obra, por exemplo.

O desenvolvimento de equipamentos cada vez mais sofisticados, mais pequenos, portáteis, e até *wearable*, poderá facilitar a utilização da RA em cenários de obra, diminuindo a interferência que estes poderão causar nas tarefas a realizar pelos operadores.[21]

2.3.2. LIMITAÇÕES DA APLICAÇÃO DE RA

Existem limitações para o desenvolvimento e uso de ferramentas de RA destacando-se a necessidade de portabilidade dos equipamentos, a elevada quantidade de dados transferidos em tempo real que implica recorrer a métodos ou bases de dados capazes de realizar esse mesmo processamento. Assim sendo, terão que ser avaliadas medidas que recorram a suporte tecnológico complementar para que a RA se possa implementar.

Atualmente os métodos de interação de equipamentos de RA carecem de interfaces intuitivas, o que se reflete na produtividade dos utilizadores deste tipo de tecnologia em ambientes dinâmicos, como locais de construção. A interação com o equipamento deverá ser o mais natural possível, independente da maior ou menor experiência do trabalhador, a fim deste poder operar

corretamente com o dispositivo em causa. Funções relacionadas com o controlo exercem um papel de relevo na forma como o um operador poderá realizar com sucesso as tarefas que lhe são atribuídas num ambiente complexo. O manuseamento de mensagens, objetos virtuais e cenários num *display* de um equipamento não deverá, por questões de segurança e produtividade, comprometer a atenção do utilizador. Não se pretende que os utilizadores despendam mais atenção em como interagir com o equipamento de RA, do que na realização do trabalho em causa. [21]

O termo "imersivo" surge frequentemente associado à RV pois, de modo distinto da RA, a primeira permite que o utilizador "se encontre, ainda que virtualmente, no interior do projeto".

A opção de caminhar pelo modelo (*walkthrough*) dificilmente poderia ser produzida no caso de se optar pela RA. A sensação de habitar o modelo digital é exclusiva da capacidade de imersão da RV.

2.4. RV: HARDWARE

Uma das principais razões que desencadeou uma nova aposta por parte das empresas no desenvolvimento de RV, prendeu-se ao fato de as tecnologias necessárias se terem desenvolvido, sendo nos dias de hoje mais pequenas, portáteis e financeiramente acessíveis. [22] Contrariamente ao verificado na década de 90 do século passado e na primeira década do séc. XXI, nos últimos anos houve um grande contributo de desenvolvimentos de tecnologia de suporte para a RV. A redução de preços atingiu componentes especializadas como por exemplo os acelerómetros, sendo estes responsáveis pela monitorização de movimentos do corpo. De modo análogo, os ecrãs (associados aos *smartphones*) pelo seu aumento de resolução, contraste, taxa de *refresh* ou densidade de pixéis são na sua generalidade acessíveis à maioria da população. No mesmo sentido, foi relevante o aumento da capacidade computacional nos últimos tempos, o desenvolvimento de projetores, sistemas embebidos e sensores. [13]

A conjugação destes fatores tornou o panorama propício para que a RV voltasse a fazer parte da discussão no mundo das tecnologias, apresentando-se como um assunto da ordem do dia e com novos argumentos.

Neste subcapítulo serão apresentados os mais relevantes equipamentos de RV disponíveis no mercado durante os últimos 15 anos.

2.4.1. HMD

Os HMD's não são uma novidade para o mercado da RV. De facto, é um conceito detentor de algumas décadas e os primeiros passos na exploração do mesmo poderão ser atribuídos a pioneiros como Ivan Sutherland, Jaron Lanier, entre muitos outros.

No entanto, é irrefutável o recente entusiasmo em torno da RV. Novos HMD's estão prestes a ser comercializados. Para além do mundo do entretenimento, espera-se que outras áreas possam beneficiar de uma mudança de paradigma associada à introdução destas novas tecnologias.

2.4.1.1 OCULUS RIFT

Palmer Luckey apresentou em 2012 o que viria a tornar-se um fator de transformação na aproximação entre tecnologia e consumidores. Os Oculus Rift inicialmente lançados numa plataforma de angariação de fundos públicos para o desenvolvimento de projetos, Kickstarter, arrecadou 2,5 milhões de dólares. [13] Após dois *kits* disponíveis para *developers* (SDK's –

Software Development Kit), este equipamento HMD estará disponível para o público em agosto do presente ano por sensivelmente 530 euros.

FIG.2.16 - ESQUEMA DA CONSTITUIÇÃO DO OCULUS RIFT DK2 [13]

1 – As lentes estabilizam o ponto de foco de modo a que o utilizador tenha a perceção de profundidade.

2 – Através de dois ecrã de alta resolução (AMOLED), um para cada olho, é possível criar a sensação de imersão num ambiente virtual. Através da forma como os humanos utilizam a visão binocular para se aperceberem do efeito de profundidade, a distância e a distorção das imagens nos ecrãs criam a "ilusão" de total imersão para o utilizador.

3 – Do *development kit* 2 lançado em 2014, fazem parte, entre outros, um giroscópio, acelerómetro e bússola. Desta forma, os três sensores agregam os dados relativos ao movimento da cabeça e sincronização daquilo que o utilizador vê na simulação virtual. Existem adicionalmente sensores de infravermelho e uma câmara para aumento de precisão de deteção da posição e monitorização.

A análise dos dados é feita num computador próximo ao HMD com o auxílio de *software* especializado. Imagens de um mundo virtual são renderizadas adequando-se à posição da cabeça do utilizador. [23]

2.4.1.2 SONY PLAYSTATION VR

A Sony Playstation revelou em 2014 durante a Game Developers Conference (GDC) o protótipo do Projeto Morpheus, um HMD que segundo Anton Mikhailov, Engenheiro de *software* na Sony U.S R&D, começou a ser desenvolvido em 2011. O HMD da Sony destina-se a complementar a experiência dos utilizadores da plataforma de jogos da mesma, a Playstation 4. [24]

A versão apresentada durante a GDC de 2015 contou com um ecrã FHD (Full High Definition) com um *display* de 1920 por 1080 pixéis de 5,7 polegadas (aproximadamente 14,5 cm) dotando

o sistema de um campo de visão de 100º. Do equipamento revelado constavam ainda pixéis RGB, para auxílio na "fluidez da imagem" e uma frequência de 120 Hz (na GDC de 2014 o valor apresentado ficou pelos 60 Hz.). Importa ainda referir o aumento do número de LEDs de monitorização dos movimentos da cabeça, de 6 para 9, e a redução dos tempos de latência. Combinando estes efeitos produz-se uma redução substancial do *lag*, ou seja, diminui-se o que seria uma diferença percetível no tempo de resposta entre uma ação do utilizador e a reação do servidor. [25] [26]

Atualmente intitulado de Playstation VR (Fig.2.16), terá o seu lançamento previsto para 2016 e o seu custo para o público rondará os 353 euros.

FIG.2.17 - APARÊNCIA DO EQUIPAMENTO DE RV DA SONY, PLAYSTATION VR [19]

2.4.1.3 HTC VIVE

Anunciado em 2015 durante a MWC (Mobile World Congress) em Barcelona, o HTC Vive (Fig.2.18) surge como um HMD desenvolvido em parceria com a *Valve*, empresa norte americana que produz e publica jogos eletrónicos de outras firmas.

FIG.2.18 – EQUIPAMENTO DE RV VIVE DESENVOLVIMENTO EM CONJUNTO PELA VALVE E HTC [27]

A última versão apresentada em 2016 (MWC, fevereiro), o HTC Vive Pre, revelou pormenores do produto final que brevemente estará disponível comercialmente para o público com um preço aproximado de 710 euros.

Alguns aspetos inerentes a este equipamento prendem-se com a quantidade de cabos que forçosamente estarão envolvidos em toda a experiência. É o caso do cabo de cinco metros, que embora confira oportunidade para percorrer uma distância considerável, poderá causar incomodidade para o utilizador. Com efeito, a questão agrava-se se se considerar os três cabos adicionais para ligação USB, HDMI e áudio.

O HTC Vive possuí dois ecrãs de resolução 1080 por 1200 pixéis, com uma frequência de 90 Hz. É utilizada uma relação de 9:5 nos ecrãs, conferindo ao utilizador maior distância visual na vertical sem ter que mover a cabeça. No que respeita a sensores, são incorporados 37 no HMD, sendo no total 70 contabilizando os instalados nos acessórios essenciais para o funcionamento do equipamento. São ainda utilizados dois controlos manuais, cada um deles monitorizado por emissores de infravermelhos, sem fios. Estes emissores deverão posicionar-se em dois cantos do espaço, respetivamente. Desta forma, não só se registam os movimentos da cabeça do utilizador, mas inclusive a sua posição no espaço e os movimentos que as mãos (controlos) executam em relação a essa mesma posição.

O *software* "Chaperone" permite, em conjunto com uma grelha azul, visualizar uma silhueta dos objetos e pessoas na envolvente através de um duplo toque no botão "home" do controlo manual. Um pormenor relevante se se considerar que os espaços onde este tipo de equipamento estará a ser usado poderão variar consideravelmente em área. Deste modo, atua-se ao nível da segurança do utilizador que, à distância de um duplo toque, poderá ganhar consciência da sua envolvente. Simultaneamente o equipamento torna-se mais versátil ao disponibilizar tecnologias que adequam o seu uso em espaços de diferentes dimensões.

Importa referir que no Vive os dois controlos têm que ser permanentemente agarrados pelo utilizador, sendo que nestes existem botões adicionais para comandos mais específicos.

É possível estabelecer ligação com o *smarthpone* (IOS ou Android), o que possibilita a receção de mensagens e notificações através do HMD. Foi acautelada a utilização deste equipamento por pessoas que necessitem de óculos auxiliares de visão. Utilizadores nesta situação poderão trabalhar com o dispositivo de RV sem incómodos no seu manuseamento. [28]

TABELA 1 - OUTROS EXEMPLOS DE HMDS [24]

Outros equipamentos HMD:	
Samsung Gear VR [29] 	Utiliza as capacidades dos *smartphones* da linha Samsung Galaxy como processador e *display*. A conecção micro usb permite que o *smartphone* se conecte ao equipamento HMD, situando-se à frente das lentes. O seu ecrã de alta resolução associado ao efeito das lentes permite a imersão no mundo da RV. O seu lançamento ocorreu em novembro de 2015. Preço aproximado: 100 euros.

Zeiss VR ONE [30] 	Utiliza o motor Unity 3D disponível para plataforma Android e IOS, tornando mais versátil que a solução da Samsung. Qualquer *smartphone* que respeite as medidas da *dock* e requisitos mínimos do *software* poderá funcionar com este equipamento. Preço aproximado: 157 euros.
FOVE VR [31] 	Recorre a um sensor de infravermelhos para detetar o movimento dos olhos do utilizador. Assim, possibilita uma experiência distinta na medida em que é possível obter maior profundidade na imagem, conhecendo-se o foco visual do utilizador. Em soluções concorrentes não existe esta distinção fazendo com que a experiência com o FOVE VR se aproxime mais da realidade. O olho humano restringe o seu ponto de focagem, ou seja, em cada instante só uma parte do visível possui maior definição. Preço aproximado: 310 euros.
Avegant Glyph [32] 	A imagem chega ao utilizador através da reflexão feita por um conjunto de micro espelhos. Uma resolução de 1280x720 pixéis para cada ótica, estando o ângulo de visão restringido a 45º. Preço aproximado: 620 euros.

Razer OSVR [33]  	No seu desenvolvimento é usada uma plataforma *open source*, ou seja, é possível aceder ao código de desenvolvimento do equipamento. É ainda menos restritivo nos requisitos de *hardware* ou *software*. Preço aproximado: 177 euros
Google Cardboard [34] 	Solução de baixo custo para o utilizador possibilitando a experiencia de imersão em RV. Tira partido dos sensores preexistentes no dispositivo móvel como giroscópio, acelerómetro, magnetómetro, sistemas de posicionamento e capacidade de processamento para que possam ser utilizadas as aplicações compatíveis com o Google Cardboard. É constituído para além do cartão, como se pode inferir pelo nome, por imanes e duas lentes. A montagem deste tipo de equipamento pode fazer-se "artesanalmente" através de guiões que descrevem passo todo o processo. Preço aproximado: 13 euros.

2.4.1.4. CINETOSE

A sensação de enjoo provocada pela leitura dentro de um carro, transporte através de barco ou, no presente caso, pelo uso de um HMD, surge pela receção de impulsos desconexos. O líquido pleural (ouvido interno) envia informação para o cérebro indicando que o corpo se encontra em movimento, no entanto o corpo do utilizador está de fato parado. De forma simples, a sensação de enjoo consiste na resposta do organismo para um estímulo semelhante à ingestão de alguma toxina, que poderia ser a causa hipotética da divergência nos sinais detetados pelo cérebro. Este efeito é conhecido como cinetose ou *motion sickness* [35].

Nos HMDs de gama mais baixa, um conjunto de sensores detetam o movimento da cabeça do utilizador, sendo este movimento replicado para o ambiente renderizado (ambiente virtual). O mapeamento dos movimentos da cabeça deve ser o mais fiel possível, a fim de "enganar" os sentidos do utilizador, desvanecendo a fronteira entre o mundo real e aquele criado artificialmente. A perceção do movimento captada pelo ouvido interno e aquela que é detetada pela visão consegue ser facilmente distinguível, ainda que a variação em causa seja reduzida. Uma discrepância de milissegundos é suficiente para que algumas pessoas sintam mau estar ou efeitos agravantes. Os equipamentos de elevado desempenho apresentam a particularidade de poderem monitorizar a posição da cabeça do utilizador tendo ainda a capacidade de controlar o momento em que apresentam determinado conjunto de *frames*. O observador só irá visualizar *frames* coincidentes com a exata posição da cabeça. Equipamentos como o Google Cardboard, baseados nas qualidades de processamento de um *smartphone*, não apresentam estas características. A monitorização é feita tendo em conta a orientação da cabeça e não ao nível da posição que esta assume. Assim, HMD's que compartilham este formato não são capazes de controlar com a mesma precisão os tempos de visualização das *frames* como os de gama mais elevada, diminuindo a qualidade da experiência. [36]

Recorrente em HMDs, o *motion-sickness*, ou seja, o desconforto, tonturas, náuseas ou enjoos que utilizadores destes equipamentos possam sentir, poderá ocorrer caso a frequência de atualização da imagem seja "demasiado lenta". [13] Num esforço de minimização deste efeito os equipamentos de RV apresentam frequências de atualização cada vez mais elevadas. Ainda assim, estes valores serão sempre condicionados pelas prestações do computador (Oculus Rift: 90Hz, HTC Vive: 90Hz, Playstation VR Pre: 120Hz, etc.).

2.4.2. OUTROS EQUIPAMENTOS

A RV abrange muito mais que HMDs, sendo relativamente extensa a variedade de dispositivos que tiram partido desta tecnologia. Serão expostos de seguida alguns exemplos da variedade de equipamentos que poderão integrar a RV.

Com a evolução da RV surgiu o conceito de sensor háptico. Háptica deriva do grego haptikos, significando "a capacidade de entrar em contacto com". Os sensores hápticos permitem ao utilizador de RV tocar, sentindo os objetos estimulados com que entra em contacto. Assim, poderá usufruir da capacidade de contactar com um objeto virtual, sentindo uma resposta deste. Este fenómeno é chamado de *feedback* háptico. [37]

Uma experiencia substancialmente diferente de RV está patente nas mesas de projeção virtual (Fig.2.19). Neste tipo de dispositivo o efeito pretendido não se base na imersão, todavia o utilizador reconhece uma determinada cena que se apresentada de forma semelhante a um holograma. Estas mesas não recorrem ao uso de sensores de movimento, hápticos ou projeções. [13]

FIG.2.19 - MESA DE PROJEÇÃO VIRTUAL UTILIZADA POR UMA ESTAÇÃO DE TELEVISÃO ARGENTINA COMO SUPORTE A COMENTÁRIOS DESPORTIVOS NO DECORRER DO CAMPEONATO MUNDIAL DE FUTEBOL DE 2014 [13]

O conceito CAVE foi criado em 1991 no Electronic Visualization Laboratory (EVL) na Universidade de Illinois em Chicago (UIC) por Daniel Sandin, Carolina Neira, Tom DeFanti e outros colaboradores. Consiste num espaço/sala que pode ser ocupado por vários utilizadores. Possui vídeo 3D e áudio e nele são projetadas imagens em todas as superfícies (paredes, pavimento e teto). Para uma experiência imersiva o utilizador terá de estar munido de óculos estereoscópicos com um sensor de localização incorporado. Poderá ainda recorrer ao uso de um controlo do "tipo varinha" para assim interagir com os vários objetos do cenário.

Quando a experiência é partilhada por vários utilizadores, um deles assume a posição de observador ativo, controlando o ponto de referência da projeção. Os restantes presentes terão a posição de observadores passivos. Este tipo de uso da CAVE fomenta a discussão e o debate de ideias estando todos os utilizadores imersos num ambiente de RV.

Este tipo de equipamento encontra-se atualmente disponível em vários estabelecimentos de ensino assim como na indústria. Para além dos exemplos referidos no ponto 2.3.1.2., salienta-se o uso deste tipo de tecnologia na indústria automóvel com o caso da instalação ao estilo CAVE da Ford. Marcando a transição dos modelos à escala real em argila, a Ford recorre à "CAVE" (Fig. 2.20) para reduzir custos e aprimorar o design dos seus automóveis. [38]

FIG.2.20 - SISTEMA DE TESTE SEMELHANTE AO CAVE UTILIZADO PELA FORD NO DESIGN DOS SEUS AUTOMÓVEIS, ADAPTADO DE AUTOTRENDS MAGAZINE [38]

2.5 RV SOFTWARE

A RV retomou expressão com a chegada de novos HMD's nos últimos anos, exemplo do Oculus Rift, HTC Vive, entre outros. A conceção destes equipamentos tem como finalidade a indústria do entretenimento, o que não inviabiliza a aplicação dos mesmos noutras áreas. Ainda assim, a RV foca-se no entretenimento, mais do que em qualquer outra indústria, como um novo paradigma para a interatividade.

Embora o âmbito da aplicabilidade da RV nos vídeos-jogos exceda o âmbito deste trabalho, uma breve referência às ferramentas auxiliadoras no desenvolvimento de jogos surge como ponto de interesse. De facto, releva conhecer o modo como os vídeos-jogos são concebidos através de motores de jogo, pois este tipo de metodologias poderá adaptar-se a outras áreas do conhecimento.

Assim, motores de jogo representam sistemas que facilitam/auxiliam o modo como os jogos são construídos. Dispondo de ferramentas que permitem ultrapassar questões que, de outro modo, consumiriam mais tempo e linhas de código para serem executadas. Em suma, representam uma via mais rápida e eficiente para desenvolver um jogo.

Algumas das vantagens da utilização de motores de jogo na criação de conteúdos predem-se com o conjunto de características dedicadas para o desenvolvimento de jogos que se disponibilizam através destes programas. Representam, de fato, *software* especializado na construção da estrutura de jogos, francamente utilizado por *developers* pelas vantagens que apresentam face à construção de um jogo a partir do zero. Existem motores específicos para determinadas tipologias de jogos, fazendo da escolha de um motor de jogo face a outro um processo criterioso com consequências a jusante.

Outras contrapartidas da utilização destes programas assumem-se na capacidade de suporte de vários tipos de *inputs* como *touchpads,* controlos de consolas, etc.; na possibilidade de produção de conteúdos de forma simplificada associada a gráficos de alta qualidade e pela interoperabilidade entre diferentes plataformas de jogo.

É possível importar para o motor de jogo objetos de outros programas. Deste modo, é dado aos criadores a hipótese de utilizarem *software* de renderização de sua preferência. Aos ativos ou *assets* importados para o motor de jogo, poderão acrescentar-se caraterísticas para realce do realismo gráfico como sombras, iluminação, pós-processamento, entre outras. O fator colisão é também considerado dentro dos motores de jogo, fazendo com que os objetos sejam detetados pelo jogador no caso da ocorrência de um confronto direto. Assim, o ambiente adquire mais realismo na medida em que deixa de ser possível atravessar elementos que "existam fisicamente".

2.5.1. MOTORES FÍSICOS

Os motores físicos integram-se no próprio motor de jogo e exercem um papel significativo na simulação das leis físicas do mundo real. Embora os motores de jogos mais atuais combinem renderização com atribuição de caraterísticas relacionadas com a física, estas não são uma parte integral da modelação visual. Por outras palavras, o aspeto visual de um modelo carece das propriedades que o fazem obedecer às leis físicas como massa, atrito, elasticidade, etc. Deste modo, um modelo renderizado exclusivamente sobre o aspeto visual não possuí os atributos suficientes para interagir de modo realista com a envolvente. Como exemplo poderá imaginar-se o caso de uma laje de piso renderizadas e importada para o motor de jogo. Caso o objeto laje não carregue consigo a componente que o permita ser reconhecido parte do mundo físico, ou seja, que lhe confira rigidez, o utilizador não conseguirá caminhar sobre este objeto.

2.5.2. SCRIPTS

Scripts são um conjunto de instruções em linguagem de código desenhados para interagir com outras linguagens de programação. Representa um conjunto de linguagens não compiladas, ou seja, não precisam de ser convertidas para um ficheiro executável para poderem ser lidas. Por sua vez, designam-se de linguagem interpretada, são de implementação imediata e facilitada, apesar de leitura mais lenta.

Poderão atribuir-se *script* aos objetos do motor de jogo. Estas pequenas linhas de código farão, mediante a verificação de determinadas condições, que se manifeste uma ação relacionada com o objeto detentor do *script*. Assim, potenciam a utilização do motor de jogo simplificando a implementação de determinadas ações no ambiente em desenvolvimento.

Os *scripts* podem utilizar-se para fins tão diversos como:

- Iniciar o jogo em determinada posição;
- Adicionar e mover câmaras;
- Manipulação de luzes;
- Acionar eventos quando um jogador atinge determinada área;
- Etc.

Não existe um limite tangível de aplicações.

Algumas linguagens de programação utilizadas na criação *scripts* são: C#, Python, PHP, Pearl, Ruby…

2.5.3. TIPOS DE MOTORES DE JOGO E PRINCIPAIS CARACTERÍSTICAS

Os motores de jogo, de modo geral, enquadram-se em quatro categorias: 2D, 3D, *Game Mods* e *Mobile*. Excluindo as diferenças óbvias, o motor 2D e 3D são semelhantes no seu funcionamento. De salientar nos motores 2D o facto de se recorrer à perspetiva quando se pretende que em determinada cena o observador detete o efeito de profundidade. Numa em fase de produção poderão incluir-se vários *layers* (semelhante a camadas) que permitem manipular os objetos, alterando a sua ordem ou posição, sobre o que se admitiria num sistema axial como eixo dos zz, representando a profundidade.

