

Mestrado

Arquitectura

Programação aplicada ao desenvolvimento do projecto

Henrique Robalo

M
2022



Programação aplicada ao desenvolvimento do projecto

Processos de optimização digital do projecto de arquitectura

Aluno:

Henrique Manuel Robalo Martinho
Aluno 201704430 da Universidade do PORTO

Orientador:

Professor Doutor Carlos Nuno Lacerda Lopes

Porto 2022 / Maio

Universidade do Porto (UP)

Faculdade de Arquitectura da Universidade do Porto (FAUP)

Mestrado Integrado de Arquitectura (MIARQ)

Dissertação de Mestrado de Arquitectura

Esta dissertação foi entregue no dia 29 de Maio de 2022

A prova pública foi realizada no dia 14 de Julho de 2022 onde o trabalho foi apresentado e defendido perante o seguinte júri:

Orientador: Professor Doutor Carlos Nuno Lacerda Lopes

Arguente: Professor Doutor José Pedro Ovelheiro Marques de Sousa

Presidente: Professor Catedrático João Pedro Sampaio Xavier

O trabalho obteve a classificação de 17 valores.

Dedicatória:

Dedico este trabalho aos meus queridos pais por me terem ensinado o valor da educação e a importância do pensamento crítico, por me apoiarem sempre no meu percurso intelectual e ao meu fiel cão pelo amor e companhia nas inúmeras horas que dediquei a este trabalho.

Agradecimentos:

Quero agradecer ao Professor Nuno Lacerda pela simpatia em orientar este trabalho com a sua visão pragmática e directa e por me ter recebido no seu escritório para me orientar.

Quero agradecer à linda Cidade do Porto por me ter recebido com tanta simpatia e carinho, pela sua personalidade e pela sua identidade.

Quero agradecer à minha família pelo apoio, amizade e pelo ambiente familiar saudável, com particular carinho pela minha prima Cátia.

Quero agradecer aos meus queridos amigos do Norte.

Ao Diogo Fernandes, à Maria Lamela, ao Daniel Carvalho, ao David Oliveira, ao Jorge Carreiras, à Maria Valentim e ao Miguel Costeira com quem passei os melhores momentos no Porto, amigos que me trataram como irmão.

Quero agradecer aos meus queridos amigos da Beira.

Ao João Dias, ao Daniel Pereira, à Diana Pereira, ao Hugo Nobre, amigos que me ensinaram a rir e a ser feliz, com quem cresci e com quem passei momentos incríveis e vivi aprendizagens que não esquecerei.

Quero agradecer também a diversas personalidades internacionais porque o seu contributo e as suas visões desenvolveram a sociedade na direção de um mundo livre através da partilha do conhecimento:

Ton Roosendaal por ter criado e oferecido o Blender ao mundo / à comunidade open-source do Blender.

Linus Torvalds pelo seu contributo na criação do Linux e do Git.

Guido van Rossum por ter criado uma linguagem de programação tão simples e ao mesmo tempo poderosa, o Python.

Jürgen Riegel, Werner Mayer, Yorik van Havre e à comunidade open-source do FreeCAD.

À comunidade open-source do LibreCAD.

Tabela de Conteúdos

Sumário

Abstract

Resumo

Dicionário de termos

Módulo 1

1.1 - Tema

1.2 - Objecto de estudo

1.3 - Objectivo do trabalho

1.4 - Metodologia de trabalho

Módulo 2

2.0 - Descrição do Módulo 2

2.1 - A evolução do projecto

2.2 - O estado de eficiência actual do projecto de arquitectura

Módulo 3

3.0 - Descrição do Módulo 3

3.1 - A evolução dos sistemas digitais

3.2 - Computer-Aided Design (CAD)

3.3 - Building Information Modeling (BIM)

3.4 - Algorithm-Aided Design (AAD)

3.5 - Algorithm-Aided Building Modeling Information (AABIM)

3.6 - Programação Escrita

Módulo 4

4.0 - Descrição do Módulo 4

4.1 - Caso de estudo CAD - Bankside Loft

4.2 - Caso de estudo BIM - Helsinki Music Center

4.3 - Caso de estudo AAD - Kai Strehlke

4.4 - Caso de estudo AABIM - Iasbik, Martinez e Gazel

4.5 - Caso de estudo Programação Escrita - Maria Gkioka

Módulo 5

5.0 - Descrição do Módulo 5

5.1 - Exemplo 1 - Automatização de tarefas específicas ou rotinas no software

5.2 - Exemplo 2 - Criação de ferramentas para realizar tarefas específicas

5.3 - Exemplo 3 - Utilização de programação para conectar/hiperligar diversas fontes de informação automaticamente

5.4 - Exemplo 4 - Utilização de Python para automatizar tarefas repetitivas.

5.5 - Exemplo 5 - Criação de modelos interactivos

Módulo 6

6.1 - Observações realizadas em relação às metodologias de trabalho existentes

6.2 - Aplicação da programação no desenvolvimento do projecto

6.3 - Como desenvolver este estudo no futuro

Bibliografia

Referências Gráficas

Anexos:

Scripts utilizados no Módulo 5:

Tabelas

Sumário

10	Abstract
11	Resumo
12	Dicionário de termos
16	Módulo 1 - Introdução
17	1.4 - Tema em estudo
18	1.5 - Introdução do Objecto de Estudo
20	1.6 - Objectivo do Trabalho
21	1.7 - Metodologia de trabalho
22	Módulo 2 - O projecto
24	2.1 - A evolução do projecto
27	2.2 - O estado de eficiência actual no projecto de arquitectura
32	Módulo 3 - Contribuição das metodologias de trabalho digital para a eficiência do trabalho
34	3.1 - Evolução dos sistemas digitais
39	3.2 - Computer-Aided Design (CAD)
46	3.3 - Building Information Modeling (BIM)
52	3.4 - Algorithm-Aided Design (AAD)
56	3.5 - Algorithm-Aided Building Modeling Information (AABIM)
61	3.6 - Programação Escrita
70	Módulo 4 - Aplicações práticas das metodologias de trabalho digital (Casos de Estudo)
72	4.1 - Caso de estudo (CAD) - Bankside Loft
76	4.2 - Caso de estudo (BIM) - Helsinki Music Center
79	4.3 - Caso de estudo (AAD) - Kai Strehlke
82	4.4 - Caso de estudo (AABIM) - Iasbik Martinez e Gazel
88	4.5 - Caso de estudo (Programação Escrita) - Maria Gkioka

92	Módulo 5 - Exemplos de aplicações práticas de programação no trabalho do projectista
94	5.1 - Exemplo 1 - Automatização de Tarefas específicas no software
98	5.2 - Exemplo 2 - Criação de ferramentas para realizar tarefas específicas
101	5.3 - Exemplo 3 - Utilização de programação para conectar/hiperligar diversas fontes de informação automaticamente
104	5.4 - Exemplo 4 - Utilização de Python para automatizar tarefas repetitivas.
106	5.5 - Exemplo 5 - Criação de modelos interactivos
110	Módulo 6 - Conclusão
111	6.1 - Observações realizadas em relação às metodologias de trabalho existentes
112	6.2 - Aplicação da programação no desenvolvimento do projecto
113	6.3 - Como desenvolver este estudo no futuro
114	Bibliografia
118	Referências de Imagens
122	Anexos

Abstract

The evolution of digital systems during the XX and XXI centuries changed the work paradigm of the architects and dynamized project development by creating new possibilities.

This work initiates by analysing the influence of digital technologies on the evolution of the project development processes and how these new tools are currently used to develop architecture projects.

Afterwards there will be analysed 5 digital work methodologies, CAD, BIM, AAD, AABIM and computer programming/scripting. These 5 working methodologies will be described with the intention of understanding how can each one of these contribute to the project development, identifying the possibilities and limits of each one of them.

Following that, there will be studied 5 case studies that demonstrate practical applications of each one of these methodologies and how their characteristics optimised and improved the development process of various projects.

At last, this work will bring into focus computer programming/scripting to better understand and demonstrate how a greater computer control through High-Level programming languages (like Python for example) can improve the working possibilities of the project developer. In this module, there will be presented some examples (that were created and developed during this work) that demonstrate some practical coding/scripting applications on the architects work and how programming knowledge can be used to overcome some limits found on the other digital work methodologies.

Resumo

A evolução dos sistemas digitais no século XX e XXI criou um novo paradigma de trabalho dos arquitectos e dinamizou o desenvolvimento do projecto ao criar novas possibilidades.

Este trabalho começa por analisar a influência das tecnologias digitais na evolução dos processo de projectar e de que forma estas novas ferramentas são utilizadas na actualidade para desenvolver projectos de arquitectura.

Em seguida vão ser analisadas 5 metodologias de trabalho digital, CAD, BIM, AAD, AABIM e a Programação Escrita. Estas 5 metodologias de trabalho vão ser descritas com a intenção de compreender de que forma cada uma destas contribui para o desenvolvimento do projecto, identificando as potencialidades e os limites de cada uma.

Posteriormente vão ser estudados 5 casos de estudo que demonstram aplicações práticas de cada uma destas metodologias e de que forma as características de cada uma delas optimizaram e potenciaram o processo de desenvolvimento de diversos projectos.

Por fim o trabalho vai-se focar na temática da programação de computadores para compreender melhor e demonstrar de que forma o domínio do computador através do conhecimento de linguagens de programação (como por exemplo o Python) pode expandir as capacidades de trabalho/desenvolvimento do projectista. Neste módulo, vão ser apresentados alguns exemplos (que foram criados e desenvolvidos no decorrer do trabalho) que demonstram algumas aplicações práticas da programação no trabalho dos projectistas e como este conhecimento pode ser utilizado para ultrapassar alguns limites existentes nas outras metodologias de trabalho.

Dicionário de termos

AAB - Algorithm-Aided BIM.

AAD - Algorithm-Aided Design.

AAM - Algorithm-Aided Modeling.

AEC - Architects Engineers Constructors - Este termo que se refere à arquitetura, engenharia e construção para caracterizar a área/indústria da concepção de edifícios.

API - Application Programming Interface - Isto é um conceito da área da programação. Este conceito permite que programadores controlem um sistema de forma interna através de um API que estabelece a comunicação entre um script externo e o sistema.

AR - Augmented Reality - Realidade Aumentada.

Assembly Language - Linguagem de Programação.

Assembler - Tradutor/conversor de Assembly Language em Machine Language.

BDS - "Building Description System" - é um software de computador desenvolvido por Charles Eastman na segunda metade do século XX.

BIM - Building Modeling Information.

CAD - Computer-Aided Design.

CAM - Computer-Aided Manufacturing.

CNC - Computer Numeric Control - processo de fabrico com auxílio de computadores.

Compilador/Compiler - Tradutor/conversor de High-Level Language em Machine Language.

CPU - "Central Processing Unit" refere-se a um componente de hardware fundamental nos computadores responsável pelo processamento das instruções.

CSV - Comma Separated Values ("untitled.csv" por ex) é um tipo de ficheiro que armazena informação numérica ou textual numa forma semelhante a uma tabela com colunas e linhas.

Display - Ecrã.

FEA - Finite Element Analysis.

GAE / VAE - Graphical Algorithm Editor / Visual Algorithm Editor.

GUI - Graphical User Interface - conceito utilizado para descrever a parte visual do software com a qual o utilizador pode geralmente interagir.

Hardware - Termo utilizado para descrever componentes físicas de um sistema digital (por exemplo cpu, gpu, joystick, teclado ou rato).

High-Level Language - Tipo de Linguagem de Programação.

Interface Visual / Interface Gráfica - No contexto deste trabalho, este termo é utilizado para descrever uma forma de visualização do conteúdo digital através de por exemplo displays/ecrãs.

Interpretador / Interpreter - Tradutor/conversor de High-Level Language em Machine Language.

Iterador - Método utilizado em linguagens de programação High-Level para repetir tarefas.

Low-Level Language - Tipo de Linguagem de Programação.

Método - esta palavra é utilizada neste trabalho em dois contextos. Ela pode referir-se a um método/abordagem de trabalho ou pode referir-se a "method", um conceito teórico utilizado na programação escrita que descreve funções como por exemplo func() def() if().

Machine Language - Linguagem de Programação interpretada por computadores (0's e 1's).

MIT - Massachusetts Institute of Technology.

Modelo Geométrico - No contexto deste trabalho este conceito refere-se ao modelo do projecto que contém informação geométrica geralmente CAD.

Modelo BIM - No contexto deste trabalho este conceito refere-se ao modelo do projecto que contém informação orientada aos seus objectos, informação BIM.

Modelo Animado - No contexto deste trabalho este conceito refere-se ao modelo do projecto que contém animações dos objectos do edifício tal como pode ser observado na figura 2 do exercício 1 (M5.1).

Modelo Interactivo - No contexto deste trabalho este conceito refere-se ao modelo do projecto em que o utilizador pode interagir em tempo real com os objectos do edifício tal como pode ser observado na figura 2 do exercício 5 (M5.5).

Modelo Económico - No contexto deste trabalho este conceito refere-se ao modelo do projecto que contém informação económica dos objectos do edifício tal como pode ser observado na figura 2 do exercício 3 (M5.3).

Modelo de Resistência ao Fogo - No contexto deste trabalho este conceito refere-se ao modelo do projecto que contém informação acerca da resistência ao fogo dos objectos do edifício tal como pode ser observado na figura 2 do exercício 3 (M5.3).

Nodes - software para criar algoritmos digitais de forma visual através do uso de Blocos/módulos.

Numerical Control - abreviado por vezes como "NC", serve para caracterizar um processo de manufactura de peças através da programação de instruções nas máquinas industriais.

Object-Oriented - Conceito utilizado para descrever uma metodologia que inclua entidades individuais.

OOM - Object-Oriented Modeling - isto refere-se a uma metodologia de modelação com base em objectos com identidade e características ao invés de simples elementos geométricos como linhas e volumes.

Planos - No contexto deste trabalho, esta palavra é por vezes utilizada em vez de "desenhos" para caracterizar geralmente elementos gráficos tais como plantas, alçados, cortes e detalhes.

Projectista - No contexto deste trabalho, "projectista" refere-se a arquitectos ou engenheiros

Software - Termo utilizado para descrever componentes não palpáveis de um sistema digital (por exemplo programas de computador, ficheiros ou sistemas operativos).

TX-2 - Computador do laboratório de Lincoln do MIT criado em 1958 (também conhecido como "Lincoln TX-2").

UI - User Interface conceito utilizado para descrever a parte visual do software com a qual o utilizador pode geralmente interagir.

VAE / GAE - Visual Algorithm Editor / Graphical Algorithm Editor.

VR - Virtual Reality.

Em diversas situações nesta dissertação são utilizados parênteses em vez de vírgulas porque os parênteses indicam de forma mais intuitiva quando a informação é complementar e não altera o sentido da frase.

Módulo 1

Introdução

1.1 - Tema

Este trabalho procura abordar a temática da integração digital na arquitectura, mais concretamente de que forma o uso de sistemas tecnológicos digitais se traduz numa mais valia para o trabalho do arquitecto em prol da qualidade do projecto de arquitectura. A integração de sistemas digitais na arquitectura é uma área relativamente ampla e, por esse mesmo motivo, este trabalho procura centrar-se na temática da eficiência no processo de trabalho através da optimização de tarefas e a ampliação das capacidades do software com base em instrumentos tais como por exemplo a programação escrita ou visual.

A programação está na base de todos os softwares existente nos computadores utilizados pelos arquitectos, no entanto estes utilizadores não precisam de saber programar para operar as tecnologias digitais porque através de um interface gráfico (que faz uma ponte na comunicação entre o utilizador e o computador) conseguem controlar o software e o resultado final. Estes interfaces gráficos foram desenhados para tornar a utilização do computador mais intuitiva e fácil de compreender e, assim sendo, facilitaram o acesso às tecnologias e expandiram a sua utilização a um público muito maior de utilizadores. Contudo, os arquitectos ficam limitados às funcionalidades ou às opções que este interface gráfico oferece e, a utilização de programação permite expandir estes limites através da criação de funcionalidades/ferramentas que satisfaçam as necessidades específicas/ pessoais desde que o proprietário do software disponha API's "Application Programming Interface" que suportem a criação de funcionalidades por terceiros "third party developers".

No fundo, o tema do trabalho explora de que forma o arquitecto consegue tirar proveito dos instrumentos/programas de trabalho digitais de arquitectura ou aprender a criar instrumentos, mecanismos ou técnicas que optimizem e complementem o seu trabalho, rentabilizando ainda mais o uso destes instrumentos digitais já utilizados pelos arquitectos.

1.2 - Objecto de estudo

A disciplina de projectar em arquitectura existe há muitos mais anos do que a ciência dos computadores, contudo, apenas desde os finais do século XX é que esta começou a ser utilizada para o trabalho de arquitectura. No século XXI, as tecnologias digitais estão cada vez mais integradas no trabalho dos arquitectos e o uso do computador como instrumento de trabalho na arquitectura é importante ao ponto de se ter tornado em si num assunto de estudo/análise académica estudada por diversos autores como por exemplo, Michael Kilkelly¹.

No entanto, a utilização de sistemas digitais na arquitectura, é frequentemente estudada do ponto de vista das capacidades do software com interface gráfica e apenas mais recentemente passou a ser estudada do ponto de vista da utilização/integração de algoritmos no processo de projectar com o aparecimento da programação visual (através de Visual Algorithm Editors, também conhecidos como Node Systems) tal como o Grasshopper para Rhinoceros 3D (Setembro de 2007), o Dynamo para Autodesk Revit (2013), Geometry Nodes no Blender (versão 2.92 - Fevereiro 2021) ou Xpresso no Maxon Cinema4D.

A utilização de AAD (Algorithm-Aided Design) está a ter cada vez mais atenção/popularidade no ponto de vista da arquitectura ou de outras áreas e está a ser integrado nos softwares mais utilizados como o Graphisoft Archicad, o Autodesk Revit, o Rhinoceros3D e o Blender.

Autores como Harri Humpi e Toni Österlund² apontam também para a criação de novas ideologias/metodologias de trabalho que estes descrevem no seu trabalho "Algorithm-Aided BIM" e denominam de AAB (Algorithm-Aided BIM), resultado da integração de AAD (Algorithm-Aided Design) no BIM (Building Information Modeling).

A integração de computadores na área da arquitectura revolucionou a dinâmica de trabalho dos arquitectos e os softwares digitais dispõem de diversas ferramentas que realizam o trabalho de forma optimizada, no entanto por vezes o trabalhador não possui a ferramenta ideal que cumpra ou satisfaça a sua necessidade. Quando não existe uma ferramenta que execute de forma automática a tarefa que se pretende realizar, essa tarefa tem de se realizar de forma manual e demorada (desperdiçando valiosas horas de trabalho no processo). Por outro lado, se o trabalhador tiver os conhecimentos e o software dispuser de um API (Application Programming Interface), ele pode desenvolver uma forma automática de realizar essa mesma tarefa com o auxílio da programação visual para tarefas mais gerais e rápidas e a programação escrita para as situações mais específicas.

¹ KILKELLY, Michael James - Object-Oriented Representations. Massachusetts Institute of Technology (MIT). 1996

² HUMPI, Harri; ÖSTERLUND, Toni - Algorithm-Aided BIM. 2015.

É neste contexto que é introduzido o objecto de estudo deste trabalho, ou seja, o uso de programação em arquitectura, mais concretamente, de que forma os arquitectos podem utilizar conhecimentos de programação para expandir as capacidades do software de trabalho e criar novas técnicas e mecanismos de automatização que permitem executar as tarefas do projecto de forma mais rápida mantendo a mesma qualidade do trabalho.

Com a optimização de tarefas, pretende-se melhorar a eficiência do processo de trabalho do arquitecto em benefício da qualidade do projecto.

Saber trabalhar com tecnologias digitais cria conexões disciplinares e amplia os conhecimentos da arquitectura a outras áreas intelectuais e vice-versa (tais como a engenharia civil ou a engenharia mecânica) permitindo uma interoperabilidade/cooperação mais próxima com estas áreas. Articular diversas áreas de trabalho evita que arquitectos e engenheiros de várias áreas trabalhem/operem de forma isolada e independente tal como afirmado por Maria Gkioka no seu trabalho "Novel Parametric Knowledge Modelling approach applied to Viaducts"³.

O domínio digital permite também uma arquitectura pensada de forma dinâmica com a ajuda de simulações em tempo real como por exemplo o uso de formas cinéticas⁴ ou estruturas calculadas/generadas em tempo real de acordo com a forma geométrica⁵ através de software de tipologia AAD (Algorithm-Assisted Design) como por exemplo o GrassHopper, o Dynamo ou os Geometry Nodes do Blender.

³ GKIOKA, Maria et al. - Novel Parametric Knowledge Modelling approach applied to Viaducts. 2019) 9.

⁴ SHARAIDIN, Kamil; BURRY, Jane; SALIM, Flora - Integration of Digital Simulation Tools With Parametric Designs to Evaluate Kinetic Façades for Daylight Performance.

⁵ DANHAIVE, Renaud Alexis; MUELLER, Caitlin T. - Combining parametric modeling and interactive optimization for high-performance and creative structural design. 2015

1.3 - Objectivo do trabalho

O uso de tecnologias digitais obriga cada vez mais os arquitectos a desenvolverem as suas aptidões nesta área para conseguirem acompanhar o ritmo exponencial de desenvolvimento tecnológico. O computador transformou-se num instrumento essencial ao arquitecto porque consegue automatizar tarefas de forma mais rápida e flexível. Esta capacidade de cálculo quase instantânea permitiu a criação de novas ideologias e metodologias de trabalho que contribuíram em grande parte para melhorar a qualidade do projecto de arquitectura. No entanto, os arquitectos são formados para trabalharem com o software de computador e não para perceber como este funciona ou como pode ser adaptado às necessidades do trabalho.

É neste contexto que este trabalho procura novas maneiras de aproximar os conhecimentos da programação de computadores aos conhecimentos da arquitectura.

Com este trabalho, pretende-se explorar, demonstrar e partilhar de que maneira o uso de programação (escrita ou visual) pode ser utilizado para adaptar os instrumentos e ferramentas de trabalho digitais às necessidades do projecto de arquitectura (como por exemplo automatizar as tarefas repetitivas ou criar formas complexas quase instantaneamente) e melhorar a eficiência do trabalho do arquitecto. Procura-se também explorar e demonstrar se a capacidade de automatizar o trabalho pode ajudar a resolver problemas de trabalho excessivo extra-horário (como por exemplo trabalhar nas horas de repouso "noitadas") ou melhorar a qualidade trabalho do projecto de arquitectura.

1.4 - Metodologia de trabalho

Este trabalho vai-se iniciar pela contextualização da relação entre a arquitectura e as tecnologias digitais para se perceber de que forma estas foram introduzidas/integradas gradualmente no trabalho do arquitecto desde o final do século XX. Depois de analisada a evolução da relação entre as tecnologias digitais e a arquitectura, vai ser apurado o estado do projecto na actualidade procurando compreender quais as metodologias e processos de trabalho mais procuradas comercialmente.

Em seguida serão explorados com maior detalhe as metodologias de trabalho digital procurando compreender as suas diferenças, o seu potencial e os seus limites no desenvolvimento dos projectos.

Posteriormente, serão estudados 5 casos de estudo (um caso de estudo por cada metodologia) para se compreender de que forma diversos autores aplicaram as diferentes metodologias em benefício do projecto de arquitectura.

Por fim, pretende-se descobrir a utilização da programação escrita para desenvolver o projecto. Neste contexto, foram realizados alguns exemplos/exercícios para demonstrar como é possível utilizar o conhecimento de programação para executar tarefas específicas ou expandir as capacidades do computador e dos softwares em benefício da eficiência do trabalho do arquitecto.

Para terminar o trabalho vão ser tiradas conclusões com o objectivo de determinar até que ponto a integração de programação na arquitectura traz benefícios para o desenvolvimento do projecto.

Módulo 2

O Projecto

2.0 - Descrição do Módulo 2

O Módulo 2 serve para contextualizar a prática do projecto no desenvolvimento de edifícios. Este Módulo está dividido em 2 blocos.

O primeiro bloco tem como objectivo descrever brevemente a evolução da prática projectual ao longo de vários séculos e as suas ferramentas de trabalho.

O segundo bloco tem como objectivo determinar o estado actual da prática projectual e determinar a sua eficiência.

2.1 - A evolução do projecto

Formas primitivas de projectar:

A prática de construir nem sempre foi associada ao acto de projectar, a construção começou por ser uma tarefa intuitiva do ser Humano como forma de satisfazer as suas necessidades como por exemplo a de se abrigar. Apenas mais tarde, é que o desenho começou a ser utilizado pelo ser Humano como forma de organizar e comunicar as suas ideias/projectos com outros indivíduos. No seu livro, David E. Weisberg, indicou que algumas das invenções mais antigas foram documentadas em pergaminhos ou até mesmo em tabletes de barro. Weisberg explica também que o parthenon Grego não teria sido possível de construir sem a utilização de documentos gráficos para calcular a forma de todas as pedras, especialmente porque na sua construção participaram muitas pessoas.⁶

Segundo o historiador David Turnbull, a utilização do desenho técnico como instrumento de trabalho permitiu ao Homem organizar e gerir melhor a crescente complexidade dos seus projectos. "The achievement on the order of complexity and structural innovation involved in the construction of the cathedrals required a high degree of precision in the production of the stones and larger numbers of workers as well as types of workers. These factors create organizational difficulties that turn crucially on a fundamental problem: communication. Knowledge and instructions had to move among many participants (Turnbull 1993, 320)"⁷ O desenho como instrumento de trabalho resultou no desenvolvimento da disciplina de projectar e permitiu que a arquitectura se tornasse um trabalho independente da construção e gestão do projecto segundo Michael Kilkelly. "Concurrent with the evolution of the drawing as a communication tool was the development of architecture as a separate profession from construction"⁸ Esta mudança aliviou o arquitecto da tarefa de gerir a construção da obra e permitiu que este pudesse dedicar a sua atenção ao projecto em si com o objectivo de desenvolver melhor as suas formas. "the drawing was eventually recognized as a means to free the individual from direct supervision on the site and allowed him to work on several buildings simultaneously"⁹

⁶ WEISBERG, David E. - A Brief Overview of the History of CAD. Em The Engineering Design Revolution [Em linha] Disponível em WWW:<URL:www.cadhistory.net>. 2008. p2-1

⁷ TURNBULL, David apud KILKELLY, Michael James - Object-Oriented Representations. Massachusetts Institute of Technology (MIT). 1996. p14

⁸ KILKELLY, Michael James - Object-Oriented Representations. Massachusetts Institute of Technology (MIT). 1996. p14

⁹ KILKELLY, Michael James - Object-Oriented Representations. Massachusetts Institute of Technology (MIT). 1996. p14

Projecto como conjunto de documentos com informações multi-disciplinares:

A arquitectura evoluiu como uma área de investigação intelectual mais independente da construção do edifício. Esta evolução resultou no desenvolvimento tecnológico dos edifícios e com isso aumentou a complexidade associada a estes. Resultado da crescente complexidade, a função do projecto evoluiu para além da sua forma geométrica e passou a integrar outros tipos de informação e áreas disciplinares vitais ao funcionamento dos edifícios actuais. Alguns exemplos destas novas exigências são: os sistemas elétricos, a legislação, o programa do edifício, sistemas de proteção contra incêndios, condições térmicas, condições energéticas, condições de ventilação, orçamentos, a manutenção do edifício depois de construído e a acessibilidade/mobilidade.

A figura do arquitecto serve de mediador entre estas diversas áreas disciplinares e permite a articulação entre elas através dos documentos visuais do projecto. Neste contexto, os arquitectos e os engenheiros têm a responsabilidade de articular toda a informação necessária à execução/construção do edifício, garantindo a coerência entre todos os documentos mesmo que desenvolvidos por diferentes entidades independentes. Para evitar as incoerências nos documentos é necessário comunicação frequente e actualização em tempo real da informação para todas as entidades envolvidas no projecto.

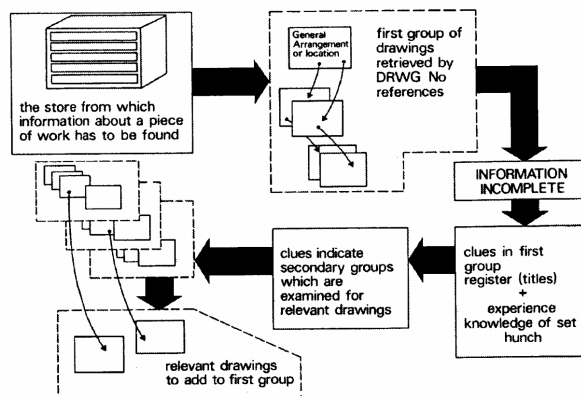


Fig 1 - Métodos tradicionais para consultar informações/documentos relativos a um projecto
Referência: Michael Kilkelly

Projecto Digital:

Os computadores modernos apareceram através de uma série de evoluções tecnológicas desenvolvidas principalmente no século XX, período em que diversos investigadores começaram também a experimentar o potencial dos sistemas digitais aplicados à geometria e consequentemente estes sistemas adaptaram-se às necessidades da arquitectura. Em 1963, surge o primeiro programa de computador para criar formas geométricas (numa metodologia CAD), desenvolvido no MIT por Ivan Sutherland no desenvolvimento do seu trabalho de Doutoramento, o Sketchpad. Os sistemas digitais ditaram um novo período na disciplina da arquitectura, o período digital e com ele trouxeram novas técnicas e metodologias de trabalho. Este período pode ser dividido em 3 eras segundo Robert Aish em 2013 "The 2D Drafting era, the Building Information Modeling era and the Design Computation era"¹⁰.

A primeira era caracteriza-se pela utilização do software CAD para produzir de forma mais eficiente o mesmo tipo de documentos das metodologias de trabalho em papel. A segunda era é definida pela utilização da metodologia BIM pelos benefícios e vantagens que oferece relativamente ao CAD. A terceira era explora o uso de algoritmos para desenvolver de forma paramétrica os projectos de arquitectura num processo em que o arquitecto desenvolve scripts para criar o projecto num processo comparável a uma receita de culinária (metáfora que será melhor explicada no módulo 3.4 - AAD).

Relação entre os instrumentos de trabalho e a eficiência do projecto:

Durante a evolução da disciplina de projectar é possível observar que ao desenvolver os seus instrumentos de trabalho, o projectista consegue realizar projectos mais complexos, ou seja, quanto melhor a tecnologia maior a capacidade de projectar. Os instrumentos de trabalho aumentaram imenso a capacidade de projectar dos arquitectos na actualidade, no entanto estes instrumentos digitais ainda não permitem a utilização mais eficiente/otimizada do computador porque os softwares utilizados são em si mesmo sistemas complexos e isolados que nem sempre oferecem compatibilidade com outros softwares de natureza semelhante. Para resolver este tipo de dificuldades, foram inventadas/introduzidas soluções tais como formatos de armazenamento de informação universais como por exemplo o ".ifc" (International Foundation Classes).

¹⁰ HUMPPPI, Harri; ÖSTERLUND, Toni - Algorithm-Aided BIM. 2015. p602

2.2 - O estado de eficiência actual do projecto de arquitectura

A evolução do computador durante o século XX e a sua comercialização ao público contribuiu para o desenvolvimento de diversas ferramentas de software úteis para a indústria AEC. Por este motivo, no século XXI, as tecnologias digitais dominam esta indústria e o trabalho de arquitectos e engenheiros é maioritariamente realizado com computadores digitais através de softwares de tipologia CAD ou BIM. A integração do computador não foi uniforme na arquitectura porque alguns arquitectos (que estão mais habituados ao uso do papel) mantêm uma postura mais conservadora e têm mais dificuldade em aderir ao uso de tecnologias digitais do que outros arquitectos. As gerações mais novas cresceram com um maior contacto com as tecnologias digitais do que as gerações anteriores que tiveram contacto com estes aparelhos mais tarde na sua vida. Segundo Bergin, o domínio dos sistemas digitais pode traduzir-se numa vantagem no trabalho do arquitecto porque permite novas possibilidades de concretizar e expressar a sua ideia. "Some have taken a negative stance on BIM and parametrics as they assume so much about the design process and limit any work produced to the user's knowledge of the program. This can enable a novice designer who has learned how to perform basic commands to become an incredibly prolific producer while a highly educated and experienced architect can be crippled from inexperience with a programs interface or underlying concepts. This creates a potential for a generational break line that becomes more harsh as a new technology gains market parity."¹¹

A tipologia CAD popularizou-se mais rapidamente no início desta era digital porque oferece ferramentas de trabalho semelhantes ao trabalho realizado em papel, mas com o benefício das capacidades de cálculo e automatização que o computador oferece (Undo-Redo, Arrays, Copy-Paste, entre outras). Por este motivo, a metodologia de trabalho CAD é mais intuitiva, ao contrário do BIM, em que a curva de aprendizagem é um pouco mais acentuada.

