# Mestrado Próprio

## Programação de Videojogos


tech universidade tecnológica

tech universidade tecnológica

# Mestrado Próprio
## Programação de Videojogos

- Modalidade: **online**
- Duração: **12 meses**
- Certificação: **TECH Universidade Tecnológica**
- Créditos: **60 ECTS**
- Tempo Dedicado: **16 horas/semana**
- Horário: **ao seu próprio ritmo**
- Exames: **online**

Acesso ao site: www.techtitute.com/pt/videojogo/mestrado-proprio/mestrado-proprio-programacao-videojogos

# Índice


01
Apresentação
*pág. 4*

02
Objetivos
*pág. 8*

03
Competências
*pág. 12*

04
Estrutura e conteúdo
*pág. 16*

05
Metodologia
*pág. 30*

06
Certificação
*pág. 38*

01

# Apresentação

Uma das tarefas mais importantes e delicadas quando se realiza um projeto de videojogo é a programação. A programação está no centro do videojogo, pois é o processo que cria as suas instruções básicas e dita o seu funcionamento global. Ou seja, sem o código feito pelos programadores, os elementos visuais, a história e a jogabilidade não se poderiam destacar numa obra audiovisual deste tipo. Assim, este curso oferece aos seus alunos todo o conhecimento para se tornarem os melhores programadores da indústria, de forma a que as melhores empresas queiram contar com eles para desenvolverem os seus projetos.

*Aprenda a programar os melhores videojogos do mundo graças a este Mestrado Próprio"*

A indústria dos videojogos tem experienciado uma grande expansão nos últimos anos. À medida que esta forma de entretenimento se tornou mais popular, as empresas do setor têm sido obrigadas a conceber e publicar jogos com mais frequência. Além disso, também tem sido necessária mais criatividade à medida que os jogadores exigem cada vez mais títulos mais variados, de diferentes géneros e que ofereçam experiências inovadoras.

Por esta razão, a indústria exige que os especialistas em programação de videojogos se encarreguem da tarefa fundamental de criar o código para as suas novas obras. Este trabalho é delicado e requer uma grande especialização, pelo que é aconselhável ter levado a cabo um processo de aprendizagem minucioso e otimizado para se tornar um verdadeiro especialista.

Assim, este Mestrado Próprio em Programação de Videojogos é o que os profissionais precisam para poderem ter acesso a uma grande empresa da indústria, trabalhando no seu departamento de desenvolvimento. Ao longo desta capacitação, os alunos aprenderão as noções básicas de programação e engenharia de *software*, estrutura de dados e algoritmos, programação orientada para objectos e outras questões mais específicas, tais como motores de videojogos ou programação em tempo real.

Desta forma, garante-se que os estudantes adquirem os melhores conhecimentos possíveis para que possam aplicá-los diretamente nos seus campos de trabalho.

Este **Mestrado Próprio em Programação de Videojogos** conta com o conteúdo científico mais completo e atualizado do mercado. As suas principais características são:

- O desenvolvimento de casos práticos apresentados por especialistas em programação e desenvolvimento de videojogos
- O conteúdo gráfico, esquemático e eminentemente prático do livro fornece informações científicas e práticas sobre as disciplinas que são essenciais para a prática profissional
- Exercícios práticos onde o processo de autoavaliação pode ser levado a cabo a fim de melhorar a aprendizagem
- A sua ênfase especial em metodologias inovadoras
- As lições teóricas, perguntas ao especialista, fóruns de discussão sobre questões controversas e atividades de reflexão individual
- A disponibilidade de acesso ao conteúdo a partir de qualquer dispositivo fixo ou portátil com ligação à internet

*As melhores empresas do setor vão querer contar consigo"*

*Se quer desenvolver os melhores videojogos do mundo, este curso ensina-lhe a fazê-lo"*

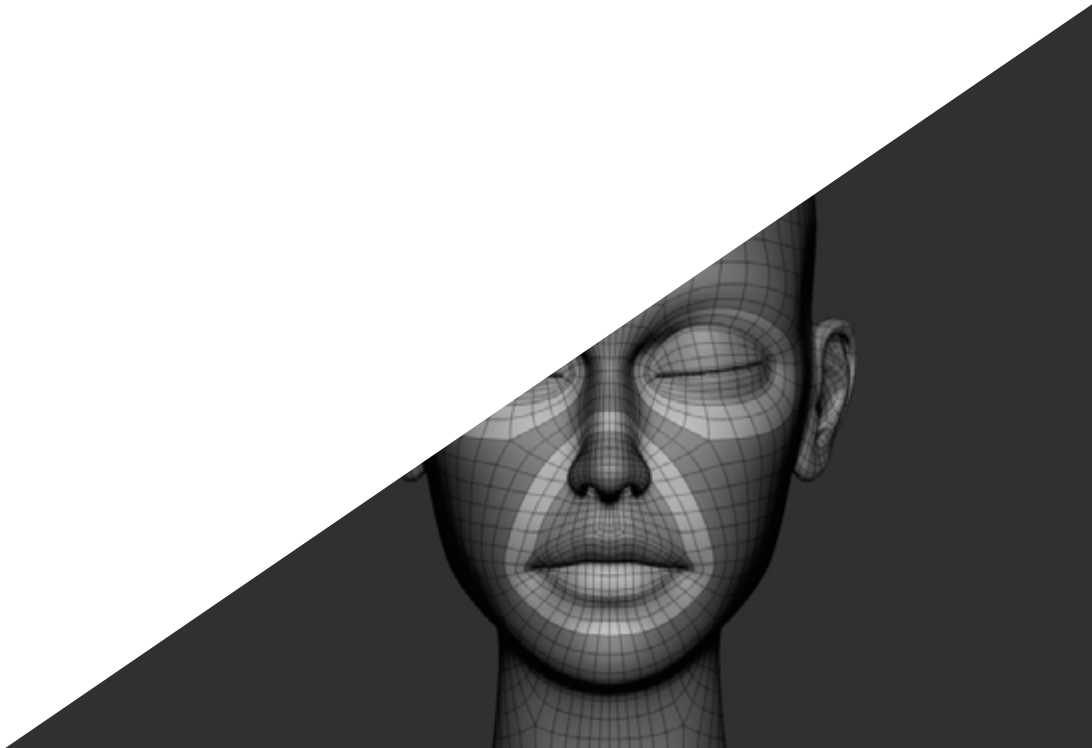
*Programe os videojogos dos seus sonhos graças a este Mestrado Próprio.*

*Não espere mais: programe videojogos como os melhores especialistas.*

O corpo docente do curso inclui profissionais do setor que trazem a sua experiência profissional para esta formação, para além de especialistas reconhecidos de sociedades de referência e universidades de prestígio.

Graças ao seu conteúdo multimédia, desenvolvido com a mais recente tecnologia educacional, o profissional terá acesso a uma aprendizagem situada e contextual, ou seja, um ambiente de simulação que proporcionará um programa imersivo programado para se formar em situações reais.

A conceção deste programa baseia-se na Aprendizagem Baseada nos Problemas, através da qual o profissional deve tentar resolver as diferentes situações da prática profissional que surgem ao longo do curso académico. Para tal, contará com a ajuda de um sistema inovador de vídeo interativo desenvolvido por especialistas reconhecidos.



02

# Objetivos

O principal objetivo deste Mestrado Próprio em Programação de Videojogos é oferecer aos alunos os melhores conhecimentos para que se tornem os melhores especialistas em desenvolvimento de videojogos no seu meio. Para tal, esta capacitação oferece-lhes uma série de ferramentas aplicadas a este âmbito que irão melhorar o seu trabalho como programadores e levá-los a alcançar todos os seus objetivos profissionais, podendo programar os melhores videojogos do mundo.