Relativamente aos motores de jogo 3D a sua complexidade aumenta, quando comparados aos 2D. Com efeito, para além da renderização de modelos 3D estes motores aplicam adicionalmente texturas 2D e possuem os seus próprios motores físicos.

Outro aspeto relevante consiste no uso cada vez mais regular de programação em motores 3D face ao sistema *drag-and-drop* mais característico dos motores 2D. Este último não requer familiarização com linguagens de programação possibilitando, deste modo, que utilizadores sem conhecimentos especializados em programação possam desenvolver estruturas para jogos.

Motores de jogos *Mobile* destinam-se a dispositivos móveis, tais como *smartphones, tablets,* etc. A evolução deste tipo de dispositivos é inegável, assim como a permanente atualização dos motores que possibilitam o desenvolvimento de jogos dedicados para estes equipamentos. [36]

O leque de motores de jogo é verdadeiramente extenso e não compete a este trabalho explorar a variedade de opções disponíveis para o utilizador. Serão focados apenas os motores de jogos com capacidade de suporte de RV, salientando quais as suas características. A opção por um determinado motor de jogo é feita, em primeira ordem, pelo maior número de vantagens que este possa oferecer para o projeto a desenvolver. No entanto, a preferência por uma determinada

linguagem de programação, a viabilidade económica ou disponibilidade para adquirir um determinado motor ou equipamento face a outro, podem ditar o rumo dessa escolha.

TABELA 2 - CARACTERÍSTICAS DE ALGUNS MOTORES DE JOGOS QUE SUPORTAM RV

Motores de jogo	Plataformas RV	Linguagem de programação	Sistema Operativo	Outros
BlenderVR	Oculus Rift, CAVE, Video Wall, Plano Vision, Mesa 3D…	Python (API)	Windows PC, Mac OS, Ubuntu, entre outros.	Grátis e *open source*.
Unity VR	Oculus Rift, Vive, Samsung Gear VR, Sony Playstation VR e Google Cardboard.	C#, Boo e JavaScript	Windows PC, Android e IOS através do Unity Player Plug-ing (Firefox, Opera, Mozilla, Camino, Chrome, Netscape, entre outros).	Possui versão gratuita
UE4	Oculus Rift, Vive, Samsung Gear VR e Playstation VR.	C++	Windows PC, Mac OS , Android, Linux, Steam OS e HTML5.	Possui versão gratuita

O utilizador poderá optar por vários *workflows* para executar a transferência de informação a partir do modelo desenvolvido no programa BIM (ponto de partida), até ao objetivo de o exportar para um equipamento de suporte de RV. Monteiro (2013) desenvolveu pesquisa na área dos *workflows* entre programas BIM e vários motores de jogo. Este tipo de operação não é imediato e poderá comportar vários passos intermédios com diferentes alternativas. Na sua pesquisa, apresenta 36 processos de transferência de informação entre programas BIM (Autodesk Revit 2013 e ArchiCAD 16) e motores de jogo (Unity 4 e UDK – Unreal Development Kit). São explorados vários tipos de ficheiro de importação e exportação como FBX, OBJ, DXF, e descritas as consequências de adoção de cada um destes [39].

Nos testes desenvolvidos por Monteiro (2013) o Unity revelou maior preservação de informação do modelo inicial. As características sobre análise foram imagem, GUID (Globally Unique Identifier), escala, material e textura. Estes fatores aliados à capacidade de programação utilizando a linguagem C# determinaram a escolha do Unity como motor de jogo para a produção de uma interação entre equipamentos de RV e *software* BIM.

3
INTERAÇÃO ENTRE RV E *SOFTWARE* BIM

3.1 PROCESSOS DE INTERAÇÃO

A Realidade Virtual atuando em simultâneo com *software* BIM poderá influenciar o modo como a informação se transmite e na forma como esta é apreendida pela variedade de responsáveis integrantes do projeto ou obra. Uma comunicação mais eficaz, a jusante, poderá instigar uma execução mais objetiva, rápida e com menos custos.

Este tipo de método começa a ser utilizado por algumas empresas internacionais e os resultados desenham-se positivos, especulando-se sobre um provável crescimento na adoção da tecnologia nos próximos anos.

O avanço tecnológico chegou ao ponto de ser capaz de replicar a sensação de imersão num edifício. Através de um HMD, como o Oculus Rift, engenheiros, donos de obra, arquitetos, empreiteiros, etc., poderão visitar virtualmente os seus projetos e percorrer os espaços que se propõem a construir.

Por outro lado, as ilustrações com grande componente realista, mais conhecidas como *renderings*, são genericamente as mais utilizadas para a apresentação de um projeto a um cliente. No entanto, constata-se que representam uma alternativa mais cara quando confrontada com a utilização de um modelo com RV para a mesma finalidade. [40]

FIG.3.1 - VISUALIZAÇÃO DE UM MODELO ARQUITETÓNICO ATRAVÉS DO HMD OCULUS RIFT DK2 [40]

3.2 INTERFACE DESENVOLVIDA PARA INTERAÇÃO COM RV

Um dos objetivos deste trabalho consiste na criação de uma ferramenta base, assumindo-se como

prova de conceito, sobre a interação entre equipamentos de RV e *software* BIM. Neste sentido, importa referir o conceito, ideia e visão que guiaram e simultaneamente foram acompanhando todo o processo.

Uma das ideias chave que rege este trabalho consiste numa clara intenção de facilitar o diálogo entre utilizadores familiarizados com a metodologia BIM e utilizadores que não possuam formação ou dominem esta tecnologia. Pode concretizar-se imaginado uma situação em que dono de obra seja interpelado a pronunciar-se sobre um pormenor do projeto. O diálogo sairia prejudicado caso o dono de obra, neste cenário hipotético, não domine a terminologia do mundo da construção nem disponha de formação para leitura de documentação associada ao projeto. Aqui se exemplifica um caso prático que beneficiaria com o recurso a um meio facilitador do discurso, e que em simultâneo recorra à metodologia BIM sem que esta se materialize numa barreira para os interlocutores.

Em suma, uma aposta na atenuação das barreiras entre utilizadores BIM e restantes intervenientes no processo construtivo, poderá traduzir um diálogo mais objetivo e participativo, assim como vantagens na realização de determinadas tarefas.

3.2.1. CRIAÇÃO DE UMA PLATAFORMA DE INTERAÇÃO

No desenvolvimento da interface foi utilizado o motor de jogo Unity (versão 5.3.4f1). Neste trabalho, o Unity surge como intermediário para que os modelos BIM, importados para este programa em formato próprio, possam ser manuseados em ambiente virtual.

Dada a forma expedita com que o motor de jogo interage com equipamentos de RV é possível, na sua versão mais recente e através de pequenos passos, conectar o modo de visualização da cena a um HMD. O utilizador terá apenas de efetuar os seguintes passos: *Edit* → *Project Settings* → *Player* e no separador *Other Settings* selecionar a opção *Virtual Reality Supported*.

Neste programa é criado um cenário destinado à produção de um ambiente de jogo. Para a finalidade desta dissertação o Unity foi utilizado para visualização e edição de uma pequena parte de um modelo, por exemplo, um excerto de uma vista 3D de um modelo BIM.

Dentro do cenário disponibilizado podem ser importados vários objetos vindos do *software* BIM desde que respeitem determinados tipos de formato de ficheiro como o FBX (*Filmbox*) e o OBJ (*Object file*). Estes formatos de ficheiro garantem uma interoperabilidade entre programas distintos, permitindo que a geometria dos objetos "a transportar" seja preservada. Fatores como texturas e cores não são comportadas neste tipo de formato e caso sejam relevantes para um determinado projeto devem ser adicionadas recorrendo a programas dedicados como 3ds Max ou Autodesk Maya ou através de aplicações como o Universal Material Converter.

O Unity utiliza *assets* como ferramentas que se podem introduzir na cena com aplicações largamente diversificadas, fazendo aumentar o leque de funcionalidades à disposição. Os *assets* revelam-se essenciais para um maior controlo do cenário em causa. Exemplificando, é possível escrever algumas linhas de código, criando um *script* e fazê-lo chegar à cena como um *asset*. Esse *script* terá uma função muito específica preenchendo um requisito particular do projeto em mãos. O mesmo seria o caso de um modelo exterior ao Unity ou um pacote específico de funções que teriam que ser importados como novos *assets*.

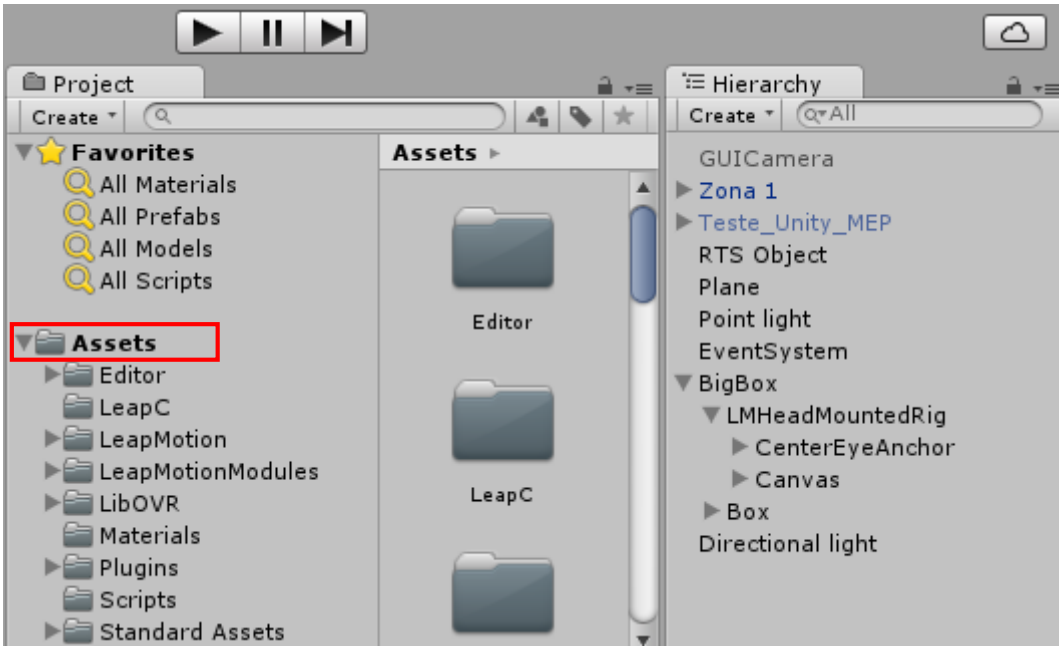


FIG.3.2 – EXEMPLO DE ALGUNS DOS *ASSETS* UTILIZADOS NUM PROJETO UNITY

3.2.2. EQUIPAMENTO EXPLORADO NA CRIAÇÃO DA INTERFACE EM RV

Para além do HMD Oculus Rift DK2 foi ainda utilizado um sensor de leitura de movimentos gestuais. O Leap Motion "lê" os movimentos das mãos do utilizador através de duas câmaras e três LEDs de infravermelhos até uma distância de 80 cm, afastamento limitado pela propagação da luz LED no espaço. O aparelho recebe a informação captada pelos sensores sendo esta enviada via USB para o Leap Motion *tracking software*. A informação recolhida assume uma escala de cinzentos próxima do espectro da luz de infravermelho, sendo ainda separada entre câmara esquerda e direita.

Através de um protocolo de transporte, os dados comunicam com bibliotecas dedicadas e com o painel de controlo do Leap Motion.

FIG.3.3 – UTILIZAÇÃO DO SENSOR LEAP MOTION

Para alcançar uma interação distinta daquela que é tradicionalmente feita com um computador, a relação obtida entre estas duas peças de *hardware*, Oculus Rift e Leap Motion, foi testada e consecutivamente sujeita a ajustes. O objetivo seria criar, ou reutilizar comandos para que, de forma natural, permitissem editar a geometria de objetos do projeto.

Como representado na Fig.3.3 basta ao utilizador erguer as mãos até que estas estejam no seu campo de visão para que possam surgir no cenário virtual e interagir com o projeto a decorrer.

A integração do módulo *Pinch Utilities* desenvolvido pela Leap Motion para a sua mais recente API (*Application Programming Interface*) Orion (atualmente em fase beta), tornou-se imprescindível para desempenhar as funcionalidades e comandos pretendidos. Este módulo permite ao utilizador interagir dentro da cena do Unity com os objetos importados, recorrendo ao sensor de captação de gestos. São apresentadas duas versões de demonstração, o Pinch Draw e o Pinch Movement, para desenho através do comando *pinch* e manuseamento da geometria de objetos, respetivamente.

No âmbito deste trabalho os recursos utilizados repousaram nas potencialidades do Pinch Movement (Fig.3.4). Esta versão de demonstração fornece *scripts* editáveis responsáveis pela interação com os comandos de escala, rotação e translação de um objeto através do comando gestual *pinch*. O gesto consiste na união entre a extremidade do indicador e do polegar da mesma mão. Este comando pode ser executado apenas com uma das mãos, ou simultaneamente com as duas, desde que ambas estejam no "campo visual" do sensor. Ilustrando um exemplo prático, é possível executar a translação e rotação de um objeto recorrendo apenas a uma das mãos. No entanto, o comando de rotação torna-se mais eficaz se usado com ambas. Outras funções como a manipulação da escala, terão que ser executadas exclusivamente com as duas mãos (dentro deste módulo, sem nenhuma alteração ao seu código base).

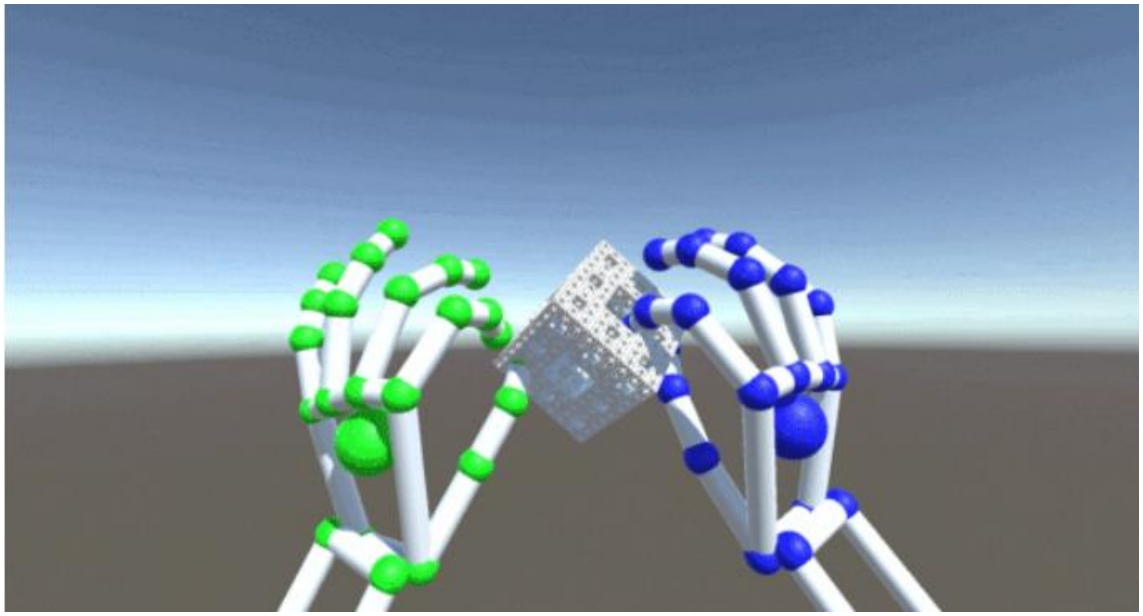


FIG.3.4 – EXEMPLO DA APLICAÇÃO DIRETA DO DEMO PINCH MOVEMENT (INCLUÍDO NO MÓDULO PINCH UTILITIES) NO MOTOR DE JOGO UNITY, ADAPTADO DE LEAP MOTION [41]

O módulo Pinch Utilities, mais concretamente a versão de demonstração Pinch Movement, no seu estado original apresentou limitações para o desenvolvimento da interação pretendida com os objetos BIM. Para facilitar a leitura de cada uma das limitações encontradas, será apresentada uma tabela com cada caso de dificuldade descrito detalhadamente no texto que se lhe segue. A cada limitação presente na Tabela 3 será associado um número para que possa ser apresentada de forma discriminada cada uma das soluções encontradas numa segunda tabela.

TABELA 3 - PINCH MOVEMENT – LIMITAÇÕES PARA O DESENVOLVIMENTO DE UMA PLATAFORMA DE INTERAÇÃO ENTRE RV E *SOFTWARE* BIM

1 - Manusear individualmente diferentes objetos	Especificamente a versão de demonstração Pinch Movement permite a edição de qualquer objeto em cena. Todavia, é necessário que o objeto alvo da edição esteja contido, no editor do Unity, num objeto virtual. Por sua vez, este objeto virtual possui como componente o *script* Leap RTS (RTS – *Rotation Translation Scale*). O Leap RTS consiste num código que, de forma essencial, sinaliza "a mão esquerda e direita" (respetivamente Leap Hand Controller Left e Right) e define ainda como cada utilizador interage com o comando de rotação e escala. O comando translação é definido no código do Leap Hand Controller. Resumidamente, existe uma "caixa" virtual à qual está associado o código Leap RTS e qualquer objeto introduzido na cena e colocado dentro desta estabelece com "a caixa". Esta relação hierárquica ou de parentesco, permite que o objeto seja controlado recorrendo ao comando gestual Pinch do Leap Motion. A dificuldade surge quando no projeto existe um número superior a um de objetos a serem manuseados. Supondo cumprida a obrigatoriedade de se encontrarem dentro da "caixa" para que o *script* que viabiliza os comandos gestuais lhes possa ser associado. O resultado obtido ao tentar rodar um objeto, por exemplo, traduz-se numa rotação de todos os objetos dentro da caixa virtual. Em suma, todos os objetos cumprirão de forma idêntica um comando que estaria a ser aplicado somente a um deles. Manusear individualmente apenas um objeto não seria uma tarefa possível.
2 – Rotação de um objeto sobre si mesmo	O comando rotação inicia-se pela deteção de um simples *pinch*, quer seja feito por uma mão, ou pelas duas em simultâneo. A rotação associada à mão do utilizador fará o objeto selecionado rodar no mesmo sentido. A limitação acontece no eixo que é definido como charneira da rotação. No caso de um Pinch singular (só uma mão), a charneira terá a sua origem na posição espacial da mão.

	Por outro lado, caso o comando Pinch se efetue com ambas as mãos, a origem do eixo de rotação será o ponto médio (no espaço) "entre os dois *pinchs*". Este ponto médio poderá ser entendido como um CIR (centro instantâneo de rotação). $$Eixo\ de\ rotação = (u + v)/2$$ Onde u e v se referem às coordenadas espaciais dos gestos de *pinch* realizados por cada uma das mãos Caso a localização do objeto não coincida com a origem do eixo de rotação ou CIR, a rotação acontecerá em torno de um eixo que não corresponde ao próprio eixo do objeto. Quer isto dizer, que este rodará em torno de um CIR muito distinto daquele que iria coincidir com o seu centro geométrico. Por outras palavras, não irá exercer uma rotação sobre si mesmo.
3 – Alterar a escala de um objeto segundo um só eixo ortogonal (eixo xx, por exemplo)	O comando que permite alterar a escala de um objeto, incorporado no script Leap RTS, só atua mediante a deteção de dois *Pinchs* em simultâneo. Significa que é um comando direcionado para ambas as mãos. No seu código a escala de um objeto é alterada em função do afastamento. Se existir a necessidade de alterar a escala de um objeto segundo uma dimensão, hipoteticamente segundo o eixo xx da peça, com o comando *default* do *Pinch* Movement não seria possível. A resolução para esta limitação foi encontrada, todavia, optou-se por não a incluir na interface final deste trabalho. Esta opção é discutida com maior pormenor no ponto 3 da criação de novas funcionalidades.
4 – Identificação do objeto selecionado	Por defeito o *Pinch* Movement não foi criado para o manuseamento de mais que do que um objeto ao mesmo tempo. Deste modo, poderá justificar-se o porquê da inexistência de uma distinção visual que advertisse para qual o objeto que estivesse a ser alvo de seleção.
5 - Movimentação da câmara e deslocamento do observador	O Unity recorre a objetos do tipo câmara para reproduzir visualmente (modo *play*) os acontecimentos que ocorram na cena. A câmara à qual estão associados os controlos virtuais da mão esquerda e direita, surge estática. A sua posição é alterada exclusivamente no editor e não em tempo real pressionando alguma tecla. Neste sentido, o observador encontra-se fixo a uma posição (x, y, z) no espaço e todos os comandos que decretar terão que ser exercidos a partir desse ponto de vista.

Pela ordem numérica da Tabela 3 descrevem-se de seguida os procedimentos que levaram à criação de novas funcionalidades para que o módulo importado se pudesse enquadrar nos objetivos deste trabalho.

1 - Manusear individualmente diferentes objetos

O editor do Unity permite a criação de novos objetos de jogo, ou como o próprio *software* os designa, *game objects*. Os *game objects* são uma unidade básica de trabalho para a criação de uma cena neste motor de jogo. Podem ser visíveis ou invisíveis, físicos ou sem propriedades físicas, sofrer a influência da gravidade ou ser independentes desta lei física. As propriedades dos objetos são definidas pelos componentes a eles associados. Poderão ser de origem, ou seja, previamente definidas no motor de jogo, ou criadas através de *scripts*.

Na interface em estudo houve a necessidade de criar um novo *game object*, uma "Big Box" (nome atribuído a este objeto dentro do projeto). Como a designação permite inferir, esta vai atuar como uma grande caixa virtual (invisível no decorrer da cena). A "Big Box" assume pelo motor de jogo o atributo de parente dos objetos que se coloquem "no seu interior". Dentro deste grande contentor de objetos foram criados outros três. Um "contentor mais pequeno" por razões de organização e, dentro deste, dois *game objects* (ver Fig. 3.5). A um deles foi atribuído o *script* Leap RTS e ao restante um novo código designado por App.

Na "Big Box" existe ainda um *game object* (LMHeadMountedRig) que encerra todos os códigos responsáveis pela interação entre a câmara de RV e os controlos gestuais efetuados através do Leap Motion. Este objeto é criado por defeito dentro da versão de demonstração Pinch Movement (Fig.3.5).

