Master Privato

Programmazione di Videogiochi

tech università tecnologica

tech università tecnologica

Master Privato
Programmazione di Videogiochi

- Modalità: **online**
- Durata: **12 mesi**
- Titolo: **TECH Università Tecnologica**
- Dedizione: **16 ore/settimana**
- Orario: **a scelta**
- Esami: **online**

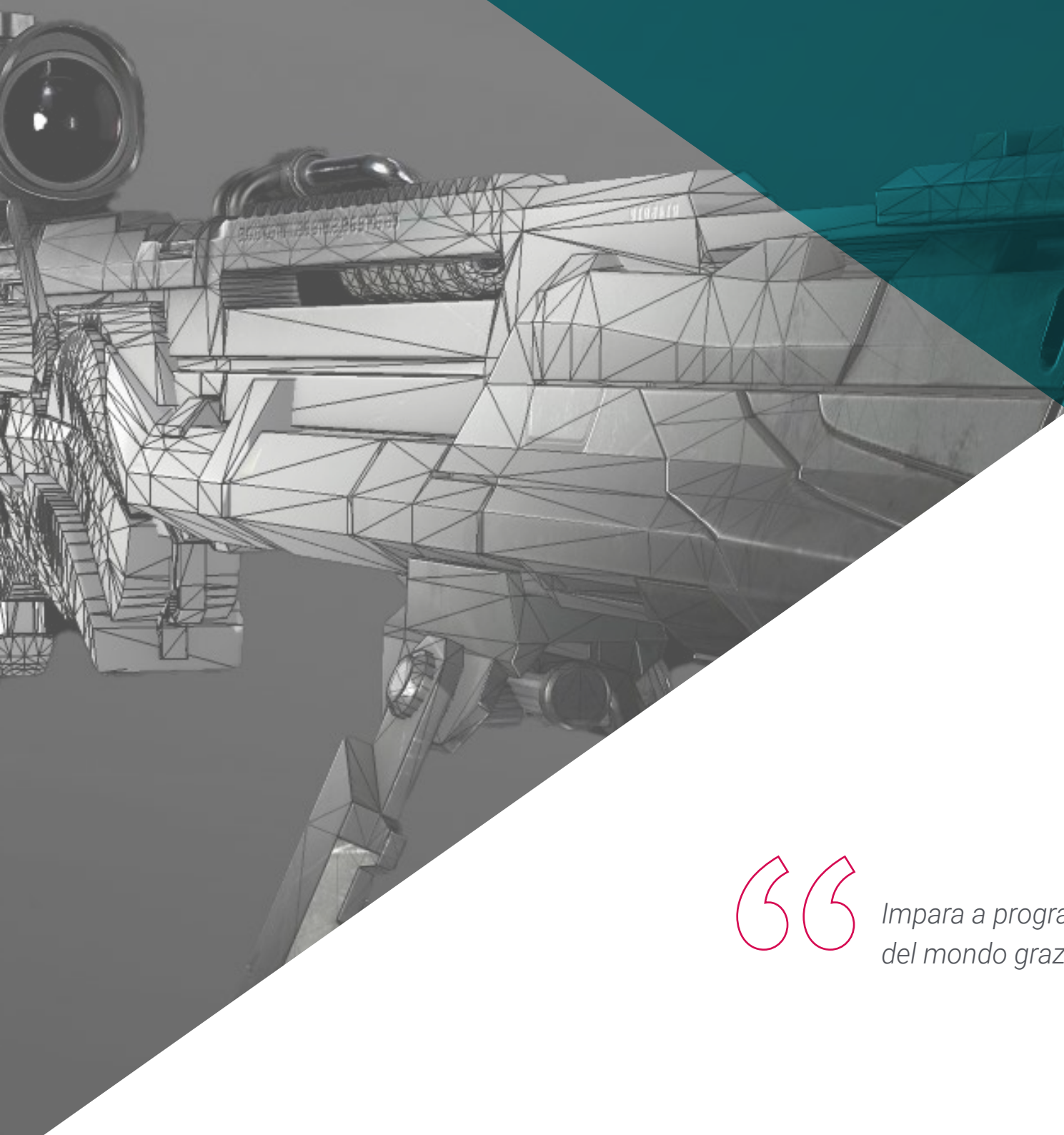
Accesso al sito web: www.techtitute.com/it/videogiochi/master/master-programmazione-videogiochi

Indice

01

Presentazione

Una delle attività più importanti e delicate nella realizzazione di un progetto videoludico è la programmazione. La programmazione è il cuore del videogioco, in quanto è il processo che crea le istruzioni di base e ne detta il funzionamento complessivo. In altre parole, senza il codice realizzato dagli sviluppatori, la grafica, la storia e il gameplay non potrebbero spiccare in un'opera audiovisiva di questo tipo. Questa specializzazione offre quindi ai suoi studenti tutte le conoscenze per diventare i migliori programmatori del settore, in modo che le più importanti aziende vogliano contare su di loro per sviluppare i propri progetti.



Impara a programmare i migliori videogiochi del mondo grazie a questo Master Privato"

L'industria videoludica ha subito una forte espansione negli ultimi anni. Dovuto alla crescente popolarità di questo tipo di intrattenimento, le aziende del settore sono state costrette a progettare e pubblicare giochi con maggiore frequenza. Si è inoltre resa necessaria una maggiore creatività, in quanto i giocatori richiedono sempre più titoli diversificati, appartenenti a generi diversi e in grado di offrire esperienze inedite.

Per questo motivo, l'industria è alla ricerca di specialisti nella programmazione di videogiochi che si assumano il ruolo fondamentale di creare il codice per le loro nuove opere. Si tratta di un lavoro delicato e che richiede una grande specializzazione, per cui è opportuno aver seguito un processo di apprendimento approfondito e ottimale per diventare un vero esperto.

Questo Master Privato in Programmazione di Videogiochi è quindi lo strumento di cui i professionisti hanno bisogno per poter lavorare nel dipartimento di sviluppo di una grande azienda del settore. Nel corso di questa specializzazione, gli studenti apprenderanno le basi della programmazione e dell'ingegneria del software, la struttura dei dati e gli algoritmi, la programmazione orientata agli oggetti e altre tematiche più specifiche come i motori grafici per videogiochi e la programmazione in tempo reale.

Questo assicura che gli studenti acquisiscano le migliori conoscenze possibili, in modo da poterle applicare direttamente nei loro campi di lavoro.

Questo **Master Privato in Programmazione di Videogiochi** possiede il programma più completo e aggiornato del mercato. Le caratteristiche principali del programma sono:

- Sviluppo di casi pratici presentati da esperti in programmazione e sviluppo di videogiochi
- Contenuti grafici, schematici ed eminentemente pratici che forniscono informazioni scientifiche e pratiche sulle discipline essenziali per l'esercizio della professione
- Esercizi pratici che offrono un processo di autovalutazione per migliorare l'apprendimento
- Speciale enfasi sulle metodologie innovative
- Lezioni teoriche, domande all'esperto e/o al tutor, forum di discussione su questioni controverse e compiti di riflessione individuale
- Contenuti disponibili da qualsiasi dispositivo fisso o portatile provvisto di connessione a internet

Le migliori aziende del settore vorranno contare su di te"

Questa specializzazione ti insegna come sviluppare i migliori videogiochi del mondo"

Grazie a questo Master Privato sarai in grado di programmare i videogiochi dei tuoi sogni.

Non aspettare oltre: programma videogiochi come fanno i migliori esperti.

Il personale docente del programma comprende rinomati specialisti del settore, che forniscono agli studenti le competenze necessarie a intraprendere un percorso di studio eccellente.

I contenuti multimediali, sviluppati in base alle ultime tecnologie educative, forniranno al professionista un apprendimento coinvolgente e localizzato, ovvero inserito in un contesto reale.

La creazione di questo programma è incentrata sull'Apprendimento Basato su Problemi, mediante il quale lo specialista deve cercare di risolvere le diverse situazioni che gli si presentano durante il corso. Lo studente potrà usufruire di un innovativo sistema di video interattivi creati da esperti di rinomata fama.

