Privater Masterstudiengang
Programmierung von Videospielen

tech technologische universität

tech technologische universität

Privater Masterstudiengang
Programmierung von Videospielen

- Modalität: **online**
- Dauer: **12 Monate**
- Qualifizierung: **TECH Technologische Universität**
- Aufwand: **16 Std./Woche**
- Zeitplan: **in Ihrem eigenen Tempo**
- Prüfungen: **online**

Internetzugang: www.techtitute.com/de/videospiele/masterstudiengang/masterstudiengang-programmierung-von-videospielen

Index

01

Präsentation

Eine der wichtigsten und heikelsten Aufgaben bei der Durchführung eines Videospielprojekts ist die Programmierung. Die Programmierung ist das Herzstück eines Videospiels, denn sie ist der Prozess, der die grundlegenden Befehle erstellt und die gesamte Funktionsweise bestimmt. Das heißt, ohne den von den Entwicklern erstellten Code könnten die Optik, die Geschichte und das Gameplay in einem audiovisuellen Werk dieser Art nicht herausragen. Dieser Abschluss bietet den Studenten also das gesamte Wissen, um die besten Programmierer der Branche zu werden, so dass die besten Unternehmen bei der Entwicklung ihrer Projekte auf sie zählen wollen.

Mit diesem private Masterstudiengang lernen Sie, die besten Videospiele der Welt zu programmieren"

Die Videospielindustrie hat in den letzten Jahren eine starke Expansion erfahren. Da diese Form der Unterhaltung immer beliebter wird, sind die Unternehmen des Sektors gezwungen, immer häufiger Spiele zu entwickeln und zu veröffentlichen. Darüber hinaus ist auch mehr Kreativität erforderlich, denn die Spieler verlangen zunehmend nach Titeln, die abwechslungsreicher sind, aus verschiedenen Genres stammen und neue Erfahrungen bieten.

Aus diesem Grund sind in der Branche Spezialisten für die Programmierung von Videospielen gesucht, die die grundlegende Aufgabe der Erstellung des Codes für ihre neuen Werke übernehmen. Diese Arbeit ist heikel und erfordert ein hohes Maß an Spezialisierung, so dass es ratsam ist, einen gründlichen und optimalen Lernprozess durchlaufen zu haben, um ein echter Experte zu werden.

Dieser Private Masterstudiengang in Programmierung von Videospielen ist also genau das, was Fachleute brauchen, um in der Entwicklungsabteilung eines großen Unternehmens der Branche arbeiten zu können. Während dieses Studiums lernen die Studenten die Grundlagen der Programmierung und des Software-Engineerings, Datenstrukturen und Algorithmen, objektorientierte Programmierung und andere spezifischere Themen wie Videospiel-Engines oder Echtzeitprogrammierung.

So wird sichergestellt, dass die Studenten das bestmögliche Wissen erwerben, um es direkt in ihren Arbeitsbereichen anwenden zu können.

Dieser **Private Masterstudiengang in Programmierung von Videospielen** enthält das vollständigste und aktuellste Programm auf dem Markt. Die hervorstechendsten Merkmale sind:

- Die Entwicklung praktischer Fälle, die von Experten für Programmierung und Entwicklung von Videospielen vorgestellt werden
- Der anschauliche, schematische und äußerst praxisnahe Inhalt soll wissenschaftliche und praktische Informationen zu den für die berufliche Praxis wesentlichen Disziplinen vermitteln
- Er enthält praktische Übungen in denen der Selbstbewertungsprozess durchgeführt werden kann um das Lernen zu verbessern
- Ihr besonderer Schwerpunkt liegt auf innovativen Methoden
- Theoretische Vorträge, Fragen an den Experten, Diskussionsforen zu kontroversen Themen und individuelle Reflexionsarbeit
- Die Verfügbarkeit des Zugangs zu Inhalten von jedem festen oder tragbaren Gerät mit Internetanschluss

Die besten Unternehmen des Sektors werden auf Sie zählen wollen"

Sie wollen die besten Videospiele der Welt entwickeln, und dieser Abschluss bringt Ihnen bei, wie man das macht"

Mit diesem private Masterstudiengang werden Sie in der Lage sein, die Videospiele Ihrer Träume zu entwickeln.

Zu den Lehrkräften des Programms gehören Fachleute aus der Branche, die ihre Berufserfahrung in diese Fortbildung einbringen, sowie renommierte Fachleute von Referenzgesellschaften und angesehenen Universitäten.

Die multimedialen Inhalte, die mit den neuesten Bildungstechnologien entwickelt wurden, ermöglichen den Fachleuten ein situiertes und kontextbezogenes Lernen, d. h. eine simulierte Umgebung, die ein immersives Training ermöglicht, das auf reale Situationen ausgerichtet ist.

Das Konzept dieses Studiengangs konzentriert sich auf problemorientiertes Lernen, bei dem die Fachkräfte versuchen müssen, die verschiedenen Situationen aus der beruflichen Praxis zu lösen, die während des gesamten Studiengangs gestellt werden. Zu diesem Zweck werden sie von einem innovativen interaktiven Videosystem unterstützt, das von renommierten Experten entwickelt wurde.

Warten Sie nicht länger: Programmieren Sie Videospiele wie die besten Experten.

02

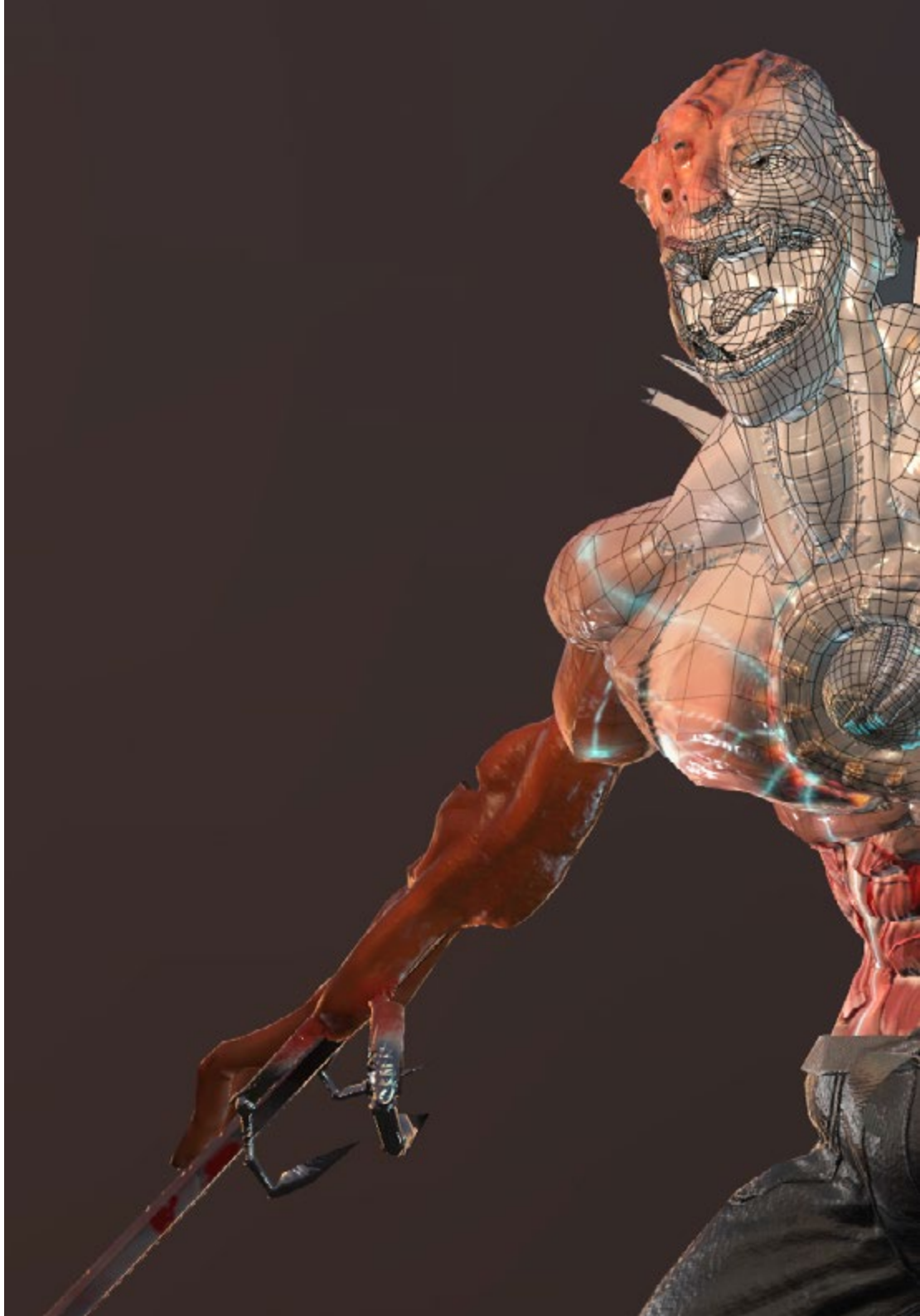
Ziele

Das Hauptziel dieses Private Masterstudiengangs in Programmierung von Videospielen besteht darin, den Studenten die besten Kenntnisse zu vermitteln, damit sie die besten Experten für die Entwicklung von Videospielen in ihrem Umfeld werden. Zu diesem Zweck bietet ihnen dieser Abschluss eine Reihe von Tools für diesen Bereich, die ihre Arbeit als Entwickler verbessern und sie dazu bringen werden, alle ihre beruflichen Ziele zu erreichen, indem sie die besten Videospiele der Welt programmieren können.

