Herramienta de soporte para la evaluación de videojuegos
Tool for suporting the evaluation of videogames

Trabajo de Fin de Grado
Curso 2022–2023

Autor
Jorge Bello Martín, Eva Lucas Leiro

Director
Guillermo Jiménez Díaz

Colaborador
Colaborador 1
Colaborador 2

Grado en Desarrollo de Videojuegos
Facultad de Informática
Universidad Complutense de Madrid

Herramienta de soporte para la evaluación de videojuegos
Tool for suporting the evaluation of videogames

Trabajo de Fin de Grado en Desarrollo de Videojuegos

Autor
Jorge Bello Martín, Eva Lucas Leiro

Director
Guillermo Jiménez Díaz

Colaborador
Colaborador 1
Colaborador 2

Convocatoria: *Junio 2023*

Grado en Desarrollo de Videojuegos
Facultad de Informática
Universidad Complutense de Madrid

23 de octubre de 2023

Dedicatoria

A Pedro Pablo y Marco Antonio, por crear
TeXiS e iluminar nuestro camino

Agradecimientos

A Guillermo, por el tiempo empleado en hacer estas plantillas. A Adrián, Enrique y Nacho, por sus comentarios para mejorar lo que hicimos. Y a Narciso, a quien no le ha hecho falta el Anillo Único para coordinarnos a todos.

Resumen

Herramienta de soporte para la evaluación de videojuegos

Un resumen en castellano de media página, incluyendo el título en castellano. A continuación, se escribirá una lista de no más de 10 palabras clave.

Palabras clave

Máximo 10 palabras clave separadas por comas

Abstract

Tool for suporting the evaluation of videogames

An abstract in English, half a page long, including the title in English. Below, a list with no more than 10 keywords.

Keywords

10 keywords max., separated by commas.

Índice

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Motivación

1.2. Objetivos

El objetivo de la investigación es el desarrollo de una herramienta distribuida como un paquete para Unity que facilite a los desarrolladores la implantación de métodos de evaluación cuantitativos y cualitativos para la fase de producción y preproducción de un videojuego. Nuestra herramienta pretende facilitar la realización de pruebas con usuarios para los desarrolladores que utilicen Unity y no dispongan de los medios o el conocimiento para realizarlas correctamente siguiendo buenas prácticas de GUR.

1.3. Plan de trabajo

Capítulo 2

Estado de la Cuestión

Capítulo 3

Descripción del Trabajo

Con el objetivo de permitir a un desarrollador de videojuegos integrar una prueba de GUR en su proyecto, la herramienta pondrá a disposición diferentes pruebas predefinidas entre las que podrá escoger a conveniencia, teniendo que preguntarse previamente algunas cuestiones: £Qué tipo de videojuego se está desarrollando? £Qué aspecto se pretende evaluar en el mismo? Una vez el desarrollador tiene claro qué tipo de prueba quiere realizar, comienza el trabajo de nuestra herramienta para poder implementarla en su videojuego.

En este apartado desarrollaremos cómo se utiliza la herramienta, qué aspectos se han tenido en cuenta a la hora de diseñar estas pruebas de usuario y qué resultados se pretenden conseguir. Para ello, utilizaremos un ejemplo concreto dentro del abanico de propuestas que disponemos en la herramienta: utilizar la herramienta para evaluar la dificultad de un videojuego o segmento del mismo donde existe una mecánica de penalización. El objetivo principal es averiguar cuál es el nivel de dificultad percibido por el usuario y cómo se ajusta al nivel de dificultad deseado por el desarrollador, partiendo de la certeza de que esta mecánica de penalización sirve como reflejo o consecuencia de los niveles de dificultad del videojuego, y por lo tanto, existe una relación directa.

