

Mastère Spécialisé

Programmation de Jeux Vidéo



Mastère Spécialisé Programmation de Jeux Vidéo

- » Modalité: en ligne
- » Durée: 12 mois
- » Qualification: TECH Université Technologique
- » Intensité: 16h/semaine
- » Horaire: à votre rythme
- » Examens: en ligne

Accès au site: www.techtitute.com/fr/informatique/master/master-programmation-jeux-video

Sommaire

01

Présentation

page 4

02

Objectifs

page 8

03

Compétences

page 12

04

Structure et contenu

page 16

05

Méthodologie

page 32

06

Diplôme

page 40

01 Présentation

La plus grande attraction d'un jeu vidéo réside dans ses aspects les plus visuels, tels que ses graphismes ou ses conceptions. Mais ils ne pouvaient pas se démarquer sans programmation. La programmation est la clé de tout jeu vidéo, car elle détermine sa facilité de jeu ou la façon dont les graphiques interagissent avec le joueur. Sans une bonne programmation, tout jeu serait voué à l'échec, car il comporterait de nombreux bogues et ne serait pas une expérience agréable. Les entreprises en sont conscientes, c'est pourquoi elles ont besoin de développeurs de haut niveau. Ce diplôme répond à cette demande, car il prépare les étudiants à pouvoir faire face à tous les défis du secteur, de sorte qu'ils obtiendront de nombreuses opportunités professionnelles grâce à lui.



“

Les entreprises savent que la clé d'un jeu vidéo réside dans la programmation. Spécialisez-vous et devenez le développeur le plus recherché dans votre environnement"

Derrière chaque grand jeu vidéo se cache une énorme équipe de professionnels spécialisés dans chaque domaine de travail qui s'efforcent d'apporter le succès à leur entreprise. Normalement, les sections les plus remarquables pour les fans sont celles qu'ils peuvent percevoir directement, comme les Visuels ou celles liées au Contrôle du personnage, à la Mécanique ou à l'Interaction des Objets.

Cependant, pour que tous ces éléments fonctionnent et soient correctement intégrés, il existe une tâche essentielle qui n'est généralement pas prise en compte: la programmation. Le développement d'un jeu vidéo comporte différentes phases et implique différents départements, mais la programmation est celle qui donne un sens à tout et forme le squelette de base sur lequel le reste des domaines sera incorporé.

C'est pourquoi les entreprises du secteur accordent tant d'attention à cette question, car elles savent qu'un développement correct et efficace de leurs jeux vidéo facilitera l'avancement du projet et évitera l'apparition d'erreurs et *Bugs*. C'est pourquoi ils recherchent les meilleurs programmeurs spécialisés dans ce domaine.

Mais il n'est pas facile de trouver de véritables spécialistes dans ce domaine. Et ce Mastère Spécialisé en Programmation de Jeux Vidéo répond à cette demande, en faisant des étudiants de grands experts en développement de jeux vidéo qui peuvent évoluer dans l'industrie avec facilité, obtenant de grandes opportunités de carrière grâce aux compétences et capacités acquises tout au long de ce diplôme.

Ce **Mastère Spécialisé en Programmation de Jeux Vidéo** le programme éducatif le plus complet et le plus actuel du marché. Ses principales caractéristiques sont:

- ◆ Le développement de cas pratiques présentés par des experts en Programmation et de Développement de Jeux vidéo
- ◆ Les contenus graphiques, schématiques et éminemment pratiques avec lesquels ils sont conçus fournissent des informations scientifiques et sanitaires essentielles à la pratique professionnelle
- ◆ Des exercices pratiques où le processus d'auto-évaluation peut être réalisé pour améliorer l'apprentissage
- ◆ Il met l'accent sur les méthodologies innovantes
- ◆ Des cours théoriques, des questions à l'expert, des forums de discussion sur des sujets controversés et un travail de réflexion individuel
- ◆ La possibilité d'accéder aux contenus depuis n'importe quel appareil fixe ou portable doté d'une connexion internet



Développer toutes sortes de jeux vidéo dans les meilleures entreprises du monde grâce à ce Mastère Spécialisé"

“

La programmation est de plus en plus essentielle dans le développement d'un jeu vidéo. Devenez un élément essentiel de l'industrie grâce à ce diplôme"

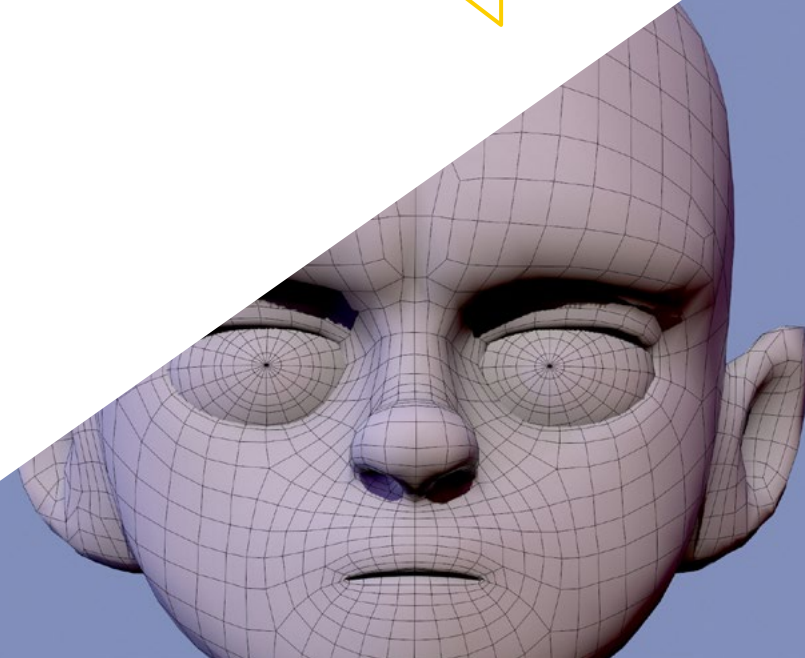
Le programme comprend, dans son corps enseignant, des professionnels du secteur qui apportent à cette formation l'expérience de leur travail, ainsi que des spécialistes reconnus de grandes sociétés et d'universités prestigieuses.

Grâce à son contenu multimédia développé avec les dernières technologies éducatives, les spécialistes bénéficieront d'un apprentissage situé et contextuel. Ainsi, ils se formeront dans un environnement simulé qui leur permettra d'apprendre en immersion et de s'entraîner dans des situations réelles.

La conception de ce programme est axée sur l'apprentissage par les problèmes, grâce auquel le professionnel doit essayer de résoudre les différentes situations de pratique professionnelle qui se présentent tout au long du cours académique. Pour ce faire, l'étudiant sera assisté d'un innovant système de vidéos interactives, créé par des experts reconnus.

Les jeux sont votre passion et vous voulez devenir un grand développeur. N'attendez pas plus longtemps et inscrivez-vous à ce Mastère Spécialisé.

Les meilleures entreprises du secteur vous attendent. Spécialisez-vous maintenant.



02 Objectifs

L'objectif principal de ce Mastère Spécialisé est de faire des étudiants de grands développeurs de jeux vidéo. Ce secteur est en pleine expansion et a besoin de plus en plus de programmeurs et de spécialistes ayant une formation de haut niveau. Ce diplôme est donc parfait pour de grandes opportunités de carrière dans certaines des entreprises les plus prestigieuses du monde. Ainsi, ce programme offre à ses étudiants toutes les compétences nécessaires pour devenir des experts très recherchés dans ce secteur, en réalisant des progrès significatifs et immédiats dans leur carrière.