*Alcance todos os seus objetivos graças a esta capacitação”*

## Objetivos gerais

- Conhecer as diferentes linguagens e métodos de programação aplicados aos videojogos
- Aprofundar-se no processo de produção de um videojogo e na integração da programação nestas fases
- Aprender as bases do *design* de videojogos e os conhecimentos teóricos que um *designer* de videojogos deve conhecer
- Dominar as linguagens de programação básicas utilizadas nos videojogos
- Aplicar conhecimentos de engenharia de *software* e programação especializada a videojogos
- Compreender o papel da programação no desenvolvimento um videojogo
- Conhecer as diferentes consolas e plataformas existentes
- Desenvolver videojogos para a web e de multijogadores

*Se quer um emprego nas melhores empresas do mundo, este curso ajuda-lo-á a consegui-lo"*

## Objetivos específicos

### Módulo 1. Fundamentos de programação

- Compreender a estrutura básica de um computador, *software* e linguagens de programação de uso geral
- Analisar os elementos essenciais de um programa informático, tal como os diferentes tipos de dados, operadores, expressões, declarações, I/O e declarações de controlo
- Interpretar algoritmos que são a base necessária para o desenvolvimento de programas informáticos

### Módulo 2. Estrutura de dados e algoritmos

- Aprender as principais estratégias para a conceção de algoritmos, bem como os diferentes métodos e medidas para o cálculo de algoritmos
- Distinguir o funcionamento dos algoritmos, a sua estratégia e os exemplos da sua utilização nos principais problemas conhecidos
- Compreender a técnica do *Backtracking* e as suas principais utilizações

### Módulo 3. Programação orientada para objetos

- Conhecer os diferentes padrões de conceção para problemas orientados a objetos
- Compreender a importância da documentação e das provas no desenvolvimento de *software*
- Aprender a gerir a utilização dos fios a e sincronização, bem como a resolução de problemas comuns no âmbito da programação concorrente

### Módulo 4. Consolas e dispositivos de videojogos

- Conhecer o funcionamento básico dos periféricos principais de entrada e saída
- Compreender as principais implicações de conceção das diferentes plataformas
- Estudar a estrutura, a organização, o funcionamento e a interligação de dispositivos e sistemas
- Compreender o papel do sistema operativo e dos kits de desenvolvimento para dispositivos móveis e plataformas de videojogos

### Módulo 5. Engenharia de *software*

- Distinguir os fundamentos da engenharia de *software*, bem como o processo de *software* e os diferentes modelos para o seu desenvolvimento, incluindo as tecnologias ágeis
- Reconhecer a engenharia de requisitos, o seu desenvolvimento, a elaboração, negociação e validação, a fim de compreender as principais normas relacionadas com a qualidade do *software* e a gestão de projetos

### Módulo 6. Motores de videojogos

- Descubrir o funcionamento e a arquitetura de um motor de videojogos
- Compreender as características básicas dos motores de jogo existentes
- Programar aplicações de forma correta e eficiente aplicados a motores de videojogos
- Escolher o paradigma e as linguagens de programação mais apropriadas para programar aplicações aplicadas a motores de videojogos

### Módulo 7. Sistemas inteligentes

- Estabelecer os conceitos relacionados com a teoria e a arquitetura dos agentes e o seu processo de raciocínio
- Assimilar a teoria e a prática por detrás dos conceitos de informação e conhecimento, bem como as diferentes formas de representação do conhecimento
- compreender o funcionamento dos raciocinadores semânticos, dos sistemas baseados no conhecimento e dos sistemas especializados

### Módulo 8. Programação em tempo real

- Analisar as características-chave de uma linguagem de programação em tempo real que a diferencia de uma linguagem de programação tradicional
- Compreender os conceitos básicos dos sistemas informáticos
- Adquirir a capacidade de aplicar as principais bases e técnicas de programação em tempo real

### Módulo 9. Conceção e desenvolvimento de jogos para a web

- Design de jogos e aplicações web interativas com a documentação correspondente
- Avaliar as principais características dos jogos e aplicações interativas da web, a fim de comunicar de uma forma profissional e correta

### Módulo 10. Redes e sistemas multijogador

- Descrever a arquitetura do protocolo de controlo de transmissão/protocolo de internet (TCP/IP) e o funcionamento básico das redes sem fios
- Análise da segurança aplicada aos videojogos
- Adquirir a capacidade de desenvolver jogos online para multijogadores

03

# Competências

Os alunos deste Mestrado Próprio obterão uma série de competências que os transformarão em verdadeiros especialistas no desenvolvimento de videojogos, de forma a poderem juntar-se a qualquer tipo de projeto na indústria. Desta forma, os estudantes dominarão questões relacionadas com diferentes linguagens de programação específicas utilizadas neste tipo de produtos audiovisuais, bem como competências transversais que devem conhecer, tais como o âmbito das consolas e plataformas e dos motores de videojogos.

*Saberá tudo o que precisa para desenvolver grandes videojogos”*

## Competências gerais

- Conceber todas as fases de um videojogo, desde a ideia inicial até ao lançamento final
- Especializar-se como programador de videojogos
- Investigar todas as partes do desenvolvimento, desde a arquitetura inicial, a programação da personagem do jogador e todos os elementos envolvidos no processo do jogo
- Obter uma visão global do projeto, sendo capaz de apontar soluções para os diferentes problemas e desafios que surjam no *design* de um videojogo

*Domine todos os tipos de linguagens de programação aplicadas aos videojogos com este Mestrado Próprio"*



## Competências específicas

- Conhecer o *software* necessário para ser um profissional do desenvolvimento de videojogos
- Compreender a experiência do jogador e saber analisar a jogabilidade do videojogo
- Compreender todo o procedimento teórico e prático do processo de programação de um videojogo
- Dominar as linguagens de programação mais úteis para o mundo dos videojogos
- Integrar a programação aprendida em diferentes tipos de consolas e plataformas
- Programar videojogos para a *web* e de multijogadores
- Assimilar o conceito de motor de videojogo de modo a poder programar corretamente
- Aplicar conhecimentos de engenharia de *software* à programação de videojogos

04

# Estrutura e conteúdo

Este Mestrado Próprio em Programação de VIdeojogos oferece aos seus alunos o melhor conteúdo no desenvolvimento de videojogos, graças à sua cuidadosa conceção e estruturada em 10 módulos com 10 disciplinas cada. Através deles, os estudantes poderão aprender tudo o que precisam para participar em qualquer tipo de projeto de videojogos, para que o seu processo educacional seja completo, integral e totalmente focado na prática.

*Os conteúdos de que precisa para se especializar em programação de videojogos"*

## **Módulo 1.** Fundamentos de programação

1.1. Introdução à programação
- 1.1.1. Estrutura básica de um ordenador
- 1.1.2. *Software*
- 1.1.3. Linguagens de programação
- 1.1.4. Ciclo de vida uma aplicação informática

1.2. Conceção de algoritmos
- 1.2.1. Resolução de problemas
- 1.2.2. Técnicas descritivas
- 1.2.3. Elementos e estrutura de um algoritmo

1.3. Elementos de um programa
- 1.3.1. Origem e caraterísticas da linguagem C++
- 1.3.2. O ambiente de desenvolvimento
- 1.3.3. Conceito de programa
- 1.3.4. Tipos de dados fundamentais
- 1.3.5. Operadores
- 1.3.6. Expressões
- 1.3.7. Declarações
- 1.3.8. Entrada e saída de dados

1.4. Declarações de controlo
- 1.4.1. Declarações
- 1.4.2. Bifurcações
- 1.4.3. Laços

1.5. Abstração e modularidade: funções
- 1.5.1. Desenho modular
- 1.5.2. Conceito de função e utilidade
- 1.5.3. Definição de uma função
- 1.5.4. Fluxo de execução na chamada de uma função
- 1.5.5. Protótipo de uma função
- 1.5.6. Devolução de resultados
- 1.5.7. Chamada a uma função: parâmetros
- 1.5.8. Passagem de parâmetros por referência e por valor
- 1.5.9. Âmbito identificador

1.6. Estruturas de dados estáticas
- 1.6.1. *Arrays*
- 1.6.2. Matrizes. Poliedros
- 1.6.3. Pesquisa e ordenação
- 1.6.4. Cadeias. Funções de E/S para cadeias
- 1.6.5. Estruturas Uniões
- 1.6.6. Novos tipos de dados

1.7. Estruturas de dados dinâmicas: ponteiros
- 1.7.1. Conceito Definição de ponteiro
- 1.7.2. Operadores e operações com ponteiros
- 1.7.3. *Arrays* de ponteiros
- 1.7.4. Ponteiros e *arrays*
- 1.7.5. Ponteiros a cadeias
- 1.7.6. Ponteiros a estruturas
- 1.7.7. Indireção múltipla
- 1.7.8. Ponteiros a funções
- 1.7.9. Passagem de funções, estruturas e *arrays* como parâmetros de funções

1.8. Ficheiros
- 1.8.1. Conceitos básicos
- 1.8.2. Operações com ficheiros
- 1.8.3. Tipos de ficheiros
- 1.8.4. Organização dos ficheiros
- 1.8.5. Introdução aos ficheiros CC++
- 1.8.6. Gestão de ficheiros

1.9. Recursividade
- 1.9.1. Definição de recursividade
- 1.9.2. Tipos de recursividade
- 1.9.3. Vantagens e desvantagens
- 1.9.4. Considerações
- 1.9.5. Conversão recursivo-iterativa
- 1.9.6. A pilha de repetição