En esta prueba de usuario partimos de una premisa necesaria que debe existir en el videojuego a evaluar: contiene una mecánica que se interpreta como una penalización al usuario. Definimos esta penalización como un evento del videojuego en el cual el jugador no consigue superar el obstáculo propuesto, reflejándose en una parada de la progresión (normalmente un reinicio), o, de manera más general, en una etiqueta que indicará que el jugador no ha superado el objetivo. En los videojuegos, se suele ver este evento como el clásico game over, pero se puede extender a cualquier tipo de castigo hacia el jugador: golpes recibidos en un videojuego de acción, resolución errónea en un juego de rompecabezas, etc. Pretendemos que sea lo suficientemente ambiguo como para que se pueda aplicar en muchos videojuegos, pero al mismo tiempo, lo suficientemente concreto para poder realizar una prueba de usuario más pormenorizada.

El interés en esta prueba de usuario se debe a que la mecánica de penalización está ligada directamente con las curvas de interés en el diseño de un videojuego. Se trata de una variable delicada, ya que la ausencia de veces en la que ocurre

puede llevar a que el jugador sienta menos desafío y por ende, menos interés; o por el contrario la repetición excesiva de la misma puede llevar a frustraciones. Según cómo sea el videojuego que se está desarrollando, puede existir más o menos interés en que se sucedan cierta cantidad de castigos en una partida, y por ello esta prueba empieza a ganar interés desde un punto de vista generalizado, en el que ofrezcamos las herramientas para observar ese número de penalizaciones (además de otros posibles datos relacionados con ella) y sacar conclusiones a raíz de estos datos recogidos.

La relación que existe entre la dificultad percibida y las penalizaciones es la siguiente: muy fácil pocas penalizaciones; muy difícil muchas penalizaciones. Debemos entender que lo que estamos evaluando es la dificultad, y el número de penalizaciones es una métrica en la que nos podemos apoyar. Esto quiere decir que el problema de la poca o excesiva dificultad se ve reflejado en las penalizaciones, pero no es generado por las mismas. Las penalizaciones son una consecuencia del nivel de dificultad, no un causante. Además, pueden existir otros factores que no dependan de estas penalizaciones que reflejen extremos no deseados en el nivel de dificultad.

Cabe destacar que la dificultad que se evaluará será la percepción general de la dificultad del videojuego o segmento a probar, no la dificultad de realizar una mecánica concreta o del diseño de niveles. Realizar una prueba para evaluar la dificultad general sin concretar en las particularidades del videojuego a tratar puede parecer contraproducente, pero a la hora de generalizar una implementación de esta prueba de usuario (con el objetivo principal de que sirva para el mayor número de desarrollos posibles) se debe abarcar desde un punto de vista más amplio, que no lo limite a ciertos tipos de videojuegos. Por esta razón, la definición que ofrecemos de una penalización o castigo acaba siendo la más acertada para poder empezar a realizar esta prueba de usuario. A partir de la realización de esta prueba, se podrán obtener ciertos resultados con los que poder realizar pruebas sucesivas intentando atacar de manera más concreta estos aspectos que puedan haber causado problemas específicos en cuanto a la dificultad.

En la prueba diseñada se proponen una serie de técnicas que combinarán métricas y cuestionarios que serán recogidos en diferentes momentos de la partida. Ambos apartados están diseñados para que en conjunto puedan ofrecer ciertas respuestas sobre las acciones del jugador que realiza la prueba. El hecho de que exista el cuestionario además de los datos objetivos y numéricos captados en el sistema de telemetría sobre las penalizaciones podrán hacer que el desarrollador entienda los diferentes perfiles de jugadores según cómo perciben la dificultad y cómo se han relacionado con la mecánica de penalización.

Comenzando por el apartado del cuestionario, dadas las características de la prueba definida, éste aparecerá una sola vez y sería al final de la realización de la prueba. No necesitamos que el cuestionario se muestre durante la prueba ni que lo haga más de una vez, ya que consiste en un cuestionario acerca de la dificultad sobre todo el segmento a evaluar de manera general, no de secciones de la prueba en específico. El cuestionario nos proporcionará datos subjetivos numéricos acerca de la cuestión a tratar de una manera mucho más directa. Se trata de la forma más típica de recibir feedback directo de los probadores. Siguiendo diferentes recomendaciones en cuanto a cómo realizar las preguntas y los sesgos a evitar (libro GUR, 9.4), hemos

realizado el siguiente cuestionario:

![Esto sería una captura de pantalla del cuestionario desde Unity](https://prod-files-secure.s3.us-west-2.amazonaws.com/e73f2072-0f15-4bb3-a291-db8ac100dd4e/b8ce095a-0b47-4b5d-a4ea-1b0598ad20df/Untitled.png)

Esto sería una captura de pantalla del cuestionario desde Unity

Como podemos observar, encontramos en primer lugar unas preguntas relacionadas con el perfil del probador. Este tipo de preguntas nos sirve como desarrollador para poder encontrar una relación entre las cuestiones a tratar y los diferentes tipos de jugadores. Siempre será útil poder categorizar según edad y experiencia jugando a videojuegos tanto general como específica.

En segundo lugar, nos encontramos con una breve pregunta en la cual se deberá indicar según una escala la conformidad con afirmaciones relacionadas con los picos o caídas de dificultad, frustración y habilidad. Serán las respuestas más útiles para el desarrollador, puesto que pueden tener relación directa con los datos de telemetría que describiremos posteriormente.

Finalmente hemos implementado un cuadro de texto en el que el probador podrá hacer cualquier tipo de comentario que vea pertinente sobre la dificultad. Siendo una herramienta amplia, con la intención de que sirva para el mayor número de proyectos, este tipo de preguntas son muy útiles para abarcar cualquier dato específico a cada posible videojuego que utilice nuestra herramienta, sin necesidad de tener que implementar caso por caso diferentes tipos de preguntas o elementos relacionados con la telemetría que no pudiesen utilizarse de manera general.

En combinación al cuestionario, tendremos una serie de eventos que serán recogidos por el sistema de telemetría integrado en la herramienta. Los eventos serán los siguientes:

- Evento de inicio y fin de sesión

- Eventos de progresión: comienzo de partida, comienzo de nivel, pausa de una partida, reanudación de una partida, fin de nivel (SUCCESS, LOSE, MATCH, NONE), abandono de nivel.

- Evento de penalización

- Evento de posición de objetos

Estos eventos serán recogidos con sus marcas de tiempo y ordenados según ese valor. Algunos de ellos serán opcionales o necesitarán de la implementación concreta por cada desarrollador, ya que dependerán de las características de cada videojuego. Esto se explicará detalladamente más adelante.

Por último, una vez realizada esta prueba, se pretende mostrar un análisis de los resultados en los que se evaluarán las pruebas tanto individualmente como en su conjunto, con el objetivo de dar una respuesta a la pregunta inicial en este ejemplo. Los datos propuestos a mostrar en el análisis serían los siguientes:

DATOS PARA EL CONJUNTO DE PRUEBAS:

- Media de penalizaciones por partida

- Media de tiempo por partida

- Grafico de media de penalizaciones y de tiempo por cada una de las preguntas del cuestionario, además de con la experiencia.

- Grafica de tiempo vs penalizaciones

- Media de posición de objetos durante la sesión
DATOS PARA LAS PRUEBAS INDIVIDUALMENTE:
- Número de penalizaciones del jugador en la partida
- Tiempo de partida
- Respuesta de los datos de su cuestionario.
- Posición de objeto durante la sesión.

Este tipo de relaciones entre datos y las respuestas de los cuestionarios son aplicables de forma automática y nos pueden permitir llegar a conclusiones de manera certera tanto como para las pruebas individuales como para el conjunto de todas las pruebas. Siguiendo la relación entre el número de penalizaciones y la dificultad previamente comentada, sumado a la directa percepción subjetiva de cada probador ante esta dificultad comparado con las métricas numéricas, llegar a conclusiones resultará mucho menos aventurado para los desarrolladores. Se proporciona un colchón de datos objetivos y subjetivos que se podrán relacionar siempre y cuando la propia prueba se haya cultivado en los marcos necesarios para que una se de por válida, con variables como el número de probadores general y el número de probadores según cada sub-grupo (en caso de querer sacar conclusiones según los sub-grupos) o la correcta realización de las pruebas dentro del tiempo máximo establecido.

Con este ejemplo descrito podemos ver un caso real de uso de la herramienta, pero aún debemos abordar el uso de la misma preguntándonos cómo podemos realizar correctamente estos casos y pruebas concretas y cómo se implementan en un proyecto.