No seu livro de 2011, Eastman, Teicholz, Sacks e Liston, identificam a tarefa de projectar e construir um edifício como um processo fragmentado dependente da comunicação entre as várias entidades que nem sempre é eficiente.¹² A utilização destas ferramentas digitais optimizou a execução do projecto, mas no início do século XXI a comunicação do projecto ainda se realizava de forma muito semelhante ao uso de papel, através de representações gráficas bidimensionais. Segundo os autores,

¹¹ BERGIN, Michael - History of BIM. Architecture Research Lab. 2011) 6. p5

¹² EASTMAN, Charles M. et al. (EDS.) - BIM handbook: a guide to building information modeling for owners, managers, designers, engineers and contractors. 2nd ed ed. Hoboken, NJ : Wiley, 2011. ISBN 978-0-470-54137-1. p4

estes modos de comunicação do projecto originam erros, omissões e ambiguidades resultando em atrasos e aumentando os custos económicos na construção do edifício. Para resolver estes problemas foram criadas algumas soluções (tais como tecnologias de comunicação de documentos em tempo real e ferramentas de CAD 3D) que, apesar de melhorarem a comunicação da informação, não evitam os conflitos e ambiguidades presentes nas representações 2D. O facto das metodologias CAD trabalharem sobre modelos exclusivamente geométricos resulta num outro problema referido pelos autores, a dificuldade em realizar simulações, análises e estimativas de custos. "One of the most common problems associated with 2D-based communication during the design phase is the considerable time and expense required to generate critical assessment information about a proposed design, including cost estimates, energy-use analysis, structural details, and so forth. These analyses are normally done last, when it is already too late to make important changes."¹³

Para além disto, os autores indicam que este processo de comunicação do trabalho requer um grande número de documentos, o que dificulta a gestão do projecto e traduz um gasto significativo de recursos. Na sequência destas dificuldades e para ultrapassar os limites da metodologia CAD, cada vez mais arquitectos, engenheiros e construtores estão a aderir à metodologia BIM tal como demonstrado no trabalho desenvolvido por Michael Bergin em 2011 através de um gráfico em que se pode observar uma utilização de BIM que registou uma adoção pelas companhias de 28% em 2007, 49% em 2009 e 71% em 2012.¹⁴

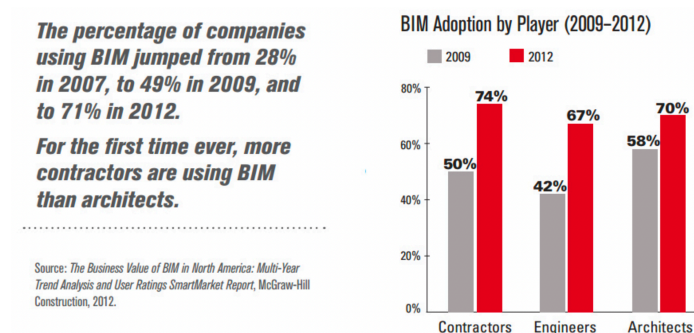


Fig 2 - Utilização de BIM na indústria AEC
Referência: Michael Bergin

¹³ EASTMAN, Charles M. et al. (EDS.) - BIM handbook: a guide to building information modeling for owners, managers, designers, engineers and contractors. 2nd ed ed. Hoboken, NJ : Wiley, 2011. ISBN 978-0-470-54137-1. p4

¹⁴ BERGIN, Michael - History of BIM. Architecture Research Lab. 2011) 6. p1

A construção de edifícios é realizada através de etapas/fases que definem os contratos e que distribuem as funções de cada entidade. Actualmente, existem alguns processos convencionais para a execução do projecto e do edifício, dos quais se destacam o Design-Bid-Build, o Design-Build e o Integrated Project Delivery.

DESIGN-BID-BUILD:

O DESIGN-BID-BUILD É um dos processos mais comuns na indústria da construção para organizar a execução do edifício. Este processo, tal como nome indica, é composto por três fases principais. Na primeira fase "DESIGN", os arquitectos são responsáveis por desenvolver um projecto para um determinado local e um determinado programa. A execução deste projecto é realizada através de diversas fases começando numa fase conceptual e terminando numa fase mais detalhada. Durante a execução do projecto, os arquitectos contratam especialistas para auxiliar a execução de partes específicas do edifício (tais como a estrutura do edifício, canalizações, sistemas energéticos e sistemas de ventilação). O edifício projectado tem que cumprir as normas e as regras estabelecidas para o local onde vai ser construído e o resultado final do projecto tem que estar devidamente documentado através de planos (plantas, cortes, alçados e axonometrias por ex). Na segunda fase "BID", o projecto (depois de concluído) é licitado a diversos construtores que vão estimar os custos da construção com base nas suas preferências e determinar um orçamento (Este orçamento inclui também a contratação de terceiros "sub-contractors"). Segundo Eastman, Teicholz, Sacks e liston, o construtor que determinar o "preço responsável" mais baixo costuma ser escolhido para realizar a obra. Na terceira fase "BUILD" o construtor geralmente tem que reeditar/corrigir os planos do edifício para os adaptar às suas metodologias de construção e para corrigir algumas inconsistências que possam dificultar a construção do edifício. Estas edições alteram relativamente o projecto inicial segundo os autores e necessitam de ser registadas e por vezes resultam em desentendimentos com as entidades projectistas. As alterações realizadas durante o processo de construção e a falta de coordenação entre os projectistas e os construtores resultam geralmente em atrasos e custos adicionais para a obra.

DESIGN-BUILD:

Um outro processo utilizado para a construção de edifícios é conhecido como "DESIGN-BUILD". Este processo tem como objectivo integrar a criação e a construção do projecto no mesmo contracto com o propósito de promover uma maior coordenação entre a entidade projectista e a entidade construtora. Segundo Eastman Teicholz Sacks e Liston, neste processo, o cliente assina um contrato com uma entidade com capacidade de projectar e construir. Esta entidade vai realizar um projecto que visa servir as necessidades do cliente e quando este estiver de acordo com o projecto, o construtor vai estimar o orçamento da obra e o tempo necessário para a construir. A entidade construtora vai contratar especialistas para efectuar os detalhes do projecto e sub-contratar terceiros para executar tarefas construtivas. Segundo os autores, este modelo/processo de construção de edifícios DESIGN-BUILD permite reduzir alguns custos associados à comunicação do projecto.

INTEGRATED PROJECT DELIVERY:

"INTEGRATED PROJECT DELIVERY" é um procedimento que evoluiu com a integração dos softwares BIM na indústria AEC e que consiste na utilização do modelo IFC como meio de comunicação entre os vários participantes do projecto. Os programas de BIM oferecem a capacidade de trabalhar no modelo IFC em 3D e também contêm ferramentas de trabalho que permitem vários participantes trabalharem de forma colaborativa neste modelo IFC (o modelo IFC é frequentemente actualizado nos diversos computadores para que todos os participantes trabalhem sobre as alterações mais recentes). Para além disto, a metodologia de trabalho BIM especialmente em 3D evita as ambiguidades e as omissões geométricas frequentemente encontradas nos planos 2D. Segundo Bergin, programas como o Autodesk Revit começaram a implementar soluções de trabalho colaborativo desde a versão 6 (publicada no ano de 2004) que permitem distribuir a posse dos diversos objectos do modelo IFC entre os participantes/projectistas do projecto através de atributos, o que promoveu o trabalho colaborativo entre arquitectos, engenheiros civis e engenheiros mecânicos neste software.¹⁵ As capacidades de trabalho colaborativo provenientes da metodologia BIM têm vindo a solucionar o problema de fragmentação na indústria AEC que Eastman, Teicholz, Sacks e Liston referiram, porque promove a comunicação e a coordenação mais fluida entre as diferentes entidades participantes do projecto. Esta vantagem tem movido a indústria AEC a utilizar cada vez mais o procedimento "Integrated project delivery" em vez do "Design-Bid-Build".

¹⁵ BERGIN, Michael - History of BIM. Architecture Research Lab. 2011) 6. p5

A metodologia BIM e o modelo IFC representam actualmente uma presença dominante na indústria AEC pela sua eficiência e pelas possibilidades que oferecem. Esta metodologia tem sido fortemente desenvolvida nos últimos anos por diversas entidades e diversos softwares que melhoram as capacidades de análise ou o trabalho colaborativo por exemplo. Existem actualmente novas tendências nas tecnologias de execução do projecto. Algumas dessas tendências prendem-se com a utilização de algoritmos em articulação com o BIM, com o objectivo de utilizar capacidade de processamento do computador para criar soluções inovadoras, dificilmente reproduzidas de forma manual.

Módulo 3

Contribuição das metodologias de trabalho digital
para a eficiência do trabalho

3.0 - Descrição do Módulo 3

O módulo 3 tem como objectivo compreender/descrever de que forma as tecnologias digitais e os seus softwares melhoraram a eficiência do trabalho dos arquitectos no desenvolvimento do projecto. Para tal, este módulo vai ser dividido em seis blocos.

A primeiro bloco (3.1) vai estudar a evolução dos sistemas digitais e de que forma estes começaram a ser integrados no projecto de arquitectura

O segundo bloco (3.2) vai estudar a metodologia CAD começando por descrever a sua origem, determinar as suas vantagens e desvantagens e por fim descrever alguns softwares exemplares.

O terceiro bloco (3.3) vai estudar a metodologia BIM começando por descrever origem desta ideologia, determinar de que forma o BIM optimizou a execução do projecto e, por fim vai descrever alguns softwares exemplares.

O quarto bloco (3.4) vai estudar a metodologia AAD com o objectivo de perceber o contexto desta ideologia, determinar como a parametrização pode ser aplicada eficientemente na execução do projecto e por fim vai descrever alguns softwares exemplares.

O quinto bloco (3.5) vai estudar a metodologia AABIM com o objectivo de perceber o potencial em articular a parametrização orientada a objectos, determinar como a esta metodologia pode revolucionar a execução do projecto e por fim vai descrever alguns softwares exemplares.

O sexto bloco (3.6) vai estudar a origem da Programação Escrita identificando alguns conceitos base com o objectivo de perceber o potencial em associar esta disciplina com a disciplina do projecto abrindo novas possibilidades de trabalho e também vai descrever algumas linguagens de programação exemplares.

A intenção deste módulo é sobretudo clarificar os conceitos principais de algumas metodologias de trabalho. Estes conceitos vão ser utilizados para se perceber melhor o conteúdo presente nos módulos seguintes.

3.1 - A evolução dos sistemas digitais

A origem dos sistemas digitais:

Os computadores são engenhosas máquinas de cálculo que evoluíram ao longo de anos, principalmente no século XX. Estes modelaram a realidade actual do século XXI, estando presentes em vários aspectos do quotidiano da sociedade actual. Eles permitem grandes capacidades de automatização e cálculo (úteis para a gestão de grande quantidade de informação em tempo real) e, por esse motivo foram adoptados como instrumentos de trabalho em diversas áreas e por diversas profissões. Para compreender melhor o funcionamento destas máquinas, é interessante perceber de que forma é que elas evoluíram até à forma como são conhecidas actualmente.

O ábaco é reconhecido como o primeiro computador utilizado pelo ser humano, inventado há 5000 anos, ele permitiu executar cálculos simples de contagem. Apenas a partir do século XVII é que houve evoluções significativas desta tecnologia, a partir do momento em que Blaise Pascal inventou uma calculadora mecânica, em 1642, que funcionava com rodas dentadas e permitia fazer somas (a "Pascaline") e mais tarde em 1820, Charles Xavier Thomas de Colmar inventou o "Aritmómetro", um aparelho mecânico com a capacidade de executar diferentes operações aritméticas.

No entanto, foi em 1812 que o matemático inglês Charles Babbage viu o potencial de associar as máquinas mecânicas à matemática porque percebeu que estas têm a capacidade de executar a mesma tarefa repetidamente com precisão, o que podia ser benéfico para analisar a grande quantidade de informação presente nas tabelas. Em 1822, Babbage inventou uma máquina diferencial (a vapor) que tinha um programa pré-definido capaz de resolver cálculos instantaneamente e imprimir os seus resultados (output). Posteriormente, Charles Babbage juntamente com Augusta Ada King desenvolveram uma máquina analítica que permitia receber cartões perfurados com instruções de execução/operação através de um dispositivo de entrada (input), a partir dos quais executava/processava os cálculos e imprimia os resultados (output). "A máquina de Babbage era constituída por cerca de 50 000 componentes e, no seu desenho básico, incluía dispositivos de entrada de cartões perfurados contendo as instruções de operação (...) Tinha também uma unidade de controlo que permitia processar instruções em qualquer sequência e dispositivos de saída que produziam resultados impressos."¹⁶

¹⁶ MAGALHÃES, José - Hardware Tecnologias e Soluções. FCA-Editora de Informática Lda ed. Lisboa : Lidel - Edições Técnicas Lda, 2019. ISBN 978-972-722-892-8. p4

A utilização de cartões perfurados foi uma ideia introduzida por Joseph Marie Jaccquard em 1820 para definir os padrões a serem tecidos pela sua máquina de tear. A utilização dos cartões perfurados foi posteriormente utilizado em 1889 por Herman Hollerith para processar os dados do census de 1890 nos EUA através de tabuladores electro-mecânicos. Ao contrário de Babbage que utilizou os cartões para inserir na máquina as instruções/programa de como processar os dados, Hollerith utilizou os cartões para armazenar os dados, que seriam posteriormente inseridos na máquina para serem processados.

Em 1896 Hollerith fundou a "Tabulating Machine Company" para comercializar o leitor de cartões perfurados. Em 1911, esta companhia foi juntamente com outras 3 companhias combinada na CTR (Computing-Tabulating-Recording Company) mais tarde renomeada para IBM (International Business Machines). O sistema de cartões foi utilizado para inserir e retirar dados nas máquinas de computação (input/output) até à década de 1960. "Em vez de, como Babbage, os usar para dar instruções à máquina, Hollerith olhou para os cartões como um sistema de armazenamento de dados (...) Cada furo no cartão representava um número e a combinação de dois furos uma letra. (...) em 1896, fundou a Tabulating Machine Company, que mais tarde se transformou na nossa conhecida International Business Machine (IBM)."¹⁷

A primeira geração de computadores:

A Segunda Guerra Mundial potenciou o desenvolvimento de diversas tecnologias, uma das quais, os sistemas digitais. Konrad Zuse desenvolveu o Z3, em 1941, um computador que foi utilizado para a criação de aviões e mísseis. Posteriormente, Howard Aiken desenvolveu um computador capaz de executar cálculos de balística. Este computador de enormes dimensões tinha 23 metros cúbicos, uma altura de 2,4 m uma profundidade de 60 cm e o comprimento 16 m. Por esta altura foi também inventado o ENIAC (Electronic Numerical Integrator And Computer) que utilizava válvulas a vácuo, resultado de uma colaboração entre a universidade da Pensilvânia e o governo dos EUA. John von Neumann desenvolveu, em 1945, o EDVAC (Electronic Discrete Variable Automatic Computer) que se destacava por ter uma unidade central de processamento (CPU), isto permitiu que um único componente pudesse gerir todas as funções do computador. Seis anos depois, Remington Rand desenvolveu o UNIVAC (Universal Automatic Computer), o primeiro computador disponibilizado para uso comercial. Esta geração de computadores eram geralmente programada com códigos binários (0's e 1's), tarefa demorada e complexa.

¹⁷ MAGALHÃES, José - Hardware Tecnologias e Soluções. FCA-Editora de Informática Lda ed. Lisboa : Lidel - Edições Técnicas Lda, 2019. ISBN 978-972-722-892-8. p4

A segunda geração de computadores:

A segunda geração de computadores foi marcada pela invenção do transistor em 1948, que substituiu as válvulas a vácuo. Esta mudança permitiu que os computadores pudessem ter dimensões mais compactas, ser mais rápidos e mais eficientes energeticamente. "Com a invenção do transistor, em 1948, o desenvolvimento dos computadores alterou-se completamente. O transistor veio substituir as grandes válvulas a vácuo nos aparelhos eletrônicos. Como resultado imediato, o tamanho dos aparelhos eletrônicos começou a diminuir."¹⁸ Os LARC foram os dois primeiros computadores a tirar partido desta transição. Foi durante esta geração de computadores que apareceu a linguagem Assembly, que pode ser muito mais facilmente interpretada (por Humanos) do que o código binário. A década de 1960 marca uma grande comercialização dos computadores para servir grandes empresas privadas, empresas governamentais e instituições universitárias. Estes equipamentos eram fornecidos por companhias privadas, nomeadamente a IBM, a Burroughs, a Control Data e a Sperry. Foi também durante este período que começaram a ser desenvolvidos equipamentos como por exemplo impressoras, fitas magnéticas, discos de armazenamento, memórias, sistemas operativos e programas armazenados dentro do próprio computador. Durante este período apareceram as primeiras linguagem de programação High-Level (como por exemplo o COBOL e o FORTRAN) que permitiam uma linguagem mais fluída, intuitiva e compreensível ao ser Humano, resultando num processo de programação mais eficiente e rápido.



Fig 3 - IBM 1401
Referência:
José Magalhães

¹⁸ MAGALHÃES, José - Hardware Tecnologias e Soluções. FCA-Editora de Informática Lda ed. Lisboa : Lidel - Edições Técnicas Lda, 2019. ISBN 978-972-722-892-8. p6

A terceira geração de computadores:

Neste período, os computadores ainda eram máquinas de consideráveis dimensões que dependiam de grandes quantidades de componentes eletrónicos interligados por quilómetros de cabos, o que resultava numa grande dificuldade na manutenção e durabilidade/robustez destas máquinas electrónicas. Com o intuito de resolver este problema, o engenheiro Jack Kilby desenvolveu em 1958 o primeiro circuito integrado onde ele provou que era possível utilizar uma placa de um material semi-condutor para articular diversos componentes eletrónicos numa abordagem híbrida (Hybrid Integrated Circuits). Esta invenção viria mais tarde a ser melhorada por Robert Noyce que apostou numa abordagem mais prática (Monolithic Integrated Circuit). Estas invenções permitiram que os cientistas pudessem compactar os componentes do computador através dos circuitos integrados e como resultado, os computadores podiam ter dimensões mais reduzidas.

Este período foi também marcado pelo desenvolvimento de sistemas operativos. "que permitiu às máquinas correrem muitos programas diferentes simultaneamente com um programa central que monitoriza e coordena a memória do computador"¹⁹

Em 1965, Kristen Wygaard e Ole-John Dahl inventaram a primeira linguagem de programação orientada a objectos (Object-Oriented Programming Language), que apelidaram de SIMULA.

Quatro anos depois, Kenneth Thompson e Dennis Ritchie, inventaram o UNIX, um sistema operativo que foi mais tarde utilizado como base/fundação do Linux e do MacOS, dois dos sistemas operativos mais utilizados actualmente.

A quarta geração de computadores:

A evolução dos circuitos integrados durante as décadas seguintes (1970 e 1980) resultou na criação de componentes eletrónicos mais compactos e eficientes e contribuiu para a redução do tamanho e do preço dos computadores. Para além disso, em 1971, a Intel inventou o revolucionário microprocessador i4004. Estas evoluções contribuíram para que o computador pudesse ser comercializado para um público geral para além das grandes empresas. A quarta geração foi marcada pelo aparecimento do PC (Personal computer), computadores de tamanho e preço acessíveis que eram vendidos ao utilizador individual como por exemplo, o Apple II, o Sinclair ZX80, o Sinclair Spectrum, o IBM PC e Osborne 1. Esta geração foi também marcada pelo Apple Lisa, Apple Macintosh e Commodore Amiga que se destacaram por terem uma interface gráfica que permitia ao utilizador executar tarefas (como imprimir um documento) utilizando um rato em vez de escrever comandos para tal.

¹⁹ MAGALHÃES, José - Hardware Tecnologias e Soluções. FCA-Editora de Informática Lda ed. Lisboa : Lidel - Edições Técnicas Lda, 2019. ISBN 978-972-722-892-8. p8

Foi no Apple Lisa que em 1984 foi criado o RadarCH (mais tarde renomeado para Archicad), um dos primeiros programas de BIM a ser comercializado para utilização em arquitectura até à data de hoje.²⁰

O aparecimento do Personal Computer foi uma revolução importante na Arquitectura porque mudou o paradigma do trabalho manual (realizado em folhas de papel) para o trabalho digital (realizado maioritariamente em computadores).

Actualmente a utilização de computadores para realizar o projecto é quase imperativa no trabalho dos arquitectos porque estes aparelhos permitem realizar o projecto de maneira eficiente e oferecem possibilidades e metodologias de trabalho impensáveis há décadas atrás.



Fig 4 - Apple Macintosh Plus 1Mb
Referência: <https://www.bukowskis.com/fi/lots/843598-dator-macintosh-plus-1mb-apple-med-tillbehor-sent-1980-tal>

²⁰ BERGIN, Michael - History of BIM. Architecture Research Lab. 2011) 6. p3

3.2 - Computer-Aided Design (CAD)

Breve descrição do CAD:

A metodologia CAD refere-se à utilização de sistemas digitais/computadores para criar modelos geométricos bidimensionais ou tridimensionais. Estes modelos geométricos são geralmente desenvolvidos em softwares dedicados e consequentemente armazenados em formatos específicos através de ficheiros (como por exemplo: .3dm .dae .dxf .fbx .obj .svg). Os modelos CAD armazenam informação relativa à geometria, textura, e animações dos objectos. Na metodologia CAD cada objecto refere-se a elementos geométricos como pontos, arestas, superfícies, sólidos, Meshes ou NURBS por exemplo.

Origem do CAD:

Em 1957, Patrick J. Harratty, um programador da General Electric, desenvolveu o primeiro programa de tipologia CAM, o PRONTO (Programme for Numerical Tooling Operations), que tinha como objectivo controlar máquinas industriais para criar peças de forma mecânica num processo conhecido como numerical control. Apesar de este ser o primeiro programa de computador conhecido a trabalhar com geometria, ele era operado com folhas de instruções e não tinha um interface visual interactivo que permitisse a criação de formas geométricas de uma forma eficiente e intuitiva que pudesse ser utilizado por designers e projectistas. "One of Hanratty's early projects while at GE was the development of an early numerical control software package called PRONTO (Programme for Numerical Tooling Operations). It basically took machining statements that an NC specialist entered onto coding sheets and processed these statements in a batch mode after they had been keypunched into punch cards and produced the actual digital instructions the machine tool controller needed to operate the machine tool. PRONTO was a point-to-point NC package developed for the Kearney & Trecker Milwaukee 3 machine tool used at GE's Schenectady, NY facility"²¹

Apenas em 1963, Ivan Sutherland, um cientista de computadores do MIT, criou o primeiro programa de tipologia CAD com um interface gráfico interactivo que apelidou de Sketchpad. Este programa conseguia realizar operações de cálculo geométrico consideravelmente avançadas e permitia que o seu utilizador pudesse criar formas geométricas de uma forma intuitiva apontando uma caneta de luz ao display/ecrã do TX-2, (um computador do laboratório do MIT). A criação de Ivan Sutherland é considerada uma pioneira nesta área tecnológica e demonstrou o

²¹ WEISBERG, David E. - Patrick Hanratty and Manufacturing & Consulting Services. Em The Engineering Design Revolution [Em linha] Disponível em WWW:<URL:www.cadhistory.net>. 2008 p15-1

potencial do uso de software para beneficiar o trabalho de designers e projectistas como arquitectos e engenheiros, contudo, o Sketchpad foi desenvolvido antes do aparecimento do computador pessoal e o acesso a este programa era portanto muito restrito/exclusivo. "Sketchpad was the first program that contained a graphical user interface (GUI). By using a light pen, users were able to draw directly on the CRT (cathode-ray tube) screen and that was the first time that engineering drawings could be drawn directly on a computer display. Sketchpad was developed and run on the TX-2"²²

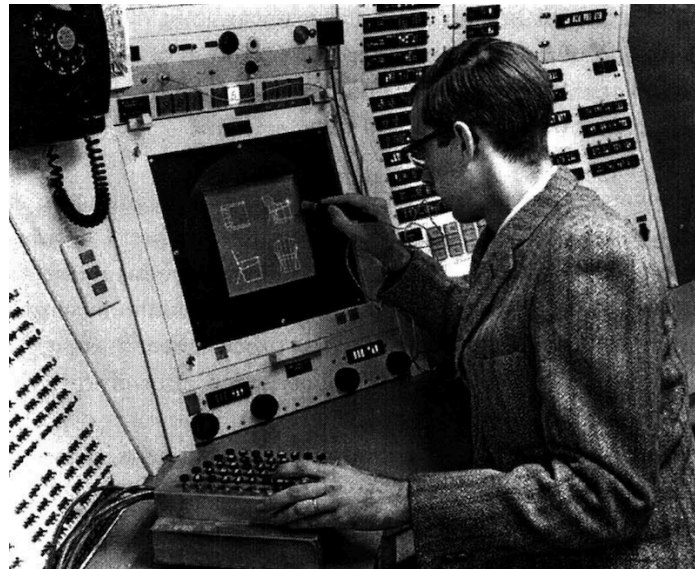


Fig 5 - Ivan Sutherland a utilizar o Sketchpad no MIT (1963)
Referência: Michael Kilkelly

O software CAD para uso comercial começou a ser desenvolvido sensivelmente a partir da década de 70 quando a indústria automóvel e de aviação começou a explorar a utilização de sistemas digitais para desenvolver os seus projectos. Segundo Weisberg²³, empresas como a Renault, a Dassault Systèmes e a Ford desenvolveram alguns dos primeiros softwares CAD para uso comercial. "In 1977 Avions Marcel Dessault started the development of a 3D CAD program called

²² CVETKOVIĆ, Ivana et al. - Historical Development of Computer-Aided Design (Cad) and Its Application in Education at Faculty of Mechanical Engineering in Belgrade. Em Proceedings of the International Scientific Conference - Sinteza 2019 [Em linha]. Novi Sad, Serbia : Singidunum University, 2019 [Consult. 9 abr. 2022]. Disponível em WWW:<URL:<http://portal.sinteza.singidunum.ac.rs/paper/644>>. ISBN 978-86-7912-703-7. p60

²³ WEISBERG, David E. - A Brief Overview of the History of CAD. Em The Engineering Design Revolution [Em linha] Disponível em WWW:<URL:www.cadhistory.net>. 2008. pp 2-8 / 2-9

CATI. (...) In 1981, Avions Marcel Dassault created Dassault Systèmes in order to develop and market 3D CAD software CATIA V1 (first commercial version of CATI)"²⁴
Este programa tornou-se um marco nesta tipologia pelas suas capacidades inovadoras e serviu como base de desenvolvimento de outros programas com funções semelhantes tal como afirmado por Michael Bergin. "Digital Project is a similar program was developed by Gehry Technologies around 2006 based on CATIA, a design program (and one of the first CAD programs) that was developed as an in house project by Dassault systems, a French airplane manufacturer."²⁵

Nas décadas seguintes, foram desenvolvidos diversos softwares da tipologia CAD, que apesar de manterem uma componente comum, diferem nas suas características e metodologias para conseguirem melhorar a eficiência e a rapidez de trabalho. Por norma, a enorme diversidade de funções/utensílios/ferramentas que cada um tem, obriga a uma boa organização do espaço de trabalho (GUI "Graphical User Interface") agrupando as diversas funções/ferramentas por categorias, menus e sub-menus. A metodologia para aceder às ferramentas/instrumentos do programa tornou-se também um ponto importante porque estes programas têm geralmente muitas ferramentas que precisam de ser acedidas pelo projectista rapidamente. Para dar resposta a esta questão foram desenvolvidas duas posturas teóricas.

A primeira postura teórica usa o teclado como fonte de input para aceder às ferramentas de forma rápida, escrevendo as primeiras letras da ferramenta pretendida (as ferramentas mais frequentes aparecem primeiro necessitando geralmente apenas de uma letra seguida da tecla Enter ou Spacebar). Nesta primeira postura destacam-se softwares como por exemplo o AutoCad ou o LibreCad.

A segunda postura teórica prioriza o rato como fonte de input tendo em conta que este elemento é muito flexível no ambiente de trabalho podendo adquirir inúmeras funções. Por cada ação do rato, aparece um menu local com as ferramentas de edição no display. Nesta segunda postura destacam-se softwares como por exemplo o MicroStation da Bentley Systems.

Muitos programas da tipologia CAD oferecem também a opção de atribuir Comandos/Ferramentas/Funções às teclas do teclado principal ou externo. Esta característica é conhecida como "Macros" ou "Hotkeys" e o objectivo é tornar cada uma das ferramentas do GUI em botões de teclado físicos para um rápido acesso.

²⁴ CVETKOVIĆ, Ivana et al. - Historical Development of Computer-Aided Design (Cad) and Its Application in Education at Faculty of Mechanical Engineering in Belgrade. Em Proceedings of the International Scientific Conference - Sinteza 2019 [Em linha]. Novi Sad, Serbia : Singidunum University, 2019 [Consult. 9 abr. 2022]. Disponível em WWW:<URL:<http://portal.sinteza.singidunum.ac.rs/paper/644>>. ISBN 978-86-7912-703-7. p60

²⁵ BERGIN, Michael - History of BIM. Architecture Research Lab. 2011) 6. p5

Esta postura assemelha-se um pouco à ideia de um cockpit de avião em que cada botão tem uma função específica.

É também relevante destacar a importância da criação de softwares Open-Source que possibilitam a utilização gratuita de CAD a todos os utilizadores, contribuindo para a partilha e o livre acesso de conhecimento. Enquanto que uma grande percentagem do software CAD é distribuído comercialmente (por um preço nem sempre acessível), o software Open-Source tem como objectivo fazer chegar as ferramentas de trabalho criativas ao público em geral e ser disponibilizado/acedido gratuitamente por todos. Como o software Open-Source não é financiado por vendas, ele depende geralmente de doações (não obrigatórias) de dinheiro para suportar os custos de manutenção e desenvolvimento. Os utilizadores do software Open-Source podem também contribuir voluntariamente para ajudar a desenvolver este software (escrevendo módulos/scripts ou traduzindo a linguagem por exemplo). Para além disto, o software é considerado Open-Source quando o código interno é igualmente disponibilizado livremente para quem o quiser analisar, editar ou complementar. Numa analogia simples de perceber, o software Open-Source funciona como um edifício de acesso livre/gratuito e os planos do seu projecto de arquitectura estão igualmente disponíveis para quem os quiser consultar, desenvolver ou fazer semelhante. Por outro lado, o software comercialmente distribuído funciona como um edifício cujo acesso é pago e os planos do seu projecto de arquitectura não estão livremente disponíveis para serem consultados.

Neste contexto destacam-se dois programas da tipologia CAD Open-Source, o LibreCad e o FreeCad. O LibreCad permite criar formas geométricas bidimensionais e tem um GUI intuitivo. O FreeCad oferece mais possibilidades, no entanto possui uma curva de aprendizagem mais íngreme, tendo em conta a grande quantidade de capacidades/funções que oferece e a possibilidade de trabalhar com parametrização.

Contribuição para a eficiência do projecto:

O software CAD tornou-se uma ferramenta de trabalho útil para desenvolver projectos porque oferece formas intuitivas e práticas de aplicar o poder de cálculo dos computadores em benefício da geometria e do projecto. Por este motivo, as ferramentas digitais (como o LibreCAD ou o SketchUp) conseguem executar o trabalho convencional mais eficientemente (com maior rapidez e precisão) do que as ferramentas de trabalho convencionais (lápiz e régua).

O domínio de ferramentas CAD mais complexas permite a criação de formas mais complexas, expandindo a capacidade geométrica do projectista. A utilização de hardware como impressoras 2D ou 3D permite a replicação rápida dos documentos e dos modelos físicos do projecto.

Diversas ferramentas CAD oferecem a possibilidade de trabalhar em 3 dimensões, esta capacidade permite que os utilizadores tenham uma melhor percepção volumétrica do edifício que estão a projectar e ajudam a detetar conflitos geométricos mais facilmente. Esta característica evita que os projectistas tenham que fazer maquetes demoradas de construir. Para além disso, os modelos 3D digitais são mais fáceis de alterar do que as maquetes físicas.

A metodologia de trabalho CAD tem geralmente uma curva de aprendizagem mais leve porque as suas funções/ferramenta têm uma proximidade conceptual aos métodos de trabalho convencionais/físicos, no entanto o grau de dificuldade pode variar entre os diversos softwares de CAD.

A capacidade de replicar os objectos é um dos grandes benefícios da capacidade de cálculo do computador e mostra-se vantajosa especialmente em situações repetitivas frequentemente encontradas nos edifícios.

