Componente formativo

Técnicas de testeo y pruebas para videojuegos

Breve descripción:

El componente de formación hace énfasis en las pruebas del prototipo del juego las cuales comienzan desde la etapa de preproducción, para optimizar el videojuego y corregir los errores o “bugs” en dicho videojuego.

Área ocupacional:

Arte, cultura, esparcimiento y deportes.

Abril 2023

Grupo de Ejecución de la Formación Virtual

Tabla de contenido

Introducción 3

1. ¿Por qué es necesario probar los juegos? 4

1.1. Tipos de técnicas de pruebas de juegos 4

1.2. Métodos de prueba de juegos 6

1.3. ¿Qué es la automatización de pruebas de juegos? 9

1.4. ¿Qué hacen los QA "testers" de juegos? 9

1.5. ¿Cómo funciona la prueba de juegos? 10

1.6. Métricas clave de las pruebas funcionales de control de calidad 11

1.7. Informes de errores 12

1.8. Tipos principales de métodos de prueba de juegos 12

1.9. Cómo funcionan las pruebas en diferentes plataformas 13

2. Cómo iniciar las pruebas o "testing" en un videojuego 15

3. Utilizando el Framework Test Runner – "unit testing" en Unity 17

Síntesis 30

Material complementario 31

Glosario 32

Referencias bibliográficas 33

Créditos 34

Introducción

Estimado aprendiz, continuando con este maravilloso aprendizaje, reflexione sobre lo siguiente: antes de que una mariposa pueda tomar su primer vuelo, debe empujar y luchar para salir de un capullo. Sin este desafiante obstáculo, sería demasiado débil para sobrevivir en el mundo exterior. ¿Qué tiene esto que ver con las pruebas de juego? Bueno, como una mariposa, cada juego debe someterse a una prueba seria que lo fortalezca antes de lanzarlo al mundo. Por lo anterior le invitamos a revisar el siguiente video para identificar el contexto de aprendizaje de este componente formativo:

Video 1. Introducción

Enlace de reproducción del video

Síntesis del video: Introducción

1. ¿Por qué es necesario probar los juegos?

Las empresas de desarrollo de juegos deben invertir lo suficiente en las pruebas de los videojuegos para garantizar que el producto final no tenga errores o "bugs". Eso haría decaer la experiencia del jugador y la parte central de los juegos es la experiencia del consumidor. Con una buena experiencia de usuario, un juego prosperará. Hay millones de juegos en la App Store y la Play Store, pero ser un ganador entre esta dura competencia requiere mucha paciencia y trabajo.

Con tantos juegos llegando al mercado diariamente, hay que ser diferente para destacar y dejar huella. Los niveles complejos no son suficientes para que los usuarios se enganchen a un juego y lo recomienden a sus amigos, ya que si el juego no funciona como los usuarios esperan, no tardarán en desinstalarlo y pasarse a otro.

1.1. Tipos de técnicas de pruebas de juegos

Los encargados de las pruebas de control de calidad en un estudio de videojuegos (QA, "Quality Assurance") buscan los problemas y los puntos débiles que se pueden pulir antes de lanzar el producto al público, además, deben asegurarse, a través de diversas técnicas, que aspectos como la conectividad emocional con el juego o el nivel de diversión estén en óptimas condiciones.

A continuación, detallamos algunas de las técnicas más comunes utilizadas por los estudios de videojuegos y los QA "testers" especialistas en pruebas de control de calidad de juegos:

Pruebas de Funcionalidad: la función de esta técnica de pruebas de juegos es comprobar si el videojuego funciona de acuerdo con las especificaciones. Algunos de los problemas que sigue la técnica de pruebas de funcionalidad son los gráficos genéricos, la interfaz, la estabilidad o los problemas mecánicos. Además, con esta técnica se pueden resolver errores como el "freezing", el "crashing" o los bloqueos de progresión.

Los “testers” del juego están atentos a cualquier fallo gráfico en la interfaz, como gráficos ausentes, colores faltantes, problemas de ubicación o problemas de animación y recortes. Después de identificar todos los fallos y errores, los “testers” del juego hacen una lista y los envían a los desarrolladores del juego para que los arreglen. Una vez que los desarrolladores resuelven los problemas, envían el juego al equipo de control de calidad para que lo vuelvan a probar.

Pruebas Combinatorias: con esta técnica de pruebas de juegos, se puede comprobar inicialmente cuántas pruebas necesita el juego. La técnica de pruebas combinatorias analiza y escudriña todas las salidas y entradas del juego para que pueda obtener una imagen clara sobre las distintas combinaciones y resultados concebibles. Este hallazgo tiene importantes implicaciones para las pruebas, ya que sugiere que probar combinaciones de parámetros puede proporcionar una detección de fallos muy eficaz.

Pruebas “ad hoc”**:** este método no es algo formal, sino que, por el contrario, todos los procedimientos se realizan de forma aleatoria. Esta técnica de pruebas de juego también se denomina “adivinación de errores” y consiste en detectar los fallos mediante un enfoque aleatorio. Por lo tanto, no es necesaria ninguna documentación especial, como documentos de requisitos, planes de pruebas, casos de prueba, ni una planificación adecuada de las pruebas en cuanto a su calendario y orden de realización.

Dado que las pruebas “ad hoc” son más bien una técnica de prueba “salvaje” que no tiene que estar estructurada, la recomendación general es que se realicen después de la ejecución del conjunto de pruebas correspondiente. Otro punto de vista es que se puede realizar cuando no se pueden realizar pruebas detalladas por falta de tiempo.

Pruebas de Compatibilidad: esta técnica de prueba de juegos permite validar si la interfaz de usuario es compatible con todos los tamaños de pantalla en los que se puede jugar. Cada dispositivo es único y puede presentar diferentes problemas, como el uso del “hardware”, los detalles gráficos, el tamaño de la pantalla o las aplicaciones de fondo.

Tras la verificación, el videojuego debe cumplir los requisitos esenciales del "software", el "hardware" y los gráficos. Realizar las pruebas de compatibilidad en dispositivos físicos es la mejor manera de determinar el rendimiento con precisión y conocer la experiencia del usuario final.