COMO DISEÑADORES

Ahora explicar como esta diseñada nuestra herramienta como diseñadores sin entrar en detalles de la programacion, hacerlo desde el punto de vista del diseño de programas y usar diagramas o conceptos

Para ello, a continuación hablaremos sobre cómo está diseñada nuestra herramienta, desde el punto de vista del diseño y arquitectura de programas, aplicaciones y herramientas, qué hemos diseñado para cumplir los objetivos de la misma y cómo se pretende que se utilice. Comenzaremos por definir de manera rápida lo que ofrece nuestra herramienta a los desarrolladores y cuál sería el workflow para poder utilizarla:

INSERTAR ESTO COMO UNA IMAGEN, DIAGRAMA

Panel de ayuda - >Selección de prueba predefinida - >Integración de prueba en el proyecto - >Realización del conjunto de pruebas - >Análisis del conjunto de pruebas

En este apartado explicaremos cómo funciona nuestra herramienta y lo que ofrece al desarrollador siguiendo este diagrama.

Como se ha comentado anteriormente, nuestra investigación está dirigida para crear una herramienta que facilite a los desarrolladores la implantación de métodos de evaluación cuantitativos y cualitativos para la fase de producción y preproducción de un videojuego. Como tal, debemos cubrir no sólo los apartados de la implementación genérica y/o automática de un programa que contenga dichos métodos de evaluación, sino que debemos acercarnos a algo parecido a una herramienta de e-learning. Aunque no entremos tanto en detalle en aprender en profundidad sobre

el GUR, sí que queremos conseguir que un desarrollador que no sepa nada sobre cómo realizar correctamente una prueba de usuario tenga una guía sobre qué hacer según lo que quiera investigar, y que pueda entender cómo está funcionando la herramienta. Debemos tener siempre en mente que esta herramienta está destinada a programadores y diseñadores de videojuegos, por lo que es necesario una buena comunicación usuario-herramienta.

Por lo tanto, a modo de acercamiento hacia esta visión más didáctica de cómo usar la herramienta y cómo funciona el GUR, la herramienta dispondrá de una ventana de ayuda, en la cual existe una guía de uso de la misma: qué funciones puede realizar el desarrollador, cómo y cuando debe utilizarlas, guía de implementación en su proyecto y glosario acerca de los términos que se deban conocer previamente. Este panel de ayuda será una ventana paginada en la que el usuario podrá observar en detalle las pruebas de usuario predefinidas, y una vez seleccionada se le indicará cómo poder integrarla en el proyecto. *RETOCAR ESTO SEGUN COMO SE IMPLEMENTE O COMO TRABAJO FUTURO También permitirá la integración automática de la prueba en el proyecto en la medida de lo posible. Más información sobre esto en los apartados a continuación.*

DIAGRAMA DE LAS VENTANAS DEL HELPER

En segundo lugar, y como es lógico, la herramienta ofrecerá implementados los métodos de evaluación necesarios para poder realizar las pruebas. Aquí entrarán tanto los cuestionarios (métodos de evaluación cualitativos) y la telemetría (métodos de evaluación cuantitativos). Estos sistemas se explicarán con más detalle en el siguiente apartado, cuando los tratemos desde un punto de vista de programador. No obstante, volviendo a su diseño, los métodos de evaluación estarán implementados de manera independiente, es decir, se podrán combinar para formar las diferentes pruebas de usuario predefinidas.

Siguiendo la filosofía didáctica o de fácil uso/iniciación, dispondremos de diferentes pruebas de usuario predefinidas como la explicada en el apartado anterior. Estas pruebas han sido diseñadas previamente por nosotros, combinando cuestionarios y elementos del sistema de telemetría. Después de la investigación acerca del estado de la cuestión, conocemos ciertos patrones a seguir para obtener buenos resultados en pruebas de usuario que estén diseñadas para realizarse con equipos de desarrollo pequeños (o secciones del mismo dedicadas a las pruebas con usuario y QA). Por lo tanto, podemos ofrecer a los desarrolladores un conjunto de pruebas básicas comunes a la mayoría de videojuegos que puedan implementar en sus proyectos.