1.10. Prova e documentação
- 1.10.1. Provas de programas
- 1.10.2. Prova da caixa branca
- 1.10.3. Prova da caixa negra
- 1.10.4. Ferramentas para realizar as provas
- 1.10.5. Documentação de programas

## **Módulo 2.** Estrutura de dados e algoritmos

2.1. Introdução às estratégias de desenho do algoritmos
- 2.1.1. Recursividade
- 2.1.2. Divide e conquista
- 2.1.3. Outras estratégias

2.2. Eficiência e análise dos algoritmos
- 2.2.1. Medidas de eficiência
- 2.2.2. Medir o tamanho da entrada
- 2.2.3. Medir o tempo de execução
- 2.2.4. Caso pior, melhor e médio
- 2.2.5. Notação assintótica
- 2.2.6. Critérios de análise matemática de algoritmos não recursivos
- 2.2.7. Análise matemática de algoritmos recursivos
- 2.2.8. Análise empírica de algoritmos

2.3. Algoritmos de ordenação
- 2.3.1. Conceito de ordenação
- 2.3.2. Ordenação da bolha
- 2.3.3. Ordenação por seleção
- 2.3.4. Ordenação por inserção
- 2.3.5. Ordenação por mistura (*Merge_Sort*)
- 2.3.6. Ordenação rápida (*Quicksort*)

2.4. Algoritmos com árvores
- 2.4.1. Conceito de árvore
- 2.4.2. Árvores binários
- 2.4.3. Caminhos de árvore
- 2.4.4. Representar expressões
- 2.4.5. Árvores binários ordenadas
- 2.4.6. Árvores binárias equilibradas

2.5. Algoritmos com heaps
- 2.5.1. Os heaps
- 2.5.2. O algoritmo Heapsort
- 2.5.3. As filas de prioridade

2.6. Algoritmos com gráficos
- 2.6.1. Representação
- 2.6.2. Caminho de largura
- 2.6.3. Percurso em profundidade
- 2.6.4. Ordenação topológica

2.7. Algoritmos Greedy
- 2.7.1. A estratégia Greedy
- 2.7.2. Elementos da estratégia Greedy
- 2.7.3. Câmbio de moedas
- 2.7.4. Problema do viajante
- 2.7.5. Problema da mochila

2.8. Pesquisa de caminhos mínimos
- 2.8.1. O problema do caminho mínimo
- 2.8.2. Arcos negativos e ciclos
- 2.8.3. Algoritmo de Dijkstra

2.9. Algoritmos Greedy sobre grafos
- 2.9.1. A árvore de extensão mínima
- 2.9.2. O algoritmo de Prim
- 2.9.3. O algoritmo Kruskal
- 2.9.4. Análise de complexidade

2.10. *Backtracking*
- 2.10.1. O *Backtracking*
- 2.10.2. Técnicas alternativas

## **Módulo 3.** Programação orientada para objetos

3.1. Introdução à programação orientada a objetos
- 3.1.1. Introdução à programação orientada a objetos
- 3.1.2. Desenho de classes
- 3.1.3. Introdução à UML para modelação de problemas

3.2. Relações entre classes
- 3.2.1. Abstração e herança
- 3.2.2. Conceitos avançados de herança
- 3.2.3. Poliformismo
- 3.2.4. Composição e agregação

3.3. Introdução aos padrões de de desenho para problemas orientados a objetos
- 3.3.1. O que é um padrão de desenho?
- 3.3.2. Padrão *Factory*
- 3.3.3. Padrão *Singleton*
- 3.3.4. Padrão *Observer*
- 3.3.5. Padrão *Composite*

3.4. Exceções
- 3.4.1. O que são as exceções
- 3.4.2. Captura e gestão de exceções
- 3.4.3. Lançamento de exceções
- 3.4.4. Criação de exceções

3.5. Interfaces de utilizadores
- 3.5.1. Introdução a Qt
- 3.5.2. Posicionamento
- 3.5.3. O que são os eventos?
- 3.5.4. Eventos: definição e captura
- 3.5.5. Desenvolvimento de interfaces de utilizador

3.6. Introdução à programação concorrente
- 3.6.1. Introdução à programação concorrente
- 3.6.2. O conceito de processo e *thread*
- 3.6.3. Interação entre processos ou *threads*
- 3.6.4. Os threads em C++
- 3.6.5. Vantagens e desvantagens da programação concorrente

3.7. Gestão de threads e sincronização
- 3.7.1. Ciclo de vida um *thread*
- 3.7.2. A classe *Thread*
- 3.7.3. Planificação de *threads*
- 3.7.4. Grupos *threads*
- 3.7.5. *Threads* de tipo demónio
- 3.7.6. Sincronização
- 3.7.7. Mecanismos de bloqueio
- 3.7.8. Mecanismos de comunicação
- 3.7.9. Monitores

3.8. Problemas comuns dentro da programação concorrente
- 3.8.1. O problema dos produtores consumidores
- 3.8.2. O problema dos leitores e escritores
- 3.8.3. O problema do jantar dos filósofos

3.9. Documentação e provas de *software*
- 3.9.1. Porque é importante documentar o *software*?
- 3.9.2. Documentação de desenho
- 3.9.3. Uso de ferramentas para a documentação

3.10. Provas de *software*
- 3.10.1. Introdução às provas de *software*
- 3.10.2. Tipos de provas
- 3.10.3. Prova de unidade
- 3.10.4. Prova de integração
- 3.10.5. Prova de validação
- 3.10.6. Prova do sistema

## **Módulo 4.** Consolas e dispositivos de videojogos

4.1. História da programação de videojogos
- 4.1.1. Período Atari (1977-1985)
- 4.1.2. Período NES e SNES (1985-1995)
- 4.1.3. Período PlayStation/PlayStation 2 (1995-2005)
- 4.1.4. Xbox 360, PS3 e período Wii (2005-2013)
- 4.1.5. Xbox One, PS4 e Wii U - Período de troca (2013-presente)
- 4.1.6. O futuro

4.2. História da jogabilidade em videojogos
- 4.2.1. Introdução
- 4.2.2. Contexto social
- 4.2.3. Diagrama estrutural
- 4.2.4. Futuro

4.3. Adaptação aos tempos modernos
- 4.3.1. Jogos baseados no movimento
- 4.3.2. Realidade Virtual (RV)
- 4.3.3. A realidade aumentada
- 4.3.4. Realidade mista

4.4. *Unidade: Scripting I* e exemplos
- 4.4.1. O que é um script?
- 4.4.2. O nosso primeiro script
- 4.4.3. Acrescentar um script
- 4.4.4. Abertura de um script
- 4.4.5. MonoBehaviour
- 4.4.6. *Debugging*

4.5. *Unidade: Scripting II* e exemplos
- 4.5.1. Entrada de teclado e rato
- 4.5.2. Raycast
- 4.5.3. Instalação
- 4.5.4. Variáveis
- 4.5.5. Variáveis públicas e seriadas

4.6. *Unidade: Scripting III* e exemplos
- 4.6.1. Obtenção de componentes
- 4.6.2. Modificação de componentes
- 4.6.3. Testes
- 4.6.4. Objetos múltiplos
- 4.6.5. *Colisores e gatilhos*
- 4.6.6. Quaterniões

4.7. Periféricos
- 4.7.1. Evolução e classificação
- 4.7.2. Periféricos e Interfaces
- 4.7.3. Periféricos atuais
- 4.7.4. Futuro próximo

4.8. Videojogos: perspetivas futuras
- 4.8.1. Jogos baseados nas nuvens
- 4.8.2. Ausência de comandos
- 4.8.3. Realidade imersiva
- 4.8.4. Outras alternativas

4.9. Arquitetura
- 4.9.1. Requisitos especiais dos videojogos
- 4.9.2. Evolução da arquitetura
- 4.9.3. Arquitetura atual
- 4.9.4. Diferenças entre arquiteturas

4.10. Kits de desenvolvimento e a sua evolução
- 4.10.1. Introdução
- 4.10.2. Kits de desenvolvimento de terceira geração
- 4.10.3. Kits de desenvolvimento de quarta geração
- 4.10.4. Kits de desenvolvimento de quinta geração
- 4.10.5. Kits de desenvolvimento de sexta geração

## **Módulo 5.** Engenharia de *software*

5.1. Introdução à engenharia do *software* e à modelação
- 5.1.1. A natureza do *software*
- 5.1.2. A natureza única das webapps
- 5.1.3. Engenharia do *software*
- 5.1.4. O processo do *software*
- 5.1.5. A prática da engenharia do *software*
- 5.1.6. Mitos do *software*
- 5.1.7. Como é que tudo começa?
- 5.1.8. Conceitos orientados a objetos
- 5.1.9. Introdução ao UML