No entanto, também existem algumas dificuldades associadas ao uso de software CAD. Por exemplo, Jeroen L. Coenders afirma que o CAD apresenta dificuldades em descrever o pensamento do projectista e o processo lógico na concepção da forma geométrica, o que tem importância no trabalho de grupo onde a lógica do processo de trabalho precisa de ser comunicada. "As demonstrated CAD (computational geometry) and BIM (three dimensional object-oriented description) have not been able to form this integral information model due to their restrictions to describe the logic (and justification) of the design."²⁶

Um outro problema também frequentemente associado ao trabalho digital é a capacidade de processamento dos computadores. Embora o cálculo de objectos geométricos seja rápido, a grande quantidade destes objectos pode tornar-se demorada e difícil de calcular, sendo necessário algoritmos mais eficientes ou computadores mais potentes. Os softwares CAD armazenam a geometria dos objectos geométricos de forma matemática em ficheiros de computador (como .dxf .svg .dae .skp por exemplo). Os ficheiros vectoriais permitem armazenar grandes quantidades de informação em formatos relativamente compactos, no entanto quando abertos nos softwares, estes ficheiros precisam de ser constantemente recalculados, sendo necessário uma grande capacidade de processamento do computador.

²⁶ COENDERS, Jeroen L. - Next Generation Parametric Design. Journal of the International Association for Shell and Spatial Structures. . ISSN 1028-365X. 62:2 (2021) 153–166. doi: 10.20898/j.iass.2021.018. p156

A metodologia CAD armazena apenas informação relativa à geometria, shaders ou animações, por este motivo, esta metodologia tem portanto alguns limites no tipo de informação que pode armazenar. Por outro lado, o BIM consegue armazenar mais atributos por cada elemento do modelo tais como resistência ao fogo, preço por volume ou o fabricante. Por estes motivos, o software CAD não armazena informações suficientes para a execução de análises (térmicas ou estruturais por exemplo).

Alguns exemplos de softwares CAD destacados:

SketchUp:

O SketchUp é um programa de modelação geométrica 3D que se destaca pela sua simplicidade e intuitividade. Ele oferece ferramentas práticas de utilizar e, por esse motivo, este programa permite executar ideias e criar objectos de forma rápida e fluída. Neste software trabalha-se quase exclusivamente em 3D e, por isso, os seus utilizadores têm uma boa percepção volumétrica do modelo que estão a desenvolver. O SketchUp tem uma comunidade bastante extensa que contribui para o seu desenvolvimento através de plugins ou de partilha de modelos na 3D Warehouse (Um arquivo de modelos 3D online exclusivo aos utilizadores do SketchUp). Para além disso, este software oferece um API em Ruby para um maior controlo. O Sketchup foi criado em agosto de 2000 pela Last Software e ganhou prestígio pouco depois de ser criado. Em 2006, a Google adquiriu o Sketchup com o objectivo de o integrar com o Google Earth. Entre 2007 e 2010 a Google publicou o Sketchup 6, 7 e 8, para os sistemas operativos Windows e MacOS, oferecendo uma versão gratuita e uma versão paga (versão Pro).

Em 2012, a Trimble Inc. adquiriu o Sketchup à Google e em 2013 foi lançado o Sketchup 2013. O modelo de vendas foi alterado diversas vezes e actualmente, o Sketchup é distribuído em 5 edições diferentes (Sketchup Studio / Sketchup Pro / Sketchup Shop / Sketchup Free / Sketchup Make). Este software é actualmente comercializado na forma de subscrição paga, não estando disponível uma edição à venda por pagamento único.

Rhinoceros:

O Rhinoceros 3D é um programa de CAD comercialmente distribuído por Robert McNeel and Associates. Este programa destaca-se por oferecer diversas ferramentas de trabalho avançadas para a criação de formas geométricas complexas e precisas. As suas ferramentas permitem criar de forma eficiente modelos geométricos 2D e 3D.

O Rhinoceros oferece um API de Python bastante completo e desde a versão 5 integra o GrassHopper, um plugin que permite a criação de modelos geométricos

paramétricos. O Rhinoceros tem uma grande comunidade de utilizadores e, por esse motivo existem diversos plugins (oficiais ou desenvolvidos por terceiros) que prolongam as capacidades deste software.

Actualmente, este software funciona em Windows e também em MacOS (desde a versão 5 em 2015). O produto pode ser utilizado sob 4 tipos de licenças: a licença gratuita (que oferece 90 dias de utilização), a licença educacional (disponível por 195\$), a licença comercial (995\$) e a licença para instituições escolares (195\$ a 975\$).

FreeCAD:

O FreeCAD é um software Open-Source criado por Jürgen Riegel e desenvolvido por Werner Mayer, Yorik van Havre e pela comunidade do FreeCAD. Este programa permite a criação de modelos geométricos em 2D e 3D. A sua interface está dividida por módulos "workbenches" que permitem realizar diversas tarefas como CAD 2D/3D, BIM, análises FEM e capacidades paramétricas por exemplo. O FreeCAD possui uma grande comunidade de utilizadores que ajudaram a desenvolver este software criando novas ferramentas como por exemplo o workbench de BIM. O número de ferramentas que o FreeCAD oferece dificultam a complexidade do software e aumentam a curva de aprendizagem mas possibilitam diversas capacidades de trabalho. Este software pode ser comparado com o SolidWorks da Dessault Systèmes no seu modo de operar e funciona em três sistemas operativos, Linux, MacOS e Windows.

LibreCAD:

O LibreCAD destaca-se por ser Open-Source e é distribuído gratuitamente sob a licença GPLv2. Este programa permite desenvolver modelos geométricos em 2D e oferece um interface simples e intuitiva de perceber, mas não tem a capacidade de desenvolver ou trabalhar com modelos 3D. O LibreCAD começou com o objectivo de adicionar capacidades CAM no QCAD e funciona em três sistemas operativos, Linux, MacOS e Windows.

3.3 - Building Information Modeling (BIM)

Breve descrição do BIM:

Building Information Modeling é uma metodologia de trabalho digital que consiste na criação de um modelo que contém as informações e os atributos de cada objecto, sendo por isso também referida como modelação orientada a objectos. "three dimensional object-oriented description"²⁷ Ao contrário do CAD, o modelo BIM armazena uma grande variedade de informações acerca de cada objecto tais como a geometria, o material do objecto, o fabricante, o volume, o preço, a resistência ao fogo ou a classe/tipologia de cada objecto. Na metodologia BIM cada objecto refere-se a elementos mais abstractos/complexos do edifício como por exemplo, vigas, paredes, janelas ou quartos.

Origem do BIM:

O aparecimento do BIM está relacionado com os desenvolvimentos realizados na área dos sistemas informáticos a nível de software e hardware posteriormente à Segunda Guerra Mundial. Segundo Michael S. Bergin, já em 1962 Douglas C. Englebart descreve uma nova forma de projectar com base em objectos organizados parametricamente.²⁸ Bergin menciona também o trabalho de Christopher Alexander que contribuiu para a criação da programação orientada a objectos (conceito utilizado frequentemente por algumas linguagens de programação actuais).

Estes desenvolvimentos teóricos contribuíram para o aparecimento do BIM, contudo, segundo Bergin, para se chegar ao conceito do BIM ainda era necessário uma interface gráfica através da qual, o utilizador pudesse visualizar e manipular o modelo geométrico. É neste contexto que ele refere a importância do Sketchpad de Ivan Sutherland como o software pioneiro na utilização de interfaces gráficos para a criação de geometria como uma alternativa mais intuitiva e rápida do que o uso de programação escrita. Para além disso, Bergin refere dois processos utilizados por softwares desta tipologia para calcular e representar a geometria digitalmente, Constructive Solid Geometry (CSG) e Boundary Representation (BREP). O processo CSG permite a criação de formas geométricas através de operações booleanas nos sólidos geométricos primitivos enquanto que o processo BREP representa formas geométricas pelos seus limites. Ele refere que o desenvolvimento de hardware, nomeadamente Human-Computer Interaction (HCI) desenvolveu a eficiência da comunicação entre o utilizador e a máquina digital.

²⁷ COENDERS, Jeroen L. - Next Generation Parametric Design. Journal of the International Association for Shell and Spatial Structures. . ISSN 1028-365X. 62:2 (2021) 153-166. doi: 10.20898/j.iass.2021.018. p156

²⁸ BERGIN, Michael - History of BIM. Architecture Research Lab. 2011) 6. p1, p2

Posteriormente, Bergin refere a importância de uma base de dados para organizar a informação de todos os elementos do modelo geométrico e neste contexto, apresenta o BDS (Building Description System) como o primeiro programa de computador com capacidade de identificar os elementos do modelo geométrico e os seus atributos. Este software é considerado pioneiro na tipologia BIM e foi criado por Charles Eastman, um arquitecto formado em Berkeley CED, California que desenvolveu conhecimentos na ciência de computadores. O facto de Eastman ter conhecimentos de arquitectura e computadores contribuiu para que ele tivesse uma boa percepção da melhor maneira de relacionar os conhecimentos destas duas tão diferentes áreas. Eastman compreendia os benefícios e as vantagens que um software da tipologia BIM podia trazer ao projecto através da capacidade de automatizar diversas tarefas, tornando o trabalho do arquitecto mais eficiente.²⁹

```
POLY PROCEDURE spiral.step(POLY centre;
    REAL riser,radius,r,angle,th)=
BEGIN
    POLY support =
        triangle(radius*0.95,-riser*0.8,th);
    POLY collar = column(12,riser,r);
    POLY plate = wedge(radius,th,angle);
    ! return the result of shape operations;
    CUT centre FROM COMBINE collar WITH
        COMBINE support WITH plate
END;

! To make spiral staircase, (dimensions in inches)
SET PROCEDURE spiral.stair(ht,radius,angle)=
    BSET; INTEGER numsteps; REAL riser;
    numsteps = ht/8.0;
    riser = ht/numsteps;
    POLY centre = column(12,ht*32.0,5.0);
    POLY step = spiral.step(centre,
        riser,radius,3.0,angle,0.625);
    FOR i TO numsteps
        DO COPY step=10,riser*0.1,angle*0.1
    ESET;

SET stair1 = spiral.stair(100.0,46.0,30.0);
```

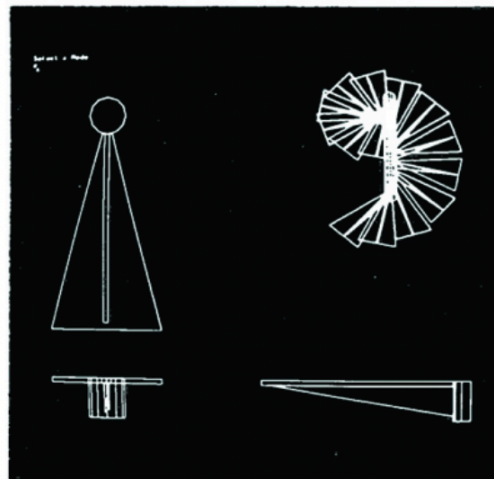


Fig 6 - Building Description System (BDS) desenvolvido por Charles Eastman
Referência: Michael Bergin

Nos anos 80 a tipologia BIM continuou a ser desenvolvida, alguns softwares desta tipologia integravam o tempo como uma quarta dimensão, outros softwares tinham a capacidade de simular a performance dos modelos geométricos. No entanto, foi nesta altura que Gábor Bojár desenvolveu o Radar CH para o sistema operativo da Apple em 1984, o primeiro software BIM comercializado e disponível para computadores pessoais. O Radar CH foi mais tarde renomeado para Archicad e representa um dos principais programas BIM actualmente.

Posteriormente ao lançamento do Radar CH, Irwin Jungreis e Leonid Raiz, começaram a desenvolver um software capaz de trabalhar com projectos complexos e em 2000 publicaram/lançaram o Revit. O Revit é um programa de tipologia BIM que tem como objectivo realizar modelos geométricos com características paramétricas.

²⁹ BERGIN, Michael - History of BIM. Architecture Research Lab. 2011) 6. p2, p3

Este programa foi programado em C++, uma linguagem de programação orientada objectos (object oriented programming language).

Segundo Bergin, o Revit revolucionou o conceito BIM porque integrava programação visual e incluía a utilização de atributos temporais para cada elemento, permitindo a visualização de um modelo geométrico nas suas diferentes fases de execução.

Por fim, Bergin fala na dificuldade que os diversos programas de computador têm em articular os diversos tipos de ficheiros, afirmando que os ficheiros podem perder características ao serem utilizados em várias plataformas diferentes. É neste contexto que foi desenvolvido em 1995 um ficheiro universal, o IFC (Industry Foundation Classes). Este ficheiro tem como objectivo principal armazenar a informação BIM num formato que possa ser interpretado por diversas plataformas/programas/softwares. Bergin afirma também que os softwares independentes de simulação como por exemplo o Ecotect, o Energy Plus, o IES e o Green Building Studio conseguem importar o modelo BIM através deste ficheiro comum, o IFC.

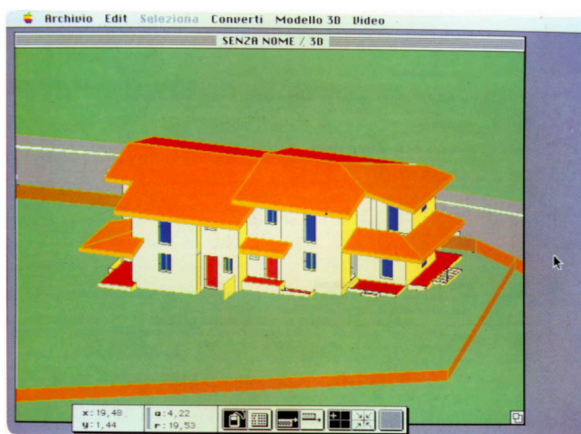


Fig 7 - Radar CH para Macintosh
Referência: Michael Bergin



Fig 8 - Logótipo do IFC
Referência: <https://www.cleanpng.com/png-industry-foundation-classes-building-information-m-6784523/>

Contribuição para a eficiência do projecto:

A utilização do BIM traduz-se certamente num grande benefício para a execução e para a qualidade do projecto de arquitectura. Esta metodologia de trabalho digital possui diversas vantagens que explicam a crescente adopção desta metodologia na indústria AEC. Embora os softwares da tipologia BIM tenham uma natureza mais complexa, estes softwares revelam a sua eficiência em fases mais avançadas do projecto através da sua capacidade de automatizar determinadas tarefas.

A utilização inteligente do modelo orientado a objectos permite otimizar uma grande quantidade de tarefas repetitivas que atrasam o desenvolvimento do projecto tais como o levantamento de quantias/orçamentos, a modelação da geometria e a articulação entre as várias disciplinas.

A criação automática de documentos (plantas, cortes, axonometrias, modelos 3d, tabelas) a partir do modelo BIM otimiza o desenvolvimento do projecto aumentando a rapidez e mantendo a precisão. A criação destes documentos de forma manual (em metodologias como o CAD) é muito mais demorada e estes documentos desactualizam com o desenvolver do projecto. Eastman, Teicholz, Sacks e Liston afirmam que o BIM é uma mudança do paradigma de trabalho que, ao automatizar/otimizar a criação de documentos gráficos, conduz o foco do trabalho para o projecto em si e não na produção de documentos como na metodologia CAD. "Unlike CADD, which primarily automates aspects of traditional drawing production, BIM is a paradigm change. By partially automating the detailing of construction-level building models, BIM redistributes the allocation of effort, placing more emphasis on conceptual design."³⁰

A metodologia BIM foi desenvolvida para a criação de edifícios e, como tal, as ferramentas digitais mimetizam conceptualmente os elementos do edifício, otimizando e facilitando a criação do modelo geométrico. Uma outra grande vantagem proporcionada pelo BIM é a capacidade que este modelo tem de armazenar diversas informações sobre cada objecto. Por este motivo, o modelo BIM pode mais facilmente ser utilizado para a realização de análises contribuindo para a interoperabilidade entre várias disciplinas.

Apesar das inúmeras vantagens promovidas pelo BIM, esta metodologia também tem os seus limites que muitos autores consideram desvantagens. É possível afirmar que o BIM tem uma curva de aprendizagem mais acentuada do que o CAD e, por isso, é frequente os seus utilizadores participarem em formações/cursos para conseguir dominar este software e aprender a utilizar a suas funções correctamente.

Tal como afirmado por Jeroen Coenders, um pouco à semelhança do CAD, o BIM também apresenta dificuldades em documentar/explicar o raciocínio lógico na criação do projecto. "One of the key issues is that Building Information Modelling (...) does not cater for analysis or simulation information (...) or design and engineering (justification) information. In other words, Building Information Modelling is able to describe a rich product model of what needs to be built and what properties this object needs to have but does not provide any information about 'the why': for example, why is the object dimensioned as it is? Or why is a particular material chosen?"³¹

³⁰ EASTMAN, Charles M. et al. (EDS.) - BIM handbook: a guide to building information modeling for owners, managers, designers, engineers and contractors. 2nd ed ed. Hoboken, NJ : Wiley, 2011. ISBN 978-0-470-54137-1. p193

³¹ COENDERS, Jeroen L. - Next Generation Parametric Design. Journal of the International Association for Shell and Spatial Structures. . ISSN 1028-365X. 62:2 (2021) 153–166. doi: 10.20898/j.iass.2021.018. p154

Coenders também indica que o BIM trabalha com objectos e informações concretas (como paredes vigas e janelas) tendo, por isso, uma maior dificuldade em desenvolver modelos mais abstractos, especialmente nas fases iniciais do projecto quando diversas informações são ainda desconhecidas e o modelo tem uma maior flexibilidade de opções. "The object definition of Building Information Modelling however often requires detailed choices about what object to use and what its properties are. At the early stages this information is unknown (...) and therefore still to be determined and not available yet. The information which leads to these choices can often not be stored in Building Information Modelling systems and the early choices are lost as they often are abstract or do not possess the required level of detail (that of an object)."³²

Alguns exemplos de softwares BIM destacados:

Archicad:

O Archicad é um dos softwares pioneiros do BIM e actualmente é também um dos mais populares na indústria AEC. O Archicad encontra-se actualmente na versão 25 e tem um GUI bem organizado e relativamente intuitivo. Este software oferece diversas ferramentas para facilitar a criação de formas e geometrias, facilitando o processo criativo. Esta plataforma oferece também muito boas soluções para o trabalho colaborativo, permitindo que os utilizadores trabalhem no mesmo modelo BIM a partir de sistemas operativos diferentes ou através de um servidor online (client-server model) que contém o modelo BIM e que pode ser acedido por diversos dispositivos remotamente. O Archicad introduziu na versão 24 um VAE (o Param-O) para a criação de objectos paramétricos e também um API de Python, no entanto esta última opção encontra-se ainda numa fase experimental.

O Archicad funciona em dois sistemas operativos (MacOS ou Windows) e oferece 4 tipos de licenças que o utilizador pode escolher: uma licença ilimitada, uma licença por subscrição/aluguer, uma licença educativa (1 ano / renovável) e uma licença trial (30 dias).

³² COENDERS, Jeroen L. - Next Generation Parametric Design. Journal of the International Association for Shell and Spatial Structures. . ISSN 1028-365X. 62:2 (2021) 153–166. doi: 10.20898/j.iass.2021.018. p155

Revit:

Criado em 2000 e adquirido pela Autodesk em 2002, o Revit é actualmente um dos programas de BIM mais utilizados pela indústria AEC. Este software oferece boas ferramentas engenharia para desenvolver as especialidades técnicas do projecto e por esse motivo promove a articulação entre as diferentes disciplinas participantes. Para além do Revit, a Autodesk possui diversos programas de computador e, por esse mesmo motivo, o Revit articula muito facilmente com outros softwares deste ecossistema com os quais mantém relações próximas. O Dynamo é um Plug-in que permite a criação de modelos no Revit de forma paramétrica, para além disso, o Revit possui um API de Python que permite um controlo mais específico do software.

O Revit funciona apenas em Windows e a Autodesk permite a utilização deste software de 4 maneiras: uma licença por subscrição (211,8\$ a 335\$ por mês), uma licença por credito/tokens (10 tokens por cada utilização), uma licença educativa (1 ano / renovável) e uma licença trial.

3.4 - Algorithm-Aided Design (AAD)

Descrição do AAD:

A metodologia AAD "Algorithm-Aided Design" oferece a possibilidade de criar formas geométricas programando o computador com instruções (através de scripts) em vez de usar ferramentas para criar a geometria em si manualmente. Esta metodologia de trabalho é mais complexa do que as metodologias de trabalho digital mais convencionais (como por exemplo o CAD), no entanto a criação de software dedicado está a simplificar esta metodologia e a torná-la mais intuitiva, sendo cada vez mais utilizada por projectistas como por exemplo arquitectos e engenheiros. "On the other hand, the use of Algorithm-Aided Design (AAD) methods are also becoming more common and integrated into the workflows of design studios."³³

Os algoritmos são geralmente criados na forma de texto recorrendo a linguagens de programação (como por exemplo C ou Python), contudo, mais recentemente, começaram a surgir novas formas de criar algoritmos de forma visual/gráfica através do uso de VAE's (Visual Algorithm Editor), softwares que oferecem uma interface visual geralmente através do uso de blocos/módulos/nodes que se interligam visualmente (como por exemplo o GrassHopper no Rhinoceros3D, o Dynamo para o Revit, o GenerativeComponents ou o Geometry Nodes no Blender). "About 18 years ago a new generation of parametric and associative design systems emerged which has provided a (partial) solution for the further expression of design and engineering logic and justification. In order of appearance GenerativeComponents, Grasshopper and Dynamo have emerged as parametric systems which share a few crucial characteristics: an explicit, visual logic graph with user-defined parameters (...) a single-directional graph solver which solves the graph on change (...) Opposed to the earlier textual programming languages these systems use visual programming languages"³⁴

Neste método de trabalho (Algorithm-Aided Design), O arquitecto não cria o modelo geométrico directamente como numa forma de trabalho convencional. Em AAD, o arquitecto programa o computador com as instruções de como criar o modelo geométrico. Numa metáfora simples de perceber, na metodologia CAD ou BIM, o arquitecto cria/prodiz um bolo (o projecto), numa abordagem AAD, o arquitecto escreve a receita (o script) com as instruções para o computador criar o bolo (o projecto) e cada vez que o arquitecto reescreve as instruções, o computador altera o bolo de acordo com as novas instruções de forma automática e rápida. "In

³³ HUMPPPI, Harri; ÖSTERLUND, Toni - Algorithm-Aided BIM. 2015. p601

³⁴ COENDERS, Jeroen L. - Next Generation Parametric Design. Journal of the International Association for Shell and Spatial Structures. . ISSN 1028-365X. 62:2 (2021) 153–166. doi: 10.20898/j.iass.2021.018. p158

algorithmic processes the form generation is controlled by user-defined scripts composed in either textual or graphical syntax. Scripts allow designers to generate and control outcomes that would otherwise be too complex to handle by manual modeling."³⁵

Contribuição para a eficiência do projecto:

A metodologia AAD permite uma maior flexibilidade de opções, contudo essa maior flexibilidade de opções resulta numa maior complexidade e, por esse motivo, uma maior curva de aprendizagem. Dentro da metodologia AAD, o grau de dificuldade pode ainda variar entre a programação visual e a programação escrita, tal como demonstrado por Robert Aish e Emanuel Mendoza no seu trabalho através de um gráfico ilustrativo onde representam essa curva de aprendizagem. "The sequence of increasing programming complexity. Abstraction barriers and consequential discontinuity may be encountered with existing domain specific applications when transitioning from visual data flow programming to text based programming."³⁶

O desenvolvimento de interfaces gráficas de programação (VAE) tal como o Grasshopper e o Dynamo, tornou esta metodologia mais intuitiva e, por esse mesmo motivo mais acessível a projectistas sem grandes conhecimentos de programação. Jeroen L. Coenders sugere estes sistemas de programação visual como uma solução alternativa eficiente aos métodos de programação escrita. Segundo este autor, a programação visual é mais intuitiva para arquitectos e engenheiros utilizarem do que a programação escrita que requer conhecimentos mais aprofundados que não fazem parte do currículo dos projectistas. "this type of programming languages (Visual Programming Languages) is far easier for designers and engineers to adopt than their textual siblings"³⁷

As formas geométricas geradas por algoritmos beneficiam da capacidade de replicar e gerir múltiplos objectos, permitindo que o utilizador possa definir ou alterar as características de um tipo de objecto e todos os semelhantes ou dependentes deste serão automaticamente actualizados/adaptados às novas características. Para além disso, o desenho paramétrico permite trabalhar com diferentes níveis de abstracção geométrica, o que pode ser benéfica numa fase inicial do projecto.

³⁵ (Harri Humpi Toni Österlund - AAB p601) - HUMPPi, Harri; ÖSTERLUND, Toni - Algorithm-Aided BIM. 2015. p601

³⁶ AISH, Robert; MENDOZA, Emmanuel - DesignScript: a domain specific language for architectural computing. Em Proceedings of the International Workshop on Domain-Specific Modeling [Em linha]. Amsterdam Netherlands : ACM, 30 out. 2016 [Consult. 20 nov. 2021]. Disponível em WWW:<URL:https://dl.acm.org/doi/10.1145/3023147.3023150>. ISBN 978-1-4503-4894-2. p2

³⁷ COENDERS, Jeroen L. - Next Generation Parametric Design. Journal of the International Association for Shell and Spatial Structures. . ISSN 1028-365X. 62:2 (2021) 153–166. doi: 10.20898/j.iass.2021.018. p158

Esta metodologia oferece também novas ferramentas de trabalho tais como "solvers" (por exemplo o Galapagos no Grasshopper) que se baseiam na grande capacidade de cálculo do hardware para testar diversas hipóteses geométricas para encontrar a solução mais eficiente através de uma metodologia do tipo "Brute-Force". AAD permite também que os arquitectos e engenheiros expressem os cálculos e a lógica geométrica/compositiva do projecto mais facilmente do que em metodologias como o BIM ou CAD. "use visual programming languages in order for the designer or engineer to express their design and engineering logic"³⁸ Por este motivo, a metodologia AAD pode facilitar os processos utilizados pelos licenciamento para se certificar se o edifício cumpre os regulamentos.

A metodologia AAD permite uma relação/interação mais próxima entre diversas áreas que constituem o projecto tais como arquitectura, estrutura, térmica ou acústica. Os VAE's têm uma grande capacidade de análise e simulação que pode ser expandida através de extensões/plugins/addons independentes que adicionam novas funções e ferramentas como o ArchicadLiveConnection ou o LadyBug para o GrassHopper por exemplo.

A metodologia AAD facilita, por isso, a integração de análises estruturais, acústicas, ambientais, entre outras, através da conexão direta com softwares (por exemplo de tipologia FEA), ou através da integração dos métodos de análise (FEA por exemplo) directamente no software AAD através de uma extensão/plugin/addon.

A programação visual pode ser criada de forma mais rápida e fluída ao contrário da programação escrita que requer geralmente mais tempo para consulta e testes. Isto deve-se ao facto da programação visual trabalhar com blocos visuais (o que reduz a probabilidade do utilizador criar de erros sintaxe) ao contrário da programação escrita onde um único caractere (como um parêntese ou aspa) a mais ou em falta resulta num erro que impede a execução correcta do código.

O recurso a algoritmos possibilita também a capacidade de criar formas geométricas complexas (difícilmente reproduzidas manualmente) de forma eficiente e rápida.

Jeroen L. Coenders salienta também que, ao contrário do BIM e do CAD, o AAD tem a capacidade de descrever o pensamento lógico na concepção da forma geométrica.³⁹

³⁸ COENDERS, Jeroen L. - Next Generation Parametric Design. Journal of the International Association for Shell and Spatial Structures. . ISSN 1028-365X. 62:2 (2021) 153–166. doi: 10.20898/j.iass.2021.018. p158

³⁹ COENDERS, Jeroen L. - Next Generation Parametric Design. Journal of the International Association for Shell and Spatial Structures. . ISSN 1028-365X. 62:2 (2021) 153–166. doi: 10.20898/j.iass.2021.018. p158

Alguns exemplos de softwares AAD destacados:

GrassHopper:

O GrassHopper é um plugin criado em 2007 que adiciona capacidades paramétricas ao Rhinoceros 3D e permite que os seus utilizadores criem algoritmos/scripts através de blocos visuais num gráfico interactivo e intuitivo de compreender. O GrassHopper passou a estar integrado no Rhinoceros3D desde a versão 6 mas pode ser instalado como um plugin em versões anteriores deste programa para o sistema operativo Windows.

Este plugin possui uma grande quantidade de nodes que oferecem diversas possibilidades para o utilizador concretizar os seus objectivos. Para além dos nodes existentes, o GrassHopper oferece a opção de articular programação escrita com programação visual no algoritmo através de nodes de Python, C# e VisualBasic, que permitem que o utilizador crie as suas próprias ferramentas para modelar formas geométricas complexas. Este plugin funciona exclusivamente no Rhinoceros 3D e como tal, para ser utilizado, o utilizador necessita de adquirir este software.

Blender Nodes:

O Blender é um programa Open-Source bastante desenvolvido com capacidades para modelar objectos 3D, animar, renderizar, texturizar entre outras características. Este programa foi desenvolvido em 1994 por Ton Roosendaal que em 2002 fundou a Blender Foundation com o objectivo de tornar este software Open-Source. Actualmente, o Blender possui nativamente dois node editors/VAE's, o Shader Nodes Editor e mais tarde o Geometry Nodes Editor.

O Shader Nodes Editor permite que o utilizador crie parametricamente materiais, shaders e texturas para aplicar nos objectos geométricos. O Geometry Nodes Editor foi introduzido mais tarde na versão 2.92 (Fevereiro de 2021) e permite a criação de formas geométricas complexas através de algoritmos visuais que o utilizador pode criar. Tendo em conta que o Blender tem capacidades de animação, os nodes podem também ser animados parametricamente.

Este programa está disponível gratuitamente em 3 sistemas operativos, Linux, MacOS e Windows. Por esse motivo, o Blender tem uma grande comunidade de utilizadores que contribuem para o desenvolvimento deste programa com conhecimento e addons, partilhando o seu trabalho em prol da evolução deste software.

3.5 - Algorithm-Aided Building Modeling Information (AABIM)

Descrição do AABIM:

O nome Algorithm-Aided BIM é proposto por Toni Österlund e Harri Humpi e refere-se a uma metodologia de trabalho digital que combina os aspectos de BIM com AAD com o intuito de aproveitar as vantagens de cada uma destas metodologias. Humpi e Österlund não são os primeiros a identificar esta nova abordagem, mas desenvolveram um trabalho escrito que clarifica muito bem esta metodologia. No seu trabalho, Humpi e Österlund identificam as diferentes metodologias de trabalho, na sua origem e nas vantagens ou desvantagens que cada uma tem no processo de projectar. Eles fazem também referência às 3 eras do CAD sugeridas por Robert Aish em 2013. "Historical eras of Computer-Aided Design (CAD) are shortly discussed through Figure 2. According to Aish (2013) the history of practical CAD can be split into three eras: the 2D Drafting era, the Building Information Modeling era and the Design Computation era."⁴⁰

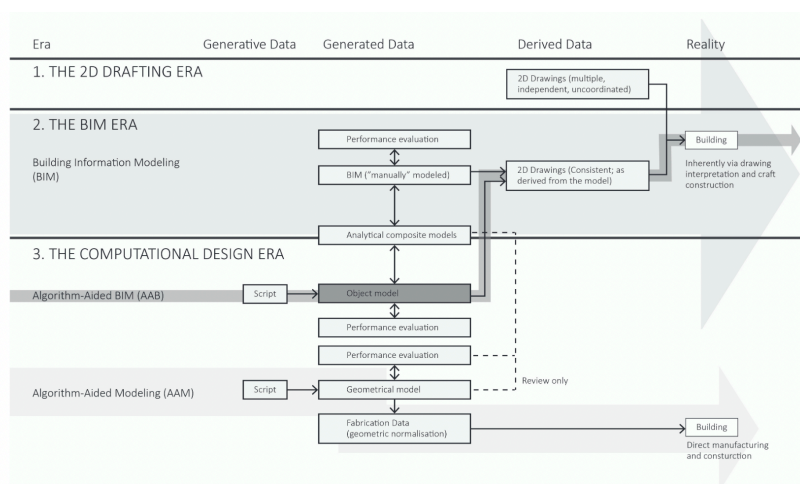


Fig 9 - As 3 Eras dos processos de trabalho digital
Referência:
Humpi & Österlund

Posteriormente, os autores identificam a tipologia AAB (Algorithm-Aided Building Information Modeling) como o tema de estudo do seu trabalho explicando as vantagens provenientes desta e de que forma os softwares contribuem para uma nova metodologia de trabalho para os arquitectos através de uma maneira de projectar edifícios flexível e eficiente. "These two design approaches have been developing rather separately until recent years. Currently, new tools have been developed that combine aspects of them both. They enable designers to generate and control object models through scripting and exchange them with BIM tools (e.g.

⁴⁰ HUMPII, Harri; ÖSTERLUND, Toni - Algorithm-Aided BIM. 2015. p601

Mirtschin 2011). These new tools affirm that advantages of BIM and AAD can be used in the same design and modeling processes."⁴¹

Os autores indicaram também que a relação entre estas duas metodologias (BIM e AAD) é pouco estudada estando a maior parte da investigação focada na criação de novos softwares e ferramentas digitais. No entanto, cada vez mais entidades têm demonstrado interesse na potencialidade desta metodologia pelas suas vantagens.