Pruebas de "cleanroom"**:** la técnica de pruebas de "cleanroom" valida y mejora el rendimiento de la consistencia y la fiabilidad del "software" de juegos. Mediante el uso de las pruebas de "cleanroom", se puede identificar la causa principal detrás de los "bugs" y pequeños errores.

El plan es que los "testers" de juegos creen pruebas que jueguen el juego igual que los jugadores. Lo que significa que les resultará más fácil averiguar qué hacen los jugadores.

Pruebas de Regresión: el propósito de esta técnica de prueba del juego es comprobar de nuevo si las funciones del juego operan de forma óptima, lo que ayuda a controlar la calidad que es tan importante para la experiencia de los jugadores. En la práctica, los "testers" de juegos vuelven a comprobar el juego y comparan los resultados actuales con los obtenidos en las pruebas. Quieren ver si se han producido nuevos errores debido a los cambios y ver si hay algún fallo antiguo.

1.2. Métodos de prueba de juegos

Aquí es donde entramos en uno de los puntos más relevantes y nos centramos en los diferentes tipos de metodologías de prueba de control de calidad de juegos disponibles

Pruebas de funcionalidad del juego

Las pruebas funcionales son una actividad que tiene como objetivo determinar si un juego funciona de acuerdo con las especificaciones e identificar cualquier error o problema que pueda afectar negativamente la experiencia del jugador. Esto se aplica tanto a las pruebas de juegos móviles como a las de otras plataformas, incluidas PC, consolas, web y VR/AR.

Podemos ordenar las pruebas en múltiples subtipos:

Tabla 1. Pruebas de juegos

Subtipos	Descripción	Cuando se necesita
Pruebas de juego de interoperabilidad	Se utiliza para probar cómo un juego interactúa con otras aplicaciones. Los probadores buscan cualquier problema de compatibilidad, incluida la inaccesibilidad de las funciones, el bajo rendimiento y los retrasos en la comunicación.	Se utiliza principalmente para juegos multijugador y aquellos que hacen uso de funciones avanzadas del dispositivo (por ejemplo, enlace ascendente de Internet, "Bluetooth", cámara, etc.).
Pruebas de juegos de regresión	Este ciclo se aplica después de las principales actualizaciones de código para asegurarse de que la actualización no haya afectado negativamente la funcionalidad existente (por ejemplo, rompiendo una función, agregando numerosos errores nuevos, etc.). El objetivo es asegurarse de que el código aún funcione.	Esta es una práctica muy común tanto durante el desarrollo (cuando se crean nuevas compilaciones) como en el período posterior al lanzamiento, cuando se implementan el mantenimiento y las actualizaciones. Puede automatizar muchas de estas pruebas.
Pruebas de juegos de humo	La prueba de humo se trata de verificar la estabilidad después de una actualización de código. Sin embargo, tiende a ser una evaluación realmente simple, verificando cosas básicas como si el juego se inicia, si la interfaz de usuario sigue respondiendo. Puede ser realizado tanto por desarrolladores como por probadores.	La prueba de humo tiende a realizarse justo antes de la prueba de regresión, por lo que podemos decir que es igual de común y esencial para todos los proyectos. Se considera una buena práctica para realizar todos los días del proyecto.

Pruebas de juego de localización	Este método tiene como objetivo garantizar que un juego sea totalmente utilizable y agradable para los jugadores de diferentes países y regiones. En primer lugar, todas las mismas características y funcionalidades deben ser accesibles en diferentes ubicaciones (a menos que se planifique lo contrario). En segundo lugar, el contenido debe adaptarse a los idiomas y culturas donde se presenta.	Este tipo de prueba es vital si planeas lanzar tu juego en varios idiomas, o si planeas hacer que el contenido esté disponible en diferentes regiones. Por lo tanto, es preferible que los evaluadores tengan un amplio conocimiento de los idiomas/culturas asociadas antes de comenzar a evaluar el "software".
Pruebas de juego de control de acceso de seguridad	Una práctica muy importante que verifica si existen lagunas o puertas de enlace no autorizadas que puedan permitir que alguien acceda al *backend* del juego o a elementos/características que de otro modo estarían restringidas. Por ejemplo, algunos jugadores intentan piratear juegos para obtener recompensas o dinero gratis en el juego, hacerse invencibles, etc. y el control de acceso lo impide o al menos reduce la probabilidad.	Aunque esta forma de prueba rara vez conduce a cambios que afecten la experiencia del usuario o que sean incluso perceptibles, se recomienda realizarla al menos una vez antes del lanzamiento y cada actualización posterior al lanzamiento. Puede ser realizado por ingenieros generales de control de calidad del juego o probadores de pluma (penetración) experimentados
Pruebas de juego de aceptación del usuario	Esta es una de las últimas etapas de prueba antes de lanzar un juego en producción. A veces, se lleva a cabo como una prueba beta, atrayendo a jugadores fuera del equipo de	Al involucrar a probadores beta externos, puede obtener una nueva perspectiva sobre la experiencia del jugador y recopilar comentarios valiosos sobre posibles mejoras que el equipo de desarrollo no

	desarrollo. Rara vez se encuentran grandes problemas, por lo que la atención se centra principalmente en pulir los errores menores restantes y realizar pequeñas mejoras en la experiencia del usuario.	conocía o no prestó suficiente atención. Por ejemplo, las pruebas beta de juegos móviles pueden atraer a un grupo de edad con ideas y deseos únicos que su equipo de pruebas de adultos no tuvo en cuenta.

1.3. ¿Qué es la automatización de pruebas de juegos?

La automatización se da en dos enfoques: en el control de calidad del juego y el otro es la prueba manual. Con la automatización, ciertas pruebas de rendimiento se pueden realizar muchas veces seguidas sin la participación del evaluador, esto ahorra tiempo y hace que sea mucho más probable detectar errores.

Aunque este enfoque es más conveniente que pedirles a los probadores que completen los mismos niveles muchas veces de forma manual, es bastante difícil crear herramientas de automatización adaptadas a una aplicación individual.

Según una encuesta de empresas de desarrollo de juegos, la mayoría prefiere probar manualmente sus juegos, citando la falta de recursos y tiempo para crear “software” de automatización.

1.4. ¿Qué hacen los QA “testers” de juegos?

