Professional Master's Degree
Video Game Programming

tech technological university

tech technological university

Professional Master's Degree

Video Game Programming

- Modality: **online**
- Duration: **12 months**
- Certificate: **TECH Technological University**
- Dedication: **16h/week**
- Schedule: **at your own pace**
- Exams: **online**

Website: www.techtitute.com/us/information-technology/professional-master-degree/master-video-game-programming

Index

01

Introduction

A video game's greatest appeal lies in its most visual aspects, such as graphics or design. But they could not stand out without programming. Programming is the key to any videogame, since it determines playability or how the graphics interact with the player. Without good programming, any game would fail, as it would have numerous bugs and would not be an overall enjoyable experience. Companies are aware of this, which is why they need high-level developers. This degree responds to this demand, as it prepares students to be able to face all the challenges in the industry, so they will obtain numerous professional opportunities thanks to it.

Companies know that the key to a video game is in the programming. Specialize and become the most sought-after developer in your work environment"

Behind every great video game, there is a huge team of professionals specialized in each area of work that will try to bring success to their company. Normally, the most outstanding sections for fans are those they can perceive directly, such as the Visuals or Character Control, Mechanics or Object Interaction.

However, for all these elements to work and be correctly integrated, there is an essential task that is not usually taken into account: programming. The development of a video game has different phases and involves different departments, but programming is what makes sense of everything and forms the basic skeleton on which the rest of the elements are incorporated.

That is why companies in the industry pay so much attention to this issue, since they know correctly and efficiently developing video games will facilitate project progress and avoid potential errors and Bugs. To that end, they look for the best programmers specialized in this area,

But it is not easy to find true specialists in the field. This Professional Master's Degree in Video Game Programming responds to that demand, making students become great experts in video game development that can thrive in the industry with ease, obtaining great career opportunities thanks to the skills and abilities they will acquire throughout this degree.

This **Professional Master's Degree in Video Game Programming** contains the most complete and up-to-date scientific program on the market. Its most notable features are:

- Practical cases presented by experts in video game programming and development
- The graphic, schematic, and eminently practical contents with which they are created, provide scientific and practical information on the disciplines that are essential for professional practice
- Practical exercises where self-assessment can be used to improve learning
- Its special emphasis on innovative methodologies
- Theoretical lessons, questions to the expert, debate forums on controversial topics, and individual reflection assignments
- Content that is accessible from any fixed or portable device with an Internet connection

Develop all types of video games in the best companies in the world thanks to this Professional Master's Degree"

Programming is increasingly essential in developing video games. Become an essential part of the industry thanks to this degree"

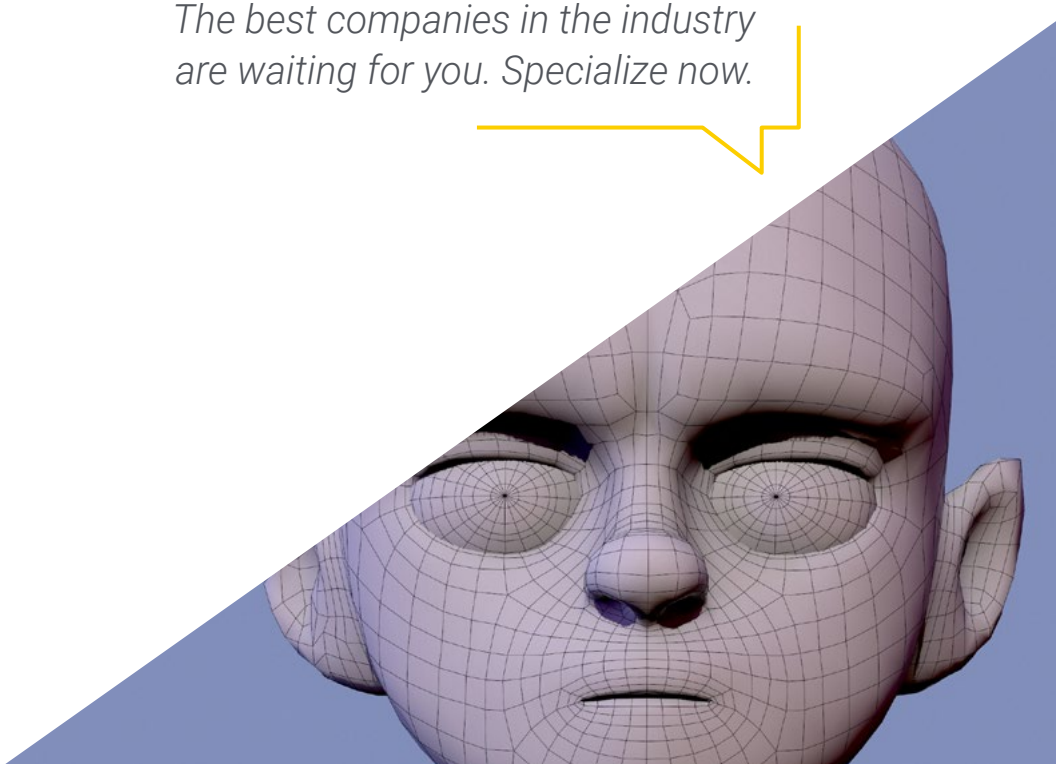
Games are your passion, and you want to become a great developer. Don't wait any longer and enroll in this Professional Master's Degree.

The program's teaching staff includes professionals from the sector who contribute their work experience to this training program, as well as renowned specialists from leading societies and prestigious universities.

The multimedia content, developed with the latest educational technology, will provide the professional with situated and contextual learning, i.e., a simulated environment that will provide immersive training programmed to train in real situations.

This program is designed around Problem-Based Learning, whereby the professional must try to solve the different professional practice situations that arise during the academic year. For this purpose, the student will be assisted by an innovative interactive video system created by renowned and experienced experts.

The best companies in the industry are waiting for you. Specialize now.



02

Objectives

The main goal of this Professional Master's Degree is to turn students into great video game developers. This industry is expanding and increasingly needs more programmers and more specialists with high-level training, so this degree is perfect for great career opportunities in some of the most prestigious companies in the world. Thus, this program offers students all the necessary skills to become highly sought-after experts in the field, achieving a significant and immediate career advancement for them.

All your dreams are now within your reach thanks to this Professional Master's Degree in Video Game Programming"

General Objectives

- Become familiar the different Programming Languages and Methods applied to Video Games
- Delve deeper into the Video Game Production Process and Integrate Programming throughout each stage
- Learn the Fundamentals of Video Game Design and the theoretical knowledge that a Video Game Designer must have
- Master the Basic Programming Languages used in Video Games
- Apply knowledge of Software Engineering and Specialized Programming to Video Games
- Understand the role of Programming in Video Game Development
- Know the different existing Consoles and Platforms
- Develop Web and Multiplayer Video Games

When you finish this Professional Master's Degree, you will be the best Video Game Developer in your environment"

Specific Objectives

Module 1. Programming Fundamentals

- Understand the Basic Structure of Computers, Software and the general purpose Programming Languages
- Analyze the essential elements of a Computer Program, such as the different Data Types, Operators, Expressions, Statements, I/O and Control Statements
- Interpret Algorithms as the necessary basis to develop Computer Programs

Module 2. Data Structure and Algorithms

- Learn the main Algorithm Design Strategies, as well as the different Methods and Measures for Algorithm Computation
- Understand Algorithm Function, strategies and examples for the most common problems
- Understand the *Backtracking* Technique and its main uses