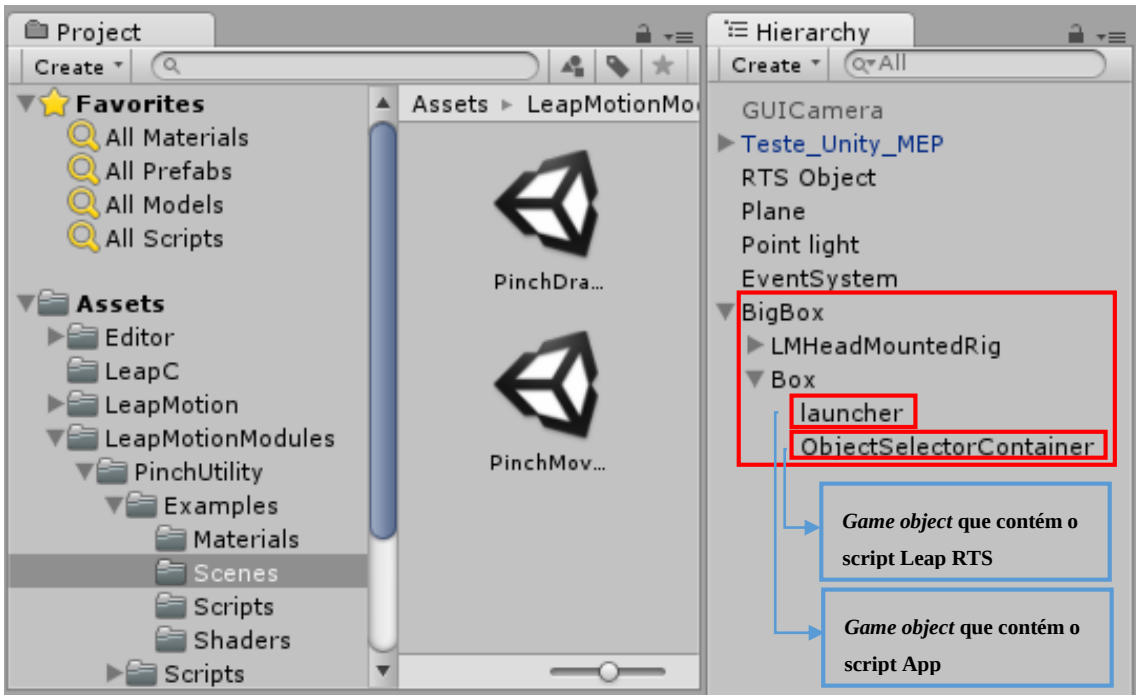


FIG.3.5 – HIERARQUIA DOS *GAME OBJECTS* CRIADOS DENTRO DA "BIG BOX"

O *script* App atua sobre várias das limitações patentes no Pinch Movement. No entanto, para o caso 1 o App procede da seguinte forma:

- Ao importar os objetos através do formato FBX do Revit para o motor de jogo Unity, antes de executar a cena através do comando *play*, o utilizador terá que atribuir *tags*. *Tags* não são mais que etiquetas que cada *game object* poderá, ou não, obter. Mediante a identificação de uma t*ag* específica poderá ser ativada uma ação. Esta funcionalidade evita que determinados elementos, como por exemplo, uma laje ou uma parede (sem *tag*) sejam selecionados por engano quando se tenta manusear um objeto anexo aos mesmos.

- São assim identificados todos objetos que possuam uma *tag* com o nome BIM. Deste modo, o utilizador define uma gama de objetos que pretende manusear.

- Perante a câmara utilizada para visualizar a cena, é criado um vetor entre a posição de observação e o objeto mais próximo (Fig.3.6). Importa reforçar que o vetor só será criado entre a posição da câmara e um objeto que possua a *tag* com o nome BIM.

- O passo seguinte do código consiste na formação de um ângulo entre dois vetores. O primeiro é designado pelo Unity como *forward*. Consiste num versor de direção frontal segundo o ponto de observação da câmara. O segundo vetor (Fig.3.8) é formado pela posição inicial do observador e a posição em que se encontra o *game object*.

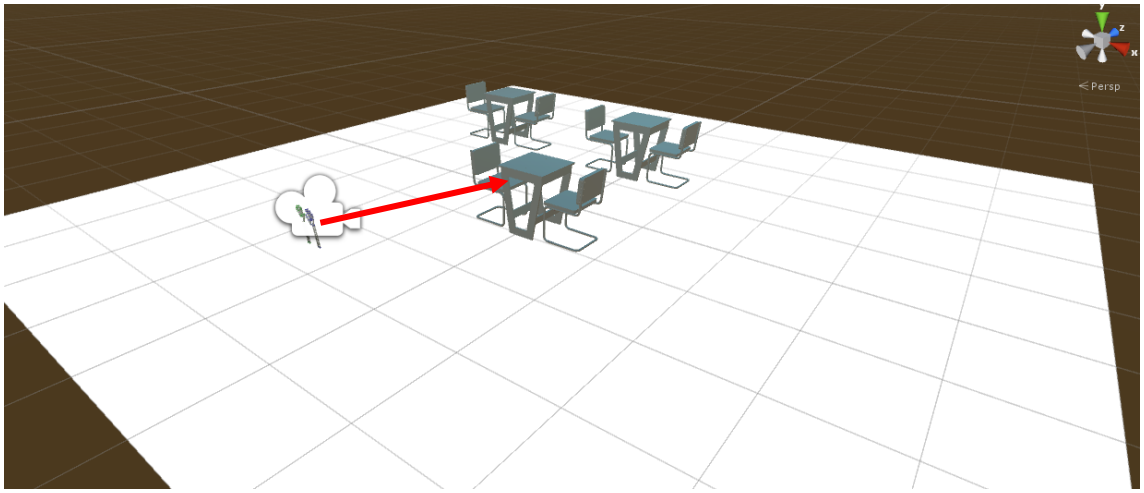


FIG.3.6 – ESQUEMA DA OPERAÇÃO QUE CRIA UM VETOR ENTRE A POSIÇÃO DA CÂMARA E O OBJETO COM *TAG* BIM MAIS PRÓXIMO

- Foi estabelecido como predefinido um ângulo de 15º $= a_0$, representado na Fig.3.7;

- Consoante o número de objetos no campo visual, por definição igual a 15º numa primeira iteração:

$f(nº\ de\ objetos\ em\ vista) = \alpha\ (°)$;

Se $f(nº\ de\ objetos\ em\ vista) \geq 4$, $a_0/2 = \alpha$;

Este ciclo poderá repetir-se até que o valor de α atinja 3,25º, definido no código como limite de precisão máximo.

- Por cada objeto que exista no campo visual do observador é avaliado o ângulo entre os vetores anteriormente referidos. Se este for inferior ao valor de α, o objeto é guardado.

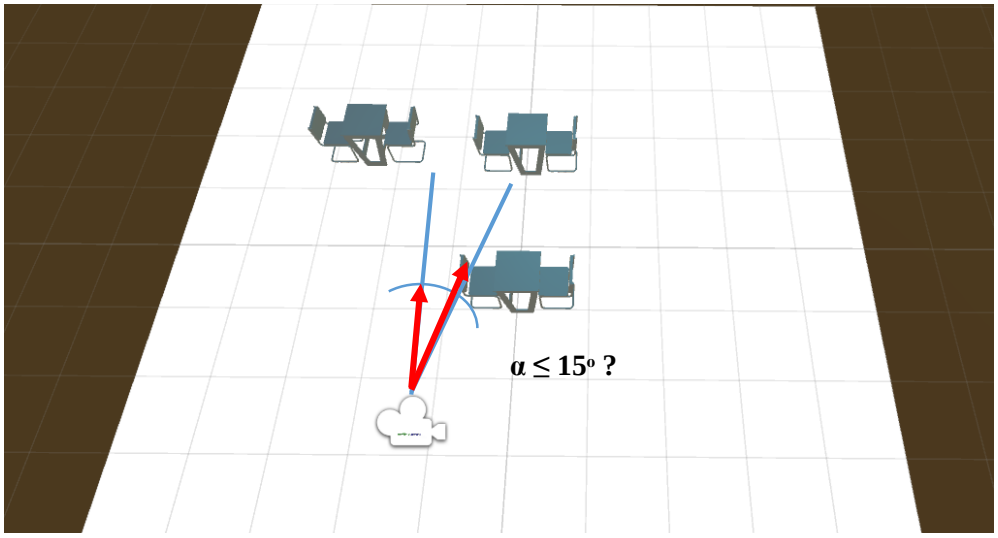


FIG.3.7 - COM APENAS 3 OBJETOS NO CAMPO DE VISÃO, α SERÁ COMPARADO COM UMA AMPLITUDE DE 15º

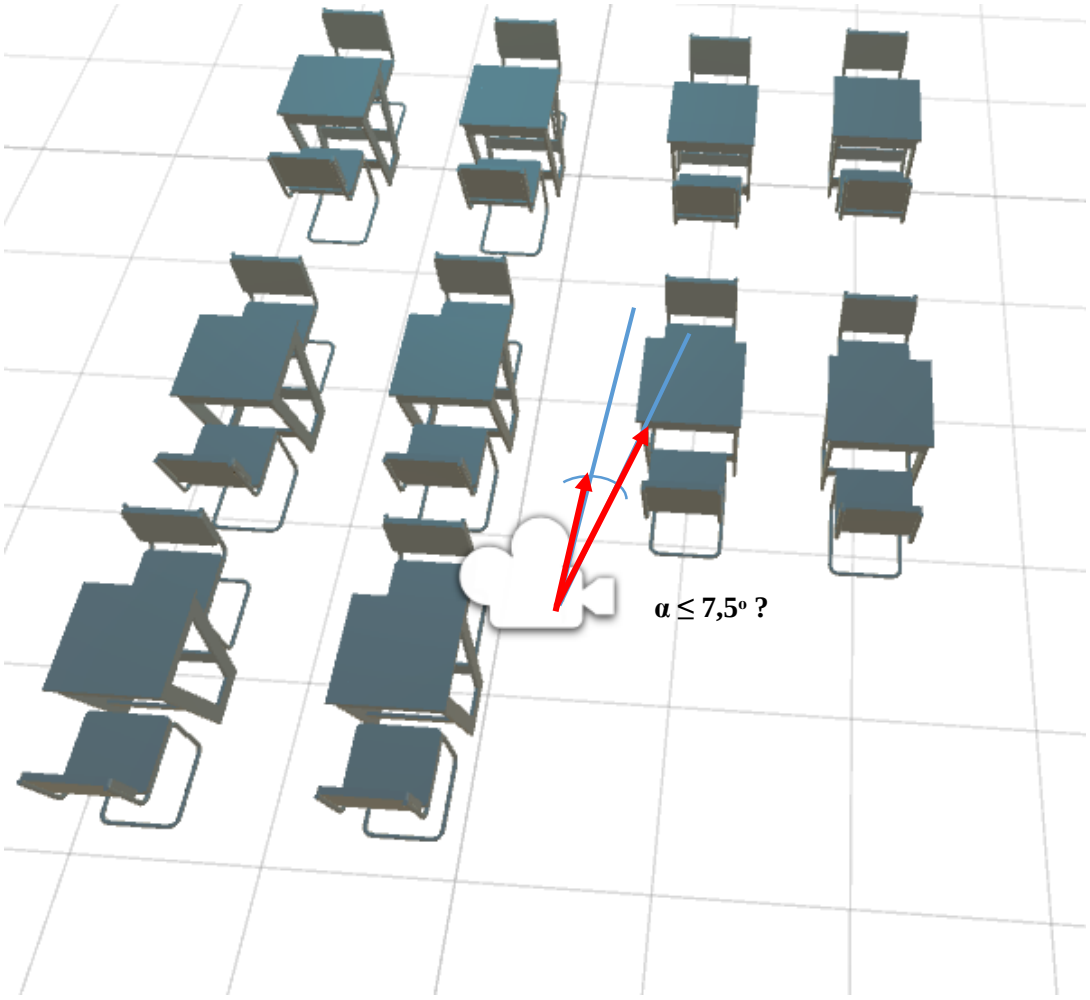


FIG.3.8 - A PARTIR DE 4 OBJETOS NO CAMPO DE VISÃO, α = 7,5º NUMA PRIMEIRA OPERAÇÃO

Se α igual a 7,5º e ainda for possível contabilizar quatro ou mais objetos dentro deste ângulo. O seu valor será novamente divido por dois, atingindo o limite máximo de 3, 25º. A partir deste valor α torna-se imutável.

- Pressionando uma teclada predefinida, o objeto é selecionado e estará pronto ser alvo de edições por parte do utilizador. Ficará assinalado com cor amarela a partir do momento em que se encontre selecionado.

- Para selecionar outro objeto basta premir a tecla destinada ao comando de desseleção para que o primeiro objeto retorne à sua cor original, indicando que não se encontra selecionado. Seguidamente é apropriado mudar a posição do observador para outra mais próxima de um *game object* que se pretenda editar. Deverá ser premida a tecla de seleção para que o novo objeto possa ser manuseado.

2 - Rotação de um objeto sobre si mesmo

O procedimento original que permite a rotação recorrendo aos comandos gestuais do Leap Motion não foi suprimido do projeto. É reconhecido que o seu uso implicará hipoteticamente uma maior habituação em termos de interação com os comandos. Fazer com o que o centro geométrico do objeto que se pretende editar coincida com o próprio eixo de rotação, requer alguma familiarização com a interface do Leap Motion. Neste sentido, foi criada uma opção de recurso que para além de teoricamente mais imediata para o utilizador, permite exercer o comando rotação com maior precisão.

No *script* App, foi redigido um código que assume um vetor com direção vertical, parâmetros x, y, z iguais a 0,1,0 partindo do centro geométrico do objeto selecionado (o parâmetro y corresponde à direção vertical). Este vetor sofre uma rotação a uma velocidade de x m/s por *frame* cada vez que uma determinada tecla preconfigurada é premida.

Durante o desenvolvimento da interface associou-se este código aos botões representados na Fig.3.9.

FIG.3.9 - BOTÕES ASSOCIADOS AO COMANDO DE ROTAÇÃO [42]

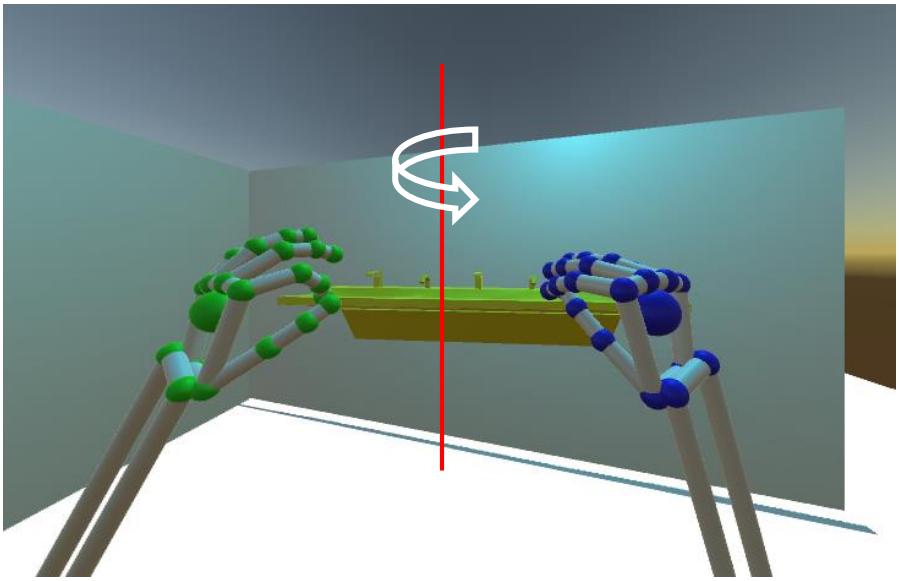


FIG.3.10 – COMANDO DE ROTAÇÃO EXECUTADO ATRAVÉS DO SENSOR LEAP MOTION

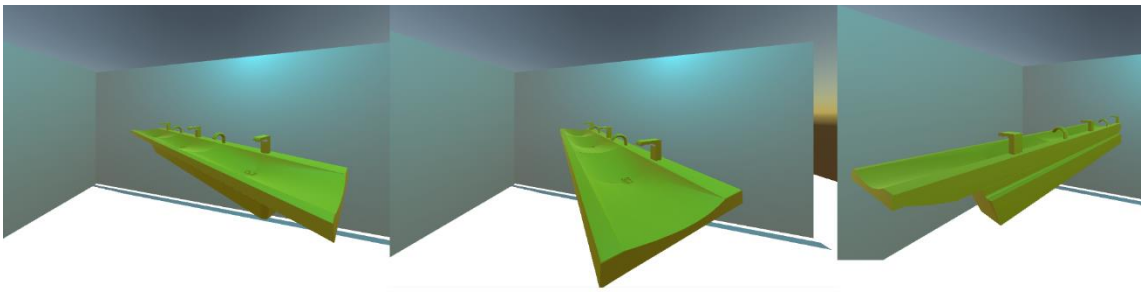


FIG.3.11 – ROTAÇÃO DE UM OBJETO EM TORNO DO SEU EIXO VERTICAL RECORRENDO AO COMANDO DE ROTAÇÃO ATRAVÉS DO *JOYSTICK*

3 - Alterar a escala de um objeto segundo um só eixo ortogonal (eixo xx, por exemplo)

Em determinados casos aumentar a escala de um objeto segundo x, y e z não constitui uma mais valia. Por exemplo, quando se pretende que o comprimento de uma viga aumente até atingir determinado ponto no espaço. O mesmo se aplicaria no caso de um pilar ou porção de tubagem, cujo objetivo passaria por utilizar o comando escala afetando apenas uma direção. Assim, não descartando a função escala do Pinch Movement, que poderá ser igualmente útil para outros cenários que não se revejam nos exemplos supracitados, definiu-se inicialmente um novo comando "escala" afetando uma só direção.

Este novo comando, na mesma lógica da rotação, seria acionado quando o utilizador premisse uma de duas teclas predefinidas no *joystick*, aumentando ou reduzindo a escala de um objeto respetivamente.

Na programação deste comando acedeu-se às propriedades do *game object*, mais concretamente aos seus parâmetros x, y e z associados à escala no referencial do próprio objeto (escala local). Através da escala local é adicionado um vetor com estes três componentes, sendo dois deles, y e z iguais a zero. Este vetor é multiplicado "n" vezes por segundo durante um *frame*. Sempre que uma tecla associada ao comando escala é pressionada, o vetor é somado ou subtraído aos

parâmetros da escala local do objeto selecionado. A subtração ou adição materializam-se no aumento ou diminuição da escala.

Numa fase posterior deste trabalho observou-se a ineficiência do comando escala na tentativa de o executar através de um *plugin* (adiciona uma funcionalidade especial a um programa maior) no Autodesk Revit 2016. Modificar a escala de um objeto, sobre todos os eixos ou numa só direção, não se ajusta à "lógica BIM" do Revit. Cada objeto, neste programa, representa um elemento paramétrico de uma família. Sendo uma família composta por tipos e instâncias. Hierarquicamente surge a seguinte relação: Família → Tipo → Instância.

Uma família é constituída por vários parâmetros. Um parâmetro poderá estar associado à categoria Tipo, ou a uma Instância (um elemento/objeto com particularidades individuais).

Imagine-se o seguinte caso: Pretende-se alterar o parâmetro "altura" das pernas de uma mesa. Com esta simples operação poderão surgir duas possibilidades. Caso a altura seja um parâmetro do objeto mesa, ou seja, Instância, ou o parâmetro altura seja de Tipo.

Na primeira situação seria possível alterar a escala das pernas da mesa sobre uma só direção aumentado ou diminuindo o seu comprimento. O problema surgiria quando o utilizador tentasse selecionar um outro tipo de mesa que não partilhasse a referência "altura". Sendo um parâmetro de Instância, estará alocado a um determinado objeto. Por outras palavras, só seria possível alterar a "altura" das pernas de uma se se tratasse de um objeto igual à instância inicial deste exemplo. Todas os outros elementos mesa estariam excluídos por representarem instâncias diferentes. Como resultado, existiria a obrigatoriedade de criar código específico para cada componente de Instância. Um trabalho moroso e muito pouco ineficaz, dado que o utilizador poderá criar novas famílias dentro de um projeto. Deste modo, novas linhas de código teriam de ser produzidas no sentido de serem identificados os novos parâmetros definidos pelo utilizador.

Uma situação alternativa consiste na alteração de um parâmetro de Tipo. Por exemplo numa parede selecionada pelo utilizador, o comando escala executado poderia alterar a espessura deste elemento. Inicialmente de 20 cm, com a alteração de escala a espessura da parede modificar-se-ia, alterando-se a espessura para 35 cm. Considere-se a situação hipotética de existir uma família no Revit de paredes com espessura de 20 cm. Um objeto que sofra a alteração de escala referida lançaria um erro no *software*, pois teria que pertencer a uma família diferente daquela onde foi criado como instância. Originalmente concebido como um elemento da família paredes com espessura de 20 cm, deixaria de poder continuar a pertencer à mesma.

A alteração de escala é um problema geométrico computacionalmente simples, mas que pode ser inviável num ambiente BIM, onde os objetos não são simples representações gráficas de uma entidade física. Com efeito, neste domínio colocam-se restrições dimensionais que dependem da natureza de cada objeto e que frequentemente impedem o seu redimensionamento livre.

Em programas como o AutoCad também da Autodesk, os elementos representam entidades exclusivamente gráficas, ou seja, constituídos por pontos, linhas, etc. A modificação da escala seria adequada para este tipo de programa, no entanto, o âmbito deste trabalho contempla *software* BIM o que exclui programas como o Autocad. Dadas estas considerações, optou-se por não incluir o comando escala na plataforma final de RV produzida nesta dissertação

4 – Identificação do objeto selecionado

Criada a possibilidade de operar diferentes objetos de um modo independente, surgiu inevitavelmente a necessidade de criar uma distinção visual entre o objeto selecionado e os demais.

No *script* App, existe uma condição que identifica a cor que um objeto deverá assumir caso seja selecionado. Assim, estabeleceu-se código para aceder a um determinado componente de um *game object*, o *renderer* e posteriormente à cor do material. O código define que se pressionada determinada tecla para seleção de um objeto, este irá modificar a sua cor original. O "objeto ativo" seria deste modo representado através da cor amarelo (decretada por código RGB) permitindo uma identificação mais eficaz.

Pressionando a tecla associada ao comando de desseleção fará retornar o elemento selecionado à sua cor original.

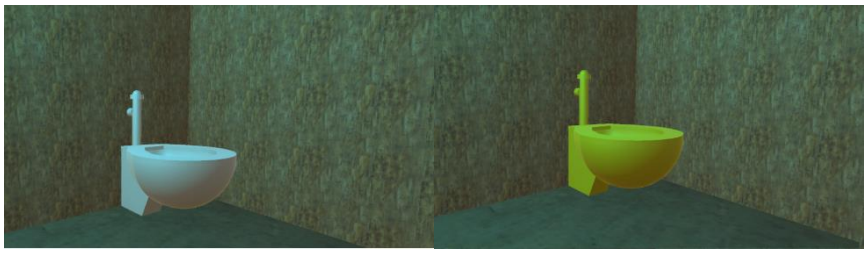


FIG.3.12 – ASPETO VISUAL DE UM OBJETO SEM SELEÇÃO ATIVADA (À ESQUERDA) E APÓS PREMIDA A TECLA DE SELEÇÃO (À DIREITA)

5 - Movimentação da câmara, deslocamento do observador

A movimentação do observador ao longo do modelo importado para o Unity é uma propriedade que confere dinâmica à exploração do modelo. Caso contrário todos os comandos decretados executar-se-iam inevitavelmente a partir de um ponto de vista fixo. Desta forma, as limitações seriam patentes ao nível da perceção espacial, assim como no acesso a objetos mais distantes dessa posição originalmente fixa.