02

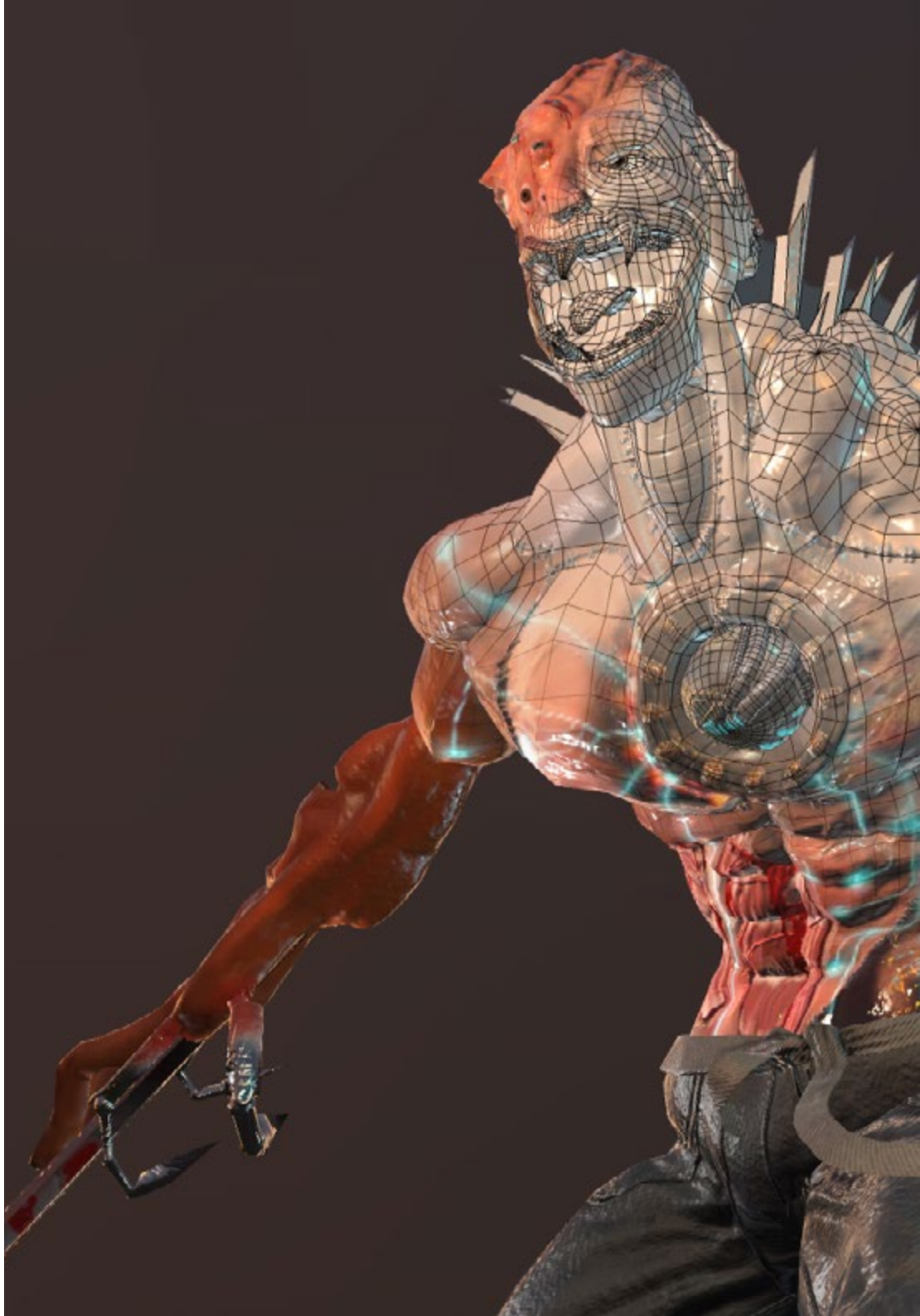
Obiettivi

L'obiettivo principale di questo Master Privato in Programmazione di Videogiochi è quello di offrire agli studenti le migliori competenze affinché diventino i migliori esperti nello sviluppo di videogiochi. Questa specializzazione offre loro una serie di strumenti specifici per questo settore, che miglioreranno il loro lavoro di sviluppatori e li porteranno a raggiungere tutti gli obiettivi professionali, riuscendo a programmare i migliori videogiochi del mondo.

Raggiungi tutti i tuoi obiettivi grazie a questa specializzazione"

Obiettivi generali

- Conoscere i diversi linguaggi e metodi di programmazione applicati ai videogiochi
- Approfondire il processo di produzione di un videogioco e l'integrazione della programmazione in queste fasi
- Imparare le basi della progettazione di videogiochi e le conoscenze teoriche che un progettista di videogiochi dovrebbe avere
- Padroneggiare i linguaggi di programmazione di base utilizzati nei videogiochi
- Applicare la conoscenza dell'ingegneria del software e della programmazione specializzata ai videogiochi
- Comprendere il ruolo della programmazione nello sviluppo di un videogioco
- Conoscere le diverse console e piattaforme esistenti
- Sviluppare videogiochi web e multiplayer



Obiettivi specifici

Modulo 1. Fondamenti di programmazione

- Comprendere la struttura di base di un computer, il software e i linguaggi di programmazione di uso generale
- Analizzare gli elementi essenziali di un programma informatico, come i diversi tipi di dati, gli operatori, le espressioni, le dichiarazioni, le istruzioni di I/O e di controllo
- Interpretare gli algoritmi, che sono la base necessaria per lo sviluppo del software

Modulo 2. Struttura dei dati e algoritmi

- Imparare le principali strategie per la progettazione di algoritmi, nonché i diversi metodi e misure per il loro calcolo
- Distinguere il funzionamento degli algoritmi, la loro strategia ed esempi del loro utilizzo nei principali problemi noti
- Comprendere la tecnica del *Backtracking* e i suoi principali utilizzi

Modulo 3. Programmazione orientata agli oggetti

- Conoscere i diversi modelli di progettazione per i problemi orientati agli oggetti
- Comprendere l'importanza della documentazione e dei test nello sviluppo del software
- Gestire l'uso dei thread e della sincronizzazione, nonché la risoluzione di problemi comuni nell'ambito della programmazione concorrente

Modulo 4. Console e dispositivi per videogiochi

- Conoscere il funzionamento di base delle principali periferiche di input e output
- Comprendere le principali esigenze progettuali delle diverse piattaforme
- Studiare la struttura, l'organizzazione, il funzionamento e l'interconnessione di dispositivi e sistemi
- Comprendere il ruolo del sistema operativo e dei kit di sviluppo per i dispositivi mobili e le piattaforme videoludiche

Modulo 5. Ingegneria del software

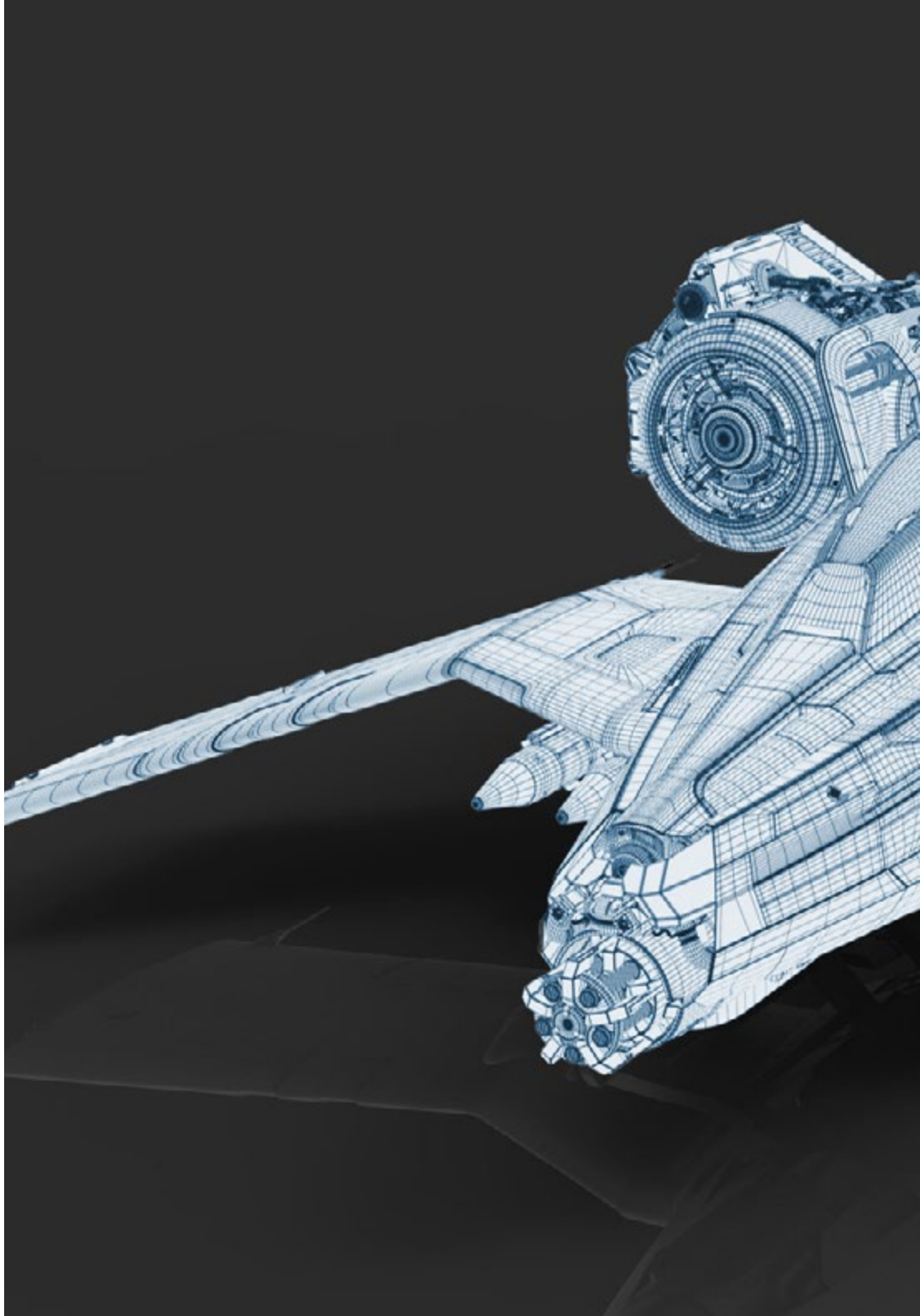
- Distinguere le basi dell'ingegneria del software, nonché il suo processo e i diversi modelli di sviluppo che comprendono le tecnologie agili
- Riconoscere l'ingegneria dei requisiti, il suo sviluppo, l'elaborazione, la negoziazione e la convalida per comprendere i principali standard relativi alla qualità del software e alla gestione dei progetti