Para diseñar correctamente estas pruebas de usuario, seguimos diferentes metodologías. Estas metodologías van desde la correcta encapsulación de preguntas en un cuestionario, la duración de las pruebas o la relación entre los métodos cuantitativos con los cualitativos. Todas estas metodologías las hemos recogidos en una biblia de diseño en la que se define un workflow que seguir para crear una nueva prueba de usuario: definición del objetivo de la prueba y preguntas de investigación, definición del cuestionario, definición de qué recoger en el sistema de telemetría, y cómo implementar el análisis (WIP: relacionar con el apartado de trabajo futuro ->integrar la biblia de diseño para que los desarrolladores hagan sus propias pruebas de usuario). Una vez definida la prueba, se le pasa por un test de calidad en el que comparamos la prueba con otras pruebas predefinidas que sean casos de éxito reales,

o diferentes preguntas para saber si es una prueba de usuario útil para el objetivo de nuestra herramienta, que no podemos olvidar que es crear pruebas sencillas de entender y aplicar para la mayor cantidad de proyectos posibles. Estas pruebas de usuario predefinidas se encuentran integradas en la propia herramienta, por lo que una vez que el desarrollador acceda a la ventana de ayuda y decida qué prueba utilizar, deberá integrarla en su proyecto siguiendo las indicaciones pertinentes.

Las pruebas de usuario se han realizado de manera modular creando diferentes elementos para realizar y representar los métodos cualitativos y cuantitativos. Entrando en detalle en el apartado de los métodos cualitativos, hemos integrado los cuestionarios utilizando las herramientas del motor en el que integremos la herramienta para crear los diferentes tipos de preguntas-respuestas (más detalles en el apartado de programación). Estas preguntas estarán paginadas a conveniencia de la prueba (según el tamaño del cuestionario o diferentes secciones que se encuentren en el mismo). Es el conjunto de estas preguntas-respuestas lo que consideraremos como un cuestionario.

Los diferentes módulos que se han decidido utilizar para crear cuestionarios son los siguientes:

- Respuestas de elección según escalas (ejemplo: Indica, siguiendo la siguiente escala 0 (nada conforme) - 5 (totalmente conforme), tu grado de conformidad con la afirmación x/y/z). Se agrupan un conjunto de afirmaciones en las cuales el usuario responderá utilizando la escala

- Cuadros de texto (numéricos y alfanuméricos)

- Cuadros de selección individual o múltiple

DIAGRAMA O FOTOS DE LOS CUESTIONARIOS

Según la prueba que se esté utilizando, tendrán más o menos peso en la balanza estos cuestionarios. El contrapeso se trata de los métodos cuantitativos integrados en el sistema de telemetría. Los métodos en cuestión son los eventos que se captarán mediante el sistema de telemetría.

añadir aqui una definicion o relacion hacia el apartado donde se explique (estado de la cuestion?) qué es el sistema de telemetria y los eventos. si no, hacia el glosario

Los eventos cuantitativos se tratan de sucesos concretos y contextuales que se dan durante la partida. Registrarlos nos ayudará a tener un seguimiento del comportamiento del jugador, además de los diferentes momentos claves en la sesión de pruebas. Estos eventos se introducen en los proyectos de diferentes maneras, ya que dependerá de cómo de específicos sean, lo cual requerirá más o menos acción por parte del desarrollador. Por lo tanto, podemos catalogarlos en tres tipos diferentes de eventos, según cómo el desarrollador deba interactuar con ellos:

- Eventos automáticos: son los eventos que nuestro sistema de telemetría captará automáticamente sin necesidad de introducirse de manera individual en el proyecto. Los eventos que hemos integrado son los siguientes:

- Inicio de sesión: indicará cuando la prueba inicia.

- Fin de sesión: indicará cuando la prueba finaliza.

- Inicio del cuestionario

- Fin del cuestionario

- Eventos básicos: son eventos sencillos y genéricos que se compartirán en muchos

proyectos, pero el usuario deberá integrar de manera individual, pues dependerá de las características de su proyecto: no todos se lanzarán en el mismo momento. Los eventos básicos integrados son los siguientes:

- Inicio / fin de un nivel : El usuario puede integrarlo en caso de que su videojuego se divida en niveles.