Muchos jóvenes sueñan con convertirse en probadores de juegos porque pueden jugar todo el día, pero esto solo rasca la superficie de las responsabilidades del puesto. Aparte de revisar la mecánica del juego, tienen tareas menos románticas y dinámicas como:

a) Navegación y muestreo de todas las opciones del menú.
b) Ejecución de análisis de rendimiento y CPU.
c) Muestreo de todas las opciones de personalización.

d) Fallar intencionalmente en el juego y probar obstáculos.
e) Comprobación y análisis de las especificaciones del sistema.
f) Compilar informes y completar formularios de comentarios.
g) Ejecución de “scripts” de prueba y casos de prueba para aplicaciones de juegos.

Como se puede observar, hay muchas repeticiones involucradas en las pruebas de funcionalidad del juego y los expertos tienen que dedicar gran parte de su tiempo a otras tareas como hacer informes y trabajar con código y “scripts”. Para cualquiera que esté considerando este tipo de trabajo, no solo debe amar los juegos, sino también estar preparado para verlos desde un punto de vista analítico y constructivo.

1.5. ¿Cómo funciona la prueba de juegos?

Como mencionamos anteriormente, las pruebas de juegos implican más que solo iniciar un juego y jugarlo de principio a fin para asegurarse de que funcione sin problemas. Hay docenas de complejidades en el proceso de prueba del juego, así que echemos un vistazo más de cerca.

Etapas de la prueba del juego: la estructura general del proceso de control de calidad es simple, esto es:

1. Pruebe o testee el video juego.
2. Proporcione comentarios en su proceso de testeo.
3. Vuelva a probar después de realizar las mejoras.
4. Repita los mismos pasos si es necesario.

Sin embargo, ¿dónde encaja la revisión de control de calidad en el gran esquema de desarrollo? Durante las fases iniciales de desarrollo (planificación y producción), los ingenieros de control de calidad tienden a tener muy poca participación. Estas etapas son manejadas principalmente por gerentes de proyecto, expertos en desarrollo comercial y

desarrolladores. Aun así, se les puede pedir comentarios sobre cosas como la lógica y el flujo del juego, el diseño de UX y el diseño de interacción.

Algunas compañías optan por agregar también juegos de prueba beta al expediente. Con este enfoque, se pide a personas ajenas a la empresa que realicen una vista previa y una prueba beta de los juegos y lanzamientos de PC en otras plataformas (las consolas también son comunes). Esto brinda una comprensión mucho mejor de la impresión general que se puede esperar de los jugadores que la que una empresa podría obtener de unos pocos ingenieros de control de calidad.

1.6. Métricas clave de las pruebas funcionales de control de calidad

La retroalimentación sobre la calidad del juego no es algo formulado ampliamente como "El juego funciona sin problemas". Incluye cifras concretas, estadísticas y terminología que ofrece información mucho más útil.

Repasemos algunas de las métricas clave de control de calidad del juego:

Eficacia de la prueba: esta métrica se recopila en función de docenas de otras medidas, incluida la cantidad de pruebas realizadas, la cantidad de errores encontrados/corregidos y la cantidad de problemas que persisten después de la corrección.

Variables económicas: esta métrica se refiere al costo general de las pruebas y el costo por corrección de errores, la variación del costo planificado y real, y la diferencia entre el tiempo planificado y el tiempo real.

Cambiar métricas: los cambios aplicados durante las pruebas se analizan para determinar cuántos problemas se solucionaron, qué nuevos problemas surgieron y cuántos, y otros cambios estadísticamente significativos a lo largo del tiempo.

Muchos otros: dependiendo de a quién le pregunte, puede haber hasta cientos de métricas que una empresa registra y analiza.

1.7. Informes de errores

Los informes de errores son una parte indeleble de la revisión de control de calidad y representan la información/los comentarios recopilados en función de las pruebas de los ingenieros que se pueden utilizar para realizar mejoras adicionales.

Un informe de error tradicional incluirá los siguientes detalles:

1. **Resumen**: una breve descripción que identifica el problema y su impacto.
2. **Información general**: datos sobre la plataforma y el número de compilación, junto con otra información contextual relevante.
3. **Pasos para reproducir:** instrucciones para que los desarrolladores verifiquen el problema desde su propio lado.
4. **Resultados reales y esperados:** un breve texto que indica cómo el problema afecta actualmente la experiencia del jugador y cómo debería funcionar sin el problema.
5. **Otra información relevante:** cualquier otro detalle que pueda ayudar a los desarrolladores/diseñadores a comprender y resolver el problema.

1.8. Tipos principales de métodos de prueba de juegos

Cada empresa tiene sus propias percepciones sobre qué pruebas son necesarias para un juego, y la lista incluso puede variar de un proyecto a otro. Aun así, podemos delinear las pruebas más comunes, dividiéndolas en funcionales (pruebas de funcionalidad del juego) y no funcionales (asociadas con el rendimiento general y UX), veamos esto de una manera más detallada:

Tabla 2.

Principales métodos de “testing”*.*

Pruebas funcionales	Pruebas no funcionales
Componente/módulo (comprobación del rendimiento de las unidades de "software" más pequeñas).	Rendimiento (velocidad de carga del juego en tiempo real).
Integración (encontrar defectos en interfaces e interacciones de componentes).	Carga/Estrés (comprobación del rendimiento en condiciones de gran actividad/tráfico de usuarios).
Humo (determinando la estabilidad de construcción).	Instalación (qué tan bien se guarda el juego en diferentes dispositivos).
Regresión (verificación de correcciones de errores).	Usabilidad (conveniencia de los mecanismos de juego).
Localización (verificación de la consistencia del contenido traducido).	Recuperación (cómo funciona la aplicación después de fallar).
Seguridad y control de acceso (identificación de vulnerabilidades y verificación de permisos de usuarios).	

1.9. Cómo funcionan las pruebas en diferentes plataformas

Los juegos a menudo se lanzan en varias plataformas a la vez e incluso dentro del alcance de una plataforma, los evaluadores deben asegurarse de que todo funcione en diferentes dispositivos. Por lo tanto, hay muchas peculiaridades en lo que respecta a la revisión de control de calidad en diferentes plataformas, esto se entenderá de manera más detallada a continuación:

PC / Ordenador Personal: los dispositivos de escritorio tienden a tener más potencia y más flexibilidad en términos de resolución de pantalla/gráficos. Por lo tanto, si un juego se crea con varias opciones de personalización y optimización en mente, gran parte del trabajo de prueba del juego de PC se centra en asegurarse de que cada jugador con el "hardware"

compatible con el juego pueda elegir las opciones de personalización (o configurarlas automáticamente) para aprovechar al máximo la experiencia.