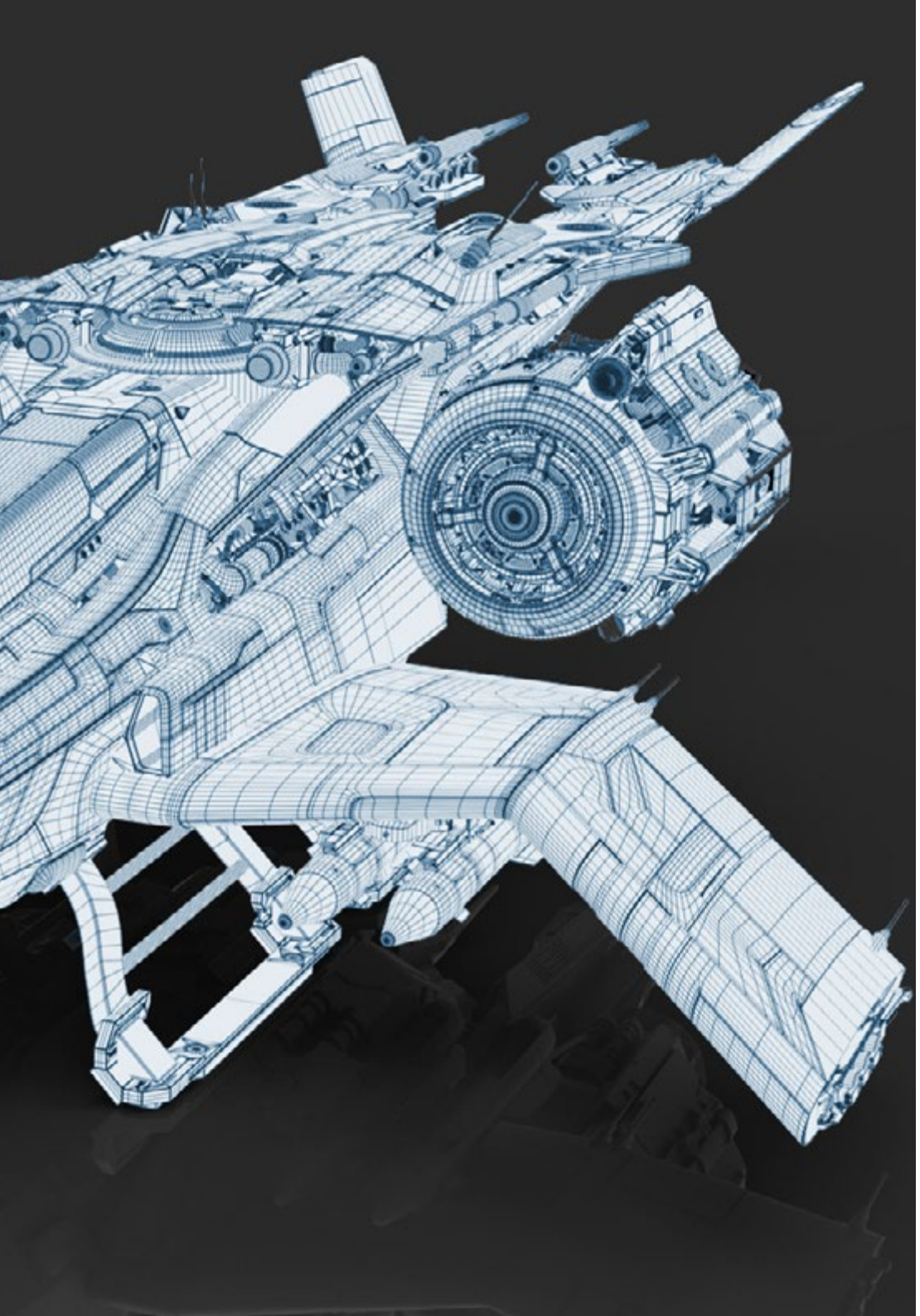
Modulo 6. Motori grafici per videogiochi

- Scoprire il funzionamento e la struttura di un motore grafico per videogiochi
- Comprendere le caratteristiche di base dei motori di gioco esistenti
- Programmare applicazioni utilizzate in modo corretto ed efficiente per i motori dei videogiochi
- Scegliere il paradigma e i linguaggi di programmazione più appropriati per la programmazione di applicazioni applicate ai motori dei videogiochi

Modulo 7. Sistemi intelligenti

- Stabilire i concetti relativi alla teoria e all'architettura dei vari elementi e il loro processo di ragionamento
- Assimilare la teoria e la pratica dei concetti di informazione e conoscenza, nonché i diversi modi di rappresentare la conoscenza
- Comprendere il funzionamento dei ragionatori semantici, dei sistemi basati sulla conoscenza e dei sistemi esperti





Modulo 8. Programmazione in tempo reale

- Analizzare le caratteristiche principali di un linguaggio di programmazione in tempo reale che lo differenziano da un linguaggio di programmazione tradizionale
- Comprendere i concetti di base dei sistemi informatici
- Acquisire la capacità di applicare i principali fondamenti e le tecniche di programmazione in tempo reale

Modulo 9. Design e sviluppo di giochi online

- Progettare giochi e applicazioni online interattive con la relativa documentazione
- Valutare le caratteristiche principali di giochi e applicazioni online interattive per comunicare in modo professionale e corretto

Modulo 10. Reti e sistemi multiplayer

- Descrivere l'architettura del transmission control protocol/internet protocol (TCP/IP) e il funzionamento di base delle reti wireless
- Analizzare la sicurezza applicata ai videogiochi
- Acquisire la capacità di sviluppare giochi online multiplayer

Se desideri lavorare nelle migliori aziende del mondo, questa specializzazione ti aiuterà a raggiungere questo obiettivo"

03

Competenze

Gli studenti di questo Master Privato otterranno una serie di competenze tali da diventare veri e propri esperti nello sviluppo di videogiochi, così da poter entrare a far parte di qualsiasi tipo di progetto nel settore. In questo modo, acquisiranno la padronanza dei diversi linguaggi di programmazione specifici utilizzati in questo tipo di prodotti audiovisivi, oltre alle competenze trasversali che devono conoscere, come il campo delle console e delle piattaforme e i motori dei videogiochi.

Saprai tutto quello che ti serve per sviluppare grandi videogiochi"

Competenze generali

- Progettare tutte le fasi di un videogioco, dall'idea iniziale al lancio sul mercato
- Specializzarsi come programmatore di videogiochi
- Approfondire tutte le parti dello sviluppo, a partire dall'architettura iniziale, la programmazione del personaggio del giocatore e tutti gli elementi coinvolti nel processo di gioco
- Ottenere una visione complessiva del progetto, potendo fornire soluzioni ai diversi problemi e difficoltà che si presentano nella progettazione di un videogioco

Grazie a questo Master Privato potrai padroneggiare tutti i tipi di linguaggi di programmazione applicati ai videogiochi"

Competenze specifiche

- Conoscere il software necessario per essere uno sviluppatore di videogiochi professionista
- Capire l'esperienza utente e saper analizzare il gameplay
- Comprendere l'intera procedura teorica e pratica del processo di programmazione dei videogiochi
- Padroneggiare i linguaggi di programmazione più utili per il mondo dei videogiochi
- Integrare la programmazione appresa con diversi tipi di console e piattaforme
- Programmare videogiochi online e multiplayer
- Assimilare il concetto di motore videoludico per poter programmare correttamente
- Applicare le conoscenze di ingegneria del software alla programmazione di videogiochi

04

Struttura e contenuti

Questo Master Privato in Programmazione di Videogiochi offre ai propri studenti i migliori contenuti nel campo dello sviluppo di videogiochi, in virtù di un'attenta elaborazione, strutturata in 10 moduli di 10 argomenti ciascuno. Grazie ai contenuti gli studenti potranno apprendere tutto ciò che serve per realizzare qualsiasi tipo di progetto videoludico, rendendo il loro processo educativo completo, esaustivo e totalmente incentrato sulla pratica.

I contenuti necessari per specializzarsi nella programmazione di videogiochi"

Modulo 1. Fondamenti di programmazione

1.1. Introduzione alla programmazione
- 1.1.1. Struttura di base di un computer
- 1.1.2. Software
- 1.1.3. Linguaggi di programmazione
- 1.1.4. Ciclo di vita di un'applicazione software

1.2. Progettazione dell'algoritmo
- 1.2.1. Risoluzione dei problemi
- 1.2.2. Tecniche descrittive
- 1.2.3. Elementi e struttura di un algoritmo

1.3. Elementi di un programma
- 1.3.1. Origine e caratteristiche del linguaggio C++
- 1.3.2. L'ambiente di sviluppo
- 1.3.3. Concetto di programma
- 1.3.4. Tipi di dati fondamentali
- 1.3.5. Operatori
- 1.3.6. Espressioni
- 1.3.7. Frasi
- 1.3.8. Input e output di dati

1.4. Frasi di controllo
- 1.4.1. Frasi
- 1.4.2. Biforcazioni
- 1.4.3. Loop

1.5. Astrazione e modularità: funzioni
- 1.5.1. Progettazione modulare
- 1.5.2. Concetto di funzione e utilità
- 1.5.3. Definizione di una funzione
- 1.5.4. Flusso di esecuzione in una chiamata di funzione
- 1.5.5. Prototipo di una funzione
- 1.5.6. Restituzione dei risultati
- 1.5.7. Chiamata di una funzione: parametri
- 1.5.8. Passaggio di parametri per riferimento e per valore
- 1.5.9. Area di identificazione

1.6. Strutture dati statiche
- 1.6.1. *Array*
- 1.6.2. Matrici. Poliedri
- 1.6.3. Ricerca e ordinamento
- 1.6.4. Stringhe. Funzioni di I/O per le stringhe
- 1.6.5. Strutture. Unioni
- 1.6.6. Nuovi tipi di dati

1.7. Strutture dati dinamiche: puntatori
- 1.7.1. Concetto. Definizione di puntatore
- 1.7.2. Operatori e operazioni con i puntatori
- 1.7.3. *Array* di puntatori
- 1.7.4. Puntatori e *array*
- 1.7.5. Puntatori a stringhe
- 1.7.6. Puntatori a strutture
- 1.7.7. Indirezione multipla
- 1.7.8. Puntatori a funzioni
- 1.7.9. Passaggio di funzioni, strutture e *array* come parametri di funzione