Erreichen Sie dank dieses Abschlusses alle Ihre Ziele"

Allgemeine Ziele

- Kenntnis der verschiedenen Programmiersprachen und -methoden, die für Videospiele verwendet werden
- Vertiefung des Produktionsprozesses eines Videospiels und der Integration der Programmierung in diesen Phasen
- Die Grundlagen des Videospieldesigns und das theoretische Wissen lernen, das ein Videospieldesigner kennen sollte
- Beherrschung der grundlegenden Programmiersprachen, die in Videospielen verwendet werden
- Anwendung von Kenntnissen über Softwaretechnik und spezielle Programmierung auf Videospiele
- Die Rolle der Programmierung bei der Entwicklung eines Videospiels verstehen
- Kenntnis der verschiedenen vorhandenen Konsolen und Plattformen
- Entwicklung von Web- und Multiplayer-Videospielen



Spezifische Ziele

Modul 1. Grundlagen der Programmierung

- Verständnis der grundlegenden Struktur eines Computers, von Software und allgemeinen Programmiersprachen
- Analyse der wesentlichen Elemente eines Computerprogramms, wie die verschiedenen Datentypen, Operatoren, Ausdrücke, Anweisungen, E/A- und Steueranweisungen
- Algorithmen interpretieren, die die notwendige Grundlage für die Softwareentwicklung sind

Modul 2. Datenstruktur und Algorithmen

- Die wichtigsten Strategien für den Entwurf von Algorithmen sowie die verschiedenen Methoden und Maßnahmen zur Berechnung von Algorithmen lernen
- Unterscheidung der Funktionsweise von Algorithmen, ihrer Strategie und Beispiele für ihre Anwendung bei bekannten Problemen
- Verstehen der *Backtracking* und ihrer wichtigsten Anwendungen

Modul 3. Objektorientierte Programmierung

- Kenntnis der verschiedenen Entwurfsmuster für objektorientierte Probleme
- Die Bedeutung von Dokumentation und Tests bei der Softwareentwicklung verstehen
- Die Verwendung von Threads und Synchronisierung sowie die Lösung üblicher Probleme bei der nebenläufigen Programmierung verwalten

Modul 4. Videospielkonsolen und -geräte

- Kenntnis der grundlegenden Funktionsweise der wichtigsten Ein- und Ausgabeperipheriegeräte
- Die wichtigsten Auswirkungen der verschiedenen Plattformen auf die Gestaltung verstehen

- Untersuchung der Struktur, Organisation, Funktionsweise und Verbindung von Geräten und Systemen
- Die Rolle von Betriebssystemen und Entwicklungskits für mobile Geräte und Spieleplattformen verstehen

Modul 5. Software-Entwicklung

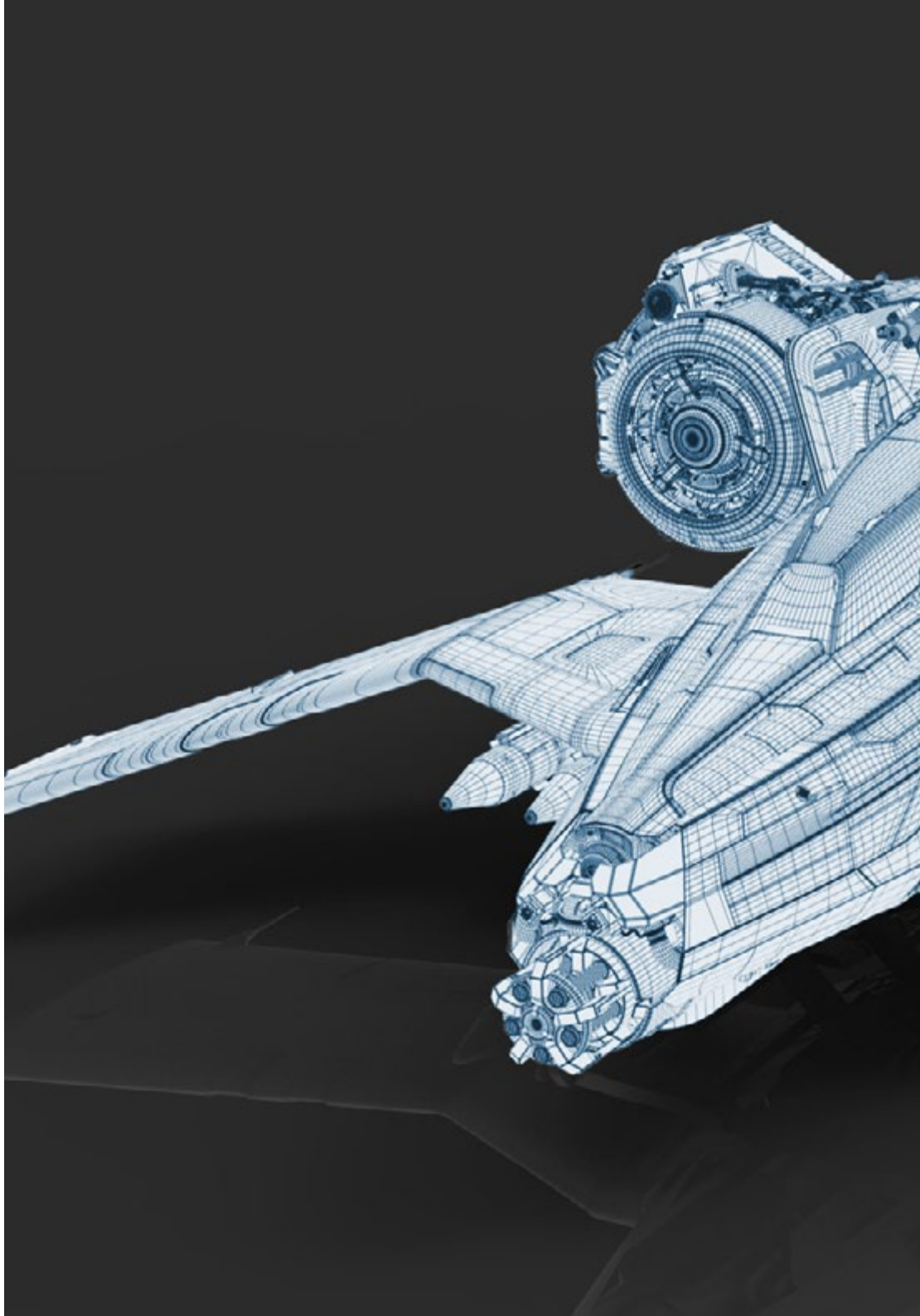
- Die Grundlagen des Software-Engineerings sowie den Softwareprozess und die verschiedenen Modelle für die Softwareentwicklung einschließlich agiler Technologien kennen
- Erkennen von Requirements Engineering, dessen Entwicklung, Ausarbeitung, Verhandlung und Validierung, um die wichtigsten Standards im Zusammenhang mit Softwarequalität und Projektmanagement zu verstehen

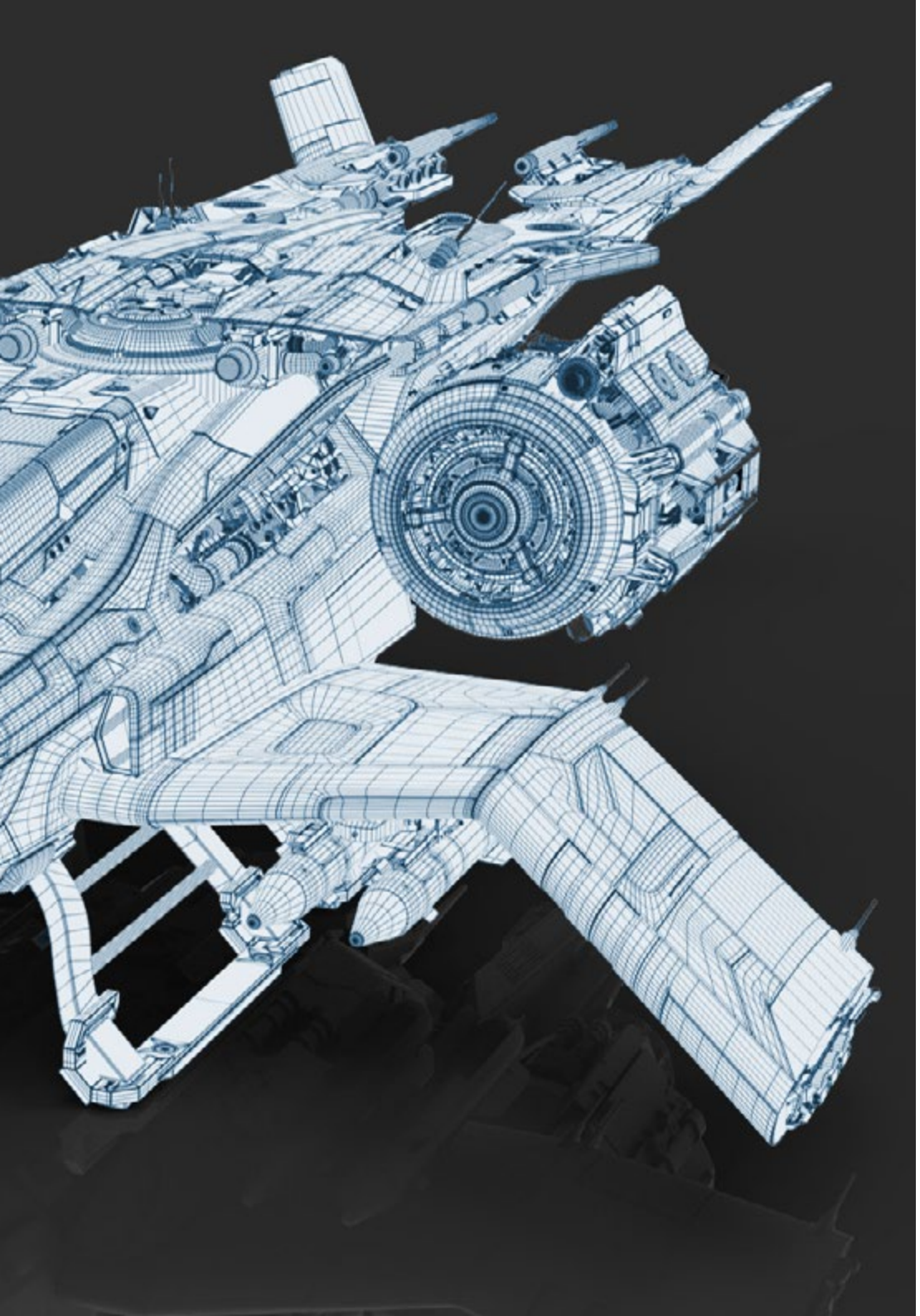
Modul 6. Videospiel-Engines

- Die Funktionsweise und Architektur einer Videospiel-Engine entdecken
- Verständnis ihrer grundlegenden Funktionen und Modifizierung bestehender Spiel-Engines
- Programmieren Sie Anwendungen, die korrekt und effizient auf Videospiel-Engines angewendet werden
- Auswahl des am besten geeigneten Paradigmas und der Programmiersprachen für die Programmierung von Anwendungen für Videospiel-Engines

