BACK ON TIME

MEMORIA DEL PROYECTO

ALEJANDRO GARCÍA MUÑOZ

Ingeniería de Software

Videojuegos

Nombre Consultor/a

Jordi Duch Gavaldà

Helio Tejedor Navarro

Profesores:

Javier Luis Cánovas Izquierdo

Ficha del Trabajo Final

Título del trabajo:	**Proyecto Back On Time**
Nombre del autor:	***Alejandro García Muñoz***
Nombre del consultor/a:	**Jordi Duch Gavaldà** **Helio Tejedor Navarro**
Nombre del PRA:	**Javier Luis Cánovas Izquierdo**
Fecha de entrega (mm/aaaa):	**01/2020**
Titulación:	**Ingeniería de Software**
Área del Trabajo Final:	**Videojuegos**
Idioma del trabajo:	**Español/Inglés**
Palabras Clave	**Unity Sigilo Puzzle**

Resumen del Trabajo (máximo 250 palabras):

Back On Time es un proyecto de software centrado en el desarrollo de un videojuego indie. Con un proceso similar al de un proyecto de software, se ha utilizado Unity para implementar un producto final que sirva de prototipo para demostrar la idea general del juego. El jugador controla a Laura, una espía con poderes para controlar el tiempo con una misión plagada de obstáculos. Laura debe evitar ser capturada por los guardias que patrullan su objetivo.

Durante la planificación, investigación, y desarrollo de este proyecto, se han utilizado fuentes diversas de estudio para crear las diferentes mecánicas planteadas. Los objetivos principales del Trabajo son crear un prototipo funcional, aprendiendo en el proceso, y llevar a cabo un proyecto de software de estas dimensiones, en solitario.

El resultado final es un producto que incluye todas las funcionalidades básicas planteadas. Un nivel con objetivos y obstáculos, en el que debemos alcanzar una meta

sin ser descubiertos.

Los objetivos principales se han cumplido, y el producto final es un prototipo que muestra lo aprendido durante los meses de desarrollo. Una planificación y gestión correcta del proyecto han permitido que se pueda implementar todo lo deseado, a pesar de los contratiempos sufridos durante el Trabajo.

Abstract (in English, 250 words or less):

Back On Time is a software project covering the development of an indie video game. Following a similar process to a regular Software Project, Unity has been used to implement a final prototype that showcases the general concepts of the game. The player controls Laura, a spy with the ability to control time, and a mission filled with obstacles. Laura must avoid being captured by the guards that patrol her objective.

During the planning, research, and development of this project, varied sources have been used to implement the designed mechanics. The main goals of Back On Time were implementing a working prototype, learning in the process, and finishing a solo Software Project in its entirety.

The end result is a product that includes all the planned basic functionalities. A playable level with objectives and obstacles, in which we must reach the goal without being detected.

The main objectives have been accomplished, and the final product is a prototype that features everything learned during development. A correct project planning and management have made is possible to implement everything, despite the setbacks suffered.

Capítulo 1: Introducción

Contexto y justificación del Trabajo

El trabajo sigue el proceso de desarrollo de un videojuego indie. En los últimos años, el mercado de los videojuegos ha estado en aumento **[1]**, con una mayor variedad y accesibilidad para desarrolladores y usuarios. Como consecuencia, el mercado de trabajadores es más variado, diferentes estudios y talentos, pero una pasión común. Esta mayor competitividad en el mercado ha hecho que sea necesario crear un portfolio de presentación de los proyectos/juegos en los que se haya trabajado.

El estudio del desarrollo del software es una materia relacionada con el desarrollo de un videojuego, ya que ambos son proyectos de software. Se pretende aplicar los conocimientos adquiridos durante la carrera para planificar, diseñar, e implementar un videojuego básico prototipo. El juego tendrá un nivel, y servirá para mostrar las mecánicas básicas del juego.