- Pausa / reanudación de la partida: El usuario puede integrarlo en caso de que el juego pueda pausarse.

- Seguimiento de la posición de un objeto: El usuario puede integrarlo en caso de que quiera seguir la posición de un objeto.

- Eventos específicos a pruebas predefinidas: son eventos más específicos que se aplicarán únicamente en el contexto de las pruebas predefinidas. Al igual que los eventos básicos, requerirán de la integración individual por parte del usuario. Los eventos integrados son los siguientes:

- Eventos de la prueba predefinida evaluación de la dificultad donde existe una mecánica de penalización

- Registro de penalización: El evento se lanzará en el momento en el que ocurra una penalización

Al igual que las preguntas-respuestas de los cuestionarios, los eventos se recogen y se han desarrollado de manera independiente para su posterior encapsulación en cada una de las pruebas predefinidas, con el objetivo de terminar de crear el balance de lo cualitativo - cuantitativo. El panel de ayuda definido previamente indicará al desarrollador cómo debe integrar los eventos que sean recomendados o necesarios para realizar la prueba de usuario deseada y que no sean automáticos.

Ya hemos definido cómo se han diseñado los elementos necesarios para crear las diferentes pruebas predefinidas, además de cómo se utilizan en un caso real. No obstante, no hemos tratado cómo se recogen estos datos una vez se integran en el proyecto. Siguiendo el workflow definido, el siguiente paso para el desarrollador será la distribución de las pruebas de usuario. £Cómo se realiza entonces la recogida de datos de cada usuario? *RETOCAR EN EL FUTURO: AÚN NO ESTÁ CLARO SI INTERVENDREMOS EN EL PROCESO DE DISTRIBUCIÓN Y CONTACTO CON LOS USUARIOS.*

Para esta cuestión es por lo que existe el sistema de telemetría. Se tratará con más detalle posteriormente desde un punto de vista de la programación. Podemos, aún así, definir cómo funciona desde un punto de vista del diseño y su integración en el workflow del desarrollador que quiera usar nuestra herramienta.

El sistema de telemetría está formado por tres sub-sistemas: la recogida de eventos, la escritura en el registro de los mismos y su almacenaje. Anteriormente hemos explicado cómo funcionan los eventos y cómo se recogen en cada caso. De la escritura en el registro se encarga el sub-sistema de serialización, y de su almacenaje se encarga el sub-sistema de persistencia. El desarrollador podrá elegir en su proyecto entre las diferentes opciones de serialización y persistencia que ofrecerá nuestra herramienta, según las preferencias personales, puesto que es una elección que no afectan a la realización correcta de ninguna prueba predefinida.

Por lo tanto, una vez el desarrollador escoge cómo quiere que se escriba y se almacenen los registros de eventos de cada futuro usuario, podrá distribuir las pruebas

y simplemente esperar a que estos registros le lleguen de vuelta.

TERMINAR ESTE PARRAFO SEGÚN COMO ACABE SIENDO EL ANÁLISIS: SI ES AUTOMATICO, O SI EL DESARROLLADOR DEBE REALIAZARLO SEGÚN LOS DATOS OFRECIDOS

COMO PROGRAMADORES

Aqui explicar como esta programada la herramienta que explicamos justo antes metiendo las cosas necesarias sobre unity y c y la distribucion mediante la unity store

La herramienta ha decidido desarrollarse y distribuirse utilizando el motor Unity. Esta decisión viene impulsada debido a la inmensa base de usuarios existente en Unity, además de la fácil distribución de la herramienta a modo de Unity Package utilizando la Unity Asset Store. De esta manera, cualquier usuario podrá acceder a la página de descarga de la herramienta e integrarlo en su proyecto directamente, aunque también existen métodos alternativos para añadir el paquete a un proyecto de Unity. Ha sido desarrollado utilizando Unity 2021.3.18f1, pero es igual de utilizable en versiones posteriores. Cuando el usuario descarga e importa el paquete a su proyecto de Unity, podrá acceder a una carpeta con los recursos de la herramienta:

![Untitled](https://prod-files-secure.s3.us-west-2.amazonaws.com/e73f2072-0f15-4bb3-a291-db8ac100dd4e/807d6798-f443-4ad4-800a-d8918738a9a4/Untitled.png)

TODO ESTO ESTÁ SUJETO A CAMBIOS SEGÚN EVOLUCIONE LA HERRAMIENTA

Además, el desarrollador podrá acceder a la ventana de ayuda en el apartado de Window - GUR Helper.