Para ultrapassar esta dificuldade foi produzido um *script* específico para o movimento do observador:

- A mais recente versão do Unity (à data 5.3.4f1) reserva o acesso à componente *transform* do objeto câmara quando se inicia o modo *play* de determinada cena. Neste sentido, o acesso aos parâmetros x, y e z associados à câmara não se realizaram pelo método mais direto: gameObject.transform.position (linguagem de programação c#).

- Associou-se um *script* com diretivas que relacionam determinadas teclas a direções de movimento pela cena.

- Este *script* foi posteriormente adicionado a um objeto virtual ("Big Box") como componente.

- Estabeleceu-se que a posição da câmara seria a mesma que a da "Big Box, o que se traduz num único movimento para os dois objetos sempre que premida uma tecla direcional.

A "Big Box" é hierarquicamente superior à câmara que reside no seu interior. Estes objetos partem da mesma posição inicial e devido ao código de movimento associado ao contentor virtual é possível obter uma deslocação simultânea para ambos. Atuando a "Big Box" como uma caixa virtual, ou seja, não possuí representação física no cenário, o resultado observável é materializado apenas no movimento da câmara.

Todas as soluções apresentadas foram criadas segundo uma limitação primária da API (Orion) utilizada pelo sensor Leap Motion. O uso desta API prendeu-se pela maior compatibilidade com

RV. Contudo, a Orion suprime a possibilidade de personalização/criação de novos comandos gestuais para novas funcionalidades. Em suma, todas as soluções para as restrições anteriormente indicadas, encontraram inicialmente uma dificuldade primária. Não ser possível, de uma forma prática, personalizar novos comandos gestuais reconhecíveis pelo sensor Leap Motion, condicionou o desenvolvimento das estratégias de superação das limitações da versão demonstrativa do Pinch Movement.

Criar comandos gestuais utilizando a API Orion, requer a utilização de *software* adicional e dedicado para reconhecimento gestual. O suporte para este tipo de alternativa é escasso, implicando o dispêndio de tempo considerável com uma abordagem tentativa/erro.

3.3. REGISTO DAS OPERAÇÕES EFETUADAS NO MOTOR DE JOGO

Um dos objetivos mais relevantes foi a prova de conceito que se pretendeu estabelecer: recorrer a comandos simples para edição de objetos através de RV e, em tempo real, verificar as mesmas transformações no modelo BIM.

Para a manipulação de objetos BIM foi feita uma escolha criteriosa por comandos simples, no entanto fulcrais para operar com a geometria dos elementos em cena (importação do modelo para o Unity). A translação, rotação representam operações elementares em edição, sendo comandos de base para uma aplicação mais complexa que se possa elaborar no futuro. O comando cópia surge como complemento, demonstrando a possibilidade de gerar cópias de elementos BIM existente no projeto a partir de uma interação em RV.

As operações apontadas materializam-se em coordenadas espaciais. Translação e rotação fazem parte da componente *transform* de um *game object* e possuem três parâmetros x, y e z. Estes componentes são gravados de forma automática num ficheiro de texto. através de um *script* criado especificamente para o efeito, o "Gravador". Neste código, por cada vez que uma destas operações é executada no ambiente virtual do motor de jogo, ocorre uma gravação num ficheiro de texto (Fig.3.14).

O formato da gravação é representado por linhas, estando em cada uma identificado o objeto (ID), a operação que realizou e as respetivas coordenados espaciais x, y e z. ID representa a identificação de um objeto no Revit. Em concreto, a característica *Element ID* é constituída por uma série de algarismos que permitem identificar e distinguir um objeto no ambiente do Autodesk Revit.

Relativamente à operação (Op.), foi estipulado o seguinte código para leitura dos comandos executados:

Tabela 4 – Operações e simbologia utilizada no ficheiro de registo

Código para translação	Código para rotação	Código para cópia
1	2	3

TABELA 5 – PARÂMETRO GRAVADOS NO FICHEIRO DE TEXTO

Parâmetros gravados		
Translação	Rotação	Cópia
x, y e z	y	

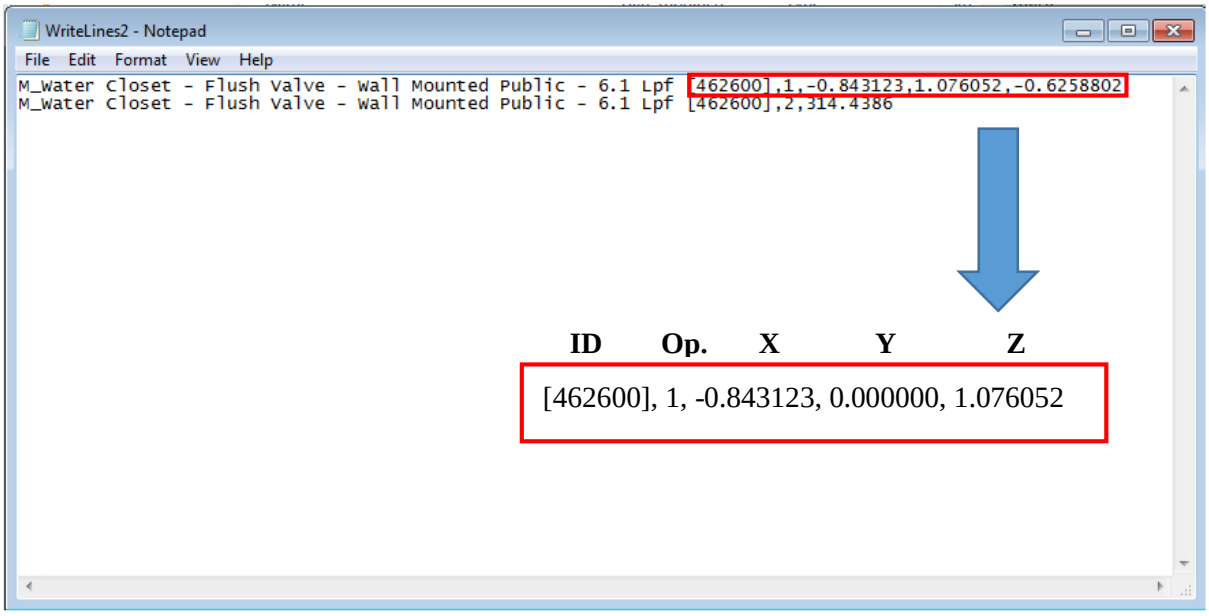


FIG.3.13– EXEMPLO DA INFORMAÇÃO GRAVADA RELATIVA ÀS OPERAÇÕES QUE SERÃO RECONHECIDAS NO AUTODESK REVIT 2016

O comando de cópia consiste na criação de um "clone" do objeto que se encontra selecionado em determinado momento. No ficheiro de texto uma cópia de um objeto é identificada pelo valor negativo do ID referente ao objeto original e pela operação com algarismo 3.

ID original	Op.	Novo ID	X	Y	Z
[-46200],	3,	-1,	-0.01589966,	0.000000,	-0.07406721

FIG.3.14 - ASPETO DA OPERAÇÃO DE CÓPIA REGISTADA NO FICHEIRO DE TEXTO

O parâmetro correspondente ao ID original, refere-se ao ID de um objeto recebido pelo Unity no momento da importação para este programa do ficheiro FBX criado no Revit. Neste ficheiro consta a geometria de todos os objetos do projeto BIM visíveis numa vista 3D personalizada. Nesta vista são selecionados os objetos cuja intenção seja disponibilizar no Unity em formato de cenário.

Na Fig.3.14 o ID 46200 pertence a um objeto originalmente importado para o Unity cuja cópia foi efetuada. O algarismo 3 refere-se à operação (Op.) cópia, e indica que este comando foi ativado pelo utilizador. O parâmetro Novo ID refere-se a uma nova identificação que será atribuída ao objeto copiado. Desta forma, se o *plugin* desenhado para o Revit encontrar uma operação com o parâmetro Op. igual a 3, irá criar um novo objeto a partir da leitura do ID original. X, Y, e Z representam a posição em que o objeto é criado. Este novo elemento terá como ID próprio [-1]. Sempre que o utilizador efetuar uma cópia de um objeto, os seus clones serão identificados com ID's incrementais e únicos: [-1], [-2], [-3], etc.

Uma operação de translação de um objeto copiado será gravada no ficheiro de texto como indicado na Fig.3.15. Sublinha-se o facto destes novos IDs serem negativos. Assim, estes novos IDs atribuídos no ambiente virtual não são os mesmos dos respetivos elementos BIM a criar no Revit. Será apresentado no subcapítulo 3.5 o processo de mapeamento destes IDs negativos para IDs Revit.

ID Op. X Y Z

[-1] , 1 , -0.9543823, 0.000000, 2.046812

FIG.3.15 – OPERAÇÃO DE TRANSLAÇÃO DE UM OBJETO COPIADO

Em geral, o utilizador não terá acesso aos registos de operações lidos pelo *plugin* do Revit, por outras palavras, não terão acesso aos dados gravados no ficheiro texto. Este ficheiro de texto representa informação exclusiva do desenvolvimento da interface, pelo que o seu acesso é restrito. Significa isto que não é relevante, da perspetiva do utilizador, saber o significado de cada um dos parâmetros que regem as linhas do ficheiro de texto.

No Autodesk Revit 2016, o eixo vertical é identificado pelo eixo dos yy. Este fato foi tido em consideração na elaboração do código quer para o Unity como para ao *plugin* do Revit. Pode verificar-se que o *script* Gravador (Unity) regista os parâmetros das operações que ocorrem no motor de jogo, gravando-os segundo as especificações do Revit. Apresenta-se na Fig.3.14, a título de exemplo, o registo de uma translação. O parâmetro Y surge igualado a zero, enquanto o Z se encontra preenchido. Deste modo, conclui-se que o utilizador moveu um objeto na horizontal no Revit, isto é, segundo os eixos x e z.

3.4. APLICAÇÃO DA INTERFACE À ENGENHARIA DE INSTALAÇÕES – REVIT MEP

A interface de RV poderá ser utilizada em conjugação com variados tipos de *software*. A sua arquitetura é flexível, podendo ser ajustada para interagir adequadamente com a API específica de cada programa BIM.

No trabalho desenvolvido optou-se por explorar o *software* Revit 2016 da Autodesk. Representa um dos programas BIM mais utilizados a nível mundial e disponibiliza uma licença de estudante gratuita. O Revit é também a ferramenta BIM mais utilizada para a produção de desenhos segundo o National BIM Report de 2016 no setor da construção do Reino Unido, seguindo o ArchiCAd e o Vectorworks. [43]

Na linguagem BIM, especificamente na terminologia associada ao *software* Revit 2016 da Autodesk, existe um conjunto de disciplinas/especialidades que intervêm nos modelos. MEP (M*echanical Electrical Plumbing*) refere-se ao conjunto de Engenharia de instalações, referindo-se no caso do Revit aos projetos de engenharia mecânica, hidráulica e de sistemas elétricos.

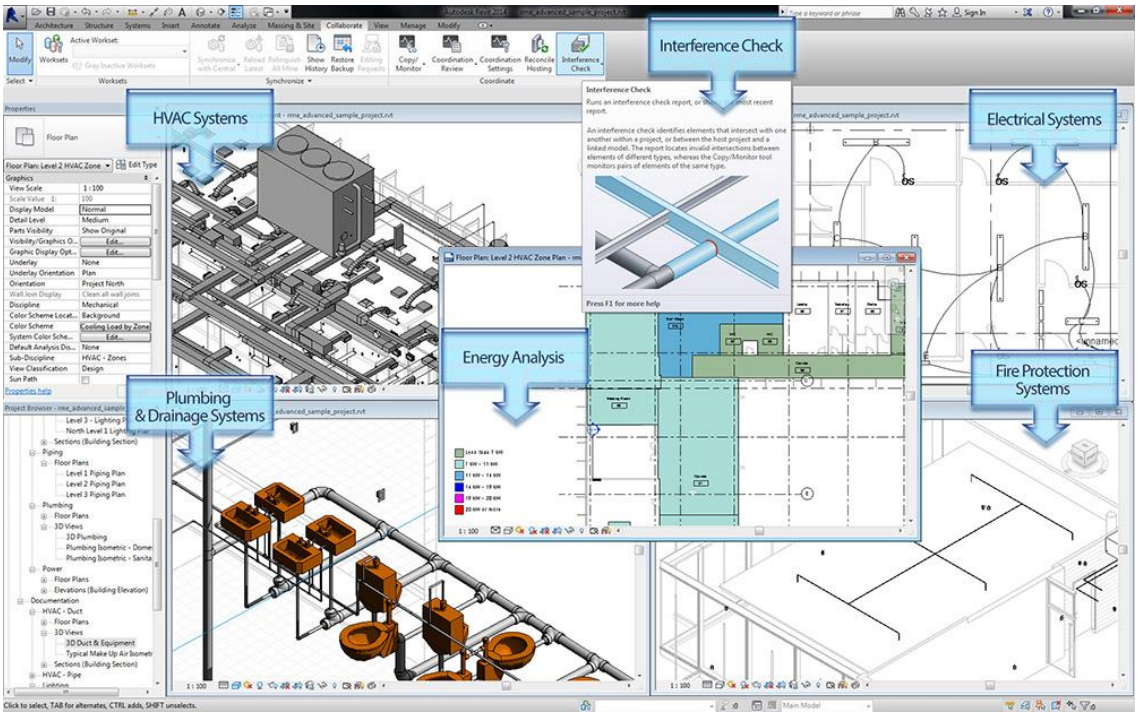


FIG.3.16 – DIFERENTES TIPOS DE PROJETO QUE PODEM SER EXECUTADOS ATRAVÉS DA VERTENTE MEP DO REVIT[44]

Através da interação com os Oculus Rift e o sensor Leap Motion é possível alterar a geometria e posição de um objeto. Foi produzido um *plugin* e acrescentado ao Revit, cuja função reproduz automaticamente as modificações do Unity ao projeto de engenharia ou arquitetura.

Um aspeto relevante do MEP reside na particularidade de os objetos poderem estar ligados entre si, seja por tubagens, peças específicas de ligação como "tês", "cotovelos", etc. Essas ligações surgem quando dois elementos se fixam entre si ativando uma conexão. Esta dependência gera um movimento conjunto entre os objetos conectados. A agregação entre os elementos permite ao mover um só objeto, fazer com que todos os outros ligados ao primeiro se movimentem em conjunto. Na Fig.3.17pode-se constatar o símbolo , referente ao ponto de conexão entre dois objetos. De salientar o diâmetro da tubagem de abastecimento (25 mm) aliado ao símbolo de conexão ( ). Importa referir que para além do movimento em simultâneo, as tubagens poderão modificar o seu comprimento adequando-se à nova localização do objeto a que encontram conectadas.

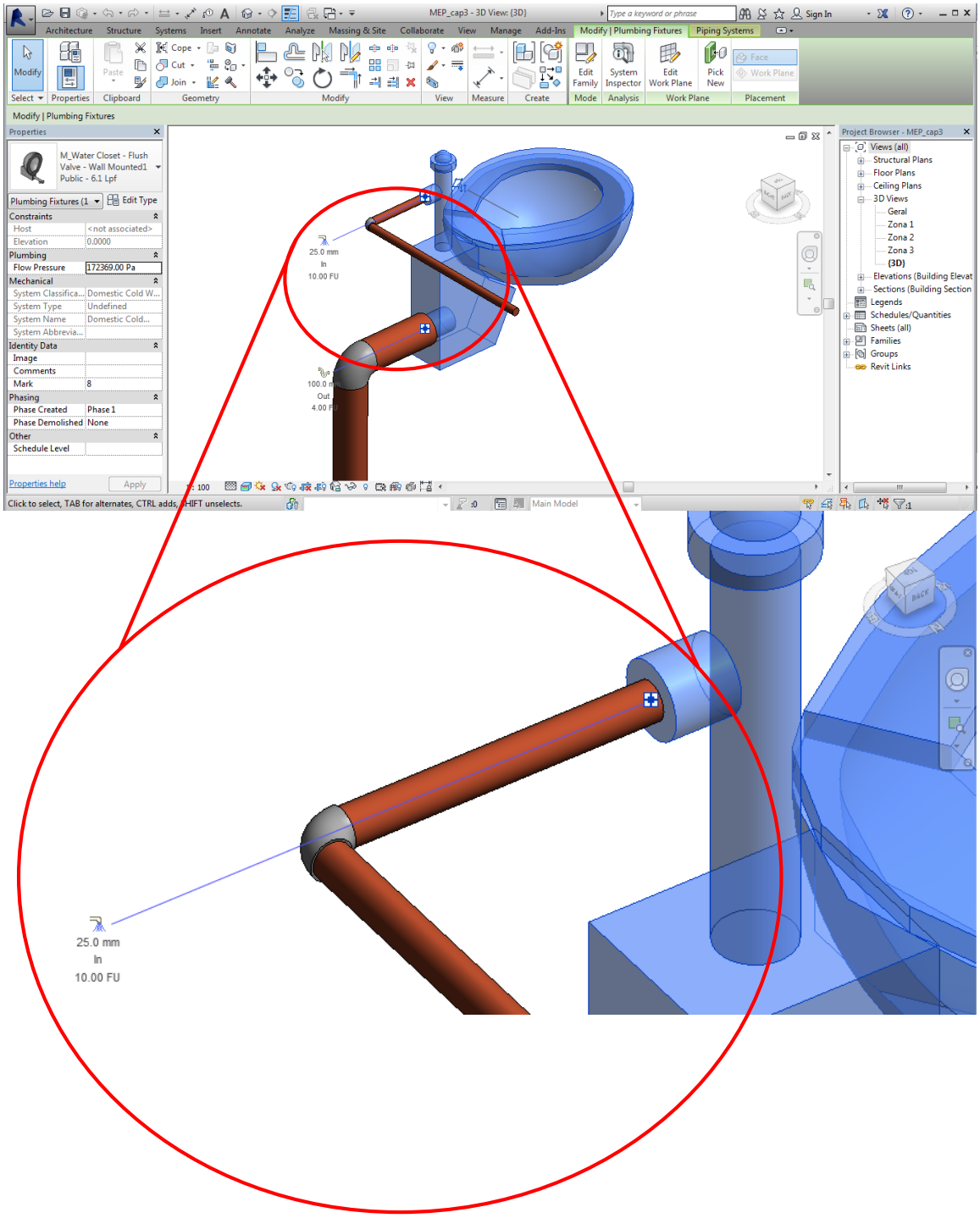


FIG.3.17– PORMENOR DA CONEXÃO ENTRE BACIA DE RETRETE E TUBAGEM

3.5 ELABORAÇÃO DE UM *PLUGIN* PARA O AUTODESK REVIT 2016

A elaboração de um plugin que estabelece a comunicação entre as alterações produzidas nos objetos importados para o Unity e as reproduzisse no Autodesk Revit 2016 foi um processo essencial para completar os objetivos deste trabalho.

Através da linguagem de programação C# produziu-se um código com diversas funcionalidades. Primeiramente foi necessário introduzir no código do *plugin* a localização do ficheiro de texto onde se registaram todas as alterações em objetos produzidas no Unity. Este ficheiro é lido linha por linha, sendo que o Revit reconhece as operações (Op.), como explicitado em 3.3. O ID do elemento é também detetado e, por conseguinte, é efetuada a operação decretada dentro do Revit. Através de C# foi criado um dicionário que associa cada ID negativo, correspondente a uma cópia de um objeto no Unity a um novo ID no Revit. Assim, quando é feita uma cópia de um objeto no Unity, por exemplo, com ID original igual a [4000], este é registado como [-4000] no ficheiro de texto O Revit irá criar uma cópia deste objeto e de imediato é reconhecido no dicionário (*plugin*) que o ID [-4000] será interpretado pelo Revit como um novo ID de valor igual a [-1]. Desta forma, o resultado visível no Revit é uma cópia de um objeto já existente cujo ID corresponde a [-1]. Foi programado um contador para que caso se fizesse uma nova cópia, o ID desta seria [-2] e assim sucessivamente por cada nova cópia de um objeto já existente.

Os comandos de translação e rotação tiverem que ser programados de acordo com a API do Revit. Assim, o comando translação estabelece que caso seja lida no ficheiro de texto uma operação de translação de um objeto, esta será representada no Revit através do vetor deslocamento entre a posição original e a final do objeto em causa. Quanto ao comando de rotação, em primeiro lugar é lido um valor em ângulos representando a rotação em torno de um eixo vertical de um objeto. Este valor é posteriormente reconhecido pelo comando: location.Rotate(axis, operation.Parameters[0] * 2 * Math.PI / 360 * -1), que consiste em detetar a localização do objeto a rodar no Revit, e através de uma linha de referência aplicar a rotação pretendida.

Todas as operações do reconhecidas pelo *plugin* correspondem a gravações feitas em linhas do ficheiro de texto, enviadas através do Unity. Recorreu-se à função Idling da API do Revit que permite que o código do *plugin* seja executado praticamente de forma contínua, desde que nenhum outro comando do Revit esteja a ser invocado. [45] Desta forma, caso sejam escritas novas linhas no ficheiro de texto irá ocorrer uma resposta no Revit, ou seja, o projeto será atualizado com novas posições, rotações ou cópias de objetos.

4

AVALIAÇÃO DA INTERFACE QUANTO AO GRAU DE ADEQUABILIDADE E NATURALIDADE DOS COMANDOS PRODUZIDOS

4.1. PROPOSTA DE CASO DE ESTUDO

Após a criação de uma plataforma que possibilita a utilização de RV e interação com *software* BIM, pretende-se que a mesma possa ser alvo de experimentação e teste. A aplicabilidade da solução desenvolvida deverá ser alvo de utilização para além da experiência decorrente da sua produção.

Pretende-se obter uma resposta através de um caso de estudo testado pelo utilizador final. Assim, embora questionável em termos de representatividade da população alvo, poderão inferir-se cuidadosamente algumas conclusões à cerca da recetividade dos inquiridos face à interação dos comandos programados.

A prova de conceito transposta no formato de caso de estudo resumiu-se à experiência por parte dos utilizadores num modelo BIM, de todos os comandos programados e interações consideradas.

