unity

unity

Certified

Expert

Gameplay

Programmer

Objetivos del examen

Experto certificado de Unity
Programador de Gameplay

El papel

Los profesionales de la programación de Gameplay se centran en la ejecución del diseño del juego. Los programadores de Gameplay dan vida al juego añadiendo scripts a los assets creados por el equipo artístico. También son responsables de implementar la interfaz de usuario (UI), el entorno, los personajes y los objetos, así como la optimización y los scripts necesarios para ejecutar las mecánicas del juego. También aportan funcionalidad a los personajes no jugadores (NPC) que habitan en el juego.
Los programadores de Gameplay trabajan codo con codo con los diseñadores (o pueden ser los propios diseñadores) y toman como referencia el documento de diseño de juego (GDD) para que el juego llegue a materializarse. Son personas muy técnicas y entienden lo que se necesita para dar vida a las ideas. Esto incluye saber cuándo y cómo usar componentes de Unity, escribir sus propios componentes, configurar Prefabs y organizar adecuadamente cada escena. Todo esto lo hacen con el documento de diseño técnico (TDD) en mente.

Nombres de este oficio

- Programador de Gameplay
- Programador/ingeniero de juego
- Diseñador de niveles

Requisitos previos

Se recomienda esta certificación de experto para las personas que hayan pasado varios años en este campo y hayan acumulado una amplia experiencia aplicada, avanzada y práctica, como:

- Experiencia en un estudio de desarrollo de videojuegos
- Experiencia en el diseño de juegos inmersivos, juegos web o móviles, y juegos digitales
- Buen conocimiento de creación de scripts y programación con C#
- Experiencia en el ciclo de vida completo de un juego, desde el concepto inicial hasta el lanzamiento
- Comprensión experta de todos los aspectos del diseño del juego (niveles, personajes no jugadores [NPC] y mecánicas del juego)
- Experiencia en el diseño y la implementación de servicios de Unity
- Comprensión del flujo de trabajo de assets y animación, incluyendo la creación de personajes y del entorno
- Habilidades organizacionales buenas que se adhieran a las estructuras de archivos, convenciones de nombres y protocolos establecidos
- Capacidad de crear y utilizar una biblioteca de Prefabs para hacer el mejor juego posible

Habilidades principales

La certificación de programador de Gameplay garantiza que los candidatos tienen las habilidades necesarias para crear y optimizar correctamente un juego. Los candidatos que la aprueben obtendrán una experiencia avanzada en las siguientes áreas:

Creación de prototipos

- Creación y evaluación de prototipos del juego principal para una rápida iteración
- Diseño e implementación de GameObjects para probar la funcionalidad de las mecánicas de juego
- Creación de una biblioteca de Prefabs basada en el GDD y utilización de los assets generados por el equipo de arte
- Organización y configuración de la estructura de carpetas para una biblioteca de Prefabs

Diseño de niveles

- Diseño y creación de Prefabs interactivos utilizando colliders y componentes de Rigidbody
- Configuración de la capa apropiada de físicas con el uso de Prefabs interactivos
- Configuración de Prefabs generados en tiempo de ejecución e implementación de una jugabilidad dinámica
- Configuración e implementación de activaciones de eventos y de eventos programados con colliders como activadores
- Creación y programación de componentes de lógica personalizada para GameObjects, con el fin de vincular máquinas de estados a interacciones y activaciones en la escena
- Colocación y configuración de efectos a lo largo de la escena, incluyendo colocaciones estáticas y generadas en tiempo de ejecución
- Configuración de niveles de detalle (LOD) por especificaciones de plataforma
- Evaluación de colocación de GameObjects y dependencias por especificación de plataforma
- Evaluación de escenas para transmisión frente a una carga de escena estática
- Construcción de niveles de juego con múltiples escenas
- Configuración de cinemáticas para mejorar la jugabilidad

Diseño de personajes no jugadores (NPC)

- Diseño y creación de scripts de lógica e inteligencia artificial (AI) de NPC
- Implementación de navegación y búsqueda de rutas para los NPC con NavMesh, NavMeshAgent, obstáculos de NavMesh y Off-Mesh Links
- Configuración de tipos y costos de zonas NavMesh
- Configuración de activadores para activar o desactivar zonas NavMesh
- Implementación de evasión de NavMeshAgent y simulación de multitudes
- Evaluación de colocación de NPC y dependencias por especificación de plataforma
- Configuración de audio y efectos basados en fotogramas de clips de animación