Para poder utilizar la herramienta, el desarrollador deberá incluir el objeto GURManager que se encuentra en la carpeta Runtime/Prefabs dentro de su escena. Este objeto debera existir en la primera escena del proyecto a partir de la cual se quiera empezar a realizar la prueba. El objeto se trata del controlador principal de la herramienta. En la jerarquía de la escena, podemos ver que contiene los elementos para poder mostrar el cuestionario en pantalla (objeto hijo GURCanvas). Además, el objeto GURManager contiene el script homónimo que se encargará de controlar todos los sistemas y sub-sistemas mencionados en el apartado anterior: la persistencia, la serialización y la creación en ejecución del sistema de telemetría que se encargará de recoger los eventos. Además, también se encarga de la recogida de eventos básicos que se hacen automáticamente.

![Untitled](https://prod-files-secure.s3.us-west-2.amazonaws.com/e73f2072-0f15-4bb3-a291-db8ac100dd4e/d2ba2972-a649-4e60-a28c-cbc655c44724/Untitled.png)

![Untitled](https://prod-files-secure.s3.us-west-2.amazonaws.com/e73f2072-0f15-4bb3-a291-db8ac100dd4e/63fb6529-0f6f-49ee-948b-8601727e7a6d/Untitled.png)

En el inspector podremos elegir que tipos de persistencia y serialización queremos utilizar en el proyecto, pudiendo realizar cualquier tipo de combinatoria entre las siguientes opciones: Para la serialización, tenemos la posibilidad de guardar los archivos en formato JSON y/o binario. Estos registros se guardarán según la persistencia indicada, que será o local en el equipo en el que se ejecute la prueba (FILE) o alojada en un servidor (SERVER).

Además, el desarrollador podrá seleccionar cuando quiere que comience el cues-

tionario. Esto se encuentra en la variable Trigger Type. Disponemos de las siguientes opciones:

- CUSTOM: El desarrollador deberá realizar la llamada al inicio del cuestionario de manera personalizada, en el momento que vea conveniente.

- ONLOAD: El cuestionario será lo primero que se realizará según cargue la primera escena en la que esté el prefab GURManager.

- TIME: El cuestionario se lanzará cuando pasen *Minutes* minutos*,* siendo *Minutes* una variable que aparecerá en el inspector en caso de elegir este tipo de Trigger Type.

Además, disponemos de una variable llamada Tracker Type. Esta variable se utiliza como control o seguridad, para poder catalogar cada tipo de prueba predefinida en una serie de eventos. Para simplificar el uso de la herramienta, no queremos dar la oportunidad a que un evento que no esté relacionado con la prueba predefinida que se esté realizando pueda llegar a registrarse. Por ejemplo, si no estamos realizando la prueba predefinida evaluación de la dificultad con una mecánica de penalización, no queremos que se utilice el evento asociado y particular a esa prueba evento de penalización, porque ensuciaría los registros. Además, también nos sirve para poder dar avisos al desarrollador en caso de que se tenga que asegurar de implementar un tipo de evento para poder realizar la prueba correctamente. Siguiendo el mismo ejemplo, si estamos realizando la prueba evaluación de la dificultad con una mecánica de penalización, nos queremos asegurar que el usuario sea consciente de que debe registrar el evento asociado, así que cuando GURManager detecte este tipo de TrackerType, podrá enviar un aviso al desarrollador.

Será desde la ventana de ayuda donde indiquemos al usuario que variables debe utilizar en este prefab según qué tipo de prueba vaya a realizar, aunque también existe la posibilidad de que a raíz de esta ventana de ayuda se integre este prefab con las variables necesarias ajustadas automáticamente a la escena indicada.