4.2. METODOLOGIA DE PESQUISA

Existem diferentes métodos de pesquisa ou estratégias. Yin (1994) define cinco estratégias comuns para investigação dentro das ciências sociais: pesquisas, métodos experimentais, análise de arquivos, relatos (*histories*, relacionado com o método histórico de pesquisa) e casos de estudos. A fronteira entre cada um destes estilos não é bem definida e Yin (1994) propõe que a determinação do tipo de investigação se debruce sobre três tópicos:

- Tipo de operação de pesquisa, questões de investigação (o quê, como, porquê, etc.);
- Grau de controlo exercido sobre as variáveis envolvidas na investigação;
- Foco da investigação: no passado ou sobre atividades a decorrer no presente.

A seguinte tabela apresenta os requisitos a ter em conta na seleção do estilo de investigação. [46]

TABELA 6 – REQUISITOS PARA A SELEÇÃO DO TIPO DE INVESTIGAÇÃO

Estilo/Estratégia	**Questões de investigação**	**Controlo sobre variáveis independentes**	**Foco sobre atividades**
Pesquisa (*Survey*)	Quem, o quê, onde, quantos, quanto?	Não é um requisito	Contemporâneas
Experimental	Como, porquê?	É um requisito	Contemporâneas
Análise de arquivos	Quem, o quê, onde, quantos, quanto?	Não é requisito	Contemporâneas/passadas
Histórico	Como, porquê?	Não é requisito	Passadas
Caso de estudo	Como, porquê?	Não é requisito	Contemporâneas

De acordo com as opções sugeridas na Tabela 6, poderá identificar-se a estratégia de pesquisa pretendida como um *Caso de estudo*. Os requisitos selecionados basearam-se nas seguintes opções da Tabela 6:

Questões de Investigação:

- Como: Através de um modelo passível de ser explorado pelo utilizador.
- Porquê: Por forma a avaliar o grau de naturalidade com que os comandos e interface criados conseguem ser manuseados.

Controlo sobre variáveis independentes:

No contexto presente, a identificação e definição de variáveis difere consideravelmente do que aconteceria, por exemplo, na identificação da independência ou dependência de variáveis quantitativas.

Por exemplo, determinação do peso de um material:

$$P = m \times g\ (N) \tag{1}$$

Numa pesquisa hipotética com o objetivo de identificação do peso de vários tipos de materiais, uma das fórmulas a utilizar seria (1). Neste caso, o peso (P) em newton, representaria a variável dependente, a aceleração da gravidade (g) a variável ambiental, expressa em $^{m}/_{s^2}$ e, por fim, a massa expressa em kg seria a variável independente.

Contrariamente ao caso apresentado, a "experiência" que se pretende avaliar é do tipo qualitativo. A naturalidade/ intuitividade da interface ou ausência desta, é a variável qualitativa por excelência que será alvo de apreciação por parte dos utilizadores e posteriormente analisada.

Todavia, poderia realizar-se uma tentativa por identificar a relação de dependência entre a naturalidade da interface e o comando em utilização num dado momento. Neste esforço,

"comando utilizado" corresponderia a uma variável independente e, por conseguinte, a "naturalidade da interface" (relativa ao comando) poderia declarar-se como variável dependente.

A identificação do grau de dependência entre estas duas variáveis não produz vantagens diretas em eventuais conclusões deste estudo. É um fato que a naturalidade, entendida como variável, depende inevitavelmente do comando que irá ser testado. Contudo, se se pretendesse avaliar a naturalidade da interface sobre um ponto de vista global, esta relação de dependência desvaneceria. A interface estaria a ser avaliada não como dependente de um, mas do conjunto de comandos testados. Desta forma, assumiu-se como "Não é um requisito" o controlo sobre variáveis independentes na elaboração do caso de estudo.

Foco sobre atividades:

- Contemporâneas.

O caso de estudo remete para a viabilidade e aplicabilidade da interface produzida e para sua relação direta com o utilizador. O foco desta pesquisa interceta a indústria da AEC mediante as aplicações que se poderão gerar através da base desenvolvida neste trabalho.

Contemporaneidade e eventualmente atividade futura da Engenharia Civil, Arquitetura, assim como de todos os seus intervenientes, representam o foco deste caso de estudo.

4.3. CONSTITUIÇÃO E FORMULAÇÃO DO CASO DE ESTUDO

A cada participante neste caso de estudo apresentou-se um modelo do edifício O (cafetaria) da FEUP, dividido em dois espaços. Nestes locais virtuais decorreram os testes para avaliação do grau de interação dos comandos programados. Deste ponto em diante os espaços de teste serão apelidados por "zonas", definindo-se como frações do modelo geral onde os participantes experienciaram e se propuseram ao cumprimento de determinados objetivos.

Neste sentido, o caso de estudo compõe-se por dois momentos, um prático e outro vocacionado para o preenchimento de um questionário. Durante a fase prática utilizaram-se diferentes tipos de *hardware*, que para melhor compreensão do teste serão agora enumerados:

- HMD – Oculus Rift DK2 (*Development Kit 2*). Função de orientação do movimento e visualização da cena em Unity.
- *Joystick* (Logitech Gamepad 310). Aplicação do comando de movimento, seleção, translação, rotação e cópia.
- Sensor de deteção de comandos gestuais – Leap Motion. Deteção do comando *pinch* que permite mover, exercer a rotação e criar cópias de elementos da cena.

O modelo BIM com as respetivas alterações em função do caso de estudo, foi importado para o motor de jogo Unity a partir do Autodesk Revit 2016, via FBX.

Exportar o modelo integral seria incompatível com o intuito deste estudo. Pretende-se que o utilizador possa concentrar-se em realizar determinada tarefa, confinada a um espaço, e não a exploração do modelo. Como consequência da exportação de um modelo menos complexo, com menos informação, o processo de transferência de dados entre programas é agilizado. Neste sentido, a opção por dividir o projeto em zonas revelou-se uma alternativa conveniente.

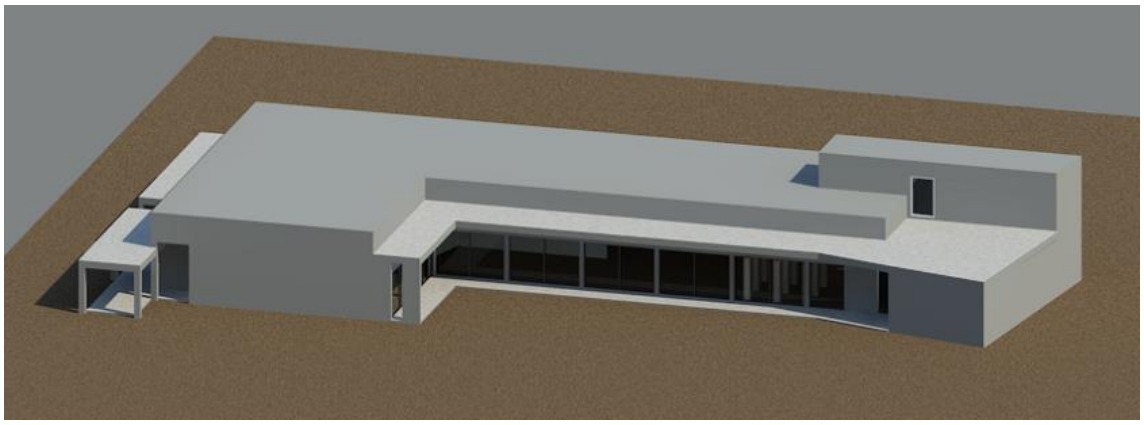


FIG.4.1 - MODELO BIM REALIZADO NO REVIT 2016

4.3.1. ZONA 1

A fase prática do teste inicia-se na zona 1 (Fig.4.2) cuja intenção consiste na familiarização do utilizador com o ambiente em RV. Nesta zona decorrem testes em dois momentos. O primeiro destina-se ao domínio de dois comandos básicos de interação: comando de movimento e comando de seleção (ambos a partir do *joystick*). Para muitos participantes deste caso de estudo, o primeiro contato com RV ocorreu nesta zona 1. Assim, o domínio do movimento pela cena e a capacidade de selecionar individualmente um objeto, representam duas funções fundamentais para a primeira experiência com a interface. O comando de movimento é ativado através do deslocamento do botão analógico esquerdo (ver Fig.4.3). O utilizador movimenta-se pelo cenário mudando de direção consoante o sentido em que a sua cabeça estiver orientada. O comando de seleção é ativado ao premir o botão A, e desativado caso o botão B seja pressionado. O objeto em causa terá a sua cor original modificada para amarelo, indicando que se encontra selecionado.

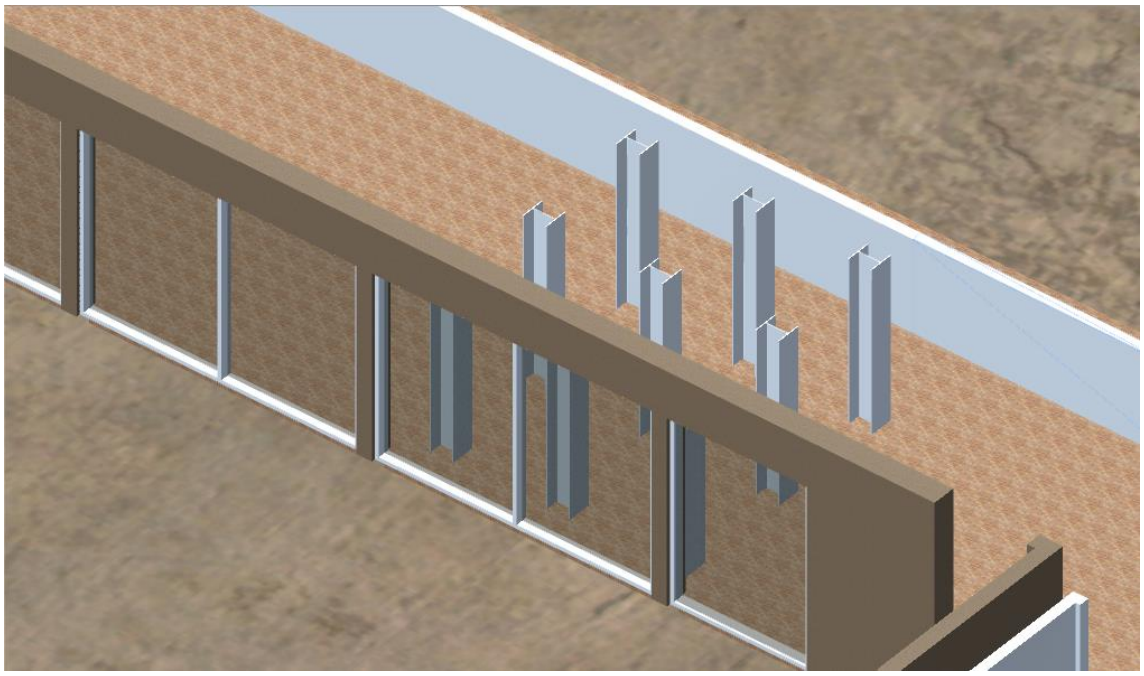


FIG.4.2 - ZONA 1 VISUALIZADA ATRAVÉS DO UNITY

Numa segunda fase os objetivos para a zona 1 renovam-se. Pretende-se que neste segundo momento o utilizador experimente os comandos de translação, rotação e cópia. A translação é ativada de duas formas distintas. A primeira consiste em selecionar um objeto (botão A) e através do botão analógico esquerdo arrastá-lo até à posição desejada. Como alternativa é possível selecionar um objeto pretendido (botão A) e executar um *pinch*. O Leap Motion, caso se coloque uma das mãos em frente do sensor, unindo o polegar ao indicador, deteta que o utilizador executou o comando gestual (*pinch*). De seguida, basta mobilizar a mão numa direção e o objeto selecionado acompanhará o movimento de translação.

O comando de rotação, semelhante ao de movimento, poderá ser ativado por dois meios. Ambos requerem que um objeto seja selecionado através do botão do *joystik* que executa o comando de seleção. Após a escolha do objeto a manusear, o utilizador poderá optar por uma das teclas destinadas ao comando de rotação através do *joystick*. Desta forma, o objeto em causa irá sofrer uma rotação segundo o seu próprio eixo dos zz. O processo alternativo para a execução do comando de rotação, após a seleção de um objeto, tira partido das propriedades do Pinch Move, como referido no Capítulo 3.

Através da deteção de dois comandos *pinch*, o Unity estabelece o ponto médio entre a posição das duas mãos. Em rigor, esta posição é calculada como o ponto médio entre a união de um polegar com um indicador de uma mão (*pinch* da mão esquerda, por exemplo) e o seu correspondente na mão direita (*pinch* detetado com a mão direita). O objeto de cor amarela irá rodar, para esquerda ou direita, atendendo ao deslocamento de uma mão em relação à outra (ver Fig.3.12).

Para copiar um elemento previamente selecionado, será necessário recorrer ao comando cópia. Esta funcionalidade permite que o utilizador crie um número ilimitado de cópias de um objeto que esteja a ser selecionado. O comando é ativado através do botão X do *joystick*. As cópias serão posicionadas no mesmo local onde se encontrar o primeiro elemento em determinado momento. Ou seja, um objeto é selecionado (altera a sua cor para amarelo) e, de seguida, é criada uma cópia do mesmo pressionando o botão X. A cópia estará na posição original do objeto selecionado. Todavia, é possível deslocá-lo (objeto original) tendo em conta que se encontra com a seleção ativada. Deste modo, evita-se a partilha da mesma posição por dois objetos distintos.

FIG.4.3 – COMANDOS ASSOCIADOS A CADA BOTÃO DO *JOYSTICK* [47]

4.3.2. ZONA 2

A zona 2 foi concebida tendo como finalidade um teste integral da interface e das suas funcionalidades. Ou seja, todos os comandos estarão à disposição do utilizador, movimento, seleção translação, rotação e criação de novo objeto. Pretende-se que o participante no caso de estudo percorra cada um dos tópicos indicados:

1. Criar uma cópia do lavatório existe neste espaço
2. Alterar a posição da cópia, colocando-a sobre uma nova parede, para além da que suporta o lavatório original.

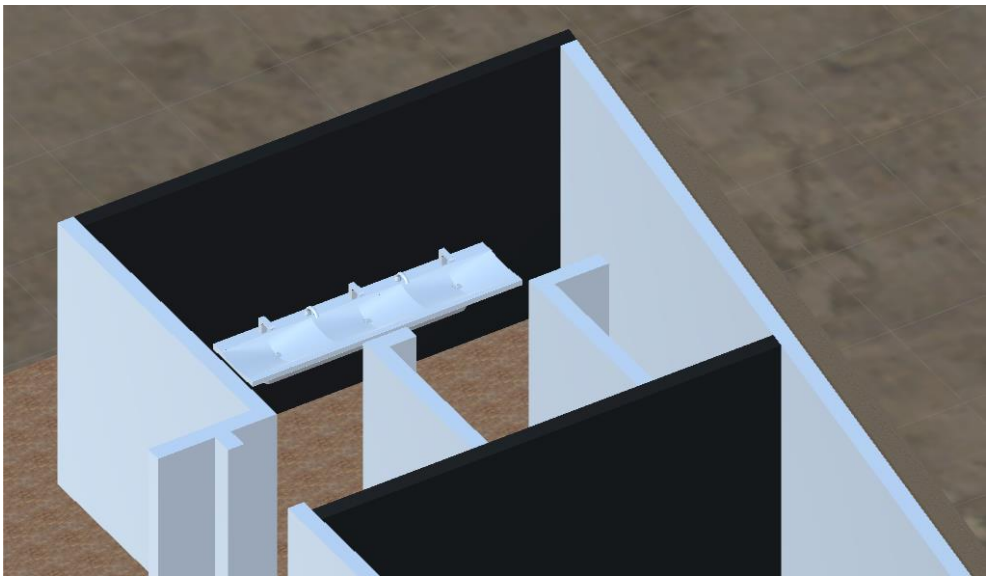


FIG.4.4 – CENÁRIO CORRESPONDENTE À LOCALIZAÇÃO DA ZONA 2

4.3.3. COMPOSIÇÃO DO INQUÉRITO

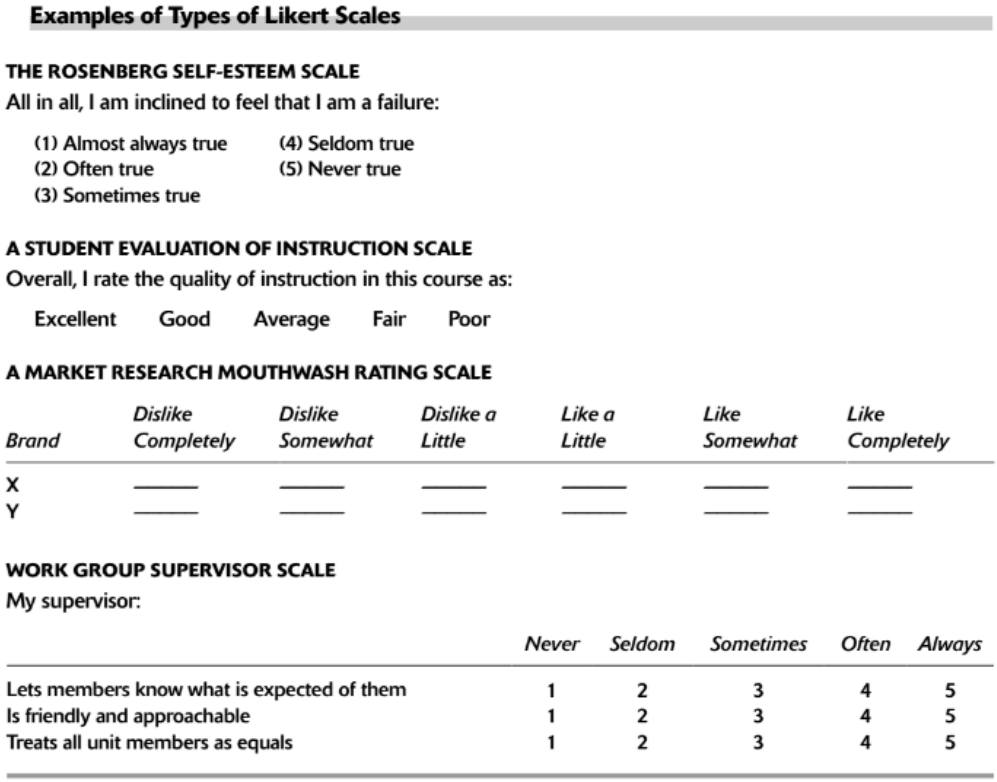
Para além do modelo 3D, foi composto e fornecido um questionário. Na sua formulação respeitaram-se determinados princípios, ou linhas guia, no sentido de o tornar o mais claro possível. O modelo escolhido baseou-se no de resposta fechada, ou seja, disponibilizou-se um conjunto de alternativas de resposta ao inquirido. Este tipo de pergunta agiliza o preenchimento do inquérito tornando o processo mais célere. Todavia, foi considerado que tal modelo de questionário confere riscos. Ao atribuir um conjunto premeditado de opiniões, limita-se artificialmente a resposta por parte do inquirido. [46]

Na zona 1, parte prática do caso de estudo, analisou-se diferentes comandos em duas fases distintas. Cada um destes comandos foi posteriormente avaliado em particular, recorrendo a uma escala métrica não comparativa. Não comparativa significa que cada objeto é avaliado por si só.

Das escalas métricas não comparativas fazem parte as escalas de *items*, amplamente utilizadas em investigação na área da gestão da construção. Possuem diferentes classificações, tais como: escala de Likert, escala diferencial semântica e escala de Stapel. A escala de Likert é aplicada associando uma descrição a um valor correspondente, geralmente numa gama de 1 a 5 valores, embora não haja um número fixo que deva ser cumprido. De modo distinto, a escala de Stapel é geralmente aplicada na vertical com um espetro de valores de -5 a 5, inexistindo ponto neutro (zero). A escala semântica diferencial apresenta semelhanças com a de Likert, contudo remete para 7 valores classificativos. Do ponto de vista semântico, o valor inicial e final seriam diretamente opostos. [43]

O facto de se utilizar uma escala de 4 valores no inquérito do presente caso de estudo revelou alguma ponderação. As escalas ordinais que pretendem obter o grau de concordância/discordância com determinada variável são geralmente aplicadas com 5 a 7 graus classificativos. Na escala de Likert utilizada no inquérito formulado é relativamente comum na investigação para a construção. Contudo, uma gama classificativa com 5 ou 7 pontos poderá gerar a tentação nos inquiridos "por não optar", mais especificamente, por selecionar o valor intermédio. A existência de um ponto intermédio é propensa a este tipo de comportamento, que por vezes poderá prejudicar a pesquisa caso reflita uma resposta negligente ou sem ponderação. Bell (1993) [46]

Assim, os inquiridos serão submetidos a uma avaliação de diferentes objetos de forma individual numa escala de 4 valores (escala de Likert). Estes valores encontram-se acompanhados de uma pequena descrição explicitando a escala de classificação associada a cada algarismo. Neste sentido, os valores são ordenados em termos incrementais, fazendo com que os inquiridos selecionem um grau de preferência que melhor se identifique com o objeto/categoria a ser classificado(a).



Examples of Types of Likert Scales

THE ROSENBERG SELF-ESTEEM SCALE

All in all, I am inclined to feel that I am a failure:

(1) Almost always true (4) Seldom true
(2) Often true (5) Never true
(3) Sometimes true

A STUDENT EVALUATION OF INSTRUCTION SCALE

Overall, I rate the quality of instruction in this course as:

Excellent Good Average Fair Poor

A MARKET RESEARCH MOUTHWASH RATING SCALE

Brand	*Dislike Completely*	*Dislike Somewhat*	*Dislike a Little*	*Like a Little*	*Like Somewhat*	*Like Completely*
X	_____	_____	_____	_____	_____	_____
Y	_____	_____	_____	_____	_____	_____

WORK GROUP SUPERVISOR SCALE

My supervisor:

	Never	*Seldom*	*Sometimes*	*Often*	*Always*
Lets members know what is expected of them	1	2	3	4	5
Is friendly and approachable	1	2	3	4	5
Treats all unit members as equals	1	2	3	4	5

FIG.4.5 – EXEMPLOS DE DIFERENTES TIPOS DE ESCALAS DE LIKERT [48]

TABELA 7 - EXEMPLO DE UMA DAS TABELAS CLASSIFICATIVAS PRESENTES NO INQUÉRITO DO CASO DE ESTUDO BASEADA NA ESCALA DE LIKERT.

Comando	Nada intuitivo	Pouco intuitivo	Requer muito pouco treino	Natural, não necessita de treino
Translação (*joystick*)	—	—	—	—
Rotação (*joystick*)	—	—	—	—
Criação de cópia (*joystick*)	—	—	—	—
Rotação (Leap Motion)	—	—	—	—
Translação (Leap Motion)	—	—	—	—

4.3.4. INVESTIGAÇÃO QUALITATIVA

O objetivo de uma investigação qualitativa não requer a definição de uma amostra representativa de uma população alvo da pesquisa. [48] No estudo de caso referido neste capítulo, pretende-se

encontrar uma resposta por parte dos participantes, conferindo um ponto de vista alternativo na avaliação da interface desenvolvida. Neste sentido, é procurada uma análise qualitativa da prova de conceito criada com este trabalho.