Además, los juegos de escritorio a menudo aprovechan los teclados grandes al admitir docenas de combinaciones de teclas. El trabajo de un probador de juegos de computadora es asegurarse de que estos comandos funcionan de manera consistente en computadoras con diferentes nombres y composiciones de teclas (por ejemplo, dispositivos macOS, Linux y Windows).

Consola: los dos aspectos únicos más importantes de la revisión por parte de un probador de consolas de juegos son: 1) garantizar el cumplimiento y la compatibilidad con los estándares de los fabricantes de consolas; 2) brindar una experiencia de juego uniforme en todas las plataformas y generaciones de consolas. En el primer caso, Xbox, Sony y Nintendo tienen pautas estrictas sobre los parámetros y el contenido de los juegos que se pueden ejecutar en su plataforma, por lo que los evaluadores deben asegurarse de que el producto cumpla con todos los requisitos.

En lo que respecta a la jugabilidad multiplataforma y multi generacional, recientemente hemos visto un ejemplo de trabajo deficiente por parte de un probador de videojuegos de PlayStation con el lanzamiento de “Cyberpunk” 2077 en PS4. Los desarrolladores tienen un trabajo muy difícil para asegurarse que su aplicación se ejecute tanto en las consolas de generación anterior más débiles, como en la próxima generación, sin mencionar los lanzamientos multiplataforma que son varias veces más complejos.

A su vez, un probador de juegos funcional puede pasar meses eliminando los errores y problemas de rendimiento que surgen antes y después del lanzamiento.

Móvil: una de las mayores dificultades en el desarrollo para dispositivos móviles es hacer uso del espacio de pantalla limitado, por lo que cualquier persona que pruebe juegos móviles de AB debe trabajar duro para verificar que los jugadores puedan acceder a todas las funciones del juego incluso en las pantallas de teléfonos inteligentes más pequeñas.

Otra peculiaridad de estas aplicaciones es que suelen tener funciones multijugador y de redes sociales, que son notoriamente difíciles de desarrollar correctamente. Por lo tanto,

las pruebas de carga de juegos móviles y la evaluación multiplataforma son parte integral de la experiencia de control de calidad móvil tradicional.

Por último, es más probable que las aplicaciones móviles utilicen código y elementos de fuente abierta que otras plataformas, lo que las deja vulnerables a piratas informáticos y actores malintencionados. Los desarrolladores a menudo desconocen estos problemas, pero el probador de control de calidad del juego móvil debe estar atento para encontrar y cubrir cualquier vulnerabilidad.

VR/AR/MR: las aplicaciones inmersivas presentan desafíos únicos para los probadores, pero que se pueden superar. Por ejemplo, el elemento de movimiento de la mayoría de los juegos de realidad virtual puede causar mareos y mareos en algunos jugadores, lo que los desarrolladores a menudo no tienen en cuenta.

Además de eso, quienes prueban los juegos de realidad virtual deben asegurarse de que el juego transmita adecuadamente las recomendaciones de seguridad, como despejar el área donde el usuario está jugando. De lo contrario, existe la posibilidad de que se lastimen.

2. Cómo iniciar las pruebas o “testing” en un videojuego

En teoría, cualquiera puede aprender a probar un juego, incluso si no está capacitado para ser un ingeniero de control de calidad. Sin embargo, tener éxito y eficacia en las pruebas es otra cuestión, y los proyectos a gran escala generalmente necesitan evaluadores calificados para tener éxito. Por lo tanto, puede considerar contratar a estos profesionales o delegar las pruebas a una empresa con amplia experiencia en el campo.

Cualquiera que sea la opción que elija, el proceso de prueba será más o menos el mismo. Entendemos que las docenas de tipos de pruebas que enumeramos son únicas y requieren “software”, metodología y conocimientos diferentes, pero las etapas de las pruebas suelen ser uniformes como se observa a continuación:

1. **Planificación:** tienes que caminar antes de correr y las pruebas no son diferentes. En lugar de probar cuando surge la necesidad, puede hacer que el proceso sea más eficiente creando un plan integral con una lista de casos de prueba y cronología.
2. **Pruebas:** no hay mucho que decir sobre esta etapa. Una prueba puede durar entre unos minutos y varios días. No es raro que se realicen varias pruebas simultáneamente.
3. **Informes de errores y correcciones:** los ingenieros de control de calidad encuentran muchos tipos de errores en las pruebas de juegos y es su trabajo documentarlos todos. Los datos de la prueba y los errores/problemas identificados se empaquetan en un informe de errores y se envían a los desarrolladores para que los corrijan en una compilación futura.
4. **Informes de errores y correcciones:** los ingenieros de control de calidad encuentran muchos tipos de errores en las pruebas de juegos y es su trabajo documentarlos todos. Los datos de la prueba y los errores/problemas identificados se empaquetan en un informe de errores y se envían a los desarrolladores para que los corrijan en una compilación futura.

Evaluación y pruebas

Este apartado tiene como objetivo comentar las pruebas realizadas en el juego desarrollado. Las pruebas son un elemento fundamental en un desarrollo “software” y deben realizarse siempre para evitar fallos no deseados. Existen los siguientes tipos de pruebas:

Pruebas unitarias: estas pruebas tienen como objetivo verificar la funcionalidad y estructura de cada componente individual de código realizado. Son pruebas a nivel de métodos codificados.

Pruebas de integración: su objetivo es comprobar el correcto ensamblaje entre los distintos componentes del código, cubren la funcionalidad establecida y se ajustan, en mayor medida, a los requisitos funcionales.

Pruebas de sistema: están destinadas a probar el sistema como un todo y se centran en los requisitos no funcionales como por ejemplo la gestión de recursos, compatibilidad, usabilidad. Suelen estar enfocadas a la gestión de los recursos como el procesador o la memoria.

Pruebas de aceptación: se usan para validar que el sistema cumple con el funcionamiento esperado y permitir al usuario del sistema que determine su aceptación sobre la funcionalidad y el rendimiento.