“

Tous vos rêves sont désormais à votre portée grâce à ce Mastère Spécialisé en Programmation de Jeux Vidéo”



Objectifs généraux

- ◆ Apprendre les différents langages et méthodes de programmation appliqués aux jeux vidéo
- ◆ Approfondir le processus de production des jeux vidéo et l'intégration de la programmation dans ces étapes
- ◆ Pour apprendre les principes fondamentaux de la conception de jeux vidéo et les connaissances théoriques qu'un concepteur de jeux vidéo doit connaître
- ◆ Maîtriser les langages de programmation de base utilisés dans les jeux vidéo
- ◆ Appliquer les connaissances du génie logiciel et de la programmation spécialisée aux jeux vidéo
- ◆ Comprendre le rôle de la programmation dans le développement d'un jeu vidéo
- ◆ Connaître les différentes consoles et plateformes existantes
- ◆ Développer des jeux vidéo web et multijoueurs



Lorsque vous aurez terminé ce diplôme, vous serez le meilleur développeur de jeux vidéo de votre environnement"



Objectifs spécifiques

Module 1. Bases de la Programmation

- ◆ Comprendre la structure de base d'un ordinateur, les logiciels et les langages de programmation à usage général
- ◆ Analyser les éléments essentiels d'un programme informatique, tels que les différents types de données, les opérateurs, les expressions, les déclarations, les entrées/sorties et les déclarations de contrôle
- ◆ Interpréter des algorithmes, qui constituent la base nécessaire pour pouvoir développer des programmes informatiques

Module 2. Structure de Données et Algorithmes

- ◆ Apprenez les principales stratégies de conception d'algorithmes, ainsi que les différentes méthodes et mesures pour le calcul des algorithmes
- ◆ Distinguer le fonctionnement des algorithmes, leur stratégie et des exemples de leur utilisation dans les principaux problèmes connus
- ◆ Comprendre la technique du *Backtracking* et ses principales utilisations

Module 3. Programmation orientée Objets

- ◆ L'apprenant découvrira les différents modèles de conception pour les problèmes orientés objets
- ◆ Comprendre l'importance de la documentation et des tests dans le développement de Software
- ◆ Gérer l'utilisation du Threading et de la Synchronisation ainsi que la résolution des problèmes courants dans le cadre de la Programmation Concurrente

Module 4. Consoles et Dispositifs de Jeux Vidéo

- ◆ Connaître le fonctionnement de base des principaux périphériques d'entrée et de sortie
- ◆ Comprendre les principales implications de conception des différentes plateformes
- ◆ Étudier la structure, l'organisation, le fonctionnement et l'interconnexion des dispositifs et des systèmes
- ◆ Comprendre le rôle du système d'exploitation et des kits de développement pour les appareils mobiles et les plateformes de jeux vidéo

Module 5. Ingénierie du Logiciel

- ◆ Distinguer les bases de l'ingénierie logicielle, ainsi que le processus logiciel et les différents modèles pour son développement, y compris les technologies agiles
- ◆ Reconnaître l'ingénierie des exigences, son développement, son élaboration, sa négociation et sa validation afin de comprendre les principales normes liées à la qualité des logiciels et à la gestion de projet

Module 6. Moteurs de Jeux Vidéo

- ◆ Découvrir le fonctionnement et l'architecture d'un moteur de jeu vidéo
- ◆ Comprendre les caractéristiques de base des moteurs de jeu existants
- ◆ Applications de programme correctement et efficacement appliquées aux moteurs de jeux vidéo
- ◆ Choisir le paradigme et les langages de programmation les plus appropriés pour programmer des applications appliquées aux moteurs de jeux vidéo

Module 7. Systèmes Intelligents

- ◆ Établir les concepts liés à la théorie des agents, à l'architecture des agents et à leur processus de raisonnement
- ◆ Assimiler la théorie et la pratique des concepts d'information et de connaissance, ainsi que les différentes manières de représenter la connaissance
- ◆ Comprendre le fonctionnement des raisonneurs sémantiques, des systèmes à base de connaissances et des systèmes experts

Module 8. Programmation en Temps Réel

- ◆ Analyser les principales caractéristiques d'un langage de programmation en temps réel qui le différencie d'un langage de programmation traditionnel
- ◆ Comprendre les concepts de base des systèmes informatiques
- ◆ Acquérir la capacité d'appliquer les principales bases et techniques de la programmation en temps réel

Module 9. Conception et Développement de jeux Web

- ◆ L'apprenant sera capable de concevoir des jeux et des applications Web interactives avec la documentation correspondante
- ◆ Évaluer les principales caractéristiques des jeux et des applications Web interactives pour communiquer de manière professionnelle et correcte

Module 10. Réseaux et Systèmes Multi-joueurs

- ◆ Décrire l'architecture du protocole de contrôle de transmission/protocole Internet (TCP/IP) et le fonctionnement de base des réseaux sans fil
- ◆ Analyse de la sécurité appliquée aux jeux vidéo
- ◆ Acquérir la capacité de développer des jeux en ligne multi-joueurs

03

Compétences

Ce Mastère Spécialisé en Programmation de Jeux Vidéo fait des étudiants de véritables spécialistes du développement de ce type d'œuvres audiovisuelles grâce aux compétences et aux capacités qu'il leur confère. Ainsi, grâce à cet excellent programme, les étudiants obtiendront une série d'outils professionnels avec lesquels ils pourront relever tout type de défi lié à la programmation de jeux vidéo, devenant ainsi un personnel essentiel pour leurs entreprises.