Module 3. Object Oriented Programming

- Know the different Design Patterns for Object Oriented problems
- Understand the importance of Documentation and Testing in Software Development
- Manage the use of Threading and Synchronization, and solve common problems in Concurrent Programming

Module 4. Video Game Consoles and Devices

- Know the Basic Operation of the main Input and Output Peripherals
- Understand the main Design implications on different Platforms
- Study the Structure, Organization, Operation and Interconnection of Devices and Systems
- Understand the Role of Operating Systems and Development Kits for Mobile Devices and Video Game Platforms

Module 5. Software Engineering

- Become familiar with the Bases of Software Engineering, as well as the Software Process and the different Development Models, including Agile Technologies
- Recognize requirements engineering, its development, elaboration, negotiation and validation in order to understand the Main Standards in terms of Software Quality and Project Management

Module 6. Video Game Engines

- Discover Video Game Engine Operation and Architecture
- Understand the Basic Features of existing Game Engines
- Correctly and efficiently program applications applied to Video Game Engines
- Choose the most appropriate paradigm and programming languages to program applications applied to Video Game Engines

Module 7. Intelligent Systems

- Establish Agent Theory concepts, Agent Architecture and the Reasoning Process behind it
- Assimilate the Theory and Practice behind the Concepts of Information and Knowledge, as well as the different ways of Representing Knowledge
- Understand the functioning of Semantic Reasoners, Knowledge-Based Systems and Expert Systems

Module 8. Real-Time Programming

- Analyze the key features of Real-Time Programming Languages that differentiate them from traditional programming languages
- Understand the basic concepts behind Computer Systems
- Acquire the ability to apply the main Bases and Techniques of Real-Time Programming

Module 9. Web Game Design and Development

- Design Games and Interactive Web Applications with the corresponding Documentation
- Evaluate the main features of Games and Interactive Web Applications for professional and adequate communication

Module 10. Multiplayer Networks and Systems

- Describe the Transmission Control Protocol/Internet Protocol (TCP/IP) Architecture and the Basic Operation of Wireless Networks
- Analyze Video Games Security
- Develop Multiplayer Online Games

03

Skills

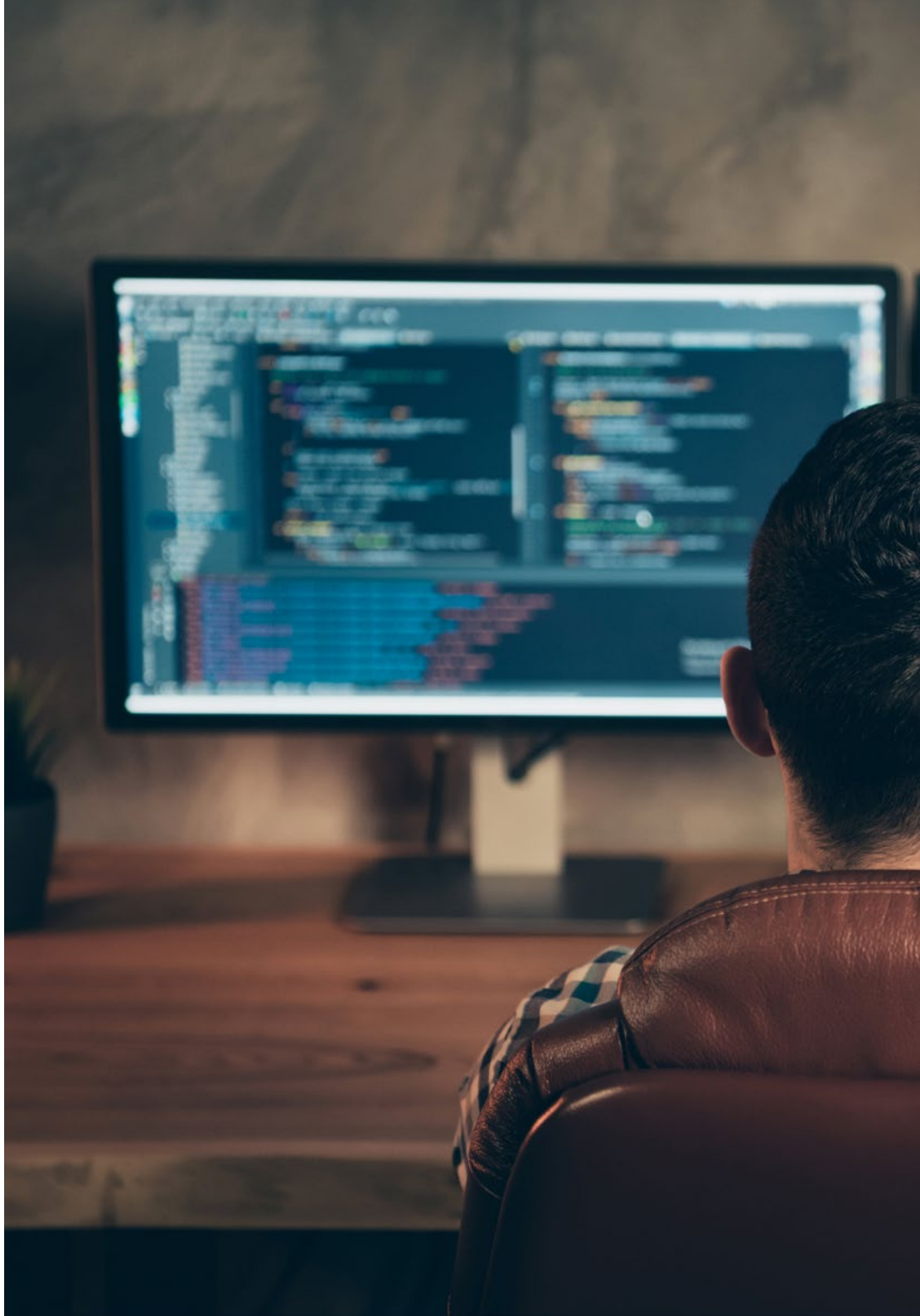
This Professional Master's Degree in Video Game Programming turns students into true specialists in developing this type of audiovisual projects thanks to the expertise and skills it provides them with. Thus, thanks to this excellent program, students will obtain a series of Professional Tools to face any kind of challenge in Video Game Programming, becoming essential employees for their companies in the process.

You will master everything there is to know about Video Game Development"

General Skills

- Design all phases of a Video Game, from the initial idea to Final Release
- Specialize as a Video Game Programmer
- Delve deeper into all the Developmental Stages, from the Initial Architecture and Player Character Programming to every Element involved in the Game Process
- Obtain an overall vision of projects, being able to provide solutions to the different problems and challenges that arise in video game design

Achieve excellence as a Video Game Programmer thanks to this Professional Master's Degree"



Specific Skills

- Know the necessary Software to be a Professional Video Game Developer
- Understand Player Experience and analyze Video Game Gameplay
- Understand all the Theoretical and Practical Procedures involved in the Video Game Programming Process
- Master the most useful Programming Languages for the Video Game World
- Integrate the Programming learned to different types of Consoles and Platforms
- Program Web and Multiplayer Video Games
- Assimilate the concept of Video Game Engine for correct Programming
- Apply knowledge of Software Engineering to Video Game Programming

04

Structure and Content

The contents of this Professional Master's Degree in Video Game Programming have been carefully designed by a team of great specialists in the field who are perfectly aware of the current state of the industry. Thus, thanks to this program, students will be able to learn all the necessary knowledge to respond to company demands in the field, since they will have been trained in the particulars and specifics of these demands, which are complex and constantly changing.