1.8. File
- 1.8.1. Concetti di base
- 1.8.2. Operazioni con i file
- 1.8.3. Tipi di file
- 1.8.4. Organizzazione dei file
- 1.8.5. Introduzione ai file C++
- 1.8.6. Gestione dei file

1.9. Ricorsività
- 1.9.1. Definizione di ricorsività
- 1.9.2. Tipi di ricorsività
- 1.9.3. Vantaggi e svantaggi
- 1.9.4. Considerazioni
- 1.9.5. Conversione ricorsiva-iterativa
- 1.9.6. Lo stack di ricorsività

1.10. Test e documentazione
- 1.10.1. Test del programma
- 1.10.2. Test della scatola bianca
- 1.10.3. Test della scatola nera
- 1.10.4. Strumenti di test
- 1.10.5. Documentazione di programma

Modulo 2. Struttura dei dati e degli algoritmi

2.1. Introduzione alle strategie di progettazione degli algoritmi
- 2.1.1. Ricorsività
- 2.1.2. Divide et impera
- 2.1.3. Altre strategie

2.2. Efficienza e analisi degli algoritmi
- 2.2.1. Misure di efficienza
- 2.2.2. Misurare le dimensioni dell'ingresso
- 2.2.3. Misurare il tempo di esecuzione
- 2.2.4. Caso peggiore, caso migliore e caso medio
- 2.2.5. Notazione asintotica
- 2.2.6. Criteri di analisi matematica per algoritmi non ricorsivi
- 2.2.7. Analisi matematica degli algoritmi ricorsivi
- 2.2.8. Analisi empirica degli algoritmi

2.3. Algoritmi di ordinamento
- 2.3.1. Concetto di ordinamento
- 2.3.2. Ordinamento della bolla
- 2.3.3. Ordinamento per selezione
- 2.3.4. Ordinamento per inserimento
- 2.3.5. Ordinamento per fusione (*Merge_Sort*)
- 2.3.6. Ordinamento rapido (*Quick_Sort*)

2.4. Algoritmi con alberi
- 2.4.1. Concetto di albero
- 2.4.2. Alberi binari
- 2.4.3. Percorsi ad albero
- 2.4.4. Rappresentare le espressioni
- 2.4.5. Alberi binari ordinati
- 2.4.6. Alberi binari bilanciati

2.5. Algoritmi con gli heap
- 2.5.1. Gli heap
- 2.5.2. L'algoritmo heapsort
- 2.5.3. Code di priorità

2.6. Algoritmi con grafi
- 2.6.1. Rappresentazione
- 2.6.2. Percorso in larghezza
- 2.6.3. Percorso in profondità
- 2.6.4. Ordinamento topologico

2.7. Algoritmi greedy
- 2.7.1. La strategia greedy
- 2.7.2. Elementi della strategia greedy
- 2.7.3. Cambio di valuta
- 2.7.4. Problema del viaggiatore
- 2.7.5. Problema dello zaino

2.8. Ricerca di percorsi minimi
- 2.8.1. Problema del percorso minimo
- 2.8.2. Archi e cicli negativi
- 2.8.3. Algoritmo di Dijkstra

2.9. Algoritmi greedy sui grafi
- 2.9.1. L'albero a sovrapposizione minima
- 2.9.2. Algoritmo di Prim
- 2.9.3. Algoritmo di Kruskal
- 2.9.4. Analisi della complessità

2.10. Backtracking
- 2.10.1. Il *Backtracking*
- 2.10.2. Tecniche alternative

Modulo 3. Programmazione orientata agli oggetti

3.1. Introduzione alla programmazione orientata agli oggetti
- 3.1.1. Introduzione alla programmazione orientata agli oggetti
- 3.1.2. Progettazione delle lezioni
- 3.1.3. Introduzione a UML per la modellazione dei problemi

3.2. Relazioni tra lezioni
- 3.2.1. Astrazione ed ereditarietà
- 3.2.2. Concetti avanzati di ereditarietà
- 3.2.3. Polimorfismo
- 3.2.4. Composizione e aggregazione

3.3. Introduzione ai design pattern per i problemi orientati agli oggetti
- 3.3.1. Cosa sono i design pattern?
- 3.3.2. Pattern *Factory*
- 3.3.3. Pattern *Singleton*
- 3.3.4. Pattern *Observer*
- 3.3.5. Pattern *Composite*

3.4. Eccezioni
- 3.4.1. Quali sono le eccezioni?
- 3.4.2. Gestione e acquisizione delle eccezioni
- 3.4.3. Avvio delle eccezioni
- 3.4.4. Creazione di eccezioni

3.5. Interfacce utente
- 3.5.1. Introduzione a Qt
- 3.5.2. Posizionamento
- 3.5.3. Cosa sono gli eventi?
- 3.5.4. Eventi: definizione e acquisizione
- 3.5.5. Sviluppo di interfacce utente

3.6. Introduzione alla programmazione concorrente
- 3.6.1. Introduzione alla programmazione concorrente
- 3.6.2. Il concetto di processo e di thread
- 3.6.3. Interazione tra processi o thread
- 3.6.4. Thread in C++
- 3.6.5. Vantaggi e svantaggi della programmazione concorrente

3.7. Gestione e sincronizzazione dei thread
- 3.7.1. Ciclo di vita di un thread
- 3.7.2. La classe *Thread*
- 3.7.3. Pianificazione del thread
- 3.7.4. Gruppi di thread
- 3.7.5. Thread di tipo demoniaco
- 3.7.6. Sincronizzazione
- 3.7.7. Meccanismi di bloccaggio
- 3.7.8. Meccanismi di comunicazione
- 3.7.9. Monitor

3.8. Problemi comuni nella programmazione concorrente
- 3.8.1. Il problema dei produttori-consumatori
- 3.8.2. Il problema dei lettori e degli scrittori
- 3.8.3. Il problema dei filosofi a cena

3.9. Documentazione e test del software
- 3.9.1. Perché è importante documentare il software?
- 3.9.2. Documentazione di progettazione
- 3.9.3. Utilizzo di strumenti per la documentazione

3.10. Test di software
- 3.10.1. Introduzione al test del software
- 3.10.2. Tipi di test
- 3.10.3. Test dell'unità
- 3.10.4. Test di integrità
- 3.10.5. Test di convalida
- 3.10.6. Test del sistema

Modulo 4. Console e dispositivi per videogiochi

4.1. Storia della programmazione dei videogiochi
- 4.1.1. Periodo Atari (1977-1985)
- 4.1.2. Periodo NES e SNES (1985-1995)
- 4.1.3. Periodo PlayStation/PlayStation 2 (1995-2005)
- 4.1.4. Periodo Xbox 360, PS3 y Wii (2005-2013)
- 4.1.5. Periodo Xbox One, PS4 e Wii U - Switch (2013-oggi)
- 4.1.6. Il futuro

4.2. Storia del gameplay nei videogiochi
- 4.2.1. Introduzione
- 4.2.2. Contesto sociale
- 4.2.3. Diagramma strutturale
- 4.2.4. Futuro

4.3. Adattamento ai tempi moderni
- 4.3.1. Giochi basati sul movimento
- 4.3.2. Realtà virtuale
- 4.3.3. Realtà aumentata
- 4.3.4. Realtà mista

4.4. *Unity: Scripting I* ed esempi
- 4.4.1. Che cos'è uno Script?
- 4.4.2. Il nostro primo Script
- 4.4.3. Aggiunta di uno Script
- 4.4.4. Apertura di uno Script
- 4.4.5. MonoBehaviour
- 4.4.6. *Debugging*

4.5. *Unity: Scripting II* ed esempi
- 4.5.1. Input da tastiera e mouse
- 4.5.2. Raycast
- 4.5.3. Istanziazione
- 4.5.4. Variabili
- 4.5.5. Variabili pubbliche e in serie

4.6. *Unity: Scripting III* ed esempi
- 4.6.1. Ottenere i componenti
- 4.6.2. Modificare i componenti
- 4.6.3. Test
- 4.6.4. Oggetti multipli
- 4.6.5. *Collider e trigger*
- 4.6.6. Quaternioni

4.7. Periferiche
- 4.7.1. Evoluzione e tipologie
- 4.7.2. Periferiche e interfacce
- 4.7.3. Periferiche attuali
- 4.7.4. Nel prossimo futuro

4.8. Videogiochi: prospettive future
- 4.8.1. Giochi basati sul cloud
- 4.8.2. Assenza di controller
- 4.8.3. Realtà immersiva
- 4.8.4. Alternative

4.9. Architettura
- 4.9.1. Esigenze particolari dei videogiochi
- 4.9.2. Evoluzione dell'architettura
- 4.9.3. Architettura attuale
- 4.9.4. Differenze tra architetture

4.10. Kit di sviluppo e la loro evoluzione
- 4.10.1. Introduzione
- 4.10.2. Kit di sviluppo di terza generazione
- 4.10.3. Kit di sviluppo di quarta generazione
- 4.10.4. Kit di sviluppo di quinta generazione
- 4.10.5. Kit di sviluppo di sesta generazione

Modulo 5. Ingegneria software

5.1. Introduzione all'ingegneria del software e alla modellazione
- 5.1.1. La natura del software
- 5.1.2. La natura unica delle webapp
- 5.1.3. Ingegneria software
- 5.1.4. Il processo del software
- 5.1.5. La pratica dell'ingegneria del software
- 5.1.6. Miti del software
- 5.1.7. Come nasce il tutto?
- 5.1.8. Concetti orientati agli oggetti
- 5.1.9. Introduzione alla UML

5.2. Il processo del software
- 5.2.1. Un modello generale di processo
- 5.2.2. Modelli di processo prescrittivi
- 5.2.3. Modelli di processo specializzati
- 5.2.4. Il processo unificato
- 5.2.5. Modelli di processo personali e di gruppo
- 5.2.6. Che cos'è l'agilità?
- 5.2.7. Che cos'è un processo agile?
- 5.2.8. Scrum
- 5.2.9. Kit di strumenti per i processi agili