5.2. O processo do *software*
- 5.2.1. Um modelo geral de processo
- 5.2.2. Modelo de processo prescritivo
- 5.2.3. Modelo de processo especializado
- 5.2.4. O processo unificado
- 5.2.5. Modelos do processo pessoal e da equipa
- 5.2.6. O que é a agilidade?
- 5.2.7. O que é um processo ágil?
- 5.2.8. Scrum
- 5.2.9. Conjunto de ferramentas para o processo ágil

5.3. Princípios que orientam a prática da engenharia do *software*
- 5.3.1. Princípios que orientam o processo
- 5.3.2. Princípios que orientam a prática
- 5.3.3. Princípios de comunicação
- 5.3.4. Princípios de planificação
- 5.3.5. Princípios de modelação
- 5.3.6. Princípios de construção
- 5.3.7. Princípios de implantação

5.4. Compreensão dos requisitos
- 5.4.1. Engenharia de requisitos
- 5.4.2. Estabelecer as bases
- 5.4.3. Indagação dos requisitos
- 5.4.4. Desenvolvimento de casos de utilização
- 5.4.5. Elaboração do modelo dos requisitos
- 5.4.6. Negociação dos requisitos
- 5.4.7. Validação de requisitos

5.5. Modelação dos requisitos: cenários, informação e classes análise
- 5.5.1. Análise dos requisitos
- 5.5.2. Modelação baseada em cenários
- 5.5.3. Modelos UML que fornecem o caso de utilização
- 5.5.4. Conceitos de modelação de dados
- 5.5.5. Modelação baseada em classes
- 5.5.6. Diagrama de classes

5.6. Modelação de requisitos: fluxo, comportamento e padrões
- 5.6.1. Requisitos que modelam as estratégias
- 5.6.2. Modelação orientada ao fluxo
- 5.6.3. Diagramas de estado
- 5.6.4. Criação de um modelo de comportamento
- 5.6.5. Diagrama de sequência
- 5.6.6. Diagramas de comunicação
- 5.6.7. Padrões para modelação de requisitos

5.7. Conceitos de *design*
- 5.7.1. *Design* no contexto da engenharia do *software*
- 5.7.2. O processo de *design*
- 5.7.3. Conceitos de *design*
- 5.7.4. Conceitos de *design* orientado para objetos
- 5.7.5. O modelo do *design*

5.8. *Design* da arquitetura
- 5.8.1. Arquitetura do *software*
- 5.8.2. Géneros arquitetónicos
- 5.8.3. Estilos arquitetónicos
- 5.8.4. *Design* arquitetónico
- 5.8.5. Evolução dos *designs* alternativos para a arquitetura
- 5.8.6. Mapeamento da arquitetura com a utilização do fluxo de dados

5.9. *Design* no nível de componentes e baseado em padrões
- 5.9.1. O que é um componente?
- 5.9.2. *Design* de componentes baseados em classe
- 5.9.3. Realização do *design* a nível de componentes
- 5.9.4. *Design* de componentes tradicionais
- 5.9.5. Desenvolvimento baseado em componentes
- 5.9.6. Padrões de *design*
- 5.9.7. *Design* de *software* baseado em padrões
- 5.9.8. Padrões arquitetónicos
- 5.9.9. Padrões de desenho a nível de componentes
- 5.9.10. Padrões do *design* da interface de utilizador

5.10. Qualidade do *software* e administração de projetos
- 5.10.1. Qualidade
- 5.10.2. Qualidade do *software*
- 5.10.3. O dilema da qualidade do *software*
- 5.10.4. Conseguir a qualidade do *software*
- 5.10.5. Garantia de qualidade de *software*
- 5.10.6. O espetro administrativo
- 5.10.7. O pessoal
- 5.10.8. O produto
- 5.10.9. O processo
- 5.10.10. O projeto
- 5.10.11. Princípios e práticas

## **Módulo 6.** Motores de videojogos

6.1. Os videjogos e as TIC
- 6.1.1. Introdução
- 6.1.2. Oportunidades
- 6.1.3. Desafios
- 6.1.4. Conclusões

6.2. História dos motores de videojogos
- 6.2.1. Introdução
- 6.2.2. Anos Atari
- 6.2.3. Anos 80
- 6.2.4. Primeiros motores. Anos 90
- 6.2.5. Motores atuais

6.3. Motores de videojogos
- 6.3.1. Tipos de motores
- 6.3.2. Partes de um motor de videojogo
- 6.3.3. Motores atuais
- 6.3.4. Seleção de um motor para o nosso projeto

6.4. Motor *Game Maker*
- 6.4.1. Introdução
- 6.4.2. Desenho de cenários
- 6.4.3. *Sprites* e animações
- 6.4.4. Colisões
- 6.4.5. Scripting em GML

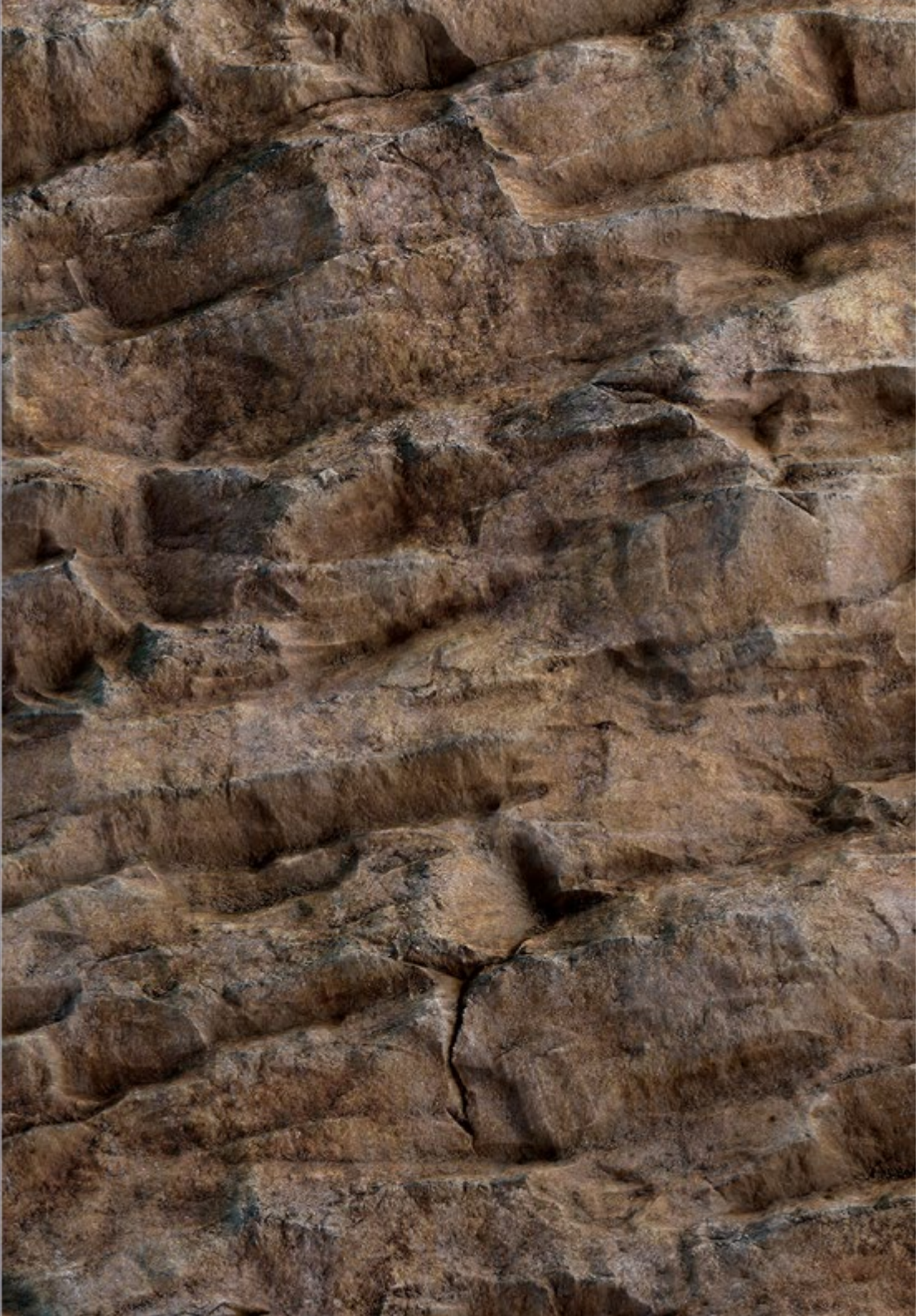
6.5. Motor *Unreal Engine* 4: introdução
- 6.5.1. O que é o *Unreal Engine* 4? Qual é a sua filosofia?
- 6.5.2. Materiais
- 6.5.3. UI
- 6.5.4. Animações
- 6.5.5. Sistema de partículas
- 6.5.6. Inteligência artificial
- 6.5.7. FPS