Modul 7. Intelligente Systeme

- Festlegung der Konzepte im Zusammenhang mit der Agententheorie und der Agentenarchitektur sowie deren Argumentationsprozess
- Aneignung der Theorie und Praxis der Konzepte von Information und Wissen sowie der verschiedenen Formen der Wissensdarstellung
- Verstehen der Funktionsweise von semantischen Reasonern, wissensbasierten Systemen und Expertensystemen





Modul 8. Echtzeit- Programmierung

- Analyse der wichtigsten Merkmale einer Echtzeit-Programmiersprache, die sie von einer traditionellen Programmiersprache unterscheiden
- Verstehen der grundlegenden Begriffe von Computersystemen
- Die Fähigkeit zu erwerben, die wichtigsten Grundlagen und Techniken der Echtzeitprogrammierung anzuwenden

Modul 9. Design und Entwicklung von Webspielen

- Gestaltung von Spielen und interaktiven Webanwendungen mit entsprechender Dokumentation
- Die wichtigsten Merkmale von Spielen und interaktiven Webanwendungen zu bewerten, um professionell und korrekt zu kommunizieren

Modul 10. Multiplayer-Netze und -Systeme

- Beschreibung der Architektur von TCP/IP (Transmission Control Protocol/Internet Protocol) und der grundlegenden Funktionsweise von drahtlosen Netzwerken
- Analyse der Sicherheit bei Videospielen
- Die Fähigkeit erlangen, Multiplayer-Online-Spiele zu entwickeln

Sie wollen einen Job in den besten Unternehmen der Welt, und dieser Abschluss wird Ihnen dabei helfen"

03

Kompetenzen

Die Studenten dieses private Masterstudiengangs erwerben eine Reihe von Fähigkeiten, die sie zu echten Experten in der Videospielentwicklung machen, so dass sie an jeder Art von Projekt in der Branche teilnehmen können. Auf diese Weise beherrschen die Studenten die verschiedenen spezifischen Programmiersprachen, die in dieser Art von audiovisuellen Produkten verwendet werden, sowie übergreifende Fähigkeiten, die sie kennen müssen, wie z. B. den Bereich der Konsolen und Plattformen und der Videospielmotoren.

Sie werden alles wissen, was Sie brauchen, um großartige Videospiele zu entwickeln"

Allgemeine Kompetenzen

- Entwicklung aller Phasen eines Videospiels, von der ersten Idee bis zur endgültigen Markteinführung
- Spezialisierung als Videospielprogrammierer
- Alle Teile der Entwicklung, von der anfänglichen Architektur über die Programmierung der Spielerfigur bis hin zu allen am Spielprozess beteiligten Elementen, sollen untersucht werden
- Eine Gesamtvision des Projekts erhalten und in der Lage sein, Lösungen für die verschiedenen Probleme und Herausforderungen finden, die bei der Entwicklung eines Videospiels auftreten

Beherrschen Sie alle Arten von Programmiersprachen, die für Videospiele verwendet werden, mit diesem private Masterstudiengang"

Spezifische Kompetenzen

- Die notwendigen Programme kennen, um im Design und in der Entwiclung von Videospielen ein Profi zu sein
- Die Erfahrung des Spielers verstehen und wissen, wie man den Spielverlauf analysiert
- Den theoretischen und praktischen Prozess der Programmierung eines Videospiels verstehen
- Die nützlichsten Programmiersprachen für die Welt der Videospiele beherrschen
- Integration der erlernten Programmierung in verschiedene Arten von Konsolen und Plattformen
- Programmierung von Web- und Multiplayer-Videospielen
- Sich das Konzept der Videospiel-Engine aneignen, um richtig programmieren zu können
- Anwendung der Kenntnisse der Softwaretechnik auf die Programmierung von Videospielen

04

Struktur und Inhalt

Dieser Private Masterstudiengang in Videospielprogrammierung bietet seinen Studenten dank seiner sorgfältigen Konzeption, die in 10 Module zu je 10 Themen gegliedert ist, die besten Inhalte in der Videospielentwicklung. Durch sie werden die Studenten in der Lage sein, alles zu lernen, was sie brauchen, um an jeder Art von Videospielprojekt teilzunehmen, so dass ihr Bildungsprozess vollständig, umfassend und völlig praxisorientiert ist.

Die Inhalte, die Sie brauchen, um sich auf die Programmierung von Videospielen zu spezialisieren"

Modul 1. Grundlagen der Programmierung

1.1. Einführung in die Programmierung
- 1.1.1. Grundlegende Struktur eines Computers
- 1.1.2. Software
- 1.1.3. Programmiersprachen
- 1.1.4. Lebenszyklus einer Softwareanwendung

1.2. Entwurf eines Algorithmus
- 1.2.1. Lösung von Problemen
- 1.2.2. Beschreibende Techniken
- 1.2.3. Elemente und Struktur eines Algorithmus

1.3. Elemente eines Programms
- 1.3.1. Ursprung und Eigenschaften der Sprache C++
- 1.3.2. Die Entwicklungsumgebung
- 1.3.3. Das Programmkonzept
- 1.3.4. Arten von grundlegender Daten
- 1.3.5. Operatoren
- 1.3.6. Ausdrücke
- 1.3.7. Anweisungen
- 1.3.8. Dateneingabe und -ausgabe

1.4. Kontrollanweisungen
- 1.4.1. Anweisungen
- 1.4.2. Verzweigungen
- 1.4.3. Schleifen

1.5. Abstraktion und Modularität: Funktionen
- 1.5.1. Modularer Aufbau
- 1.5.2. Begriff der Funktion und des Nutzens
- 1.5.3. Definition einer Funktion
- 1.5.4. Ausführungsablauf bei einem Funktionsaufruf
- 1.5.5. Prototype einer Funktion
- 1.5.6. Rückgabe der Ergebnisse
- 1.5.7. Aufrufen einer Funktion: Parameter
- 1.5.8. Parameterübergabe durch Verweis und durch Wert
- 1.5.9. Identifizierungsbereich

1.6. Statische Datenstrukturen
- 1.6.1. *Arrays*
- 1.6.2. Matrizen. Polyeder
- 1.6.3. Suche und Sortierung
- 1.6.4. Ketten E/A-Funktionen für Zeichenketten
- 1.6.5. Strukturen Verbindungen
- 1.6.6. Neue Datentypen

1.7. Dynamische Datenstrukturen: Zeiger
- 1.7.1. Konzept. Definition von Zeiger
- 1.7.2. Operatoren und Operationen mit Zeigern
- 1.7.3. *Arrays* von Zeigern
- 1.7.4. Zeiger und *arrays*
- 1.7.5. Zeiger auf Strings
- 1.7.6. Zeiger auf Strukturen
- 1.7.7. Mehrfache Indirektion
- 1.7.8. Zeiger auf Funktionen
- 1.7.9. Übergabe von Funktionen, Strukturen und *Arrays* als Funktionsparameter

1.8. Dateien
- 1.8.1. Grundlegende Konzepte
- 1.8.2. Dateioperationen
- 1.8.3. Dateitypen
- 1.8.4. Organisation der Dateien
- 1.8.5. Einführung in C++-Dateien
- 1.8.6. Umgang mit Dateien

1.9. Rekursion
- 1.9.1. Definition von Rekursion
- 1.9.2. Arten der Rekursion
- 1.9.3. Vorteile und Nachteile
- 1.9.4. Überlegungen
- 1.9.5. Rekursiv-iterative Umwandlung
- 1.9.6. Der Rekursionsstapel

1.10. Nachweise und Dokumentation
- 1.10.1. Programm-Tests
- 1.10.2. White-Box-Test
- 1.10.3. Black-Box-Test
- 1.10.4. Prüfwerkzeuge
- 1.10.5. Dokumentation des Programms

Modul 2. Datenstruktur und Algorithmen

2.1. Einführung in Algorithmenentwurfsstrategien
- 2.1.1. Rekursion
- 2.1.2. Teilen und erobern
- 2.1.3. Andere Strategien

2.2. Effizient und Analyse von Algorithmen
- 2.2.1. Maßnahmen zur Steigerung der Effizienz
- 2.2.2. Messung der Größe des Eingangs
- 2.2.3. Messung der Ausführungszeit
- 2.2.4. Schlimmster Fall, bester Fall und mittlerer Fall
- 2.2.5. Asymptotische Notation
- 2.2.6. Mathematische Analysekriterien für nicht-rekursive Algorithmen
- 2.2.7. Mathematische Analyse von rekursiven Algorithmen
- 2.2.8. Empirische Analyse von Algorithmen

2.3. Sortierungsalgorithmen
- 2.3.1. Management-Konzept
- 2.3.2. Sortierung der Blase
- 2.3.3. Sortierung nach Auswahl
- 2.3.4. Reihenfolge der Einfügung
- 2.3.5. Mischsortierung (*Merge_Sort*)
- 2.3.6. Schnelle Sortierung (*Quick_Sort*)

2.4. Algorithmen mit Bäumen
- 2.4.1. Baum-Konzept
- 2.4.2. Binäre Bäume
- 2.4.3. Traversierung
- 2.4.4. Darstellung von Ausdrücken
- 2.4.5. Binärbäume
- 2.4.6. Ausgeglichene Binärbäume

2.5. Algorithmen mit Heaps
- 2.5.1. Die Heaps
- 2.5.2. Der Heapsort-Algorithmus
- 2.5.3. Vorrangwarteschlange

2.6. Graphalgorithmen
- 2.6.1. Vertretung
- 2.6.2. Reisen in die Breite
- 2.6.3. Reisen in die Tiefe
- 2.6.4. Topologische Ordnung

2.7. Greedy-Algorithmen
- 2.7.1. Greedy Strategie
- 2.7.2. Elemente der Greedy-Strategie
- 2.7.3. Währungsumtausch
- 2.7.4. Das Problem des Reisenden
- 2.7.5. Problem mit dem Rucksack

2.8. Suche nach minimalen Wegen
- 2.8.1. Das Problem des minimalen Weges
- 2.8.2. Negative Bögen und Zyklen
- 2.8.3. Dijkstra-Algorithmus

2.9. Greedy-Algorithmen über Graphen
- 2.9.1. Der minimal überlappende Baum
- 2.9.2. Prim-Algorithmus
- 2.9.3. Kruskal-Algorithmus
- 2.9.4. Komplexitätsanalyse

2.10. Backtracking
- 2.10.1. Das *Backtracking*
- 2.10.2. Alternative Techniken

Modul 3. Objektorientierte Programmierung

3.1. Einführung in die objektorientierte Programmierung
- 3.1.1. Einführung in die objektorientierte Programmierung
- 3.1.2. Entwurf der Klassen
- 3.1.3. Einführung in UML für die Modellierung von Problemen

3.2. Beziehungen zwischen den Klassen
- 3.2.1. Abstraktion und Erbe
- 3.2.2. Fortgeschrittene Konzepte der Erbe
- 3.2.3. Polymorphismus
- 3.2.4. Zusammensetzung und das Hinzufügen

3.3. Einführung in die Entwurfsmuster für objektorientierte Probleme
- 3.3.1. Was sind die Entwurfsmuster?
- 3.3.2. *Factory-Muster*
- 3.3.3. *Singleton-Muster*
- 3.3.4. *Observer-Muster*
- 3.3.5. *Composite-Muster*

3.4. Ausnahmen
- 3.4.1. Was sind die Ausnahmen?
- 3.4.2. Erfassung und Behandlung von Ausnahmen
- 3.4.3. Starten von Ausnahmen
- 3.4.4. Erstellung von Ausnahmen

3.5. Benutzerschnittstellen
- 3.5.1. Einführung in Qt
- 3.5.2. Positionierung
- 3.5.3. Was sind die Events?
- 3.5.4. Events: Definition und Erfassung
- 3.5.5. Entwicklung von Benutzerschnittstellen

3.6. Einführung in die gleichzeitige Programmierung
- 3.6.1. Einführung in die gleichzeitige Programmierung
- 3.6.2. Das Konzept des Prozesses und des Threads
- 3.6.3. Interaktion zwischen Prozessen oder Threads
- 3.6.4. Die Threads in C++
- 3.6.5. Vor- und Nachteile der gleichzeitigen Programmierung

3.7. Thread-Management und Synchronisierung
- 3.7.1. Lebenszyklus eines Threads
- 3.7.2. Die *Threadklasse*
- 3.7.3. Planung von Threads
- 3.7.4. Threadsgruppen
- 3.7.5. Dämonenartige Threads
- 3.7.6. Synchronisierung
- 3.7.7. Verriegelungsmechanismen
- 3.7.8. Mechanismen der Kommunikation
- 3.7.9. Monitoren

3.8. Häufige Probleme bei der gleichzeitigen Programmierung
- 3.8.1. Das Problem der Erzeuger/Verbraucher
- 3.8.2. Das Problem der Leser und der Schreiber
- 3.8.3. Das Problem mit dem Abendessen der Philosophen

3.9. Dokumentation und Prüfung von Software
- 3.9.1. Warum ist ein Software- Dokumentation wichtig?
- 3.9.2. Entwurfsdokumentation
- 3.9.3. Einsatz von Tools zur Dokumentation

3.10. Software-Tests
- 3.10.1. Einführung in Tests des Software
- 3.10.2. Arten von Tests
- 3.10.3. Einheitstest
- 3.10.4. Integrationstest
- 3.10.5. Validierungstest
- 3.10.6. Systemtest

Modul 4. Videospielkonsolen und -geräte

4.1. Geschichte der Programmierung von Videospielen
- 4.1.1. Atari-Zeit (1977-1985)
- 4.1.2. NES- und SNES-Zeit (1985-1995)
- 4.1.3. PlayStation / PlayStation 2-Zeit (1995-2005)
- 4.1.4. Xbox 360, PS3 und Wii-Zeit (2005-2013)
- 4.1.5. Xbox one, PS4 und Wii U-Zeit –(2005- jetzt)
- 4.1.6. Die Zukunft

4.2. Geschichte der Spielbarkeit in Videospielen
- 4.2.1. Einführung
- 4.2.2. Sozialer Kontext
- 4.2.3. Strukturelles Diagramm
- 4.2.4. Zukunft

4.3. Anpassung an die moderne Zeit
- 4.3.1. Bewegungsbasierte Spiele
- 4.3.2. Virtuelle Realität
- 4.3.3. Erweiterte Realität
- 4.3.4. Gemischte Realität

4.4. *Unity Scripting I* und Beispiele
- 4.4.1. Was ist ein Skript?
- 4.4.2. Unser erstes Skript
- 4.4.3. Hinzufügen eines Skripts
- 4.4.4. Öffnen eines Skripts
- 4.4.5. MonoBehaviour
- 4.4.6. *Debugging*

4.5. *Unity Scripting II* und Beispiele
- 4.5.1. Tastatur- und Mauseingabe
- 4.5.2. Raycast
- 4.5.3. Instanziierung
- 4.5.4. Variablen
- 4.5.5. Öffentliche und serialisierte Variablen

4.6. *Unity Scripting III* und Beispiele
- 4.6.1. Beschaffung von Komponenten
- 4.6.2. Änderung von Komponenten
- 4.6.3. Das Testen
- 4.6.4. Mehrere Objekte
- 4.6.5. *Colliders und Triggers*
- 4.6.6. Quaternionen

4.7. Peripheriegeräte
- 4.7.1. Entwicklung und Klassifizierung
- 4.7.2. Peripheriegeräte und Schnittstellen
- 4.7.3. Aktuelle Peripheriegeräte
- 4.7.4. Nahe Zukunft

4.8. Videospiele: Zukunftsperspektiven
- 4.8.1. Cloud-basiertes Spielen
- 4.8.2. Abwesenheit von Kontrolleuren
- 4.8.3. Immersive Realität
- 4.8.4. Andere Alternativen

4.9. Architektur
- 4.9.1. Besondere Bedürfnisse von Videospielen
- 4.9.2. Entwicklung der Architektur
- 4.9.3. Aktuelle Architektur
- 4.9.4. Unterschiede zwischen den Architekturen

4.10. Entwicklungskits und ihre Entwicklung
- 4.10.1. Einführung
- 4.10.2. Entwicklungskits der dritten Generation
- 4.10.3. Entwicklungskits der Vierte Generation
- 4.10.4. Entwicklungskits der Fünfte Generation
- 4.10.5. Entwicklungskits der sechsthäufigste Generation

Modul 5. Software-Entwicklung

5.1. Einführung in die Softwaretechnik und -modellierung
- 5.1.1. Das Wesen der Software
- 5.1.2. Die Einzigartigkeit von Webapps
- 5.1.3. Software-Entwicklung
- 5.1.4. Der Software-Prozess
- 5.1.5. Die Praxis der Softwareentwicklung
- 5.1.6. Software-Mythen
- 5.1.7. Wie fängt das alles an?
- 5.1.8. Objektorientierte Konzepte
- 5.1.9. Einführung in UML

5.2. Der Software-Prozess
- 5.2.1. Ein allgemeines Prozessmodell
- 5.2.2. Vorgeschriebene Prozessmodelle
- 5.2.3. Spezialisierte Prozessmodelle
- 5.2.4. Der einheitliche Prozess
- 5.2.5. Personal- und Teamprozessmodelle
- 5.2.6. Was ist Gewandtheit?
- 5.2.7. Was ist ein wendiger Prozess?
- 5.2.8. Scrum
- 5.2.9. Werkzeugkasten für agile Prozesse

5.3. Grundsätze für die Praxis der Softwareentwicklung
- 5.3.1. Leitprinzipien des Prozesses
- 5.3.2. Grundsätze für die Praxis
- 5.3.3. Grundsätze der Kommunikation
- 5.3.4. Grundsätze der Planung
- 5.3.5. Grundsätze der Modellierung
- 5.3.6. Grundsätze der Konstruktion
- 5.3.7. Grundsätze der Entfaltung

5.4. Verstehen der Anforderungen
- 5.4.1. Anforderungsmanagement
- 5.4.2. Festlegung der Grundlagen
- 5.4.3. Erkundung der Anforderungen
- 5.4.4. Entwicklung von Anwendungsfällen
- 5.4.5. Ausarbeitung des Anforderungsmodells
- 5.4.6. Verhandlung der Anforderungen
- 5.4.7. Validierung der Anforderungen