Interfaz de usuario y diseño mecánico de juegos

- Uso del sistema de animación para Animator Controllers, máquinas de estados y scripts lógicos para las mecánicas de juego
- Evaluación y optimización de colliders, Rigidbody y materiales físicos para GameObjects interactivos
- Implementación de interfaces de usuario relacionadas con la jugabilidad, como HUD, minimapas, sistemas de radar, barras de salud y otros elementos basados en datos

Optimizaciones de rendimiento y plataformas de destino

- Implementación de la descarga y colocación de un AssetBundle en los niveles de juego
- Diseño y modificación de controles para diferentes plataformas y realidad virtual (VR)
- Análisis de GameObjects y escenas para la optimización del tiempo de ejecución y almacenamiento por especificación de plataforma
- Optimización de Occlusion Culling en todas las escenas
- Depuración y testeo de niveles de juego en tiempo de ejecución

Implementación de servicios de Unity: anuncios, compras en la aplicación y Analytics

- Implementación de anuncios de Unity sencillos y con recompensa
- Implementación de compras en la aplicación de Unity (IAP)
- Integración de Design Unity Analytics en el GDD y TDD
- Configuración de la supervisión de eventos de datos personalizados mediante Analytics para supervisar el comportamiento del jugador
- Análisis y evaluación de los niveles existentes, y recomendación de cambios basados en datos de Analytics
- Utilización de Unity Performance Reporting para modificar y optimizar las versiones de juegos por plataforma

Temas del examen de certificación

Creación de prototipos (del juego principal para iteraciones rápidas)

- Creación de prototipos del juego principal
- Conflictos y soluciones durante la fase de prototipo

Programación de diseño de niveles

- Configuración de físicas
- Raycasting
- Prefabs generados por scripts durante el tiempo de ejecución.
- Lógica de nivel y comportamientos
- Población de niveles con Particle Systems y efectos
- Optimización de plataforma
- Carga y descarga de escenas
- Métodos de muestra de cinemáticas

Programación de diseño de NPC

- Lógica y comportamiento de NPC
- Navegación y rutas
- Aparición y colocación de NPC

Implementación de interfaz de usuario

- Sistemas coordenados de UI y scripts de UI

Optimización de rendimiento y plataformas de destino

- Optimización de renderización
- Configuración y descarga de AssetBundles
- Depuración de jugabilidad
- Diferencias por plataformas e impacto en la jugabilidad

Implementación de servicios de Unity

- Publicidad de Unity
- Compras en la aplicación de Unity
- Unity Analytics
- Unity Cloud Build

Preguntas de ejemplo

Pregunta 1

Un programador de Gameplay está creando un prototipo de juego de disparos con cámara lateral. El personaje principal es un avión y los enemigos son diferentes tipos de ovnis. Cuando el jugador destruye un ovni, se muestra un efecto de explosión donde estaba el ovni. Cuando el jugador muere, se muestra un efecto de explosión especial de la nave del jugador.

El avión puede disparar hasta 64 balas en pantalla. El número máximo de ovnis de cualquier tipo que se pueden mostrar en la pantalla es 128. El número máximo de balas de ovnis en pantalla es 1024. El número máximo de efectos de explosión en pantalla es 128.

El programador de Gameplay necesita determinar qué GameObjects deberían colocarse en el editor y cuáles deberían generarse en tiempo de ejecución (por ejemplo, desde un pool de objetos).

¿Cómo podría lograrse esto?

A El jugador, los ovnis, el efecto de explosión del jugador y el efecto de explosión de los ovnis deben colocarse en el editor. Las balas del jugador y las balas de los ovnis se deben generar en tiempo de ejecución.

B El jugador, las balas del jugador, los ovnis y las balas de los ovnis deben colocarse en el editor. El efecto de explosión del jugador y el efecto de explosión de los ovnis deben generarse en tiempo de ejecución.

C El jugador y los ovnis deben colocarse en el editor. Las balas del jugador, las balas de los ovnis, el efecto de explosión del jugador y el efecto de explosión de los ovnis deben generarse en tiempo de ejecución.