“

*Vous maîtriserez tous les aspects
du développement de Jeux Vidéo”*

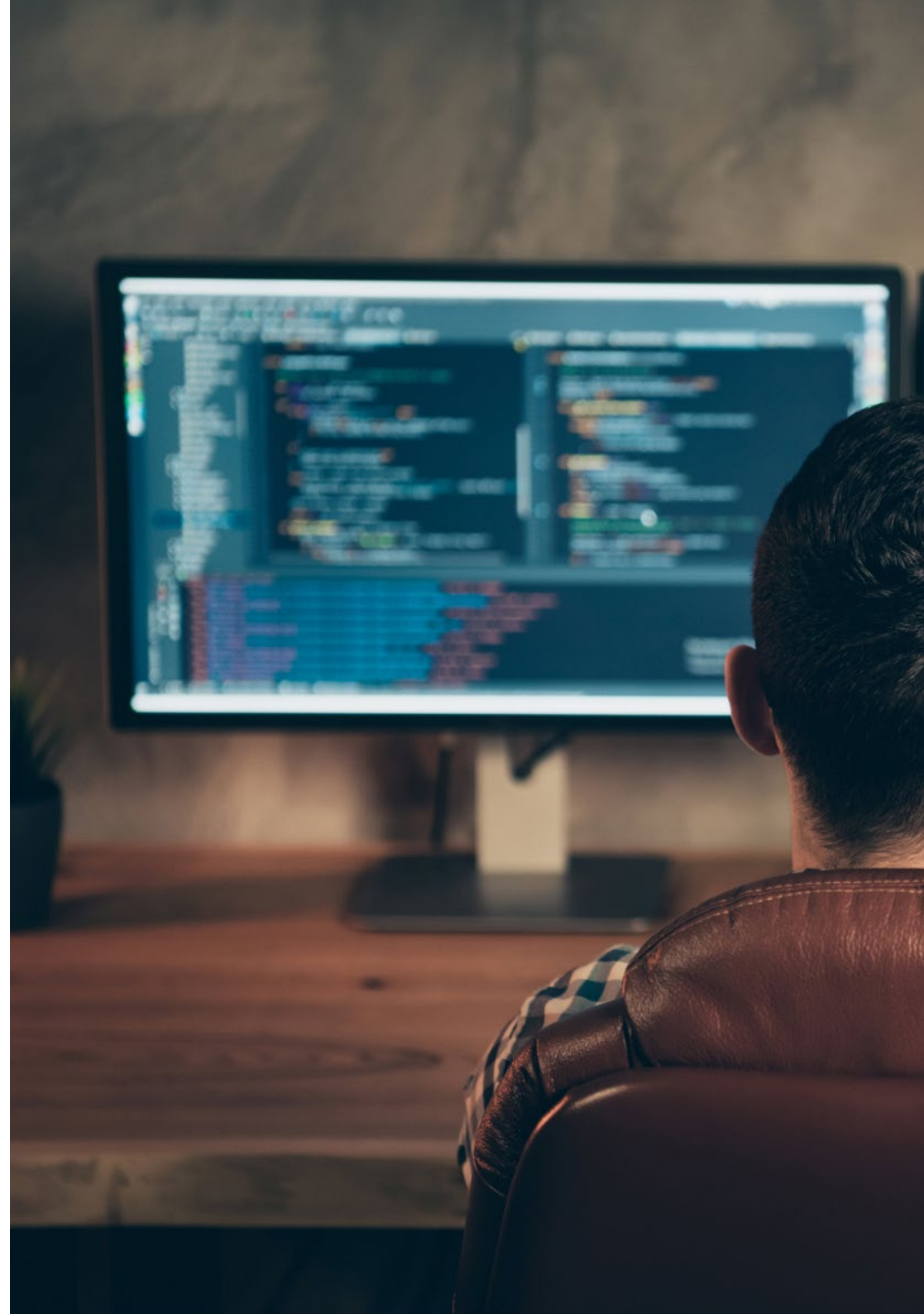


Compétences générales

- ◆ Concevoir toutes les phases d'un jeu vidéo, de l'idée initiale au lancement final
- ◆ Se spécialiser en tant que programmeur de jeux vidéo
- ◆ Étudier en profondeur toutes les parties du développement, depuis l'architecture initiale, la programmation du personnage du joueur et tous les éléments impliqués dans le processus du jeu
- ◆ Obtenir une vision globale du projet, en étant capable d'apporter des solutions aux différents problèmes et défis qui se posent dans la conception d'un jeu vidéo

“

Atteignez l'excellence en tant que programmeur de jeux vidéo grâce à ce Mastère Spécialisé”





Compétences spécifiques

- ◆ Connaître les logiciels nécessaires pour être un développeur professionnel de jeux vidéo
- ◆ Comprendre l'expérience du joueur et savoir analyser le gameplay du jeu vidéo
- ◆ Comprendre toutes les procédures théoriques et pratiques du processus de programmation de jeux vidéo
- ◆ Maîtriser les langages de programmation les plus utiles pour le monde du jeu vidéo
- ◆ Intégrer la programmation apprise à différents types de consoles et de plates-formes
- ◆ Programmer des jeux vidéo web et multijoueurs
- ◆ Assimiler le concept de moteur de jeu vidéo afin de pouvoir programmer correctement
- ◆ Appliquer les connaissances du génie logiciel à la programmation de jeux vidéo

04

Structure et contenu

Le contenu de ce Mastère Spécialisé en Programmation de Jeux Vidéo a été soigneusement conçu par une équipe de grands spécialistes du domaine qui connaissent parfaitement l'état actuel du secteur. Ainsi, grâce à ce programme, les étudiants pourront acquérir toutes les connaissances nécessaires pour pouvoir répondre aux demandes des entreprises du secteur, car ils ont été spécialement préparés à leurs particularités et spécificités, qui sont complexes et en constante évolution.





“

Ces contenus feront de vous un grand expert en Programmation de Jeux Vidéo”

Module 1. Bases de la Programmation

- 1.1. Introduction à la Programmation
 - 1.1.1. Structure de base d'un ordinateur
 - 1.1.2. Software
 - 1.1.3. Langages de programmation
 - 1.1.4. Cycle de vie de l'application informatique
- 1.2. Conception de l'algorithmes
 - 1.2.1. La résolution de problèmes
 - 1.2.2. Techniques descriptives
 - 1.2.3. Éléments et structure d'un algorithmes
- 1.3. Éléments d'un programme
 - 1.3.1. Origine et caractéristiques du langage C++
 - 1.3.2. L'environnement de développement
 - 1.3.3. Concept du programme
 - 1.3.4. Types de données fondamentales
 - 1.3.5. Opérateurs
 - 1.3.6. Expressions
 - 1.3.7. Phrases
 - 1.3.8. Entrée et sortie de données
- 1.4. Déclarations de contrôle
 - 1.4.1. Phrases
 - 1.4.2. Branches
 - 1.4.3. Boucles
- 1.5. Abstraction et modularité: Fonctions
 - 1.5.1. Conception modulaire
 - 1.5.2. Concept de fonction et d'utilité
 - 1.5.3. Définition d'une fonction
 - 1.5.4. Flux d'exécution dans l'appel d'une fonction
 - 1.5.5. Prototypes d'une fonction
 - 1.5.6. Retour des résultats
 - 1.5.7. Appel d'une fonction: Paramètres
 - 1.5.8. Passage de paramètres par référence et par valeur
 - 1.5.9. Identifiant du champ d'application
- 1.6. Structures de données statiques
 - 1.6.1. Arrays
 - 1.6.2. Tableaux Polyèdres
 - 1.6.3. Recherche et tri
 - 1.6.4. Cordes. Fonctions d'E/S pour les chaînes de caractères
 - 1.6.5. Structures Unions
 - 1.6.6. Nouveaux types de données
- 1.7. Structures de données dynamiques: Pointeurs
 - 1.7.1. Concept. Définition du pointeur
 - 1.7.2. Opérateurs et opérations avec des pointeurs
 - 1.7.3. Arrays de pointeurs
 - 1.7.4. Pointeurs et Arrays
 - 1.7.5. Pointeurs vers les cordes
 - 1.7.6. Pointeurs vers les structures
 - 1.7.7. Indirectivité multiple
 - 1.7.8. Pointeurs vers les fonctions
 - 1.7.9. Transmission de fonctions, de structures et de tableaux comme paramètres de fonctions
- 1.8. Fichiers
 - 1.8.1. Concepts de base
 - 1.8.2. Opérations sur les fichiers
 - 1.8.3. Types de fichiers
 - 1.8.4. Organisation des fichiers
 - 1.8.5. Introduction aux fichiers C++
 - 1.8.6. Traitement des fichiers
- 1.9. Récursion
 - 1.9.1. Définition de la récursion
 - 1.9.2. Types de récursions
 - 1.9.3. Avantages et inconvénients
 - 1.9.4. Considérations
 - 1.9.5. Conversion récursive-iterative
 - 1.9.6. La pile de récursion

- 1.10. Tests et documentation
 - 1.10.1. Test du programme
 - 1.10.2. Test boîte blanche
 - 1.10.3. Test de la boîte noire
 - 1.10.4. Outils de test
 - 1.10.5. Documentation de programmes

Module 2. Structure de Données et Algorithmes

- 2.1. Introduction aux stratégies de conception d'algorithmes
 - 2.1.1. Récursion
 - 2.1.2. Diviser pour mieux régner
 - 2.1.3. Autres stratégies
- 2.2. Efficacité et analyse des algorithmes
 - 2.2.1. Mesures d'efficacité
 - 2.2.2. Mesurer la taille de l'entrée
 - 2.2.3. Mesurer le temps d'exécution
 - 2.2.4. Cas le plus défavorable, le meilleur et le moyen
 - 2.2.5. Notation asymptotique
 - 2.2.6. Critères d'analyse mathématique pour les algorithmes non récursifs
 - 2.2.7. Analyse mathématique des algorithmes récursifs
 - 2.2.8. Analyse empirique des algorithmes
- 2.3. Algorithmes de tri
 - 2.3.1. Concept de tri
 - 2.3.2. Classement de la bulle
 - 2.3.3. Classement par sélection
 - 2.3.4. Classement par insertion
 - 2.3.5. Classement par fusion (*merge_sort*)
 - 2.3.6. Classement Rapide (*quick_sort*)
- 2.4. Algorithmes avec des arbres
 - 2.4.1. Concept d'arbre
 - 2.4.2. Arbres binaires
 - 2.4.3. Chemins d'arbres
 - 2.4.4. Représentation des expressions
 - 2.4.5. Arbres binaires ordonnés
 - 2.4.6. Arbres binaires équilibrés
- 2.5. Algorithmes avec *Heaps*
 - 2.5.1. Les *Heaps*
 - 2.5.2. L'Algorithme *Heapsort*
 - 2.5.3. Les files d'attente prioritaires
- 2.6. Algorithmes avec graphiques
 - 2.6.1. Représentation
 - 2.6.2. Traversée en largeur
 - 2.6.3. Déplacement en profondeur
 - 2.6.4. Classement topologique
- 2.7. Algorithmes *Greedy*
 - 2.7.1. La stratégie *Greedy*
 - 2.7.2. Éléments de la stratégie *Greedy*
 - 2.7.3. Changement de monnaie
 - 2.7.4. Problème du voyageur
 - 2.7.5. Problème de sac à dos
- 2.8. Recherche de chemin minimal
 - 2.8.1. Le problème du chemin minimal
 - 2.8.2. Arcs et cycles négatifs
 - 2.8.3. Algorithme de Dijkstra
- 2.9. Algorithmes *Greedy* sur les graphes
 - 2.9.1. L'arbre à couverture minimale
 - 2.9.2. L'algorithme de Prim
 - 2.9.3. L'algorithme de Kruskal
 - 2.9.4. Analyse de la complexité
- 2.10. *Backtracking*
 - 2.10.1. Le *Backtracking*
 - 2.10.2. Techniques alternatives