Objetivos del trabajo

Estos objetivos, aunque generales, ofrecen una idea de lo que se espera conseguir durante el Trabajo:

1. Adecuada planificación y gestión de un proyecto de videojuegos
2. Investigación y estudio de la plataforma de desarrollo, Unity
3. Diseñar el videojuego
4. Implementar un prototipo del juego con las mecánicas necesarias

Enfoque y método seguido

Se pueden utilizar diferentes herramientas para la creación de videojuegos, cuya elección depende, entre otras cosas, del producto final deseado. En nuestro caso, Back On Time está desarrollado en Unity.

Unity es una de las herramientas actuales del mercado más utilizadas. Muchas empresas indies la utilizan por su accesibilidad y flexibilidad. Además, es una herramienta gratuita, que ofrece una gran variedad de personalización. El contenido de aprendizaje, aunque disperso, es bastante accesible y generalmente gratuito.

Se podrían utilizar herramientas de pago ofrecidas en el Asset Store para mejorar el producto y su rendimiento. Sin embargo, uno de nuestros objetivos es el aprendizaje y familiarización con el software, y por eso se ha descartado utilizar herramientas de terceros.

Para llevar a cabo el proyecto, se ha tomado como referencia diversos tutoriales y lecciones disponibles en internet, tanto del propio Unity como independientes. Las diferentes mecánicas del juego han sido divididas en tareas más simples para facilitar su investigación.

Planificación del trabajo

Los recursos principales utilizados han sido Unity Learn y un conjunto de contenido educativo, en forma de vídeos y artículos/blogs. La planificación inicial del proyecto fue incluída en la PEC 1, con un roadmap y gráficos incluidos a continuación:

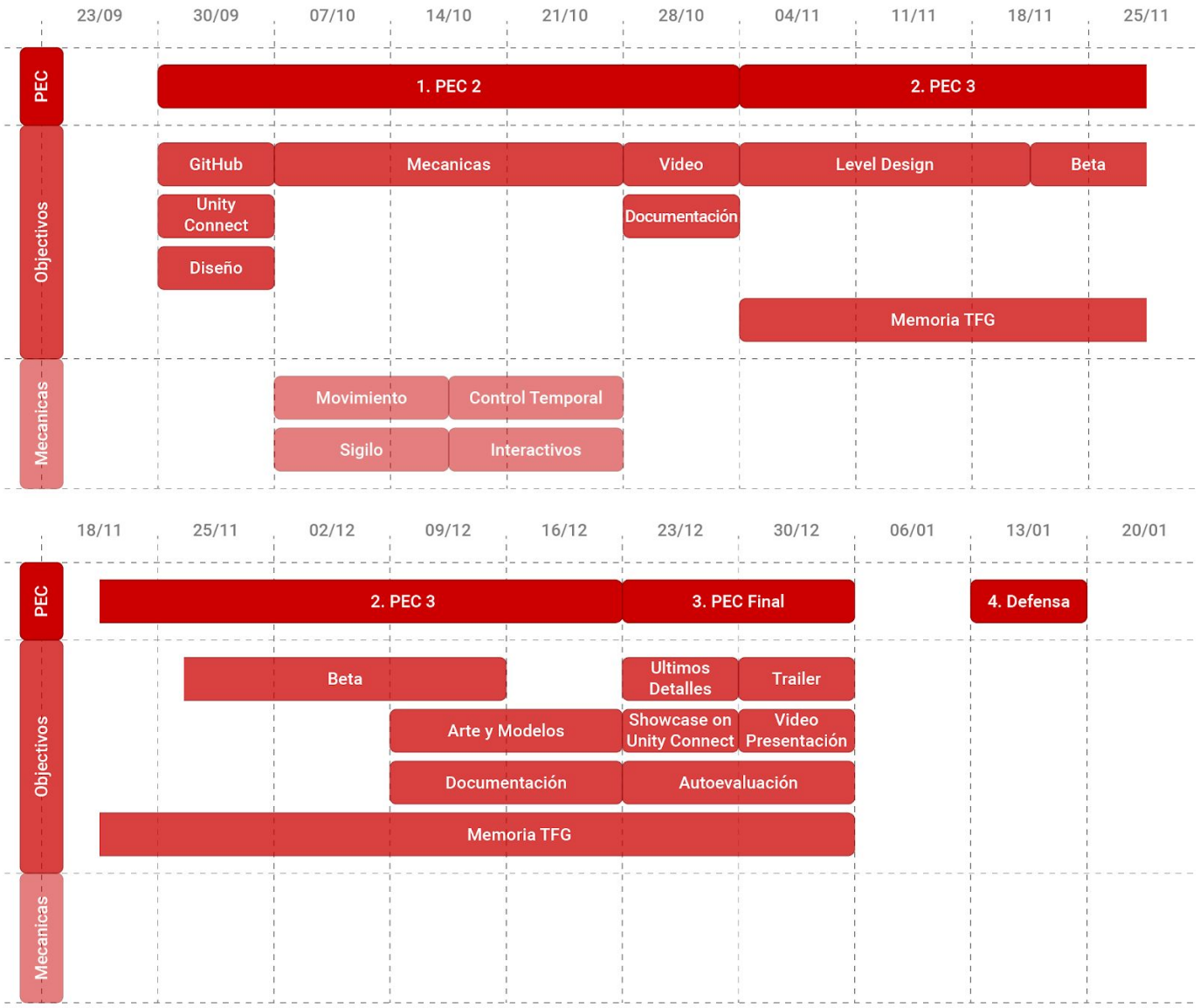
Roadmap

1. PEC 2 - Versión parcial 30/09/2019 - 03/11/2019
    - 1.1. GitHub
    - 1.2. Unity Connect
    - 1.3. Diseño de implementación
    - 1.4. Mecánicas
        - 1.4.1. Movimiento
        - 1.4.2. Sigilo
        - 1.4.3. Control temporal
        - 1.4.4. Interactivos
    - 1.5. Vídeo
    - 1.6. Documentación

2. PEC 3 - Versión jugable 04/11/2019 - 22/12/2019
    - 2.1. Empezar Memoria
    - 2.2. Level Design
    - 2.3. Implementación de Beta
    - 2.4. Documentación
    - 2.5. Búsqueda de arte y modelos (assets)

3. PEC Final - Versión final 23/12/2019 - 05/01/2020
    - 3.1. Últimos detalles juego
    - 3.2. Terminar Memoria
    - 3.3. Showcase en Unity Connect
    - 3.4. Trailer del juego (1 min. máx)
    - 3.5. Vídeo presentación (15 min. máx)
    - 3.6. Informe de autoevaluación

4. Defensa 13/01/2020 - 19/01/2020



Esta planificación inicial permite tener una idea general del proyecto y su progreso. Durante el desarrollo, la organización general ha sido sólo ligeramente alterada para adaptarse a las necesidades del proyecto.

Cada fase de Mecánicas requiere de investigación, planificación, desarrollo y testeo. Además, parte de la implementación final no está relacionada con las mecánicas de forma directa, y requiere de recursos adicionales. Estos aspectos (HUD, cámara, controles…) no fueron considerados correctamente en la planificación inicial. Sin embargo, la organización permitía extender a la PEC 3 parte de la carga de trabajo heredada de la etapa anterior.

Durante la PEC 2, se implementaron las mecánicas de Movimiento, Sigilo, parte de los poderes temporales, además de un sistema de Inteligencia Artificial (IA) y un sistema de rutinas para los NPC. Durante la investigación sobre IA, se consultaron los patrones de diseño, para encontrar si alguno se adapta a nuestras necesidades **[2]**. De este estudio se concluye que se puede aplicar el patrón Observer.

Existen numerosos Frameworks para diferentes lenguajes y aplicaciones que permiten implementar este patrón. Entre ellos está UniRx, un Reactive Extension open-source creado para Unity por la comunidad. Este tipo de programación, la programación reactiva, sería otro método para gestionar ciertas mecánicas del juego. Lamentablemente, dado el marco de tiempo disponible, no ha sido posible implementarla en el proyecto en su estado actual. Quedando pendiente de implementarse en una versión futura.