A metodologia AABIM está a conquistar uma presença cada vez mais frequente na área da arquitectura com um crescente interesse por parte dos utilizadores de softwares de tipologia BIM como o Archicad ou o Revit que estão a integrar as capacidades paramétricas do AAD através de plugins como por exemplo o Dynamo, o Rhino.Inside.Revit, o GeometryGym, o GrassHopper-ArchicadLiveConnection ou o Param-O.

Para além disto Österlund e Humpi referem a dificuldade e a complexidade do projecto de arquitectura e indicam que um só software não consegue dar resposta a todas as necessidades do mesmo. É neste contexto que afirmam/indicam a importância da interoperabilidade com outros softwares especializados/dedicados à realização de diferentes tarefas. Para tal, a comunicação de dados/informação entre os diferentes softwares tem que ser compatível, universal, fluente e constante para se conseguir atingir processos automáticos em tempo real.

Os autores indicam que o projecto consiste numa série de processos que o projectista tem que realizar a fim de descobrir uma solução e indicam que estes processos não são lineares sendo necessário recorrer à repetição para encontrar uma solução final satisfatória, resultando num processo laborioso e demorado. "All in all, the design process is a series of methodological steps that the designer takes in order to find a sufficient design solution. In practice, the design process is nonlinear, so repetition is needed to find the final solution."⁴²

Contribuição para a eficiência do projecto:

A metodologia AABIM por ter uma mesma mecânica de execução idêntica à metodologia AAD, oferece uma vantagem de trabalho neste processo repetitivo em relação às outras metodologias. Enquanto que numa metodologia convencional de trabalho todas as alterações têm que ser realizadas manualmente, nas metodologias paramétricas (AAD e AABIM) o projectista apenas tem que mudar as instruções do script (a receita) e o computador executará as alterações a todos os objectos de maneira automática (os ingredientes) resultando num novo modelo (o bolo). Pode-se

⁴¹ HUMPPi, Harri; ÖSTERLUND, Toni - Algorithm-Aided BIM. 2015. p601

⁴² HUMPPi, Harri; ÖSTERLUND, Toni - Algorithm-Aided BIM. 2015. p602

perceber melhor esta vantagem através do seguinte exemplo. Se o projectista tiver que mudar a forma da planta do edifício num software de tipologia CAD ou BIM, ele vai ter que reajustar os elementos manualmente (Paredes, pilares, vigas, entre outros). Se o projectista tiver que mudar a forma da planta do edifício num modelo paramétrico, ao mover a linha-eixo de uma parede, toda a estrutura adaptar-se-á automaticamente à nova forma da planta desde que devidamente parametrizado.

No desenvolvimento da sua observação, HUMPPPI e ÖSTERLUND representam a metodologia AABIM através de um gráfico onde esta serve como um elo de ligação entre o desenho orientado por objectos (BIM) que serve os standards de construção industrial e o desenho paramétrico (AAD) que oferece uma grande capacidade de customização.

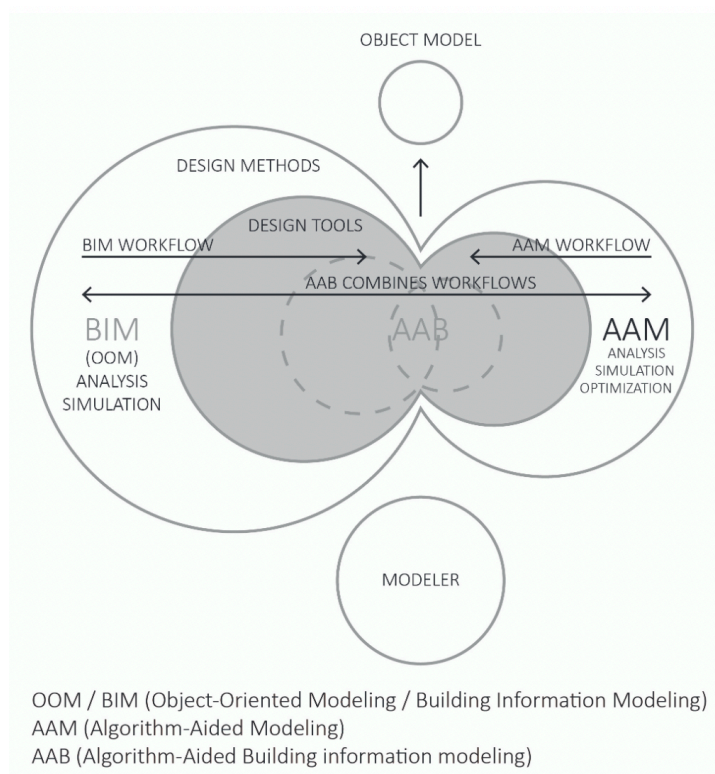


Fig 10 - Relação entre AAD/AAM e BIM
 proposta por Humppi e Österlund
 Referência: Humppi & Österlund

Ao contrário do CAD e do AAD, o AABIM facilita também a a Interoperabilidade entre os diversos softwares necessários à execução do projecto (como por exemplo análise/simulação térmica ou estrutural) porque consegue partilhar parâmetros com informações dos objectos (característica herdada do BIM). Em oposição, a metodologia CAD tem uma enorme dificuldade de interoperabilidade pois os documentos precisam de ser interpretados por um ser humano.

Apesar da utilização de interfaces gráficos/visuais para o projectista ter uma melhor percepção (mais intuitiva) dos algoritmos, esta metodologia (AABIM) requer

uma interpretação programática e lógica do modelo, aproximando o utilizador a alguns conceitos de programação como arrays ou parâmetros.

Diversos softwares da tipologia AAD foram criados para funcionar numa plataforma específica, por exemplo, o GrassHopper foi desenvolvido para o Rhinoceros3D e o Dynamo foi desenvolvido para os programas da Autodesk como por exemplo o Revit. Por esse mesmo motivo, a integração de softwares de desenho paramétrico (AAD) é um pouco exclusiva a alguns programas e não existem soluções universais ou convenções que ajudem a integrar o AAD com todos os programas (de tipologia BIM ou até mesmo CAD) porque cada programa tem o seu próprio sistema independente.

Para ultrapassar esta dificuldade e para que alguns VAE's (como o GrassHopper) possam ser utilizados com outros programas é necessário o uso de plugins. Por exemplo, para que o Grasshopper possa funcionar/articular com o sistema do Archicad é preciso instalar um plug-in, o GrassHopper-Archicad Live Connection.

O AABIM tem a potencialidade de melhorar a eficiência do trabalho dos arquitectos e melhorar a qualidade do projecto, dando respostas a muitos problemas que outras metodologias não conseguem resolver tão eficientemente (como por exemplo executar alterações ao projecto ou otimizar a forma geométrica com solvers para obter a melhor eficiência térmica).

Alguns exemplos de softwares AABIM destacados:

Dynamo:

O Dynamo é um VAE que permite a criação de modelos BIM de forma paramétrica através de algoritmos visuais. Este software articula em proximidade com o Autodesk Revit e existem duas versões do Dynamo, o Dynamo Studio e o Dynamo Revit. O Dynamo Revit é um plugin que articula directamente com o API do Revit e, como tal, permite que o utilizador utilize os elementos deste software de forma interactiva. Este plugin está incluído com o Revit. O Dynamo Studio funciona como um programa independente e tem a capacidade de operar sem a necessidade de utilizar os elementos do Revit. O Dynamo Studio está disponível através de uma subscrição Autodesk. Em ambos os casos, o Dynamo só funciona no sistema operativo Windows.

Param-O:

O Param-O foi introduzido na versão 24 do Archicad com o objectivo de adicionar capacidades paramétricas neste software BIM. No entanto, o Param-O permite criar objectos paramétricos individuais no modelo e ainda não tem a capacidade de parametrizar/controlar todos os objectos constituintes do edifício. O Param-O vem embutido directamente no Archicad e, por esse motivo, pode ser utilizado nos sistemas operativos MacOS e Windows.

ArchicadLiveConection e RhinoInsideRevit:

O ArchicadLiveConnection e o RhinoInsideRevit são dois plugins diferentes que permitem interligar o GrassHopper (AAD) com o Archicad (BIM) e o Revit (BIM) respectivamente.

O ArchicadLiveConnection é desenvolvido pela Graphisoft e permite estabelecer uma ponte entre o GrassHopper e o Archicad. Esta ferramenta consiste em dois plugins, um plugin para o Grasshopper com as ferramentas/nodes BIM e um plugin para o Archicad com uma ferramenta para sincronizar e actualizar o modelo paramétrico em tempo real.

O Rhino.Inside.Revit é um plugin desenvolvido pela Robert McNeel & Associates que permite embutir o Rhinoceros 7 e o Grasshopper no Revit. Este plugin é instalado no Revit a partir do qual, ao ser activado vai tentar abrir o Rhinoceros e o GrassHopper dentro do Revit em si.

3.6 - Programação Escrita

Neste módulo 3.6, a programação é estudada/interpretada de um ponto de vista funcional de automatização, ao contrário do módulo 3.4 Algorithm-Aided Design, onde a programação está directamente associada à criação do projecto e é estudada como uma metodologia de criação de geometria.

Breve descrição à programação de computadores:

Para melhor se perceber a importância/utilização das linguagens de programação, é preciso compreender a relação que estas estabelecem com os computadores. Um computador é um dispositivo multi-funções "general-purpose device"⁴³ que executa as instruções que lhe são fornecidas para processar dados.

Os computadores executam as instruções uma por uma, no entanto têm a capacidade de processar grandes quantidades de instruções por segundo. "Modern computers can execute many millions of these instructions in a second"⁴⁴

Um ser Humano consegue executar/interpretar instruções abstratas e complexas, através de uma linguagem corrente como por exemplo inglês. Ao contrário do ser Humano, um computador apenas consegue executar instruções lógicas e concretas (como por exemplo somar dois números) através de linguagens técnicas que permitem a comunicação entre o ser humano e a máquina (como por exemplo o Python). Por este motivo, instruções mais complexas terão que ser compostas por vários passos lógicos que o computador possa executar. Estes conjuntos de algoritmos ou de instruções cujo o objectivo é executar tarefas específicas de forma reutilizável pelo computador são conhecidos como programas.

Algoritmos:

Um algoritmo é uma série de instruções que ao serem executadas permitem realizar uma tarefa. A receita de um bolo pode ser interpretada como um algoritmo porque contém instruções de como processar os ingredientes (input) de forma a obter um bolo (output). Numa aproximação estruturada da programação "structured approach to programming"⁴⁵, o CPU do computador é considerado o "actor" das instruções através de uma orientação unidirecional que, segundo os autores deste

⁴³ Python textbook - readthedocs-io-en-1.0 - [Em linha]. Cape Town : University of Capetown, 2017
Disponível em WWW:<URL:https://python-textbok.readthedocs.io/_/downloads/en/1.0/pdf/>. p3

⁴⁴ Python textbook - readthedocs-io-en-1.0 - [Em linha]. Cape Town : University of Capetown, 2017
Disponível em WWW:<URL:https://python-textbok.readthedocs.io/_/downloads/en/1.0/pdf/>. p4

⁴⁵ Python textbook - readthedocs-io-en-1.0 - [Em linha]. Cape Town : University of Capetown, 2017
Disponível em WWW:<URL:https://python-textbok.readthedocs.io/_/downloads/en/1.0/pdf/>. p8

livro, é uma solução eficaz em aplicações simples, contudo, num programa mais complexo, é difícil um só actor ter o controlo de tudo.

É neste contexto que é introduzido o conceito de programação orientada a objectos. Neste conceito, em vez de um actor, são considerados múltiplos actores no algoritmo, objectos/entidades independentes umas das outras com características únicas que actuam no algoritmo (instances of classes por exemplo). Na programação orientada a objectos (object-oriented programming), os diversos objectos têm instruções específicas, no entanto eles articulam uns com os outros para executar a mesma tarefa. Neste caso, o algoritmo "like a play script"⁴⁶ gere os diversos objectos de modo a executar as suas instruções pela ordem correcta.

Os programas de computador são a maneira convencional de converter um algoritmo em instruções que a máquina possa interpretar/compreender. É neste contexto que é utilizada uma linguagem de programação que melhor traduz o pensamento do ser Humano em instruções para o computador poder executar através de "machine language" (zeros e uns).

Actualmente, existem diversas linguagens de programação que foram criadas maioritariamente ao longo do século XX mas também no século XXI. Estas linguagens de programação podem ser categorizadas em diversos níveis de abstracção (machine language, assembly language e high-level language).

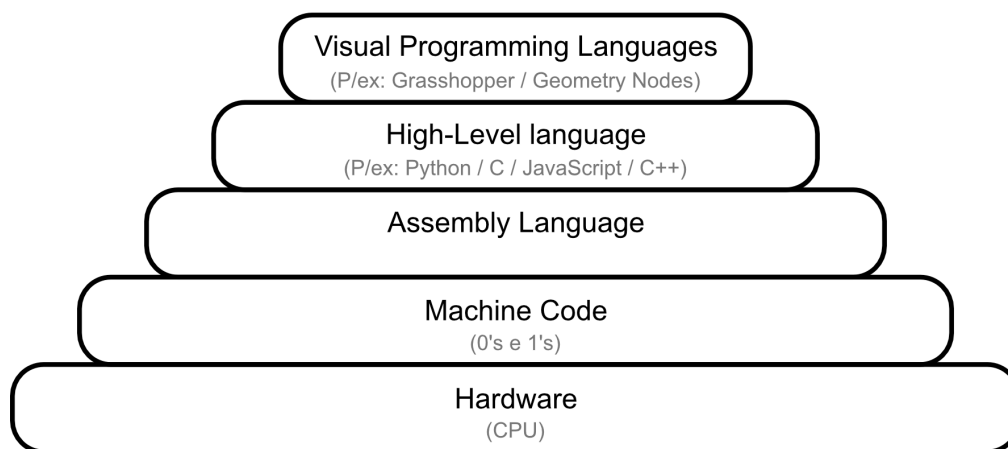


Fig 11 - Níveis de programação

⁴⁶ Python textbook - readthedocs-io-en-1.0 - [Em linha]. Cape Town : University of Capetown, 2017
Disponível em WWW:<URL:https://python-textbok.readthedocs.io/_/downloads/en/1.0/pdf/>. p8

Machine Language (low-level language) é a forma mais directa de introduzir instruções num computador e trata-se da utilização de código binário (0's e 1's) directamente no hardware (CPU por exemplo). Trata-se de uma linguagem concreta que não oferece conceitos de abstracção intuitivos à percepção humana e, por este motivo não é frequentemente utilizada nos dias de hoje, mas na década de 1940 era a única opção que os programadores tinham.

The reason that we do not write computer instructions directly is that they are difficult for humans to read and understand. For example, these are the computer instructions (in the Intel 8086 machine language, a subset of the Intel Pentium machine language) required to add 17 and 20:

```
1011 0000 0001 0001
0000 0100 0001 0100
1010 0010 0100 1000 0000 0000
```

The first line tells the computer to copy 17 into the AL register: the first four characters (1011) tell the computer to copy information into a register, the next four characters (0000) tell the computer to use register named AL, and the last eight digits (0001 0001, which is 17 in binary) specify the number to be copied.

Fig 12 - Exemplo de machine language para somar 17 e 20 - Referência: Python Textbook, 2017

Para programar os computadores mais eficientemente, foi criado assembly language que, apesar de ser mais intuitiva de compreender, ainda é considerada low-level language. Cada linha de assembly é equivalente a uma linha de machine code como se pode ver no exemplo seguinte. Convém realçar que os computadores (hardware) não interpretam directamente a linguagem assembly e para tal, esta precisa de ser traduzida para machine code/language através de um assembler.

Each assembly instruction corresponds to one machine language instruction, but it is more easily understood by humans, as can be seen in the equivalent addition program in the 8086 assembly language:

```
MOV AL, 17D
ADD AL, 20D
MOV [SUM], AL
```

Fig 13 - Exemplo de assembly language para somar 17 e 20 - Referência: Python Textbook, 2017

Esta linguagem foi utilizada na década de 1960 pela NASA para escrever o programa de computador que realizava os cálculos da trajectória do Apollo. A fotografia seguinte mostra o programa do Apollo escrito em Assembly ao lado de Margaret Hamilton, a directora de programação do computador de voo do Apollo. Nesta fotografia é possível perceber que a natureza pouco compacta do código Assembly resultava em programas extensos, em contraste com as linguagens de programação actuais. Assembly language tornou a programação de



Fig 14 - Margaret Hamilton ao lado do programa Apollo
Ref: <https://www.nasa.gov>

computadores mais eficiente a nível de percepção e de escrita, no entanto esta linguagem ainda necessita de muitas linhas de instruções para realizar uma tarefa simples. É neste contexto que são introduzidas as "High-Level Programming Languages". As linguagens de programação High-Level melhoraram muito a eficiência de programar e facilitaram ainda mais esta tarefa porque uma instrução de High-Level é feita de diversas linhas de machine language. Para além desta optimização, as High-Level Language introduzem conceitos de abstracção intuitivos (como por exemplo classes, listas, dicionários e variáveis) que facilitam a percepção Humana do código resultando numa sintaxe mais fácil de ler e compreender.

In a high-level language, an instruction may correspond to several machine language instructions, which makes programs easier to read and write. This is the Python equivalent of the code above:

```
sum = 17 + 20
```

Fig 15 - Exemplo de Python para somar 17 e 20 - Referência: Python Textbook, 2017

Alguns exemplos de Linguagens de Programação High-Level:

Tal como o assembly language, high-Level Languages não são directamente interpretadas pelos computadores e, para tal, precisam de ser convertidas em machine language através de interpretadores ou compiladores. Algumas linguagens de programação (como por exemplo Python, Ruby ou Perl) utilizam interpretadores para serem executadas. Os interpretadores convertem e executam o código em machine language linha por linha. Isto permite que um mesmo código Python possa ser interpretado de forma semelhante em vários sistemas diferentes porque se utiliza o código python como base. Convém referir que este processo pode ser um pouco menos eficiente do que uma linguagem de programação compilada porque ao interpretar, o computador tem que estar a converter o código em machine language ao mesmo tempo que o executa.

Outras linguagens de programação (como por exemplo Swift, Java, C, C++ ou C#) usam compiladores que criam um novo ficheiro com todo o código convertido em machine language que é executado pelo computador. As linguagens de programação compiladas tendem a ser mais rápidas de executar do que as correspondentes interpretadas por este mesmo motivo. No entanto, cada compilador converte o código High-Level para o sistema onde este está a ser utilizado, por esse motivo, o mesmo código compilado não pode ser executado correctamente em dois sistemas operativos diferentes.

Fig 16 - Genealogia de algumas das Linguagens de Programação High-Level
Ref: Farooq et al.

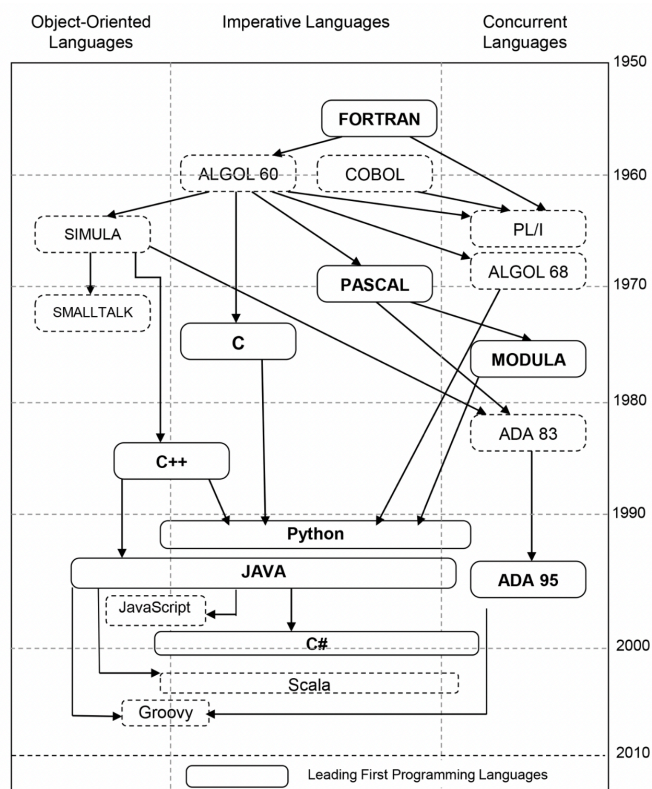


Figure 1. Genealogy of Programming Languages.
doi:10.1371/journal.pone.0088941.g001

C é uma linguagem de programação compilada que oferece grande controlo e otimização dos recursos do computador, por este motivo, costuma ser utilizadas em aplicações/programas mais intensivos como por exemplo jogos/motores de renderização. No entanto ao programar em C é necessário gerir manualmente a memória do computador, aumentando a dificuldade e o tempo utilizado para se programar.

Por outro lado, Python é uma linguagem de programação interpretada que facilita a gestão da memória resultando num processo de trabalho eficiente. "For applications which are not resource-intensive, a language like Python allows programs to be developed more easily and rapidly, and the speed difference on a modern computer is usually negligible."⁴⁷

⁴⁷ Python textbook - readthedocs-io-en-1.0 - [Em linha]. Cape Town : University of Capetown, 2017
Disponível em WWW:<URL:https://python-textbok.readthedocs.io/_/downloads/en/1.0/pdf/>. p9

Programação Processual (Procedural Programming Languages):

As linguagens de programação processuais (também conhecidas como linguagens de programação imperativas) funcionam como uma lista de instruções que são seguidas pela ordem de leitura do script (de cima para baixo). Neste método as variáveis servem como forma de comunicação entre os diversos métodos e objectos do programa. Nas linguagens de programação processuais, as variáveis podem adquirir valores (como por exemplos números ou texto), estes valores podem mais tarde ser alterados durante o processo da execução do programa. O valor das variáveis é armazenado na memória principal do computador.

Alguns exemplos destas linguagens de programação são: o BASIC, o COBOL, o FORTRAN e o C.

Programação Funcional (Functional and Logic Programming Languages):

As linguagens de programação funcionais (também conhecidas como linguagens de programação lógicas) assemelham-se muito à forma das expressões matemáticas. Ao contrário das linguagens de programação processuais, nas linguagens funcionais, o valor das variáveis não pode ser alterado durante a execução do programa. Nestas linguagens de programação funcional também não costumam ser utilizadas variáveis porque é frequente utilizar os parâmetros das funções para comunicar a informação entre as diversas partes do programa.

Alguns exemplos destas linguagens de programação são: o Common Lisp, o ML, a MIRANDA e o Prolog. "A logic language is based on formal rules of logic and inference. An example of such a language is Prolog. Prolog's variables cannot be changed once they are set, which is typical of a logic language."⁴⁸

Programação Orientada a objectos (Object-oriented Programming Languages):

Tal como o próprio nome indica, nas linguagens de programação orientadas a objectos, são utilizados "objectos" (como por exemplo "classes" ou "structures") que ajudam a representar melhor entidades com características e comportamentos (Behaviors) específicos.

Alguns exemplos destas linguagens de programação são: o Python, o Swift, o C++, o Java e o Smalltalk. "Simula (Simulation Language), invented in 1967, was the first language to take the object-oriented approach. However, Smalltalk (early 1980s) was the first purely object-oriented language – everything in the language is an object. C++ is a hybrid OO language in that it has the procedural aspects of C. A C++ program can be completely procedural, completely OO or a hybrid. Most OO languages make use of variables in a similar fashion to procedural languages. Java

⁴⁸ Python textbook - readthedocs-io-en-1.0 - [Em linha]. Cape Town : University of Capetown, 2017
Disponível em WWW:<URL:https://python-textbok.readthedocs.io/_/downloads/en/1.0/pdf/>. p8

was introduced in 1995 by Sun Microsystems, who were purchased by Oracle Corporation during 2009-2010. It is an OO language but not as pure as Smalltalk. For example, in Java primitive values (numbers and characters) are not objects – they are values. In Smalltalk, everything is an object. Initially it was designed for use in appliances, but the first released version was designed for use on the Internet. Java syntax is quite similar to C and C++, but functions cannot be defined outside of objects."⁴⁹

Contribuição para a eficiência do projecto:

Alguns autores (como por exemplo Jeroen L. Coenders) indicam o potencial de associar a programação escrita ao projecto e referem uma possível relação entre as áreas da AEC e a ciência de computadores. No seu trabalho de 2021, Coenders descreve brevemente a função de diversas tipologias de software digitais (CAD, BIM e FEA) no desenvolvimento do projecto, onde ele identifica algumas dificuldades/limites destas metodologias actuais. Para responder a estas dificuldades, este autor sugere a utilização de programação (escrita e visual). "Another means which some designers and engineers (the author being amongst them) have employed from a very early stage, is to use computer programming languages to encode design and engineering logic in homegrown computer applications which either generate the object or perform automation tasks similar to what designers and engineers use spreadsheets for."⁵⁰

Coenders indica/identifica 5 vantagens na utilização de programação escrita.

Em primeiro lugar, a programação escrita tem uma capacidade superior às tabelas porque permite outras formas de expressar a lógica do pensamento e gerir a informação/dados. Para além disso a programação escrita tem uma grande facilidade em expandir as suas capacidades e funções através da utilização de modules frameworks e libraries. "Computer programming languages often have far richer means to express logic than spreadsheets as well as provide toolkits in the form of frameworks and libraries with many advanced functionalities."⁵¹

Em segundo lugar, a programação escrita oferece formas mais avançadas de gerir os dados (tais como lists, data trees, dictionaires, hash tables entre outros) e

⁴⁹ Python textbook - readthedocs-io-en-1.0 - [Em linha]. Cape Town : University of Capetown, 2017
Disponível em WWW:<URL:https://python-textbok.readthedocs.io/_/downloads/en/1.0/pdf/>. p12, p13

⁵⁰ COENDERS, Jeroen L. - Next Generation Parametric Design. Journal of the International Association for Shell and Spatial Structures. . ISSN 1028-365X. 62:2 (2021) 153–166. doi: 10.20898/j.iass.2021.018. p156

⁵¹ COENDERS, Jeroen L. - Next Generation Parametric Design. Journal of the International Association for Shell and Spatial Structures. . ISSN 1028-365X. 62:2 (2021) 153–166. doi: 10.20898/j.iass.2021.018. p156/157

permite gerir e organizar a informação em sistemas mais complexos/avançados do que tabelas.

Em terceiro lugar, a programação escrita consegue gerir maiores quantidades de informação com maior rapidez do que por exemplo folhas de tabelas "spreadsheets".

Em quarto lugar as aplicações/programas de trabalho podem conectar-se uns com os outros através de API's. As API's permitem que as diversas aplicações/programas de computador possam comunicar umas com as outras de forma automática.

Em quinto lugar, a programação oferece opções ilimitadas naquilo que o programador quer fazer com ela, pois todas as aplicações das tipologias CAD e BIM foram desenvolvidas/criadas com programação de computador.

No entanto, Jeroen L. Coenders identifica também algumas dificuldades em implementar conhecimentos de programação no projecto. "While computer programming seems the most powerful approach and the approach that delivers the most freedom to the designer or engineer (...) Unfortunately, this approach also has some severe drawbacks as the author has found out over his years"⁵²

Em primeiro lugar, ele afirma que designers e engenheiros não são formados com conhecimentos de programação porque a ciência de computadores é uma área de conhecimento bastante ampla/grande e a construção de edifícios é também uma outra área de conhecimento bastante ampla, sendo por este motivo difícil integrar o conhecimento de dois currículos num só currículo. Jeroen afirma que é aparentemente impossível para um indivíduo ser eficiente nas duas áreas ao mesmo tempo, mas admite poder haver raras exceções.

Em segundo lugar, o autor propõe a possibilidade de projectistas e programadores trabalharem colaborativamente na forma de equipas com conhecimentos multidisciplinares. Ele afirma que esta abordagem é possível e já é implementada por algumas companhias, mas questiona a possibilidade desta abordagem se expandir para o resto da indústria AEC porque os programadores são raros e caros.

Em terceiro lugar, a programação é uma disciplina difícil de compreender e isso resulta num fator de "exclusão" e desinteresse para muitas pessoas.

Em quarto lugar, a capacidade de automatizar/interligar diferentes programas digitais ou até mesmo computadores é muito mais complexa do que a capacidade de automatizar uma simples tarefa. Segundo o autor, conectar/interligar múltiplos

⁵² COENDERS, Jeroen L. - Next Generation Parametric Design. Journal of the International Association for Shell and Spatial Structures. . ISSN 1028-365X. 62:2 (2021) 153–166. doi: 10.20898/j.iass.2021.018. p156

utilizadores (em diferentes programas e computadores) é benéfico para o trabalho multidisciplinar, mas a falta de conhecimento suficiente para resolver esta tarefa complexa de interoperabilidade pode causar instabilidade no software.

Em quinto lugar, o processo de programar é relativamente demorado e a automatização de uma simples rotina para resolver uma designada tarefa geralmente é específica a essa situação e não é reutilizável. Para além disso, a criação de aplicações mais complexas que possam ser utilizadas por um maior número de utilizadores e ter uma maior flexibilidade de utilizações, necessitam de um controlo/gestão maior (especialmente quando se trata de trabalho de equipa) e o software produzido pode não estar pronto a tempo de ser utilizado no projecto.

Módulo 4

Aplicações práticas das metodologias de trabalho digital - Casos de Estudo

4.0 - Descrição do Módulo 4

O módulo 4 tem como objectivo exemplificar cada uma das metodologias introduzidas no módulo anterior. Para tal, este módulo vai apresentar/descrever 5 casos de estudo com o objectivo de identificar as vantagens e desvantagens provenientes da utilização destas metodologias de trabalho no projecto. Este módulo vai ser dividido em seis blocos, cada uma correspondente a um caso de estudo.

O primeiro caso de estudo (4.1) vai analisar o uso do CAD na execução do projecto de arquitectura. Para tal, vai ser estudado um projecto realizado com software CAD para um edifício localizado na zona de Bankside em Londres.

O segundo caso de estudo (4.2) analisa o uso do BIM através do estudo do desenvolvimento de um edifício desenvolvido para o centro a cidade de Helsinki na Finlândia.

O terceiro caso de estudo (4.3) analisa a utilização de softwares AAD para desenvolver geometrias complexas e numerosas no projecto de arquitectura. Para exemplificar esta metodologia, vai ser analisado o trabalho realizado por Herzog & De Meuron descrito por Kai Strehlke.

O quarto caso de estudo (4.4) trata-se de um estudo desenvolvido por Iasbik Martinez e Gazel que demonstra uma aplicação do AABIM para desenvolver projectos de forma algorítmica através da criação de habitação modular.

O quinto caso de estudo (4.5) demonstra como a programação escrita pode ser implementada para desenvolver metodologias de trabalho automatizada. Para tal, vai ser analisado o trabalho desenvolvido por Maria Gkioka onde ela propõe a criação de um viaduto de forma paramétrica articulando diversos softwares com Python.

4.1 - Caso de estudo CAD - Bankside Loft

Para melhor demonstrar a função das tecnologias CAD e de que maneira estas possibilitam novas capacidades no trabalho dos projectistas, Kihong Ku desenvolveu um trabalho⁵³ no qual estudou a integração de diversas ferramentas digitais desta tipologia na concepção do projecto da extensão da Bankside Loft rooftop em Londres, desenvolvido pelo professor do MIT, Mark Goulthorpe. Segundo Ku, o objectivo do seu estudo é demonstrar como estas ferramentas digitais aumentam as possibilidades geométricas do projectista e também a percepção tridimensional do projecto.

Este projecto foi desenvolvido desde 2002 por Mark Goulthorpe, fundador da dECOi architects e trata-se da extensão de um apartamento em Bankside, uma zona localizada na margem sul do rio Thames na cidade de Londres. Este apartamento foi construído em 1999 e tem a forma de uma escada espiral para não prejudicar a qualidade de luz dos edifícios vizinhos. A extensão tem como objectivo criar um estrutura em vidro que encerre a cobertura e algumas varandas, de forma a promover uma vista panorâmica sobre a paisagem urbana.

Com este projecto, Goulthorpe analisou também as características geométricas da triangularização de uma forma volumétrica orgânica. Para tal, este projectista recorre à utilização de programas CAD para desenvolver a forma geométrica da estrutura e posteriormente software AAD para executar a parte construtiva e facilitar o processo de manufactura. Este projecto contou com o apoio/colaboração de especialistas em programas como o CATIA da Dessault Systèmes e o GenerativeComponents da Bentley.

