unity

unity

Certified

Programmer

Objetivos del examen

Programador
certificado Unity

El cargo

Los profesionales en programación de Unity desarrollan contenido interactivo usando esta herramienta. Junto a otros miembros del equipo de desarrollo (como profesionales en audio y diseño), el programador de Unity le da vida a su visión de la aplicación usando las funcionalidades de editor de Unity, en adición de los recursos sonoros y visuales creados por los demás miembros del equipo de desarrollo de software. El programador de Unity es un generalista experto en resolver problemas difíciles de código, y es responsable de colaborar en un amplio número de posibles tareas técnicas, que incluyen la integración de recursos de arte, la codificación de la interfaz de usuario, el desarrollo de las interacciones del usuario y las reglas del sistema de juego, la implementación de la lógica de estado de la aplicación, la emulación de leyes físicas, la depuración del código y la optimización del rendimiento.

La certificación de Programador certificado Unity es una certificación profesional para programadores novatos y de nivel intermedio y para estudiantes con estudios superiores que buscan cargos de programación en varias industrias. Esta certificación le demuestra al posible empleador que la persona:

- Emplea su conocimiento en programación dentro del contexto de procesos profesionales de desarrollo de software para crear y mantener aplicaciones construidas con el motor Unity
- Tiene aptitudes para procesos técnicos, está orientada a la lógica y es recursiva
- Puede ocuparse del manejo de rutinas de tareas de programación de nivel medio de forma independiente y puede trabajar en desafíos técnicos complejos con ingenieros más experimentados

Títulos para este cargo

- Programador de juego
- Ingeniero de software
- Desarrollador de software
- Desarrollador Unity
- Desarrollador de aplicaciones para dispositivos móviles

Requisitos previos

Esta certificación se creó para los programadores que acaban de graduarse de la universidad en Programación de juegos, Ciencia computacional o campos relacionados; aprendices independientes que hayan completado 2 años o más de un estudio equivalente a la universidad o que tengan experiencia en programación; o profesionales novatos o con poca experiencia que han usado Unity en su trabajo. Los candidatos deberán asistir a la prueba con experiencia práctica en programación de aplicaciones interactivas con Unity por su cuenta o como parte de un equipo multifuncional, resultando en un prototipo completo o en una demostración técnica.

Experiencia previa:

- Dos años o más de experiencia práctica en programación de videojuegos o en programación interactiva 3D usando Unity
- Dos años o más de experiencia práctica en programación de computadoras, incluyendo C#
- Experiencia en el ciclo de vida completo del desarrollo de un software, desde el concepto inicial hasta su finalización
- Comprensión de las aplicaciones profesionales del desarrollo de software con Unity, incluidos el desarrollo de videojuegos, el ocio interactivo y la visualización de diseño
- Comprensión básica del flujo de trabajo de los assets visuales/3D y de animación en Unity, incluida la configuración de personajes y entornos
- Comprensión de las prácticas de desarrollo de software de equipos profesionales, incluidos unit testing y control de versiones
- Conocimiento de los servicios de Unity para la colaboración, la monetización, las operaciones en vivo (live ops) y el multijugador
- Comprensión de las matemáticas fundamentales para el desarrollo interactivo de 3D, incluidas el álgebra lineal y las operaciones con matrices

Nota: esta certificación se desarrolló para la versión 2017.3 de Unity.

Habilidades principales

Las habilidades principales en esta línea de trabajo se centran en la contribución a la ejecución técnica de un proyecto, desde la concepción de la idea, hasta el lanzamiento y más allá.

Programación de interacciones principales

- Implementar y configurar la física y el comportamiento de GameObjects
- Implementar y configurar las entradas y los controles
- Implementar y configurar los movimientos y la visión de la cámara

Trabajar en el proceso de arte

- Entender materiales, texturas y shaders, y escribir scripts que interactúen con la API de renderización de Unity
- Entender la iluminación y escribir scripts que interactúen con la API de iluminación de Unity
- Entender la animación 2D y 3D y escribir scripts que interactúen con la API de animación de Unity
- Entender sistemas de partículas y efectos, y escribir scripts que interactúen con la API de sistema de partículas de Unity

Desarrollo de sistemas de aplicación

- Interpretar scripts para el flujo de la interfaz de la aplicación, como sistemas de menús, navegación por la IU y configuración de la aplicación
- Interpretar scripts de personalización controlada por el usuario, como creadores de personajes, inventarios, tiendas y compras dentro de la aplicación
- Analizar scripts para características de progresión del usuario, como sistemas de puntos, subida de nivel y economía dentro del juego usando tecnologías como Unity Analytics y PlayerPrefs
- Analizar scripts para superposiciones 2D, como HUD, minimapas y anuncios
- Identificar scripts para guardar y recuperar datos de usuario y aplicación
- Reconocer y evaluar el impacto de la funcionalidad del trabajo en red y el multijugador

Programación de diseño de entornos y escenas

- Determinar scripts para la implementación de recursos de audio
- Identificar métodos para la implementación de instanciación, destrucción y gestión de GameObjects
- Determinar scripts para la búsqueda de rutas en el sistema de navegación de Unity

Optimización de rendimiento y plataformas

- Evaluación de errores y problemas de rendimiento utilizando herramientas como el Profiler
- Identificar optimizaciones para abordar requisitos para plataformas de desarrollo o configuraciones de hardware específicos
- Determinar aplicaciones y optimizaciones de IU comunes para plataformas XR

Trabajar en equipos de desarrollo de software profesionales

- Reconocer conceptos asociados con los usos e impactos del control de versión, usando tecnologías como Unity Collaborate
- Demostrar conocimiento de pruebas de desarrollador y su impacto en el proceso de desarrollo de software, incluyendo al Profiler y otras técnicas de pruebas y depuración tradicionales
- Reconocer técnicas de estructuración de scripts para la modularidad, legibilidad y reusabilidad

Temas del examen de certificación

Programación de interacciones principales

- Implementar comportamientos e interacciones de GameObjects y entornos
- Identificar métodos para implementar entradas y controles
- Identificar métodos para implementar vistas de cámaras y movimientos

Trabajar en el proceso de arte

- Conocimiento de los materiales, texturas y shaders: API de renderización de Unity
- Conocimiento de iluminación: API de iluminación de Unity
- Conocimiento de animación 2D y 3D: API de animación de Unity
- Conocimiento de sistemas de partículas: API de partículas de Unity

Desarrollo de sistemas de aplicación

- Aplicación del flujo de interfaz, como sistemas de menús, navegación por la IU y configuración de la aplicación
- Personalización controlada por el usuario, como los creadores de personajes, inventarios, tiendas y compras dentro de la aplicación
- Implementación de características de progresión del usuario, como el sistema de puntos, subida de nivel y economías dentro del juego usando tecnologías como Unity Analytics
- Implementación de superposiciones 2D, como HUD, minimapas y anuncios
- Guardado y recuperación de la aplicación y los datos de usuario
- Reconocimiento del valor y el impacto de la funcionalidad de trabajo en red y multijugador

Programación de diseño de entornos y escenas

- Determinar scripts para la implementación de recursos de audio
- Identificar métodos para la implementación de instanciación, destrucción y gestión de GameObjects
- Determinar scripts para la búsqueda de rutas en el sistema de navegación de Unity

Optimización de rendimiento y plataformas

- Evaluación de errores y problemas de rendimiento utilizando herramientas como el Profiler
- Identificar optimizaciones para abordar requisitos para plataformas de desarrollo o configuraciones de hardware específicos
- Determinar aplicaciones y optimizaciones de IU comunes para plataformas XR

Trabajo en equipos de desarrollo de software

- Control de versiones: impactos y usos de herramientas como Unity Collaborate
- Pruebas e impacto en el proceso de desarrollo de software
- Reconocer técnicas de estructuración de scripts para la modularidad, legibilidad y reusabilidad

Preguntas de ejemplo

Pregunta 1

Un programador debe implementar un sistema de menús para la IU. Cada menú consiste de un UI Panel y uno o más UI Buttons, todos agrupados bajo un objeto UI Canvas. Todo el sistema de menús de IU se creará en una escena separada que se carga aditivamente.

El estilo de arte de los paneles y botones debe ser consistente (color, textura, tipo de transición de botones, etc.), pero la directora de arte no ha tomado estas decisiones. A ella le gustaría trabajar en estos ajustes al mismo tiempo que el programador trabaja en la IU. Sus cambios surtirán efecto en todos los objetos nuevos y existentes en la escena.

¿Cuál sería la mejor manera de que el programador use la funcionalidad de Unity para crear un sistema de menús que sea funcional al tiempo que cumple con el requisito de permitirle a la directora de arte trabajar en paralelo (y de forma independiente) en la apariencia y sensaciones que transmite (look and feel)?

A Crear subclases para `UI.Button` y `UI.Panel` y establecer los valores de apariencia y sensaciones de manera programática.

B Crear nuevos materiales de paneles y botones, y asignarlos a todos los botones y paneles en la escena.

C Usar prefabs para el botón y el panel, y pedirle a la directora de arte que modifique los prefabs.

D Escribir un script para buscar y reemplazar valores en el archivo de la escena basándose en los comentarios de la directora de arte.

Pregunta 2

Un juego de desplazamiento infinito (endless runner) en 3D se desarrolla sobre varias vías férreas y un patio de maniobras. El jugador siempre está corriendo hacia adelante sobre las vías y debe esquivar los trenes saltando sobre ellos o pasándose a la vía adyacente.
Cada nuevo tren en la vía se añade detrás de todos los otros trenes en esa vía. No obstante, debido a que los trenes se mueven hacia el jugador a distintas velocidades (o no se mueven), en ocasiones se solapan entre sí, problema que debe arreglarse.

¿Cuál es la mejor forma de evitar que los nuevos trenes se solapen con los trenes que ya están en la vía?

A Al generar un tren sobre la vía, se debe determinar una posición de generación que evitará el problema usando las velocidades del nuevo tren y del último tren puesto en esa vía, además del punto en el que el último tren colocado en la vía desaparecerá al pasar al jugador.

B Cuando un tren se está moviendo, usar Raycast hacia adelante desde la parte frontal del tren y empujar hacia adelante cualquier tren golpeado por Raycast con la velocidad del tren más rápido.

C Al generar un tren en las vías, agregarle un Rigidbody y luego usar fuerzas para mover los trenes.

D Al generar un tren sobre una vía, usar un BoxCast con una longitud proporcional a la velocidad del tren para asegurarse de que no chocará con otro tren hasta que esté detrás de la cámara.

Pregunta 3

Un programador está trabajando en una habitación oscura y gris y debe crear una antorcha parpadeante que lanza una sombra temblorosa e inquietante sobre los muros, los pisos y el techo. El programador escribe estas funciones en un `MonoBehaviour` adjunto a la antorcha:

```
void Start()
{
    Light light = GetComponent<Light>();
    light.lightMapBakeType = LightMapBakeType.Mixed;
    light.type = LightType.Area;
    light.shadows = LightShadows.Soft;
    light.range = 5f;
}
void Update()
{
    GetComponent<Light>().intensity = Mathf.PerlinNoise(Time.time, 0);
}
```

La antorcha no lanza ninguna luz o sombra durante la ejecución. La luz se establece en los valores por defecto en el editor.

¿Qué debe cambiar el programador para que el código funcione como se debe?

A Establecer `light.lightBakeType` a `LightmapBakeType.Realtime`

B Establecer `light.range` a `10`

C Establecer `light.shadows` a `LightShadows.Hard`

D Establecer `light.type` a `LightType.Point`

Pregunta 4

Un programador está desarrollando un juego de simulación minera en el que el jugador puede excavar el suelo en busca de minerales. En uno de los sitios, el jugador puede crear un túnel que se cruza con un sistema de cuevas existente. El documento de diseño especifica que cualquier sonido que ocurra en alguna de las cuevas y en los nuevos túneles debe tener un poco de reverberación. El programador debe asegurarse de que el usuario esté sistemáticamente en la ReverbZone de la cueva más cercana.

¿Cómo debe manipular el programador las propiedades de AudioReverbZone para cumplir estos requisitos?

A Aumentar los reflejos para ajustarse a la nueva área.

B Aumentar la maxDistance de ambas ReverbZones para que se toquen dentro de la nueva área de conexión.

A Aumentar las reverberaciones para acomodarlas en la nueva área.

D Aumentar el decayTime en la nueva área.

Pregunta 5

Mientras escribe una función de carga, un programador recibe un error de compilación:

```
error CS1624: The body of `CustomAnalytics.LevelLoading()' cannot be
an iterator block because `void' is not an iterator interface type
```

```
void LevelLoading(){
    AsyncOperation async = SceneManager.LoadSceneAsync("Level_01");
    while (!async.isDone)
    {
        yield return null;
    }
}
```

¿Qué debe hacer el programador para resolver este problema?

A Cambiar `yield return null` a `yield return WaitForSeconds(0)`

B Cambiar `void LevelLoading()` a `IEnumerator LevelLoading`

C Cambiar `SceneManager.LoadSceneAsync("Level_01")` a `Application.LoadLevelAdditiveAsync("Level_01")`

D Cambiar `while (!async.isDone)` a `while (!async.allowSceneActivation)`

Pregunta 6

El sistema de entrada de un juego de conducción se configura para que el eje de entrada horizontal controle la dirección. Durante las pruebas, se descubre que algunos dispositivos con joystick registran entradas de dirección incluso cuando la palanca está centrada.

¿Qué se debe cambiar en el eje del sistema de entrada para resolver este problema?

A Aumentar Gravity

B Configurar Snap como verdadero

C Aumentar Deadzone

D Reducir Sensitivity

Pregunta 7

En un juego de aventuras ambientado en un planeta alienígena, el jugador debe exterminar varias formas de vida. El puntaje del jugador aumenta con cada baja. El documento de diseño dice que el puntaje debe estar vinculado a la cuenta del jugador para poderlo recuperar más tarde, incluso si el jugador está en una sesión de juego distinta o en otro dispositivo.

¿Cuál es el método más confiable para que el programador pueda almacenar los datos de puntaje?

A Usar `DontDestroyOnLoad()` en el GameObject que guarda los datos de puntaje y subir los datos a un servidor justo antes de que se cierre la aplicación.

B Guardar el puntaje en `PlayerPrefs` cada vez que se actualice y subirlo a un servidor justo antes de que se cierre la aplicación.

C Usar un valor estático para almacenar los datos de puntaje de manera que estén disponibles en la próxima sesión de juego.

D Usar serialización de datos para, de manera persistente, almacenar los datos de puntaje y cargarlos a un servidor.

Respuestas correctas: C, A, D, B, B, C, D