D El jugador y el efecto de explosión del jugador deben colocarse en el editor. Los ovnis, las balas de los ovnis, las balas del jugador y el efecto de explosión de los ovnis deben generarse en tiempo de ejecución.

Pregunta 2

El documento de diseño de juego (GDD) es un juego de mundo abierto en tercera persona. El GDD especifica dos tipos de jugabilidad para el jugador principal:

1. Caminar por el mundo a pie
2. Conducción en motocicleta

La altura relativa de cada modo es similar, pero cuando el jugador está en la motocicleta, puede ir mucho más rápido que a pie.

El programador de Gameplay determina varias áreas del juego que deben ajustarse durante el cambio entre caminar y manejar la motocicleta:

- Campo de visión de la cámara (FOV)
- Distancias del nivel de detalles (LOD)
- Transmisión del nivel

Cuando el jugador está en la motocicleta, ¿qué estrategia debe utilizar el programador de Gameplay para abordar cada punto de este GDD?

A El campo de visión FOV de la cámara debe tener un valor menor.
El nivel debe transmitirse mucho más rápido.

B El campo de visión FOV de la cámara debe tener un valor mayor.
El nivel debe transmitirse mucho más lento.

C Las distancias de LOD deben tener un valor mayor.
El nivel debe transmitirse mucho más lento.

D El campo de visión FOV de la cámara debe tener un valor mayor.
El nivel debe transmitirse mucho más rápido.

Pregunta 3

El GDD está situado en una gran ciudad donde el jugador puede caminar e interactuar con cualquier persona de la ciudad. El jugador puede entonces completar varias misiones aleatorias para los ciudadanos. El jugador solo puede tener 5 misiones activas en cualquier momento. Si el jugador camina de un extremo a otro de la ciudad, tardará aproximadamente 4 horas.

El juego contiene un minimapa de vista superior en el Heads-Up Display (HUD) que muestra un icono 2D para lo siguiente:

- la ubicación del jugador;
- todas las misiones activas;
- todos los diversos ciudadanos caminando por la ciudad;

El minimapa se puede alejar y acercar. Cuando se acerca completamente, solo muestra el icono del jugador. Cuando se aleja completamente, se puede ver el 25 % de la ciudad. Hay una desaceleración que ocurre solo cuando el jugador acerca completamente el minimapa.

¿Cuál es una posible causa de la desaceleración?

A Hay demasiadas llamadas de descarga para todas las texturas de los íconos de ciudadanos de la memoria.

B Los íconos de misiones activas recuperan información con demasiada frecuencia para saber cuántos están activos actualmente.

C El algoritmo de selección del minimapa está tardando demasiado en eliminar la mayor parte de la ciudad.

D El HUD es muy complejo y se vuelve a crear en cada frame.

Pregunta 4

El GDD es un juego móvil con un presupuesto de memoria ajustado. El juego tiene secciones cargadas en AssetBundles para asegurar un rendimiento sin problemas. Cada sección es independiente, por lo que los AssetBundles se puede cargar y descargar sin afectar a otra sección.

Sin embargo, a veces aparecen texturas rosadas.

¿Cuál podría ser la causa de este problema?

A AssetBundle.mainAsset no funciona correctamente.

B Se llama a AssetBundle.Unload() demasiado pronto.

C Se llama a AssetBundle.LoadAllAssets() con un tipo incorrecto.

D Se llama a AssetBundle.LoadAssetWithSubAssets() con un segmento y un tipo incorrectos.

Pregunta 5

El GDD es un juego de agilidad mental en 3D para móviles con más de 300 niveles. El juego contiene los siguientes eventos personalizados de Analytics para cada nivel:

1. "levelStarted": activado cuando un jugador inicia el nivel
2. "levelCompleted": activado cuando un jugador completa el nivel

El equipo desea determinar cuáles son los 5 niveles más difíciles del juego. Los niveles tienen límite de tiempo.

¿Qué evento personalizado adicional sería necesario para determinar esos 5?

A Agregar un "levelTime" para que envíe la cantidad de tiempo transcurrido en un nivel.

B Agregar un "levelRestarted" para que se active cuando la aplicación móvil se cierra por el sistema operativo.

C Agregar un "levelFailed" para que se active cuando el jugador no logra superar el nivel o se sale antes de tiempo.

D Agregar un "levelResumed" para que se active cuando la aplicación móvil se reanuda desde un proceso en segundo plano.

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