3. Utilizando el Framework Test Runner – "unit testing" en Unity

Se ha decidido crear un solo epígrafe para las pruebas tanto unitarias como de integración dado que, por la naturaleza del propio programa, Unity y de los juegos como "software" desarrollado es casi imposible encontrar elementos aislados de código que funcionen sin depender de otros.

Es importante mencionar que en este apartado aparte de las pruebas realizadas también se han considerado como pruebas los logs generados durante el desarrollo del proyecto, así como las ejecuciones en el editor como parte de la integración porque al tratarse de un juego, los elementos debían coordinarse a la perfección para no obtener "bugs" indeseados.

De esta manera se subsanan errores que aparecían al producirse excepciones en el código. Para llevar a cabo estas pruebas se ha hecho uso de una funcionalidad implementada por Unity llamada Test Runner que permite su configuración y ejecución.

Algo que la mayoría de las veces solemos dejar completamente de lado a la hora de desarrollar nuevas "features" en nuestros juegos son, sin duda, los Unit Tests. El Unity Test Runner hace uso de la librería NUnit que es una librería de pruebas de "software" libre para lenguajes .Net. El Test Runner permite ejecutar los test tanto en modo editor ("edit mode") como en modo de juego ("play mode").

Sin embargo, el uso de estas herramientas que proporciona el "software" de Unity 3D se deja casi siempre de lado ya sea por poco conocimiento sobre el uso de este "framework", ya que este posee muchas bondades. De esta manera cabe resaltar que la implementación de estos test en nuestros proyectos siempre nos va a demandar un buen gasto de tiempo y esfuerzo extra que no siempre podemos o estamos dispuestos a perder, sumado a la falsa sensación de pérdida de tiempo por estar invirtiendo una parte importante de nuestro trabajo en desarrollar código que ni siquiera pertenece al producto final, puede que el hacer un repaso de algunos de sus pros y contras nos haga verlo de otro modo. Al respecto:

Ventajas.

a) Aplicar test en nuestro código nos da seguridad y confianza en el comportamiento esperado de las lógicas que implementamos.
b) Algo muy importante es que sirve como documentación para nuevos desarrolladores que trabajen en tu código.
c) Nos obliga a implementar las cosas enfocándonos siempre a su testeo.
d) Nos ayuda a anticipar y localizar posibles "bugs" de manera más rápida y eficiente.
e) Nos convierte en mejores profesionales.

Desventajas.

a) Inversión de tiempo extra de desarrollo (a veces puede llegar a ser demasiado).
b) No siempre es fácil testear todo (por ejemplo, comportamientos de UI).
c) No implementarlos de manera adecuada puede crearnos una falsa seguridad que puede ser peligrosa.

La clave está en mantener el equilibrio que las propias circunstancias de nuestro proyecto nos permitan. No se trata de hacer "testing" de todo o de nada, siempre va a ser mejor hacerlo de algunas partes que no hacerlo de nada.

Por ello, creo que es posible buscar siempre un balance que nos permita llevar adelante el desarrollo de nuestro proyecto de manera que encaje en tiempos/planificaciones y, a su vez, tenga en cuenta la implementación de test en la medida de lo posible (un enfoque correcto sería por ejemplo priorizar "unit testing" para aquellas mecánicas que consideremos más importantes o prioritarias).

Configurar Test Runner

Antes de comenzar a desarrollar el código para los respectivos tests, se debe inicialmente configurar Test Runner, que es la herramienta que viene integrada con Unity para gestionar todo lo relacionado con "unit testing" y la cual utiliza el "framework" NUnit. Veamos los siguientes pasos:

.

Video 2. Colocar aquí el nombre del video que está en el canal

Enlace de reproducción del video

Síntesis del video: Colocar aquí el nombre del video del canal

Ejercicio práctico: querido aprendiz, con el fin de iniciar la práctica de los temas que se abordan y las herramientas, le invitamos a desarrollar este ejercicio, atendiendo los siguientes puntos:

Es momento de abrir su propio script "CheckPointTests" y limpiarlo para partir de cero, así:

```
1     using   NUnit.Framework;
2     using   System.Collections;
3     using   UnityEngine;
4     using   UnityEngine.TestTools;
5
6     namespace   Tests
7     {
8           public class CheckPointTests
9           {
10                // Aquí escribiremos nuestros tests...
11                 }
12    }
```

En este ejercicio, a modo de ejemplo, se abordará la implementación de los siguientes casos de prueba que, aunque por definición no puedan considerarse 100 % tests unitarios, ya que contemplan el testeo de "features" más generales, son perfectamente válidos y servirán para ilustrar el desarrollo de tests (queda a criterio propio el plantear todos los casos que estimes necesarios):

1. Caso de prueba para que un "checkpoint" se cree correctamente.
2. Caso de prueba para que un "checkpoint" sea activado por el player correctamente.
3. Caso de prueba para que el player, tras morir, sea teletransportado al último "checkpoint" activado.

Debe iniciar entonces por declarar dentro de la clase, las siguientes variables que son necesarias a lo largo de las pruebas:

```
1     public class CheckPointTests
2     {
3      private   GameObject   player ;
4            private   GameObject   enemy;
5            private   CheckPoint   newCheckPoint ;
6     }
```

Antes de iniciar a escribir el código del primer caso de prueba conviene señalar la existencia de 2 atributos especiales que van a ser útiles a la hora de implementar sus funciones de test en Unity.

Se trata **de** [“SetUp”] y [“TearDown”].

[“SetUp”]: este atributo, colocado encima de una de las funciones de su script de tests, específica a Unity que dicha función debe ser ejecutada antes de la ejecución de cada una de las funciones de test normales.

Esto permite, por ejemplo, crear e inicializar objetos comunes a todas las pruebas sin tener que estar realizándose al inicio de cada prueba. Por tanto, empiece por crear la función de “SetUp”:

```
1  [Setup]
2   public void Setup ()
3   {
4      player = MonoBehaviour.Instantiate(Resources.Load<Game0bject>(“Prefabs/Player”), new
       Vector3(0, 0, 0), Quaternion. identity);
5      player.GetComponent<Rigidbody>().useGravity = false;
6
7      enemy = MonoBehaviour.Instantiate(Resources.Load<Game0bject>(“Prefabs/Enemy”), new
       Vector3(200, 0, 0), Quaternion. identity);
8      enemy .GetComponent <Rigidbody>().useGravity = false;
9    }
```

En este caso se hace lo siguiente:

a) Instanciar el "prefab" del "player" en el punto (0, 0, 0) de la escena.
b) Desactivar la gravedad (esto se hace para facilitar las pruebas, ya que en esta escena de prueba no se cuenta con un suelo).
c) Instanciar el "prefab" del enemigo en un punto alejado al "player" (200, 0, 0).
d) Desactivar la gravedad (por el mismo motivo de antes).