These contents will make you a great expert in Video Game Programming"

Module 1. Programming Fundamentals

1.1. Introduction to Programming
- 1.1.1. Basic Computer Structure
- 1.1.2. Software
- 1.1.3. Programming Languages
- 1.1.4. Computer Application Life Cycle

1.2. Algorithm Design
- 1.2.1. Problem Solving
- 1.2.2. Descriptive Techniques
- 1.2.3. Algorithm Elements and Structure

1.3. Program Elements
- 1.3.1. C++ Origin and Features
- 1.3.2. Development Environment
- 1.3.3. Concept of Program
- 1.3.4. Types of Fundamental Data
- 1.3.5. Operators
- 1.3.6. Expressions
- 1.3.7. Statements
- 1.3.8. Data Input and Output

1.4. Control Statements
- 1.4.1. Statements
- 1.4.2. Branches
- 1.4.3. Loops

1.5. Abstraction and Modularity: Functions
- 1.5.1. Modular Design
- 1.5.2. Concept of Function and Utility
- 1.5.3. Definition of Function
- 1.5.4. Execution Flow When Function Is Called
- 1.5.5. Function Prototypes
- 1.5.6. Results Return
- 1.5.7. Calling Functions: Parameters
- 1.5.8. Parameter Passing According to Reference and Value
- 1.5.9. Scope Identifier

1.6. Statistical Data Structures
- 1.6.1. Arrays
- 1.6.2. Matrices: Polyhedra
- 1.6.3. Searching and Sorting
- 1.6.4. Chaining: I/O Functions for Chains
- 1.6.5. Structures: Unions
- 1.6.6. New Types of Data

1.7. Statistical Data Structures: Pointers
- 1.7.1. Concept: Definition of Pointer
- 1.7.2. Pointer Operators and Operations
- 1.7.3. Pointer Arrays
- 1.7.4. Pointers and Arrays
- 1.7.5. Chain Pointers
- 1.7.6. Structure Pointers
- 1.7.7. Multiple Indirection
- 1.7.8. Function Pointers
- 1.7.9. Function, Structure and Array Passing as Function Parameters

1.8. Files
- 1.8.1. Basic Concepts
- 1.8.2. File Operations
- 1.8.3. Types of Files
- 1.8.4. File Organization
- 1.8.5. Introduction to C++ Files
- 1.8.6. Managing Files

1.9. Recursion
- 1.9.1. Definition of Recursion
- 1.9.2. Types of Recursion
- 1.9.3. Advantages and Inconveniences
- 1.9.4. Considerations
- 1.9.5. Recursive-Iterative Conversion
- 1.9.6. Recursion Stack

1.10. Testing and Documentation
- 1.10.1. Program Testing
- 1.10.2. White Box Testing
- 1.10.3. Black Box Testing
- 1.10.4. Testing Tools
- 1.10.5. Program Documentation

Module 2. Data Structure and Algorithms

2.1. Introduction to Algorithm Design Strategies
- 2.1.1. Recursion
- 2.1.2. Divide and Conquer
- 2.1.3. Other Strategies

2.2. Algorithm Efficiency and Analysis
- 2.2.1. Efficiency Measures
- 2.2.2. Measuring Entry Size
- 2.2.3. Measuring Execution Time
- 2.2.4. Worst, Best and Average Case
- 2.2.5. Asymptotic Notation
- 2.2.6. Mathematical Analysis Criteria for Non-Recursive Algorithms
- 2.2.7. Mathematical Analysis for Recursive Algorithms
- 2.2.8. Empirical Analysis for Algorithms

2.3. Sorting Algorithms
- 2.3.1. Concept of Sorting
- 2.3.2. Bubble Sorting
- 2.3.3. Selection Sorting
- 2.3.4. Insertion Sorting
- 2.3.5. Mixed Sorting (*merge_sort*)
- 2.3.6. Quick Sorting (*quick_sort*)

2.4. Tree Algorithms
- 2.4.1. Concept of Tree
- 2.4.2. Binary Trees
- 2.4.3. Tree Traversal
- 2.4.4. Representing Expressions
- 2.4.5. Sorted Binary Trees
- 2.4.6. Balanced Binary Trees

2.5. Algorithms Using *Heaps*
- 2.5.1. *Heaps*
- 2.5.2. The *Heapsort* Algorithm
- 2.5.3. Priority Queues

2.6. Graph Algorithms
- 2.6.1. Representation
- 2.6.2. Width Traversal
- 2.6.3. Depth Traversal
- 2.6.4. Topological Sorting

2.7. *Greedy* Algorithms
- 2.7.1. *Greedy* Strategy
- 2.7.2. *Greedy* Strategy Elements
- 2.7.3. Currency Exchange
- 2.7.4. Traveling Salesman Problem
- 2.7.5. Knapsack Problem

2.8. Minimal Pathways Search
- 2.8.1. Shortest Path Problem
- 2.8.2. Cycles and Negative Arcs
- 2.8.3. Dijkstra's Algorithm

2.9. *Greedy* Algorithms on Graphs
- 2.9.1. Minimum Spanning Tree
- 2.9.2. Prim's Algorithm
- 2.9.3. Kruskal's Algorithm
- 2.9.4. Complexity Analysis

2.10. *Backtracking*
- 2.10.1. *Backtracking*
- 2.10.2. Alternative Techniques

Module 3. Object Oriented Programming

3.1. Introduction to Object Oriented Programming
- 3.1.1. Introduction to Object Oriented Programming
- 3.1.2. Class Design
- 3.1.3. Introduction to Unified Modeling Language (UML) for Problem Modeling

3.2. Class Relations
- 3.2.1. Abstractions and Heritage
- 3.2.2. Advanced Concepts of Heritage
- 3.2.3. Polymorphism
- 3.2.4. Composition and Aggregation

3.3. Introduction to Design Patterns for Object Oriented problems
- 3.3.1. What Are Design Patterns?
- 3.3.2. Factory Pattern
- 3.3.4. Singleton Pattern
- 3.3.5. Observer Pattern
- 3.3.6. Composite Pattern

3.4. Exceptions
- 3.4.1. What Are Exceptions?
- 3.4.2. Catching and Handling Exceptions
- 3.4.3. Launching Exceptions
- 3.4.4. Creating Exceptions

3.5. User Interface
- 3.5.1. Introduction to Qt
- 3.5.2. Positioning
- 3.5.3. What Are Events?
- 3.5.4. Events: Definition and Catching
- 3.5.5. User Interface Development

3.6. Introduction to Concurrent Programming
- 3.6.1. Introduction to Concurrent Programming
- 3.6.2. Concept of Process and Thread
- 3.6.3. Process and Thread Interaction
- 3.6.4. C++ Threads
- 3.6.5. Advantages and Disadvantages of Concurrent Programming

3.7. Thread Management and Synchronization
- 3.7.1. Thread Life Cycle
- 3.7.2. *Thread* Class
- 3.7.3. Thread Planning
- 3.7.4. Thread Groups
- 3.7.5. Daemon Threads
- 3.7.6. Synchronization
- 3.7.7. Locking Mechanisms
- 3.7.8. Communication Mechanisms
- 3.7.9. Monitors

3.8. Common Problems in Concurrent Programming
- 3.8.1. Producer-Consumer Problem
- 3.8.2. Readers-Writers Problem
- 3.8.3. Dining Philosophers Problem

3.9. Software Testing and Documentation
- 3.9.1. Why Is It Important to Document Software?
- 3.9.2. Design Documentation
- 3.9.3. Documentation Tool Use