Os inquiridos serão selecionados através de uma amostragem teórica não probabilística. Este método consiste em recolher casos que possivelmente ajudarão a revelar determinadas características teóricas. Por sua vez, estas características poderão demonstrar-se relevantes para determinado tópico da pesquisa.

Os participantes no caso de estudo serão agrupados mediante um conceito chave: Área de formação ou atividade profissional/experiência nas áreas da Engenharia Civil, Arquitetura e/ou Construção e segundo a experiência prévia que possuam, ou não, com equipamentos de RV. Naturalmente, os processos de interação entre *software* BIM e RV produzidos nesta dissertação não se destinam exclusivamente a uma população com experiência no setor da AEC. Pelo contrário, não é requerido qualquer tipo de competência nestas áreas do conhecimento.

A prova de conceito pretende atingir um grau de interação tal, passível de ser manuseado por qualquer utilizador interessado em Engenharia Civil ou Arquitetura, sem qualquer tipo de ligação a estas áreas. O mesmo se aplica à compreensão sobre a metodologia BIM; não é um requisito.

Num estado de desenvolvimento futuro da interface, poderia cogitar-se sobre uma população alvo e um estudo probabilístico. No entanto, dado o estado "pouco mais que embrionário" da aplicação, um estudo deste porte e complexidade não se ajustaria aos objetivos desta fase do trabalho.

Nos anexos do documento encontra-se a versão completa do inquérito utilizado para a análise de resultados.

4.4. ANÁLISE DOS DADOS

O caso de estudo contou com a participação de trinta voluntários cujas limitações de representatividade se reconhecem, podendo todavia, inferir-se algumas noções face à interação alcançada com a plataforma de RV desenvolvida neste trabalho. O olhar crítico dos inquiridos corresponde à apreciação de um utilizador final, e é precisamente essa ótica que importa ressalvar na análise dos dados.

Após a recolha de todos os inquéritos foram criados dois grandes grupos relativos à experiência ou formação dos participantes nas áreas da Engenharia Civil, Arquitetura e Indústria da Construção. Inquiridos em fase de estudos, trabalhadores, ou com formação em algumas destas áreas pertencem a um dos grupos, sendo o restante preenchido com participantes com áreas de formação ou experiência distintas. Como subgrupos dos dois anteriormente referidos, foi criado um critério de seleção que remete para a hipótese de os participantes no caso de estudo já possuírem experiência prévia com equipamento de RV. Pode verificar-se a separação dos inquiridos através da Fig.4.6.

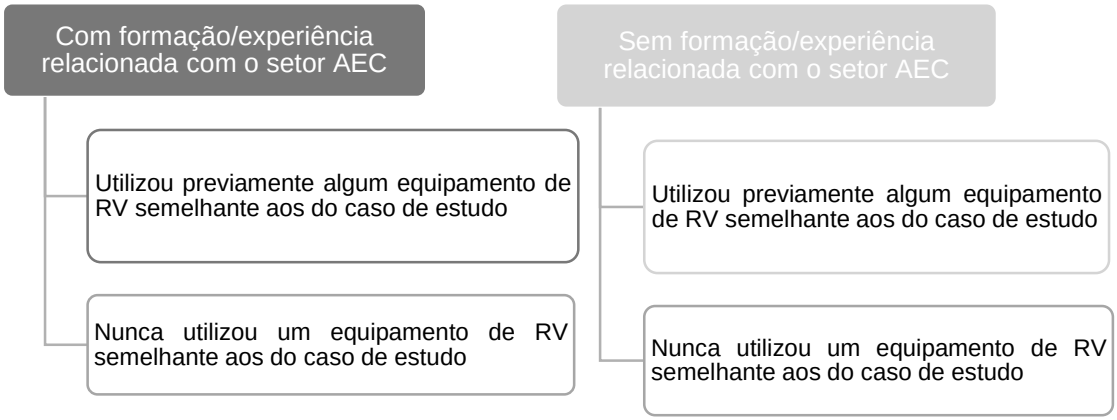


FIG.4.6 – SELEÇÃO DE GRUPOS PARA ANÁLISE DE RESULTADOS

Classificação do grau de intuitividade dos comandos produzidos para RV
Inquiridos:

- Sem formação na área da Engenharia Civil, Arquitetura ou Indústria da Construção;
- Com experiência prévia com equipamentos de RV.

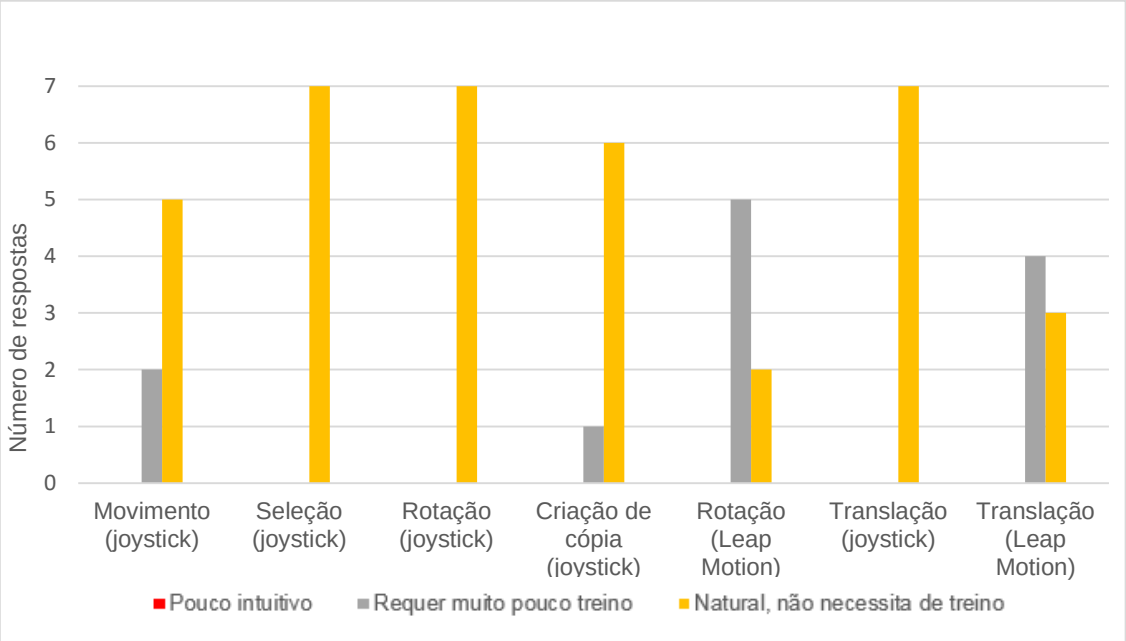


GRÁFICO 1

Inquiridos:

- Sem formação na área da Engenharia Civil, Arquitetura ou Indústria da Construção;
- Sem experiência prévia com equipamentos de RV.

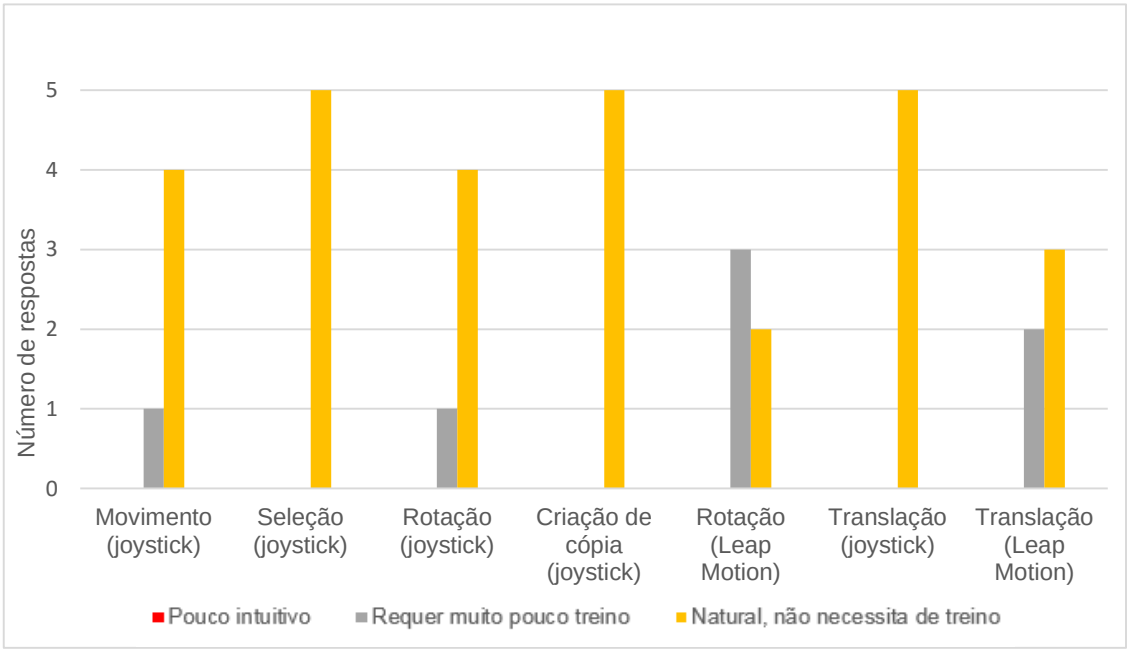


GRÁFICO 2

Inquiridos:

- Com formação na área da Engenharia Civil, Arquitetura ou Indústria da Construção;
- Com experiência prévia com equipamentos de RV

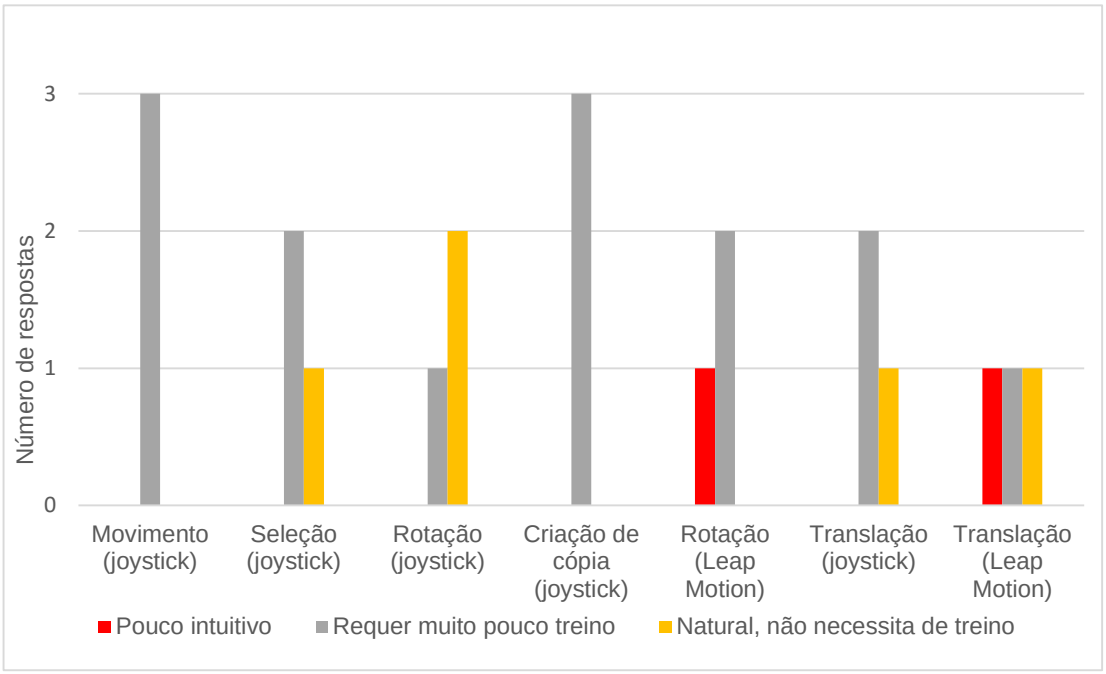


GRÁFICO 3

Inquiridos:

- Com formação na área da Engenharia Civil, Arquitetura ou Indústria da Construção;
- Sem experiência prévia com equipamentos de RV.

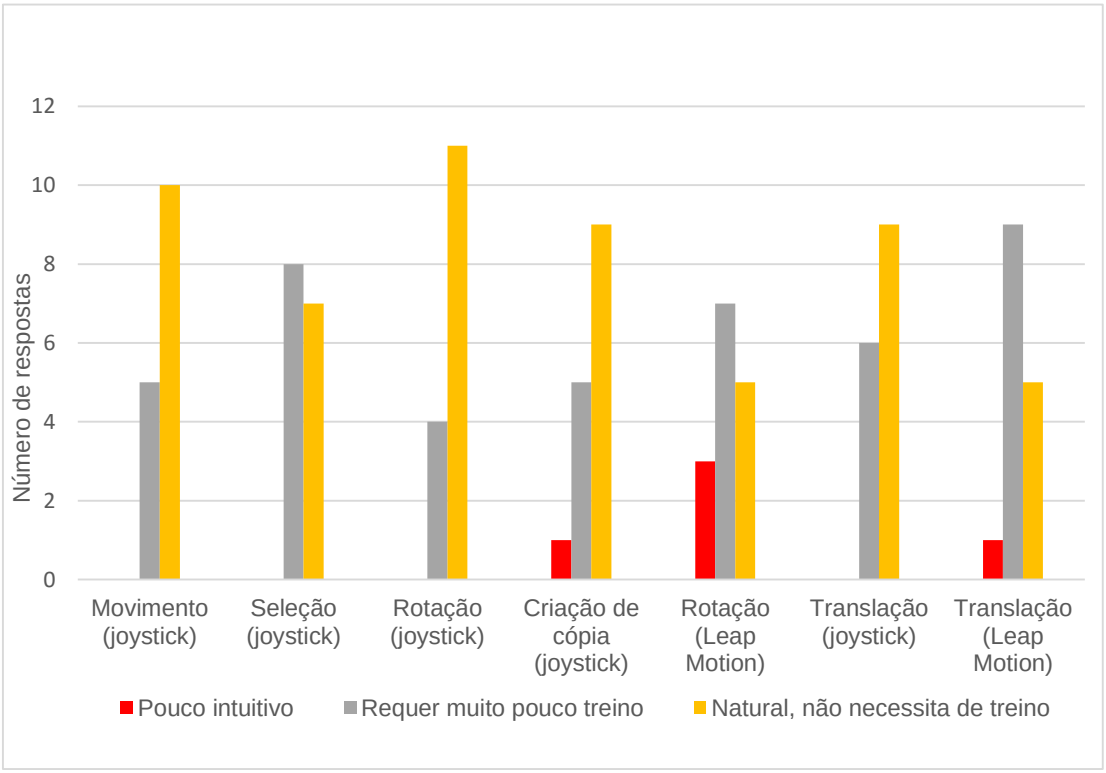


GRÁFICO 4

- Classificação geral da intuitividade de cada um dos comandos desenvolvidos para RV

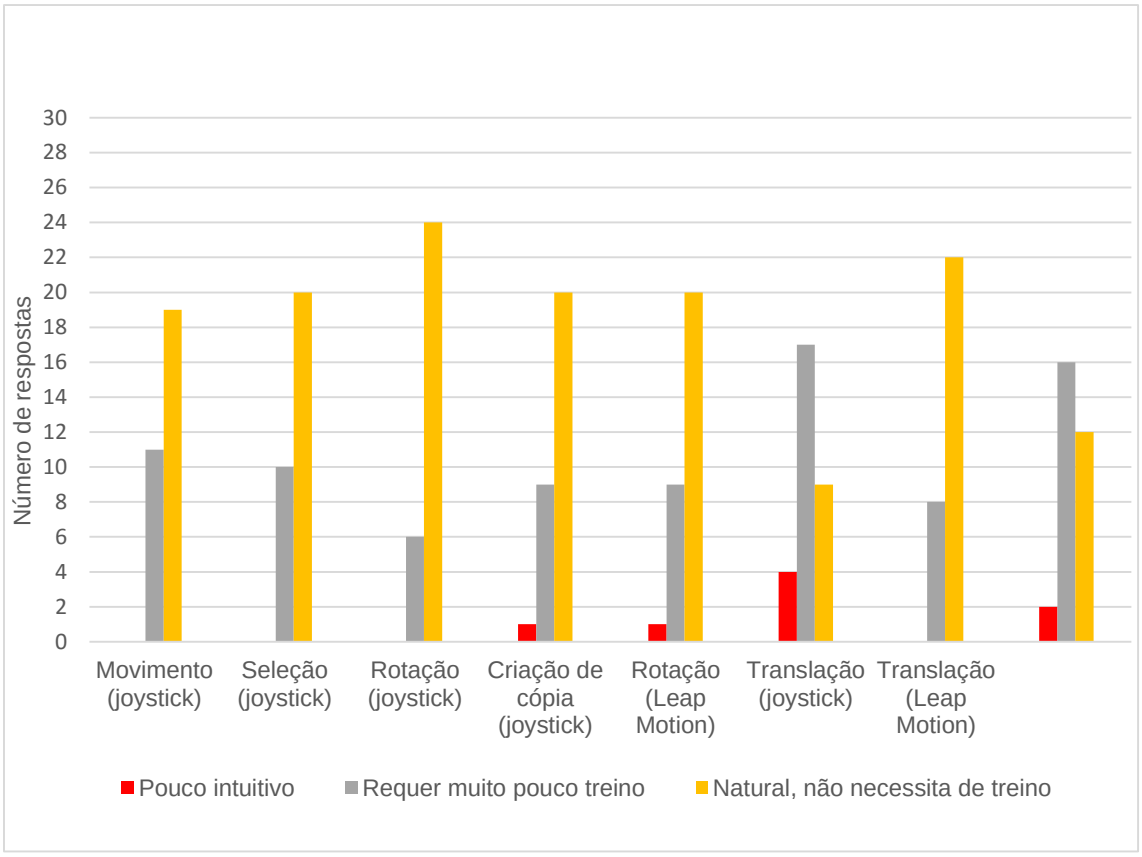


GRÁFICO 5

- Classificação acumulada da interface produzida em RV

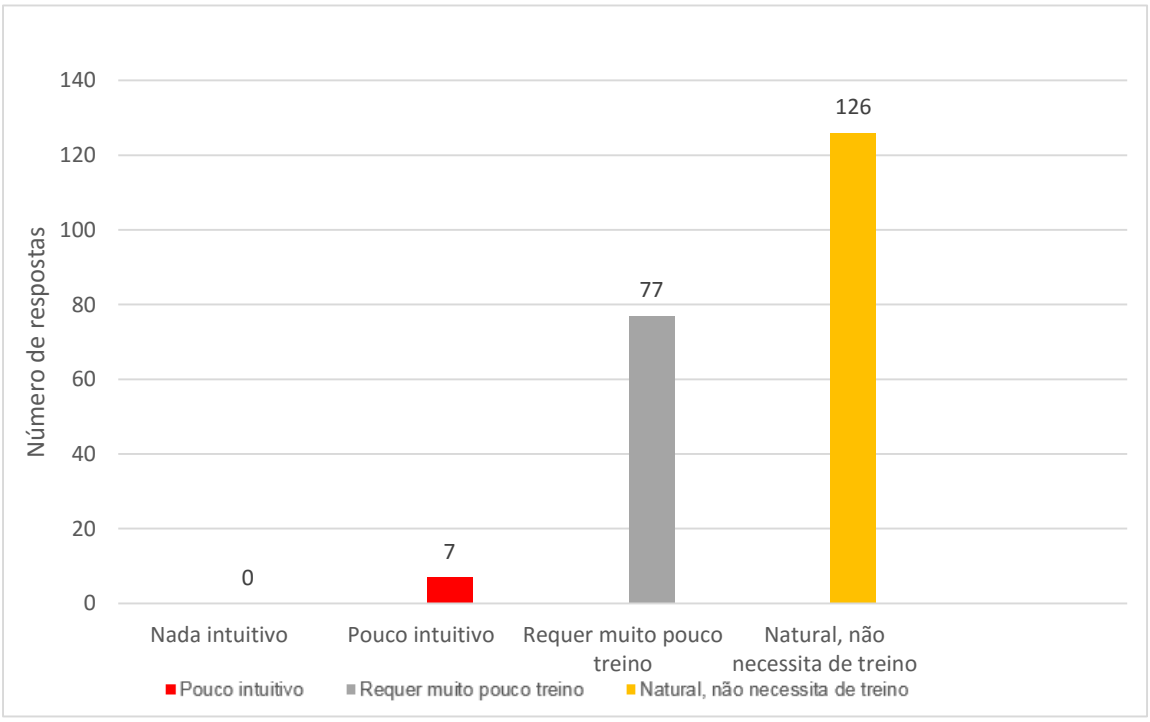


GRÁFICO 6

Um dado imediato concerne o número de participantes em cada grupo. Pode verificar-se que o grupo constituído por inquiridos sem experiência prévia relacionada com o setor AEC (grupo 1) reuniu 12 participantes. 7 deles com utilização prévia de equipamentos de RV semelhantes aos do caso de estudo, os restantes 5 sem familiarização com equipamentos deste tipo em situações anteriores.

Sem formação ou experiência nas áreas da Engenharia Civil (grupo 2), Arquitetura ou Indústria da Construção participaram no total 18 voluntários. Apenas 3 tinham testado equipamentos de RV semelhantes em situações passadas, sendo que os restantes 15 submeteram-se ao caso de estudo sem qualquer tipo de experiência prévia.

Atendendo ao número reduzido de testes realizados, não deixa de ser intrigante o pequeno domínio de inquiridos com formação no setor AEC que revelaram possuir alguma experiência com RV. Somente 3 participantes de um total de 18, comparativamente ao segundo grupo, onde 7 dos 15 participantes assumiu ter tido contato prévio com algum equipamento de RV semelhante aos do teste.

Os comandos reconhecidos como mais intuitivos para o utilizador dentro do grupo 1 com experiência prévia com RV foram os comandos de Seleção e Translação (*joystick*).

Relativamente às opções do grupo 2, a análise dos dados recolhidos indica que os participantes classificaram como mais intuitivo o comando de Rotação (*joystick*). Os inquiridos do grupo 2 com experiência prévia com RV, selecionaram o comando Movimento e Criação de cópia, ambos acessíveis pelo *joystick*. No entanto, como se resumem a 3 inquiridos a pertinência destas escolhas é questionável.

Dos equipamentos de RV utilizados, o comando Movimento (*joystick*) identificou-se com a maioria das preferências dos participantes do caso de estudo, seguindo o comando Translação ativado a partir do sensor Leap Motion. O comando classificado com menor grau de naturalidade foi a Translação de objetos através do *joystick*. (Consultar Gráfico 5)

Como nota final, pode salientar-se o fato de a grande maioria das classificações se ter feito através das opções “Requer pouco treino” e “Natural, não necessita de treino”. A globalidade dos comandos do caso estudo foi classificada com 126 (60%) de total de 210 votos (30 participantes, 7 comandos), com a classificação máxima dos 4 graus da escala (Gráfico 6).