5.5. Anforderungsmodellierung: Szenarien, Informations- und Analyseklassen
- 5.5.1. Analyse der Anforderungen
- 5.5.2. Szenariobasierte Modellierung
- 5.5.3. UML-Modelle, die den Anwendungsfall liefern
- 5.5.4. Konzepte der Datenmodellierung
- 5.5.5. Klassenbasierte Modellierung
- 5.5.6. Klassendiagramme

5.6. Anforderungsmodellierung: Fluss, Verhalten und Muster
- 5.6.1. Strategien zur Gestaltung von Anforderungen
- 5.6.2. Flussorientierte Modellierung
- 5.6.3. Zustandsdiagramme
- 5.6.4. Erstellung eines Verhaltensmodells
- 5.6.5. Sequenzdiagramme
- 5.6.6. Kommunikationsdiagramme
- 5.6.7. Muster für die Modellierung von Anforderungen

5.7. Designkonzepte
- 5.7.1. Design im Kontext der Softwareentwicklung
- 5.7.2. Der Entwurfsprozess
- 5.7.3. Designkonzepte
- 5.7.4. Objektorientierte Entwurfskonzepte
- 5.7.5. Das Entwurfsmodell

5.8. Entwurf der Architektur
- 5.8.1. Softwarearchitektur
- 5.8.2. Architektonische Genres
- 5.8.3. Architektonische Stile
- 5.8.4. Achitektonischer Entwurf
- 5.8.5. Entwicklung von alternativen Entwürfen für die Architektur
- 5.8.6. Kartierung der Architektur mit Hilfe von Datenflüssen

5.9. Entwurf auf Komponentenebene und musterbasierter Entwurf
- 5.9.1. Was ist eine Komponente?
- 5.9.2. Klassenbasiertes Komponentendesign
- 5.9.3. Erstellung des Entwurfs auf Komponentenebene
- 5.9.4. Entwurf von traditionellen Komponenten
- 5.9.5. Komponentenbasierte Entwicklung
- 5.9.6. Entwurfsmuster
- 5.9.7. Musterbasierter Softwareentwurf
- 5.9.8. Architektonische Muster
- 5.9.9. Entwurfsmuster auf Komponentenebene
- 5.9.10. Entwurfsmuster für Benutzerschnittstellen

5.10. Softwarequalität und Projektmanagement
- 5.10.1. Qualität
- 5.10.2. Qualität der Software
- 5.10.3. Das Dilemma der Softwarequalität
- 5.10.4. Erreichen von Softwarequalität
- 5.10.5. Software-Qualitätssicherung
- 5.10.6. Das Verwaltungsspektrum
- 5.10.7. Personal
- 5.10.8. Das Produkt
- 5.10.9. Der Prozess
- 5.10.10. Das Projekt
- 5.10.11. Grundsätze und Praktiken

Modul 6. Videospiel-Engines

6.1. Videospiele und IKT
- 6.1.1. Einführung
- 6.1.2. Möglichkeiten
- 6.1.3. Herausforderungen
- 6.1.4. Schlussfolgerungen

6.2. Geschichte der Videospiel-Engines
- 6.2.1. Einführung
- 6.2.2. Atari Epoche
- 6.2.3. Die 1980er Jahre
- 6.2.4. Erste Engines Die 1990er Jahre
- 6.2.5. Aktuelle Eigines

6.3. Videospiel-Engines
- 6.3.1. Enginestypen
- 6.3.2. Teile einer Videospiel-Engine
- 6.3.3. Aktuelle Eigines
- 6.3.4. Auswahl eines Engines für unser Projekt

6.4. *Motor Game Maker*
- 6.4.1. Einführung
- 6.4.2. Entwurf eines Szenarios
- 6.4.3. *Sprites* und Animationen
- 6.4.4. Kollisionen
- 6.4.5. Scripting in GML

6.5. Motor Unreal Engine 4: Einführung
- 6.5.1. Was ist Unreal Engine 4? Was ist seine Philosophie?
- 6.5.2. Materialien
- 6.5.3. UI
- 6.5.4. Animationen
- 6.5.5. Partikelsystem
- 6.5.6. Künstliche Intelligenz
- 6.5.7. FPS

6.6. Motor Unreal Engine 4: Visual Scripting
- 6.6.1. Philosophie von den *Blueprints* und *Visual Scripting*
- 6.6.2. *Debugging*
- 6.6.3. Arten von Variablen
- 6.6.4. Grundlegende Flusskontrolle

6.7. Motor Unity 5
- 6.7.1. Programmierung in C# und Visual Studio
- 6.7.2. Erstellung von *Prefabs*
- 6.7.3. Verwendung von Gizmos zur Steuerung von Videospielen
- 6.7.4. Adaptiver Motor: 2D y 3D

6.8. Godot Motor
- 6.8.1. Godots Design-Philosophie
- 6.8.2. Objektorientierter Entwurf und Komposition
- 6.8.3. Alles in einem Paket
- 6.8.4. Freie und von der Gemeinschaft betriebene Software

6.9. Motor RPG Maker
- 6.9.1. Philosophie von RPH Maker
- 6.9.2. Als Referenz genommen
- 6.9.3. Ein Spiel mit Persönlichkeit schaffen
- 6.9.4. Erfolgreiche kommerzielle Spiele

6.10. Source 2 Motor
- 6.10.1. Philosopie con Source 2
- 6.10.2. Source y Source 2: Entwicklung
- 6.10.3. Gemeinschaftsnutzung: audiovisuelle Inhalte und Videospiele
- 6.10.4. Zukunft von Motor Source 2
- 6.10.5. Mods und erfolgreiche Spiele

Modul 7. Intelligente Systeme

7.1. Agententheorie
- 7.1.1. Geschichte des Kozepts
- 7.1.2. Definition von Agent
- 7.1.3. Agenten in künstlicher Intelligenz
- 7.1.4. Agenten in Software-Entwicklung

7.2. Agenten-Architekturen
- 7.2.1. Der Denkprozess eines Agenten
- 7.2.2. Reaktiver Agent
- 7.2.3. Deduktive Agenten
- 7.2.4. Hybride Agenten
- 7.2.5. Vergleich

7.3. Information und Kenntnisse
- 7.3.1. Unterscheidung zwischen Daten, Informationen und Wissen
- 7.3.2. Bewertung der Datenqualität
- 7.3.3. Methoden zur Datenerfassung
- 7.3.4. Methoden der Informationsbeschaffung
- 7.3.5. Methoden des Wissenserwerbs

7.4. Darstellung von Wissen
- 7.4.1. Die Wichtigkeit der Wissensdarstellung
- 7.4.2. Definition der Wissensdarstellung durch ihre Rollen
- 7.4.3. Merkmale einer Wissensrepräsentation

7.5. Ontologien
- 7.5.1. Einführung in die Metadaten
- 7.5.2. Philosophischer Begriff der Ontologie
- 7.5.3. Computergestütztes Konzept der Ontologie
- 7.5.4. Bereichsontologien und Ontologien auf höherer Ebene
- 7.5.5. Wie baut man eine Ontologie auf?

7.6. Ontologiesprachen und Software für die Erstellung von Ontologien
- 7.6.1. Drillinge RDF, Turtle und N3
- 7.6.2. RDF Schema
- 7.6.3. OWL
- 7.6.4. SPARQL
- 7.6.5. Einführung in die verschiedenen Tools für die Erstellung von Ontologien
- 7.6.6. Installation und Verwendung von Protégé

7.7. Semantisches Web
- 7.7.1. Der aktuelle Stand und die Zukunft des semantischen Webs
- 7.7.2. Semantische Webanwendungen

7.8. Die aderen Modelle der Wissensdarstellung
- 7.8.1. Wortschatz
- 7.8.2. Globale Vision
- 7.8.3. Taxonomien
- 7.8.4. Thesauri
- 7.8.5. Folksonomien
- 7.8.6. Vergleich
- 7.8.7. Mind Maps

7.9. Bewertung und Integration von Wissensrepräsentationen
- 7.9.1. Null-Ordnung-Logik
- 7.9.2. Logik erster Ordnung
- 7.9.3. Beschreibungslogik
- 7.9.4. Beziehung zwischen verschiedenen Arten von Logik
- 7.9.5. Prolog: Programmierung auf der Grundlage der Logik erster Ordnung

7.10. Semantische Reasoner, wissensbasierte Systeme und Expertensysteme
- 7.10.1. Begriff des Reasoners
- 7.10.2. Anwendungen eines Reasoners
- 7.10.3. Wissensbasierte Systeme
- 7.10.4. MYCIN, Geschichte der Expertensysteme
- 7.10.5. Elemente und Architektur von Expertensystemen
- 7.10.6. Erstellung von Expertensystemen

Modul 8. Echtzeit- Programmierung

8.1. Grundlagen der gleichzeitigen Programmierung
- 8.1.1. Grundlegende Konzepte
- 8.1.2. Zusammentreffen
- 8.1.3. Vorteile des Zusammentreffens
- 8.1.4. Zusammentreffen und Hardware

8.2. Grundlegende Strukturen zur Unterstützung der Gleichzeitigkeit in Java
- 8.2.1. Java-Zusammentreffen
- 8.2.2. Erstellung von *Threads*
- 8.2.3. Methoden
- 8.2.4. Synchronisierung

8.3. *Threads*, Lebenszyklus, Prioritäten, Unterbrechungen, Zustände, Executors
- 8.3.1. *Threads*
- 8.3.2. Lebenszyklus
- 8.3.3. Prioritäten
- 8.3.4. Unterbrechungen
- 8.3.5. Zustände
- 8.3.6. Umsetzer

8.4. Gegenseitiger Ausschluss
- 8.4.1. Was ist der gegenseitige Ausschluss?
- 8.4.2. Dekker-Algorithmus
- 8.4.3. Peterson-Algorithmus
- 8.4.4. Java-Gegenseitiger Ausschluss