3.10. Software Tests
- 3.10.1. Introduction to Software Tests
- 3.10.2. Types of Tests
- 3.10.3. Unit Test
- 3.10.4. Integration Test
- 3.10.5. Validation Test
- 3.10.6. System Test

Module 4. Video Game Consoles and Devices

4.1. History of Programming in Video Games
- 4.1.1. Atari (1977-1985)
- 4.1.2. Nintendo and Super Nintendo Entertainment Systems (NES and SNES) (1985-1995)
- 4.1.3. PlayStation / PlayStation 2 (1995-2005)
- 4.1.4. Xbox 360, PlayStation 3 and Nintendo Wii (2005-2013)
- 4.1.5. Xbox One, PlayStation 4 and Nintendo Wii U - Switch (2013-present)
- 4.1.6. The Future

4.2. History of Playability in Video Games
- 4.2.1. Introduction
- 4.2.2. Social Context
- 4.2.3. Structural Diagram
- 4.2.4. Future

4.3. Adapting to Modern Times
- 4.3.1. Motion-Based Games
- 4.3.2. Virtual Reality
- 4.3.3. Augmented Reality
- 4.3.4. Mixed Reality

4.4. *Unity: Scripting* I and Examples
- 4.4.1. What Is a *Script*?
- 4.4.2. Our First *Script*
- 4.4.3. Adding a *Script*
- 4.4.4. Opening a *Script*
- 4.4.5. MonoBehaviour
- 4.4.6. *Debugging*

4.5. *Unity: Scripting* II and Examples
- 4.5.1. Enter Keyboard and Mouse
- 4.5.2. Raycast
- 4.5.3. Installation
- 4.5.4. Variables:
- 4.5.5. Public and Serialized Variables

4.6. *Unity: Scripting* III and Examples
- 4.6.1. Obtaining Components
- 4.6.2. Modifying Components
- 4.6.3. Testing
- 4.6.4. Multiple Objects
- 4.6.5. Colliders and Triggers
- 4.6.6. Quaternion

4.7. Peripherals
- 4.7.1. Evolution and Classification
- 4.7.2. Peripherals and Interface
- 4.7.3. Current Peripherals
- 4.7.4. Near Future

4.8. Video Games: Future Perspectives
- 4.8.1. Cloud-Based Games
- 4.8.2. Controller Absence
- 4.8.3. Immersive Reality
- 4.8.4. Other Alternatives

4.9. Architecture
- 4.9.1. Special Video Game Requirements
- 4.9.2. Architecture Evolution
- 4.9.3. Current Architecture
- 4.9.4. Differences between Architectures

4.10. Development Kits and Evolution
- 4.10.1. Introduction
- 4.10.2. Third Generation Development Kits
- 4.10.3. Fourth Generation Development Kits
- 4.10.4. Fifth Generation Development Kits
- 4.10.5. Sixth Generation Development Kits

Module 5. Software Engineering

5.1. Introduction to Software Engineering and Modeling
- 5.1.1. The Nature of Software
- 5.1.2. The Unique Nature of Webapps
- 5.1.3. Software Engineering
- 5.1.4. Software Process
- 5.1.5. Software Engineering Practice
- 5.1.6. Software Myths
- 5.1.7. How It All Begins
- 5.1.8. Object Oriented Concepts
- 5.1.9. Introduction to UML

5.2. Software Process
- 5.2.1. A General Process Model
- 5.2.2. Prescriptive Process Models
- 5.2.3. Specialized Process Models
- 5.2.4. Unified Process
- 5.2.5. Personal and Team Process Models
- 5.2.6. What Is Agility?
- 5.2.7. What Is an Agile Process?
- 5.2.8. Scrum
- 5.2.9. Agile Process Toolkit

5.3. Software Engineering Guiding Principles
- 5.3.1. Process Guiding Principles
- 5.3.2. Practice Guiding Principles
- 5.3.3. Communication Principles
- 5.3.4. Planning Principles
- 5.3.5. Modeling Principles
- 5.3.6. Building Principles
- 5.3.7. Deployment Principles

5.4. Understanding Requirements
- 5.4.1. Requirement Engineering
- 5.4.2. Establishing Bases
- 5.4.3. Requirements Inquiry
- 5.4.4. Use Case Development
- 5.4.5. Requirements Model Development
- 5.4.6. Requirements Negotiation
- 5.4.7. Requirements Validation

5.5. Requirements Modeling: Scenarios, Information and Types of Analysis
- 5.5.1. Requirements Analysis
- 5.5.2. Scenario-Based Modeling
- 5.5.3. UML Models Providing Use Cases
- 5.5.4. Concepts of Data Modeling
- 5.5.5. Class-Based Modeling
- 5.5.6. Class Diagrams

5.6. Requirements Modeling: Flow, Behavior and Patterns
- 5.6.1. Strategy-Modeling Requirements
- 5.6.2. Flow-Oriented Modeling
- 5.6.3. Status Diagrams
- 5.6.4. Creating Behavior Models
- 5.6.5. Sequence Diagrams
- 5.6.6. Communication Diagrams
- 5.6.7. Requirements Modeling Patterns

5.7. Design Concepts
- 5.7.1. Design in Software Engineering
- 5.7.2. Design Process
- 5.7.3. Design Concepts
- 5.7.4. Object-Oriented Design Concepts
- 5.7.5. Design Model

5.8. Architecture Design
- 5.8.1. Software Design
- 5.8.2. Architectural Genres
- 5.8.3. Architectural Styles
- 5.8.4. Architectural Design
- 5.8.5. Evolution of Alternative Designs for Architecture
- 5.8.6. Mapping Architecture Using Data Flows

5.9. Component-Level and Pattern-Based Design
- 5.9.1. What Is a Component?
- 5.9.2. Class-Based Component Design
- 5.9.3. Producing Component-Level Designs
- 5.9.4. Traditional Component Design
- 5.9.5. Component-Based Development
- 5.9.6. Design Patterns
- 5.9.7. Pattern-Based Software Design
- 5.9.8. Architectural Patterns
- 5.9.9. Component-Level Design Patterns
- 5.9.10. User Interface Design Patterns

5.10. Software Quality and Project Administration
- 5.10.1. Quality
- 5.10.2. Software Quality
- 5.10.3. The Software Quality Dilemma
- 5.10.4. Achieving Software Quality
- 5.10.5. Ensuring Software Quality
- 5.10.6. The Administrative Spectrum
- 5.10.7. The Staff
- 5.10.8. The Product
- 5.10.9. The Process
- 5.10.10. The Project
- 5.10.11. Principles and Practices

Module 6. Video Game Engines

6.1. Video Games and Information Communication Technologies (ICTs)
- 6.1.1. Introduction
- 6.1.2. Opportunities
- 6.1.3. Challenges
- 6.1.4. Conclusions

6.2. History of Video Game Engines
- 6.2.1. Introduction
- 6.2.2. Atari
- 6.2.3. The 80s
- 6.2.4. First Engines: The 90s
- 6.2.5. Current Engines

6.3. Video Game Engines
- 6.3.1. Types of Engines
- 6.3.2. Video Game Engine Parts
- 6.3.3. Current Engines
- 6.3.4. Selecting an Engine

6.4. *Motor Game Maker*
- 6.4.1. Introduction
- 6.4.2. Scenarios Design
- 6.4.3. Sprites and Animations
- 6.4.4. Collisions
- 6.4.5. *Scripting* in Game Maker Languages (GML)

6.5. Unreal Engine 4: Introduction
- 6.5.1. What Is Unreal Engine 4? What Is Its Philosophy?
- 6.5.2. Materials
- 6.5.3. UI
- 6.5.4. Animation
- 6.5.5. Particle Systems
- 6.5.6. Artificial Intelligence
- 6.5.7. Frames Per Second (FPS)

6.6. Unreal Engine 4: *Visual Scripting*
- 6.6.1. *Blueprints* and Visual *Scripting* Philosophy
- 6.6.2. *Debugging*
- 6.6.3. Types of Variables
- 6.6.4. Basic Flow Control

6.7. Unity 5 Engine
- 6.7.1. C# y Visual Studio Programming
- 6.7.2. Creating Prefabs
- 6.7.3. Using Gizmos to Control Video Games
- 6.7.4. Adaptive Engine: 2D and 3D

6.8. Godot Engine
- 6.8.1. Godot Design Philosophy
- 6.8.2. Object- and Composition-Oriented Design
- 6.8.3. All in One Package
- 6.8.4. Open and Community-Driven Software

6.9. RPG Maker Engine
- 6.9.1. RPG Maker Philosophy
- 6.9.2. Taking as a Reference
- 6.9.3. Creating a Game with Personality
- 6.9.4. Commercially Successful Games

6.10. Source 2 Engine
- 6.10.1. Source 2 Philosophy
- 6.10.2. Source and Source 2: Evolution
- 6.10.3. Community Use: Audiovisual Content and Video Games
- 6.10.4. Future of Source 2 Engine
- 6.10.5. Successful Mods and Games

Module 7. Intelligent Systems

7.1. Agents Theory
- 7.1.1. Concept History
- 7.1.2. Agent Definition
- 7.1.3. Agents in Artificial Intelligence
- 7.1.4. Agents in Software Engineering

7.2. Agent Architectures
- 7.2.1. Agent Though Process
- 7.2.2. Reactive Agents
- 7.2.3. Deductive Agents
- 7.2.4. Hybrid Agents
- 7.2.5. Comparison

7.3. Information and Knowledge
- 7.3.1. Difference between Data, Information and Knowledge
- 7.3.2. Data Quality Assessment
- 7.3.3. Data Collection Methods
- 7.3.4. Information Acquisition Methods
- 7.3.5. Knowledge Acquisition Methods

7.4. Knowledge Representation
- 7.4.1. The Importance of Knowledge Representation
- 7.4.2. Definition of Knowledge Representation According to Role
- 7.4.3. Knowledge Representation Features

7.5. Ontologies
- 7.5.1. Introduction to Metadata
- 7.5.2. Philosophical Concept of Ontology
- 7.5.3. Computing Concept of Ontology
- 7.5.4. Domain Ontologies and Higher-Level Ontologies
- 7.5.5. Building an Ontology

7.6. Ontology Languages and Ontology Creation Software
- 7.6.1. Triple RDF, Turtle and N3
- 7.6.2. RDF Schema
- 7.6.3. OWL
- 7.6.4. SPARQL
- 7.6.5. Introduction to Ontology Creation Tools
- 7.6.6. Installing and Using Protégé

7.7. Semantic Web
- 7.7.1. Current and Future Status of Semantic Web
- 7.7.2. Semantic Web Applications

7.8. Other Knowledge Representation Models
- 7.8.1. Vocabulary
- 7.8.2. Global Vision
- 7.8.3. Taxonomy
- 7.8.4. Thesaurus
- 7.8.5. Folksonomy
- 7.8.6. Comparison
- 7.8.7. Mind Maps

7.9. Knowledge Representation Assessment and Integration
- 7.9.1. Zeroth-Order Logic
- 7.9.2. First-Order Logic
- 7.9.3. Description Logic
- 7.9.4. Relation between Different Types of Logic
- 7.9.5. Prolog: Programming Based on First-Order Logic

7.10. Semantic Reasoners, Knowledge-Based Systems and Expert Systems
- 7.10.1. Concept of Reasoner
- 7.10.2. Reasoner Applications
- 7.10.3. Knowledge-Based Systems
- 7.10.4. MYCIN: History of Expert Systems
- 7.10.5. Expert Systems Elements and Architecture
- 7.10.6. Creating Expert Systems

Module 8. Real-Time Programming

8.1. Basic Concepts in Concurrent Programming
- 8.1.1. Main Concepts
- 8.1.2. Concurrency
- 8.1.3. Benefits of Concurrency
- 8.1.4. Concurrency and Hardware

8.2. Basic Concurrency Support Structures in Java
- 8.2.1. Concurrency in Java
- 8.2.2. Creating *Threads*
- 8.2.3. Methods
- 8.2.4. Synchronization

8.3. *Threads*, Life Cycles, Priorities, Interruptions, Status and Executers
- 8.3.1. *Threads*
- 8.3.2. Life Cycle
- 8.3.3. Priorities
- 8.3.4. Interruptions
- 8.3.5. Status
- 8.3.6. Executers

8.4. Mutual Exclusion
- 8.4.1. What Is Mutual Exclusion?
- 8.4.2. Dekker's Algorithm
- 8.4.3. Peterson's Algorithm
- 8.4.4. Mutual Exclusion in Java

8.5. Status Dependency
- 8.5.1. Dependency Injections
- 8.5.2. Pattern Implementation in Java
- 8.5.3. Ways to Inject Dependencies
- 8.5.4. Example

8.6. Design Patterns
- 8.6.1. Introduction
- 8.6.2. Creation Patterns
- 8.6.3. Structure Patterns
- 8.6.4. Behavior Patterns

8.7. Using Java Libraries
- 8.7.1. What Are Java Libraries?
- 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. Shader Programming
- 8.8.1. Pipeline 3D and Rasterized
- 8.8.2. Vertex Shading
- 8.8.3. Pixel Shading: Lighting I
- 8.8.4. Pixel Shading: Lighting II
- 8.8.5. Post-Effects

8.9. Real-Time Programming
- 8.9.1. Introduction
- 8.9.2. Processing Interruptions
- 8.9.3. Synchronization and Communication between Processes
- 8.9.4. Real-Time Planning Systems

8.10. Real-Time Planning
- 8.10.1. Concepts
- 8.10.2. Real-Time Systems Reference Model
- 8.10.3. Planning Policies
- 8.10.4. Cyclical Planners
- 8.10.5. Statistical Property Planners
- 8.10.6. Dynamic Property Planners

Module 9. Web Game Design and Development

9.1. Web Origins and Standards
- 9.1.1. Internet Origins
- 9.1.2. World Wide Web
- 9.1.3. First Web Standards
- 9.1.4. Rise Web Standards

9.2. HTTP and Client-Server Structure
- 9.2.1. Client-Server Role
- 9.2.2. Client-Server Communication
- 9.2.3. Recent History
- 9.2.4. Centralized Computing

9.3. Web Programming: Introduction
- 9.3.1. Basic Concepts
- 9.3.2. Preparing Web Servers
- 9.3.3. Basic Concepts of HTML5
- 9.3.4. HTML Forms

9.4. Introduction to HTML and Examples
- 9.4.1. HTML5 History
- 9.4.2. HTML5 Elements
- 9.4.3. Application Programming Interface (API)
- 9.4.4. CCS3

9.5. Document Object Model
- 9.5.1. What Is a Document Object Model?
- 9.5.2. Using DOCTYPE
- 9.5.3. The Importance of Validating the HTML
- 9.5.4. Accessing Elements
- 9.5.5. Creating Elements and Texts
- 9.5.6. Using InnerHTML
- 9.5.7. Deleting an Element or Text Node
- 9.5.8. Reading and Writing Element Attributes
- 9.5.9. Manipulating Element Styles
- 9.5.10. Attaching Multiple Files at Once

9.6. Introduction to CSS and Examples
- 9.6.1. CSS3 Syntax
- 9.6.2. Style Sheets
- 9.6.3. Labels
- 9.6.4. Selectors
- 9.6.5. CSS Web Design

9.7. Introduction to JavaScript and Examples
- 9.7.1. What Is JavaScript?
- 9.7.2. A Brief History of the Language
- 9.7.3. JavaScript Versions
- 9.7.4. Displaying Dialog Boxes
- 9.7.5. JavaScript Syntax
- 9.7.6. Understanding *Script*s
- 9.7.7. Spaces
- 9.7.8. Comments
- 9.7.9. Functions
- 9.7.10. On-Page and External JavaScript

9.8. JavaScript Functions
- 9.8.1. Function Declaration
- 9.8.2. Function Expression
- 9.8.3. Calling Functions
- 9.8.4. Recursion
- 9.8.5. Nested Functions and Closures
- 9.8.6. Variable Preservation
- 9.8.7. Multinested Functions
- 9.8.8. Name Conflicts
- 9.8.9. Closings or Closures
- 9.8.10. Function Parameters

9.9. PlayCanvas for Web Game Development
- 9.9.1. What Is PlayCanvas?
- 9.9.2. Project Configuration
- 9.9.3. Creating an Object
- 9.9.4. Adding Physics
- 9.9.5. Adding Models
- 9.9.6. Changing the Gravity and Scene Settings
- 9.9.7. Executing *Scripts*
- 9.9.8. Camara Controls

9.10. Phaser for Web Game Development
- 9.10.1. What Is Phaser?
- 9.10.2. Loading Resources
- 9.10.3. Building the World
- 9.10.4. Platforms
- 9.10.5. Players
- 9.10.6. Adding Physics
- 9.10.7. Using the Keyboard
- 9.10.8. *Pickups*
- 9.10.9. Points and Scoring
- 9.10.10. Bouncing Bombs

Module 10. Multiplayer Networks and Systems

10.1. History and Evolution of Multiplayer Video Games
- 10.1.1. The 1970s: First Multiplayer Games
- 10.1.2. The 90s: Duke Nuke, Doom and Quake
- 10.1.3. Rise of Multiplayer Video Games
- 10.1.4. Local or Online Multiplayer
- 10.1.5. Party Games

10.2. Multiplayer Business Models
- 10.2.1. Origin and Function of Emerging Business Models
- 10.2.2. Online Sales Services
- 10.2.3. Free to Play
- 10.2.4. Micropayments
- 10.2.5. Advertising
- 10.2.6. Monthly Payment Subscription
- 10.2.7. Pay to Play
- 10.2.8. Try Before You Buy

10.3. Local and Network Games
- 10.3.1. Local Games: Beginnings
- 10.3.2. Party Games: Nintendo and Family Union
- 10.3.3. Networks Games: Beginnings
- 10.3.4. Network Games Evolution

10.4. OSI Model: Layers I
- 10.4.1. OSI Model: Introduction
- 10.4.2. Physical Layer
- 10.4.3. Data Link Layer
- 10.4.4. Network Layer

10.5. OSI Model: Layers II
- 10.5.1. Transport Layer
- 10.5.2. Session Layer
- 10.5.3. Presentation Layer
- 10.5.4. Application Layer

10.6. Computer Networks and the Internet
- 10.6.1. What Are Computer Networks?
- 10.6.2. Software
- 10.6.3. Hardware
- 10.6.4. Servers
- 10.6.5. Network Storage
- 10.6.6. Network Protocols

10.7. Mobile and Wireless Networks
- 10.7.1. Mobile Networks
- 10.7.2. Wireless Networks
- 10.7.3. How Mobile Networks Work
- 10.7.4. Digital Technology

10.8. Security
- 10.8.1. Personal Security
- 10.8.2. Video Game Hacks and Cheats
- 10.8.3. Anti-Cheating Security
- 10.8.4. Anti-Cheating Security Systems Analysis

10.9. Multiplayer Systems: Servers
- 10.9.1. Server Hosting
- 10.9.2. Massively Multiplayer Online (MMO) Video Games
- 10.9.3. Dedicated Video Game Servers
- 10.9.4. Local Area Network (LAN) Parties

10.10. Multiplayer Video Game Design and Programming
- 10.10.1. Multiplayer Video Game Design Basics in Unreal
- 10.10.2. Multiplayer Video Game Design Basics in Unity
- 10.10.3. How to Make a Multiplayer Game Fun
- 10.10.4. Beyond a Controller: Multiplayer Controller Innovation

If you want to develop a great career programming world-famous video games, this is the Professional Master's Degree you are looking for"

05

Methodology

This training program offers a different way of learning. Our methodology uses a cyclical learning approach: **Relearning.**

This teaching system is used, for example, in the most prestigious medical schools in the world, and major publications such as the **New England Journal of Medicine** have considered it to be one of the most effective.

Discover Relearning, a system that abandons conventional linear learning, to take you through cyclical teaching systems: a way of learning that has proven to be extremely effective, especially in subjects that require memorization"

At TECH we use the Case Method

Our program offers a revolutionary method of skills and knowledge development. Our goal is to strengthen skills in a changing, competitive, and highly demanding environment.

At TECH, you will experience a way of learning that is shaking the foundations of traditional universities around the world"

We are the first online university to combine Harvard Business School case studies with a 100% online learning system based on repetition.

The student will learn, through collaborative activities and real cases, how to solve complex situations in real business environments.

A learning method that is different and innovative.

This intensive Information Technology program at TECH Technological University prepares you to face all the challenges in this field, both nationally and internationally. We are committed to promoting your personal and professional growth, the best way to strive for success, that is why at TECH Technological University you will use Harvard case studies, with which we have a strategic agreement that allows us, to offer you material from the best university in the world.

> *Our program prepares you to face new challenges in uncertain environments and achieve success in your career"*

The case method has been the most widely used learning system among the world's leading Information Technology schools for as long as they have existed. The case method was developed in 1912 so that law students would not only learn the law based on theoretical content. It consisted of presenting students with real-life, complex situations for them to make informed decisions and value judgments on how to resolve them. In 1924, Harvard adopted it as a standard teaching method.

What should a professional do in a given situation? This is the question that you are presented with in the case method, an action-oriented learning method. Throughout the course, students will be presented with multiple real cases. They will have to combine all their knowledge and research, and argue and defend their ideas and decisions.

Relearning Methodology

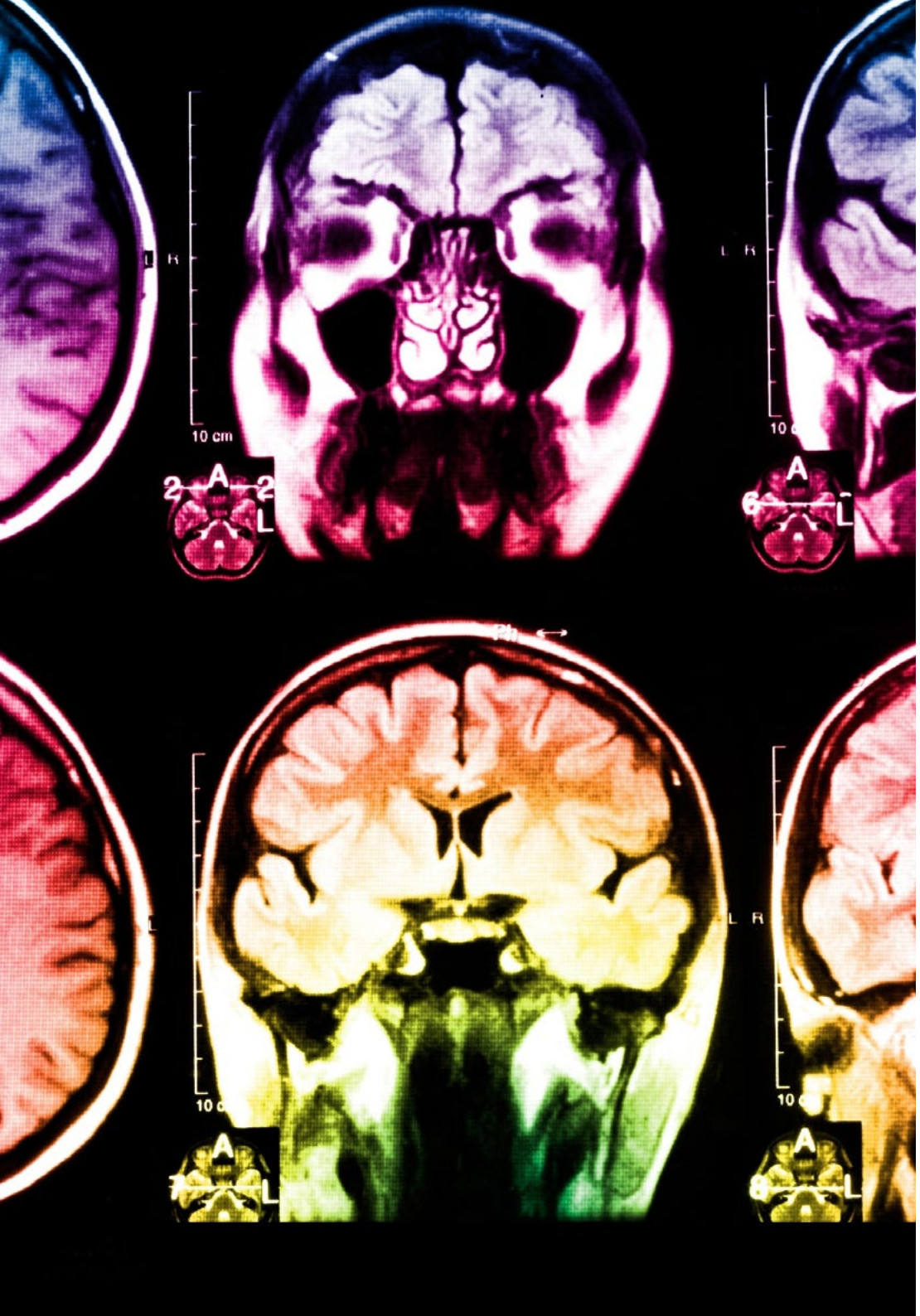
Our university is the first in the world to combine Harvard University case studies with a 100% online learning system based on repetition, which combines different teaching elements in each lesson.

We enhance Harvard case studies with the best 100% online teaching method: Relearning.

In 2019, we obtained the best learning results of all online universities in the world.

At TECH you will learn using a cutting-edge methodology designed to train the executives of the future. This method, at the forefront of international teaching, is called Relearning.

Our university is the only university in the world authorized to employ this successful method. In 2019, we managed to improve our students' overall satisfaction levels (teaching quality, quality of materials, course structure, objectives...) based on the best online university indicators.



In our program, learning is not a linear process, but rather a spiral (learn, unlearn, forget, and re-learn). Therefore, we combine each of these elements concentrically. This methodology has trained more than 650.000 university graduates with unprecedented success in fields as diverse as biochemistry, genetics, surgery, international law, management skills, sports science, philosophy, law, engineering, journalism, history, and financial markets and instruments. All this in a highly demanding environment, where the students have a strong socio-economic profile and an average age of 43.5 years.

Relearning will allow you to learn with less effort and better performance, involving you more in your training, developing a critical mindset, defending arguments, and contrasting opinions: a direct equation for success.

From the latest scientific evidence in the field of neuroscience, not only do we know how to organize information, ideas, images and memories, but we know that the place and context where we have learned something is fundamental for us to be able to remember it and store it in the hippocampus, to retain it in our long-term memory.

In this way, and in what is called neurocognitive context-dependent e-learning, the different elements in our program are connected to the context where the individual carries out their professional activity.

This program offers the best educational material, prepared with professionals in mind:

Study Material

All teaching material is produced by the specialists who teach the course, specifically for the course, so that the teaching content is highly specific and precise.

These contents are then applied to the audiovisual format, to create the TECH online working method. All this, with the latest techniques that offer high quality pieces in each and every one of the materials that are made available to the student.

Classes

There is scientific evidence suggesting that observing third-party experts can be useful.

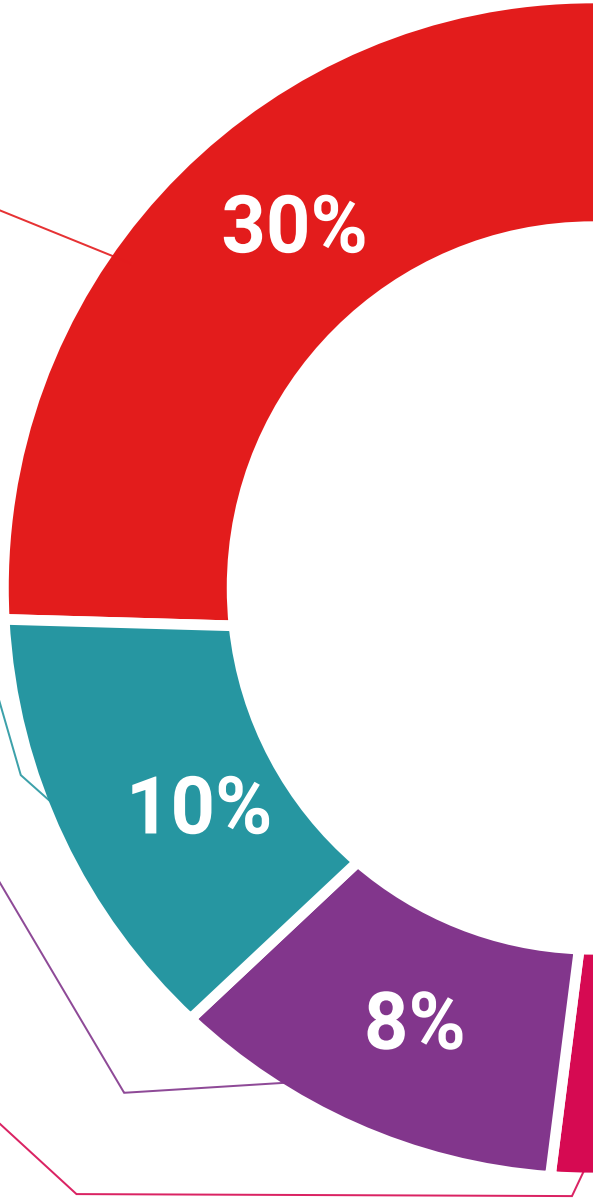
Learning from an Expert strengthens knowledge and memory, and generates confidence in future difficult decisions.

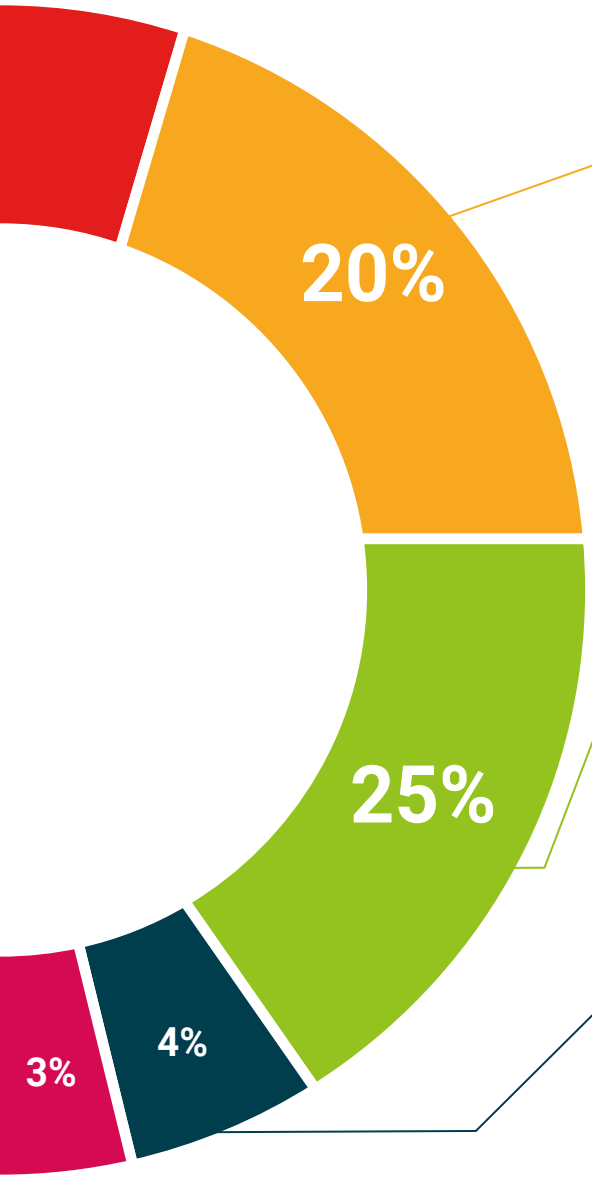
Practising Skills and Abilities

They will carry out activities to develop specific competencies and skills in each thematic area. Exercises and activities to acquire and develop the skills and abilities that a specialist needs to develop in the context of the globalization we live in.

Additional Reading

Recent articles, consensus documents and international guidelines, among others. In TECH's virtual library, students will have access to everything they need to complete their course.





Case Studies

They will complete a selection of the best case studies in the field used at Harvard. Cases that are presented, analyzed, and supervised by the best senior management specialists in the world.

Interactive Summaries

The TECH team presents the contents attractively and dynamically in multimedia lessons that include audio, videos, images, diagrams, and concept maps in order to reinforce knowledge.

This exclusive multimedia content presentation training Exclusive system was awarded by Microsoft as a "European Success Story".

Testing & Retesting

We periodically evaluate and re-evaluate students' knowledge throughout the program, through assessment and self-assessment activities and exercises: so that they can see how they are achieving your goals.

06

Certificate

The Professional Master's Degree in Video Game Programming guarantees, in addition to the most rigorous and up-to-date training, access to a qualification issued by TECH Technological University.

Successfully complete this program and receive your diploma without travel or laborious paperwork"

This **Professional Master's Degree in Video Game Programming** contains the most complete and up-to-date scientific program on the market.

After the student has passed the assessments, they will receive their corresponding **Professional Master's Degree** diploma issued by **TECH Technological University** via tracked delivery*.

The diploma issued by **TECH Technological University** will reflect the qualification obtained in the Professional Master's Degree, and meets the requirements commonly demanded by job exchanges, competitive examinations and professional career evaluation committees.

Title: **Professional Master's Degree in Video Game Programming**

Official Number of Hours: **1,500 h.**


tech technological university

Awards the following

DIPLOMA

to

Mr./Ms. _____, with identification number _____.

For having successfully passed and accredited the following program

PROFESSIONAL MASTER'S DEGREE

in

Video Game Programming

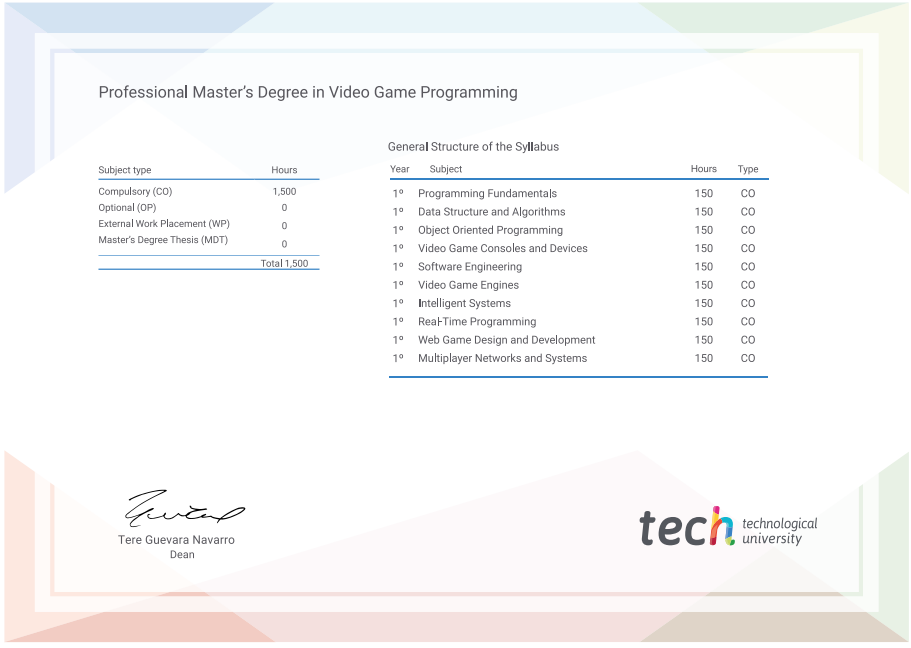
This is a qualification awarded by this University, equivalent to 1,500 hours, with a start date of dd/mm/yyyy and an end date of dd/mm/yyyy.

TECH is a Private Institution of Higher Education recognized by the Ministry of Public Education as of June 28, 2018.

June 17, 2020

Tere Guevara Navarro
Dean

This qualification must always be accompanied by the university degree issued by the competent authority to practice professionally in each country. Unique TECH Code: AFWORD23S techtitute.com/certificates


Professional Master's Degree in Video Game Programming

Subject type	Hours
Compulsory (CO)	1,500
Optional (OP)	0
External Work Placement (WP)	0
Master's Degree Thesis (MDT)	0
	Total 1,500

General Structure of the Syllabus

Year	Subject	Hours	Type
1º	Programming Fundamentals	150	CO
1º	Data Structure and Algorithms	150	CO
1º	Object Oriented Programming	150	CO
1º	Video Game Consoles and Devices	150	CO
1º	Software Engineering	150	CO
1º	Video Game Engines	150	CO
1º	Intelligent Systems	150	CO
1º	Real-Time Programming	150	CO
1º	Web Game Design and Development	150	CO
1º	Multiplayer Networks and Systems	150	CO

Tere Guevara Navarro
Dean

tech technological university

*Apostille Convention. In the event that the student wishes to have their paper diploma issued with an apostille, TECH EDUCATION will make the necessary arrangements to obtain it, at an additional cost.

tech technological university

Professional Master's Degree

Video Game Programming

- Modality: **online**
- Duration: **12 months**
- Certificate: **TECH Technological University**
- Dedication: **16h/week**
- Schedule: **at your own pace**
- Exams: **online**

Professional Master's Degree
Video Game Programming

tech technological university