Module 3. Programmation orientée Objets

- 3.1. Introduction à la programmation orientée objet
 - 3.1.1. Introduction à la programmation orientée objet
 - 3.1.2. Conception des cours
 - 3.1.3. Introduction à UML pour les problèmes de modélisation
- 3.2. Relations entre les cours
 - 3.2.1. Abstraction et héritage
 - 3.2.2. Concepts d'héritage avancés
 - 3.2.3. Polymorphismes
 - 3.2.4. Composition et agrégation
- 3.3. Introduction aux patrons de conception pour les problèmes orientés objet
 - 3.3.1. Quels sont les modèles de conception?
 - 3.3.2. Modèle Factory
 - 3.3.4. Modèle Singleton
 - 3.3.5. Modèle Observer
 - 3.3.6. Modèle Composite
- 3.4. Exceptions
 - 3.4.1. Quelles sont les exceptions?
 - 3.4.2. Capture et traitement des exceptions
 - 3.4.3. Lancement d'exceptions
 - 3.4.4. Création d'exceptions
- 3.5. Interfaces d'utilisateurs
 - 3.5.1. Introduction à Qt
 - 3.5.2. Positionnement
 - 3.5.3. Quels sont les événements?
 - 3.5.4. Événements: définition et saisie
 - 3.5.5. Développement d'interfaces d'utilisateur
- 3.6. Introduction à la programmation concurrente
 - 3.6.1. Introduction à la programmation concurrente
 - 3.6.2. Le concept de processus et de fil
 - 3.6.3. Interaction entre processus ou fils
 - 3.6.4. Les fils en C++
 - 3.6.5. Avantages et inconvénients de la programmation concurrente
- 3.7. Gestion et synchronisation des fils
 - 3.7.1. Cycle de vie d'un fil
 - 3.7.2. La Classe *Thread*
 - 3.7.3. Planification du fil
 - 3.7.4. Groupes de fils
 - 3.7.5. Fils de type Daemon
 - 3.7.6. Synchronisation
 - 3.7.7. Mécanismes de verrouillage
 - 3.7.8. Mécanismes de communication
 - 3.7.9. Moniteurs
- 3.8. Problèmes courants en programmation concurrente
 - 3.8.1. Le problème du consommateur et du producteur
 - 3.8.2. Le problème des lecteurs et des écrivains
 - 3.8.3. Le problème du souper des philosophes
- 3.9. Documentation et test des logiciels
 - 3.9.1. Pourquoi la documentation des logiciels est-elle importante?
 - 3.9.2. Documentation sur la conception
 - 3.9.3. Utilisation des outils de documentation
- 3.10. Tests du logiciel
 - 3.10.1. Introduction aux tests logiciels
 - 3.10.2. Types de tests
 - 3.10.3. Tests unitaires
 - 3.10.4. Test d'intégration
 - 3.10.5. Test de validation
 - 3.10.6. Test du système

Module 4. Consoles et Dispositifs de Jeux Vidéo

- 4.1. Histoire de la programmation des jeux vidéo
 - 4.1.1. Période Atari (1977-1985)
 - 4.1.2. Période NES et SNES (1985-1995)
 - 4.1.3. Période PlayStation / PlayStation 2 (1995-2005)
 - 4.1.4. Période Xbox 360, PS3 et Wii (2005-2013)
 - 4.1.5. Xbox One, PS4 et Wii U - Période Switch (2013-présent)
 - 4.1.6. L'avenir
- 4.2. Histoire du gameplay dans les jeux vidéo
 - 4.2.1. Introduction
 - 4.2.2. Contexte social
 - 4.2.3. Diagramme structurel
 - 4.2.4. Futur
- 4.3. Adaptation aux temps modernes
 - 4.3.1. Jeux basés sur le mouvement
 - 4.3.2. Réalité virtuelle
 - 4.3.3. Réalité augmentée
 - 4.3.4. Réalité mixte
- 4.4. *Unity: Scripting I* et exemples
 - 4.4.1. Qu'est-ce qu'un *Script*?
 - 4.4.2. Notre premier *Script*
 - 4.4.3. Ajout d'un *Script*
 - 4.4.4. Ouverture d'un *Script*
 - 4.4.5. MonoBehaviour
 - 4.4.6. *Debugging*
- 4.5. *Unity: Scripting II* et exemples
 - 4.5.1. Saisie au clavier et à la souris
 - 4.5.2. Raycast
 - 4.5.3. Installation
 - 4.5.4. Variables
 - 4.5.5. Variables publiques et sérialisées
- 4.6. *Unity: Scripting III* et exemples
 - 4.6.1. Obtention des composants
 - 4.6.2. Modification des composants
 - 4.6.3. Test
 - 4.6.4. Objets multiples
 - 4.6.5. Collisionneurs et déclencheurs
 - 4.6.6. Quaternions
- 4.7. Périphériques
 - 4.7.1. Évolution et classification
 - 4.7.2. Périphériques et interfaces
 - 4.7.3. Périphériques actuels
 - 4.7.4. Futur proche
- 4.8. Jeux vidéo: perspectives d'avenir
 - 4.8.1. Jeux en nuage ("cloud-based gaming")
 - 4.8.2. Absence de conducteur
 - 4.8.3. Réalité immersive
 - 4.8.4. Autres alternatives
- 4.9. Architecture
 - 4.9.1. Besoins spécifiques des jeux vidéo
 - 4.9.2. Évolution de l'architecture
 - 4.9.3. Architecture actuelle
 - 4.9.4. Différences entre les architectures
- 4.10. Kits de développement et leur évolution
 - 4.10.1. Introduction
 - 4.10.2. Kits de développement de troisième génération
 - 4.10.3. Kits de développement de quatrième génération
 - 4.10.4. Kits de développement de cinquième génération
 - 4.10.5. Kits de développement de sixième génération

Module 5. Ingénierie du Logiciel

- 5.1. Introduction à l'ingénierie du logiciel et à la modélisation
 - 5.1.1. La nature du logiciel
 - 5.1.2. La nature unique des Webapps
 - 5.1.3. Ingénierie du Logiciel
 - 5.1.4. Le processus du logiciel
 - 5.1.5. La pratique de l'ingénierie du logiciel
 - 5.1.6. Mythes du logiciel
 - 5.1.7. Comment tout cela commence?
 - 5.1.8. Concepts orientés objet
 - 5.1.9. Introduction à UML
- 5.2. Le processus du logiciel
 - 5.2.1. Un modèle général de processus
 - 5.2.2. Modèles de processus prescriptifs
 - 5.2.3. Modèles de processus spécialisés
 - 5.2.4. Processus unifié
 - 5.2.5. Modèles de processus personnels et d'équipe
 - 5.2.6. Qu'est-ce que l'agilité?
 - 5.2.7. Qu'est-ce qu'un processus agile?
 - 5.2.8. Scrum
 - 5.2.9. Boîte à outils du processus Agile
- 5.3. Principes guidant la pratique de l'ingénierie du logiciel
 - 5.3.1. Principes guidant le processus
 - 5.3.2. Principes guidant la pratique
 - 5.3.3. Principes de communication
 - 5.3.4. Principes de planification
 - 5.3.5. Principes de modélisation
 - 5.3.6. Principes de construction
 - 5.3.7. Principes de déploiement



- 5.4. Compréhension des besoins
 - 5.4.1. Ingénierie des besoins
 - 5.4.2. Établir les bases
 - 5.4.3. Enquête sur les besoins
 - 5.4.4. Développement de cas d'utilisation
 - 5.4.5. Élaboration du modèle de besoins
 - 5.4.6. Négociation des besoins
 - 5.4.7. Validation des besoins
- 5.5. Modélisation des besoins: Scénarios, informations et types d'analyse
 - 5.5.1. Analyse des besoins
 - 5.5.2. Modélisation basée sur des scénarios
 - 5.5.3. Modèles UML fournissant le cas d'utilisation
 - 5.5.4. Concepts de modélisation de données
 - 5.5.5. Modélisation basée en cours
 - 5.5.6. Diagrammes de cours
- 5.6. Modélisation des besoins: Flux, comportement et schémas
 - 5.6.1. Stratégies de modélisation des exigences
 - 5.6.2. Modélisation orientée au flux
 - 5.6.3. Diagrammes d'état
 - 5.6.4. Création d'un modèle comportemental
 - 5.6.5. Diagrammes de séquence
 - 5.6.6. Diagrammes de communication
 - 5.6.7. Modèles pour la modélisation des exigences
- 5.7. Concepts de design
 - 5.7.1. La conception dans le contexte de l'ingénierie logicielle
 - 5.7.2. Le processus de design
 - 5.7.3. Concepts de design
 - 5.7.4. Concepts de conception orientée objet
 - 5.7.5. Le modèle de conception
- 5.8. Architecture de design
 - 5.8.1. Architecture du logiciel
 - 5.8.2. Genres architecturaux
 - 5.8.3. Styles architecturaux
 - 5.8.4. Conception architecturale
 - 5.8.5. Évolution des conceptions alternatives de l'architecture
 - 5.8.6. Cartographie de l'architecture à l'aide du flux de données
- 5.9. Conception au niveau des composants et des schémas
 - 5.9.1. Qu'est-ce qu'un composant?
 - 5.9.2. Conception de composants basée sur les cours
 - 5.9.3. Réalisation de la conception au niveau des composants
 - 5.9.4. Conception traditionnelle des composants
 - 5.9.5. Développement basé sur les composants
 - 5.9.6. Schémas de design
 - 5.9.7. Conception de logiciels basée sur des modèles
 - 5.9.8. Schémas architecturaux
 - 5.9.9. Schémas de conception au niveau des composants
 - 5.9.10. Modèles de conception d'interface utilisateur
- 5.10. Qualité des logiciels et gestion de projet
 - 5.10.1. Qualité
 - 5.10.2. Qualité du logiciel
 - 5.10.3. Le dilemme de la qualité du logiciel
 - 5.10.4. Atteindre la qualité du logiciel
 - 5.10.5. Assurance qualité du logiciel
 - 5.10.6. Le spectre de la gestion
 - 5.10.7. Le Personnel
 - 5.10.8. Le Produit
 - 5.10.9. Le processus
 - 5.10.10. Le projet
 - 5.10.11. Principes et pratiques

Module 6. Moteurs de Jeux Vidéo

- 6.1. Les jeux vidéo et les TIC
 - 6.1.1. Introduction
 - 6.1.2. Opportunités
 - 6.1.3. Défis
 - 6.1.4. Conclusions
- 6.2. Histoire des moteurs de jeux vidéo
 - 6.2.1. Introduction
 - 6.2.2. Période Atari
 - 6.2.3. Période des années 80
 - 6.2.4. Premiers moteurs. Période des années 90
 - 6.2.5. Moteurs actuels
- 6.3. Moteurs de Jeux Vidéo
 - 6.3.1. Types de moteurs
 - 6.3.2. Parties d'un moteur de jeu vidéo
 - 6.3.3. Moteurs actuels
 - 6.3.4. Sélection d'un moteur pour notre projet
- 6.4. Moteur *Game Maker*
 - 6.4.1. Introduction
 - 6.4.2. Conception de scénarios
 - 6.4.3. Sprites et Animations
 - 6.4.4. Collisions
 - 6.4.5. *Scripting* en GML
- 6.5. Moteur Unreal 4: Introduction
 - 6.5.1. Qu'est-ce qu'Unreal Engine 4? Quelle est sa philosophie?
 - 6.5.2. Matériaux
 - 6.5.3. UI
 - 6.5.4. Animations
 - 6.5.5. Système de particules
 - 6.5.6. Intelligence artificielle
 - 6.5.7. FPS
- 6.6. Moteur Unreal 4: *Visual Scripting*
 - 6.6.1. Philosophie des *Blueprints* et des *Visual Scripting*
 - 6.6.2. *Debugging*
 - 6.6.3. Types de variables
 - 6.6.4. Contrôle de flux de base
- 6.7. Moteur Unity 5
 - 6.7.1. Programmation en C# et Visual Studio
 - 6.7.2. Création de préfabriqués
 - 6.7.3. Utiliser des gadgets pour contrôler le jeu vidéo
 - 6.7.4. Moteur adaptatif: 2D et 3D
- 6.8. Moteur Godot
 - 6.8.1. Philosophie de conception de Godot
 - 6.8.2. Conception et composition orientées objet
 - 6.8.3. Paquet tout-en-un
 - 6.8.4. Logiciels libres et communautaires
- 6.9. Moteur RPG Maker
 - 6.9.1. Philosophie de RPG Maker
 - 6.9.2. Prise en référence
 - 6.9.3. Créer un jeu avec de la personnalité
 - 6.9.4. Jeux commerciaux réussis
- 6.10. Moteur source 2
 - 6.10.1. Philosophie de la source 2
 - 6.10.2. Source et Source 2: évolution
 - 6.10.3. Utilisation communautaire: Contenu audiovisuel et jeux vidéo
 - 6.10.4. L'avenir du moteur Source 2
 - 6.10.5. Mods et jeux réussis

Module 7. Systèmes Intelligents

- 7.1. Théorie des agents
 - 7.1.1. Histoire du concept
 - 7.1.2. Définition de l'agent
 - 7.1.3. Agents en intelligence artificielle
 - 7.1.4. Agents en ingénierie du logiciel
- 7.2. Architectures d'agents
 - 7.2.1. Le processus de raisonnement de l'agent
 - 7.2.2. Agents réactifs
 - 7.2.3. Agents déductifs
 - 7.2.4. Agents hybrides
 - 7.2.5. Comparaison
- 7.3. Informations et connaissances
 - 7.3.1. Distinction entre données, informations et connaissances
 - 7.3.2. Évaluation de la qualité des données
 - 7.3.3. Évaluation de la qualité des données
 - 7.3.4. Méthodes d'acquisition des informations
 - 7.3.5. Méthodes d'acquisition des connaissances
- 7.4. Représentation des connaissances
 - 7.4.1. L'importance de la représentation des connaissances
 - 7.4.2. Définition de la représentation des connaissances à travers ses rôles
 - 7.4.3. Caractéristiques d'une représentation des connaissances
- 7.5. Ontologies
 - 7.5.1. Introduction aux métadonnées
 - 7.5.2. Concept philosophique d'ontologie
 - 7.5.3. Concept informatiques d'ontologie
 - 7.5.4. Ontologies de domaine et ontologies de niveau supérieur
 - 7.5.5. Comment construire une ontologie
- 7.6. Langages d'ontologie et logiciels de création d'ontologies
 - 7.6.1. Triplés RDF, Turtle et N3
 - 7.6.2. Schéma RDF
 - 7.6.3. OWL
 - 7.6.4. SPARQL
 - 7.6.5. Introduction aux différents outils de création d'ontologies
 - 7.6.6. Installation et utilisation du Protégé
- 7.7. Le web sémantique
 - 7.7.1. L'état actuel et l'avenir du Web sémantique
 - 7.7.2. Applications du Web sémantique
- 7.8. Autres modèles de représentation de la connaissance
 - 7.8.1. Vocabulaires
 - 7.8.2. Vision globale
 - 7.8.3. Taxonomie
 - 7.8.4. Thésaurus
 - 7.8.5. Folksonomies
 - 7.8.6. Comparaison
 - 7.8.7. Cartes mentales
- 7.9. Évaluation et intégration des représentations de la connaissance
 - 7.9.1. Logique de l'ordre zéro
 - 7.9.2. Logique du premier ordre
 - 7.9.3. Logique descriptive
 - 7.9.4. Relation entre les différents types de logique
 - 7.9.5. Prologue: Programmation basée sur la logique du premier ordre
- 7.10. Raisonners sémantiques, systèmes à base de connaissances et systèmes experts
 - 7.10.1. Concept de Raisonneur
 - 7.10.2. Applications d'un raisonneur
 - 7.10.3. Systèmes basés sur la connaissance
 - 7.10.4. MYCIN, Histoire des systèmes experts
 - 7.10.5. Éléments et architecture des systèmes experts
 - 7.10.6. Création de systèmes experts