Siguiendo con la estructura de la jerarquía del objeto GURManager, será dentro de Scroll View donde se encuentra el componente que se encarga de mostrar el cuestionario indicado. Los cuestionarios no son nada más que diferentes objetos que se encuentran en la carpeta Runtime/Prefabs/Tests, y que son objetos de UI que contienen de manera modular las preguntas-respuestas necesarias para cada cuestionario. El componente Add Content del Scroll View será el que contenga el cuestionario que se instanciará en el momento adecuado de la prueba, según la variable Trigger Type que se encuentra en GURManager. Variables de control indicarán al desarrollador si los prefabs introducidos son los correspondientes con la prueba predefinida que haya decidido realizar.

![Untitled](https://prod-files-secure.s3.us-west-2.amazonaws.com/e73f2072-0f15-4bb3-a291-db8ac100dd4e/b4aa9561-7f5f-4f06-9477-c8869216e03a/Untitled.png)

En cuanto a la integración de los eventos, hemos comentado que existen los eventos automáticos que se utilizan en el GURManager. Para la integración de eventos no automáticos, se podrá hacer haciendo una llamada al sistema de Tracker inicializado desde GURManager. Utilizando la línea de código Tracker.Instance.TrackSynchroEvent(Tracker.Insta siendo x() el método del evento que queramos registrar, podemos registrar cualquier evento. En la ventana de ayuda se indicará como usar según qué evento entre los disponibles o necesarios para poder realizar la prueba correctamente. *RETOCAR

EN UN FUTURO: Añadir la opcion de llamar a estos eventos con componentes, no mediante código*

Una vez introducido el GURManager correctamente siguiendo las indicaciones de la ventana de ayuda y personalizando los parámetros de serialización y persistencia deseados, se podrá realizar una build que contenga el sistema cerrado. Los resultados de la prueba se alojarán según lo indicado en el sistema de persistencia y el desarrollador podrá revisarlos una vez se distribuyan, ya sea mediante el servidor o mediante un sistema propio en el que recoja las pruebas que se alojen de manera local (por ejemplo, si las pruebas se realizan en equipos controlados en un único lugar).

Completar con el análisis de los datos desde el punto de vista del programador

Capítulo 4

Conclusiones y Trabajo Futuro

Introduction

Introduction to the subject area. This chapter contains the translation of Chapter 1.

Conclusions and Future Work

Conclusions and future lines of work. This chapter contains the translation of Chapter 4.

Contribuciones Personales

En caso de trabajos no unipersonales, cada participante indicará en la memoria su contribución al proyecto con una extensión de al menos dos páginas por cada uno de los participantes.

En caso de trabajo unipersonal, elimina esta página en el fichero `TFGTeXiS.tex` (comenta o borra la línea `\include{Capitulos/ContribucionesPersonales}`).

Estudiante 1

Al menos dos páginas con las contribuciones del estudiante 1.

Estudiante 2

Al menos dos páginas con las contribuciones del estudiante 2. En caso de que haya más estudiantes, copia y pega una de estas secciones.

Bibliografía

Y así, del mucho leer y del poco dormir, se le secó el celebro de manera que vino a perder el juicio.
(modificar en Cascaras\bibliografia.tex)

Miguel de Cervantes Saavedra

Apndice A

Título del Apéndice A

Los apéndices son secciones al final del documento en las que se agrega texto con el objetivo de ampliar los contenidos del documento principal.

Apndice B

Título del Apéndice B

Se pueden añadir los apéndices que se consideren oportunos.

Este texto se puede encontrar en el fichero Cascaras/fin.tex. Si deseas eliminarlo, basta con comentar la línea correspondiente al final del fichero TFGTeXiS.tex.

–£Qué te parece desto, Sancho? – Dijo Don Quijote –
Bien podrán los encantadores quitarme la ventura,
pero el esfuerzo y el ánimo, será imposible.

Segunda parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes

–Buena está – dijo Sancho –; fírmela vuestra merced.
–No es menester firmarla – dijo Don Quijote–,
sino solamente poner mi rúbrica.

Primera parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes