Smartfuture
Certifications | Training | Technology solutions

Formation & Certification • Unity Certified User

Unity Certified User – Programmer

Préparation à la certification Unity Certified User – Programmer : valide les bases du C# dans Unity pour créer des jeux, applications et expériences AR/VR. Recommandé : ~150 heures de pratique et formation Unity avant l'examen.

Distributeur officiel Certiport | Centre d'examen Certiport

Learn • Practice • Certify

Durée	Examen
30 h (projets & pratique)	UCU: Programmer
Modalité	**Niveau**
Distanciel	À partir de 15 ans, débutants motivés

INSCRIPTION / RÉSERVATION

Je m'inscris maintenant

- **Learn** : parcours structuré de formation et de projets pratiques sur Unity et C#, aligné sur les objectifs officiels de l'examen.
- **Practice** : Pratique sur projets, exercices et simulations d'examen.
- **Certify** : passage de la certification internationale Unity Certified User – Programmer dans nos centres d'examen Certiport.

OBJECTIFS PÉDAGOGIQUES

- Appliquer les bases du C# pour créer des interactions dans Unity.
- Déboguer, analyser les erreurs et utiliser l'API Unity.
- Écrire un code propre et maintenable selon les conventions Unity.
- Maîtriser l'interface Unity, l'Animator Controller et les états pour réussir l'examen.

PUBLIC CIBLE

- Adolescents (15+) découvrant la programmation de jeux.
- Étudiants et personnes en reconversion intéressés par Unity.
- Enseignants et formateurs souhaitant intégrer la certification.

PRÉREQUIS

- Connaissances de base en informatique (utilisation d'un PC, gestion de fichiers).
- Une première exposition à la logique algorithmique est un plus (variables, conditions, boucles).
- Motivation à pratiquer régulièrement sur Unity entre les séances.

PROGRAMME DE LA FORMATION – DÉTAILLÉ

Débogage, résolution de problèmes et interprétation de l'API

Messages de débogage (Debug Log)

- À partir d'un exemple de message de journal de débogage (debug log), écrire le code C# qui génère ce message dans Unity.

Gestion des références nulles

- À partir d'un extrait de code et de ses messages d'erreur associés, déterminer quel(s) objet(s) a (ont) la valeur null.

Interprétation de la documentation de l'API Unity

- Face à une tâche de programmation nécessitant l'utilisation d'une classe particulière de l'API Unity, identifier la méthode ou les propriétés appropriées, les bons arguments et la syntaxe à utiliser.

Création de code

Variables, modificateurs et collections de données

- Déterminer quand et comment initialiser et utiliser des variables en C#, y compris l'utilisation appropriée des modificateurs de variables (public, private, etc.).
- Utiliser les collections de données usuelles dans Unity : Tableaux (Arrays), Listes (Lists) et Dictionnaires (Dictionaries).

Déclaration de fonctions (methods) en C#

- À partir d'une liste de mots-clés et d'éléments de syntaxe, construire une déclaration de fonction valide (type de retour, nom, paramètres, portée, etc.).

Fonctions et contrôle des états (Animator Controller)

- À partir d'un extrait de code et de la description du résultat attendu, identifier la fonction appropriée pour contrôler ou déclencher un état.
- Inclure notamment la gestion des états via l'Animator Controller (changement d'animations, transitions, paramètres).

Gestion des entrées (input)

- À partir d'un scénario où un type d'entrée spécifique est requis et où les briques de base sont fournies, construire l'écouteur d'entrée (input listener) nécessaire.
- Gérer notamment les entrées au clavier et les entrées tactiles (touch) dans Unity.

PROGRAMME DE LA FORMATION – DÉTAILLÉ

Logique et contrôle de flux

- Comprendre quand et comment utiliser les opérateurs de logique (&&, ||, !, etc.) et de contrôle de flux (if/else, switch, boucles, etc.) en C#.
- Appliquer ces opérateurs pour gérer les comportements de jeu et la progression des états dans Unity.

Réactions aux changements d'interface utilisateur (UI)

- Face à un scénario où un élément d'interface utilisateur (UI) signale un changement (clic, saisie, glissement, etc.), identifier les actions à entreprendre dans le code.
- Programmer la logique associée aux événements UI (boutons, sliders, champs de texte, etc.).

Évaluation et analyse de code

Gestion des fonctions d'événements

- À partir d'un scénario mettant en avant la nécessité de gérer une fonction d'événement, déterminer l'action appropriée à entreprendre.
- Inclure notamment la gestion des entrées clavier et des entrées tactiles comme sources d'événements.

Erreurs de types de données

- À partir d'un extrait de code provoquant une erreur due à un type de donnée incorrect sur une variable, identifier précisément l'erreur.

Portée et visibilité (public/private) et événements d'animation

- À partir d'un extrait de code produisant une erreur parce qu'une fonction ou une variable est déclarée ou utilisée de manière incorrecte (par exemple mismatch public/private), identifier la cause de l'erreur.
- Inclure des cas impliquant les événements d'animation (Animation events) dans l'Animator.

Classes ECS et autres types de classes

- À partir d'un extrait de code contenant une définition de classe, distinguer si la classe est une classe ECS (Entity Component System) ou un autre type de classe.

Conventions de nommage Unity

- À partir d'un ensemble d'extraits de code, reconnaître celui qui respecte les conventions de nommage en vigueur dans l'écosystème Unity (classes, méthodes, variables, etc.).

Qualité des commentaires

- À partir d'un extrait de code (ou d'un ensemble d'extraits), reconnaître les commentaires qui décrivent correctement ce que fait le code.
- Identifier les commentaires imprécis, trompeurs ou insuffisants.

Navigation dans l'interface Unity

Fenêtres de l'IDE Unity

- Décrire le rôle, les fonctionnalités et les caractéristiques des différentes fenêtres de l'IDE Unity (Scene, Game, Hierarchy, Inspector, Project, Console, etc.).

Changer l'IDE de script par défaut

- Démontrer comment modifier l'éditeur de script par défaut utilisé par Unity (par exemple Visual Studio, Visual Studio Code, Rider, etc.).

Création d'une machine à états fonctionnelle (Animator)

- À partir d'un scénario incluant :
- a. une portion limitée d'un scénario de jeu,
- b. un ensemble de clips d'animation,
- c. une liste de paramètres et de propriétés à configurer,
- créer une machine à états fonctionnelle dans l'Animator Controller.

Programmation d'une machine à états dans l'Animator Controller

- Créer et programmer une machine à états de fonction dans l'Animator Controller Unity, y compris l'utilisation de la syntaxe des fonctions de l'Animator (paramètres, transitions, déclencheurs, etc.).