Module 8. Programmation en Temps Réel

- 8.1. Concepts de base de la programmation concurrente
 - 8.1.1. Concepts Fondamentaux
 - 8.1.2. Concurrency
 - 8.1.3. Avantages de la Concurrency
 - 8.1.4. Concurrency et Hardware
- 8.2. Structures de base du support de la concurrence en Java
 - 8.2.1. Concurrency en Java
 - 8.2.2. Création de *Threads*
 - 8.2.3. Méthodes
 - 8.2.4. Synchronisation
- 8.3. *Threads*, cycle de vie, priorités, interruptions, états, exécuteurs
 - 8.3.1. *Threads*
 - 8.3.2. Cycle de vie
 - 8.3.3. Priorités
 - 8.3.4. Interruptions
 - 8.3.5. Statues
 - 8.3.6. Mises en œuvre
- 8.4. Exclusion mutuelle
 - 8.4.1. Qu'est-ce que l'exclusion mutuelle?
 - 8.4.2. Algorithme de Dekker
 - 8.4.3. Algorithme de Peterson
 - 8.4.4. Exclusion mutuelle en Java
- 8.5. Unités d'État
 - 8.5.1. Injection de dépendances
 - 8.5.2. Mise en œuvre du modèle Java
 - 8.5.3. Méthodes d'injection des dépendances
 - 8.5.4. Exemple





- 8.6. Schémas de design
 - 8.6.1. Introduction
 - 8.6.2. Modèles de création
 - 8.6.3. Schémas de structure
 - 8.6.4. Schémas comportementaux
- 8.7. Utilisation des bibliothèques Java
 - 8.7.1. Que sont les bibliothèques Java?
 - 8.7.2. Mockito-All, Mockito-Core
 - 8.7.3. Goyave
 - 8.7.4. Commons-io
 - 8.7.5. Commons-Lang, Commons-Lang3
- 8.8. Programmation de shaders
 - 8.8.1. Pipeline 3D et Raster
 - 8.8.2. Vertex Shading
 - 8.8.3. Pixel Shading: Éclairage I
 - 8.8.4. Pixel Shading: Éclairage II
 - 8.8.5. Post-Effects
- 8.9. Programmation en temps réel
 - 8.9.1. Introduction
 - 8.9.2. Traitement des interruptions
 - 8.9.3. Synchronisation et communication entre les processus
 - 8.9.4. Les systèmes de planification en temps réel
- 8.10. Planification en temps réel
 - 8.10.1. Concepts
 - 8.10.2. Modèle de référence des systèmes en temps réel
 - 8.10.3. Politiques de planification
 - 8.10.4. Planificateurs cycliques
 - 8.10.5. Planificateurs avec des propriétés statiques
 - 8.10.6. Planificateurs avec des propriétés dynamiques

Module 9. Conception et Développement de jeux Web

- 9.1. Origines et normes du Web
 - 9.1.1. Origines d'Internet
 - 9.1.2. Création du World Wide Web
 - 9.1.3. Apparition des standards du Web
 - 9.1.4. L'essor des standards du Web
- 9.2. HTTP et structure client-serveur
 - 9.2.1. Rôle client-serveur
 - 9.2.2. Communication client-serveur
 - 9.2.3. Histoire récente
 - 9.2.4. Informatique centralisée
- 9.3. Programmation Web: Introduction
 - 9.3.1. Concepts de base
 - 9.3.2. Préparation d'un serveur Web
 - 9.3.3. Concepts de base du HTML5
 - 9.3.4. Formes HTML
- 9.4. Introduction au HTML et exemples
 - 9.4.1. Histoire du HTML5
 - 9.4.2. Éléments HTML5
 - 9.4.3. APIS
 - 9.4.4. CCS3
- 9.5. Modèle d'objet de document
 - 9.5.1. Qu'est-ce que le Document Object Model?
 - 9.5.2. Utilisation de DOCTYPE
 - 9.5.3. L'importance de la validation de l'HTML
 - 9.5.4. Accès aux éléments
 - 9.5.5. Création d'éléments et de textes
 - 9.5.6. Utilisation de InnerHTML
 - 9.5.7. Suppression d'un élément de texte ou d'un nœud de texte
 - 9.5.8. Lecture et écriture des attributs des éléments
 - 9.5.9. Manipulation des styles d'éléments
 - 9.5.10. Joindre plusieurs fichiers à la fois
- 9.6. Introduction au CSS et exemples
 - 9.6.1. Syntaxe CSS3
 - 9.6.2. Feuilles de style
 - 9.6.3. Étiquettes
 - 9.6.4. Sélecteurs
 - 9.6.5. Conception Web avec CSS
- 9.7. Introduction à Javascript et exemples
 - 9.7.1. Qu'est-ce que le Javascript?
 - 9.7.2. Brève histoire de la langue
 - 9.7.3. Versions de Javascript
 - 9.7.4. Afficher une boîte de dialogue
 - 9.7.5. Syntaxe du Javascript
 - 9.7.6. Compréhension de *Scripts*
 - 9.7.7. Espaces
 - 9.7.8. Commentaires
 - 9.7.9. Fonctions
 - 9.7.10. Javascript dans la page et externe
- 9.8. Fonctions de Javascript
 - 9.8.1. Déclarations de fonctions
 - 9.8.2. Expressions de fonctions
 - 9.8.3. Appeler les fonctions
 - 9.8.4. Récursion
 - 9.8.5. Fonctions imbriquées et fermetures

- 9.8.6. Préservation des variables
- 9.8.7. Fonctions multi-emboîtées
- 9.8.8. Conflits de noms
- 9.8.9. Clôtures ou Fermetures
- 9.8.10. Paramètres d'une fonction
- 9.9. PlayCanvas pour le développement de jeux Web
 - 9.9.1. Qu'est-ce que PlayCanvas?
 - 9.9.2. Configuration du projet
 - 9.9.3. Création d'un objet
 - 9.9.4. Ajout d'éléments physiques
 - 9.9.5. Ajout d'un modèle
 - 9.9.6. Modification des paramètres de gravité et de scène
 - 9.9.7. Exécution de *Scripts*
 - 9.9.8. Commandes de la caméra
- 9.10. Phaser pour le développement de jeux Web
 - 9.10.1. Qu'est-ce que Phaser?
 - 9.10.2. Chargement des ressources
 - 9.10.3. Construire le monde
 - 9.10.4. Les plateformes
 - 9.10.5. Le joueur
 - 9.10.6. Ajout d'éléments physiques
 - 9.10.7. Utilisation du clavier
 - 9.10.8. Collecter *Pickups*
 - 9.10.9. Points et notation
 - 9.10.10. Bombes rebondissantes