["TearDown"]: este atributo, colocado encima de una de las funciones del script de tests, específica a Unity que dicha función debe ser ejecutada después de la ejecución de cada una de las funciones de test normales. Esto permite, por ejemplo, destruir o limpiar objetos comunes a todas las pruebas sin tener que estar realizándose al final de cada prueba. Por tanto, debe crear la función de TearDown:

```
1     [TearDown]
2     public void Teardown()
3     {
4           Game0bject.Destroy (player);
5           Game0bject.Destroy (enemy);
6           Game0bject.Destroy (newCheckPoint.gameObject);
7           CheckPoint.CheckPointsList.Clear();
8     }
```

En este caso se hace lo siguiente:

a) Destruir el objeto instanciado del "player".
b) Destruir el objeto instanciado del enemigo.
c) Destruir el objeto instanciado del "checkpoint".
d) Limpiar la lista de "CheckPoints" en escena.

Bien, ya puede empezar a escribir sus 3 casos de prueba.

Empecemos con el primero: la creación correcta de un “checkpoint” en escena:

```
1 [UnityTest]
2 public IEnumerator CheckPointIsCreatedProper1y()
3 {
4      Game0bject checkPointGame0bject
       MonoBehaviour.Instantiate(Resources.Load<Gameobject>("Prefabs/CheckPoint"),
       new Vector3(100, 0, Quaternion. identity);
5      newCheckPoint = checkPointGameobject.GetComponent<CheckPoint>();
6      yield return new WaitForSeconds(0.1f);
7
8      Assert.AreEqual(1, CheckPoint.CheckPointsList.Count, “There should be only 1
       checkpoint created.”);
9      Assert.False(nenCheckPoint.Activated, "The created checkpoint should be
       deactivated.”);
10 }
```

En este caso se hace lo siguiente:

a) Instanciar el “prefab” del “checkpoint” en el punto (100, 0, 0) de la escena, para que no esté tocando al player.
b) Obtener su componente “checkpoint”, que es donde está el código que se quiere probar del juego (producción).
c) Esperar 0.1 segundos (esto es bastante común hacerlo en los tests cuando se quiere asegurar de darle tiempo a Unity para que refleje ciertas operaciones en la escena, en este caso la instancia del “checkpoint”).

d) Una vez instanciado el “checkpoint”, realizar los siguientes chequeos:
El número de objetos en la lista de “CheckPoints” en escena debe ser 1.
El nuevo “checkpoint” debe estar desactivado.

Analicemos el siguiente código, porque hay varias cosas interesantes que comentar:

Uso del atributo [Unity Test]: este atributo lo usaremos en cada una de nuestras funciones de test y le indica a Unity que es uno de nuestros test y que debe ejecutarlo dentro de su Test Suite correspondiente.

Nombre de la función: es siempre recomendable que los nombres de las funciones referentes a nuestros tests sean lo más claros e intuitivos posible, por tanto, no te preocupes si queda un nombre demasiado largo.

IEnumerator: nuestras funciones de test pueden no devolver nada o pueden también devolver un IEnumerator (convirtiéndose en corutinas). En este caso hemos necesitado que se comporte como corutina debido a la espera de 0.1 segundos que hacemos en el paso 3 de nuestra lógica.

Assertions: como puedes observar en el paso 4 de nuestra lógica, se hace uso de la clase Assert para realizar los diferentes chequeos de nuestro caso de prueba. Para mí una de las funciones más utilizadas es la de AreEqueal(), ya que con ella puedes contemplar la mayoría de casos, pero siéntete libre de explorar todas las que ofrece. Por último, recalcar el último parámetro de tipo string que le pasamos a los assertions: sirve para que la consola del Test Runner nos dé un mensaje personalizado cuando un determinado Assert NO se ha cumplido.

A partir de ahora ya maneja lo básico y necesario para realizar cualquier tipo de test. Así que acabemos de ver los siguientes casos de prueba:

«La activación correcta de un “checkpoint”» y «El posicionamiento del “player” al último checkpoint después de morir»

```
1 [UnityTest]
2 public IEnumerator CheckPointTsActivatedByThePlayer()
3 {
4     yield return CheckPointIsCreatedProperly();
5
6     player.transform.position = newCheckPoint.transform.position;
7     yield return new WaitForSeconds(0.1f);
8
9      Assert.True(newCheckPoint.Activated, "The checkpoint should be activated
      after player enters in it.");
10 }
11
12 [UnityTest]
13 public IEnumerator PlayerAppearsInTheRightCheckPointAfterDeath()
14 {
15    yield return CheckPointIsactivatedByThePlayer();
16
17    player.transform.position = enemy.transform.position;
18    yield return new WaitForSeconds(0.1f);
19
20    Assert.True(player.transform.position == newCheckPoint.transform.position,
   "After death, the player should be in the same position as the last activated
   checkpoint.”);
21 }
```

Nota: como vemos, dentro de una función de test, podemos perfectamente hacer llamadas a otras funciones de test siempre y cuando tenga sentido para nuestras pruebas.

Ejecutar tests

En este punto, lo único que nos queda es realizar la ejecución de las pruebas. Para ello, tenga en cuenta lo que se indica en el siguiente video:

Video 3. Colocar aquí el nombre del video que está en el canal

Enlace de reproducción del video

Síntesis del video: Colocar aquí el nombre del video del canal

¡Disfruta del “testing”!

4. Pruebas del Sistema

Se dividieron las pruebas de sistema en tres bloques diferenciados atendiendo a los requisitos o funcionalidades establecidos previamente en este documento, como son:

Pruebas de portabilidad

En cuanto a la portabilidad se había pedido que el juego pudiera ser ejecutado de manera multiplataforma. Se seleccionaron entre las plataformas disponibles las más usadas en el mundo de la informática hoy en día. Estas plataformas son los ordenadores y los dispositivos móviles como teléfonos o tabletas. De entre esta gran variedad de plataformas

se decidió que, en cuanto al ordenador, el juego pudiera ser ejecutado en entornos Windows y que en móviles pudiera ser ejecutado en entornos Android, a pesar de que hay otras alternativas como MacOS para sobremesa o IOS en móviles.

Para garantizar el cumplimiento de estos requisitos no funcionales se estableció que la prueba consistiría en que se pudiera ejecutar correctamente el juego en la plataforma deseada. Después de poder ejecutar el juego en las distintas plataformas sin ningún problema se concluyó que la prueba estaba superada.

Pruebas de compatibilidad

En cuanto a la compatibilidad lo que se buscaba era que, al tratarse de un juego multijugador, éste permitiera que se jugará desde distintos dispositivos la misma partida. A esto se le denomina, en el argot de los videojuegos, juego cruzado. Para realizar estas pruebas se procedió a crear una partida con dos jugadores.

Uno accedió al juego vía ordenador y el segundo desde su dispositivo móvil Android. Se jugó la partida completa y se concluyó entonces que la partida podía ser jugada sin ningún problema, puesto que no se produjo ningún error. Para el juego es indistinto que una persona se conecte desde un dispositivo u otro.

Pruebas de rendimiento

Para realizar y analizar las pruebas de rendimiento se hizo uso de la herramienta Profiler de Unity. El Profiler ayuda a optimizar el juego. Se trata de un reporte en ejecución que calcula el tiempo empleado en la ejecución distintas áreas del juego o los recursos consumidos, esto lo puede revisar de manera más detallada en el video que se expone a continuación:

Video 4. Colocar aquí el nombre del video que está en el canal

Enlace de reproducción del video

Síntesis del video: Colocar aquí el nombre del video del canal

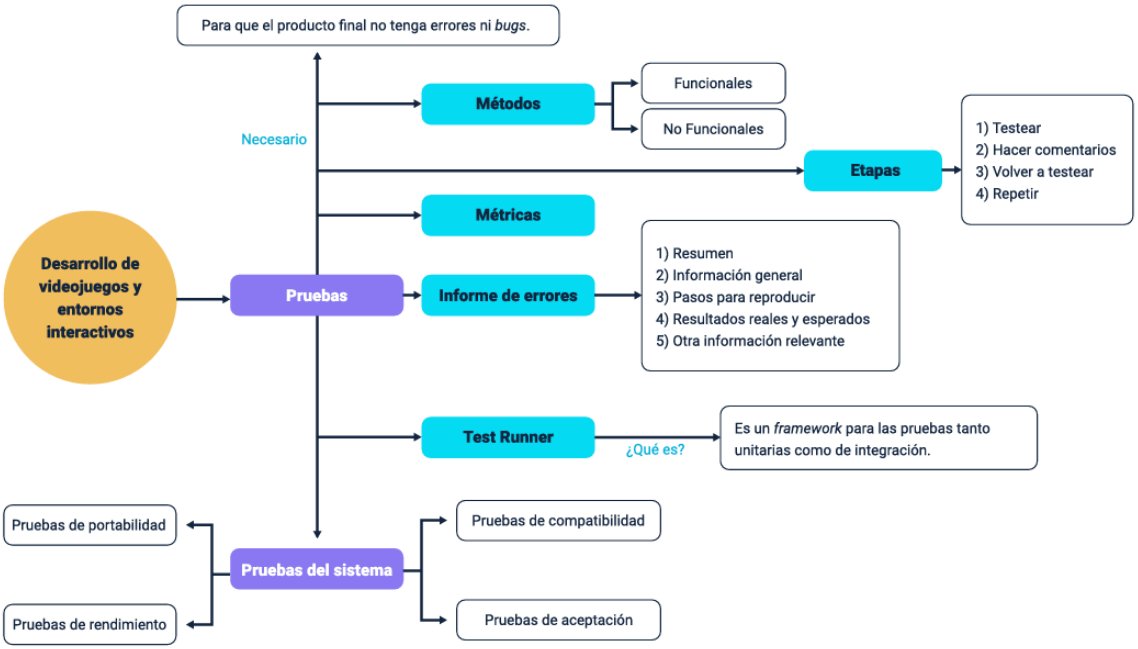
Pruebas de aceptación

Una vez finalizado el juego se deben realizar, también, las pruebas de aceptación. Para realizar dichas pruebas se debe acudir mínimo a dos fuentes, con el objetivo de que prueben el juego y den sus impresiones sobre la apariencia física del mismo. Esto proporcionará nuevos reportes y añadirá una nueva dimensión a las pruebas de aceptación. Una nueva dimensión que será tenida en cuenta para trabajos próximos.

Recuerde explorar los demás recursos que se encuentran disponibles en este componente formativo; la síntesis, la actividad didáctica para reforzar los conceptos estudiados, material complementario, entre otros.

Síntesis

Si ha llegado a este punto, usted ha finalizado con el estudio de los contenidos de este componente formativo. Aquí, haga un análisis de la estructura que se muestra a continuación. Registre esta síntesis en su libreta personal de apuntes y repase los temas que considere más importantes. **¡Adelante!**



Material complementario

Tema	Referencia APA del Material	Tipo de material	Enlace del Recurso o Archivo del documento material
Cómo iniciar las pruebas o "testing" en un videojuego.	Unity Manual. (s.f.). *Unity Test "Framework" overview.*	Otro	https://docs.unity3d.com/Packages/com.unity.test-framework@1.3/manual/index.html
Pruebas del sistema.	Kodeco. (2019). *Introduction To Unity Unit "testing"*	Otro	https://www.kodeco.com/9454-introduction-to-unity-unit-testing

Glosario

“**Ad hoc”:** que es apropiado, adecuado o especialmente dispuesto para un determinado fin.

CPU: una unidad central de procesamiento o CPU, es una pieza de “hardware” que permite que tu computadora interactúe con todas las aplicaciones y programas instalados.

“Framework”: es un esquema o marco de trabajo que ofrece una estructura base para elaborar un proyecto con objetivos específicos, una especie de plantilla que sirve como punto de partida para la organización y desarrollo de “software”.

QA: un QA (Quality Assurance) o analista QA es el profesional responsable de asegurar la calidad del “software” y de prevenir fallos en él.

SetUp: es una herramienta de los sistemas operativos y los programas informáticos que permite configurar diversas opciones de acuerdo a las necesidades del usuario.

Teastear / “testing”**:** testear es obtener la información necesaria para mejorar el sistema que se está probando, pero también para mejorar los propios procesos de desarrollo y de pruebas. Hay diferentes objetivos en el “testing”: Buscar los defectos. Ganar confianza respecto al nivel de calidad.

Test Runner: un test runner es una utilidad que nos permite escribir y correr tests para una aplicación. Existen gran variedad de tests runner como por ejemplo mocha.

UX: el diseño UX se refiere al término «diseño de experiencia de usuario», mientras que UI significa «diseño de interfaz de usuario».

Referencias bibliográficas

Awad, W. (2021). *Game "testing" Automation* Guidance. https://www.theseus.fi/bitstream/handle/10024/505977/Game%20Testing%20Automation%20Guidance.pdf?sequence=2&isAllowed=y

Da Silva Lima, G. et al. (2021). *Devops methodology in game development with Unity 3D.* https://www.ihci-conf.org/wp-content/uploads/2021/07/04_202105C029_Lima.pdf

Koepke, B, Pelletier, B., Adair, D., Jhawar, R., Macaulay, I. & Bielecki, T. (2013). *Agile Game Development*. http://kremer.cpsc.ucalgary.ca/courses/seng403/W2013/papers/05GameDevelopment.pdf

Créditos

Nombre	Cargo	Regional y Centro de Formación
Julián David Giraldo Baena	Experto Temático	Regional Risaralda - Centro de Diseño e Innovación Tecnológica Industrial
Luz Aída Quintero Velásquez	Diseñadora Instruccional	Regional Distrito Capital, Centro de Gestión Industrial
Gloria Amparo López Escudero	Diseñadora Instruccional	Regional Norte de Santander - Centro de la Industria, la Empresa y Los Servicios
Andrés Felipe Velandia Espitia	Asesor Metodológico	Regional Distrito Capital – Centro de Diseño y Metrología
Aristizábal Gutiérrez Claudia Patricia	Líder Ecosistema de Recursos Educativos Digitales (RED)	Regional Distrito Capital- Centro de Gestión de Mercados, Logística y Tecnologías de la Información.
Morales Gualdron Liliana Victoria	Responsable de la línea de Producción -2023	Regional Distrito Capital- Centro de Gestión de Mercados, Logística y Tecnologías de la Información.
López Escudero Gloria Amparo	Diseñador Instruccional	Regional Distrito Capital- Centro de Gestión de Mercados, Logística y Tecnologías de la Información.
Álzate Suarez Gloria Lida	Diseñador Instruccional	Regional Distrito Capital- Centro de Gestión de Mercados, Logística y Tecnologías de la Información.
Parra Guarín Nelly	Diseñador Instruccional	Regional Distrito Capital- Centro de Gestión de Mercados, Logística y Tecnologías de la Información.
Chinchilla Rueda Alix Cecilia	Metodólogo para la Formación Virtual	Regional Distrito Capital- Centro de Gestión de Mercados, Logística y Tecnologías de la Información.
Velandia Espitia Andrés Felipe	Metodólogo para la Formación Virtual	Regional Distrito Capital- Centro de Gestión de Mercados, Logística y Tecnologías de la Información.
Coca Salazar Carolina	Evaluador para Contenidos Inclusivos y Accesibles	Regional Distrito Capital- Centro de Gestión de Mercados, Logística y Tecnologías de la Información.

Echavarría Orozco Manuel Felipe	Desarrollador Full- Stack	Regional Distrito Capital- Centro de Gestión de Mercados, Logística y Tecnologías de la Información.
Urueta Álvarez Jhon Jairo	Desarrollador Full- Stack	Regional Distrito Capital- Centro de Gestión de Mercados, Logística y Tecnologías de la Información.
Pérez Madariaga Luis Jesús	Desarrollador Full- Stack	Regional Distrito Capital- Centro de Gestión de Mercados, Logística y Tecnologías de la Información.
Velasco Guiza Diego Fernando	Desarrollador Full- Stack Junior	Regional Distrito Capital- Centro de Gestión de Mercados, Logística y Tecnologías de la Información.
Castañeda Oviedo Jhon Édison	Desarrollador Full- Stack Junior	Regional Distrito Capital- Centro de Gestión de Mercados, Logística y Tecnologías de la Información.
Figueroa Pacheco Yazmin Rocío	Diseñador de Contenidos Digitales	Regional Distrito Capital- Centro de Gestión de Mercados, Logística y Tecnologías de la Información.
Suarez Eljure Adriana Marcela	Diseñador de Contenidos Digitales	Regional Distrito Capital- Centro de Gestión de Mercados, Logística y Tecnologías de la Información.
Orduz Amezquita Eulises	Diseñador de Contenidos Digitales	Regional Distrito Capital- Centro de Gestión de Mercados, Logística y Tecnologías de la Información.
Rey Quiñonez Yuly Andrea	Diseñador de Contenidos Digitales	Regional Distrito Capital- Centro de Gestión de Mercados, Logística y Tecnologías de la Información.
Vecino Valero Jesús Antonio	Diseñador de Contenidos Digitales	Regional Distrito Capital- Centro de Gestión de Mercados, Logística y Tecnologías de la Información.
Navarro Jaimes Ernesto	Animador y Productor Multimedia	Regional Distrito Capital- Centro de Gestión de Mercados, Logística y Tecnologías de la Información.
Ariza Luque Lady Adriana	Animador y Productor Multimedia	Regional Distrito Capital- Centro de Gestión de Mercados, Logística y Tecnologías de la Información.
Murcia Pardo Laura Gisselle	Animador y Productor Multimedia	Regional Distrito Capital- Centro de Gestión de Mercados, Logística y Tecnologías de la Información.
Pérez Manchego Lina Marcela	Validador de Recursos Digitales	Regional Distrito Capital- Centro de Gestión de Mercados, Logística y Tecnologías de la Información.
Castaño Pérez Leyson Fabián	Validador de Recursos Digitales	Regional Distrito Capital- Centro de Gestión de Mercados, Logística y Tecnologías de la Información.