8.5. Abhängigkeit von Status
- 8.5.1. Injektion von Abhängigkeiten
- 8.5.2. Implementierung des Musters in Java
- 8.5.3. Wege zur Injektion von Abhängigkeiten
- 8.5.4. Beispiel

8.6. Entwurfsmuster
- 8.6.1. Einführung
- 8.6.2. Schaffungsmuster
- 8.6.3. Strukturmuster
- 8.6.4. Verhaltensmuster

8.7. Verwendung von Java-Bibliotheken
- 8.7.1. Was sind Bibliotheken 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. Programmierung von *Shaders*
- 8.8.1. Pipeline 3D und Rastern
- 8.8.2. Vertex Shading
- 8.8.3. Pixel *Shading: Beleuchtung I*
- 8.8.4. Pixel *Shading: Beleuchtung II*
- 8.8.5. Post-Effekte

8.9. Echtzeit- Programmierung
- 8.9.1. Einführung
- 8.9.2. Verarbeitung von Unterbrechungen
- 8.9.3. Synchronisierung und Kommunikation zwischen Prozessen
- 8.9.4. Planungssysteme in Echtzeit

8.10. Echtzeit Planung
- 8.10.1. Konzepte
- 8.10.2. Referenzmodell für Echtzeitsysteme
- 8.10.3. Planungsmaßnahmen
- 8.10.4. Zyklische Planer
- 8.10.5. Planer mit statischen Eigenschaften
- 8.10.6. Planer mit Dynamische Eigenschaften

Modul 9. Design und Entwicklung von Webspielen

9.1. Ursprünge und Standards des Webs
- 9.1.1. Ursprung von Internet
- 9.1.2. Erstellung von *World Wide Web*
- 9.1.3. Entstehung von Webstandards
- 9.1.4. Der Aufstieg von Webstandards

9.2. HTTP und Client-Server-Struktur
- 9.2.1. Client-Server-Rolle
- 9.2.2. Client-Server- Kommunikation
- 9.2.3. Jüngere Geschichte
- 9.2.4. Zentralisierte Computertechnik

9.3. Webprogrammierung: Einführung
- 9.3.1. Grundlegende Konzepte
- 9.3.2. Vorbereiten eines Webservers
- 9.3.3. Grundlegende Konzepte der HTML5
- 9.3.4. HTML-Formate

9.4. Einführung in HTML und Beispiele
- 9.4.1. Geschichte von HTML5
- 9.4.2. HTML5 Elemente
- 9.4.3. APIS
- 9.4.4. CCS3

9.5. Dokument-Objektmodell
- 9.5.1. Was ist das Document Object Model?
- 9.5.2. Nutzung von DOCTYPE
- 9.5.3. Die Bedeutung der Validierung des HTML
- 9.5.4. Zugriff auf die Elemente
- 9.5.5. Elemente und Texte erstellen
- 9.5.6. Verwendung von innerHTML
- 9.5.7. Ein Textelement oder einen Knoten löschen
- 9.5.8. Lesen und Schreiben der Attribute eines Elements
- 9.5.9. Manipulation von Elementstilen
- 9.5.10. Mehrere Dateien auf einmal anhängen

9.6. Einführung in CSS und Beispiele
- 9.6.1. Sintaxis CSS3
- 9.6.2. Formatvorlagen
- 9.6.3. Kennzeichnung
- 9.6.4. Selektoren
- 9.6.5. Webgestaltung mit CSS

9.7. Einführung in JavaScript und Beispiele
- 9.7.1. Was ist JavaScript?
- 9.7.2. Eine kurze Geschichte der Sprache
- 9.7.3. JavaScript-Versionen
- 9.7.4. Ein Dialogfeld anzeigen
- 9.7.5. JavaScript-Syntax
- 9.7.6. Verstehen von Skripten
- 9.7.7. Räume
- 9.7.8. Kommentare
- 9.7.9. Funktionen
- 9.7.10. On-Page und externes JavaScript

9.8. Funktionen in JavaScript
- 9.8.1. Funktionsanweisungen
- 9.8.2. Funktionausdrücke
- 9.8.3. Aufrufen von Funktionen
- 9.8.4. Rekursion
- 9.8.5. Verschachtelte Funktionen und Abschlüsse
- 9.8.6. Beibehaltung der Variablen
- 9.8.7. Funktionen mit mehreren Kanälen
- 9.8.8. Namenskonflikte
- 9.8.9. Schließungen
- 9.8.10. Parameter einer Funktion

9.9. PlayCanvas für die Entwicklung von Webspielen
- 9.9.1. Was ist PlayCanvas?
- 9.9.2. Projektkonfiguration
- 9.9.3. Erstellen eines Objekts
- 9.9.4. Hinzufügen von physischen Elementen
- 9.9.5. Hinzufügen eines Modells
- 9.9.6. Ändern der Schwerkraft- und Szeneneinstellungen
- 9.9.7. Skripte ausführen
- 9.9.8. Steuerung der Kamera

9.10. Phaser für die Entwicklung von Webspielen
- 9.10.1. Was ist Phaser?
- 9.10.2. Ressourcen aufladen
- 9.10.3. Aufbau der Welt
- 9.10.4. Die Plattformen
- 9.10.5. Der Spieler
- 9.10.6. Physische Elemente hinzufügen
- 9.10.7. Verwendung der Tastatur
- 9.10.8. *Pickups*
- 9.10.9. Punkte und Punktevergabe
- 9.10.10. Hüpfende Pumpen

Modul 10. Multiplayer-Netze und -Systeme

10.1. Die Geschichte und Entwicklung von Multiplayer-Videospiele
- 10.1.1. 70er Jahre: erste Multiplayer-Spiele
- 10.1.2. 90er Jahre: Duke Nukem, Doom, Quake
- 10.1.3. Aufschwung der Multiplayer-Videospiele
- 10.1.4. Lokaler und Online-Multiplayer
- 10.1.5. Partyspiele

10.2. Multiplayer-Geschäftsmodelle
- 10.2.1. Entstehung und Funktionsweise neuer Geschäftsmodelle
- 10.2.2. Online-Verkaufsdienste
- 10.2.3. Frei zum Spielen
- 10.2.4. Kleinstbeträge
- 10.2.5. Werbung
- 10.2.6. Abonnement mit monatlichen Zahlungen
- 10.2.7. Bezahlung pro Spiel
- 10.2.8. Testen Sie, bevor Sie kaufen

10.3. Lokale Spiele und Netzwerkspiele
- 10.3.1. Lokale Spiele: Anfänge
- 10.3.2. Partyspiele: Nintendo und das Zusammensein mit der Familie
- 10.3.3. Netzwerkspiele: Anfänge
- 10.3.4. Entwicklung von Netzwerkspielen

10.4. OSI-Modell Schichten I
- 10.4.1. OSI-Modell Einführung
- 10.4.2. Physikalische Schicht
- 10.4.3. Datenübertragungsschicht
- 10.4.4. Netzwerkschicht

10.5. OSI-Modell Schichten II
- 10.5.1. Transportschicht
- 10.5.2. Sitzungsschicht
- 10.5.3. Präsentationsschicht
- 10.5.4. Anwendungsschicht

10.6. Computernetzwerke und das Internet
- 10.6.1. Was ist ein Computernetz?
- 10.6.2. Software
- 10.6.3. Hardware
- 10.6.4. Server
- 10.6.5. Vernetzte Speicherung
- 10.6.6. Netzwerk-Protokolle

10.7. Mobile und drahtlose Netze
- 10.7.1. Mobiles Netz
- 10.7.2. Drahtloses Netzwerk
- 10.7.3. Betrieb von Mobilfunknetzen
- 10.7.4. Digitale Technologie

10.8. Sicherheit
- 10.8.1. Persönliche Sicherheit
- 10.8.2. *Hacks und cheats* in Videospielen
- 10.8.3. Einklemmschutz
- 10.8.4. Analyse von Einklemmschutzsystemen

10.9. Multiplayer-Systeme: Server
- 10.9.1. Server-Hosting
- 10.9.2. MMO-Videospiele
- 10.9.3. Dedizierte Videospielserver
- 10.9.4. LAN Parties

10.10. Design und Programmierung von Multiplayer-Videospielen
- 10.10.1. Grundlagen der Entwicklung von Multiplayer-Videospielen in Unreal
- 10.10.2. Grundlagen der Entwicklung von Multiplayer-Spielen in Unity
- 10.10.3. Wie Multiplayer-Spiele Spaß machen
- 10.10.4. Mehr als ein Controller: Innovation bei der Multiplayer-Steuerung

05

Methodik

Dieses Fortbildungsprogramm bietet eine andere Art des Lernens. Unsere Methodik wird durch eine zyklische Lernmethode entwickelt: ***das Relearning***.

Dieses Lehrsystem wird z. B. an den renommiertesten medizinischen Fakultäten der Welt angewandt und wird von wichtigen Publikationen wie dem ***New England Journal of Medicine*** als eines der effektivsten angesehen.

Entdecken Sie Relearning, ein System, das das herkömmliche lineare Lernen aufgibt und Sie durch zyklische Lehrsysteme führt: eine Art des Lernens, die sich als äußerst effektiv erwiesen hat, insbesondere in Fächern, die Auswendiglernen erfordern"

Fallstudie zur Kontextualisierung aller Inhalte

Unser Programm bietet eine revolutionäre Methode zur Entwicklung von Fähigkeiten und Kenntnissen. Unser Ziel ist es, Kompetenzen in einem sich wandelnden, wettbewerbsorientierten und sehr anspruchsvollen Umfeld zu stärken.

Mit TECH werden Sie eine Art des Lernens erleben, die die Grundlagen der traditionellen Universitäten in der ganzen Welt verschiebt"

Sie werden Zugang zu einem Lernsystem haben, das auf Wiederholung basiert, mit natürlichem und progressivem Unterricht während des gesamten Lehrplans.

Die Studenten lernen durch gemeinschaftliche Aktivitäten und reale Fälle die Lösung komplexer Situationen in realen Geschäftsumgebungen.

Eine innovative und andersartige Lernmethode

Dieses TECH-Programm ist ein von Grund auf neu entwickeltes, intensives Lehrprogramm, das die anspruchsvollsten Herausforderungen und Entscheidungen in diesem Bereich sowohl auf nationaler als auch auf internationaler Ebene vorsieht. Dank dieser Methodik wird das persönliche und berufliche Wachstum gefördert und ein entscheidender Schritt in Richtung Erfolg gemacht. Die Fallmethode, die Technik, die diesem Inhalt zugrunde liegt, gewährleistet, dass die aktuellste wirtschaftliche, soziale und berufliche Realität berücksichtigt wird.

> *Unser Programm bereitet Sie darauf vor, sich neuen Herausforderungen in einem unsicheren Umfeld zu stellen und in Ihrer Karriere erfolgreich zu sein"*

Die Fallstudienmethode ist das am weitesten verbreitete Lernsystem an den besten Business Schools der Welt, seit es sie gibt. Die Fallmethode wurde 1912 entwickelt, damit die Jurastudenten das Recht nicht nur anhand theoretischer Inhalte erlernen, sondern ihnen reale, komplexe Situationen vorlegen, damit sie fundierte Entscheidungen treffen und Werturteile darüber fällen können, wie diese zu lösen sind. Sie wurde 1924 als Standardlehrmethode in Harvard eingeführt.

Was sollte eine Fachkraft in einer bestimmten Situation tun? Mit dieser Frage konfrontieren wir Sie in der Fallmethode, einer handlungsorientierten Lernmethode. Während des gesamten Kurses werden Sie mit mehreren realen Fällen konfrontiert. Sie müssen Ihr gesamtes Wissen integrieren, recherchieren, argumentieren und Ihre Ideen und Entscheidungen verteidigen.

Relearning Methodik

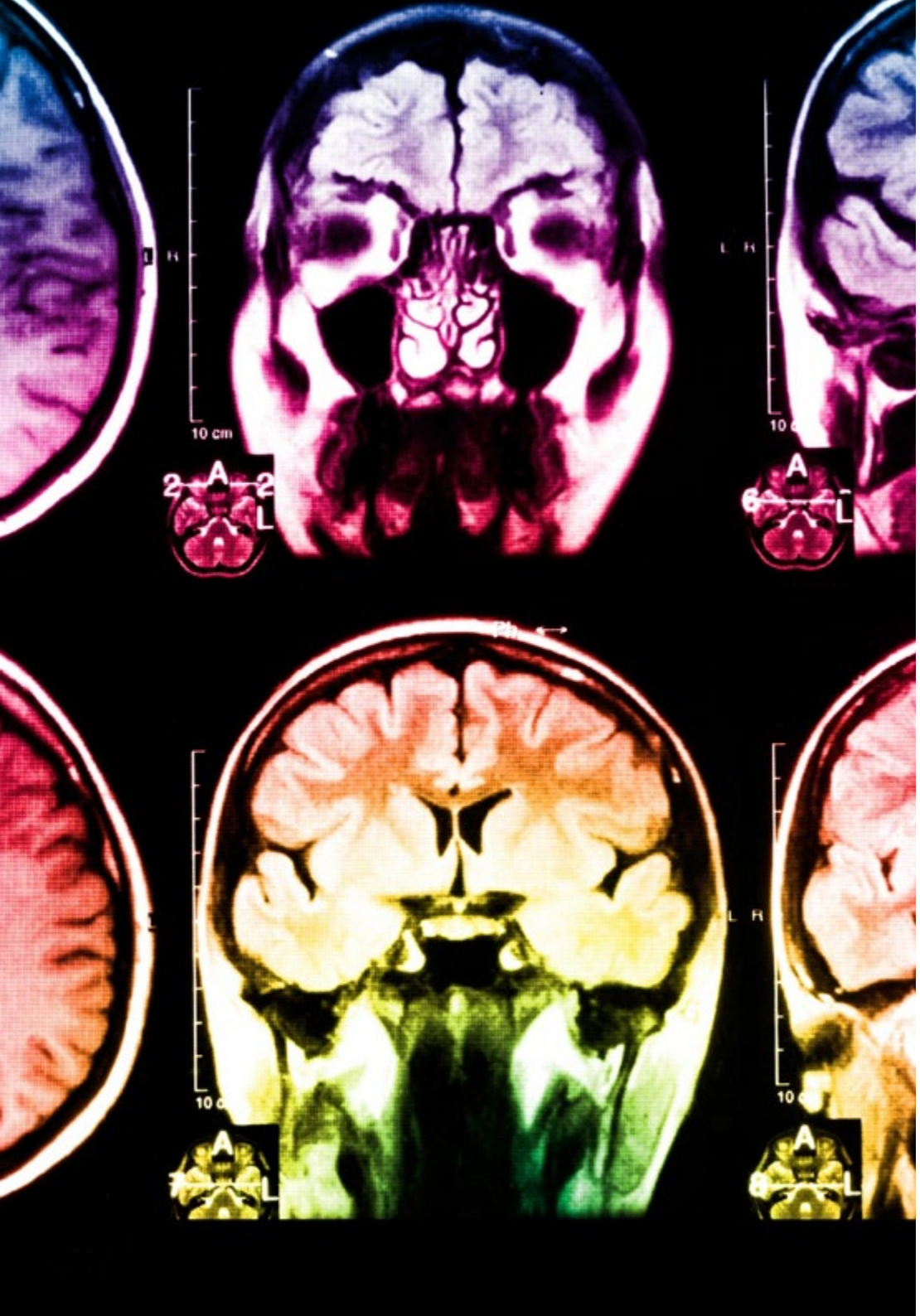
TECH kombiniert die Methodik der Fallstudien effektiv mit einem 100%igen Online-Lernsystem, das auf Wiederholung basiert und in jeder Lektion 8 verschiedene didaktische Elemente kombiniert.

Wir ergänzen die Fallstudie mit der besten 100%igen Online-Lehrmethode: Relearning.

Im Jahr 2019 erzielten wir die besten Lernergebnisse aller spanischsprachigen Online-Universitäten der Welt.

Bei TECH lernen Sie mit einer hochmodernen Methodik, die darauf ausgerichtet ist, die Führungskräfte der Zukunft auszubilden. Diese Methode, die an der Spitze der weltweiten Pädagogik steht, wird Relearning genannt.

Unsere Universität ist die einzige in der spanischsprachigen Welt, die für die Anwendung dieser erfolgreichen Methode zugelassen ist. Im Jahr 2019 ist es uns gelungen, die Gesamtzufriedenheit unserer Studenten (Qualität der Lehre, Qualität der Materialien, Kursstruktur, Ziele...) in Bezug auf die Indikatoren der besten Online-Universität in Spanisch zu verbessern.



In unserem Programm ist das Lernen kein linearer Prozess, sondern erfolgt in einer Spirale (lernen, verlernen, vergessen und neu lernen). Daher wird jedes dieser Elemente konzentrisch kombiniert. Mit dieser Methode wurden mehr als 650.000 Hochschulabsolventen mit beispiellosem Erfolg in so unterschiedlichen Bereichen wie Biochemie, Genetik, Chirurgie, internationales Recht, Managementfähigkeiten, Sportwissenschaft, Philosophie, Recht, Ingenieurwesen, Journalismus, Geschichte, Finanzmärkte und -Instrumente ausgebildet. Dies alles in einem sehr anspruchsvollen Umfeld mit einer Studentenschaft mit hohem sozioökonomischem Profil und einem Durchschnittsalter von 43,5 Jahren.

Das Relearning ermöglicht es Ihnen, mit weniger Aufwand und mehr Leistung zu lernen, sich mehr auf Ihr Fachgebiet einzulassen, einen kritischen Geist zu entwickeln, Argumente zu verteidigen und Meinungen zu kontrastieren: eine direkte Gleichung zum Erfolg.

Nach den neuesten wissenschaftlichen Erkenntnissen der Neurowissenschaften wissen wir nicht nur, wie wir Informationen, Ideen, Bilder und Erinnerungen organisieren, sondern auch, dass der Ort und der Kontext, in dem wir etwas gelernt haben, von grundlegender Bedeutung dafür sind, dass wir uns daran erinnern und es im Hippocampus speichern können, um es in unserem Langzeitgedächtnis zu behalten.

Auf diese Weise sind die verschiedenen Elemente unseres Programms im Rahmen des so genannten neurokognitiven kontextabhängigen E-Learnings mit dem Kontext verbunden, in dem der Teilnehmer seine berufliche Praxis entwickelt.

Dieses Programm bietet die besten Lehrmaterialien, die sorgfältig für Fachleute aufbereitet sind:

Studienmaterial

Alle didaktischen Inhalte werden von den Fachleuten, die den Kurs unterrichten werden, speziell für den Kurs erstellt, so dass die didaktische Entwicklung wirklich spezifisch und konkret ist.

Diese Inhalte werden dann auf das audiovisuelle Format angewendet, um die TECH-Online-Arbeitsmethode zu schaffen. Und das alles mit den neuesten Techniken, die dem Studenten qualitativ hochwertige Stücke aus jedem einzelnen Material zur Verfügung stellen.

Meisterklassen

Die Nützlichkeit der Expertenbeobachtung ist wissenschaftlich belegt.

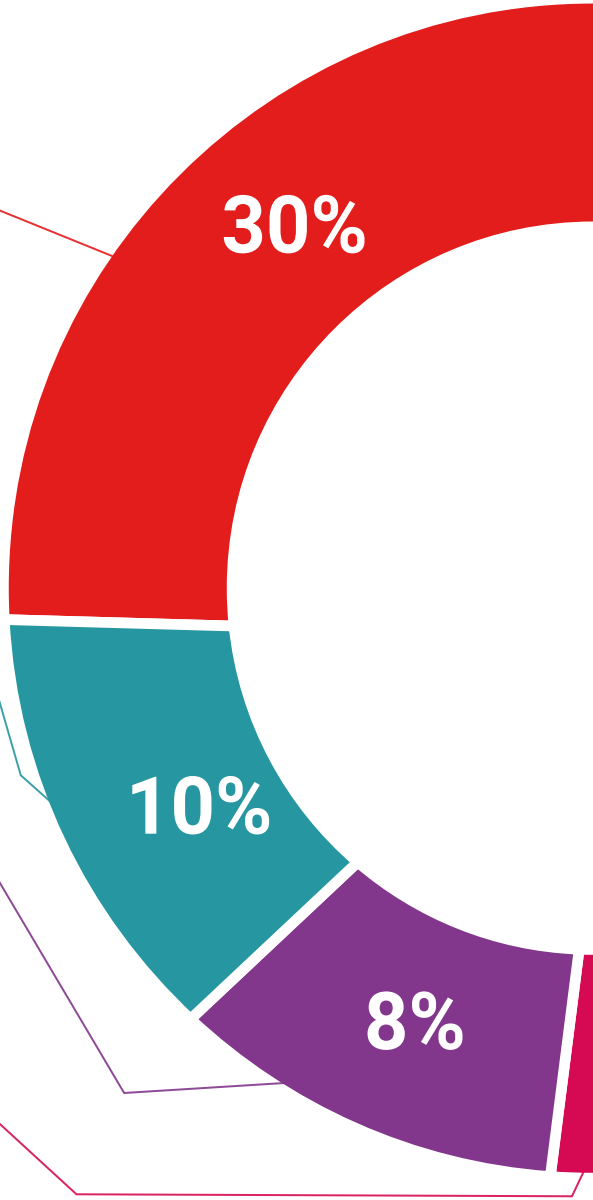
Das sogenannte Learning from an Expert baut Wissen und Gedächtnis auf und schafft Vertrauen für zukünftige schwierige Entscheidungen.

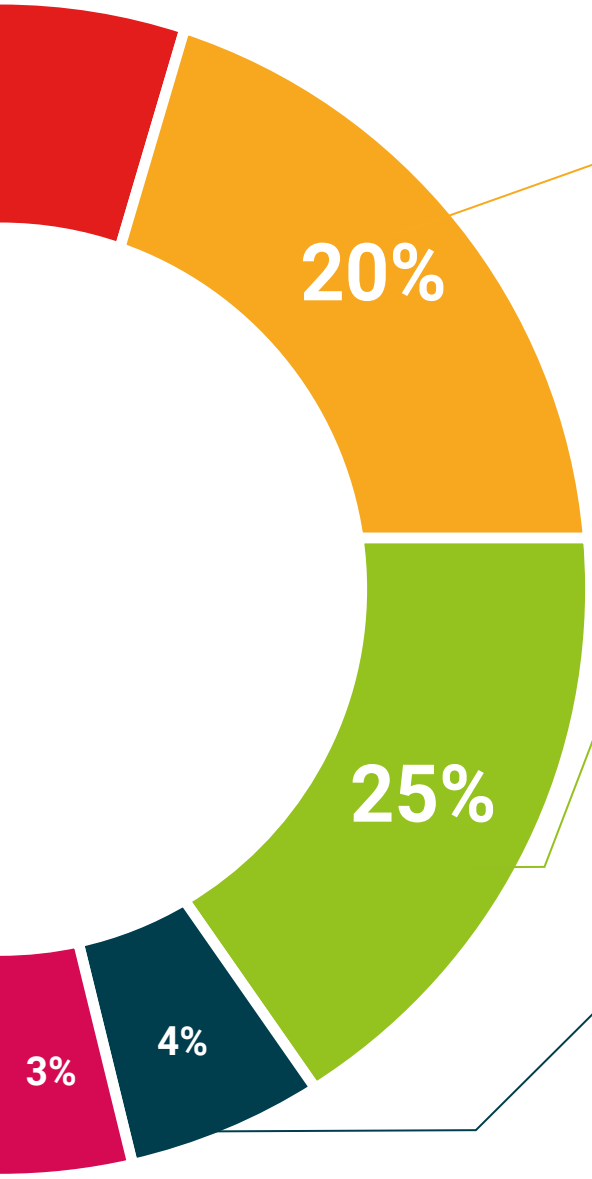
Fertigkeiten und Kompetenzen Praktiken

Sie werden Aktivitäten durchführen, um spezifische Kompetenzen und Fertigkeiten in jedem Fachbereich zu entwickeln. Praktiken und Dynamiken zum Erwerb und zur Entwicklung der Fähigkeiten und Fertigkeiten, die ein Spezialist im Rahmen der Globalisierung, in der wir leben, entwickeln muss.

Weitere Lektüren

Aktuelle Artikel, Konsensdokumente und internationale Leitfäden, u.a. In der virtuellen Bibliothek von TECH haben die Studenten Zugang zu allem, was sie für ihre Ausbildung benötigen.





Fallstudien

Sie werden eine Auswahl der besten Fallstudien vervollständigen, die speziell für diese Qualifizierung ausgewählt wurden. Die Fälle werden von den besten Spezialisten der internationalen Szene präsentiert, analysiert und betreut.

Interaktive Zusammenfassungen

Das TECH-Team präsentiert die Inhalte auf attraktive und dynamische Weise in multimedialen Pillen, die Audios, Videos, Bilder, Diagramme und konzeptionelle Karten enthalten, um das Wissen zu vertiefen.

Dieses einzigartige Bildungssystem für die Präsentation multimedialer Inhalte wurde von Microsoft als "europäische Erfolgsgeschichte" ausgezeichnet.

Prüfung und Nachprüfung

Die Kenntnisse der Studenten werden während des gesamten Programms regelmäßig durch Bewertungs- und Selbsteinschätzungsaktivitäten und -übungen beurteilt und neu bewertet, so dass die Studenten überprüfen können, wie sie ihre Ziele erreichen.

06

Qualifizierung

Der Privater Masterstudiengang in Videospielprogrammierung garantiert neben der strengsten und aktuellsten Ausbildung auch den Zugang zu einem von der TECH Technologischen Universität ausgestellten Diplom.

Schließen Sie dieses Programm erfolgreich ab und erhalten Sie Ihren Universitätsabschluss ohne lästige Reisen oder Formalitäten"

Dieser **Privater Masterstudiengang in Programmierung von Videospielen** enthält das vollständigste und aktuellste Programm auf dem Markt.

Sobald der Student die Prüfungen bestanden hat, erhält er/sie per Post* mit Empfangsbestätigung das entsprechende Diplom, ausgestellt von der **TECH Technologischen Universität.**

Das von **TECH Technologische Universität** ausgestellte Diplom drückt die erworbene Qualifikation aus und entspricht den Anforderungen, die in der Regel von Stellenbörsen, Auswahlprüfungen und Berufsbildungsausschüssen verlangt werden.

Titel: **Privater Masterstudiengang in Programmierung von Videospielen**

Anzahl der offiziellen Arbeitsstunden: **1.500 Std.**


tech technologische universität

Verleiht dieses

DIPLOM

an

Herr/Frau _____, mit Ausweis-Nr. _____

Für den erfolgreichen Abschluss und die Akkreditierung des Programms

PRIVATER MASTERSTUDIENGANG

in

Programmierung von Videospielen

Es handelt sich um einen von dieser Universität verliehenen Abschluss, mit einer Dauer von 1.500 Stunden, mit Anfangsdatum tt/mm/jjjj und Enddatum tt/mm/jjjj.

TECH ist eine private Hochschuleinrichtung, die seit dem 28. Juni 2018 vom Ministerium für öffentliche Bildung anerkannt ist.

Zum 17. Juni 2020

Tere Guevara Navarro
Rectora

Diese Qualifikation muss immer mit einem Hochschulabschluss einhergehen, der von der für die Berufsausübung zuständigen Behörde des jeweiligen Landes ausgestellt wurde. einzigartiger Code TECH: AFWOR23S techtitute.com/titel


Privater Masterstudiengang in Programmierung von Videospielen

Fachkategorie	Stunden
Obligatorisch (OB)	1.500
Wahlfach(OP)	0
Externes Praktikum (PR)	0
Masterarbeit (TFM)	0
	Summe 1.500

Allgemeiner Aufbau des Lehrplans

Kurs	Modul	Stunden	Kategorie
1°	Grundlagen der Programmierung	150	OB
1°	Datenstruktur und Algorithmen	150	OB
1°	Objektorientierte Programmierung	150	OB
1°	Videospielkonsolen und -geräte	150	OB
1°	Software-Entwicklung	150	OB
1°	Videospiel-Engines	150	OB
1°	Intelligente Systeme	150	OB
1°	Echtzeit- Programmierung	150	OB
1°	Design und Entwicklung von Webspielen	150	OB
1°	Multiplayer-Netze und -Systeme	150	OB

Tere Guevara Navarro
Rectora

tech technologische universität

*Haager Apostille. Für den Fall, dass der Student die Haager Apostille für sein Papierdiplom beantragt, wird TECH EDUCATION die notwendigen Vorkehrungen treffen, um diese gegen eine zusätzliche Gebühr zu beschaffen.

tech technologische universität

Privater Masterstudiengang
Programmierung von Videospielen

- Modalität: **online**
- Dauer: **12 Monate**
- Qualifizierung: **TECH Technologische Universität**
- Aufwand: **16 Std./Woche**
- Zeitplan: **in Ihrem eigenen Tempo**
- Prüfungen: **online**

Privater Masterstudiengang
Programmierung von Videospielen

tech technologische universität