Module 10. Réseaux et Systèmes Multi-joueurs

- 10.1. Histoire et évolution des jeux vidéo multi-joueurs
 - 10.1.1. 1970s: Premiers jeux multi-joueurs
 - 10.1.2. 1990s: Duke Nukem, Doom, Quake
 - 10.1.3. Lessor des jeux vidéo multi-joueurs
 - 10.1.4. Multijoueur local et en ligne
 - 10.1.5. Jeux de société
- 10.2. Modèles d'entreprise multi-joueurs
 - 10.2.1. Origine et fonctionnement des modèles commerciaux émergents
 - 10.2.2. Services de vente en ligne
 - 10.2.3. Libre à jouer
 - 10.2.4. Micro paiements
 - 10.2.5. Publicité
 - 10.2.6. Abonnement avec paiements mensuels
 - 10.2.7. Payer pour jouer
 - 10.2.8. Essayez avant d'acheter
- 10.3. Jeux en ligne et jeux en réseau
 - 10.3.1. Jeux locaux: les débuts
 - 10.3.2. Jeux de société: Nintendo et la convivialité en famille
 - 10.3.3. Jeux en réseau: les débuts
 - 10.3.4. Évolution des jeux en réseau
- 10.4. Modèle OSI: Couches I
 - 10.4.1. Modèle OSI: Introduction
 - 10.4.2. Couche physique
 - 10.4.3. Couche liaison de données
 - 10.4.4. Couche réseau
- 10.5. Modèle OSI: Couches II
 - 10.5.1. Couche de transport
 - 10.5.2. Couche session
 - 10.5.3. Couche de présentation
 - 10.5.4. Couche d'application

- 10.6. Réseaux informatiques et Internet
 - 10.6.1. Qu'est-ce qu'un réseau informatique?
 - 10.6.2. Software
 - 10.6.3. Hardware
 - 10.6.4. Serveurs
 - 10.6.5. Stockage en réseau
 - 10.6.6. Protocoles de réseau
- 10.7. Réseaux mobiles et sans fil
 - 10.7.1. Réseau mobile
 - 10.7.2. Réseau sans fil
 - 10.7.3. Fonctionnement des réseaux mobiles
 - 10.7.4. Technologie numérique
- 10.8. Sécurité
 - 10.8.1. Sécurité personnelle
 - 10.8.2. Hacks et Cheats dans les jeux vidéo
 - 10.8.3. Sécurité Anti-Cheat
 - 10.8.4. Analyse des systèmes de sécurité anti-cheat
- 10.9. Systèmes multi-joueurs: Serveurs
 - 10.9.1. Hébergement de serveurs
 - 10.9.2. Jeux vidéo MMO
 - 10.9.3. Serveurs dédiés aux jeux vidéo
 - 10.9.4. Soirées LAN
- 10.10. Conception et programmation de jeux multi-joueurs
 - 10.10.1. Principes de base de la conception de jeux multi-joueurs Unreal
 - 10.10.2. Principes de base de la conception de jeux multi-joueurs dans Unity
 - 10.10.3. Comment rendre un jeu multi-joueurs amusant?
 - 10.10.4. Au-delà d'un contrôleur: Innovation dans les contrôles multi-joueurs





“

*Si vous souhaitez développer
une grande carrière dans la
programmation de Jeux Vidéo
de renommée mondiale, ce
diplôme est fait pour vous”*

05 Méthodologie

Ce programme de formation offre une manière différente d'apprendre. Notre méthodologie est développée à travers un mode d'apprentissage cyclique: ***le Relearning***.

Ce système d'enseignement est utilisé, par exemple, dans les écoles de médecine les plus prestigieuses du monde et a été considéré comme l'un des plus efficaces par des publications de premier plan telles que le ***New England Journal of Medicine***.



“

Découvrez Relearning, un système qui renonce à l'apprentissage linéaire conventionnel pour vous emmener à travers des systèmes d'enseignement cycliques: une façon d'apprendre qui s'est avérée extrêmement efficace, en particulier dans les matières qui exigent la mémorisation”

Étude de Cas pour mettre en contexte tout le contenu

Notre programme offre une méthode révolutionnaire de développement des compétences et des connaissances. Notre objectif est de renforcer les compétences dans un contexte changeant, compétitif et hautement exigeant.

“

Avec TECH, vous pouvez expérimenter une manière d'apprendre qui ébranle les fondations des universités traditionnelles du monde entier”



Vous bénéficierez d'un système d'apprentissage basé sur la répétition, avec un enseignement naturel et progressif sur l'ensemble du cursus.



L'étudiant apprendra, par des activités collaboratives et des cas réels, à résoudre des situations complexes dans des environnements commerciaux réels.

Une méthode d'apprentissage innovante et différente

Cette formation TECH est un programme d'enseignement intensif, créé de toutes pièces, qui propose les défis et les décisions les plus exigeants dans ce domaine, tant au niveau national qu'international. Grâce à cette méthodologie, l'épanouissement personnel et professionnel est stimulé, faisant ainsi un pas décisif vers la réussite. La méthode des cas, technique qui constitue la base de ce contenu, permet de suivre la réalité économique, sociale et professionnelle la plus actuelle.

“ Notre programme vous prépare à relever de nouveaux défis dans des environnements incertains et à réussir votre carrière ”

La méthode des cas est le système d'apprentissage le plus largement utilisé dans les meilleures écoles d'informatique du monde depuis qu'elles existent. Développée en 1912 pour que les étudiants en Droit n'apprennent pas seulement le droit sur la base d'un contenu théorique, la méthode des cas consiste à leur présenter des situations réelles complexes afin qu'ils prennent des décisions éclairées et des jugements de valeur sur la manière de les résoudre. En 1924, elle a été établie comme méthode d'enseignement standard à Harvard.

Dans une situation donnée, que doit faire un professionnel? C'est la question à laquelle nous sommes confrontés dans la méthode des cas, une méthode d'apprentissage orientée vers l'action. Tout au long du programme, les étudiants seront confrontés à de multiples cas réels. Ils devront intégrer toutes leurs connaissances, faire des recherches, argumenter et défendre leurs idées et leurs décisions.

Relearning Methodology

TECH combine efficacement la méthodologie des Études de Cas avec un système d'apprentissage 100% en ligne basé sur la répétition, qui associe différents éléments didactiques dans chaque leçon.

Nous enrichissons l'Étude de Cas avec la meilleure méthode d'enseignement 100% en ligne: le Relearning.

En 2019, nous avons obtenu les meilleurs résultats d'apprentissage de toutes les universités en ligne du monde.

À TECH, vous apprendrez avec une méthodologie de pointe conçue pour former les managers du futur. Cette méthode, à la pointe de la pédagogie mondiale, est appelée Relearning.

Notre université est la seule université autorisée à utiliser cette méthode qui a fait ses preuves. En 2019, nous avons réussi à améliorer les niveaux de satisfaction globale de nos étudiants (qualité de l'enseignement, qualité des supports, structure des cours, objectifs...) par rapport aux indicateurs de la meilleure université en ligne.





Dans notre programme, l'apprentissage n'est pas un processus linéaire, mais se déroule en spirale (apprendre, désapprendre, oublier et réapprendre). Par conséquent, chacun de ces éléments est combiné de manière concentrique. Cette méthodologie a permis de former plus de 650.000 diplômés universitaires avec un succès sans précédent dans des domaines aussi divers que la biochimie, la génétique, la chirurgie, le droit international, les compétences en gestion, les sciences du sport, la philosophie, le droit, l'ingénierie, le journalisme, l'histoire, les marchés financiers et les instruments. Tout cela dans un environnement très exigeant, avec un corps étudiant universitaire au profil socio-économique élevé et dont l'âge moyen est de 43,5 ans.

Le Relearning vous permettra d'apprendre avec moins d'efforts et plus de performance, en vous impliquant davantage dans votre formation, en développant un esprit critique, en défendant des arguments et en contrastant les opinions: une équation directe vers le succès.

À partir des dernières preuves scientifiques dans le domaine des neurosciences, non seulement nous savons comment organiser les informations, les idées, les images et les souvenirs, mais nous savons aussi que le lieu et le contexte dans lesquels nous avons appris quelque chose sont fondamentaux pour notre capacité à nous en souvenir et à le stocker dans l'hippocampe, pour le conserver dans notre mémoire à long terme.

De cette manière, et dans ce que l'on appelle Neurocognitive context-dependent e-learning, les différents éléments de notre programme sont reliés au contexte dans lequel le participant développe sa pratique professionnelle.

Ce programme offre le support matériel pédagogique, soigneusement préparé pour les professionnels:



Support d'étude

Tous les contenus didactiques sont créés par les spécialistes qui enseigneront le cours, spécifiquement pour le cours, afin que le développement didactique soit vraiment spécifique et concret.

Ces contenus sont ensuite appliqués au format audiovisuel, pour créer la méthode de travail TECH en ligne. Tout cela, avec les dernières techniques qui offrent des pièces de haute qualité dans chacun des matériaux qui sont mis à la disposition de l'étudiant.



Cours magistraux

Il existe des preuves scientifiques de l'utilité de l'observation par un tiers expert.

La méthode "Learning from an Expert" renforce les connaissances et la mémoire, et donne confiance dans les futures décisions difficiles.



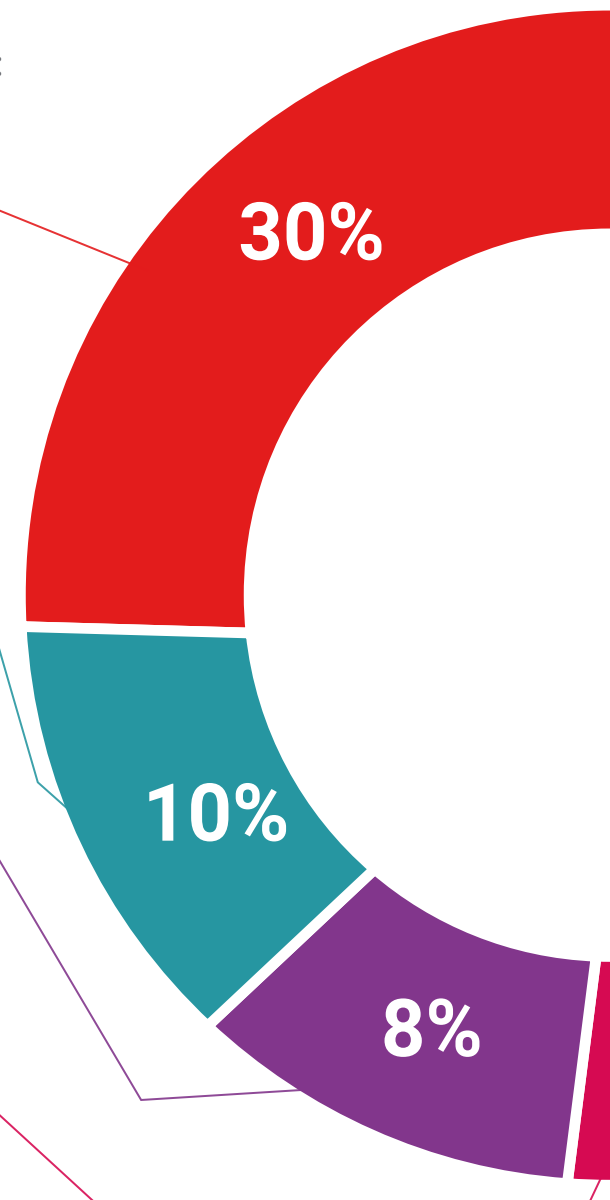
Pratiques en compétences et aptitudes

Les étudiants réaliseront des activités visant à développer des compétences et des aptitudes spécifiques dans chaque domaine. Des activités pratiques et dynamiques pour acquérir et développer les compétences et aptitudes qu'un spécialiste doit développer dans le cadre de la mondialisation dans laquelle nous vivons.



Lectures complémentaires

Articles récents, documents de consensus et directives internationales, entre autres. Dans la bibliothèque virtuelle de TECH, l'étudiant aura accès à tout ce dont il a besoin pour compléter sa formation.





Case studies

Ils réaliseront une sélection des meilleures études de cas choisies spécifiquement pour ce diplôme. Des cas présentés, analysés et tutorés par les meilleurs spécialistes de la scène internationale.



Résumés interactifs

L'équipe TECH présente les contenus de manière attrayante et dynamique dans des pilules multimédia comprenant des audios, des vidéos, des images, des diagrammes et des cartes conceptuelles afin de renforcer les connaissances. Ce système éducatif unique pour la présentation de contenu multimédia a été récompensé par Microsoft en tant que "European Success Story".



Testing & Retesting

Les connaissances de l'étudiant sont périodiquement évaluées et réévaluées tout au long du programme, par le biais d'activités et d'exercices d'évaluation et d'auto-évaluation, afin que l'étudiant puisse vérifier comment il atteint ses objectifs.



06 Diplôme

Le Mastère Spécialisé en Programmation de Jeux Vidéo vous garantit, en plus de la formation la plus rigoureuse et la plus actuelle, l'accès à un diplôme universitaire de Mastère Spécialisé délivré par TECH Université Technologique.



“

Terminez ce programme avec succès et recevez votre Mastère Spécialisé sans avoir à vous soucier des contraintes de déplacements ou des formalités administratives”

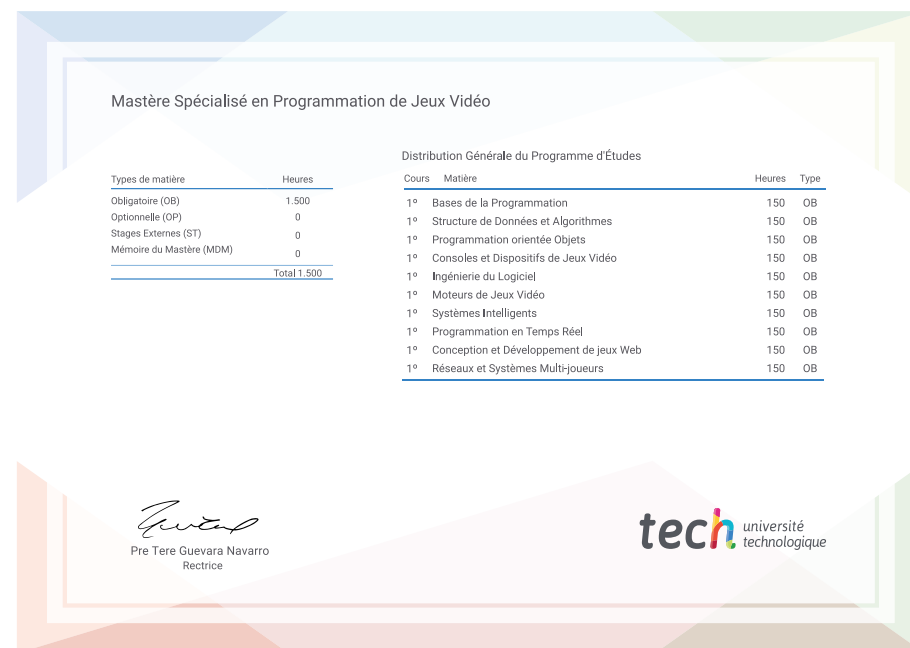
Ce **Mastère Spécialisé en Programmation de Jeux Vidéo** contient le programme le plus complet et le plus à jour du marché.

Après avoir réussi l'évaluation, l'étudiant recevra par courrier postal* avec accusé de réception son correspondant diplôme de **Mastère Spécialisé** délivré par **TECH Université Technologique**.

Le diplôme délivré par **TECH Université Technologique** indiquera la note obtenue lors du Mastère Spécialisé, et répond aux exigences communément demandées par les bourses d'emploi, les concours et les commissions d'évaluation des carrières professionnelles.

Diplôme: **Mastère Spécialisé en Programmation de Jeux Vidéo**

N.º d'heures officielles: **1.500 h.**



*Si l'étudiant souhaite que son diplôme version papier possède l'Apostille de La Haye, TECH EDUCATION fera les démarches nécessaires pour son obtention moyennant un coût supplémentaire.

future
santé confiance personnes
éducation information tuteurs
garantie accréditation enseignement
institutions technologie apprentissage
communauté engagement
service personnalisé innovation
connaissance présent qualité
en ligne formation
développement institutions
classe virtuelle langues

tech université
technologique

Mastère Spécialisé Programmation de Jeux Vidéo

- » Modalité: en ligne
- » Durée: 12 mois
- » Qualification: TECH Université Technologique
- » Intensité: 16h/semaine
- » Horaire: à votre rythme
- » Examens: en ligne

Mastère Spécialisé

Programmation de Jeux Vidéo