5.3. Principi che guidano la pratica dell'ingegneria del software
- 5.3.1. Principi che guidano il processo
- 5.3.2. Principi che guidano la pratica
- 5.3.3. Principi di comunicazione
- 5.3.4. Principi di pianificazione
- 5.3.5. Principi di modellazione
- 5.3.6. Principi di costruzione
- 5.3.7. Principi di implementazione

5.4. Comprendere i requisiti
- 5.4.1. Ingegneria dei requisiti
- 5.4.2. Porre le basi
- 5.4.3. Indagine sui requisiti
- 5.4.4. Sviluppo di casi d'uso
- 5.4.5. Elaborazione del modello dei requisiti
- 5.4.6. Negoziazione dei requisiti
- 5.4.7. Convalida dei requisiti

5.5. Modellazione dei requisiti: scenari, informazioni e classi di analisi
- 5.5.1. Analisi dei requisiti
- 5.5.2. Modellazione basata su scenari
- 5.5.3. Modelli UML che forniscono il caso d'uso
- 5.5.4. Concetti di modellazione dei dati
- 5.5.5. Modellazione basata sulle classi
- 5.5.6. Diagrammi di classe

5.6. Modellazione dei requisiti: flusso, comportamento e modelli
- 5.6.1. Strategie di definizione dei requisiti
- 5.6.2. Modellazione orientata al flusso
- 5.6.3. Diagrammi di stato
- 5.6.4. Creare un modello comportamentale
- 5.6.5. Diagrammi di sequenza
- 5.6.6. Diagrammi di comunicazione
- 5.6.7. Schemi per la modellazione dei requisiti

5.7. Concetti di design
- 5.7.1. Il design nel contesto dell'ingegneria del software
- 5.7.2. Processo di design
- 5.7.3. Concetti di design
- 5.7.4. Concetti di design orientati agli oggetti
- 5.7.5. Il modello di design

5.8. Design architettonico
- 5.8.1. Architettura del software
- 5.8.2. Generi architettonici
- 5.8.3. Stili architettonici
- 5.8.4. Design architettonico
- 5.8.5. Evoluzione dei design alternativi per l'architettura
- 5.8.6. Mappatura dell'architettura con l'uso di flussi di dati

5.9. Design a livello di componente e basato su pattern
- 5.9.1. Che cos'è un componente?
- 5.9.2. Design dei componenti basato sulle classi
- 5.9.3. Realizzazione del progetto a livello di componenti
- 5.9.4. Design dei componenti tradizionali
- 5.9.5. Sviluppo basato su componenti
- 5.9.6. Modelli di design
- 5.9.7. Il design del software basato su modelli
- 5.9.8. Modelli architettonici
- 5.9.9. Modelli di design a livello di componenti
- 5.9.10. Modelli di design dell'interfaccia utente

5.10. Qualità del software e gestione dei progetti
- 5.10.1. Qualità
- 5.10.2. Qualità del software
- 5.10.3. Il dilemma della qualità del software
- 5.10.4. Raggiungere la qualità del software
- 5.10.5. Garanzia di qualità del software
- 5.10.6. Lo spettro amministrativo
- 5.10.7. Il personale
- 5.10.8. Il prodotto
- 5.10.9. Il processo
- 5.10.10. Il progetto
- 5.10.11. Principi e pratiche

Modulo 6. Motori grafici per videogiochi

6.1. Videogiochi e TIC
- 6.1.1. Introduzione
- 6.1.2. Opportunità
- 6.1.3. Difficoltà
- 6.1.4. Conclusioni

6.2. Storia dei motori grafici per videogiochi
- 6.2.1. Introduzione
- 6.2.2. Epoca Atari
- 6.2.3. Epoca anni '80
- 6.2.4. Primi motori grafici. Epoca anni '90
- 6.2.5. Motori grafici attuali

6.3. Motori grafici per videogiochi
- 6.3.1. Tipi di motori grafici
- 6.3.2. Parti che compongono un motore grafico
- 6.3.3. Motori grafici attuali
- 6.3.4. Selezione di un motore grafico per il nostro progetto

6.4. *Game Maker*
- 6.4.1. Introduzione
- 6.4.2. Progettazione degli scenari
- 6.4.3. *Sprite* e animazioni
- 6.4.4. Collisioni
- 6.4.5. Scripting in GML

6.5. Motore grafico Unreal Engine 4: introduzione
- 6.5.1. Che cos'è Unreal Engine 4? Qual è la sua filosofia?
- 6.5.2. Materiali
- 6.5.3. UI
- 6.5.4. Animazioni
- 6.5.5. Sistema di particelle
- 6.5.6. Intelligenza artificiale
- 6.5.7. FPS

6.6. Motore grafico Unreal Engine 4: Visual Scripting
- 6.6.1. Filosofia dei *Blueprint* e *Visual Scripting*
- 6.6.2. *Debugging*
- 6.6.3. Tipi di variabili
- 6.6.4. Controllo del flusso di base

6.7. Motore grafico Unity 5
- 6.7.1. Programmazione in C# e Visual Studio
- 6.7.2. Creazione di *Prefabs*
- 6.7.3. Utilizzo di Gizmos per il controllo dei videogiochi
- 6.7.4. Motore grafico adattivo: 2D e 3D

6.8. Motore grafico Godot
- 6.8.1. Filosofia progettuale di Godot
- 6.8.2. Progettazione e composizione orientata agli oggetti
- 6.8.3. Tutto in un unico pacchetto
- 6.8.4. Software gratuito e promosso dalla comunità

6.9. Motore grafico RPG Maker
- 6.9.1. Filosofia di RPG Maker
- 6.9.2. Prendere come riferimento
- 6.9.3. Creare un gioco con personalità
- 6.9.4. Giochi commerciali di successo

6.10. Motore grafico Source 2
- 6.10.1. Filosofia di Source 2
- 6.10.2. Source e Source 2: evoluzione
- 6.10.3. Uso da parte della comunità: contenuti audiovisivi e videogiochi
- 6.10.4. Futuro del motore grafico Source 2
- 6.10.5. Mod e giochi di successo

Modulo 7. Sistemi intelligenti

7.1. Teoria degli agenti
- 7.1.1. Storia del concetto
- 7.1.2. Definizione di agente
- 7.1.3. Agenti nell'intelligenza artificiale
- 7.1.4. Agenti nell'Ingegneria del Software

7.2. Architetture di agenti
- 7.2.1. Il processo di ragionamento di un agente
- 7.2.2. Agenti reattivi
- 7.2.3. Agenti deduttivi
- 7.2.4. Agenti ibridi
- 7.2.5. Confronto

7.3. Informazioni e conoscenze
- 7.3.1. Distinzione tra dati, informazioni e conoscenza
- 7.3.2. Valutazione della qualità dei dati
- 7.3.3. Metodi di acquisizione dei dati
- 7.3.4. Metodi di acquisizione delle informazioni
- 7.3.5. Metodi di acquisizione delle conoscenze

7.4. Rappresentazione della conoscenza
- 7.4.1. L'importanza della rappresentazione della conoscenza
- 7.4.2. Definire la rappresentazione della conoscenza attraverso i loro ruoli
- 7.4.3. Caratteristiche di una rappresentazione della conoscenza

7.5. Ontologie
- 7.5.1. Introduzione ai metadati
- 7.5.2. Concetto filosofico di ontologia
- 7.5.3. Concetto informatico di ontologia
- 7.5.4. Ontologie di dominio e ontologie di livello superiore
- 7.5.5. Come costruire un'ontologia?

7.6. Linguaggi ontologici e software per la creazione di ontologie
- 7.6.1. RDF Triple, Turtle e N3
- 7.6.2. Schema RDF
- 7.6.3. OWL
- 7.6.4. SPARQL
- 7.6.5. Introduzione ai diversi strumenti per la creazione di ontologie
- 7.6.6. Installazione e utilizzo di Protégé

7.7. Web semantica
- 7.7.1. Stato attuale e futuro del web semantico
- 7.7.2. Applicazioni web semantiche

7.8. Altri modelli di rappresentazione della conoscenza
- 7.8.1. Vocabolari
- 7.8.2. Panoramica globale
- 7.8.3. Tassonomie
- 7.8.4. Tesauri
- 7.8.5. Folksonomie
- 7.8.6. Confronto
- 7.8.7. Mappe mentali

7.9. Valutazione e integrazione delle rappresentazioni della conoscenza
- 7.9.1. Logica dell'ordine zero
- 7.9.2. Logica del primo ordine
- 7.9.3. Logica descrittiva
- 7.9.4. Relazione tra i diversi tipi di logica
- 7.9.5. Prolog: programmazione basata sulla logica del primo ordine

7.10. Ragionatori semantici, sistemi basati sulla conoscenza e sistemi esperti
- 7.10.1. Concetto di ragionatore
- 7.10.2. Applicazioni di un ragionatore
- 7.10.3. Sistemi basati sulla conoscenza
- 7.10.4. MYCIN, storia dei Sistemi Esperti
- 7.10.5. Elementi e Architetture dei Sistemi Esperti
- 7.10.6. Creare Sistemi Esperti

Modulo 8. Programmazione in tempo reale

8.1. Nozioni di base di programmazione concorrente
- 8.1.1. Concetti fondamentali
- 8.1.2. Concorrenza
- 8.1.3. Vantaggi della concorrenza
- 8.1.4. Concorrenza e hardware

8.2. Strutture di supporto alla concorrenza di base in Java
- 8.2.1. Concorrenza in Java
- 8.2.2. Creazione di *Thread*
- 8.2.3. Metodi
- 8.2.4. Sincronizzazione

8.3. *Thread*, ciclo di vita, priorità, interrupt, stati, esecutori
- 8.3.1. *Thread*
- 8.3.2. Ciclo di vita
- 8.3.3. Priorità
- 8.3.4. Interrupt
- 8.3.5. Stati
- 8.3.6. Esecutori

8.4. Esclusione reciproca
- 8.4.1. Che cos'è l'esclusione reciproca?
- 8.4.2. Algoritmo di Dekker
- 8.4.3. Algoritmo di Peterson
- 8.4.4. Esclusione reciproca in Java

8.5. Dipendenze di stati
- 8.5.1. Immissione di dipendenze
- 8.5.2. Implementazione del modello in Java
- 8.5.3. Modi per immettere le dipendenze
- 8.5.4. Esempio

8.6. Modelli di progettazione
- 8.6.1. Introduzione
- 8.6.2. Modelli di creazione
- 8.6.3. Modelli di struttura
- 8.6.4. Modelli di comportamento

8.7. Uso delle librerie Java
- 8.7.1. Cosa sono le librerie in 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. Programmazione degli *Shader*
- 8.8.1. Pipeline 3D e rendering
- 8.8.2. Vertex Shading
- 8.8.3. Pixel *Shading: Illuminazione I*
- 8.8.4. Pixel *Shading: Illuminazione II*
- 8.8.5. Post-effetti

8.9. Programmazione in tempo reale
- 8.9.1. Introduzione
- 8.9.2. Elaborazione degli interrupt
- 8.9.3. Sincronizzazione e comunicazione tra processi
- 8.9.4. Sistemi di pianificazione in tempo reale

8.10. Pianificazione in tempo reale
- 8.10.1. Definizioni
- 8.10.2. Modello di riferimento per i sistemi in tempo reale
- 8.10.3. Politiche di pianificazione
- 8.10.4. Pianificatori ciclici
- 8.10.5. Pianificatori con proprietà statiche
- 8.10.6. Pianificatori con proprietà dinamiche

Modulo 9. Design e sviluppo di giochi online

9.1. Origini e standard dell'online
- 9.1.1. Le origini di Internet
- 9.1.2. Creazione del *World Wide Web*
- 9.1.3. Nascita degli standard web
- 9.1.4. L'ascesa degli standard web

9.2. HTTP e struttura client-server
- 9.2.1. Ruolo client-server
- 9.2.2. Comunicazione client-server
- 9.2.3. Storia recente
- 9.2.4. Informatica centralizzata

9.3. Programmazione web: introduzione
- 9.3.1. Concetti di base
- 9.3.2. Preparazione di un server web
- 9.3.3. Nozioni di base di HTML5
- 9.3.4. Moduli HTML

9.4. Introduzione all'HTML ed esempi
- 9.4.1. Storia di HTML5
- 9.4.2. Elementi HTML5
- 9.4.3. APIS
- 9.4.4. CCS3

9.5. Modello a Oggetti del Documento
- 9.5.1. Che cos'è il Modello a Oggetti del Documento?
- 9.5.2. Uso di DOCTYPE
- 9.5.3. L'importanza della validazione dell'HTML
- 9.5.4. Accesso agli elementi
- 9.5.5. Creare elementi e testi
- 9.5.6. Uso di innerHTML
- 9.5.7. Eliminazione di un elemento di testo o di un nodo
- 9.5.8. Lettura e scrittura degli attributi di un elemento
- 9.5.9. Manipolazione degli stili degli elementi
- 9.5.10. Allegare più file contemporaneamente

9.6. Introduzione all'CSS ed esempi
- 9.6.1. Sintassi CSS3
- 9.6.2. Fogli di stile
- 9.6.3. Etichette
- 9.6.4. Selezionatori
- 9.6.5. Web design con i CSS

9.7. Introduzione a JavaScript ed esempi
- 9.7.1. Che cos'è JavaScript?
- 9.7.2. Breve storia del linguaggio
- 9.7.3. Versioni JavaScript
- 9.7.4. Visualizzare una finestra di dialogo
- 9.7.5. Sintassi di JavaScript
- 9.7.6. Capire gli Script
- 9.7.7. Spazi
- 9.7.8. Commenti
- 9.7.9. Funzioni
- 9.7.10. JavaScript esterno e sulla pagina

9.8. Funzioni in JavaScript
- 9.8.1. Dichiarazioni di funzione
- 9.8.2. Espressioni di funzione
- 9.8.3. Chiamare le funzioni
- 9.8.4. Ricorsività
- 9.8.5. Funzioni e chiusure annidate
- 9.8.6. Conservazione delle variabili
- 9.8.7. Funzioni multinanellate
- 9.8.8. Conflitti di denominazione
- 9.8.9. Chiusure
- 9.8.10. Parametri di una funzione

9.9. PlayCanvas per lo sviluppo di giochi online
- 9.9.1. Che cos'è PlayCanvas?
- 9.9.2. Configurazione del progetto
- 9.9.3. Creare un oggetto
- 9.9.4. Aggiunta della fisica
- 9.9.5. Aggiunta di un modello
- 9.9.6. Modifica delle impostazioni di gravità e della scena
- 9.9.7. Esecuzione di Script
- 9.9.8. Controlli della telecamera

9.10. Phaser per lo sviluppo di giochi online
- 9.10.1. Che cos'è Phaser?
- 9.10.2. Ricarica delle risorse
- 9.10.3. Costruire il mondo
- 9.10.4. Le piattaforme
- 9.10.5. Il giocatore
- 9.10.6. Aggiungere la fisica
- 9.10.7. Usare la tastiera
- 9.10.8. Raccogliere i *Pickup*
- 9.10.9. Punti e punteggi
- 9.10.10. Pompe rimbalzanti

Modulo 10. Reti e sistemi multiplayer

10.1. Storia ed evoluzione dei videogiochi multiplayer
- 10.1.1. Anni '70: i primi giochi multiplayer
- 10.1.2. Anni '90: Duke Nukem, Doom, Quake
- 10.1.3. L'ascesa dei videogiochi multiplayer
- 10.1.4. Multiplayer locale e online
- 10.1.5. Giochi di società

10.2. Modelli di business multiplayer
- 10.2.1. Origine e funzionamento dei modelli di business emergenti
- 10.2.2. Servizi di vendita online
- 10.2.3. Gioco gratuito
- 10.2.4. Microtransazioni
- 10.2.5. Pubblicità
- 10.2.6. Abbonamento con pagamento mensile
- 10.2.7. Pay to play
- 10.2.8. Provare prima di acquistare

10.3. Giochi multiplayer locale e online
- 10.3.1. Giochi multiplayer locale: gli inizi
- 10.3.2. Giochi di società: Nintendo e unione familiare
- 10.3.3. Giochi online: gli inizi
- 10.3.4. Evoluzione dei giochi online

10.4. Modello OSI: Livello I
- 10.4.1. Modello OSI: introduzione
- 10.4.2. Livello fisico
- 10.4.3. Livello di collegamento dati
- 10.4.4. Livello di rete

10.5. Modello OSI: Livello II
- 10.5.1. Livello di trasporto
- 10.5.2. Livello di sessione
- 10.5.3. Livello di presentazione
- 10.5.4. Livello di applicazione

10.6. Reti informatiche e internet
- 10.6.1. Che cos'è una rete di computer?
- 10.6.2. Software
- 10.6.3. Hardware
- 10.6.4. Server
- 10.6.5. Archiviazione in rete
- 10.6.6. Protocolli di rete

10.7. Reti mobili e wireless
- 10.7.1. Rete mobile
- 10.7.2. Rete wireless
- 10.7.3. Funzionamento delle reti mobili
- 10.7.4. Tecnologia digitale

10.8. Sicurezza
- 10.8.1. Sicurezza personale
- 10.8.2. *Hack e cheat* nei videogiochi
- 10.8.3. Sicurezza anti-trappola
- 10.8.4. Analisi dei sistemi di sicurezza anti-trappola

10.9. Sistemi multiplayer: server
- 10.9.1. Hosting server
- 10.9.2. Videogiochi MMO
- 10.9.3. Server dedicati ai videogiochi
- 10.9.4. LAN Parties

10.10. Design e programmazione di videogiochi multiplayer
- 10.10.1. Fondamenti di design di videogiochi multigiocatore in Unreal
- 10.10.2. Fondamenti di design di videogiochi multigiocatore in Unity
- 10.10.3. Come rendere divertente il gioco multiplayer
- 10.10.4. Oltre il controller: l'innovazione nei controlli multigiocatore

05

Metodologia

Questo programma ti offre un modo differente di imparare. La nostra metodologia si sviluppa in una modalità di apprendimento ciclico: ***il Relearning***.

Questo sistema di insegnamento viene applicato nelle più prestigiose facoltà di medicina del mondo ed è considerato uno dei più efficaci da importanti pubblicazioni come il ***New England Journal of Medicine.***

Scopri il Relearning, un sistema che abbandona l'apprendimento lineare convenzionale, per guidarti attraverso dei sistemi di insegnamento ciclici: una modalità di apprendimento che ha dimostrato la sua enorme efficacia, soprattutto nelle materie che richiedono la memorizzazione"

Caso di Studio per contestualizzare tutti i contenuti

Il nostro programma offre un metodo rivoluzionario per sviluppare le abilità e le conoscenze. Il nostro obiettivo è quello di rafforzare le competenze in un contesto mutevole, competitivo e altamente esigente.

Con TECH potrai sperimentare un modo di imparare che sta scuotendo le fondamenta delle università tradizionali di tutto il mondo"

Avrai accesso a un sistema di apprendimento basato sulla ripetizione, con un insegnamento naturale e progressivo durante tutto il programma.

Imparerai, attraverso attività collaborative e casi reali, la risoluzione di situazioni complesse in ambienti aziendali reali.

Un metodo di apprendimento innovativo e differente

Questo programma di TECH consiste in un insegnamento intensivo, creato ex novo, che propone le sfide e le decisioni più impegnative in questo campo, sia a livello nazionale che internazionale. Grazie a questa metodologia, la crescita personale e professionale viene potenziata, effettuando un passo decisivo verso il successo. Il metodo casistico, la tecnica che sta alla base di questi contenuti, garantisce il rispetto della realtà economica, sociale e professionale più attuali.

> *Il nostro programma ti prepara ad affrontare nuove sfide in ambienti incerti e a raggiungere il successo nella tua carriera"*

Il metodo casistico è stato il sistema di apprendimento più usato nelle migliori business school del mondo da quando esistono. Sviluppato nel 1912 affinché gli studenti di Diritto non imparassero la legge solo sulla base del contenuto teorico, il metodo casistico consisteva nel presentare loro situazioni reali e complesse per prendere decisioni informate e giudizi di valore su come risolverle. Nel 1924 fu stabilito come metodo di insegnamento standard ad Harvard.

Cosa dovrebbe fare un professionista per affrontare una determinata situazione? Questa è la domanda con cui ti confrontiamo nel metodo dei casi, un metodo di apprendimento orientato all'azione. Durante il corso, ti confronterai con diversi casi reali. Dovrai integrare tutte le tue conoscenze, fare ricerche, argomentare e difendere le tue idee e decisioni.

Metodologia Relearning

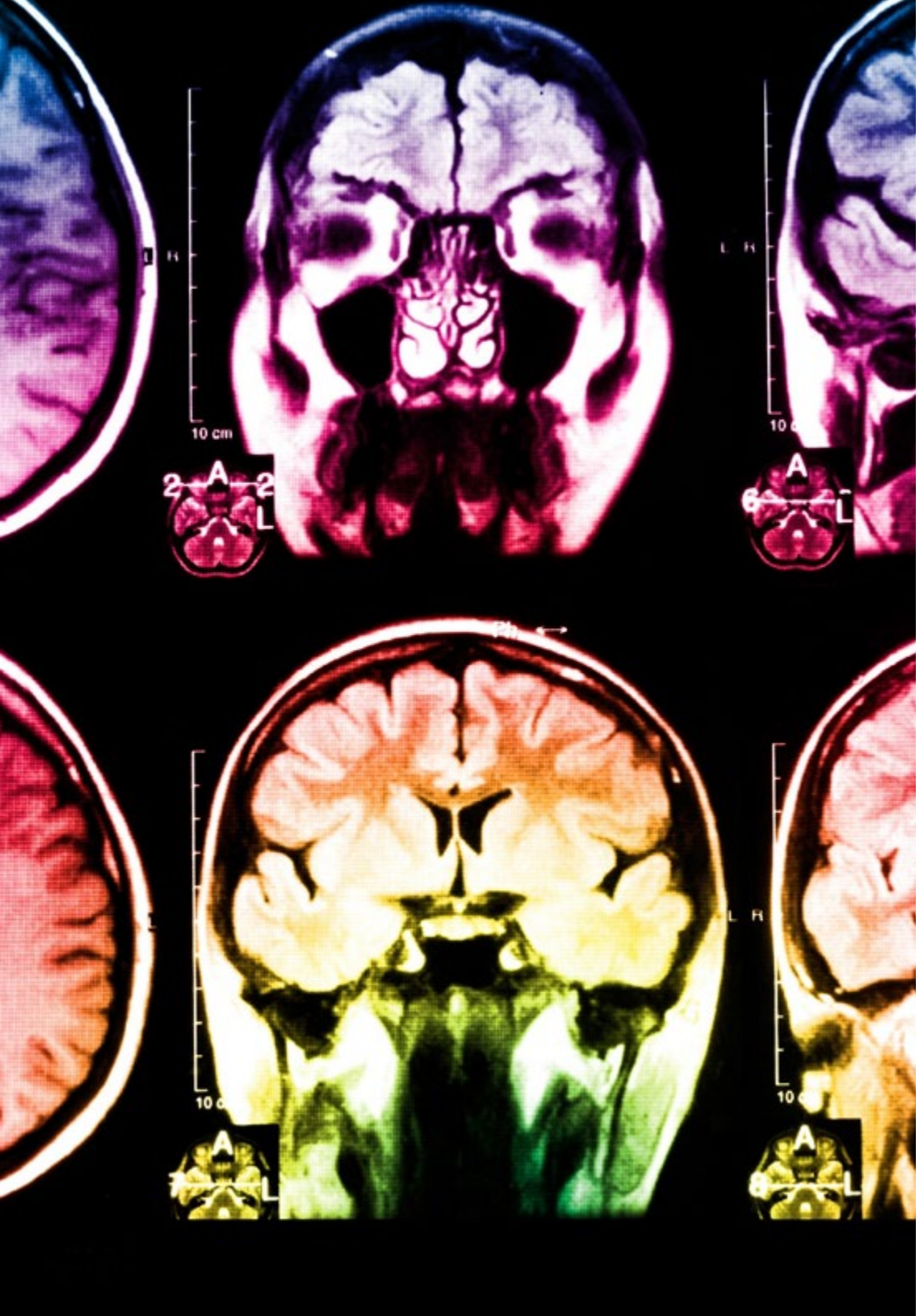
TECH coniuga efficacemente la metodologia del Caso di Studio con un sistema di apprendimento 100% online basato sulla ripetizione, che combina 8 diversi elementi didattici in ogni lezione.

Potenziamo il Caso di Studio con il miglior metodo di insegnamento 100% online: il Relearning.

Nel 2019 abbiamo ottenuto i migliori risultati di apprendimento di tutte le università online del mondo.

In TECH imparerai con una metodologia all'avanguardia progettata per formare i manager del futuro. Questo metodo, all'avanguardia della pedagogia mondiale, si chiama Relearning.

La nostra università è l'unica autorizzata a utilizzare questo metodo di successo. Nel 2019, siamo riusciti a migliorare il livello di soddisfazione generale dei nostri studenti (qualità dell'insegnamento, qualità dei materiali, struttura del corso, obiettivi...) rispetto agli indicatori della migliore università online.



Nel nostro programma, l'apprendimento non è un processo lineare, ma avviene in una spirale (impariamo, disimpariamo, dimentichiamo e re-impariamo). Pertanto, combiniamo ciascuno di questi elementi in modo concentrico. Questa metodologia ha formato più di 650.000 laureati con un successo senza precedenti in campi diversi come la biochimica, la genetica, la chirurgia, il diritto internazionale, le competenze manageriali, le scienze sportive, la filosofia, il diritto, l'ingegneria, il giornalismo, la storia, i mercati e gli strumenti finanziari. Tutto questo in un ambiente molto esigente, con un corpo di studenti universitari con un alto profilo socio-economico e un'età media di 43,5 anni.

Il Relearning ti permetterà di apprendere con meno sforzo e più performance, impegnandoti maggiormente nella tua specializzazione, sviluppando uno spirito critico, difendendo gli argomenti e contrastando le opinioni: un'equazione diretta al successo.

Dalle ultime evidenze scientifiche nel campo delle neuroscienze, non solo sappiamo come organizzare le informazioni, le idee, le immagini e i ricordi, ma sappiamo che il luogo e il contesto in cui abbiamo imparato qualcosa è fondamentale per la nostra capacità di ricordarlo e immagazzinarlo nell'ippocampo, per conservarlo nella nostra memoria a lungo termine.

In questo modo, e in quello che si chiama Neurocognitive Context-dependent E-learning, i diversi elementi del nostro programma sono collegati al contesto in cui il partecipante sviluppa la sua pratica professionale.

Questo programma offre i migliori materiali didattici, preparati appositamente per i professionisti:

Materiale di studio

Tutti i contenuti didattici sono creati appositamente per il corso dagli specialisti che lo impartiranno, per fare in modo che lo sviluppo didattico sia davvero specifico e concreto.

Questi contenuti sono poi applicati al formato audiovisivo che supporterà la modalità di lavoro online di TECH. Tutto questo, con le ultime tecniche che offrono componenti di alta qualità in ognuno dei materiali che vengono messi a disposizione dello studente.

Master class

Esistono evidenze scientifiche sull'utilità dell'osservazione di esperti terzi.

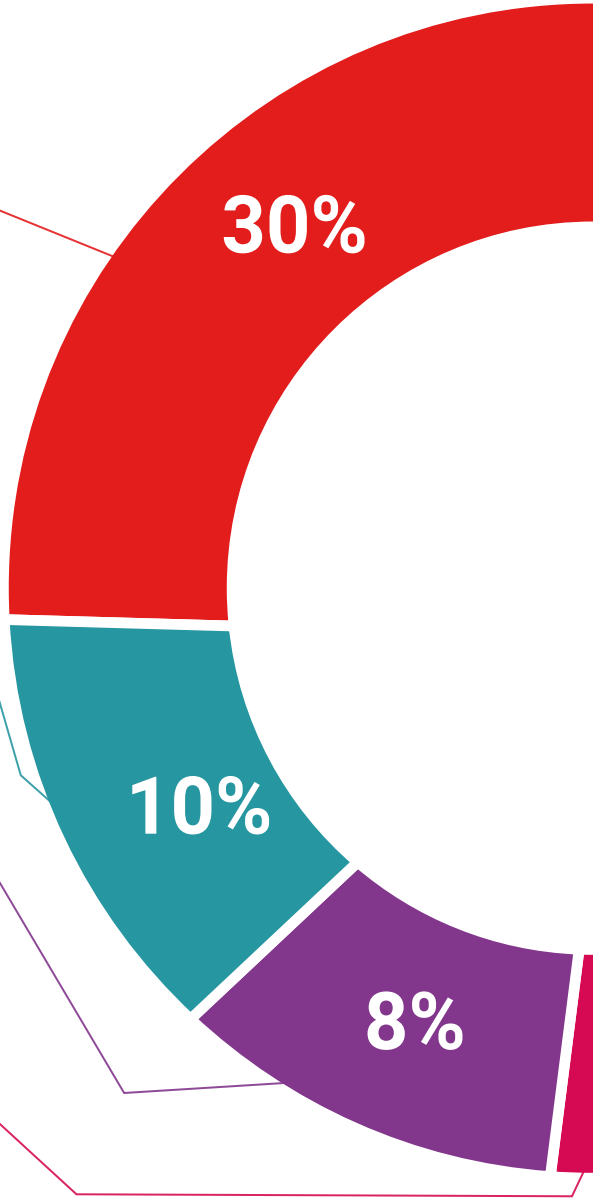
Imparare da un esperto rafforza la conoscenza e la memoria, costruisce la fiducia nelle nostre future decisioni difficili.

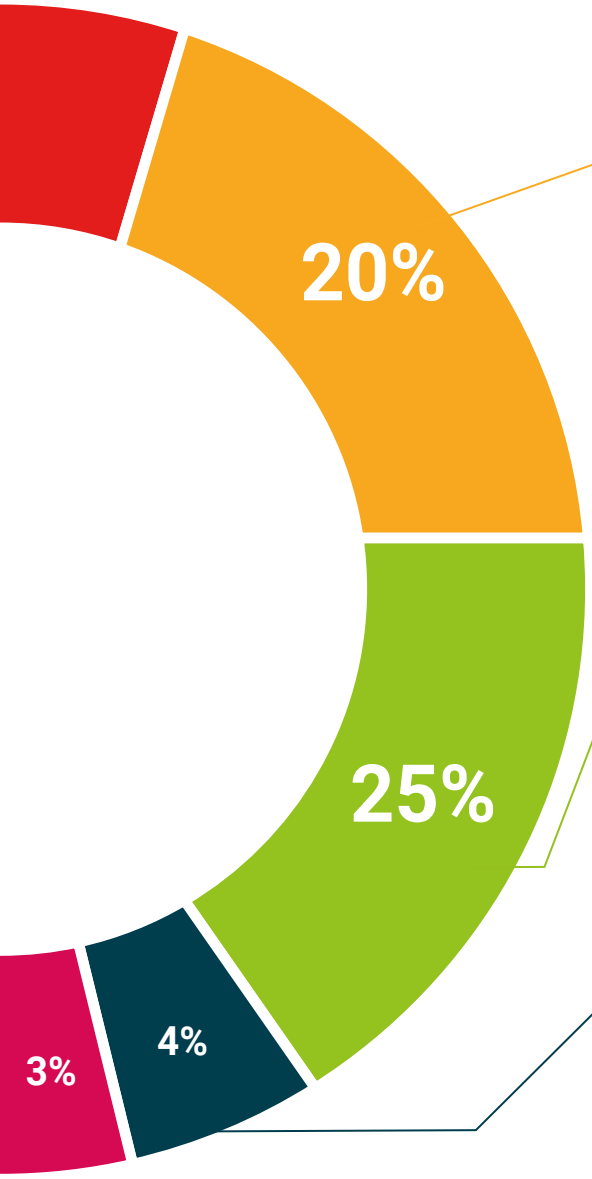
Pratiche di competenze e competenze

Svolgerai attività per sviluppare competenze e capacità specifiche in ogni area tematica. Pratiche e dinamiche per acquisire e sviluppare le competenze e le abilità che uno specialista deve sviluppare nel quadro della globalizzazione in cui viviamo.

Letture complementari

Articoli recenti, documenti di consenso e linee guida internazionali, tra gli altri. Nella biblioteca virtuale di TECH potrai accedere a tutto il materiale necessario per completare la tua specializzazione.





Casi di Studio

Completerai una selezione dei migliori casi di studio scelti appositamente per questo corso. Casi presentati, analizzati e monitorati dai migliori specialisti del panorama internazionale.

Riepiloghi interattivi

Il team di TECH presenta i contenuti in modo accattivante e dinamico in pillole multimediali che includono audio, video, immagini, diagrammi e mappe concettuali per consolidare la conoscenza.

Questo esclusivo sistema di specializzazione per la presentazione di contenuti multimediali è stato premiato da Microsoft come "Caso di successo in Europa".

Testing & Retesting

Valutiamo e rivalutiamo periodicamente le tue conoscenze durante tutto il programma con attività ed esercizi di valutazione e autovalutazione, affinché tu possa verificare come raggiungi progressivamente i tuoi obiettivi.

06

Titolo

Il Master Privato in Programmazione di Videogiochi i garantisce, oltre alla preparazione più rigorosa e aggiornata, l'accesso a una qualifica di Master Privato rilasciata da TECH Università Tecnologica.

Porta a termine questo programma e ricevi la tua qualifica universitaria senza spostamenti o fastidiose formalità”

Questo **Master Privato in Programmazione di Videogiochi** possiede il programma più completo e aggiornato del mercato.

Dopo aver superato la valutazione, lo studente riceverà mediante lettera certificata* con ricevuta di ritorno, la sua corrispondente qualifica di **Master Privato** rilasciata da **TECH Università Tecnologica.**

Il titolo rilasciato da **TECH Università Tecnologica** esprime la qualifica ottenuta nel Master Privato, e riunisce tutti i requisiti comunemente richiesti da borse di lavoro, concorsi e commissioni di valutazione di carriere professionali.

Titolo: **Master Privato in Programmazione di Videogiochi**
Nº Ore Ufficiali: **1.500 O.**


tech università tecnologica

Conferisce il presente

DIPLOMA

a

Dott./Dott.ssa _____, con documento d'identità nº_____

Per aver completato con esito positivo e accreditato il programma di

MASTER PRIVATO

in

Programmazione di Videogiochi

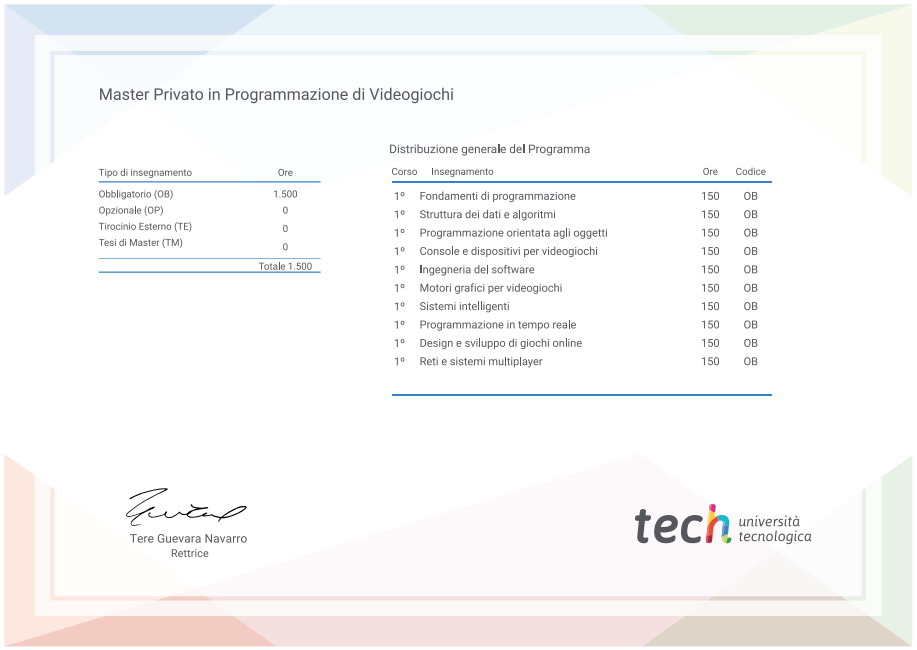
Si tratta di un titolo rilasciato da questa Università ed equivalente a 1.500 ore, con data di inizio gg/mm/aaaa e con data di fine gg/mm/aaaa.

TECH è un Istituto Privato di Istruzione Superiore riconosciuto dal Ministero della Pubblica Istruzione a partire dal 28 giugno 2018.

In data 17 Giugno 2020

Tere Guevara Navarro
Rettrice

Questo titolo deve essere sempre accompagnato da un titolo universitario rilasciato dall'autorità competente per l'esercizio della pratica professionale in ogni paese. codice unico TECH: AFWORD23S techtitute.com/titulos


Master Privato in Programmazione di Videogiochi

Tipo di insegnamento	Ore
Obbligatorio (OB)	1.500
Opzionale (OP)	0
Tirocinio Esterno (TE)	0
Tesi di Master (TM)	0
	Totale 1.500

Distribuzione generale del Programma

Corso	Insegnamento	Ore	Codice
1º	Fondamenti di programmazione	150	OB
1º	Struttura dei dati e algoritmi	150	OB
1º	Programmazione orientata agli oggetti	150	OB
1º	Console e dispositivi per videogiochi	150	OB
1º	Ingegneria del software	150	OB
1º	Motori grafici per videogiochi	150	OB
1º	Sistemi intelligenti	150	OB
1º	Programmazione in tempo reale	150	OB
1º	Design e sviluppo di giochi online	150	OB
1º	Reti e sistemi multiplayer	150	OB

Tere Guevara Navarro
Rettrice

tech università tecnologica

*Se lo studente dovesse richiedere che il suo diploma cartaceo sia provvisto di Apostille dell'Aia, TECH EDUCATION effettuerà le gestioni opportune per ottenerla pagando un costo aggiuntivo.

tech università tecnologica

Master Privato
Programmazione di Videogiochi

- Modalità: **online**
- Durata: **12 mesi**
- Titolo: **TECH Università Tecnologica**
- Dedizione: **16 ore/settimana**
- Orario: **a scelta**
- Esami: **online**

Master Privato
Programmazione di Videogiochi

tech università tecnologica