5
CONCLUSÕES

A prova de conceito a que esta dissertação se propôs, procurando processos de interação entre RV e *software* BIM, foi produzida com sucesso e testada publicamente, apesar da falta de experiência iniciais com linguagem C# e Unity.

A interação com RV é, de facto, um modo único de estabelecer diálogo com a máquina computacional e entre diferentes interlocutores com valências igualmente distintas. Através do Oculus Rift e do sensor Leap Motion foi possível observar que utilizadores de diferentes ramos do conhecimento conseguiram operar com *software* BIM sem consciência de que o estavam a realizar. As alterações no modelo em Unity reproduzem-se automaticamente no projeto em Revit, pelo que sem conhecimento prévio de trabalho com esta ferramenta BIM, é possível executar modificações simples. A interface não detém o rigor da ferramenta Autodesk Revit, nem faria sentido dentro dos objetivos do presente trabalho. O rigor do projeto é conferido através de *software* dedicado como o Revit, ArchiCAD, MicrosStation, entre outros. Um os objetivos principais é a evidência de uma modificação no projeto exercida através de RV e visualizada num programa BIM.

As aplicações das interações criadas ao longo deste trabalho poderão abordar diversas situações, tais como:

- Facilitador de diálogo entre os diferentes participantes no processo construtivo;
- Exploração de modelos 3D em detrimento de *renders* muitas vezes com custos superiores;
- Estudo de projetos particulares como hospitais onde a experiência de quem trabalha diariamente nestes espaços é relevante para alterações que possam surgir nos projetos de Engenharia e Arquitetura. O mesmo se aplicaria a hotéis, centros desportivos, museus, etc. A imersão no modelo possibilita a experiência do espaço de uma forma inacessível através de um ecrã 2D.
- Análise de rotas de saída de emergência em planos de segurança contra incêndio;
- Produção de projetos de Arquitetura para utilizadores especiais, por exemplo, com mobilidade reduzida. Através da imersão em RV é possível simular o movimento de uma cadeira de rodas, testando largura de passagens, declives de rampas, altura de elementos construtivos, etc.
- Embora ainda em fase de prova de conceito, é possível alterar um projeto da especialidade de hidráulica de forma automática. Os objetos com conexões de tubagens, caso sofram translações em RV, terão os comprimentos dos tubos, caleiras, etc., alterados dentro do modelo BIM.

- A RV poderá ser utilizada para diferentes propósitos no âmbito da fiscalização de obra, seja através da exploração do projeto com um HMD, até a uma visita virtual ao espaço de implantação através de *drones*.
- No ensino poderão estudar-se novas formas de docência, mais interativas e absorventes para o observador.

A lista de aplicações não se encerra nos tópicos apresentados. De facto, não existe uma limitação de aplicações, tudo depende da interligação que se pretenda estabelecer entre as metodologias de trabalho atuais e as mais valias da RV.

Importa referir que durante o caso de estudo contemplado no capítulo 4, houve entre os 30 participantes um caso de náusea e enjoo provocado pelo chamado *motion sickness*. A habituação à utilização e movimento "dentro" de um modelo através de RV tende a reduzir este efeito. No entanto, as pausas regulares e a qualidade do computador onde interface de RV está a ser utilizada influência no que diz respeito a este inconveniente.

Em suma, o mercado para a RV está em expansão e a Engenharia Civil poderá usufruir de muitas das vantagens deste tipo de tecnologia. Esta dissertação poderá contribuir para o desenvolvimento de interfaces mais complexas onde o leque de possibilidades será uma tarefa difícil de antever.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Ireland, B. 2010 *Despite an increase in the software's adoption, the electrical industry is still experiencing barriers with BIM* http://ecmweb.com/design/growing-pains. Março 2016.

[2] Clemente, J., Cachadinha 2012 N. *Building Information Modeling como ferramenta de visualização de realidade aumentada em obras de reabilitação – Um Caso de Estudo* https://run.unl.pt/bitstream/10362/10013/1/ID57.pdf. Março 2016.

[3] Whyte, J., Bouchlaghem, N., Thorpe, A., McCaffer, R 2000, *From CAD to virtual reality: modelling approaches, data exchange and interactive 3D building design tools* http://www.sciencedirect.com/science/article/pii/S0926580599000126. Março 2016.

[4] Hampton, V., Rubenstone, J., Sawyer, T 2016 *How Video Games Became Design and Construction Tools A new generation is hacking into its childhood toys—and driving better project decisions.* 2016 http://www.enr.com/articles/38932-how-video-games-became-design-and-construction-tools. Março 2016.

[5] http://dpr-review.com/fall-winter-2014/story/3d-mockups-with-virtual-reality-at-vcu-health-system junho 2016.

[6] Sampaio, Alcínia Z., Ferreira, Miguel M., Rosário, Daniel P., Martins, Octávio P.2010 *3D and VR models in Civil Engineering education: Construction, rehabilitation and maintenance* http://www.sciencedirect.com/science/article/pii/S092658051000083X. Março 2016.

[7] Orenstein, D.2016 *VizLab "An ecologist, an architect, and a sociologist walk into a darkened theater..."* http://www.focus.technion.ac.il/Dec13/campus_story1.asp. Março 2016.

[8] https://www.waplag.net/5/2015/06/iii-parapet-stone-border-edging-pmr-config-roof-garden-detail.jpg junho 2016.

[9] http://www.virtualrealitytimes.com/2014/09/30/3d-virtual-sculpting-with-oculus-rift/ junho 2016.

[10] http://sixense.com/ces-2015-highlights maio 2016.

[11] http://sixense.com/makevr maio 2016.

[12] http://meiobit.com/wp-content/uploads/2013/07/20130716fpvrift.jpg maio 2016.

[13] Pereira, I., Nogueira, N. *Realidade Virtual* http://web.ist.utl.pt/ist170613/. Março 2016.

[14] http://gis.wustl.edu/ abril 2016.

[15] https://upload.wikimedia.org/wikipedia/commons/1/1f/Charles_Wheatstone-mirror_stereoscope_XIXc.jpg abril 2016.

[16] http://images.ifanr.cn/wp-content/uploads/2016/05/VPL-Research.jpg abril 2016.

[23]http://images.complex.com/complex/image/upload/t_article_image/ikwxnv91zqot7bcbxuhd.jpg abril 2016.

[18] https://www.pubnub.com/wp-content/uploads/2015/08/leap-motion-3d-motion-gesture-controller-10-large.jpg abril 2016.

[19] Robertson, A., Zelenko, M., Drummond, K., Newton, C., Smith, M., Hamburger, E., Houston, T., Irvine, T., Tieu, U., Lai, R., Lathrop, D., Mazza, C., Schnipper, M., Thonis, S. *The Rise and Fall and Rise of Virtual Reality* http://www.theverge.com/a/virtual-reality março 2016.

[20] http://lexicon.ft.com/Term?term=augmented-reality abril 2016.

[21] Chi, Hung-Lin, Kang, Shih-Chung, Wang, Xiangyu 2013 *Research trends and opportunities of augmented reality applications in architecture, engineering, and construction* http://dx.doi.org/10.1016/j.autcon.2012.12.017 fevereiro 2016.

[22] Wilson,M. 2015 *The Layman's Guide To Virtual* Reality http://www.fastcodesign.com/3042928/the-laymans-guide-to-virtual-reality Março 2016

[23] Nuñez, M.2015 *How it works: The Oculus Rift* http://www.popsci.com/oculus-rift-how-it-works março 2016.

[24] Rosenberg 2014 A. *Sony's long-standing interest in virtual reality gives Project Morpheus an edge http://www.digitaltrends.com/gaming/project-morpheus-sony-interview-gdc-2014/*. Março 2016.

[25] Yoshida, S.2015 *GDC 2015 – Project Morpheus Update The latest news from GDC* https://blog.eu.playstation.com/2015/03/03/gdc-2015-project-morpheus-update/. Março 2016.

[26] Lamkin, P.2016 *Sony PlayStation VR: Essential guide to the hardware, games, release and more* http://www.wareable.com/project-morpheus/sony-project-morpheus-release-date-price-games. Março 2016.

[27] http://media.bestofmicro.com/Y/I/571482/gallery/14-vive-parts-Edit-developed-fixed2_w_600.jpg abril 2016.

[28] Charara 2016, S. *HTC Vive first impressions: Gaming on the second-gen Vive Pre headset*

We strap on HTC's second-gen headset http://www.wareable.com/vr/htc-vive-review. Março 2016.

[29] https://www.wired.com/wp-content/uploads/2015/09/Image_Samsung-Gear-VR_R-Perspective.jpg abril 2016.

[30] https://www.google.pt/search?q=zeiss+vr&client=firefox-b-ab&biw=1366&bih=659&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjLo8WD5ozOAhVF7BQKHT9ABc4Q_AUIBigB#imgrc=LphbrIIH8D4GDM%3A abril 2016.

[31]https://heavyeditorial.files.wordpress.com/2015/07/fove.jpg?quality=65&strip=all&strip=all abril 2016.

[32] https://i.ytimg.com/vi/Q53RtnJpouA/maxresdefault.jpg abril 2016.

[33] http://cdn.mos.cms.futurecdn.net/rBLkyPkiF4dc5M2ebdAq55-650-80.jpg abril 2016.

[34]https://s.aolcdn.com/hss/storage/midas/9ac754eb3bbd61776cad03db2f94a1c/203480239/google-cardboard-store.jpg abril 2016.

[35] http://nerdpai.com/motion-sickness-cinetose-nerd-pai-explica-3/ junho 2016.

[36] Smith, W. 2015 *Stop Calling Google Cardboard's 360-Degree Videos 'VR'* http://www.wired.com/2015/11/360-video-isnt-virtual-reality/. Março 2016.

[37] Berkley,J. 2003 *Haptic Devices* http://www.hitl.washington.edu/people/tfurness/courses/inde543/READINGS-03/BERKLEY/White%20Paper%20-%20Haptic%20Devices.pdf. Junho 2016.

[38] http://www.autotrends.org/2010/05/21/ford-showcases-its-technical-prowess/ abril 2016.

[39] Monteiro, M. *Desenvolvimento de interfaces tridimensionais para aplicações móveis a partir e tecnologia BIM* Dissertação de Mestrado Integrado em Engenharia Civil, Faculdade de Engenharia da Universidade do Porto, 2013.

[40] http://archvirtual.com/2014/01/19/bim-goes-virtual-oculus-rift-and-virtual-reality-take-architectural-visualization-to-the-next-level/ junho 2016.

[41] https://developer.leapmotion.com/gallery/pinch-move abril 2016.

[42]http://gaming.logitech.com/assets/47832/10/f310-gaming-gamepad-images.png abril 2016.

[43] https://www.thenbs.com/knowledge/national-bim-report-2016 junho2016.

[44] http://www.formwelkin.com/revitmep/?lang=en junho 2016.

[45] http://thebuildingcoder.typepad.com/blog/2010/04/idling-event.html junho 2016.

[46] Fellows, R., Liu, A. *Research Methods for Construction* Blackwell Publishing Ltd, West Sussex, 2008.

[47] http://ecx.images-amazon.com/images/I/91RsGVBf1IL._SL1500_.jpg junho 2016.

[48] Neuman, L. *Social Research Methods: Qualitative and Quantitative Approaches* Pearson Education Limited, Edinburgh Gate, 2014.

ANEXOS

ANEXO I

Script mais relevante utilizado no Unity

Nome: LeapRTS.cs

Objetivo: Identificação da "mão esquerda e direita" (respetivamente Leap Hand Controller Left e Right) e define ainda como cada utilizador interage com o comando de rotação, translação e escala (Leap Motion).

Linguagem de programação: C#.

```
using UnityEngine;

namespace Leap.Unity.PinchUtility {

  /// <summary>
  /// Usar este script para permitir que um objeto posa controlado pelo
comando Pinch do sensor Leap Motion
  /// Permite alterar: rotação , translação e escala de um objeto

  public class LeapRTS : MonoBehaviour {

    public enum RotationMethod {
      None,
      Single,
      Full
    }

    [SerializeField]
    private LeapPinchDetector _pinchDetectorA;

    [SerializeField]
    private LeapPinchDetector _pinchDetectorB;

    [SerializeField]
    private RotationMethod _oneHandedRotationMethod;

    [SerializeField]
    private RotationMethod _twoHandedRotationMethod;

    [SerializeField]
    private bool _allowScale = true;

    [Header("GUI Options")]
    [SerializeField]
    private KeyCode _toggleGuiState = KeyCode.None;

    [SerializeField]
    private bool _showGUI = true;

    private Transform _anchor;

    private float _defaultNearClip;

    void Awake() {
      if (_pinchDetectorA == null || _pinchDetectorB == null) {
```

```
        Debug.LogWarning("Both Pinch Detectors of the LeapRTS component must
be assigned. This component has been disabled.");
        enabled = false;
      }

      GameObject pinchControl = new GameObject("RTS Anchor");
      _anchor = pinchControl.transform;
      _anchor.transform.parent = transform.parent;
      transform.parent = _anchor;
    }

    void Update() {
      if (Input.GetKeyDown(_toggleGuiState)) {
        _showGUI = !_showGUI;
      }

      bool didUpdate = false;
      didUpdate |= _pinchDetectorA.DidChangeFromLastFrame;
      didUpdate |= _pinchDetectorB.DidChangeFromLastFrame;

      if (didUpdate) {
        transform.SetParent(null, true);
      }

      if (_pinchDetectorA.IsPinching && _pinchDetectorB.IsPinching) {
        transformDoubleAnchor();
      } else if (_pinchDetectorA.IsPinching) {
        transformSingleAnchor(_pinchDetectorA);
      } else if (_pinchDetectorB.IsPinching) {
        transformSingleAnchor(_pinchDetectorB);
      }

      if (didUpdate) {
        transform.SetParent(_anchor, true);
      }
    }

    void OnGUI() {
      if (_showGUI) {
        GUILayout.Label("One Handed Settings");
        doRotationMethodGUI(ref _oneHandedRotationMethod);
        GUILayout.Label("Two Handed Settings");
        doRotationMethodGUI(ref _twoHandedRotationMethod);
        _allowScale = GUILayout.Toggle(_allowScale, "Allow Two Handed
Scale");
      }
    }

    private void doRotationMethodGUI(ref RotationMethod rotationMethod) {
      GUILayout.BeginHorizontal();

      GUI.color = rotationMethod == RotationMethod.None ? Color.green :
Color.white;
      if (GUILayout.Button("No Rotation")) {
        rotationMethod = RotationMethod.None;
      }

      GUI.color = rotationMethod == RotationMethod.Single ? Color.green :
Color.white;
```

```
      if (GUILayout.Button("Single Axis")) {
        rotationMethod = RotationMethod.Single;
      }

      GUI.color = rotationMethod == RotationMethod.Full ? Color.green :
Color.white;
      if (GUILayout.Button("Full Rotation")) {
        rotationMethod = RotationMethod.Full;
      }

      GUI.color = Color.white;

      GUILayout.EndHorizontal();
    }

    private void transformDoubleAnchor() {
      _anchor.position = (_pinchDetectorA.Position +
_pinchDetectorB.Position) / 2.0f;

      switch (_twoHandedRotationMethod) {
        case RotationMethod.None:
          break;
        case RotationMethod.Single:
          Vector3 p = _pinchDetectorA.Position;
          p.y = _anchor.position.y;
          _anchor.LookAt(p);
          break;
        case RotationMethod.Full:
          Quaternion pp = Quaternion.Lerp(_pinchDetectorA.Rotation,
_pinchDetectorB.Rotation, 0.5f);
          Vector3 u = pp * Vector3.up;
          _anchor.LookAt(_pinchDetectorA.Position, u);
          break;
      }

      if (_allowScale) {
                        _anchor.localScale = Vector3.right *
Vector3.Distance(_pinchDetectorA.Position, _pinchDetectorB.Position);
      }
    }

    private void transformSingleAnchor(LeapPinchDetector singlePinch) {
      _anchor.position = singlePinch.Position;

      switch (_oneHandedRotationMethod) {
        case RotationMethod.None:
          break;
        case RotationMethod.Single:
          Vector3 p = singlePinch.Rotation * Vector3.right;
          p.y = _anchor.position.y;
          _anchor.LookAt(p);
          break;
        case RotationMethod.Full:
          _anchor.rotation = singlePinch.Rotation;
          break;
      }

      _anchor.localScale = Vector3.one;
    }
  }
```

ANEXO II

Scripts criados para interação no Unity

Nome: Gravador.cs

Objetivo:. Estabelece que comandos são gravados no ficheiro e em que formato para poderem ser posteriormente interpretados pelo *plugin* do Autodesk Revit 2016.

Linguagem de programação: C#.

```
using UnityEngine;
using System.Collections;
using System.IO;
using System;
using System.Linq;
using System.Collections.Generic;

namespace Scripts{
      public class Gravador : MonoBehaviour {

            public static string fileName = (@"C:\Users\Fabio Dinis
\Desktop\Dissertação\WriteLines2.txt");

            public static void SaveToFile (GameObject gameObject, Vector3
gameObjectInitialPosition){

                  Debug.Log ("Starting saving file!");

                  StreamWriter writer= new StreamWriter(Gravador.fileName,
true);
                  Vector3 _position = gameObject.transform.position -
gameObjectInitialPosition;
                  string formatedPosition =
string.Format("{0},{1},{2},{3},{4}",  gameObject.name, "1", _position.x,
_position.y, _position.z);
                  writer.WriteLine(formatedPosition);

                  Vector3 _rotation =
gameObject.transform.rotation.eulerAngles;
                  string formatedRotation = string.Format("{0},{1},{2}",
gameObject.name, "2",  _rotation.y);
                  writer.WriteLine(formatedRotation);

                  writer.Close ();
                  Debug.Log ("Finished!!!");
            }

            public static void SaveClone (GameObject newObject, string
originalName, Vector3 gameObjectInitialPosition){

                  StreamWriter writer= new StreamWriter(Gravador.fileName,
true);
                  Vector3 _clone_position = newObject.transform.position -
gameObjectInitialPosition;

                  string objName = newObject.name.Split
('[')[1].Split(']')[0];
```

```
                string cloneId = string.Format("{0},{1},{2},{3},{4},{5}",
originalName, "3", objName, _clone_position.x, _clone_position.y,
_clone_position.z);
                writer.WriteLine(cloneId);

                Vector3 _rotation =
newObject.transform.rotation.eulerAngles;
                string formatedRotation = string.Format("{0},{1},{2}",
newObject.name, "2",  _rotation.y);
                writer.WriteLine(formatedRotation);
                writer.Close ();
            }


      }
}
```

Nome: App.cs

Objetivo: O *script* App atua sobre várias das limitações patentes no Pinch Movement descritas em pormenor no Capítulo 3, em 3.2.1.

Linguagem de programação: C#.

```
using UnityEngine;
using System.Collections;
using Leap.Unity.PinchUtility;
using System.Collections.Generic;
using System;

namespace Scripts{
      public class App : MonoBehaviour {

            //Aplica o script LeapRTS ao objecto com a tag BIM mais próximo.

            private LeapRTS my_leaprts;
                  private GameObject RtsContainer;

            private GameObject activeObject;
            private Vector3 activeObjectInitialPosition;

            private bool waiting_for_link;

            private static int defaultAngle = 15;
            private static int lastClonedId = 0;
            private int tooManyObjectsInSight;

            Color originalColor;



            void Start() {

                  Debug.Log ("Starting nearest_BIM Script");

                  message = GameObject.FindGameObjectWithTag("Message");
                  message.SetActive(false);
                  waiting_for_link = true;

                  RtsContainer =
GameObject.FindGameObjectWithTag("RtsContainer");
                  activeObject = null;

                  Convert.ToInt32(GameObject.FindGameObjectsWithTag
("BIM").Length * .5);

                        tooManyObjectsInSight = 20;
```

```
                Debug.Log ("FOUND CONTAINER!: ");
                Debug.Log (RtsContainer.name);
            }

            void Update() {

                if (waiting_for_link == true) {
                    activeObject = NearestObjectWithTag ("BIM");

                    if (activeObject != null) {



                    if  (Input.GetKey(KeyCode.JoystickButton0)) {
                        Debug.Log ("Selected!");


                                    activeObjectInitialPosition =
activeObject.transform.position;

                                    // Todas as transformaçẽs de RTS
Container
                                    activeObject.transform.SetParent
(RtsContainer.transform);

                                    //Cor
                                    originalColor =
activeObject.gameObject.GetComponent<Renderer>().material.color;

      activeObject.gameObject.GetComponent<Renderer>().material.color = new
Color(1f, 0.92f, 0.016f, -0.5f);
                                    Debug.Log
(activeObject.gameObject.GetComponent<Renderer> ().material.color.a);

                                    waiting_for_link = false;
                                    message.SetActive (false);
                                }
                            }
                }


                if (waiting_for_link == false) {

                    //clone

                    if  (Input.GetKeyUp(KeyCode.JoystickButton2)) {
                        Vector3 myVector = new Vector3 (0, 0, 0.15f);
                        GameObject newObject =
(GameObject)UnityEngine.Object.Instantiate (activeObject,
activeObject.transform.position + myVector, Quaternion.Euler(270,90,90));

      newObject.GetComponent<Renderer>().material.color = originalColor;

                        // Mudar o nome do objeto copiado (ID)
                        string objName = newObject.name.Split
('[')[1].Split(']')[0];
                        objName = string.Format("[{0}]", (-1 *
Math.Abs(Int32.Parse (objName))).ToString ());
```

```
                              lastClonedId--;
                              newObject.name = string.Format("[{0}]",
lastClonedId);

                              Gravador.SaveClone (newObject, objName,
activeObjectInitialPosition);
                        }



                        //Rotação em torno do eixo dos yy
                        if  (Input.GetKey(KeyCode.JoystickButton4)){

      activeObject.transform.RotateAround(activeObject.transform.position,
Vector3.up, 50 * Time.deltaTime);
                        }

                        if  (Input.GetKey(KeyCode.JoystickButton5)){

      activeObject.transform.RotateAround(activeObject.transform.position,
Vector3.up, 50 * Time.deltaTime*-1);
                        }

                        //Desselecionar objeto

                        if  (Input.GetKey(KeyCode.JoystickButton1)) {
                              Debug.Log ("Zero Pressed UP!");

      activeObject.GetComponent<Renderer>().material.color = originalColor;

                              activeObject.transform.SetParent (null);


                              Gravador.SaveToFile (activeObject,
activeObjectInitialPosition);

                              waiting_for_link = true;

                        }

                  }
            }
            GameObject[] ObjectsInSight (GameObject[] tagObjects)
            {
                  return ObjectsInSight (tagObjects, defaultAngle);
            }

            GameObject[] ObjectsInSight (GameObject[] tagObjects, int
defaultAngle)
            {
                  List<GameObject> filteredObjects = new List<GameObject>();

                  foreach(GameObject TagObject in tagObjects) {

                        var objectPosition = gameObject.transform.position;
                        // Vector2 dir = new Vector2(.x,
gameObject.transform.position.z);
                        var vectorCameraToObject = (objectPosition +
Camera.main.transform.position);
```

```
                        float angle = Vector2.Angle(vectorCameraToObject,
Camera.main.transform.position);
                        //Debug.DrawLine ( Camera.main.transform.position,
vectorCameraToObject, Color.green);

                        //Se o objeto estiver dentro do ângulo de 15 graus
previamente definido será adicionado ao array ObjectsInSight
                        if (angle < defaultAngle) {

                              filteredObjects.Add (TagObject);

                        }

                  }

                  return filteredObjects.ToArray();

            }

            //Procura o objecto com a tag BIM mais próximo dentro da lista
que foi criada.
            GameObject NearestObjectWithTag(string tag) {
                  GameObject[] TagObj;
                  GameObject nearest = null;

                  try {
                        TagObj = GameObject.FindGameObjectsWithTag("BIM");
                        GameObject[] BIMObjectsInSight =
ObjectsInSight(TagObj);
                        Debug.Log(BIMObjectsInSight);

                        if(BIMObjectsInSight.Length < tooManyObjectsInSight)
{
                              BIMObjectsInSight = ObjectsInSight(TagObj,
(int)defaultAngle);

                              if(BIMObjectsInSight.Length >
tooManyObjectsInSight) {
                              BIMObjectsInSight = ObjectsInSight(TagObj,
(int)defaultAngle/2);

                                    if(BIMObjectsInSight.Length >
tooManyObjectsInSight) {
                                    BIMObjectsInSight =
ObjectsInSight(TagObj, (int)defaultAngle/4);
                                    }
                              }
                        }

                        nearest = NearestObject(BIMObjectsInSight);
                        Debug.DrawLine (Camera.main.transform.position,
nearest.transform.position, Color.yellow);


                         //Debug.Log ("Got nearest object: ");
                        // Debug.Log (nearest.name);
                  } catch {

                  }
```

```
                  return nearest;

            }

            GameObject NearestObject(GameObject[] TagObj)
            {
                  GameObject nearest_object = null;
                  float minDist = Mathf.Infinity;
                  Vector3 currentPos = Camera.main.transform.position;
                  foreach (GameObject gameObject in TagObj)
                  {
                        Transform t = gameObject.transform;
                        float dist = Vector3.Distance(currentPos,
t.position);
                        if (dist < minDist)
                        {

                              nearest_object = gameObject;
                              minDist = dist;
                        }
                  }

                  //adiciona o script LeapRTS como componente do objeto mais
próximo.
                  //LeapRTS nearest_t = gameObject.AddComponent<LeapRTS>();
                  return nearest_object;

            }

      }
}
```

Nome: MoveObject.cs

Objetivo: Permite o movimento do utilizador pelo cenário através do botão analógico esquerdo do *joystick* utilizado no caso de estudo.

Linguagem de programação: C#.

```
using UnityEngine;
using System.Collections;

public class MoveObject : MonoBehaviour
{


      void Start ()
      {

      }
      void Update ()
      {
            if(Input.GetAxisRaw("Y axis")>0.4) {
                  translateWithoutY (Camera.main.transform.forward);

            }

            if(Input.GetAxisRaw("X axis")> 0.45){
                  translateWithoutY(Camera.main.transform.right * -1);
            }
            if(Input.GetAxisRaw("Y axis")< -0.65){
                  translateWithoutY(Camera.main.transform.forward * -1);
            }
            if(Input.GetAxisRaw("X axis")< -0.45){
                  translateWithoutY(Camera.main.transform.right);
            }
      }

      void translateWithoutY(Vector3 translation) {
            translation.y = 0;

            transform.Translate (translation * Time.deltaTime);
      }
}
```

Nome: Unity_to_Revit

Objetivo: Permite efetuar cópias elementos do projeto assim como executar alterações (posição e rotação) em objetos do modelo BIM em tempo real (ciclos de 16 milissegundos) através de uma interface em RV utilizada a partir do motor de jogo Unity.

Linguagem de programação: C#.

```
using System;
using System.Collections.Generic;
using System.IO;

using Autodesk.Revit.DB;
using Autodesk.Revit.UI;
using Autodesk.Revit.Attributes;
using System.Diagnostics;
using System.Globalization;
using System.Timers;
using System.Threading;
using Autodesk.Revit.UI.Events;
using Autodesk.Revit.ApplicationServices;

[Transaction(TransactionMode.Manual)]
public class Unity_To_Revit : IExternalCommand
{
    private static DateTime _lastUpdate = DateTime.Now;
    private static TimeSpan _interval = new TimeSpan(0, 0, 1);
    private static bool _executing = false;

    public Result Execute(
      ExternalCommandData commandData,
      ref string message,
      ElementSet elements)
    {
        UIApplication uiApp = commandData.Application;
        Document doc = uiApp.ActiveUIDocument.Document;

        uiApp.Idling += new EventHandler<IdlingEventArgs>(OnIdling);

        return Autodesk.Revit.UI.Result.Succeeded;
    }

    static void OnIdling(object sender, IdlingEventArgs e)
    {
        if (DateTime.Now.Subtract(_lastUpdate) > _interval && !_executing)
        {
            _executing = true;

            UIApplication uiApp = sender is UIApplication
                ? sender as UIApplication
                : new UIApplication(sender as Application);

            UIDocument uidoc = uiApp.ActiveUIDocument;
            Document doc = uidoc.Document;
```

```
            var readFile = new ReadFromFile();
            readFile.Read(uiApp, doc);

            _lastUpdate = DateTime.Now;
            _executing = false;


            // TaskDialog.Show("Idling", "Imported!");
        }
    }
  }

class ReadFromFile
{
    private static int linesRead = 0;
    private static Dictionary<int, int> cloneDictionary = new
Dictionary<int, int>();

    public void Read(UIApplication uiApp, Document doc)
    {
        List<Operation> fileOperations = getFileOperations(@"C:\Users\Sara
Cardoso 7\Desktop\Dissertação\WriteLines2.txt");

        // PERCORRER AS OPERAÇÕES
        while (linesRead < fileOperations.Count)
        {
            Operation operation = fileOperations[linesRead];
            // Declarar elementos
            ElementId elementId = null;

            // Buscar elementos do documento
            // Se o ID for negativo, assumir clone
            if (operation.ObjectId < 0)
            {
                // Se o elemento clone já foi adicionado e movido
                if (cloneDictionary.ContainsKey(operation.ObjectId))
                {
                    // TaskDialog.Show("Revit",
operation.ObjectId.ToString());
                    // Procura o ID associado a este no dicionário
                    elementId = new
ElementId(cloneDictionary[operation.ObjectId]);

                    // TaskDialog.Show("Revit",
elementId.IntegerValue.ToString());
                }
                // Se não foi adicionado --> vou buscar a copiar que é o
positivo do enviado
                else
                {

                    elementId = new ElementId(Math.Abs(operation.ObjectId));
                }
            }
            else
            {
```

```
                elementId = new ElementId(operation.ObjectId);
            }

            Element documentElement = doc.GetElement(elementId);

            // Identificar a operação
            // Usar os parametros para aplicar operação ao elemento
(Document element)

            // Translação no Revit
            if (operation._Type == 1)
            {
                MoveElement(uiApp, doc, elementId, operation);
            }

            if (operation._Type == 2)
            {
                RotateElement(uiApp, doc, documentElement, operation);
            }

            if(operation._Type == 3)
            {
                ElementId newElementId = DuplicateAndMoveElement(uiApp, doc,
elementId, operation);
                AddToDictionary(Convert.ToInt32(operation.Parameters[0]),
newElementId);
            }


            linesRead++;

        }
    }

    private void AddToDictionary(int negativeElementId, ElementId
newElementId)
    {
        cloneDictionary.Add(negativeElementId, newElementId.IntegerValue);
    }

    private ElementId DuplicateAndMoveElement(UIApplication app, Document
document, ElementId element, Operation operation)
    {
        ICollection<ElementId> newElementId = null;

        double Revit_X = operation.Parameters[1] * 3.281;
        double Revit_Y = operation.Parameters[3] * 3.281;

        // Move the element to new location.
        XYZ newPlace = new XYZ(Revit_X, Revit_Y, 0.00);

        using (Transaction transaction = new Transaction(document))
        {
            if (transaction.Start("Move Element") ==
TransactionStatus.Started)
            {
                Plane geomPlane = app.Application.Create.NewPlane(new XYZ(0,
0, 1), XYZ.Zero);
```

```
                SketchPlane sketch = SketchPlane.Create(document,
geomPlane);

                newElementId = ElementTransformUtils.CopyElement(document,
element, newPlace);

                if (TransactionStatus.Committed != transaction.Commit())
                {
                    TaskDialog.Show("Failure", "Transaction could not be
committed");
                }

            }
        }

        foreach (var elementId in newElementId)
            return elementId;

        return null;
    }

    void RotateElement(UIApplication application, Document document, Element
element, Operation operation)
    {
        // TaskDialog.Show("Revit", "Rotating");
        bool rotated = false;
        LocationPoint location = element.Location as LocationPoint;

        if (null != location)
        {
            using (Transaction transaction = new Transaction(document))
            {
                if (transaction.Start("Move Element") ==
TransactionStatus.Started)
                {
                    Plane geomPlane =
application.Application.Create.NewPlane(new XYZ(0, 0, 1), XYZ.Zero);

                    // Create a sketch plane in current document
                    SketchPlane sketch = SketchPlane.Create(document,
geomPlane);

                    XYZ aa = location.Point;
                    XYZ cc = new XYZ(aa.X, aa.Y, aa.Z + 10);
                    Line axis = Line.CreateBound(aa, cc);
                    location.Rotate(axis, operation.Parameters[0] * 2 *
Math.PI / 360 * -1);

                if (TransactionStatus.Committed != transaction.Commit())
                    {
                        TaskDialog.Show("Failure", "Transaction could not be
committed");
                    }

                }
            }

        }
```

```
    }

    public void MoveElement(UIApplication app, Document document, ElementId
elementId, Operation operation)
    {
        // get the element current location
        LocationPoint elementLocation =
document.GetElement(elementId).Location as LocationPoint;

        XYZ oldPlace = elementLocation.Point;

        double Revit_X = operation.Parameters[0] * 3.281;
        // TaskDialog.Show("Revit", Revit_X.ToString());
        double Revit_Y = operation.Parameters[2] * 3.281;
        // TaskDialog.Show("Revit", Revit_Y.ToString());


        // Move the element to new location.
        XYZ newPlace = new XYZ(Revit_X, Revit_Y, operation.Parameters[1]);
        // TaskDialog.Show("Revit", elementId.ToString() + " " +
Revit_X.ToString() + " " + Revit_Y.ToString() + " " +
operation.Parameters[1].ToString());

        using (Transaction transaction = new Transaction(document))
        {
            if (transaction.Start("Move Element") ==
TransactionStatus.Started)
            {
                Plane geomPlane = app.Application.Create.NewPlane(new
XYZ(0,0,1), XYZ.Zero);

                // Create a sketch plane in current document
                SketchPlane sketch = SketchPlane.Create(document,
geomPlane);

                ElementTransformUtils.MoveElement(document, elementId,
newPlace);

                if (TransactionStatus.Committed != transaction.Commit())
                {
                    TaskDialog.Show("Failure", "Transaction could not be
committed");
                }

            }
        }
        // Agora procura a nova posição no elemento
        elementLocation = document.GetElement(elementId).Location as
LocationPoint;
        XYZ newActual = elementLocation.Point;

        /*string info = "Original Z location: " + oldPlace.Z +
                        "\nNew Z location: " + newActual.Z;

        // TaskDialog.Show("Revit", info);*/
    }

    private static List<Operation> getFileOperations(string filepath)
    {
        List<Operation> fileOperations = new List<Operation>();
```

```
        string[] lines = File.ReadAllLines(filepath);


        // Linha
        foreach (string line in lines)
        {
            Operation newOp = new Operation();
            List<double> parList = new List<double>();

            // Divide colunas
            string[] columns = line.Split(new[] { ',' });

            // Trabalha nas colunas
            newOp.ObjectName = columns[0];

            newOp.ObjectId =
int.Parse(columns[0].Split('[')[1].Split(']')[0]);

            newOp._Type = int.Parse(columns[1]);

            for (int counter = 2; counter < columns.Length; counter++)
            {
                // TaskDialog.Show("Revit", columns[counter]);
                parList.Add(double.Parse(columns[counter],
CultureInfo.InvariantCulture));
                // TaskDialog.Show("Revit", double.Parse(columns[counter],
CultureInfo.InvariantCulture).ToString());
            }

            newOp.Parameters = parList;

            fileOperations.Add(newOp);
        }

        return fileOperations;
    }
}


class Operation
{
    public string ObjectName;
    public int ObjectId;
    public int _Type;
    public List<double> Parameters;
}
```

ANEXO III

INQUÉRITO APRESENTADO NA REALIZAÇÃO DO CASO DE ESTUDO

Leia cuidadosamente as instruções:

O presente inquérito tem como objetivo a avaliação da plataforma de interface entre Realidade Virtual (RV) e *software* BIM (*Building Information Modeling*) desenvolvida na unidade curricular de Dissertação em Construções do Mestrado Integrado em Engenharia Civil. A avaliação será no âmbito da naturalidade ou grau de intuição na utilização dos comandos criados para a plataforma de RV.

O caso de estudo consiste em duas fases, prática e teórica. Durante a parte prática será testado um modelo através de RV. Na parte teórica irá ocorrer o preenchimento de um inquérito.

O tempo estimado para a conclusão do teste será de 20 min.

Parte prática:

O teste contará com uma **apresentação do modelo em Autodesk Revit**, seguindo-se a **exploração de um cenário no motor de jogo Unity**. A interação com o motor de jogo será feita através de um *joystick* para controlos mais precisos, um *Head Mounted Display* (Oculus Rift DK2) e um sensor de captação de gestos (Leap Motion).

O modelo 3D compreende duas zonas para interação, concebidas para que o utilizador possa estabelecer contato com a plataforma de modo gradual.

A **zona 1** será explorada em dois momentos distintos, cada um deles com objetivos específicos.

Na **zona 2**, contrariamente ao que sucede com a anterior, estarão à disposição todos os comandos e funcionalidades da interface.

Pretende-se que os inquiridos percorram cada uma das diferentes zonas cumprindo respetivamente os objetivos propostos.

As instruções para interação com os comandos serão fornecidas oralmente.

Dúvidas durante o teste, parte prática e/ou teórica, poderá expô-las ao investigador em qualquer momento.

- Possui formação/experiência na área de Engenharia Civil, Arquitetura ou Construção? (Assinale com X)

  SIM ___ NÃO ___

- Está familiarizado com a metodologia BIM (*Building Information Modeling*)? (Assinale com X)

  SIM ___ NÃO ___

- Já experimentou algum equipamento de RV? (Assinale com X)
  SIM ___ NÃO ___

- Se respondeu SIM à questão anterior, qual? _____

Zona 1

Primeiro teste:

Objetivos:

1. Domínio do movimento em cena;
2. Familiarização com a funcionalidade de seleção.

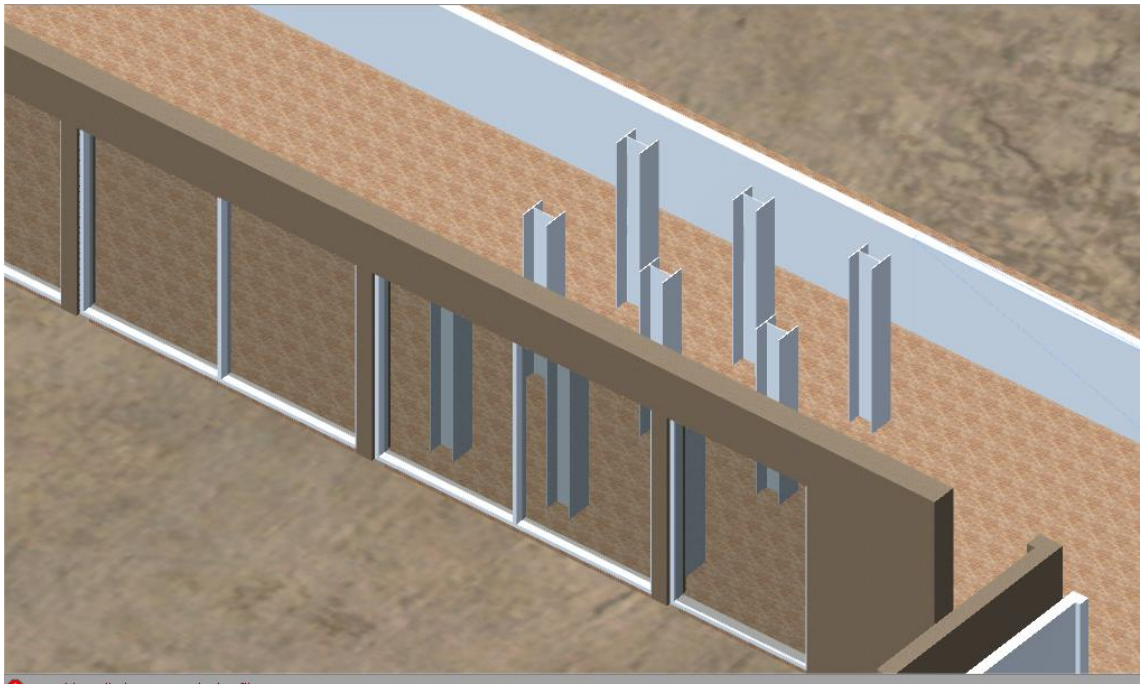


FIG.A.3.1 – ZONA 1

- Completou todos os objetivos da Zona 1? (Assinale com X)

  SIM ___ NÃO ___

- Se respondeu NÃO à questão anterior, indique qual/quais do/dos objetivo/objetivos não completou. (Assinale com X)

  Objetivo 1 __ Objetivo 2 __

A partir da escala ordinal classifique quanto ao grau de intuição/naturalidade cada um dos comandos utilizados.

(Assinale com X **uma** das opções da escala)

Comando	Nada intuitivo	Pouco intuitivo	Requer muito pouco treino	Natural, não necessita de treino
Movimento (*joystick*)	—	—	—	—
Seleção (*joystick*)	—	—	—	—

Zona 1

Segundo teste:

Objetivos:

1. Domínio do comando de translação (*joystick*);
2. Domínio do comando de rotação (*joystick*);
3. Domínio do comando de cópia (*joystick*);
4. Domínio do comando de rotação (Leap Motion);
5. Domínio do comando de translação (Leap Motion).

- Completou todos os objetivos desta zona? (Assinale com X)

  SIM ___ NÃO ___

- Se respondeu NÃO à questão anterior, indique qual/quais do/dos objetivo/objetivos não completou. (Assinale com X)

  Objetivo 1 __ Objetivo 2 __ Objetivo 3 __ Objetivo 4 __ Objetivo 5 __

A partir da escala ordinal classifique quanto ao grau de intuição/naturalidade cada um dos comandos utilizados.

(Assinale com X **uma** das opções da escala)

Comando	Nada intuitivo	Pouco intuitivo	Requer muito pouco treino	Natural, não necessita de treino
Translação (*joystick*)	—	—	—	—
Rotação (*joystick*)	—	—	—	—
Criação de cópia (*joystick*)	—	—	—	—
Rotação (Leap Motion)	—	—	—	—
Translação (Leap Motion)	—	—	—	—

Zona 2

Objetivos:

1. Criar uma cópia do lavatório existe neste espaço
2. Alterar a posição da cópia, colocando-a sobre uma nova parede, para além da que suporta o lavatório original.

- Completou todos os objetivos da zona 2? (Assinale com X)

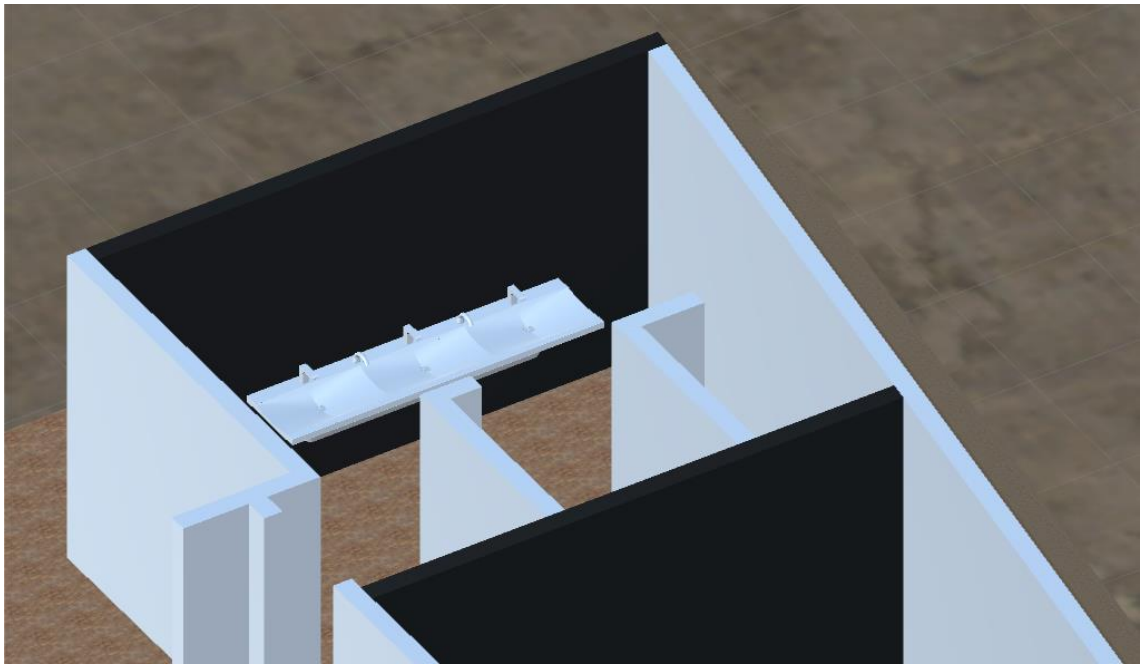


FIG.A.3.2 – ZONA 2

SIM ___ NÃO ___

- Se respondeu NÃO à questão anterior, indique qual/quais do/dos objetivo/objetivos não completou. (Assinale com X)

  Objetivo 1 __ Objetivo 2 __

A partir da escala ordinal classifique quanto ao grau de intuição/naturalidade **a globalidade da interação na execução dos objetivos propostos para a zona 2.**

(Assinale com X **uma** das opções da escala)

Nada intuitiva	Pouco intuitiva	Requer muito pouco treino	Natural, não necessita de treino
__	__	__	__

No final do teste confira, por favor, todas as respostas.

Obrigado!