Na criação deste projecto era necessário ter em conta quatro factores principais condicionantes à forma, a visão panorâmica de 360°, a eficiência térmica, restrições no volume e uma estrutura leve. Tendo em conta os requisitos, o arquitecto projectou uma estrutura em forma de bolha que acompanhava a forma espiral do edifício. Para executar um modelo tridimensional desta proposta, os arquitectos utilizaram o 3D Studio MAX, um programa de tipologia CAD que permite modelação e animação 3D. Definida a forma da proposta, os arquitectos desenvolveram 3 soluções construtivas. A primeira solução propunha utilizar exclusivamente painéis curvos para fazer a forma exterior arredondada. A segunda opção propunha utilizar painéis de revestimento curvos, mas com painéis de vidro planos. A terceira opção, propunha a

⁵³ KU, Kihong - CAD/CAM-MEDIATED DESIGN AND CONSTRUCTION: A CASE STUDY OF THE BANKSIDE LOFT PROJECT. Second International Conference World of Construction Project Management 2007. 2007)

utilização exclusiva de painéis planos através de uma forma triangulada com painéis de vidro quadriláteros. A terceira opção foi escolhida porque a estrutura triangulada, executada com painéis planos parecia ser a solução mais económica.⁵⁴ O modelo realizado com triângulos foi executado com uma ferramenta de triangulação existente no 3D Studio MAX, no entanto os projectistas editaram manualmente este resultado de acordo com as suas preferências. Este modelo realizado em CAD serviu mais tarde como base de trabalho para desenvolver o modelo detalhado com a ajuda do software paramétrico.

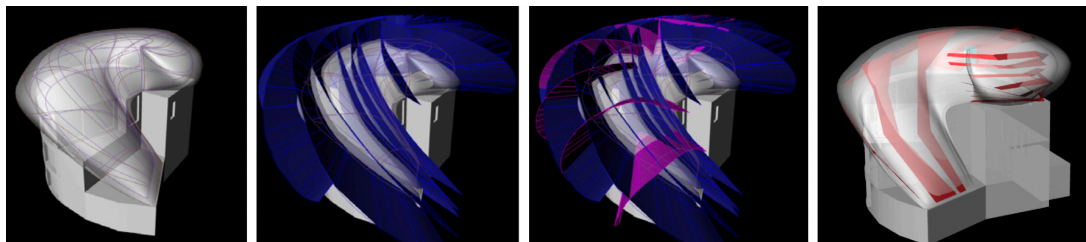


Fig 17 - Bankside Loft (Modelo 3d realizado no Rhinoceros)
Referência: Kihong Ku

Depois de definida a ideia, os arquitectos migraram o trabalho do 3D Studio MAX para o Rhinoceros3D (um software de tipologia CAD) para articular a geometria com os engenheiros civis, onde o modelo iria ser desenvolvido com uma superfície dupla (offset). Os arquitectos iriam trabalhar na superfície interior do modelo enquanto que os engenheiros trabalhariam na superfície exterior para definir a espessura necessária. O modelo geométrico foi seccionado com vários planos longitudinais e transversais para definir melhor a estrutura de suporte deste volume. Neste modelo, os arquitectos tiveram dificuldade em inserir planos de vidro quadriláteros porque era frequente o quarto vértice não ser coincidente com a superfície, sendo necessário ajustar o plano adjacente manualmente. A equipa de engenheiros determinou que a estrutura não era auto-sustentável e, por esse motivo, sugeriu que a estrutura fosse feita de forma treliçada (wireframe) com tubos de aço sobre a qual iria ser aplicado o revestimento. O arquitecto e os engenheiros da Arup acabaram por concordar com uma solução feita com painéis triangulares pré-fabricados para minimizar os custos e as dificuldades associadas ao processo de soldar em obra. O revestimento seria realizado através de painéis compósitos pré-fabricados através de processos/ tecnologia CNC (Computer Numeric Control).⁵⁵ Estes painéis seriam cortados através

⁵⁴ KU, Kihong - CAD/CAM-MEDIATED DESIGN AND CONSTRUCTION: A CASE STUDY OF THE BANKSIDE LOFT PROJECT. Second International Conference World of Construction Project Management 2007. 2007) 11. p3

⁵⁵ KU, Kihong - CAD/CAM-MEDIATED DESIGN AND CONSTRUCTION: A CASE STUDY OF THE BANKSIDE LOFT PROJECT. Second International Conference World of Construction Project Management 2007. 2007) 11. p5

do molde fornecido pelos softwares CAD e CAM e o objectivo da construção era colar estes painéis com cola à base de resina e reforçar a sua ligação/união com parafusos.

Na execução do modelo geométrico da estrutura, os arquitectos identificaram dificuldades em modelar manualmente as conexões entre cada painel triangular porque as arestas chanfradas originavam vértices/cantos complexos. Uma outra dificuldade foi modelar o vão das janelas. Para descobrir se o modelo era robusto, Goulthorpe encomendou um modelo protótipo à fábrica para testar fisicamente o resultado. Este protótipo demonstrou robustez estrutural e revelou um elevado grau de precisão no processo de fabrico com base em softwares CAM.⁵⁶

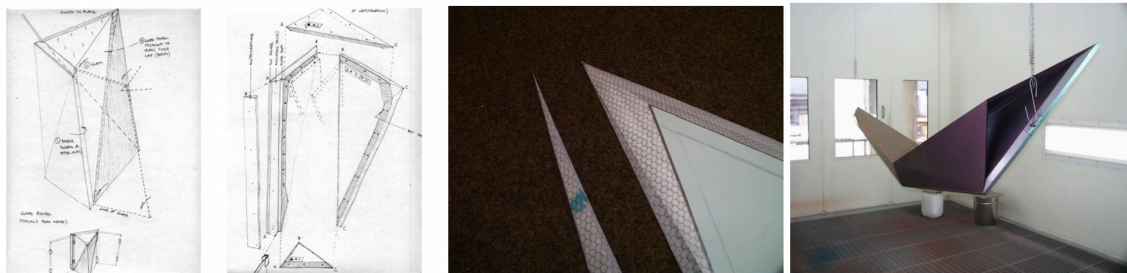


Fig 18 - Bankside Loft (Pormenor construtivo do painel de revestimento)
Referência: Kihong Ku

Para melhorar a capacidade térmica da estrutura, os painéis de vidro não eram coincidentes com as arestas, introduzindo ainda mais complexidade no modelo geométrico. Na opinião do autor, o Rhinoceros 3D não era adequado para esta tarefa e, para resolver esta complexidade, os arquitectos decidiram mudar de novo o trabalho para um outro programa de computador CAD, o CATIA.

Ao extrudir as superfícies triangulares (na direção da sua normal) do modelo geométrico no CATIA (processo conhecido como offset) os arquitectos identificaram anomalias geométricas nas intercepções dos painéis triangulares. Para resolver esta dificuldade, os projectistas decidiram fazer o "offset" para o lado interior do modelo geométrico, onde estas exceções podiam ser escondidas.

Os projectistas decidiram posteriormente mudar mais uma vez de software, abandonando o CATIA e optaram por continuar a desenvolver o modelo geométrico no GenerativeComponents da Bentley, um programa de tipologia AAD com mais capacidades paramétricas.

⁵⁶ KU, Kihong - CAD/CAM-MEDIATED DESIGN AND CONSTRUCTION: A CASE STUDY OF THE BANKSIDE LOFT PROJECT. Second International Conference World of Construction Project Management 2007. 2007) 11. p6

Apesar da comunicação do modelo geométrico ser feita através de elementos 2D (pelo tipo de fabrico) com a fábrica, estes documentos são concebidos com base no modelo 3D. "The final 2D cutting templates are directly derived from the model, as opposed to traditional 2D construction documents which are only design intents to be reinterpreted and regenerated by a contractor."⁵⁷ Goulthorpe reconhece uma dificuldade na comunicação de projectos através de elementos 2D e apoia alternativas de comunicação 3D. "We have to move into 3D descriptions as I don't see how else you communicate something like this."⁵⁸

Por fim, Kihong Ku identifica que a utilização de software CAD permite um maior domínio da forma geométrica, mas também sugere a metodologia CAD como forma de desenvolver soluções de construção alternativas aos métodos de trabalho convencionais. Ele identifica que os processos actuais de criação e construção de projectos são muito fragmentados e pouco eficientes. Para tal ele sugere que os software CAD sejam aplicados não só como forma de realizar o projecto, mas também para incorporar/integrar o processo construtivo do mesmo.

⁵⁷ KU, Kihong - CAD/CAM-MEDIATED DESIGN AND CONSTRUCTION: A CASE STUDY OF THE BANKSIDE LOFT PROJECT. Second International Conference World of Construction Project Management 2007. 2007) 11. p9

⁵⁸ GOULTHORPE, Mark apud KU, Kihong - CAD/CAM-MEDIATED DESIGN AND CONSTRUCTION: A CASE STUDY OF THE BANKSIDE LOFT PROJECT. Second International Conference World of Construction Project Management 2007. 2007) 11. p9

4.2 - Caso de estudo BIM - Helsinki Music Center



Fig 19 - Helsinki Music Center

Referência: <https://www.uniarts.fi/en/locations/helsinki-music-centre/>

O centro musical de Helsinki, na Finlândia foi um edifício criado para servir os eventos musicais desta cidade que tem como objectivo fornecer condições acústicas ideais/confortáveis. Este edifício pode ser encontrado na zona central da cidade próximo do parlamento e do museu Kiasma e serve diversos grupos musicais entre outras atividades musicais relacionadas com a educação musical.

O Helsinki Music Center foi inaugurado em 2011 e tem um auditório principal com capacidade para 1700 pessoas e 5 auditórios secundários com capacidades entre as 140 e as 400 pessoas. Um dos objectivos desta obra é oferecer melhores condições acústicas do que o edifício existente construído por Alvar Aalto em 1971, o Finlandia Hall.

O concurso para o projecto foi ganho no ano de 2000 pelo grupo "LPR Arkkitehdit Oy" que iria trabalhar em conjunto com o grupo de consultorias acústicas "Nagata Acoustics".

Quando a proposta do projecto inicial foi submetida para avaliação, concluíram que esta excedia significativamente o orçamento previsto para a obra e, por esse motivo, o projecto foi consecutivamente recriado. A primeira proposta foi realizada com base em metodologias CAD, no entanto em 2004, a proposta foi

recomeçada na metodologia BIM e os planos existentes foram convertidos para este modelo.⁵⁹

Aproveitando as capacidades de análise que o modelo BIM oferece, a Finlândia, mais concretamente "Senate Properties" promove o uso de modelos BIM, estabelecendo regras e convenções para a construção nacional. Estas medidas são cada vez mais necessárias neste País e o seu objectivo é promover/incorporar estratégias de desenvolvimento sustentável nos seus edifícios e na construção.⁶⁰ Um dos maiores focos destas medidas é reduzir o consumo de energia, uma vez que se trata de uma região fria e, para que a população tenha conforto, os consumos energéticos são geralmente maiores.

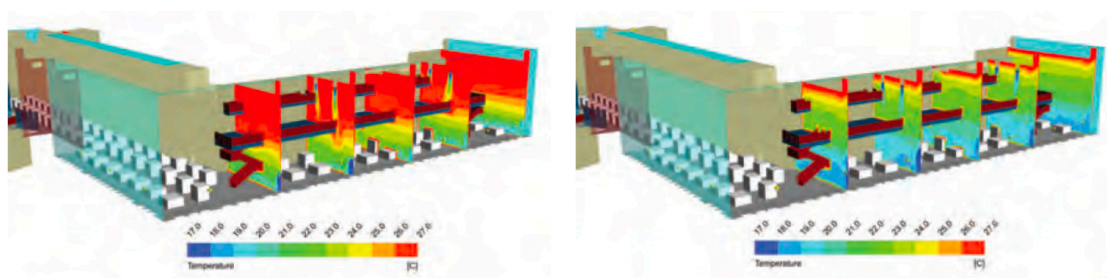


Fig 20 - Helsinki Music Center (Análise da temperatura do ar no Foyer)
Referência: Charles Eastman et al.

Durante a execução deste projecto foram realizadas diferentes análises. Uma dessas análises pretendia analisar a permanência de pessoas nos diversos espaços do edifício ao longo do tempo. Esta análise permitiu aos arquitectos distribuírem de forma eficiente o programa do projecto nos diferentes espaços do edifício. "The architects used BIM to calculate the floor area usage and generated the area distribution report for the main occupant groups, sometimes as frequently as every 15 minutes. The floor area distribution reports helped in guiding design decisions with regard to spatial allocation among the tenants and to determine preliminary cost estimates. It was thus very important to generate up-to-date and accurate area distribution reports."⁶¹

Nas metodologias de trabalho convencionais (como o CAD), a realização de análises (energéticas ou acústicas por exemplo) requer a criação de modelos

⁵⁹ EASTMAN, Charles M. et al. (EDS.) - BIM handbook: a guide to building information modeling for owners, managers, designers, engineers and contractors. 2nd ed ed. Hoboken, NJ : Wiley, 2011. ISBN 978-0-470-54137-1. p542

⁶⁰ EASTMAN, Charles M. et al. (EDS.) - BIM handbook: a guide to building information modeling for owners, managers, designers, engineers and contractors. 2nd ed ed. Hoboken, NJ : Wiley, 2011. ISBN 978-0-470-54137-1. p543

⁶¹ EASTMAN, Charles M. et al. (EDS.) - BIM handbook: a guide to building information modeling for owners, managers, designers, engineers and contractors. 2nd ed ed. Hoboken, NJ : Wiley, 2011. ISBN 978-0-470-54137-1. p544

específicos que exigem horas de trabalho. Assim sendo, estas análises costumam ser realizadas apenas nas fases mais avançadas do trabalho em que a forma final do projecto está mais consolidada. Esta solução não é muito eficiente porque nas fases finais do trabalho, o projecto tem menos flexibilidade para alterar as suas formas em prol das condições energéticas ou acústicas. A metodologia de trabalho BIM oferece uma solução eficiente a este problema porque o modelo IFC incorpora não só informação geométrica, mas também armazena as características dos materiais. Por este motivo, o modelo BIM pode também ser utilizado para se realizarem análises. Esta solução permite que as análises possam ser executadas nas fases iniciais do trabalho e que o projecto possa nestas fases adaptar a sua forma para responder melhor às questões energéticas ou acústicas.

O Helsinki Musical Center é um caso de estudo que demonstrou esta vantagem presente na metodologia de trabalho BIM. "This case study demonstrates that advanced energy simulation can be integrated with the design process. It shows how the team used the results of these analyses to modify and influence aspects of the design."⁶²

Neste projecto, o modelo IFC foi utilizado para determinar a exposição solar, o que ajudou a decidir o tipo de janelas na fachada de vidro. Um outro exemplo que demonstra como o projecto beneficiou do modelo IFC foi a análise energética realizada para o foyer que determinou que o sistema inicial estava a produzir calor em excesso na parte superior deste espaço. Esta análise ajudou a determinar uma posição ideal dos sistemas AVAC para tornar mais eficiente o conforto neste espaço.

Durante a execução de projectos como este há também uma grande necessidade de comunicação entre as diversas entidades participantes (arquitectos, engenheiros civis, engenheiros de especialidades, consultores, etc) para que todos tenham conhecimento das alterações realizadas em paralelo e trabalhem sempre na versão mais actualizada do projecto. Neste contexto, o modelo BIM (também conhecido como modelo IFC) funciona como uma plataforma flexível na qual podem participar diversas entidades através de diversos softwares.

A utilização do modelo BIM foi vantajosa para o Helsinki Musical Center porque foi criado para a construção deste edifício e actualmente é utilizado para auxiliar a manutenção do mesmo. Uma das virtudes em usar o modelo BIM é que este pode ser vantajoso para desenvolver projectos não só para a sua construção, mas também para a sua manutenção ao longo da sua existência.

⁶² EASTMAN, Charles M. et al. (EDS.) - BIM handbook: a guide to building information modeling for owners, managers, designers, engineers and contractors. 2nd ed ed. Hoboken, NJ : Wiley, 2011. ISBN 978-0-470-54137-1. p546

4.3 - Caso de estudo AAD - Kai Strehlke

Este escritório foi fundado na cidade de Basel na Suíça em 1978 por Jacques Herzog e Pierre de Meuron e, desde então, foi expandindo a sua atividade ao longo dos anos estando actualmente representado por sedes em Madrid, Hamburgo, Londres e Nova Iorque.

Kai Strehlke participou neste escritório no Digital Technology Group onde desenvolveu soluções paramétricas e algorítmicas para os projectos desenvolvidos. Segundo Strehlke, a expansão do escritório e a crescente complexidade dos seus projectos tornou-se cada vez mais difícil de gerir e, para dar resposta a este problema, o escritório teve que se adaptar. Para além disso, a quantidade de informação que é gerida pelos arquitectos é cada vez maior para um reduzido tempo de trabalho. No gabinete de Herzog & de Meuron os instrumentos analógicos (maquetes, desenhos e esboços por exemplo) são utilizados em articulação com os instrumentos digitais, especialmente na fase inicial do projecto para desenvolver melhor a ideia/conceito geométrico. Com o intuito de melhorar a eficiência de trabalho neste escritório, foram introduzidas metodologias BIM para realizar alguns projectos mais regulares. No entanto, Strehlke afirma que nem sempre BIM é a ferramenta de trabalho mais ideal para concretizar os projectos deste escritório (Herzog & de Meuron). Em primeiro lugar, porque alguns arquitectos preferem utilizar ferramentas/metodologias de CAD por serem mais intuitivas. Em segundo lugar, porque neste escritório as equipas de trabalho desenvolvem diversas propostas semelhantes para o mesmo projecto e, segundo ele, é difícil modelar um projecto pouco consolidado. Em terceiro lugar, os projectos realizados por este escritório têm formas únicas, não padronizadas ("standardized") e resultam numa dificuldade em aplicar a metodologia BIM. O BIM continua a ser utilizado neste escritório mas, por vezes, é difícil responder às necessidades dos seus projectos utilizando esta metodologia. "the uniqueness of each project makes it difficult to apply a BIM model. We have been working with BIM software and are continuing to use it, but not always with success."⁶³

Para resolver algumas tarefas mais complexas do ponto de vista da geometria são desenvolvidas ferramentas digitais específicas. A utilização de software digital tem uma vantagem porque neste escritório existe uma oficina onde são utilizadas diversas máquinas para construir modelos 3D físicos/reais para que os projectistas possam manusear no escritório. "Our in-house machine park consists of three generic

⁶³ STREHLKE, Kai - Digital Technologies, Methods, and Tools in Support of the Architectural Development at Herzog & de Meuron. ACADIA. 2009. p26

CNC-machines: a large, size 3-axes milling machine, a laser-cutting system, and a cutting plotter."⁶⁴

O "Digital Technology Group" foi criado neste escritório para ajudar a execução dos projectos mais complexos e com o objectivo de melhorar as capacidades criativas dos trabalhos desenvolvidos. Este grupo tem como função desenvolver ferramentas digitais (com programação) específicas para dar resposta a cada uma das ideias desenvolvidas neste escritório. Segundo Strehlke, o objectivo é reduzir o número de inputs necessários automatizando a maior parte do processo de projetar mantendo um resultado/output que possa ser facilmente adaptado às exigências do projecto. Para elaborar este processo digital de projetar, o grupo procura criar um algoritmo que melhor represente as ideias do projecto de uma forma geral mas, por vezes, o número de variações/situações específicas é elevado e isto transforma o algoritmo num conjunto de excessões, o que não aproveita a eficiência dos sistemas digitais. Quando isso acontece, Strehlke afirma que a equipa do projecto tem duas opções, ou os projectistas têm de aceitar soluções geométricas mais genéricas e regulares ou têm de abdicar da automatização paramétrica e seguir metodologias de trabalho mais convencionais/manuais. Para além disso, Strehlke fala da importância da velocidade no desenvolvimento do trabalho, os dois grupos (grupo digital e grupo de projecto) têm de trabalhar à mesma velocidade.

Strehlke fala também na vantagem de articular diversas ferramentas de trabalho digital que denomina como "Digital Chain". Este conceito refere-se à parametrização de todos os processos desde a criação das formas geométricas ao fabrico das peças e do edifício, onde ele indica que é difícil aplicar/inserir a totalidade do projecto numa digital chain. Segundo Strehlke, a digital chain tem de ser flexível porque é frequente os projectistas interromperem-na para inserir excessões através de inputs manuais. Uma das tarefas do grupo digital é garantir que a "Digital Chain" funciona mesmo com os inputs manuais inseridos pelos arquitectos. Quando a "Digital Chain" estiver completa, as formas geométricas digitais podem ser automaticamente exportadas para o processo de fabricação e consequentemente para a produção do edifício. Segundo Strehlke, uma das maiores vantagens da "Digital Chain" é resolver a ineficiência entre a criação dos projectos e a execução do edifícios. "The architect takes responsibility for the design; the construction company, for the construction. The complexity of some of today's architecture makes this process expensive and inefficient. The digital chain overcomes this separation. This is

⁶⁴ STREHLKE, Kai - Digital Technologies, Methods, and Tools in Support of the Architectural Development at Herzog & de Meuron. ACADIA. 2009. p27

a very new reality in architecture and has yet to be evaluated from the viewpoint of liability and risks."⁶⁵

O escritório de Herzog & de Meuron já executou alguns projectos onde estas posturas teóricas foram aplicadas e que demonstram de que forma a programação digital pode funcionar a favor da qualidade do projecto através da parametrização. Um desses projectos é o centro cultural desenvolvido para a cidade de Flamenco em Jerez de la Frontera em Espanha no ano de 2003. Um outro edifício realizado por este escritório é a expansão do Tate Modern em Londres onde as ferramentas paramétricas como o Grasshopper foram utilizadas para calcular o pormenor construtivo da colocação do tijolo na fachada e posteriormente se fabricarem os componentes necessários à construção.

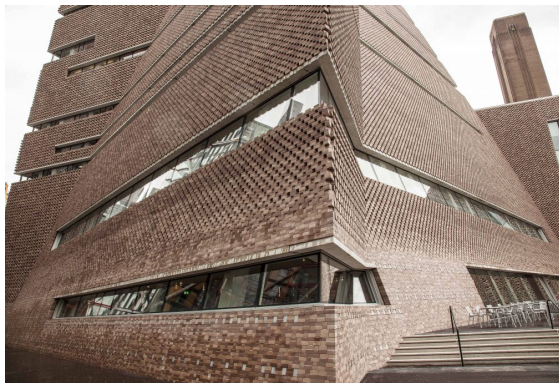


Fig 21 - Expansão do Tate Modern
Referência: <http://www.grandtourmagazine.com/magic-brick-new-tate-modern-extension/>



Fig 22 - Centro Cultural em Ciudad del Flamenco
Referência: <https://www.herzogdemeuron.com/index/projects/complete-works/226-250/240-ciudad-del-flamenco/image.html>

⁶⁵ STREHLKE, Kai - Digital Technologies, Methods, and Tools in Support of the Architectural Development at Herzog & de Meuron. ACADIA. 2009. p27

4.4 - Caso de estudo AABIM - lasbik, Martinez e Gazel

A última década viu crescer o uso de editores de programação visual aplicados a software de criação geométrica (tais como o Rhinoceros3D GrassHopper, o Autodesk Dynamo ou o Archicad Param-O), resultando numa nova metodologia de trabalho AAD. Algumas entidades observaram o benefício em associar a programação visual com a metodologia BIM tais como Harri Humpi e Toni Österlund que apelidaram esta nova metodologia de AAB (Algorithm-Aided BIM). AABIM é uma metodologia de trabalho digital relativamente recente e, por esse motivo, conta com um ainda reduzido número de programas de computador que permitem trabalhar de forma eficiente nesta metodologia de trabalho digital. É neste contexto que Marina lasbik, Andressa Martinez e Jorge Gazel desenvolveram um estudo no qual analisaram a integração do Grasshopper com programas BIM.

Os autores começam por afirmar que actualmente os softwares de tipologia AAD por si só não gerem a informação técnica dos elementos construtivos presentes no modelo IFC (como por exemplo a materialidade). Ao associar os softwares de criação algorítmica com os softwares de tipologia BIM, é possível integrar possibilidades/características paramétricas ao modelo IFC. Para aplicarem os conhecimentos desta postura teórica, os autores vão aplicar esta metodologia (AABIM) para criarem um modelo de habitação social. Este caso de estudo vai permitir compreender mais facilmente as virtudes e os limites de aplicar características paramétricas ao modelo IFC com os recursos actuais.

Ao iniciarem o trabalho, os autores começaram por fazer uma investigação bibliográfica nos temas do BIM, parametria e interoperabilidade para identificar como customizar em massa o projecto de arquitectura. Posteriormente, fizeram um apuramento dos softwares existentes que permitem integrar o GrassHopper com dois dos principais programas BIM (o Graphisoft Archicad e o Autodesk Revit). Esta interoperabilidade é geralmente realizada sob a forma de plugins/addons em que a informação é partilhada entre programas através de API, CSV ou através do modelo IFC. Nesta Investigação, os autores escolheram dois plugins, o RhinoInsideRevit para fazer a ligação do GrassHopper com o Revit e o ArchicadLiveConnection para articular o Grasshopper com o Archicad. Estes dois plugins oferecem a capacidade de trabalhar no modelo BIM em tempo real e permitem ao utilizador ver as mudanças no software BIM. Esta foi uma das razões que motivou a escolha destes plugins. "Os plugins escolhidos (ALC e RIR) permitem que a geometria seja criada no Grasshopper de acordo com a linguagem, a descrição de componentes construtivos e configurações nativas do Archicad e do Revit, respectivamente. Estes foram os únicos plugins testados com conexão em tempo real com os software BIM, uma vez

que operam bidirecionalmente, mantendo as propriedades intrínsecas do modelo BIM, sem a necessidade de parar a exploração formal, exportar a geometria e importar para o novo software. Segundo Guidoux Gonzaga et al. (2018) este processo representa uma continuidade no fluxo projetual dentro do ambiente digital, tornando-o mais eficiente em comparação aos demais plugins analisados, citados adiante."⁶⁶

Para executar o modelo de habitação social, os autores realizaram testes de interoperabilidade entre as duas plataformas. Todos os elementos construtivos (como lajes, vigas ou paredes por exemplo) são parametrizadas facilitando a integração dos atributos do BIM no modelo AAD.

Para articular softwares AAD e softwares BIM, os autores identificaram 3 processos/fluxos de trabalho (workflows).

O primeiro processo é exportar a geometria do programa AAD num formato .sat ou .dwg para o programa BIM, mas concluíram que deste processo resultam dificuldades na edição dos objectos.

O segundo processo é a exportação do modelo paramétrico sob a forma de superfícies para o software BIM. Depois de importadas no software BIM a partir destas superfícies, podem ser criados elementos como paredes, lajes e coberturas. Esta abordagem também não é a mais eficiente porque segundo os autores "nem sempre o software BIM reconhece ou redefine o elemento host, sendo necessária a criação de novos elementos"⁶⁷.

O terceiro processo consiste na utilização de plugins que, segundo os autores permite o resultado mais eficiente.

Os autores reconhecem que alguns softwares BIM já possuem sistemas de programação visual próprios através de plugins embutidos (como por exemplo o Dynamo ou o Param-O), mas identificam alguns limites nestes editores paramétricos e, por esse motivo, recorrem à integração do GrassHopper nos programas BIM através de plugins. Para realizar a interoperabilidade entre o GrassHopper e os sistemas BIM, os autores examinaram/experimentaram diversos plugins tais como: o Hummingbird, o Lyrebird, o Grevit, o Rhynamo, o Mantis Shrimp, o Geometry Gym, o VisualARQ, o Archicad Live Connection e por fim o RhinoInsideRevit.

⁶⁶ IASBIK, Marina Pires; MARTINEZ, Andressa Carmo Pena; GAZEL, Jorge Lira De Toledo - Integration of BIM and Algorithmic Design logics through data exchange between Grasshopper plugin and Revit and Archicad software. Em Blucher Design Proceedings [Em linha]. Medellín, Colombia : Editora Blucher, dez. 2020 [Consult. 9 abr. 2022]. Disponível em WWW:<URL:<http://www.proceedings.blucher.com.br/article-details/35473>>. p471

⁶⁷ IASBIK, Marina Pires; MARTINEZ, Andressa Carmo Pena; GAZEL, Jorge Lira De Toledo - Integration of BIM and Algorithmic Design logics through data exchange between Grasshopper plugin and Revit and Archicad software. Em Blucher Design Proceedings [Em linha]. Medellín, Colombia : Editora Blucher, dez. 2020 [Consult. 9 abr. 2022]. Disponível em WWW:<URL:<http://www.proceedings.blucher.com.br/article-details/35473>>. p472

De acordo com Janssen, Chen e Mohanty, os autores de trabalho classificam os procedimentos resultantes do uso destes plugins em duas categorias: as "abordagens fortemente acopladas" e as "abordagens fracamente acopladas".⁶⁸

Nas "abordagens fortemente acopladas" "Tightly Coupled" (conseguidas com o Hummingbird, o Lyrebird, o Grevit, o Rhynamo e o Mantis Shrimp) os programas AAD e BIM interligam-se através do API (Application Programming Interface) fornecido pelo programa BIM.

Nas "abordagens fracamente acopladas" "Loosely Coupled" (conseguidas com o Geometry Gym e o VisualARQ) os programas AAD e BIM interligam-se através do modelo IFC, permitindo a capacidade para escolher o material directamente no modelo AAD.

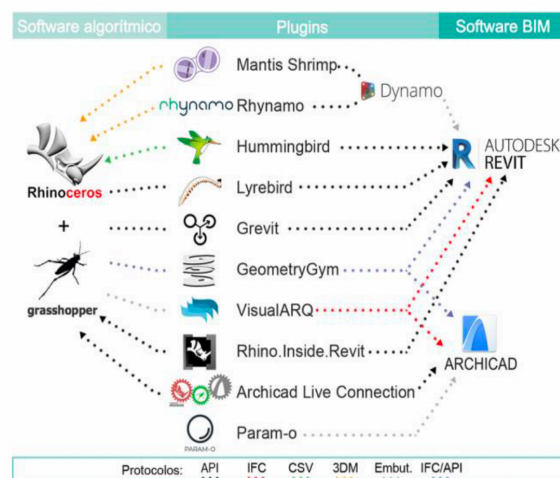


Fig 23 - Processos de trabalho AABIM

Para porem em prática os conceitos teóricos promovidos por esta metodologia (AABIM) e também para executarem testes de interoperabilidade, os autores propuseram a criação de um projecto habitacional paramétrico realizado nestes softwares mencionados. Este projecto habitacional faz parte de um trabalho acerca da reconstrução de habitações e realojamento das populações do município de Bento Rodrigues no Brasil que, em 2015 sofreu um desastre ambiental por causa do rompimento de barragens, resultando no desalojamento de 226 famílias.⁶⁹

⁶⁸ IASBIK, Marina Pires; MARTINEZ, Andressa Carmo Pena; GAZEL, Jorge Lira De Toledo - Integration of BIM and Algorithmic Design logics through data exchange between Grasshopper plugin and Revit and Archicad software. Em Blucher Design Proceedings [Em linha]. Medellín, Colombia : Editora Blucher, dez. 2020 [Consult. 9 abr. 2022]. Disponível em WWW:<URL:http://www.proceedings.blucher.com.br/article-details/35473>. p472

⁶⁹ IASBIK, Marina Pires; MARTINEZ, Andressa Carmo Pena; GAZEL, Jorge Lira De Toledo - Integration of BIM and Algorithmic Design logics through data exchange between Grasshopper plugin and Revit and Archicad software. Em Blucher Design Proceedings [Em linha]. Medellín, Colombia : Editora Blucher, dez. 2020 [Consult. 9 abr. 2022]. Disponível em WWW:<URL:http://www.proceedings.blucher.com.br/article-details/35473>. p473

Os autores sugerem esta metodologia de trabalho (AABIM) para criarem projectos habitacionais porque oferece ferramentas que permitem reduzir os custos na construção de edifícios mantendo a qualidade, através da utilização de computadores na fabricação. Para além disso, a utilização de solvers como por exemplo o Galapagos no GrassHopper permite utilizar a capacidade de processamento do computador para encontrar os modelos mais eficientes/baratos.

A criação do modelo geométrico do edifício habitacional no GrassHopper consiste na utilização de módulos estruturais que podem ser agrupados em diversas composições de dois módulos até um máximo de 5 módulos. Estes têm 4,8m por 6m (comprimento e largura) e o seu sistema estrutural tem como objectivo facilitar o transporte e a montagem no local.

O script de programação visual criado no GrassHopper para este projecto permite ajustar diversos parâmetros como por exemplo as dimensões dos pilares, as dimensões das vigas, o número de pisos ou o posicionamento dos módulos por exemplo. Com a ajuda dos plugins (Archicad Live Connection e Rhino Inside Revit), esta estrutura geométrica é importada em tempo real para o software BIM sobre a qual são criados/gerados os elementos do modelo IFC.

Criação dos pilares:

Para se criarem os pilares do modelo estrutural no Archicad através do ArchicadLiveConnection são necessários o ponto do início e o ponto do fim do pilar para se inserirem no node "column" do ALC (este node também permite escolher atributos como a materialidade e a função estrutural). Para se criarem os mesmos pilares no Revit através do RhinoInsideRevit é necessário definir a linha do eixo deste pilar no node "add column" (este node também permite escolher atributos como o tipo de pilar e a categoria do modelo).

Criação das vigas:

Para se criarem as vigas do modelo no Archicad através do ArchicadLiveConnection é necessário definir a linha do eixo da viga no node "beam" (este node também permite escolher atributos como o ID, a função estrutural e o material por exemplo). Para se criarem as mesmas vigas no Revit através do RhinoInsideRevit é realizado o mesmo procedimento no node "add beam" (este node também permite escolher atributos como o tipo de viga e a categoria do modelo).

Criação das paredes:

Para se criarem as paredes do modelo no Archicad através do ALC é possível definir 4 arestas do plano da superfície no node "wall" (este node também permite escolher atributos semelhantes às vigas e aos pilares). Para se criarem os mesmos revestimentos no Revit através do RIR é necessário definir apenas um eixo no node "add wall" (este node também permite escolher outros atributos específicos do Revit).

Criação das lajes:

Para se criarem as lajes do modelo no Archicad através do ALC o procedimento é muito semelhante ao das paredes sendo necessário definir um polígono no node "slab" (este node pode ser ligado ao node "slab settings" para se escolherem os atributos). Para se criarem estas lajes no Revit através do RIR é necessário definir os limites da laje através de "boundaries" no node "add floor" (este node permite escolher atributos específicos do Revit).

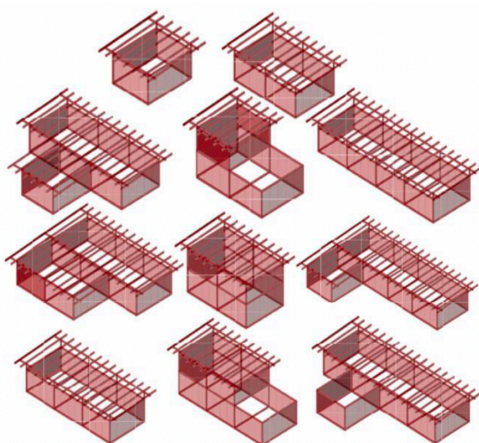


Fig 24 - Tipologias dos Módulos de Habitação - Ref: lasbik, Martinez e Gazel



Fig 25 - Renderização dos Módulos de Habitação - Ref: lasbik, Martinez e Gazel

No final do trabalho, os autores identificaram diversas observações. O facto dos programas BIM aceitarem elementos geométricos para definir a forma dos elementos construtivos facilitou a integração do GrassHopper. Para além disso, os autores identificaram que o plugin RhinoInsideRevit oferece um maior número de atributos e organização de informação em listas.

Eles observaram que o ArchicadLiveConnection é mais rápido a executar os cálculos e que o RhinoInsideRevit necessita de maior capacidade de processamento. Entre estas e outras observações retiradas houve alguns motivos que levaram os autores a considerar o Archicad Live Connection uma solução mais eficiente para transferir dados paramétricos para o modelo BIM.

Os autores descobriram também que os inputs na metodologia AABIM têm uma tendência linear mono-direcional. No caso do Rhino Inside Revit, todas as edições/inputs ao modelo paramétrico têm que ser realizadas através de programação visual (no GrassHopper) e não ao contrário, porque se o utilizador desbloquear as entidades no Revit, estas entidades deixarão de ser recalculadas pelo modelo paramétrico (GrassHopper).

Ao contrário do RIR, no caso do Archicad Live Connection, os autores observaram que mesmo depois de desbloqueadas, as entidades ainda permitem algum controlo. Os autores identificaram este problema do fluxo de trabalho mono-direcional como um dos limites desta metodologia AABIM porque em diversas situações, o utilizador necessita de fazer inputs através da plataforma BIM e estes inputs são dificilmente integrados no modelo paramétrico (AAD).

Os autores referem também uma outra dificuldade nesta metodologia relativamente à portabilidade e compatibilidade dos modelos paramétricos e dos softwares BIM. Isto deve-se ao facto de cada software BIM desenvolver características específicas ao seu produto (Revit ou Archicad por ex). Estes softwares não seguem uma convenção universal/comum como o IFC e isso resulta numa dificuldade de compatibilidade entre as ferramentas de cada um. Neste contexto, os autores afirmam que partes do script GrassHopper não são portáteis quando se muda de software e, por esse motivo, têm que ser traduzidas (por exemplo os nodes do RIR não são compatíveis com o Archicad e os nodes do ALC não são compatíveis com o Revit). "Por sua vez, o recurso de portabilidade dos plugins do Grasshopper possui algumas limitações. Em diversos casos, partes significativas do script não são portáteis e várias operações são refeitas para a geração da geometria em diferentes software. (...) As barreiras de interoperabilidade devem-se em parte às limitações da API do software BIM e também à falta de desenvolvimento do formato de arquivo IFC, que busca interoperabilidade universal no setor de AEC."⁷⁰

Por fim, os autores relembram uma grande vantagem proveniente destes modelos algorítmicos. Eles permitem reduzir/resumir número de inputs/variáveis do projecto a uma lista simples porque muitos parâmetros do projecto são automaticamente recalculados pelo computador. Segundo eles, isto permite ao arquitecto criar um modelo paramétrico e oferecer a possibilidade de um cliente ajustar/personalizar os parâmetros do projecto (como materiais de revestimento ou o número de pisos) em função do seu gosto ou orçamento.

⁷⁰ IASBIK, Marina Pires; MARTINEZ, Andressa Carmo Pena; GAZEL, Jorge Lira De Toledo - Integration of BIM and Algorithmic Design logics through data exchange between Grasshopper plugin and Revit and Archicad software. Em Blucher Design Proceedings [Em linha]. Medellín, Colombia : Editora Blucher, dez. 2020 [Consult. 9 abr. 2022]. Disponível em WWW:<URL:<http://www.proceedings.blucher.com.br/article-details/35473>>. p473

4.5 - Caso de estudo Programação Escrita - Maria Gkioka

Em 2018, Maria Gkioka, desenvolveu o seu trabalho de mestrado de engenharia civil na Universidade Holandesa DELFT. O trabalho de Maria é importante porque representa um caso de estudo que exemplifica muito bem como a programação textual pode ser utilizada/aplicada para beneficiar o trabalho dos engenheiros na realização do projecto. (Esta postura teórica pode também ser adaptada/aplicada ao trabalho dos arquitectos)

No seu trabalho, a autora desenvolveu um exercício no qual utilizou a programação (visual e textual) para criar algoritmos capazes de parametrizar o projecto de um viaduto tirando partido das capacidades de cálculo do computador. Esta abordagem otimiza a execução do projecto porque divide os parâmetros opcionais dos parâmetros necessários. Cada vez que o projectista define/altera os parâmetros flexíveis (como os alinhamentos), o computador recalcula automaticamente os parâmetros fixos do projecto. "The results of this conceptual study indicate that the designers can highly benefit from such an approach. Indeed in the case of a symmetrical viaduct design the number of user-defined parameters are 30 out of 82 (36.5%). This means that the design time can be drastically reduced, since 63.5% of the design issues considered in this study are automatically addressed by the platform."⁷¹

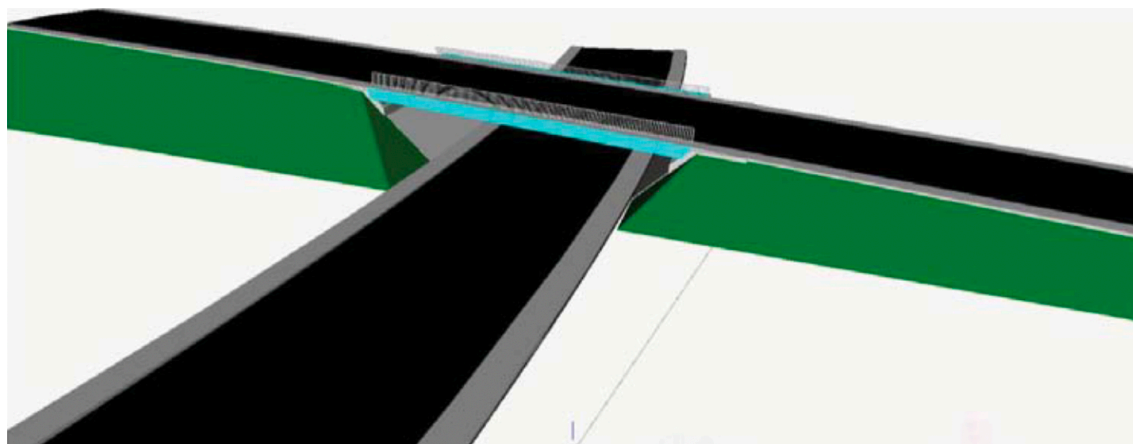


Fig 26 - Modelo 3D do Viaducto
Referência: Maria Gkioka

⁷¹ GKIOKA, Maria et al. - Novel Parametric Knowledge Modelling approach applied to Viaducts. 2019) 9. p1114

Um outro aspecto a salientar foram as metodologias de trabalho digital utilizadas por Gkioka para conseguir atingir este resultado. Gkioka utilizou programação escrita para articular múltiplos programas de computador em tempo real. Esta característica permite expandir os limites de cada programa de computador porque estabelece pontes de comunicação de informação com outros programas de computador através do uso de API's. "Fourthly, the resulting computer applications often do not have to sit on their own but can connect to other applications through API's (Application Programming Interfaces) which allow them to retrieve data from other applications or send data to other applications and use the functionality in other applications."⁷²

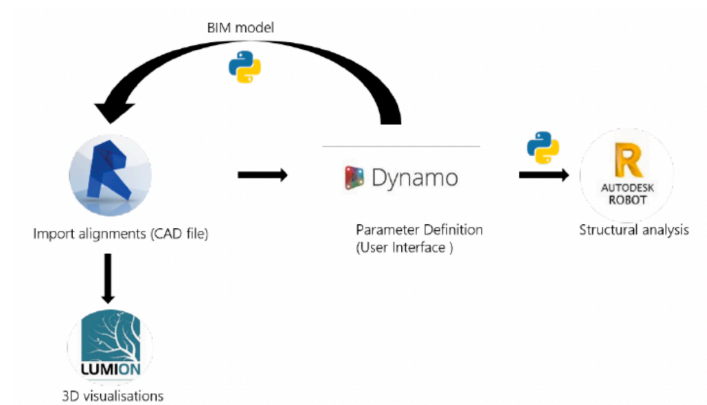


Fig 27 - Hiperligações/conexões entre os diversos softwares utilizados para criar o viaduto
Referência: Maria Gkioka

A autora salienta que uma das suas motivações era aproximar o trabalho dos arquitectos, engenheiros e construtores através das metodologias de trabalho paramétricas como meio para combater a fragmentação entre estas áreas. A dissertação de Maria Gkioka é realizada com base num contexto existente, o projecto rodoviário para Hoevelaken na qual ela desenvolve um viaduto para a interseção da A28 com a A1. O modelo geométrico tem como base alguns parâmetros criados no Autodesk Revit (tais como alinhamentos, e os eixos das estradas). Com base nestes parâmetros iniciais, Gkioka desenvolveu o modelo geométrico do viaduto de forma paramétrica no Dynamo (através de programação visual e escrita). A intenção inicial era utilizar o Office Excel como input para gerir os parâmetros e as variáveis utilizadas para controlar o modelo geométrico (por exemplo a espessura da laje, dimensões e o número de apoios), no entanto o excel estava a causar dificuldades/demoras na computação do processo paramétrico. Por este motivo, Gkioka decidiu criar um interface GUI com Python utilizando o WinForms, (uma library para criar aplicações/

⁷² COENDERS, Jeroen L. - Next Generation Parametric Design. Journal of the International Association for Shell and Spatial Structures. . ISSN 1028-365X. 62:2 (2021) 153-166. doi: 10.20898/j.iass.2021.018. p157

interfaces gráficas GUI) existente no .NET Framework desenvolvido pela Microsoft. (Isto é um exemplo em que o trabalhador construiu a própria ferramenta digital, porque as existentes não satisfaziam a necessidade do trabalho).

Gkioka interligou o modelo paramétrico criado no Dynamo com o Autodesk Robot para efectuar a análise estrutural cada vez que altera o modelo geométrico e ver estes resultados em tempo real. Para conseguir estabelecer este link, Gkioka utilizou programação escrita e plugins para conectar o Dynamo ao Autodesk Robot. Gkioka utilizou também o API do Autodesk Revit e do Robot para aceder às propriedades intrínsecas dos objectos geométricos a partir do Dynamo porque segundo ela, isto permite criar um algoritmo mais especializado. A utilização de programação textual através de "Python code blocks" ajuda também a compactar o tamanho do algoritmo visual. "Revit allows the user to access its API, the application programming interface, that provides a way to programmatically access (using Python) the internal representation of BIM data in Revit. This enables the user to understand and alter the implicit information, hidden behind the Revit object families and redefine the attributes and spatial data of the objects. Gaining access to the way Revit represents objects, enables a more sophisticated and customized algorithm development and the ability to address the needs of the designer for the specific project of viaducts. Moreover, the use of Python code blocks can decrease the volume of the Dynamo script and the number of complicated wires connecting the components, referred to as "Spaghetti Problem". This solution however, leads to the development of a "black box", where the logic is concealed in textual coding nodes."⁷³

O modelo paramétrico criado com o Dynamo, é representado geometricamente no Autodesk Revit, a partir do qual é exportado para o Lumion para renderizar uma imagem mais realística do projecto do viaducto.

Neste caso de estudo da criação de um viaduto, Maria Gkioka conseguiu reduzir/resumir a execução do projecto a 82 parâmetros dos quais apenas 30 precisam de ser directamente definidos pelo engenheiro (os outros 52 parâmetros são automaticamente calculados pelo computador). No final do seu trabalho, Gkioka identificou diversas conclusões em diferentes áreas. Segundo ela, o modelo geométrico criado com esta metodologia pode ser muito facilmente ser convertido no modelo de BIM a partir do qual podem ser directamente calculados os volumes e o preço dos materiais. Este modelo geométrico 3D permite mais facilmente identificar erros e conflitos geométricos do que os métodos de trabalho 2D.

No entanto, Gkioka também identifica algumas desvantagens. O algoritmo feito com programação visual tem tendência a ficar complexo e grande resultando numa dificuldade de orientação. Para resolver esta questão, Gkioka sugere a

⁷³ GKIOKA, Maria - Knowledge Model for the Parametric Design of Concrete Viaducts. 2018. p17

utilização de blocos de programação textual e a criação de um interface visual GUI para controlar este algoritmo.

A programação visual do Dynamo permite uma compreensão mais intuitiva de algoritmo, no entanto a utilização de blocos programação textual permite compactar as instruções e, para além disso, a programação textual é executada muito mais rapidamente do que a programação visual. "It can be argued that introducing coding could facilitate this process, since each component could be described by a node of textual programming. Dynamo could be then used as a canvas to connect those components. That would significantly decrease the computational time needed to execute the script, as Python nodes run faster in Dynamo than the visual programming components."⁷⁴

Gkioka publicou em 2019 em "Proceedings of the IASS Annual Symposium 2019 - Structural Membranes" um artigo onde resumiu o seu trabalho de Mestrado e identificou as principais aprendizagens.

⁷⁴ GKIOKA, Maria - Knowledge Model for the Parametric Design of Concrete Viaducts. 2018. p82

Módulo 5

Exemplos de aplicações práticas de programação no trabalho do projectista.

5.0 - Descrição do Módulo 5

O Módulo 5 centra-se na temática da programação escrita e tem como objectivo explorar mais concretamente de que forma a utilização desta pode beneficiar directamente o trabalho do arquitecto.

A programação escrita tem uma elevada curva de aprendizagem, contudo, o domínio deste conhecimento permite um maior controlo do computador, pois as linguagens de programação permitem uma maior flexibilidade de opções, expandido as capacidades do utilizador (neste caso o arquitecto). Para explorar este conceito, o Módulo 5 apresenta 5 exercícios/exemplos que demonstram de que maneira a programação escrita pode ser integrada no trabalho do projectista e utilizada a favor do mesmo para desenvolver o projecto.

Comentar os scripts é uma boa prática frequentemente utilizada por programadores para ajudar a perceber o funcionamento do código/script, no entanto os scripts apresentados neste capítulo são pouco comentados, uma vez que o seu funcionamento é explicado na descrição do exercício.

Embora os exemplos realizados neste trabalho tenham um carácter simples e básico, eles já permitem ter uma noção das potencialidades da programação aplicada ao desenvolvimento do projecto.

5.1 - Exemplo 1 - Automatização de tarefas específicas ou rotinas no software

Contexto do exercício 1:

Os arquitectos desempenham várias tarefas durante a execução do projecto e para cada uma delas utilizam os softwares de computador que oferecem as ferramentas mais dedicadas para a sua realização, tal como por exemplo: projectar, modelar, renderizar, organizar informação numérica, simular ou animar. Algumas destas tarefas (como por exemplo renderizar) estão dependentes da evolução do projecto, ou seja, quando este é alterado, as renderizações ficam desactualizadas e precisam de ser realizadas/executadas de novo. Se a renderização for realizada num software externo ao software de modelação, é necessário reimportar o modelo 3D no programa de renderização e voltar a configurar as definições de renderização. Isto resulta num processo laborioso e demorado que terá que ser repetido cada vez que o projecto for alterado. Por estas razões, tarefas como esta só costumam ser realizadas nas fases finais de entrega/apresentação, quando o projecto tem menos probabilidade de ser alterado.

Este exercício foi desenvolvido como o objectivo de tornar esta tarefa mais eficiente e superar a dificuldade da fluidez automatizando o processo com um script de Python.

Para pôr em prática esta teoria foi criado um modelo 3D de uma casa num formato CAD (".dae" COLLADA) que será importado para o Blender 3D para se adicionarem materiais e animar a construção das peças. Tradicionalmente, este processo de adicionar materiais e animar seria feito de forma manual e exaustiva para cada uma das peças do modelo (portas, paredes, pilares, vigas, etc), mas através de um script em Python, é possível realizar todo este processo laborioso de forma automática.

Tendo em conta que o modelo CAD não armazena informações/atributos personalizados sobre os objectos (como o BIM), para este script conseguir identificar que material aplicar e como animar o objecto, é preciso que o arquitecto nomeie os objectos do modelo 3D com as informações deste objecto em letras minúsculas separado pelo caractere "_" pela seguinte ordem

(tipoDeObjecto_material_eixoDaTranslaçãoXYZ_distânciaDaTranslação_frameDeInicioDaAnimação_frameDeFimDaAnimação) por exemplo, "porta_madeira_y_6_60_90".

Assim, com este processo automatizado, o arquitecto só precisa de produzir o modelo geométrico do edifício (nomeando cada objecto com as suas informações) e o trabalho de aplicar materiais e a animação serão feitas de forma automática.

Descrição do exercício 1:

Depois de abrir o Blender, criou-se uma nova colecção de objectos com o nome "ModeloCasa" para onde se importou o modelo geométrico da casa através de `file>import>Collada (.dae)`. Posteriormente criaram-se alguns materiais que vão ser aplicados no modelo 3D (o nome destes materiais é importante uma vez que este vai ser utilizado no script). Para se criar um material através do "shader editor" carrega-se em slot `> + > new` e escolhe-se um nome para diferenciar cada um destes materiais. Posteriormente é possível alterar a cor de cada um destes materiais em "base color" para se poderem diferenciar no viewport.

Em seguida, criou-se um script de Python (Anexo 1) através do "text editor" do Blender, carregando em `+ new` e escolhendo um nome para o script.

Para começar o script, importou-se o módulo `bpy`, necessário ao funcionamento deste script. (Linha 3) Posteriormente, criou-se uma lista com todos os objectos da colecção "ModeloCasa", que contém o modelo 3D do projecto. (Linha 5)

Posteriormente, criou-se um método do tipo "for loop" que vai iterar todos os objectos presentes na colecção "ModeloCasa" e aplicar instruções a cada um destes. (Linha 7 até 94)

A primeira instrução é dividir o nome do objecto em palavras independentes separadas pelo caractere `"_"` (Linha 8). Depois do nome do objecto estar dividido em várias palavras, uma nova variável vai contar o número de palavras resultante dessa separação. (Linha 9) Se o número de palavras não for exactamente 6, o iterador ignora o objecto e passa ao próximo. (Linha 11 e 12)

Se o objecto tiver 6 palavras no nome, vão ser criadas 6 variáveis, uma para cada palavra do nome do objecto. (Linha 14)

Em seguida, é criada uma função `"animar()"` (Linha 16 até 24) que como o próprio nome indica, tem como objectivo animar o objecto. Para que esta função seja executada, vão ser inseridos 4 parâmetros (`objSelecioneado`, `eixoXYZ`, `valorDoEixo`, `frameDeInicio` e `frameDeFim`). Cada vez que esta função for chamada/utilizada, ela vai criar 3 "keyframes" que definem a posição do objecto nos 3 momentos da animação: posição do objecto antes da animação (Linha 20), posição do objecto no início da animação (Linha 22) e posição do objecto no fim da animação (Linha 24).

Posteriormente, foi criada uma função `"atribuirMaterial()"` (Linha 26 até 30) com o objectivo de atribuir texturas ao objecto. Para que esta função funcione, vai ser inserido 1 parâmetro (`materialDoObjecto`). Cada vez que esta função for chamada/usada, ela vai verificar se o objecto já tem algum material atribuído. Se já tiver um material atribuído, ela vai substituí-lo pelo material descrito no nome do objecto (Linha 27 e 28). Se não tiver um material atribuído, ela vai adicionar o material descrito no nome do objecto (Linha 29 e 30).

Por fim, o iterador vai distinguir cada objecto pelo elemento que vem caracterizado na primeira palavra do seu nome (por exemplo: viga, pilar, objectogenerico, porta, etc) através de uma condição do tipo "if, elif e else".

Por exemplo, se a primeira palavra do nome do objecto for "parede", vai ser utilizado o método animar() para criar uma animação de translação e vai ser atribuído material ao objecto através do método atribuirMaterial() (Linha 33, 34 e 35). Para que estes métodos funcionem correctamente, vão ser atribuídos os parâmetros (obtidos na Linha 14) que contêm as informações retiradas do nome do objecto (objecto, eixo, valorEixo, frameInicio, frameFim e material).

Se o nome do objecto não corresponder a nenhum dos casos identificados no script "objectogenerico, base, cobertura, escadas, janela, parede, pilar, portacorrer, porta, suportedabase e viga", o método "else" terminará a condição não executando alterações ao objecto.

Esta fluidez conseguida com a automatização de processos pode beneficiar imenso o trabalho na fase do projecto de execução.

O projecto de execução necessita da criação de muitos planos e esquemas que ajudem o construtor a perceber como executar as ideias do projectista. Por vezes, a leitura destes planos pode ser um pouco equívoca levando à necessidade de criar mais elementos gráficos que ajudem à perceção do construtor. Podemos servir-nos do exemplo de empresas que vendem produtos desmontados com o intuito de serem montados pelo cliente tais como por exemplo o IKEA ou a LEGO. Para quem já montou um produto destas empresas, provavelmente fê-lo com o auxílio de um manual de instruções visual com gráficos axonométricos que explicam o processo de montagem passo-a-passo. Para os arquitectos, executar uma abordagem semelhante para o seu projecto é pouco conveniente, uma vez que a criação de esquemas axonométricos ou até mesmo animações 3D é um processo demorado tendo em conta que o edifício possui diversas situações de pormenores construtivos específicas.

No entanto, abordando esta situação com recurso a métodos de automatização como por exemplo o script apresentado anteriormente, o projectista pode programar um "workflow" que realize estes esquemas de forma automática.

Notas:

Softwares utilizados: Blender 3D (versão 3.0.0)

Linguagem de programação utilizada: Python 2.7

Módulos de Python utilizados: bpy

Imagens do exercício 1:

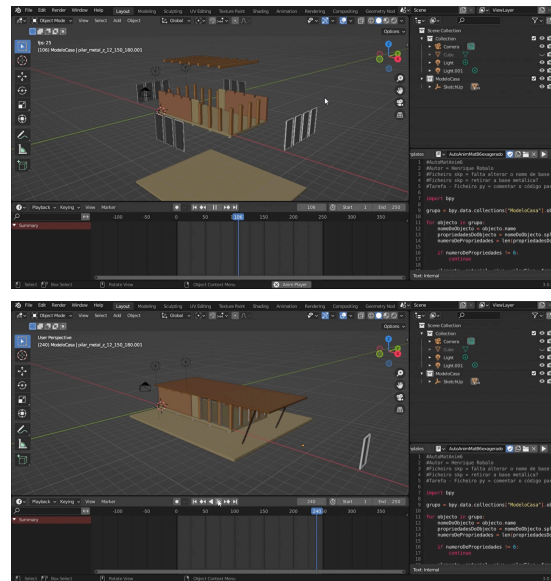
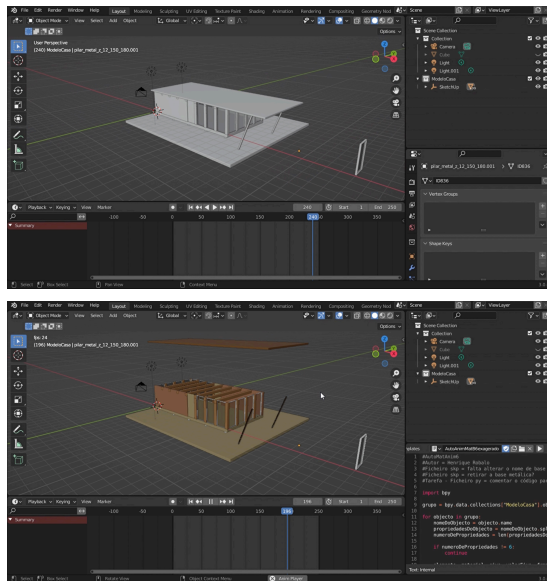


Fig 28 - Modelo da casa (Antes de executar o script) sem texturas nem animações, em branco

Fig 29 - Modelo da casa (Depois de executar o script) Frame 106 da animação, todos os objects são texturizados e animados automaticamente como o script

Fig 30 - Modelo da casa (Depois de executar o script) Frame 196 da animação, os elementos vão sendo adicionados à casa ao longo da animação

Fig 31 - Modelo da casa (Depois de executar o script) Frame 240 da animação, a casa está montada no final da animação.

5.2 - Exemplo 2 - Criação de ferramentas para realizar tarefas específicas

Contexto do exercício 2:

A intenção deste exercício é criar um modelo 3D da topografia do Concelho de Gondomar. Para conseguir este objectivo foi utilizada cartografia desenvolvida pela Câmara Municipal de Gondomar como fonte de informação topográfica. Este ficheiro de extensão (.dwg) pode ser aberto no Rhinoceros3D e tem diversa informação organizada por camadas (Layers), o que facilita a tarefa.

Modelar a topografia em 3D manualmente com base nas cotas altimétricas da cartografia não é uma solução eficiente tendo em conta que existem mais de 5000 valores de cotas distribuídos pela cartografia e, por esse motivo, modelar cada uma destas cotas iria demorar imenso tempo. Para realizar esta tarefa de forma automática pode ser utilizado um script. Este recolhe os valores altimétricos existentes num "Layer" da cartografia e organiza-os numa lista que pode ser utilizada para criar uma nuvem de pontos 3D com coordenadas x, y e z.

Descrição do exercício 2:

Depois de abrir o ficheiro da cartografia .dwg no Rhinoceros3D, iniciou-se o plugin GrassHopper (Tools > GrassHopper) e criou-se um ficheiro Grasshopper novo (File > New Document). No novo documento adicionou-se um componente de Python (Maths > Script > GH Python Script) e ao ampliar o "zoom" nesse componente é possível adicionar mais parâmetros de saída "output parameters" carregando nos botões "+" do lado direito do componente. Depois de adicionados mais dois parâmetros, os componentes de saída (output) são "out, a, b e c" do lado direito (é possível alterar o nome dos outputs carregando neles com o botão direito do rato). Para editar o script carrega-se duas vezes seguidas no componente e um editor de texto deverá aparecer.

O script Python utilizado para este exercício está disponível nos anexos (Anexo 2):

Na linha 11 do script, importou-se o módulo "rhinoscriptsyntax", necessário ao desenvolvimento do script.

Em seguida (nas linhas 16, 17 e 18), são criadas três listas vazias "a", "b" e "c" onde se vão adicionar posteriormente as coordenadas de cada ponto.

Posteriormente (desde a linha 21 até à linha 39), foi adicionado um iterador do tipo "for loop" que vai percorrer e identificar todos os objectos (objID) do Layer escolhido "03010202_TextoPontoCotado" e para cada um deles, o iterador vai:

Interpretar o objecto (Linha 24);

Identificar o valor da coordenada X do objecto (Linha 29) e adicioná-lo à lista "a" (Linha 31);

Identificar o valor da coordenada Y do objecto (Linha 33) e adicioná-lo à lista "b" (Linha 35);

Ler o valor do texto do objecto (Linha 37), convertê-lo num valor decimal do tipo "float" e adicioná-lo à lista "c" (Linha 39);

Depois de escrever o script, testa-se o resultado carregando no botão do canto superior direito do editor de texto para executar o script e fecha-se a janela do editor de texto. De cada vez que o script é executado, as listas a, b e c devem estar preenchidas com os valores recolhidos.

Para criar uma nuvem de pontos 3D no Grasshopper adiciona-se um componente "Construct Point" (Vector > Point > Construct Point) e conecta-se os parâmetros a, b e c do componente "GH Python Script" aos parâmetros x, y e z do componente "Construct Point" pela seguinte ordem (a -> x) (b -> y) (c -> z) e uma nuvem de pontos vermelha deverá aparecer no "Viewport" do Rhinoceros3D.

Para exportar os resultado do Grasshopper para o Rhinoceros3D, carrega-se no componente "Construct Point" com o botão direito do rato e posteriormente em "Bake..." e uma janela deverá aparecer para escolher o "Layer" de trabalho e carregar no "ok".

Depois destes pontos estarem no Rhinoceros 3D, exportou-se esta nuvem de pontos como um ficheiro de tipologia texto (só com os valores das coordenadas recolhidas) para o Archicad, onde foi convertido num modelo geométrico com polígonos e renderizado no Blender e no Cinema4D.

Requisitos:

Para este exemplo vão ser necessários 2 módulos (csv e rhinoscriptsyntax), o rhinoscript vem incluído com o programa rhinoceros e o csv está incluído no python 2.7. Para se verificar os módulos, é possível escrever/inserir o seguinte texto no pythonshell no terminal de comandos: help("modules")

Softwares utilizados: Rhinoceros 7 / Grasshopper

Linguagem de programação utilizada: Python

Módulos de Python utilizados: rhinoscriptsyntax / csv

Imagens do exercício 2:

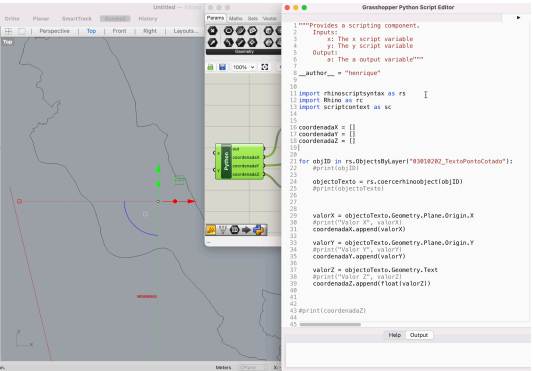


Fig 32 - Antes de Executar o Script

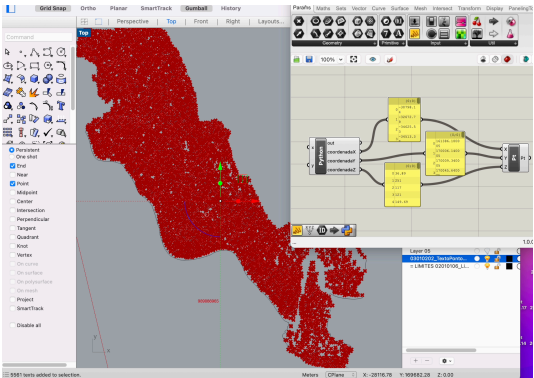


Fig 33 - Depois de executar o Script
(Nuvem de pontos vermelhos vista de cima)

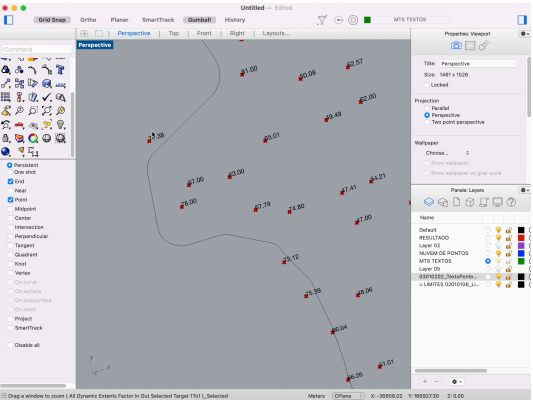


Fig 34 - Depois de executar o Script
(Nuvem de pontos vermelhos vista de cima
ampliado em Rio Tinto)

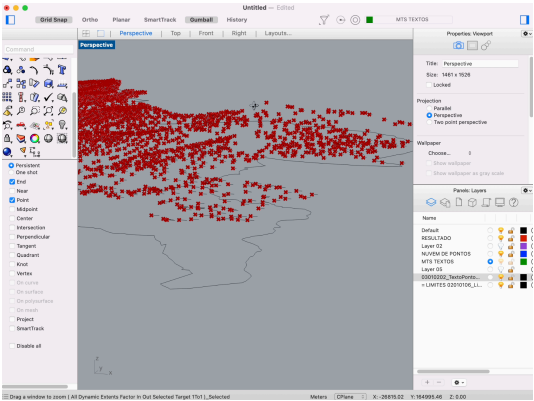


Fig 35 - Depois de executar o Script
(Nuvem de pontos vermelhos em perspectiva)



Fig 36 - Modelo Geométrico 3D realizado a
partir da nuvem de pontos

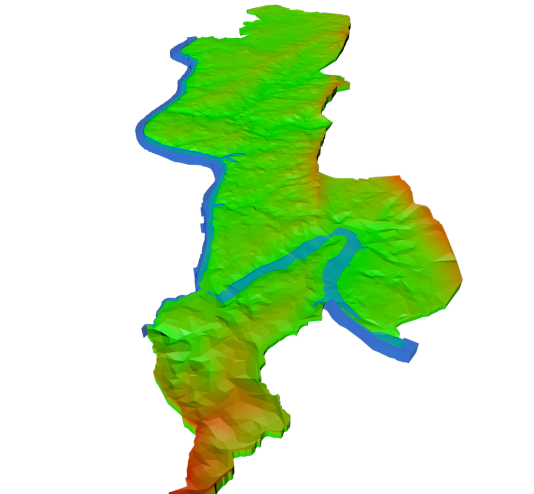


Fig 37 - Modelo Geométrico 3D realizado a
partir da nuvem de pontos colorido por altitude
(Vermelho: 500m / Verde: 0m / Azul: Rio Douro)

5.3 - Exemplo 3 - Utilização de programação para conectar/hiperligar diversas fontes de informação automaticamente

Contexto do exercício 3:

Como foi referido no módulo 3, as metodologias BIM e CAD diferem no tipo de informação que suportam, pois o BIM consegue armazenar diversas informações relativas a cada objecto do projecto, enquanto que o CAD armazena apenas informações relativas à geometria, texturas ou animação. Por esse motivo, ao converter um modelo BIM num modelo CAD, grande parte da informação/atributos dos objectos é perdida (como por exemplo os valores de resistência ao fogo, preço, ou fabricante). Para contornar esta limitação, este exercício propõe armazenar as informações do modelo BIM numa tabela independente que depois pode ser consultada pelo o modelo CAD.

Actualmente, os softwares BIM não oferecem formas eficientes para o arquitecto analisar os atributos do modelo de forma visual. Isto é uma desvantagem porque ter uma perceção visual interactiva dos custos de cada alteração, pode ajudar os arquitectos a modelar/projectar edifícios de forma mais eficiente. Neste contexto, o objectivo deste exercício é colorir todos os elementos do modelo geométrico de acordo com os seus atributos (resistência ao fogo ou o preço por m3 por exemplo) com a intenção de criar um modelo visual do preço do edifício ou da resistência ao fogo.

Descrição do exercício 3:

Para o propósito desta análise foi escolhido um modelo BIM de um escritório projectado para Budapeste na Hungria disponibilizado pelo website do archicad em: <https://community.graphisoft.com/t5/Let-s-get-started/Archicad-Sample-Projects/ta-p/304186>

Depois de aberto o modelo no Archicad, exportou-se um modelo geométrico em .dae para ser aberto no Rhinoceros 3D. Ainda no Archicad, foi utilizada a função Schedules para exportar uma tabela .xlsx com as seguintes informações (Element Type, Unique ID, Fire Resistance Rating, Price, cost per m3, total volume cost) de todos os elementos existentes no modelo tal como ilustrado na Fig 56. Esta tabela foi depois convertida num ficheiro CSV para ser lida pelo script Python.

O modelo geométrico (.dae) foi aberto no Rhinoceros 3D, onde a intenção era colorir cada objecto de acordo com o valor do seu preço ou a sua resistência ao fogo. Para tal, era necessário associar as informações da tabela .csv correspondente a cada elemento/objecto do modelo. Para realizar esta tarefa foram utilizados dicionários, um

conceito teórico para armazenar variáveis/valores no Python através de pares key[value]. O script deste exercício está disponível nos anexos (Anexo 3).

Para começar, é necessário importar os módulos utilizados neste exercício, rhinoscriptsyntax e csv. (Linhas 3 e 4)

Em seguida, foi criado uma variável do tipo string para armazenar a directoria/localização do ficheiro .csv no computador. (Linha 6)

Depois, foram criadas listas e dicionários que vão armazenar as informações que vão ser recolhidas da tabela. (Linhas 9 a 14)

Para preencher estas listas e dicionários foi criado um mecanismo para recolher as informações da tabela csv. (Linha 18 a 30)

Tendo as informações dos objectos, o passo seguinte é recolorir os objectos do modelo geométrico de acordo com os seus atributos. No caso deste modelo geométrico, cada objecto foi importado como uma instância de bloco. Conhecendo os mecanismos do Rhinoceros, não é possível editar/modificar as instâncias dos blocos em si, para tal, é preciso alterar os blocos. Para realizar esta tarefa foi criado um método do tipo for loop que vai iterar sobre todos os objectos presentes no modelo e vai aplicar as seguintes instruções a cada um deles.

A primeira instrução vai identificar o nome do bloco. (Linha 46)

Seguidamente, vão ser feitas algumas verificações não necessárias. (Linha 48 até 58)

Posteriormente, vai ser criado um "nested for loop" que vai iterar sobre todos os objectos que estão dentro do bloco iterado e, se a referência do bloco corresponder a alguma das referências da coluna "UniqueID" da tabela CSV, vai aplicar as seguintes instruções a cada um. (Linhas 61 até 73)

Em primeiro lugar, vai colorir todos os objectos de branco.

Em segundo lugar, vai colorir os objectos com o método definido nas linhas 33 a 41 que vai escolher a cor de acordo com o sistema RGB (Red Green Blue) em que o valor Red é influenciado pelo valor do atributo escolhido do objecto (Resistência ao fogo em minutos por exemplo), caso este não seja "NULL".

Notas:

Softwares utilizados: Archicad 25 / Rhinoceros 7

Linguagem de programação utilizada: Python 2.7

Módulos de Python utilizados: rhinoscriptsyntax / csv

Imagens do exercício 3:

Fig 38 - Modelo do edifício
Antes de executar o Script

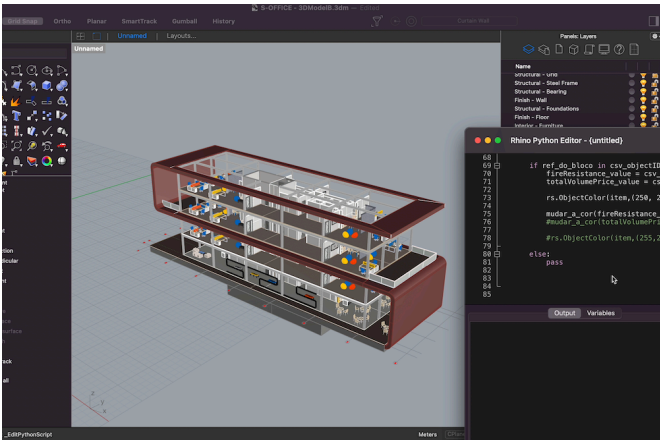


Fig 39 - Modelo do edifício depois de
executar o Script com os valores da
resistência ao fogo em minutos
Amarelo: maior valor
Verde: menor valor

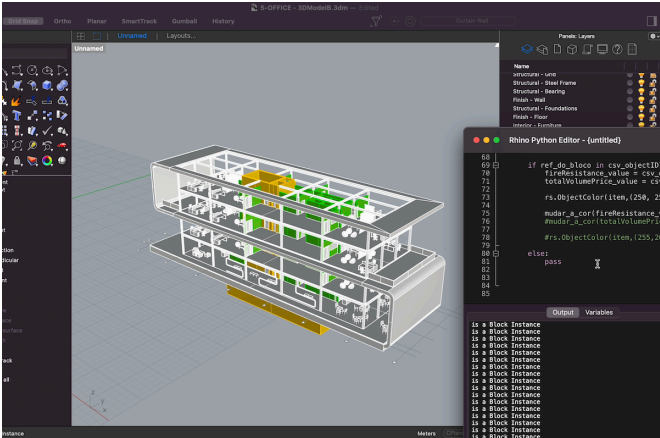


Fig 40 - Parte da tabela CSV com os
atributos de cada objecto

Element Type	Unique ID	Fire Resistance	Purchase Price	Cost Per Volume	Total Cost of Volume
Slab	341E1B01-FFF5-1A4B-A403-2008A1ED7B35	NULL	NULL	45.00	125
Wall	6FFD68C6-9220-C741-B1B6-4D2C25649695	90	NULL	45.00	162
Wall	08ECC1F7-F7A7-49CD-9A05-898FAA8FE13F	NULL	NULL	45.00	0.36
Wall	0BE70BD4-BA83-4ECB-BF28-47AC61E3399E	NULL	NULL	45.00	0.36
Wall	16053995-464B-41AB-A6F3-B35202F735E0	NULL	NULL	45.00	0.36
Wall	36660804-CAF6-4250-B19B-7D0A959CCDCD	NULL	NULL	45.00	0.36
Wall	548E36BF-2708-4C70-9B44-4BA102FF3602	NULL	NULL	45.00	0.36
Wall	5EE9D26B-8F4C-4F42-8665-13E578F20879	NULL	NULL	45.00	0.36
Wall	BDED210C-EDF0-4D2B-B99C-653073814DA5	NULL	NULL	45.00	0.36
Wall	ED516AF4-43B5-4397-8489-95540B76F56E	NULL	NULL	45.00	0.36
Wall	4B6BD0D1-EF3D-4FAF-9B4D-6FB6C15B207B	NULL	NULL	45.00	0.38
Wall	8D840F23-9C81-4D2C-9FBF-4D8C7FCE2254	NULL	NULL	45.00	0.39
Wall	BCE9E014-697E-4526-A8BF-A1E16F24E550	NULL	NULL	45.00	0.39
Wall	EC39FBFB-7E43-4187-9F9F-0156AB9D40C3	NULL	NULL	45.00	0.39
Wall	4AC4CA40-1774-4CE0-987B-A5EFEE39CB50	NULL	NULL	45.00	0.41
Wall	4C720E76-185B-4A86-AC8E-7FB6226B4EC7	NULL	NULL	45.00	0.41
Wall	A42577F5-7B7F-4BFD-8A32-CEB7E8590A16	NULL	NULL	45.00	0.41
Wall	F26D9CDB-0F41-4FB1-BA5D-68DEB4AF2F1D	NULL	NULL	45.00	0.41
Wall	63002348-2099-49F4-8508-2306C11410BD	NULL	NULL	45.00	0.43
Wall	79035BB6-CA50-416A-BBB0-71A461FCA0CC	NULL	NULL	45.00	0.44
Wall	06360F8B-1945-4420-8875-456AF26A90EC	NULL	NULL	45.00	0.45
Wall	19B3C632-2C7E-4E39-8286-553C1D970B78	NULL	NULL	45.00	0.45
Wall	1C123B63-D888-45B2-82A8-515D93F22784	NULL	NULL	45.00	0.45
Wall	64CDC022-AFF9-4E3C-98CE-A05512567F79	NULL	NULL	45.00	0.45
Wall	BE6BAE2D-00C0-4401-8219-A67714A4797D	NULL	NULL	45.00	0.45
Wall	D628436F-3959-420F-84F2-A5B45A418817	NULL	NULL	45.00	0.45
Wall	70653A88-A37E-4D77-BE18-756AEEAEB9B9	NULL	NULL	45.00	0.45

5.4 - Exemplo 4 - Utilização de Python para automatizar tarefas repetitivas.

Contexto do exercício 4:

Ao trabalhar em ficheiros com Layers como por exemplo DWG ou DXF, por vezes, existem Layers que não têm objectos e não são necessários ao trabalho. Quando o número de Layers é baixo, é uma tarefa relativamente fácil de descobrir e eliminar os ficheiros inutilizados, contudo se o número de Layers for maior (como por exemplo 60 Layers), esta tarefa pode-se tornar demorada.

A intenção deste exercício é automatizar esta tarefa repetitiva com a utilização de um script encontrado no site "<https://discourse.mcneel.com/t/the-best-way-to-identify-layers-without-objects-to-delete/7690/3>" publicado pelo utilizador Mitchell "Helvetosaur".

Descrição do exercício 4:

Depois de abrir um documento com vários Layers (dos quais alguns estão vazios) no Rhinoceros3D, iniciou-se o editor de texto de python (Tools > PythonScript > Edit) e inseriu-se o script que está disponível nos anexos, (Anexo 4):

No início do script, importou-se o módulo "rhinoscriptsyntax" (Linha 1), necessário ao desenvolvimento do script.

Posteriormente, é criada uma lista "layers" com o nome de todos os Layers do documento. (Linha 2)

Em seguida, foi adicionado um iterador do tipo "for loop" que vai percorrer todos os elementos da lista "layers" e em cada uma dessas linhas, o iterador vai usar uma condição do tipo "if" que vai testar se o layer estiver vazio e se tal acontecer, vai eliminá-lo. (Linhas 3 e 4)

Depois de escrever o script, testa-se o resultado carregando no botão do canto superior direito do editor de texto para executar o script e fecha-se a respectiva janela. De cada vez que o script é executado, os layers que não tiverem elementos/objectos devem desaparecer.

Notas:

Softwares utilizados: Rhinoceros 7

Linguagem de programação utilizada: Python 2.7

Módulos de Python necessários: rhinoscriptsyntax

Imagens do exercício 4:

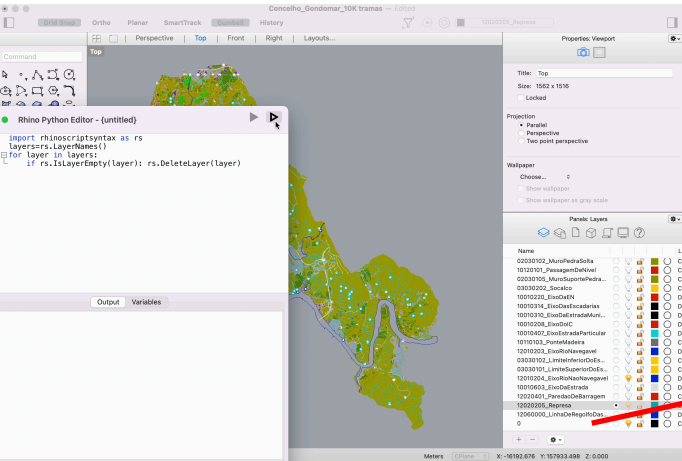


Fig 41 - Layers do modelo geométrico antes de executar o script

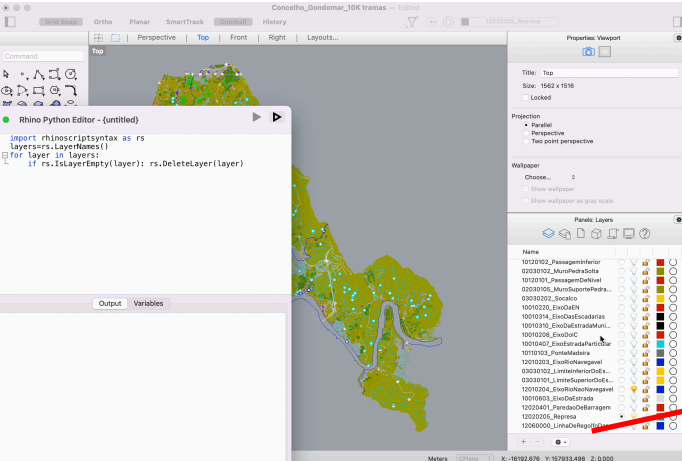
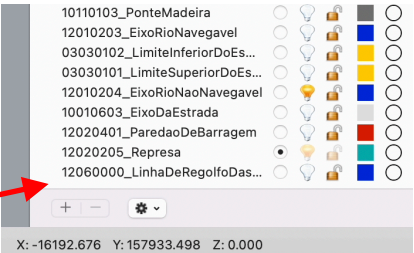


Fig 42 - Layers do modelo geométrico depois de executar o script. Todos os Layers que estavam vazios foram apagados (Como por exemplo o Layer "0")



5.5 - Exemplo 5 - Criação de modelos interactivos

Contexto do exercício 5:

O dinamismo introduzido pelos computadores permite a capacidade de simular o projecto através de modelos interactivos. Estes modelos podem ser difíceis de criar porque requerem programação de ações. Neste contexto, foram desenvolvidos alguns programas (como por exemplo o TwinMotion) que oferecem ferramentas intuitivas para criar modelos interactivos para utilizadores que não tenham conhecimentos de programação. A criação destes modelos é um processo extremamente complexo porque requer uma plataforma base 3D ou 2D onde serão programadas as ações mecânicas do modelo (um "Game Engine"). Estas plataformas são softwares complexos desenvolvidos por programadores experientes, no entanto existem algumas que estão disponíveis ao público (como o Unity Engine, o Unreal Engine por exemplo). Neste contexto, o conhecimento de programação permite ao projectista expressar melhor as suas ideias de forma interactiva e, ao mesmo tempo, simulá-las através de hardware como VR, AR, ou controladores convencionais. O objectivo deste exercício é demonstrar como o inventor pode utilizar um game engine para criar um modelo interactivo a partir do modelo geométrico ou do modelo BIM. Para tal, foi escolhido o Unity3D game engine⁷⁵ pela sua simplicidade, intuitividade e popularidade. Este programa oferece também uma versão gratuita que pode ser utilizada em diversos sistemas operativos. Alguns game engines (como o TwinMotion, o Unreal Engine e o Unity engine por exemplo) estão cada vez mais a proporcionar ferramentas e soluções que facilitem a integração com a área da arquitectura e promovem os seus produtos como uma nova ferramenta de trabalho para os arquitectos. "Architecture - Leap over the competition with instant visualizations and on-the-fly updates. - Visualize your project"^{76 77 78}

Estas novas tendências demonstram uma das formas de integrar programação no projecto, mas sobretudo, reforçam também a utilização de programação como uma vantagem/benefício para a arquitectura.

Para além disso, a programação aproxima a arquitectura de áreas como game development. Alguns estúdios de game development contratam modeladores e arquitectos para desenvolver modelos virtuais que serão utilizados em videojogos.

⁷⁵ <https://unity.com/>

⁷⁶ <https://unity.com/solutions/architecture-engineering-construction>

⁷⁷ <https://www.unrealengine.com/en-US/solutions/architecture>

⁷⁸ <https://www.twinmotion.com/en-US>

Descrição do exercício 5:

Dada a complexidade deste exercício, vão ser descritos apenas os passos mais importantes com o objectivo de manter a descrição resumida. Para compreender com mais detalhe alguns mecanismos é aconselhado consultar estes três tutoriais ^{79 80 81} e o API do Unity⁸². Ao contrário dos exercícios anteriores, este é desenvolvido com C# em vez de Python porque o Unity trabalha com C# nativamente.

Para realizar este exercício criou-se um novo projecto (com um template 3D) no Unity e este vai ter uma cena "scene" onde se desenvolvem as ações do modelo interactivo. Neste projecto Unity foram criados 2 empty GameObjects o "Utilizador" e a "Envolvente" que irão servir para agrupar diversos objectos hierarquicamente. O grupo "Utilizador" vai conter todos os GameObjects constituintes do jogador/ utilizador como câmeras e sensores de colisão (como demonstrado na figura seguinte à esquerda). O grupo "Envolvente" vai conter objectos relativos ao espaço onde o utilizador se desloca como o pavimento, escadas ou edifícios.

O GameObject "Utilizador" vai conter o componente "Character Controller" e o script C# para controlar/mover o "Character Controller" no espaço 3D.

A câmara da cena "Main Camera" vai conter o script C# para rodar o Utilizador e a Câmera.

Os objectos do grupo "Envolvente" como o chão, as paredes ou as portas vão ter detetores de colisão (mesh colliders por ex) para evitar que o jogador os atravesse como um fantasma.

Este projecto vai ter 3 scripts escritos em C#, "MoverUtilizador.cs" (Anexo 5), "RodarUtilizador.cs" (Anexo 6) e "ClickObjectos.cs" (Anexo 7). Os scripts deste exercício estão disponíveis nos anexos.

O script "MoverUtilizador.cs" tem como objectivo controlar a deslocação do jogador no espaço3D. Este script começa por importar os módulos necessários (Linhas 3, 4 e 5). Posteriormente vai ser criada a classe MoverUtilizador (herdada da classe MonoBehaviour) (Linhas 7 a 57). Dentro desta classe vão ser estabelecidas algumas variáveis para definir parâmetros (como por ex o valor da gravidade ou a velocidade do movimento) (Linhas 9 a 21). Posteriormente, vai ser definida a função Update() que é utilizada em cada frame do jogo (Linha 25 a 56). Dentro desta função vão ser criados: mecanismos para determinar a colisão do utilizador com os objectos;

⁷⁹ https://www.youtube.com/watch?v=_QajrabyTJc

⁸⁰ <https://www.youtube.com/watch?v=EANtTi6BCxk>

⁸¹ <https://www.youtube.com/watch?v=FGiCeJM743g>

⁸² <https://docs.unity3d.com/ScriptReference/Transform.html>

mecanismo para simular o efeito da gravidade e mecanismos para controlar o movimento do utilizador no espaço 3D do modelo através de inputs de hardware (Teclado ou controladores com JoyStick).

O script "RodarUtilizador.cs" tem como objectivo controlar a rotação do jogador e da câmara no eixo vertical e horizontal respectivamente. Este script começa por importar os módulos necessários (Linhas 3, 4 e 5). Posteriormente vai ser criada a classe MoverUtilizador (Linhas 7 a 34). Dentro desta classe vão ser estabelecidas algumas variáveis para definir parâmetros (como por ex o valor da sensibilidade do rato) (Linhas 9 a 12). Em seguida, vai ser definida a função Start() que é utilizada apenas uma vez no início da execução do script. Nesta função definiu-se o estado do cursor do rato para estar fixo no centro do display durante a execução do modelo interactivo.

Posteriormente, vai ser definida a função Update() (Linha 25 a 56) onde vão ser criados: mecanismo para rodar a "Main Camera" no eixo horizontal e mecanismos para controlar a rotação do utilizador no espaço 3D do modelo através de inputs de hardware (Rato ou controladores com JoyStick).

O script "ClickObjectos.cs" tem como objectivo aplicar ações (por ex abrir portas) aos objectos em que o utilizador carregue com o cursor. Este script começa por importar os módulos necessários (Linhas 4, 5 e 6). Seguidamente, vai ser criada a classe ClickObjectos (Linhas 8 a 75). Dentro desta classe vão ser estabelecidas algumas variáveis para definir parâmetros (como por ex o ângulo da porta aberta) (Linhas 10 a 15). Depois, vai ser definida a função Start() onde se define o ângulo inicial do objecto como o ângulo da porta fechada.

Posteriormente, vai ser definida a função Update() (Linha 26 a 49) onde vão ser criados mecanismos para determinar que objecto que está na mira do cursor no momento em que o utilizador clica. Depois de clicar o objecto, o script vai aproveitar a convenção para nomear os objectos criada no exercício 1 (M5.1) ("parede=madeira=z=6=30=60" por ex) e se as primeiras 5 letras do nome corresponderem a "porta", o script vai aplicar uma função que faz com que este objecto rode 90° no eixo vertical do seu ponto pivotal.

Na cena deste projecto unity foram também adicionados objectos de som para experimentar e conhecer um pouco as capacidades do sistema de audio do Unity.

Ao editar o modelo interactivo, o projectista pode ir testando o jogo ao clicar no botão "play" por cima da janela da "scene". Quando o projectista estiver satisfeito com o resultado final, pode exportar um ficheiro executável deste modelo interactivo para diversas plataformas e sistemas operativos (Linux, MacOS, Windows, Android, IOS, entre outros). Este ficheiro exportado funcionará como uma aplicação/jogo normal.

Ao abrir este executável, aparece uma janela nova no sistema operativo onde o utilizador pode jogar e experimentar o modelo interactivo/projecto de arquitectura.

Potencialidades/Possibilidades do modelo interactivo:

Os game engines como o Unity ou o Unreal suportam diversos componentes como audio, iluminação ou fumo. Com estas capacidades, o projectista pode simular diversas condições no edifício/projecto.

O projectista pode simular em tempo real os efeitos/intensidade/sombras da iluminação no espaço ao activar, desactivar ou mover as fontes de luz pelas divisões do edifício.

O projectista pode animar carros numa via rodoviária e aplicar-lhes som para simular em tempo real os efeitos sonoros/condições acústicas do edifício à medida que o utilizador/habitante se desloca para diversas posições/divisões neste espaço ou quando abre e fecha janelas, por exemplo. O unity permite simular a intensidade do som ou o efeito Doppler à medida que estes carros se aproximam e afastam do edifício e do utilizador.

O projectista pode também simular a visibilidade das divisões de um edifício em caso de fumo num incêndio e perceber de que forma a má visibilidade pode conduzir os habitantes a desorientarem-se na planta ao tentarem evacuar o edifício.

Notas do exercício 5:

Softwares utilizados: Unity3D (Versão: 2021.3.3f1)

Linguagens de programação utilizadas: C#

Módulos de C# utilizados: Unity / System

Imagens do exercício 5:

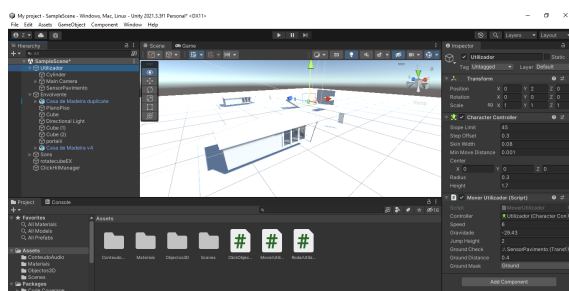
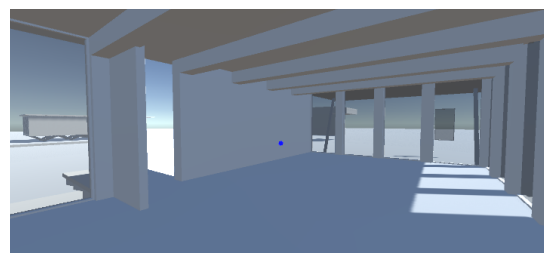
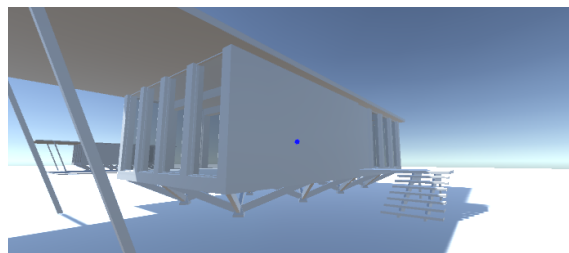


Fig 43 - Editor do Unity3D

Fig 44 - Modelo Interactivo da casa criada no ex1 (O utilizador pode percorrer o espaço 3D na primeira pessoa)

Fig 45 - Modelo Interactivo da casa criada no ex1 no seu interior (Todas as portas abrem se o utilizador clicar nelas com o cursor)



Módulo 6

Conclusão

6.1 - Observações realizadas em relação às metodologias de trabalho existentes

É possível admitir que o desenvolvimento do projecto de arquitectura não deriva de uma só metodologia de trabalho, mas sim da articulação entre as várias metodologias e instrumentos que estão ao dispor do projectista. Todas as metodologias de trabalho devem ser utilizadas desde que contribuam positivamente para a eficiência do desenvolvimento do projecto e o utilizador/arquitecto deve utilizar as ferramentas que permitam mais facilmente executar o projecto.

Um dos problemas que o arquitecto encontra ao desenvolver o seu trabalho, é que muitas vezes a ferramenta/software que utiliza não tem a capacidade de realizar a tarefa pretendida. Uma das opções utilizadas para contornar este obstáculo é alternar entre softwares que consigam resolver esse problema mas, por vezes, esta opção pode ser resolvida com a utilização de programação (visual ou escrita).

A utilização das metodologias CAD e BIM são actualmente as mais utilizadas na criação de projectos, no entanto os modelos resultantes destas metodologias são estáticos e, por esse motivo, todas as alterações significativas terão que ser realizadas manualmente.

As metodologias AAD e AABIM são neste contexto a forma mais intuitiva de associar programação ao projecto e prometem um novo paradigma de trabalho dinâmico e inovador. No entanto, a metodologia AABIM ainda se encontra num estado muito embrionário porque só pode ser acedida através de plugins em softwares BIM, tais como o ArchicadLiveConnection ou o RhinoInsideRevit. Esta articulação de AAD e BIM através de plugins não é a mais eficiente e a solução ideal para o AABIM seria um software que integrasse estas duas capacidades (BIM e AAD) nativamente permitindo inputs paramétricos e manuais simultaneamente.

6.2 - Aplicação da programação no desenvolvimento do projecto

As linguagens de programação são, no entanto muito versáteis e flexíveis, oferecendo um maior nível de controlo sobre os sistemas digitais. Os exemplos desenvolvidos no decorrer deste trabalho demonstram algumas utilizações práticas da programação no desenvolvimento do projecto, mas permitem sobretudo ter uma visão das inúmeras possibilidades que surgem em associar estes dois conhecimentos. No entanto, convém ter em conta alguns limites na utilização da programação nos softwares, pois nem toda as plataformas oferecem API's.

No século XXI, começaram a ser utilizadas tecnologias para a visualização do modelo tais como AugmentedReality, VirtualReality ou Game Engines, como por exemplo o TwinMotion, que oferece ferramentas intuitivas à criação de um modelo 3D interactivo. Neste contexto, a programação pode ser utilizada para expandir as capacidades de expressão do arquitecto porque permite a criação de modelos interactivos que o utilizador pode simular e experimentar em tempo real tal como demonstrado no exemplo 4.

Com o crescimento da utilização dos modelos interactivos, o domínio de programação num game engine (como por exemplo o Unity3D ou Unreal Engine) pode servir o arquitecto, oferecendo inúmeras formas de expressar o projecto. Caso o arquitecto não tenha capacidade de programar, as capacidades deste ficam limitadas a ferramentas disponibilizadas por softwares mais intuitivos como o Twinmotion.

As capacidades de programar aproximam os arquitectos à indústria dos jogos virtuais abrindo também novas possibilidades de emprego. A programação permite também aproximar a área da arquitectura à área da robótica, embora esta relação necessite de muitos mais conhecimentos.

O conhecimento de programação não é necessário ao desenvolvimento de projectos e, por esse motivo, não precisa de ser integrado no currículo de aprendizagem dos arquitectos, no entanto o domínio deste conhecimento equipa o arquitecto com novas capacidades de expressão, criatividade e eficiência.

6.3 - Como desenvolver este estudo no futuro

O crescente desenvolvimento dos sistemas digitais fornece novidades tecnológicas que poderão promover a relação entre a arquitectura e a programação. Caso os softwares de arquitectura decidam continuar a desenvolver as metodologias AAD, AABIM e as capacidades de Programação Escrita, talvez as relações entre a programação e a arquitectura se tornem ainda mais evidentes.

Uma das opções para desenvolver este trabalho é encontrar novas formas de integrar programação (escrita ou visual) no desenvolvimento do projecto através de exercícios semelhantes aos anteriormente descritos. No decorrer deste trabalho surgiram algumas ideias que acabaram por não ser postas em prática por falta de tempo, por dificuldades encontradas nos softwares ou porque o nível de complexidade é demasiado para o tempo disponível.

Uma das ideias era criar um mecanismo que permitisse ao arquitecto exportar os modelos desenvolvidos num software para um sistema de armazenamento online com um script. Uma sugestão era utilizar o API da Google Drive ou outro sistema semelhante para exportar os modelos automaticamente a partir do software CAD.

Uma outra ideia era interligar (Hyperlink) diversos softwares para desenvolver uma "cascata de softwares". Nesta cascata de softwares, cada vez que o projectista fizesse alterações a um modelo geométrico e salvasse as alterações, este modelo seria exportado automaticamente para outros programas (como o Blender ou o Unity por exemplo) onde seriam automaticamente realizadas diversas tarefas como renderings ou análises (Esta ideia assemelha-se ao modelo/trabalho realizado por Gkioka em 2018).

Uma outra ideia seria criar uma aplicação para telemóvel ou tablet e, a partir destes equipamentos, o utilizador poderia inserir inputs no software para escolher as ferramentas de forma semelhante a um macro pad. Para tal, iria ser utilizado um IDE como o XCode com a linguagem de programação Swift para desenvolver uma aplicação.

Uma outra hipótese seria tentar descobrir como é possível relacionar a arquitectura com a robótica de forma vantajosa para o desenvolvimento do projecto ou para a comunicação/apresentação do mesmo.

Bibliografia

AISH, Robert; MENDOZA, Emmanuel - DesignScript: a domain specific language for architectural computing. Em Proceedings of the International Workshop on Domain-Specific Modeling [Em linha]. Amsterdam Netherlands : ACM, 30 out. 2016 [Consult. 20 nov. 2021]. Disponível em WWW:<URL:<https://dl.acm.org/doi/10.1145/3023147.3023150>>. ISBN 978-1-4503-4894-2

BERGIN, Michael - History of BIM. Architecture Research Lab. 2011. 6.

BISSYANDE, Tegawende F. et al. - Popularity, Interoperability, and Impact of Programming Languages in 100,000 Open Source Projects. Em 2013 IEEE 37th Annual Computer Software and Applications Conference [Em linha]. Kyoto, Japan : IEEE, jul. 2013 [Consult. 9 abr. 2022]. Disponível em WWW:<URL:<https://ieeexplore.ieee.org/document/6649842>>. ISBN 978-0-7695-4986-6

COENDERS, Jeroen L. - Next Generation Parametric Design. Journal of the International Association for Shell and Spatial Structures. . ISSN 1028-365X. 62:2 (2021) 153–166. doi: 10.20898/j.iass.2021.018.

CONLAN, Chris - The Blender Python API [Em linha]. Berkeley, CA : Apress, 2017 [Consult. 9 abr. 2022]. Disponível em WWW:<URL:<http://link.springer.com/10.1007/978-1-4842-2802-9>>. ISBN 978-1-4842-2801-2.

CVETKOVIĆ, Ivana et al. - Historical Development of Computer-Aided Design (CAD) and Its Application in Education at Faculty of Mechanical Engineering in Belgrade. Em Proceedings of the International Scientific Conference - Sinteza 2019 [Em linha]. Novi Sad, Serbia : Singidunum University, 2019 [Consult. 9 abr. 2022]. Disponível em WWW:<URL:<http://portal.sinteza.singidunum.ac.rs/paper/644>>. ISBN 978-86-7912-703-7

DANHAIVE, Renaud Alexis; MUELLER, Caitlin T. - Combining parametric modeling and interactive optimization for high-performance and creative structural design. 2015. 11.

EASTMAN, Charles M. et al. (EDS.) - BIM handbook: a guide to building information modeling for owners, managers, designers, engineers and contractors. 2nd ed. Hoboken, NJ : Wiley, 2011. ISBN 978-0-470-54137-1.

FAROOQ, Muhammad Shoaib et al. - An Evaluation Framework and Comparative Analysis of the Widely Used First Programming Languages. PLoS ONE. ISSN 1932-6203. 9:2 (2014) e88941. doi: 10.1371/journal.pone.0088941.

FERREIRA, Luís - Habitáculo reversível. Ligações através do uso de ferramentas digitais. Braga : Universidade do Minho, mar. 2019

GKIOKA, Maria et al. - Novel Parametric Knowledge Modelling approach applied to Viaducts. 2019.

GKIOKA, Maria - Knowledge Model for the Parametric Design of Concrete Viaducts. 2018.

HUMPPI, Harri; ÖSTERLUND, Toni - Algorithm-Aided BIM. 2015.

IASBIK, Marina Pires; MARTINEZ, Andressa Carmo Pena; GAZEL, Jorge Lira De Toledo - Integration of BIM and Algorithmic Design logics through data exchange between Grasshopper plugin and Revit and Archicad software. Em Blucher Design Proceedings [Em linha]. Medellín, Colombia : Editora Blucher, dez. 2020 [Consult. 9 abr. 2022]. Disponível em WWW:<URL:<http://www.proceedings.blucher.com.br/article-details/35473>>.

KILKELLY, Michael James - Object-Oriented Representations. Massachusetts Institute of Technology (MIT). 1996

KOCHHAR, Pavneet Singh; WIJEDASA, Dinusha; LO, David - A Large Scale Study of Multiple Programming Languages and Code Quality. Em 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER) [Em linha]. Suíça : IEEE, mar. 2016 [Consult. 9 abr. 2022]. Disponível em WWW:<URL:<http://ieeexplore.ieee.org/document/7476675/>>. ISBN 978-1-5090-1855-0

KU, Kihong - CAD/CAM-MEDIATED DESIGN AND CONSTRUCTION: A CASE STUDY OF THE BANKSIDE LOFT PROJECT. Second International Conference World of Construction Project Management 2007. 2007. 11.

LOWE, Russell - COMPUTER GAME MODDING FOR ARCHITECTURE. [s.d.] 10.

MAGALHÃES, José - Hardware Tecnologias e Soluções. FCA-Editora de Informática Lda ed. Lisboa : Lidel - Edições Técnicas Lda, 2019. ISBN 978-972-722-892-8.

MARTINS, João Poças - LICENCIAMENTO AUTOMÁTICO DE PROJECTOS – UMA SOLUÇÃO PARA UM PROBLEMA DE COOPERAÇÃO? [s.d.] 10.

MONDAL, Joy - Chingree: Grasshopper Plugin to Generate 2D Compositions and 3D Embodiments Inspired from Mondrian's Neo-plastic Compositions Using Shape Grammar. Em Blucher Design Proceedings [Em linha]. Medellín, Colombia : Editora Blucher, dez. 2020 [Consult. 9 abr. 2022]. Disponível em WWW:<URL:<http://www.proceedings.blucher.com.br/article-details/35440>>.

MONTEIRO, André; MARTINS, João Poças - Licenciamento automático de projectos: um incentivo à adopção de metodologias BIM. 2011 10.

NEUFERT, Ernst; FRANCO, Benelisa - Arte de projetar em arquitectura. 18. ed ed. São Paulo : Gustavo Gili, 2013. ISBN 978-85-65985-08-6.

OZGUR, Ceyhun et al. - MatLab vs. Python vs. R. [s.d.] 18.

Python textbook - readthedocs-io-en-1.0 - [Em linha]. Cape Town : University of Capetown, 2017 Disponível em WWW:<URL:https://python-textbok.readthedocs.io/_/downloads/en/1.0/pdf/>.

SHARAIDIN, Kamil; BURRY, Jane; SALIM, Flora - Integration of Digital Simulation Tools With Parametric Designs to Evaluate Kinetic Façades for Daylight Performance. [s.d.] 10.

SHEIL, Bob et al. (EDS.) - Design Transactions: Rethinking Information Modelling for a New Material Age [Em linha]. [S.l.] : UCL Press, 2020 [Consult. 9 abr. 2022]. Disponível em WWW:<URL:<http://www.jstor.org/stable/10.2307/j.ctv13xprf6>>. ISBN 978-1-78735-502-6.

SIEGFRIED, Robert M.; SIEGFRIED, Jason P.; ALEXANDRO, Gina - A Longitudinal Analysis of the Reid List of First Programming Languages. 2016.

SOBRAL, Sonia - CS1: C, JAVA OR PYTHON? TIPS FOR A CONSCIOUS CHOICE. Em [Em linha]. Seville, Spain : [s.n.] [Consult. 9 abr. 2022]. Disponível em WWW:<URL:<http://library.iated.org/view/SOBRAL2019CS1>>.

SOBRAL, Sónia Rolland - The First Programming Language and Freshman Year in Computer Science: Characterization and Tips for Better Decision Making. Em ROCHA, ÁLVARO et al. (Eds.) - Trends and Innovations in Information Systems and TechnologiesAdvances in Intelligent Systems and Computing. [Em linha]. Cham : Springer International Publishing, 2020 [Consult. 9 abr. 2022]. Disponível em WWW:<URL:http://link.springer.com/10.1007/978-3-030-45697-9_16>. ISBN 978-3-030-45696-2v. 1161. p. 162–174.

STREHLKE, Kai - Digital Technologies, Methods, and Tools in Support of the Architectural Development at Herzog & de Meuron. ACADIA. 2009.

SYAM, Husain et al. - Turkish Journal of Computer and Mathematics Education Vol.12 No. 7 (2021), 1207- 1216 Research Article. 2021.

VALVERDE, Sergi; SOLÉ, Ricard V. - Punctuated equilibrium in the large-scale evolution of programming languages <sup/>. Journal of The Royal Society Interface. . ISSN 1742-5689, 1742-5662. 12:107 (2015) 20150249. doi: 10.1098/rsif.2015.0249.

VASCONCELOS, José - Python. Algoritmia E Programação Web. FCA-Editora de Informática ed. Lisboa : Lidel - Edições Técnicas Lda, 2015. ISBN 978-972-722-813-3.

WEISBERG, David E. - A Brief Overview of the History of CAD. Em The Engineering Design Revolution [Em linha] Disponível em WWW:<URL:www.cadhistory.net>. 2008

WEISBERG, David E. - Patrick Hanratty and Manufacturing & Consulting Services. Em The Engineering Design Revolution [Em linha] Disponível em WWW:<URL:www.cadhistory.net>. 2008

WEISBERG, David E. - The Engineering Design Revolution [Em linha] Disponível em WWW:<URL:www.cadhistory.net>. 2008

Referências

Gráficas

Fig 1 - Kilkelly Msc thesis p19 - KILKELLY, Michael James - Object-Oriented Representations. Massachusetts Institute of Technology (MIT). 1996 p19

Fig 2 - BERGIN, Michael - History of BIM. Architecture Research Lab. 2011) 6. p1

Fig 3 - MAGALHÃES, José - Hardware Tecnologias e Soluções. FCA-Editora de Informática Lda ed. Lisboa : Lidel - Edições Técnicas Lda, 2019. ISBN 978-972-722-892-8. p7

Fig 4 - Apple Macintosh - <https://www.bukowskis.com/fi/lots/843598-dator-macintosh-plus-1mb-apple-med-tillbehor-sent-1980-tal>

Fig 5 - KILKELLY, Michael James - Object-Oriented Representations. Massachusetts Institute of Technology (MIT). 1996 p25

Fig 6 - BERGIN, Michael - History of BIM. Architecture Research Lab. 2011) 6. p3

Fig 7 - BERGIN, Michael - History of BIM. Architecture Research Lab. 2011) 6. p4

Fig 8 - <https://www.cleanpng.com/png-industry-foundation-classes-building-information-m-6784523/>

Fig 9 - HUMPPPI, Harri; ÖSTERLUND, Toni - Algorithm-Aided BIM. 2015. p604

Fig 10 - HUMPPPI, Harri; ÖSTERLUND, Toni - Algorithm-Aided BIM. 2015. p607

Fig 11 - Imagem criada para este trabalho com base em alguns gráficos encontrados

Fig 12 - Python textbook - readthedocs-io-en-1.0 - [Em linha]. Cape Town : University of Capetown, 2017 Disponível em WWW:<URL:https://python-textbok.readthedocs.io/_/downloads/en/1.0/pdf/>. p8

Fig 13 - Python textbook - readthedocs-io-en-1.0 - [Em linha]. Cape Town : University of Capetown, 2017 Disponível em WWW:<URL:https://python-textbok.readthedocs.io/_/downloads/en/1.0/pdf/>. p8

Fig 14 - <https://www.nasa.gov/feature/margaret-hamilton-apollo-software-engineer-awarded-presidential-medal-of-freedom>
https://www.nasa.gov/sites/default/files/thumbnails/image/margaret_hamilton5.jpg

Fig 15 - Python textbook - readthedocs-io-en-1.0 - [Em linha]. Cape Town : University of Capetown, 2017 Disponível em WWW:<URL:https://python-textbok.readthedocs.io/_/downloads/en/1.0/pdf/>. p9

Fig 16 - FAROOQ, Muhammad Shoaib et al. - An Evaluation Framework and Comparative Analysis of the Widely Used First Programming Languages. PLoS ONE. . ISSN 1932-6203. 9:2 (2014) e88941. doi: 10.1371/journal.pone.0088941. p2

Fig 17 - KU, Kihong - CAD/CAM-MEDIATED DESIGN AND CONSTRUCTION: A CASE STUDY OF THE BANKSIDE LOFT PROJECT. Second International Conference World of Construction Project Management 2007. 2007) 11. p4

Fig 18 - KU, Kihong - CAD/CAM-MEDIATED DESIGN AND CONSTRUCTION: A CASE STUDY OF THE BANKSIDE LOFT PROJECT. Second International Conference World of Construction Project Management 2007. 2007) 11. p5

Fig 19 - <https://www.uniarts.fi/en/locations/helsinki-music-centre/>

Fig 20 - EASTMAN, Charles M. et al. (EDS.) - BIM handbook: a guide to building information modeling for owners, managers, designers, engineers and contractors. 2nd ed ed. Hoboken, NJ : Wiley, 2011. ISBN 978-0-470-54137-1. p552

Fig 21 - <http://www.grandtourmagazine.com/magic-brick-new-tate-modern-extension/>

Fig 22 - <https://www.herzogdemeuron.com/index/projects/complete-works/226-250/240-ciudad-del-flamenco/image.html>

Fig 23 - IASBIK, Marina Pires; MARTINEZ, Andressa Carmo Pena; GAZEL, Jorge Lira De Toledo - Integration of BIM and Algorithmic Design logics through data exchange

between Grasshopper plugin and Revit and Archicad software. Em Blucher Design Proceedings [Em linha]. Medellín, Colombia : Editora Blucher, dez. 2020 [Consult. 9 abr. 2022]. Disponível em WWW:<URL:<http://www.proceedings.blucher.com.br/article-details/35473>>. p472

Fig 24 - IASBIK, Marina Pires; MARTINEZ, Andressa Carmo Pena; GAZEL, Jorge Lira De Toledo - Integration of BIM and Algorithmic Design logics through data exchange between Grasshopper plugin and Revit and Archicad software. Em Blucher Design Proceedings [Em linha]. Medellín, Colombia : Editora Blucher, dez. 2020 [Consult. 9 abr. 2022]. Disponível em WWW:<URL:<http://www.proceedings.blucher.com.br/article-details/35473>>. p474

Fig 25 - IASBIK, Marina Pires; MARTINEZ, Andressa Carmo Pena; GAZEL, Jorge Lira De Toledo - Integration of BIM and Algorithmic Design logics through data exchange between Grasshopper plugin and Revit and Archicad software. Em Blucher Design Proceedings [Em linha]. Medellín, Colombia : Editora Blucher, dez. 2020 [Consult. 9 abr. 2022]. Disponível em WWW:<URL:<http://www.proceedings.blucher.com.br/article-details/35473>>. p474

Fig 26 - GKIOKA, Maria - Knowledge Model for the Parametric Design of Concrete Viaducts. 2018. p1110

Fig 27 - GKIOKA, Maria - Knowledge Model for the Parametric Design of Concrete Viaducts. 2018. p1111

Fig 28 - Imagem criada para este trabalho

Fig 29 - Imagem criada para este trabalho

Fig 30 - Imagem criada para este trabalho

Fig 31 - Imagem criada para este trabalho

Fig 32 - Imagem criada para este trabalho

Fig 33 - Imagem criada para este trabalho

Fig 34 - Imagem criada para este trabalho

Fig 35 - Imagem criada para este trabalho

Fig 36 - Imagem criada para este trabalho

Fig 37 - Imagem criada para este trabalho

Fig 38 - Imagem criada para este trabalho

Fig 39 - Imagem criada para este trabalho

Fig 40 - Imagem criada para este trabalho

Fig 41 - Imagem criada para este trabalho

Fig 42 - Imagem criada para este trabalho

Fig 43 - Imagem criada para este trabalho

Fig 44 - Imagem criada para este trabalho

Fig 45 - Imagem criada para este trabalho

Anexos:

Scripts utilizados no Módulo 5:

Anexo 1 - Script Python Utilizado no Módulo 5.1:

```
1      #Autor = Henrique Robalo
2
3      import bpy #importar o módulo do Blender
4
5      listaDeObjectos = bpy.data.collections["ModeloCasa"].objects
6
7      for objecto in listaDeObjectos:
8          propriedadesDoObjecto = objecto.name.split("_")
9          numeroDePropriedades = len(propriedadesDoObjecto)
10
11         if numeroDePropriedades != 6:
12             continue
13
14         elemento, material, eixo, valorEixo, frameInicio, frameFim =
            objecto.name.split("_")
15
16         def animar(objSelecionado, eixoXYZ, valorDoEixo, frameDeInicio, frameDeFim):
17             posicaoInicial = getattr(objSelecionado.location, eixoXYZ)
18
19             delayAnimacao = 0 #Em objectos repetidos como pilares existe um delay
                diferente na animacao de cada um deles
20             #Esta condicao cria um delay na animacao com base no nome dos objectos
21             #nomes do tipo: pilar_madeira_z_6_30_60.001 /
                pilar_madeira_z_6_30_60.002 / pilar_madeira_z_6_30_60.003
22
23             if "." in frameDeFim:
24                 delayAnimacao = frameDeFim.split(".")[1].lstrip("0")
25             else:
26                 pass
27
28             setattr(objSelecionado.location, eixoXYZ, posicaoInicial +
                int(valorDoEixo))
29             objSelecionado.keyframe_insert(data_path = "location", frame = 1)
30             setattr(objSelecionado.location, eixoXYZ, posicaoInicial +
                int(valorDoEixo))
31             objSelecionado.keyframe_insert(data_path = "location", frame =
                int(frameDeInicio))
32             setattr(objSelecionado.location, eixoXYZ, posicaoInicial)
33             objSelecionado.keyframe_insert(data_path = "location", frame =
                int(float(frameDeFim) + int(float(delayAnimacao))))
34
35
36         def atribuirMaterial(materialEscolhido = "material"):
37
38             if objecto.data.materials:
39                 objecto.data.materials[0] =
                bpy.data.materials.get(materialEscolhido)
40             else:
41                 objecto.data.materials.append(bpy.data.materials.get(materialEs
                colhido))
42             print(materialEscolhido)
43
44
45         if elemento == "objectogenerico":
46             animar(objecto, eixo, valorEixo, frameInicio, frameFim)
47             atribuirMaterial(material)
```



```
48
49
50     elif elemento == "base":
51         animar(objecto, eixo, valorEixo, frameInicio, frameFim)
52         atribuirMaterial(material)
53
54
55     elif elemento == "cobertura":
56         animar(objecto, eixo, valorEixo, frameInicio, frameFim)
57         atribuirMaterial(material)
58
59
60     elif elemento == "escadas":
61         animar(objecto, eixo, valorEixo, frameInicio, frameFim)
62         atribuirMaterial(material)
63
64
65     elif elemento == "janela":
66         animar(objecto, eixo, valorEixo, frameInicio, frameFim)
67         atribuirMaterial(material)
68
69
70     elif elemento == "parede":
71         animar(objecto, eixo, valorEixo, frameInicio, frameFim)
72         atribuirMaterial(material)
73
74
75     elif elemento == "pilar":
76         animar(objecto, eixo, valorEixo, frameInicio, frameFim)
77         atribuirMaterial(material)
78
79
80     elif elemento == "portacorrer":
81         animar(objecto, eixo, valorEixo, frameInicio, frameFim)
82         atribuirMaterial(material)
83
84
85     elif elemento == "porta":
86         animar(objecto, eixo, valorEixo, frameInicio, frameFim)
87         atribuirMaterial(material)
88
89
90     elif elemento == "suportedabase":
91         print("suportedabase")
92
```

Anexo 2 - Script Python Utilizado no Módulo 5.2:

```
1      """Provides a scripting component.
2          Inputs:
3              x: The x script variable
4              y: The y script variable
5          Output:
6              a: The a output variable"""
7
8      __author__ = "henrique"
9
10
11     import rhinoscriptsyntax as rs
12     import Rhino as rc
13     import scriptcontext as sc
14
15
16     coordenadaX = []
17     coordenadaY = []
18     coordenadaZ = []
19
20
21     for objID in rs.ObjectsByLayer("MTS TEXTOS"):
22         #print(objID)
23
24         objectoTexto = rs.coercerhinoobject(objID)
25         #print(objectoTexto)
26
27
28
29         valorX = objectoTexto.Geometry.Plane.Origin.X
30         #print("Valor X", valorX)
31         coordenadaX.append(valorX)
32
33         valorY = objectoTexto.Geometry.Plane.Origin.Y
34         #print("Valor Y", valorY)
35         coordenadaY.append(valorY)
36
37         valorZ = objectoTexto.Geometry.Text
38         #print("Valor Z", valorZ)
39         coordenadaZ.append(float(valorZ))
40
```

Anexo 3 - Script Python Utilizado no Módulo 5.3:

```
1      # Henrique Robalo
2
3      import rhinoscriptsyntax as rs
4      import csv
5
6      csv_file_path = "/Users/henrique/Desktop/S-OFFICE -
7      TabelaAtributosResumida.csv"
8
9      # 0=TipoDeElemento / 1=UniqueID / 2=FireResistance em minutos /
10     3=PurchasePrice / 4=PricePerVolume / 5=TotalVolumePrice
11     csv_objectIDlist = []
12     csv_blockReferenceNames = []
13     csv_objectID_FireResistance = {}
14     csv_objectID_PurchasePrice = {}
15     csv_objectID_PricePerVolume = {}
16     csv_objectID_TotalVolumePrice = {}
17
18     #Mecanismo para preencher os dicionarios com os valores do CSV
19     with open (csv_file_path, "r") as csv_file:
20
21         csv_reader = csv.reader(csv_file)
22         header = next(csv_reader)
23
24         for line in csv_reader: # completar os dicionarios com os valores do CSV
25
26             csv_objectID_FireResistance[line[1]] = line[2]
27             csv_objectID_PurchasePrice[line[1]] = line[3]
28             csv_objectID_PricePerVolume[line[1]] = line[4]
29             csv_objectID_TotalVolumePrice[line[1]] = line[5]
30             csv_objectIDlist.append(line[1])
31             csv_blockReferenceNames.append(line[0])
32
33
34     def mudar_a_cor(atributo, objecto3D_selecionado, valor_Max):
35         if atributo != "NULL":
36             red_color_value = int((int(float(atributo)) / int(valor_Max)) * 250)
37             green_color_value = 200
38             blue_color_value = 0
39             rs.ObjectColor(objecto3D_selecionado, (red_color_value,
40             green_color_value, blue_color_value))
41         else:
42             pass
43
44     for instancia_do_bloco in rs.ObjectsByType(4096, select=False, state=0):
45
46         ref_do_bloco = str(instancia_do_bloco).upper()
47
48         if rs.IsBlockInstance(str(instancia_do_bloco)):
49             print("is a Block Instance")
50
51         definicao_do_bloco = rs.BlockInstanceName(instancia_do_bloco)
52         if definicao_do_bloco is None: # Nao e necessario ter isto
53             print("blockinstance nao tem nome")
54
55         objectos_do_bloco = rs.BlockObjects(definicao_do_bloco)
56
57         if objectos_do_bloco is None: # Nao e necessario ter isto
58             print("blockdefinition nao tem objectos")
59
60         # Editar os objectos do bloco
61         for item in objectos_do_bloco:
62
63             if ref_do_bloco in csv_objectIDlist:
64                 fireResistance_value = csv_objectID_FireResistance[ref_do_bloco]
65                 totalVolumePrice_value =
66                 csv_objectID_TotalVolumePrice[ref_do_bloco]
67
68                 rs.ObjectColor(item, (250, 250, 250))
69
70                 mudar_a_cor(fireResistance_value, item, 90)
71                 #mudar_a_cor(totalVolumePrice_value, item, 62650)
72
73             else:
```

Anexo 4 - Script Python Utilizado no Módulo 5.4:

```
1 import rhinoscriptsyntax as rs
2
3 layers = rs.LayerNames()
4
5 for layer in layers:
6     if rs.IsLayerEmpty(layer):
7         rs.DeleteLayer(layer)
8
9
```

Anexo 5 - Script C# Utilizado no Módulo 5.5 (MoverUtilizador.cs):

```
1 //Este script foi realizado com aprendizagens de um tutorial em: "https://
  www.youtube.com/watch?v=_QajrabyTJc" por Brackeys
2
3 using System.Collections;
4 using System.Collections.Generic;
5 using UnityEngine;
6
7 public class MoverUtilizador : MonoBehaviour
8 {
9     public CharacterController controller;
10
11     public float speed = 12f;
12     public float gravidade = -9.81f;
13     public float jumpHeight = 3f;
14
15     public Transform groundCheck;
16     public float groundDistance = 0.4f;
17     public LayerMask groundMask;
18
19     Vector3 velocidade_gravidade;
20
21     bool isGrounded;
22
23
24     // Update is called once per frame
25     void Update()
26     {
27
28         isGrounded = Physics.CheckSphere(groundCheck.position, groundDistance,
18         groundMask);
29
30         if (isGrounded && velocidade_gravidade.y < 0)
31         {
32             velocidade_gravidade.y = -2f;
33
34         }
35
36
37         float input_teclado_x = Input.GetAxis("Horizontal");
38         float input_teclado_z = Input.GetAxis("Vertical");
39
40         Vector3 movimento = transform.right * input_teclado_x + transform.forward
* input_teclado_z;
41
42         controller.Move(movimento * speed * Time.deltaTime);
43
44         if(Input.GetButtonDown("Jump") && isGrounded)
45         {
46             velocidade_gravidade.y = Mathf.Sqrt(jumpHeight * -2f * gravidade);
47
48         }
49
50
51         velocidade_gravidade.y += gravidade * Time.deltaTime;
52
53         controller.Move(velocidade_gravidade * Time.deltaTime);
54
55
56     }
57 }
58
```

Anexo 6 - Script C# Utilizado no Módulo 5.5 (RodarUtilizador.cs):

```
1 //Este script foi realizado com aprendizagens de um tutorial em: "https://
  www.youtube.com/watch?v=_QajrabyTJc" por Brackeys
2
3 using System.Collections;
4 using System.Collections.Generic;
5 using UnityEngine;
6
7 public class RodarUtilizador : MonoBehaviour
8 {
9     public float sensibilidade_do_rato = 100f;
10    public Transform objecto_do_utilizador;
11
12    float rotacao_no_eixo_x = 0f;
13
14    // Start is called before the first frame update
15    void Start()
16    {
17        Cursor.lockState = CursorLockMode.Locked;
18    }
19
20    // Update is called once per frame
21    void Update()
22    {
23        float rato_posicao_x = Input.GetAxis("Mouse X") * sensibilidade_do_rato *
Time.deltaTime;
24        float rato_posicao_y = Input.GetAxis("Mouse Y") * sensibilidade_do_rato *
Time.deltaTime;
25
26        rotacao_no_eixo_x -= rato_posicao_y;
27
28        rotacao_no_eixo_x = Mathf.Clamp(rotacao_no_eixo_x, -90f, 90f);
29
30        transform.localRotation = Quaternion.Euler(rotacao_no_eixo_x, 0f, 0f);
31        objecto_do_utilizador.Rotate(Vector3.up * rato_posicao_x);
32    }
33 }
34 }
35
```

Anexo 7 - Script C# Utilizado no Módulo 5.5 (ClickObjecto.cs):

```
1 //Este script foi realizado com aprendizagens de um tutorial em: "https://
  www.youtube.com/watch?v=EANtTI6BCxk" por Omnirift
2 //Este script foi realizado com aprendizagens de um tutorial em: "https://
  www.youtube.com/watch?v=FGiCejM743g" por Quick Tutorial
3
4 using System.Collections;
5 using System.Collections.Generic;
6 using UnityEngine;
7
8 public class ClickObjectos : MonoBehaviour
9 {
10     private Vector3 vector_rotacao = new Vector3 (0f, 6f, 0f);
11     Quaternion target = Quaternion.Euler(0, 80, 0);
12     Quaternion angulo_porta_aberta = Quaternion.Euler(0, 90, 0);
13     Quaternion angulo_porta_fechada = Quaternion.Euler(0, 0, 0);
14     public Quaternion angulo_atual;
15     GameObject objecto_porta;
16
17
18     private void Start()
19     {
20         angulo_atual = angulo_porta_fechada;
21     }
22
23
24     private void Update()
25     {
26         if (Input.GetMouseButtonDown(0))
27         {
28             RaycastHit hit;
29             Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
30
31             if (Physics.Raycast(ray, out hit, 100f))
32             {
33                 //var selection = hit.transform;
34                 if (hit.transform != null &&
35                     hit.transform.gameObject.name.Substring(0, 5) == "porta")
36                 {
37                     PrintName(hit.transform.gameObject);
38                     objecto_porta = hit.transform.gameObject;
39                     alternar_aberto_fechado();
40                     print(hit.transform.gameObject.name.Substring(0, 5));
41                 }
42             }
43
44             objecto_porta.transform.rotation =
45             Quaternion.Slerp(objecto_porta.transform.rotation, angulo_atual, 0.2f);
46         }
47
48         private void PrintName(GameObject game_object)
49         {
50             print(game_object.name);
51         }
52
53         private void rodarObjecto(GameObject objecto_clicado)
54         {
55             print(objecto_clicado.name);
56             objecto_clicado.transform.Rotate(vector_rotacao * Time.deltaTime);
57         }
58
59         void alternar_aberto_fechado()
60         {
61             if (angulo_atual.eulerAngles.y == angulo_porta_fechada.eulerAngles.y)
62             {
63                 angulo_atual = angulo_porta_aberta;
64             }
65             else
66             {
67                 angulo_atual = angulo_porta_fechada;
68             }
69         }
70     }
```

Tabelas

Anexo 8 - Análise comparativa de softwares de arquitetura

posicao	Nome	Tipologia	CAD	BIM	AAD	AABIM	API	Rendering	Animação	ACESSIBILIDADE	LINUX	OSX	WINDOWS	Notas
1	LibreCAD	programa independente	X				????			Open-Source	X	X	X	
2	FreeCad	programa independente	X	X	X	X	X		X	Open-Source	X	X	X	Jürgen Riegel / Werner Mayer / York van Have
3	SketchUp	programa independente	X				X	(Plugin)	(Plugin)	Subscrição Paga		X	X	
4	3DS Max	programa independente	X				????	X	X			X	X	
5	Rhinoceros 3D	programa independente	X		X		X	X	(Plugin)	Licença Paga		X	X	Sculptura 1993 até 1994
6	MicroStation	programa independente	X		X		????					X	X	
7	AutoCAD	programa independente	X				????					X	X	
8	CATIA	programa independente	X		????				????			X	X	
11	ArchiCad	programa independente		X		(Plugin)		X		Licença Paga		X	X	
12	REVIT	programa independente		X	X	(Plugin)	????	????		Subscrição Paga			X	
13	Blender	programa independente			X		X	X	X	Open-Source	X	X	X	Ton Rosendal
14	Cinema 4D	programa independente			X		X	X	X			X	X	
15	Vray	plugin						X				????	X	

Legenda

X - Tem

(Plugin) - Tem através de plugin

???? - Falta Confirmar

Na coluna | Rendering refere-se a capacidade de Ray-Tracing

Anexo 9 - Lista de software de arquitetura

CAD Comercial	CAD Open-source	CAD Freeware	BIM	AAD
AC3D	BRL-CAD	BricsCAD Shape	Archicad	FreeCAD
Alibre Design	FreeCAD	Tinkercad (sucessor do Autodesk 123D)	REVIT	GrassHopper (Rhinoceros3D)
ArchiCAD (Graphisoft)	LibreCAD		BricsCAD	Geometry Nodes (Blender)
AutoCAD (Autodesk)	OpenSCAD		ALLPLAN ARCHITECTURE	Dynamo (Revit)
Autodesk Inventor	QCAD		actCAD	Generative Components (Bentley Systems)
AxSTREAM	Salome		VectorWorks Architect	Param-O (Archicad)
BricsCAD	SolveSpace		Tekla Structures (Engenharia civil)	Xpresso (Cinema 4D)
CATIA (Dassault Systèmes)	JSketcher		bentley systems openbuildings designer (AECOSim Building Designer)	Surface Operators (SOP) (Houdini)
Cobalt			NavisWork	
CorelCAD				
Fusion 360 (Autodesk)				
IntelliCAD				
Inventor (Autodesk)				
IRONCAD				
KeyCreator (Kubotek)				
Landscape Express				
MEDUSA				
MicroStation (Bentley Systems)				
Modelur (AgiliCity)				
Onshape				
Promine				
PTC Creo (sucessor do Pro/ENGINEER)				
PunchCAD				
Remo 3D				
Rhinoceros 3D				
Siemens NX				
SketchUp (Trimble)				
Solid Edge (Siemens)				
SolidWorks (Dassault Systèmes)				
SpaceClaim				
T-FLEX CAD				
TurboCAD				
Vectorworks (Nemetschek)				

Lista elaborada com informação recolhida em: https://en.wikipedia.org/wiki/Computer-aided_design

Programação aplicada ao desenvolvimento do projecto

Henrique Robalo

FACULDADE DE ARQUITETURA





Henrique Robalo. Programação aplicada ao desenvolvimento do projecto



M.FAUP 2022