6.6. Motor *Unreal Engine* 4: *Visual Scripting*
- 6.6.1. Filosofia dos *Blueprints* e o *Visual Scripting*
- 6.6.2. *Debugging*
- 6.6.3. Tipos de variáveis
- 6.6.4. Controlo básico do fluxo

6.7. Motor Unity 5
- 6.7.1. Programação em C# e Visual Studio
- 6.7.2. Criação de *pré-fabricados*
- 6.7.3. Utilização do Gizmos para controlar o videojogo
- 6.7.4. Motor adaptativo: 2D e 3D

6.8. Motor Godot
- 6.8.1. Filosofia do *design* de Godot
- 6.8.2. *Design* e composição orientada a objectos
- 6.8.3. Tudo incluído num pacote
- 6.8.4. *Software* livre e orientado pela comunidade

6.9. Motor RPG Maker
- 6.9.1. Filosofia do RPG Maker
- 6.9.2. Tomando como referência
- 6.9.3. Criar um jogo com personalidade
- 6.9.4. Jogos comerciais com sucesso





6.10. Motor Source 2
- 6.10.1. Filosofia do Source 2
- 6.10.2. Source e Source 2: evolução
- 6.10.3. Utilização comunitária: conteúdo audiovisual e videojogos
- 6.10.4. Futuro do motor Source 2
- 6.10.5. Mods e jogos de sucesso

## **Módulo 7.** Sistemas inteligentes

7.1. Teoria dos agentes
- 7.1.1. História do conceito
- 7.1.2. Definição de agente
- 7.1.3. Agentes em inteligência artificial
- 7.1.4. Agentes em engenharia de *software*

7.2. Arquiteturas de agentes
- 7.2.1. O processo de raciocínio de um agente
- 7.2.2. Agentes reativos
- 7.2.3. Agentes dedutivos
- 7.2.4. Agentes híbridos
- 7.2.5. Comparativa

7.3. Informação e conhecimento
- 7.3.1. Distinção entre dados, informação e conhecimento
- 7.3.2. Avaliação da qualidade dos dados
- 7.3.3. Métodos de recolha de dados
- 7.3.4. Métodos de aquisição de dados
- 7.3.5. Métodos de aquisição de conhecimentos

7.4. Representação do conhecimento
- 7.4.1. A importância da representação do conhecimento
- 7.4.2. Definição da representação do conhecimento através dos seus papéis
- 7.4.3. Características de uma representação do conhecimento

7.5. Ontologias
- 7.5.1. Introdução aos metadados
- 7.5.2. Conceito filosófico de ontologia
- 7.5.3. Conceito computacional de ontologia
- 7.5.4. Ontologias de domínio e ontologias de nível superior
- 7.5.5. Como construir uma ontologia?

7.6. Linguagens ontológicas e *software* para criação de ontologias
- 7.6.1. Tripletas RDF, Turtle e N3
- 7.6.2. Esquema RDF
- 7.6.3. OWL
- 7.6.4. SPARQL
- 7.6.5. Introdução às diferentes ferramentas para a criação de ontologias
- 7.6.6. Instalação e utilização do Protégé

7.7. A Web Semântica
- 7.7.1. O estado atual e o futuro da Web Semântica
- 7.7.2. Aplicações da Web Semântica

7.8. Outros modelos de representação do conhecimento
- 7.8.1. Vocabulários
- 7.8.2. Visão geral
- 7.8.3. Taxonomias
- 7.8.4. Thesaurus
- 7.8.5. Folksonomias
- 7.8.6. Comparativo
- 7.8.7. Mapas mentais

7.9. Avaliação e integração de representações do conhecimento
- 7.9.1. Lógica de ordem zero
- 7.9.2. Lógica de primeira ordem
- 7.9.3. Lógica descritiva
- 7.9.4. Relação entre diferentes tipos de lógica
- 7.9.5. Prolog: programação baseada em lógica de primeira ordem

7.10. Razoadores semânticos, sistemas saseados no conhecimento e sistemas especializados
- 7.10.1. Conceito de raciocinador
- 7.10.2. Aplicações de um raciocinador
- 7.10.3. Sistemas baseados no conhecimento
- 7.10.4. MYCIN, história dos sistemas de especialistas
- 7.10.5. Elementos e arquitetura de sistemas especialistas
- 7.10.6. Criação de sistemas de especialistas

## **Módulo 8.** Programação em tempo real

8.1. Conceitos básicos de programação concorrente
- 8.1.1. Conceitos fundamentais
- 8.1.2. Concorrência
- 8.1.3. Benefícios da concorrência
- 8.1.4. Concorrência e hardware

8.2. Estruturas básicas de apoio concorrencial em Java
- 8.2.1. Concorrência em *Threads*
- 8.2.2. Criação de roscas
- 8.2.3. Métodos
- 8.2.4. Sincronização

8.3. *Threads*, ciclo de vida, prioridades, interrupções, estados, executores
- 8.3.1. *Tópicos*
- 8.3.2. Ciclo de vida
- 8.3.3. Prioridades
- 8.3.4. Interrupções
- 8.3.5. Estados
- 8.3.6. Executores

8.4. Exclusão mútua
- 8.4.1. O que é a exclusão mútua?
- 8.4.2. Algoritmo de Dekker
- 8.4.3. Algoritmo de Peterson
- 8.4.4. Exclusão mútua em Java

8.5. Dependências de estados
- 8.5.1. Injeção de dependência
- 8.5.2. Implementação do padrão em Java
- 8.5.3. Formas de injetar dependências
- 8.5.4. Exemplos

8.6. Padrões de *design*
- 8.6.1. Introdução
- 8.6.2. Padrões de criação
- 8.6.3. Padrões de estrutura
- 8.6.4. Padrões de comportamento

8.7. Utilização de Bibliotecas Java
- 8.7.1. O que são Bibliotecas Java?
- 8.7.2. *Mockito-All*, *Mockito-Core*
- 8.7.3. Guava
- 8.7.4. Commons-io
- 8.7.5. Commons-lang, commons-lang3

8.8. Programação de *Shaders*
- 8.8.1. Pipeline 3D e raster
- 8.8.2. Vertex Shading
- 8.8.3. Pixel *Shading: Iluminação I*
- 8.8.4. Pixel *Shading: Iluminação II*
- 8.8.5. Pós-efeitos

8.9. Programação em tempo real
- 8.9.1. Introdução
- 8.9.2. Processamento deinterrupções
- 8.9.3. Sincronização e comunicação entre processos
- 8.9.4. Sistemas de programação em tempo real

8.10. Planeamento em tempo real
- 8.10.1. Conceitos
- 8.10.2. Modelo de referência para sistemas em tempo real
- 8.10.3. Políticas de planeamento
- 8.10.4. Programadores cíclicos
- 8.10.5. Planeadores com propriedades estáticas
- 8.10.6. Planeadores com propriedades dinâmicas

## **Módulo 9.** Conceção e desenvolvimento de jogos web

9.1. Origens e normas da Web
- 9.1.1. Origens da Internet
- 9.1.2. Criação da *World Wide Web*
- 9.1.3. Emergência de normas web
- 9.1.4. O aumento dos padrões da web

9.2. HTTP e estrutura cliente-servidor
- 9.2.1. Papel cliente-servidor
- 9.2.2. Comunicação cliente-servidor
- 9.2.3. História recente
- 9.2.4. Computação centralizada

9.3. Programação web: introdução
- 9.3.1. Conceitos básicos
- 9.3.2. Criação de um servidor web
- 9.3.3. Noções básicas de HTML5
- 9.3.4. Formulários HTML

9.4. Introdução ao HTML e exemplos
- 9.4.1. História do HTML5
- 9.4.2. Elementos de HTML5
- 9.4.3. APIS
- 9.4.4. CCS3

9.5. Modelo de Objeto de Documento
- 9.5.1. O que é o Modelo de Objeto de Documento?
- 9.5.2. Utilização de DOCTYPE
- 9.5.3. A importância de validar o HTML
- 9.5.4. Acedendo aos elementos
- 9.5.5. Criação de elementos e texto
- 9.5.6. Utilização de innerHTML
- 9.5.7. Eliminação de um elemento ou nó de texto
- 9.5.8. Leitura e escrita dos atributos de um elemento
- 9.5.9. Manipulando os estilos de elementos
- 9.5.10. Anexar vários ficheiros ao mesmo tempo

9.6. Introdução ao CSS e exemplos
- 9.6.1. Sintaxe CSS3
- 9.6.2. Folhas de Estilo
- 9.6.3. Rótulos
- 9.6.4. Seletores
- 9.6.5. Web design com CSS

9.7. Introdução ao JavaScript e exemplos
- 9.7.1. O que é JavaScript?
- 9.7.2. Breve história da linguagem
- 9.7.3. Versões de JavaScript
- 9.7.4. Exibição de uma caixa de diálogo
- 9.7.5. Sintaxe de JavaScript
- 9.7.6. Compreender os Scripts
- 9.7.7. Espaços
- 9.7.8. Comentários
- 9.7.9. Funções
- 9.7.10. JavaScript interno e externo

9.8. Funções em JavaScript
- 9.8.1. Declarações de funções
- 9.8.2. Expressões de funções
- 9.8.3. Chamar funções
- 9.8.4. Recursividade
- 9.8.5. Funções aninhadas e encerramentos
- 9.8.6. Preservação de variáveis
- 9.8.7. Funções multi-aninhadas
- 9.8.8. Conflito de nombes
- 9.8.9. Clausuras ou encerramentos
- 9.8.10. Parâmetros de uma função

9.9. PlayCanvas para o desenvolvimento de jogos web
- 9.9.1. O que é a PlayCanvas?
- 9.9.2. Configuração do projeto
- 9.9.3. Criação de um objeto
- 9.9.4. Acrescentando físicas
- 9.9.5. Acrescentando um modelo
- 9.9.6. Mudando as configurações de gravidade e da cena
- 9.9.7. Executando Scripts
- 9.9.8. Controlos da câmara

9.10. Phaser para desenvolvimento de jogos web
- 9.10.1. O que é o Phaser?
- 9.10.2. Carregamento de recursos
- 9.10.3. Construir o mundo
- 9.10.4. Plataformas
- 9.10.5. O Jogador
- 9.10.6. Acrescentar a física
- 9.10.7. Utilização do teclado
- 9.10.8. Recolha de *Pick-ups*
- 9.10.9. Pontos e Pontuação
- 9.10.10. Bombas de ressalto

## **Módulo 10.** Redes e sistemas multijogador

10.1. História e evolução dos videojogos para multijogadores
- 10.1.1. Anos 70: primeiros jogos para multijogadores
- 10.1.2. 1990s: Duque Nukem, Doom, Quake
- 10.1.3. Ascensão dos videojogos para multijogadores
- 10.1.4. Multijogador local e *online*
- 10.1.5. Jogos de festa

10.2. Modelos de negócio multijogadores
- 10.2.1. Origem e funcionamento dos modelos empresariais emergentes
- 10.2.2. Serviços de vendas *online*
- 10.2.3. Livre para jogar
- 10.2.4. Micropagamentos
- 10.2.5. Publicidade
- 10.2.6. Assinatura com pagamentos mensais
- 10.2.7. Pay-per-play
- 10.2.8. Experimente antes de comprar

10.3. Jogos locais e jogos em rede
- 10.3.1. Jogos locais: inícios
- 10.3.2. Jogos de festa: A Nintendo e a união da família
- 10.3.3. Jogos em rede: inícios
- 10.3.4. Evolução dos jogos em rede

10.4. Modelo OSI: Camadas I
- 10.4.1. Modelo OSI: introdução
- 10.4.2. Camada física
- 10.4.3. Camada de ligação de dados
- 10.4.4. Camada de rede

10.5. Modelo OSI: Camadas II
- 10.5.1. Camada de transporte
- 10.5.2. Camada da sessão
- 10.5.3. Camada de apresentação
- 10.5.4. Camada de aplicação

10.6. Redes informáticas e a internet
- 10.6.1. O que é uma rede informática?
- 10.6.2. *Software*
- 10.6.3. *Hardware*
- 10.6.4. Servidores
- 10.6.5. Armazenamento em rede
- 10.6.6. Protocolos de rede

10.7. Redes móveis e sem fios
- 10.7.1. Rede móvel
- 10.7.2. Rede sem fios
- 10.7.3. Funcionamento de redes móveis
- 10.7.4. Tecnologia digital

10.8. Segurança
- 10.8.1. Segurança Pessoal
- 10.8.2. *Hacks e Cheats* em videojogos
- 10.8.3. Segurança anti-cheating
- 10.8.4. Análise de sistemas de segurança anti-cheating

10.9. Sistemas multijogador: servidores
- 10.9.1. Alojamento de servidores
- 10.9.2. Videojogos MMO
- 10.9.3. Servidores de videojogos dedicados
- 10.9.4. LAN Parties

10.10. Design e programação de videojogos para multijogadores
- 10.10.1. Fundamentos do design de videojogos para multijogadores em Unreal
- 10.10.2. Fundamentos do design de videojogos para multijogadores em Unity
- 10.10.3. Como tornar um jogo para multijogadores divertido
- 10.10.4. Para além de um comando: inovação em comandos para multijogadores

*Aproveite esta oportunidade para adquirir conhecimentos sobre os últimos desenvolvimentos na área e aplicá-los na sua atividade diária"*

05

# Metodologia

Este programa de capacitação oferece uma forma diferente de aprendizagem.
A nossa metodologia é desenvolvida através de um modo de aprendizagem cíclico: ***o Relearning***.
Este sistema de ensino é utilizado, por exemplo, nas escolas médicas mais prestigiadas do mundo e tem sido considerado um dos mais eficazes pelas principais publicações, tais como a ***New England Journal of Medicine.***

*Descubra o Relearning, um sistema que abandona a aprendizagem linear convencional para o levar através de sistemas de ensino cíclicos: uma forma de aprendizagem que provou ser extremamente eficaz, especialmente em disciplinas que requerem memorização"*

## Estudo de Caso para contextualizar todo o conteúdo

O nosso programa oferece um método revolucionário de desenvolvimento de competências e conhecimentos. O nosso objetivo é reforçar as competências num contexto de mudança, competitivo e altamente exigente.

*Com a TECH pode experimentar uma forma de aprendizagem que abala as fundações das universidades tradicionais de todo o mundo"*


*Terá acesso a um sistema de aprendizagem baseado na repetição, com ensino natural e progressivo ao longo de todo o programa de estudos.*

*O estudante aprenderá, através de atividades de colaboração e casos reais, a resolução de situações complexas em ambientes empresariais reais.*

## Um método de aprendizagem inovador e diferente

Este programa da TECH é um programa de ensino intensivo, criado de raiz, que propõe os desafios e decisões mais exigentes neste campo, tanto a nível nacional como internacional. Graças a esta metodologia, o crescimento pessoal e profissional é impulsionado, dando um passo decisivo para o sucesso. O método do caso, a técnica que constitui a base deste conteúdo, assegura que a realidade económica, social e profissional mais atual é seguida.

> *O nosso programa prepara-o para enfrentar novos desafios em ambientes incertos e alcançar o sucesso na sua carreira"*

O método do caso tem sido o sistema de aprendizagem mais amplamente utilizado pelas melhores faculdades do mundo. Desenvolvido em 1912 para que os estudantes de direito não só aprendessem o direito com base no conteúdo teórico, o método do caso consistia em apresentar-lhes situações verdadeiramente complexas, a fim de tomarem decisões informadas e valorizarem juízos sobre a forma de as resolver. Em 1924 foi estabelecido como um método de ensino padrão em Harvard.

Numa dada situação, o que deve fazer um profissional? Esta é a questão que enfrentamos no método do caso, um método de aprendizagem orientado para a ação. Ao longo de 4 anos, será confrontado com múltiplos casos reais. Terão de integrar todo o seu conhecimento, investigar, argumentar e defender as suas ideias e decisões.

## Relearning Methodology

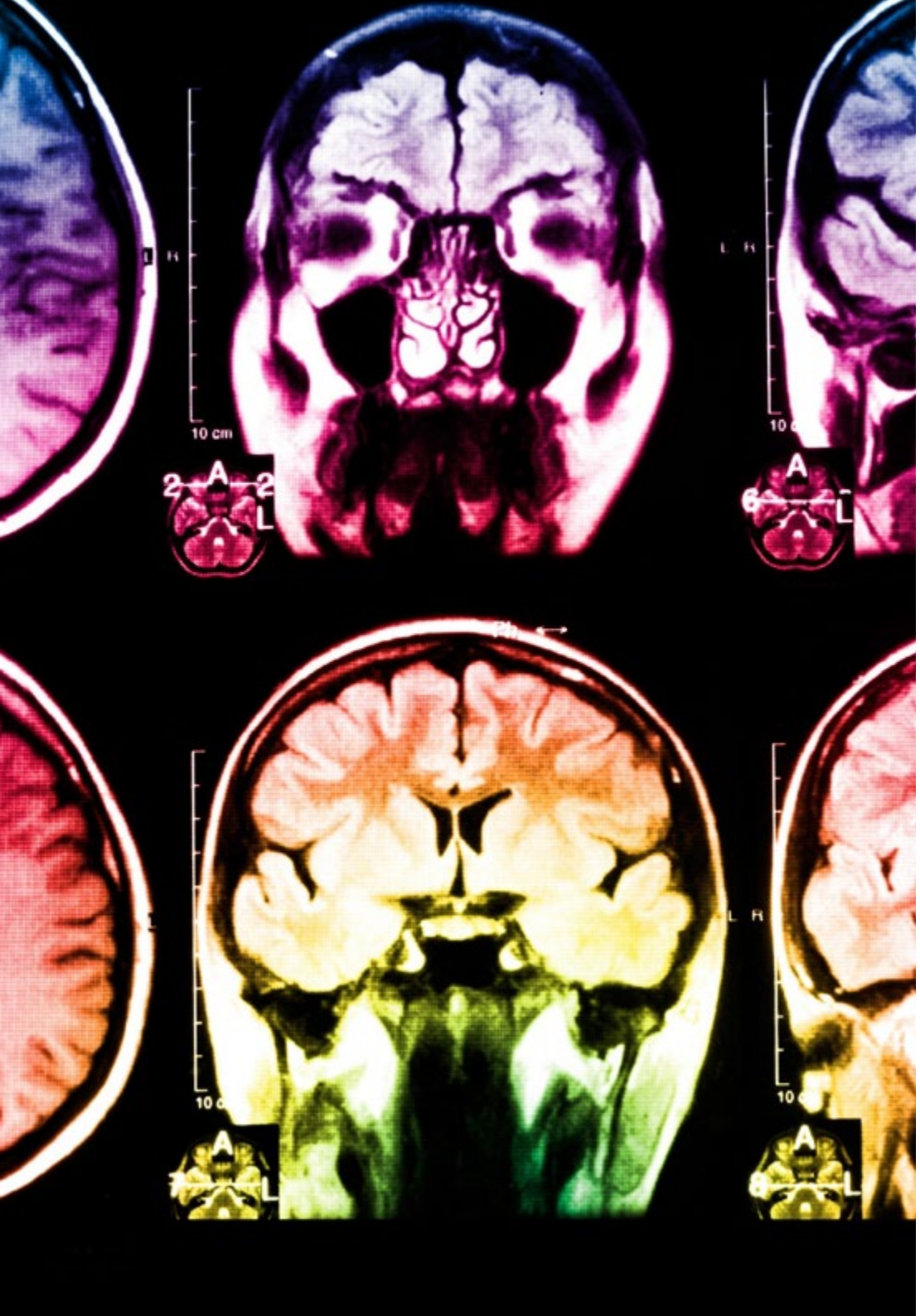
A TECH combina eficazmente a metodologia do Estudo de Caso com um sistema de aprendizagem 100% online baseado na repetição, que combina 8 elementos didáticos diferentes em cada lição.

Melhoramos o Estudo de Caso com o melhor método de ensino 100% online: o Relearning.

*Em 2019 obtivemos os melhores resultados de aprendizagem de todas as universidades online do mundo.*

Na TECH aprende- com uma metodologia de vanguarda concebida para formar os gestores do futuro. Este método, na vanguarda da pedagogia mundial, chama-se Relearning.

A nossa universidade é a única universidade de língua espanhola licenciada para utilizar este método de sucesso. Em 2019, conseguimos melhorar os níveis globais de satisfação dos nossos estudantes (qualidade de ensino, qualidade dos materiais, estrutura dos cursos, objetivos...) no que diz respeito aos indicadores da melhor universidade online do mundo.



No nosso programa, a aprendizagem não é um processo linear, mas acontece numa espiral (aprender, desaprender, esquecer e reaprender). Portanto, cada um destes elementos é combinado de forma concêntrica. Esta metodologia formou mais de 650.000 licenciados com sucesso sem precedentes em áreas tão diversas como a bioquímica, genética, cirurgia, direito internacional, capacidades de gestão, ciência do desporto, filosofia, direito, engenharia, jornalismo, história, mercados e instrumentos financeiros. Tudo isto num ambiente altamente exigente, com um corpo estudantil universitário com um elevado perfil socioeconómico e uma idade média de 43,5 anos.

*O Relearning permitir-lhe-á aprender com menos esforço e mais desempenho, envolvendo-o mais na sua capacitação, desenvolvendo um espírito crítico, defendendo argumentos e opiniões contrastantes: uma equação direta ao sucesso.*

A partir das últimas provas científicas no campo da neurociência, não só sabemos como organizar informação, ideias, imagens e memórias, mas sabemos que o lugar e o contexto em que aprendemos algo é fundamental para a nossa capacidade de o recordar e armazenar no hipocampo, para o reter na nossa memória a longo prazo.

Desta forma, e no que se chama Neurocognitive context-dependent e-learning, os diferentes elementos do nosso programa estão ligados ao contexto em que o participante desenvolve a sua prática profissional.

**Este programa oferece o melhor material educativo, cuidadosamente preparado para profissionais:**

### Material de estudo

Todos os conteúdos didáticos são criados pelos especialistas que irão ensinar o curso, especificamente para o curso, para que o desenvolvimento didático seja realmente específico e concreto.

Estes conteúdos são depois aplicados ao formato audiovisual, para criar o método de trabalho online da TECH. Tudo isto, com as mais recentes técnicas que oferecem peças de alta-qualidade em cada um dos materiais que são colocados à disposição do aluno.

### Masterclasses

Existem provas científicas sobre a utilidade da observação por terceiros especializados.

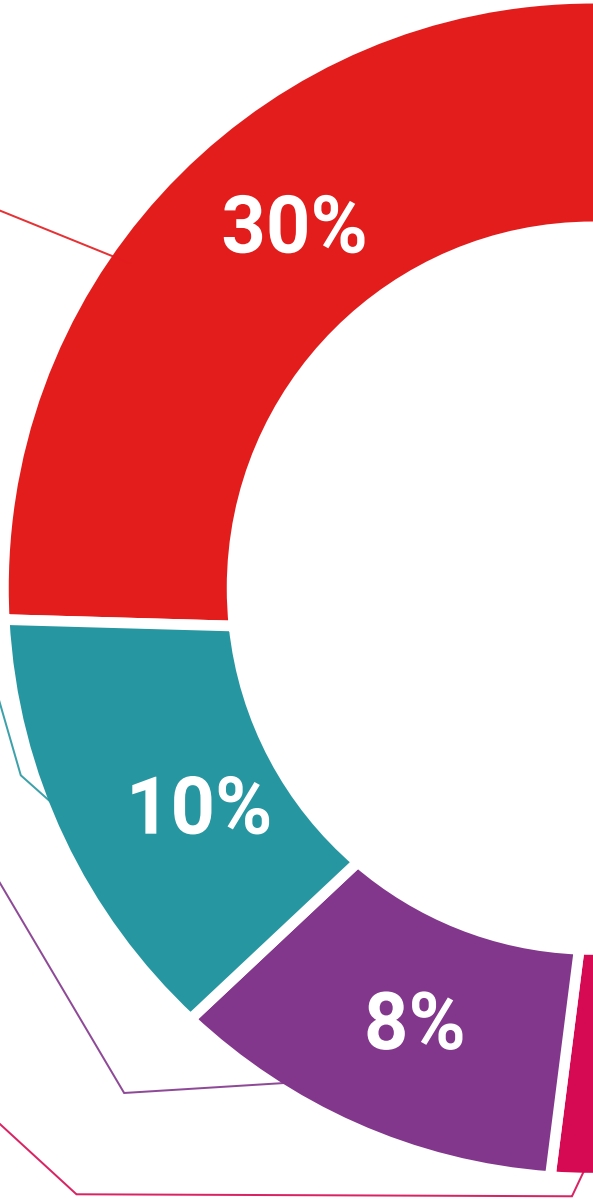
O denominado Learning from an Expert constrói conhecimento e memória, e gera confiança em futuras decisões difíceis.

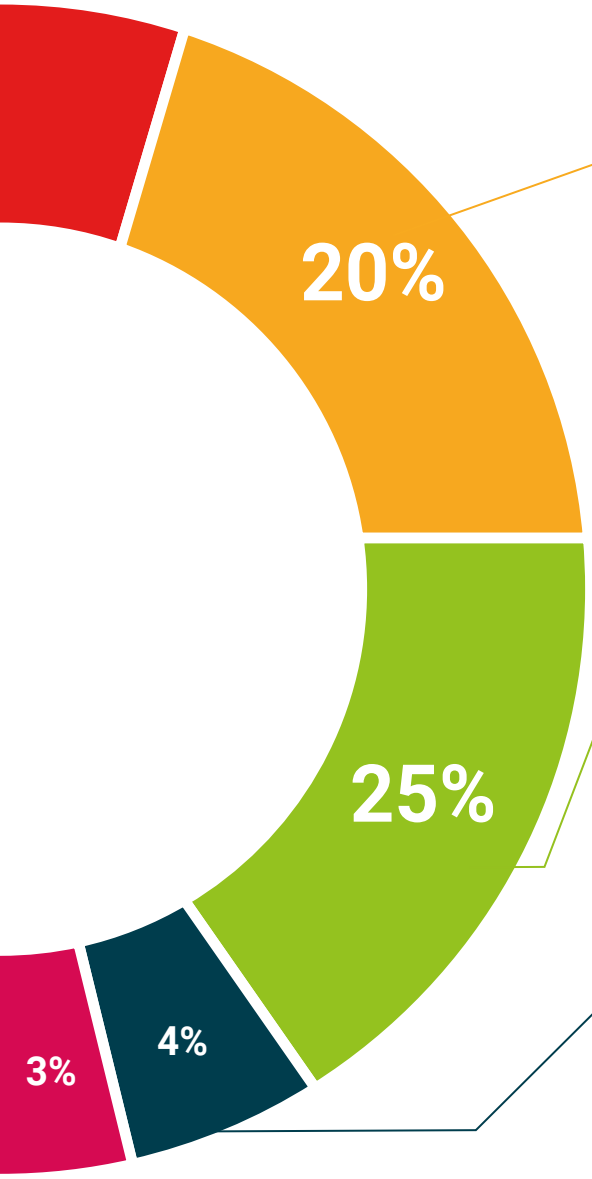
### Práticas de aptidões e competências

Realizarão atividades para desenvolver competências e aptidões específicas em cada área temática. Práticas e dinâmicas para adquirir e desenvolver as competências e capacidades que um especialista necessita de desenvolver no quadro da globalização em que vivemos.

### Leituras complementares

Artigos recentes, documentos de consenso e diretrizes internacionais, entre outros. Na biblioteca virtual da TECH o aluno terá acesso a tudo o que necessita para completar a sua capacitação






### Case studies

Completarão uma seleção dos melhores estudos de casos escolhidos especificamente para esta situação. Casos apresentados, analisados e instruídos pelos melhores especialistas na cena internacional.

### Resumos interativos

A equipa da TECH apresenta os conteúdos de uma forma atrativa e dinâmica em comprimidos multimédia que incluem áudios, vídeos, imagens, diagramas e mapas concetuais a fim de reforçar o conhecimento.

Este sistema educativo único para a apresentação de conteúdos multimédia foi premiado pela Microsoft como uma "História de Sucesso Europeu"

### Testing & Retesting

Os conhecimentos do aluno são periodicamente avaliados e reavaliados ao longo de todo o programa, através de atividades e exercícios de avaliação e auto-avaliação, para que o aluno possa verificar como está a atingir os seus objetivos.

06

# Certificação

O Mestrado Próprio em Programação de Videojogos garante, para além de um conteúdo mais rigoroso e atualizado, o acesso a um grau de Mestre emitido pela TECH Universidade Tecnológica.

*Conclua este plano de estudos com sucesso e receba o seu certificado sem sair de casa e sem burocracias"*

Este **Mestrado Próprio em Programação de Videojogos** conta com o conteúdo educacional mais completo e atualizado do mercado.

Uma vez aprovadas as avaliações, o aluno receberá por correio*, com aviso de receção, o certificado correspondente ao título de **Mestrado Próprio** emitido pela **TECH Universidade Tecnológica.**

O certificado emitido pela **TECH Universidade Tecnológica** expressará a qualificação obtida no Mestrado Próprio, atendendo aos requisitos normalmente exigidos pelas bolsas de emprego, concursos públicos e avaliação de carreiras profissionais.

Título: **Mestrado Próprio em Programação de Videojogos**

ECTS: **60**

Carga horária: **1500 horas**

tech universidade tecnológica

Concede o presente

CERTIFICADO

a

Sr./Sra. _____, com o documento de identificação nº_____

Por ter concluído e acreditado com sucesso o

**MESTRADO PRÓPRIO**

em

Programação de Videojogos

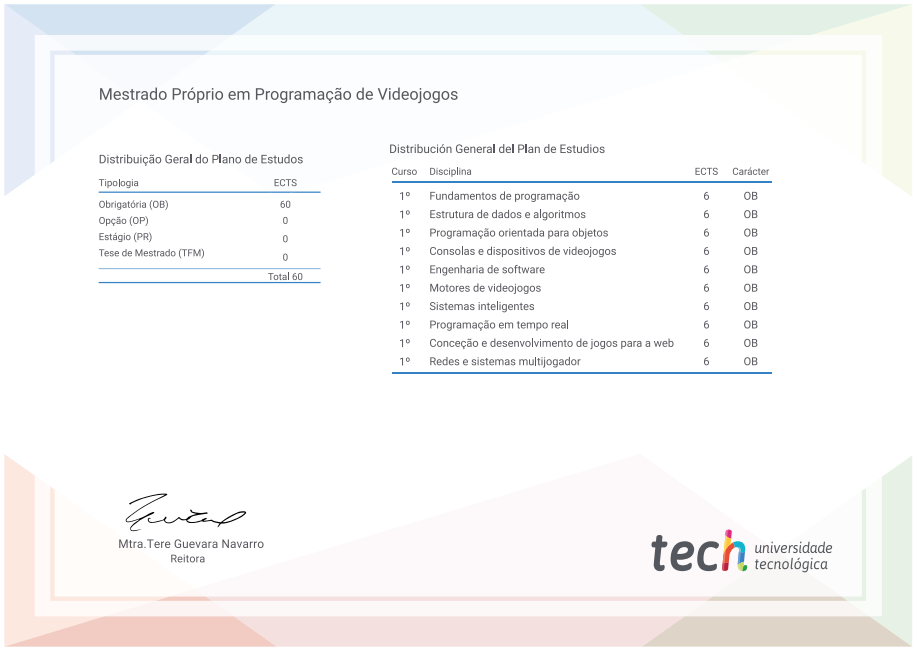
Este é um certificado atribuído por esta Universidade, reconhecido por 60 ECTS e equivalente a 1.500 horas, com data de início a dd/mm/aaaaa e data de conclusão a dd/mm/aaaaa.

A TECH é uma Instituição Privada de Ensino Superior reconhecida pelo Ministério da Educação Pública a partir de 28 de junho de 2018.

A 17 de junho de 2020

Prof.Tere Guevara Navarro
Reitora

código único TECH: AFWOR23S techtitute.com/titulos



Mestrado Próprio em Programação de Videojogos

Distribuição Geral do Plano de Estudos

| Tipologia | ECTS |
|---|---|
| Obrigatória (OB) | 60 |
| Opção (OP) | 0 |
| Estágio (PR) | 0 |
| Tese de Mestrado (TFM) | 0 |
| | Total 60 |

Distribución General del Plan de Estudios

| Curso | Disciplina | ECTS | Carácter |
|---|---|---|---|
| 1º | Fundamentos de programação | 6 | OB |
| 1º | Estrutura de dados e algoritmos | 6 | OB |
| 1º | Programação orientada para objetos | 6 | OB |
| 1º | Consolas e dispositivos de videojogos | 6 | OB |
| 1º | Engenharia de software | 6 | OB |
| 1º | Motores de videojogos | 6 | OB |
| 1º | Sistemas inteligentes | 6 | OB |
| 1º | Programação em tempo real | 6 | OB |
| 1º | Conceção e desenvolvimento de jogos para a web | 6 | OB |
| 1º | Redes e sistemas multijogador | 6 | OB |

Mtra.Tere Guevara Navarro
Reitora

tech universidade tecnológica

*Caso o aluno solicite que o seu certificado seja apostilado, a TECH EDUCATION providenciará a obtenção do mesmo a um custo adicional.

tech universidade tecnológica

# Mestrado Próprio
## Programação de Videojogos

- Modalidade: **online**
- Duração: **12 meses**
- Certificação: **TECH Universidade Tecnológica**
- Créditos: **60 ECTS**
- Tempo Dedicado: **16 horas/semana**
- Horário: **ao seu próprio ritmo**
- Exames: **online**

# Mestrado Próprio
## Programação de Videojogos

tech universidade tecnológica