Cada semana, se revisaba el calendario, y se hacían pequeños cambios para intentar mantener una meta realista para el proyecto. Durante la PEC 3, se implementaron las mecánicas de Control Temporal e Interactivos, además de un sistema de pausa, un nivel jugable, y un sistema de objetivos, entre otros.

Breve sumario de los productos obtenidos

El producto final es un nivel básico del alfa de Back On Time. Es un juego de sigilo, en el que se deben cumplir los objetivos sin ser capturado. Laura, el personaje controlado por el jugador, tiene poderes temporales que le permiten parar, o retroceder, el tiempo. En este primer nivel, debemos alcanzar el Viejo Almacén, al final del edificio. Dos puertas de seguridad bloquean el paso, y deben de ser desactivadas utilizando los ordenadores. En nuestro camino encontraremos a guardias que intentarán evitarlo.

El juego puede jugarse con teclado, o con mando. Existen versión para Windows y para Mac.

Breve descripción de los otros capítulos de la memoria

Capítulo 2: Estado del arte

Localiza nuestro juego en la cultura actual, a través del marco de su género y la tecnología.

Capítulo 3: Definición del juego

Explora en detalle el juego, su género, influencias, y conceptualización.

Capítulo 4: Diseño técnico

Explica el entorno escogido para el desarrollo y las razones, además de otros aspectos técnicos.

Capítulo 5: Diseño de niveles

Explora el primer nivel, El Viejo Almacén, desde el punto de vista del desarrollador.

Capítulo 6: Manual de usuario

Establece los requerimientos mínimos, y ofrece una pequeña guía para el usuario.

Capítulo 2: Estado del arte

Estudio del Genero: Sigilo

Los juegos de sigilo actualmente en el mercado ofrecen diferentes tipos de retos. El género suele ofrecer la posibilidad de guardar la partida, y volver a un punto guardado si cometemos un fallo. Esta técnica se conoce como *save scumming*. *Save scumming* permite tener la sensación de no haber cometido errores, ya que navegando estos puntos, podemos conseguir una partida "perfecta". Un problema de este ciclo de juego generado, o *game loop*, es el tiempo de espera, como puede ser el invertido en pantallas de carga.

En Back On Time, el jugador puede volver atrás en el tiempo, con un margen generoso. Esto permite mantener el ciclo explicado anteriormente, aunque más fluido y limitado. Además, añade otra dimensión al juego, con mecánicas que jueguen con el concepto del control del tiempo. Actualmente, pocos juegos toman como mecánica el control temporal, el cual combina bien con el género de sigilo.

Estudio de tecnología: Unity

La mayoría de la industria del videojuego actual se basa en Unity. Muchos títulos AAA se desarrollan en Unreal, pero la industria indie usa Unity como herramienta común. Además, hay un gran número de juegos para plataformas móvil que se desarrollan en Unity.

Capítulo 3: Definición del juego

Descripción del Juego

Back on Time es un juego de sigilo en el que la protagonista tiene la capacidad de controlar el flujo del tiempo.

Localizado en un edificio gubernamental con ambientación noir/años 50. En el exterior se encuentra Laura, una espía a sueldo con poderes sobrenaturales. Su plan es entrar en el edificio, acceder al viejo almacén, y encontrar unos documentos para su último cliente.

Para llegar a su objetivo, deberá superar una serie de retos, evitando la sospecha de los civiles y empleados, mientras evade los guardias que patrullan el interior. Laura tiene la capacidad de manipular el tiempo de dos formas: puede pararlo sin ser afectada, o retroceder en él. Sus habilidades le permiten navegar por el edificio con la tranquilidad de que siempre habrá un momento seguro al que volver.

Influencias

El concepto general es un juego de sigilo con toques de control temporal. Estas dos mecánicas toman como referencias juegos de diversas fuentes.

El sigilo es similar al de la saga Hitman, Payday 2, o Commandos 2: Men of Courage (2001). Sin embargo, en lugar de ser un juego de sigilo que busca las sombras, juega con aprovechar las ventanas de oportunidades que se forman por despistes, accidentes, o casualidades. El éxito de la misión depende de nuestra habilidad para aprovechar dichos momentos, que pueden ser planeados y ejecutados con mayor precisión al poder rebobinar en caso de que nuestro plan no sea exitoso.

El aspecto del control temporal está inspirado por dos ideas principales: El día de la Marmota (película del '93 protagonizada por Bill Murray) y la saga Príncipe de Persia (Ubisoft). Al igual que Bill Murray o el Príncipe, Laura tiene la capacidad de repetir acciones, con la ventaja del conocimiento obtenido en los intentos anteriores. Esto permite al jugador construir el camino perfecto para sortear los obstáculos, y salir victorioso.

Otro juego indie de éxito reciente relacionado con este tema es *Katana Zero*, el cual ofrece una experiencia de acción con control temporal limitado.

Interacción Juego-Jugador

El jugador puede interactuar con su entorno, e.g. abrir una puerta, buscar un documento, o escuchar una conversación. Además, el entorno reacciona a Laura de varias maneras: Los guardias y cámaras reconocen a Laura, por lo que debe evitar en todo momento entrar en su campo de visión; los civiles no le prestarán atención, pero alertarán a los guardias si ven a Laura realizar alguna acción sospechosa (usar sus poderes, forzar cerraduras, robar un objeto…).

Los controles se encuentran disponibles en el menú de pausa, junto a los objetivos del nivel actual.

Conceptualización

La ambientación está basada en la estética de las películas noir de los años 40. Aunque esta sea la estética del juego, la era en la que se encuentra el mundo de Back on time es ambigua. Con un nivel tecnológico que ronda los años 90 (ordenadores, cámaras, móviles…). La historia transcurre en una pequeña ciudad europea.

El juego gira en torno a Laura, una joven con poderes sobrenaturales que trabaja como espía a sueldo. Durante varios años ha estado trabajando para clientes de mayor o menor influencia, mientras aprendía a controlar mejor sus habilidades. Hace poco, Laura fue contactada por una persona desconocida, que ha conseguido su contacto a través de un aliado mutuo. La oferta de trabajo promete una cantidad de dinero muy elevada por una

tarea bastante sencilla: recuperar un archivo en el antiguo ayuntamiento, el cual fue adquirido por una empresa privada hace varios años. Aunque tiene un mal presentimiento, la compensación económica es demasiado suculenta como para dejar pasar la oportunidad.

Personajes y elementos

Los personajes principales son: Laura, guardias, civiles, empleados, y cámaras.

Podremos interactuar con el entorno además de con los diferentes actores. Algunos elementos serán tarjetas de acceso, llaves, ordenadores...

En la versión actual, están implementados Laura y los guardias.

Interacción entre actores

Laura podrá interactuar con diferentes elementos del juego. Los objetos inanimados con los que se pueda interactuar, se subrayan al acercarse a ellos.

Algunas de las interacciones entre actores son:

- Guardias: Si descubren a Laura, intentarán atraparla. Si es capturada, la misión se considerará un fracaso.
- Cámaras: Similar a los guardias, pero solo alertarán a estos sobre la última posición conocida de Laura.
- Civiles: Ignorarán a Laura, a no ser que esta realice alguna acción fuera de lugar cerca de ellos. Si este es el caso, entrarán en pánico, alertando a los guardias de alrededor.
- Empleados: Tienen un comportamiento similar a los civiles. Reconocerán a Laura, habrá que evitarlos, ya que si se acercan demasiado nos descubrirán.

Objetivos planteados al Jugador

El objetivo principal dependerá de la misión que Laura haya aceptado, como por ejemplo, obtener un objeto, llegar a cierto lugar, o encontrar a cierta persona.

Entre el vestíbulo y nuestro objetivo se encuentran diferentes objetivos secundarios (desafíos/puzzles). Como pueden ser: deberemos evitar ser descubiertos, encontrar llaves y claves de acceso…

En este primer nivel, debemos acceder a todos los ordenadores, para poder acceder al Viejo Almacén.

Concept Art

El juego tiene un sistema de cámara isométrico similar al visto en Diablo II (2000) o XCOM: Enemy Unknown (2012). Con poca UI, ya que la mayoría de la información estará presentada en forma de 2D sprites sobre el mundo 3D. Como ejemplo, tenemos el juego The Marvelous Miss Take:

Como referencia para el diseño de Laura tenemos a Carmen Sandiego.

Cuando Laura usa sus poderes, el mundo a su alrededor pasa a una escala de grises. Si para el tiempo, no se ve afectada por lo que no pierde el color.

El siguiente collage está formado por referencias para el diseño de los NPCs.

Capítulo 4: Diseño técnico

Requisitos técnicos del entorno de desarrollo

El proyecto se ha desarrollado en Unity 2019.3 (Versión beta), por lo que los requerimientos se encuentran disponibles son:

Minimum requirements	Windows	macOS	Linux (Support in Preview)
Operating system version	Windows 7 and Windows 10, 64-bit versions only.	Sierra 10.12.6+	Ubuntu 16.04, Ubuntu 18.04, and CentOS 7
CPU	X64 architecture with SSE2 instruction set support	X64 architecture with SSE2 instruction set support	X64 architecture with SSE2 instruction set support
Graphics API	DX10, DX11, and DX12-capable GPUs	Metal-capable Intel and AMD GPUs	OpenGL 3.2+/ Vulkan-capable, Nvidia and AMD GPUs.

Additional requirements	Hardware vendor officially supported drivers	Apple officially supported drivers	Gnome desktop environment running on top of X11 windowing system, Nvidia official proprietary graphics driver or AMD Mesa graphics driver. Other configuration and user environment as provided stock with the supported distribution (Kernel, Compositor, etc.)
	For all operating systems, the Unity Editor is supported on workstations or laptop form factors, running without emulation or compatibility layer.		

Herramientas empleadas

Todas las herramientas utilizadas son gratuitas, la mayoría también son ofrecidas por el propio Unity.

Movimiento

Unity ofrece muchas formas de simular movimiento. En el caso del avatar del jugador (Laura), se ha utilizado un Character Controller, un componente de Unity. Además del movimiento básico, se ha implementado una mecánica de dash.

Para el movimiento de los NPC, la herramienta que más se amolda a este proyecto es un NavMesh Agent. Un Navigation Mesh es una superficie sobre la que el Agent puede

moverse. Si definimos correctamente esta NavMesh, el agente es capaz de moverse hasta donde deseemos, usando el camino más rápido.

Rutinas y acciones NPC

Tomando como inspiración la saga Hitman, los NPCs estarán ocupados con sus rutinas diarias. Estas rutinas son circuitos cerrados en bucle, sistema basado en una serie de conferencias realizadas por Unity **[3]**. El sistema permite añadir Waypoints, destinos, que el NPC debe alcanzar. Una vez llega, realiza las acciones de ese Waypoint, e.g. Mirar en una dirección, y hacer guardia. Cuando ha terminado con el Waypoint, continúa con el siguiente. Los Waypoints pueden ser utilizados para crear rutas de patrulla, ya que no requieren de acciones si no se desea hacer nada en ellos.

Para facilitar el diseño de nivel, se han añadido Gizmos a las rutinas. Estos objetos sólo son visibles cuando se selecciona una rutina, y ayudan a visualizar el camino que realizará cada NPC. Además, el sistema de acciones de los Waypoint está decoupled, por lo que los Waypoints pueden ser utilizados para realizar acciones independientes, sin necesidad de una rutina.

Sistema de sigilo

El sistema actual de detección del personaje consiste de dos partes. La primera es la visión de los NPC, representada por un cono de luz. Si un NPC puede ver a Laura, para su rutina y la observa durante un breve instante. Acto seguido, comienza a perseguirla, aumentando su velocidad conforme continúa la persecución. Si conseguimos darles esquinazo, investigarán el último lugar en el que nos hayan visto, y reanudarán a su rutina.

El segundo sistema de detección es por proximidad. Un collider rodea a Laura en todo momento. Mostrando un Game Over si se acerca demasiado a los NPCs, o estos a ella (tras perseguirla al ser detectada).

Control Temporal

Los objetos afectados por los poderes de control temporal, tienen un componente llamado Time Body. Este componente casero dictamina el comportamiento básico de un Time Body: mantener una lista de estados en el tiempo para poder regresar a ellos al rebobinar, además de reaccionar cuando el tiempo se pare o rebobine.

Existen varios componentes que derivan de Time Body, como NPC Time Body, o Player Time Body. Cada tipo diferente de Time Body controla el comportamiento y qué información debe ser considerada para que su Gameobject reaccione correctamente. E.g. Al parar el tiempo, los NPC deben quedar congelados, mientras Laura no se ve afectada.

Sistema de interacción del jugador con el entorno

En el estado actual, el sistema de interacción se basa en los Ordenadores y Puertas de seguridad. Las puertas de seguridad bloquean el acceso a ciertas zonas del nivel. Durante el

juego, Laura debe encontrar la manera de abrir estas puertas. En este primer nivel, las puertas están controladas por ordenadores, que deben de ser accedidos y hackeados.

Una vez hayamos hackeado todos los ordenadores del nivel actual, la puerta de seguridad queda abierta. Estas mecánicas son reactivas con respecto a los poderes temporales, por lo que no es posible acceder a los ordenadores si el tiempo está pausado. Además de dejar de estar hackeados si retrocedemos hasta un estado en el que no lo estaban.

Assets y recursos

Para trabajar en los assets, Unity ofrece ProGrid y ProBuilder. Estas herramientas, anteriormente de pago, fueron adquiridas por Unity e incorporadas en el programa base. ProGrid permite utilizar un sistema de guías (grids) para construir niveles o assets. Mientras que ProBuilder, permite crear formas geométricas *low-poly*, de menor o mayor complejidad, en el propio editor de Unity. Con estas herramientas, se ha prototipado durante el desarrollo.

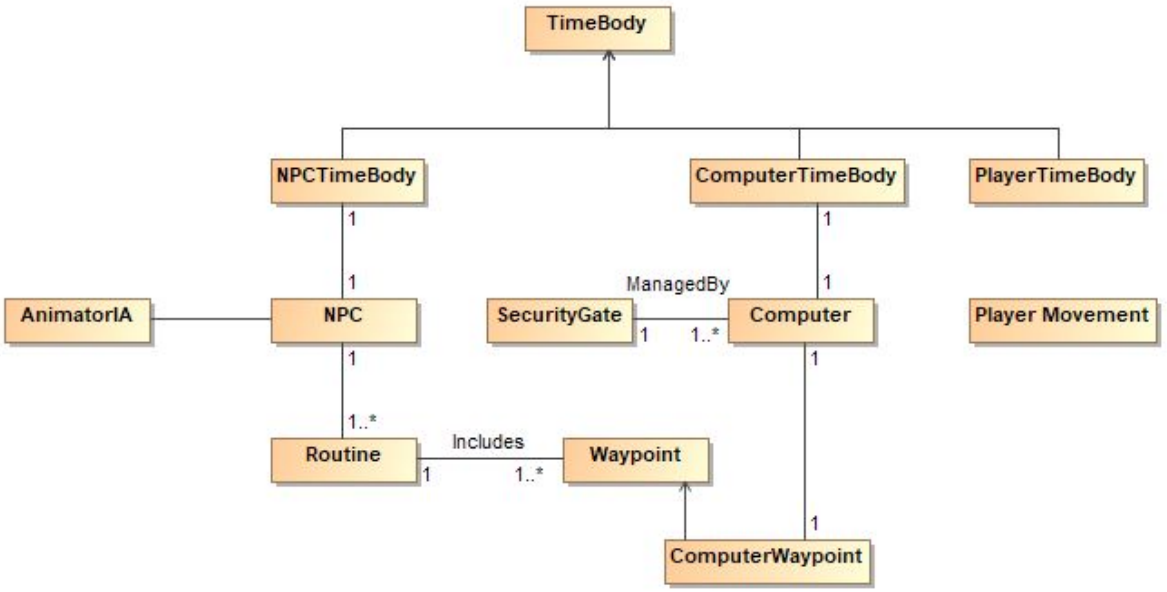
La escenografía del nivel disponible ha sido construida utilizando otra herramienta disponible en Unity, de más reciente lanzamiento. El sistema SNAPS, anunciado y lanzado hace un par de meses, permite usar formas básicas creadas con ProBuilder para crear el diseño inicial visual. Permite además reemplazar estas formas por modelos más complejos, por lo que se puede trabajar de forma más fluida.

Todos los links a los recursos utilizados en el juego se encuentran en el capítulo final.

Arquitectura del juego y componentes

La estructura diseñada para Back On Time, está compuesta por diversos componentes que interaccionan entre sí, intentando mantener un máximo de componentes por objeto que sea adecuado y manejable.

Diagrama de clases simplificado:



La clase Time Body es la que gestiona las mecánicas de los poderes temporales. Las clases derivadas de ella contienen una lista de "estados en el tiempo" que contienen la información relevante para cada objeto, a fin de poder volver a ese estado. Además, implementa la simulación de parar el tiempo para cada objeto, y PlayerTimeBody gestiona el *cooldown*.

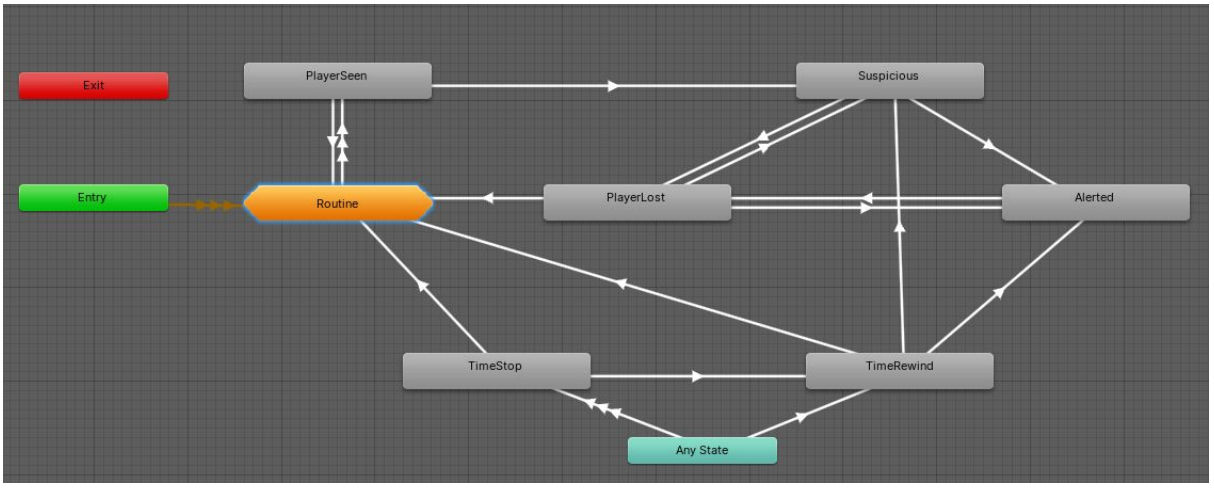
Los NPC pueden tener una o más rutinas asignadas a ellos, y cambiar entre rutinas en waypoints determinados, bajo circunstancias determinadas. El sistema de Waypoints incluye una lista de acciones que se realizan en orden.

La inteligencia del NPC va gestionada con el componente de Unity Animator, con un Animator Controller que contiene el mapa de IA.

Inteligencia Artificial

Las animaciones en Unity se gestionan con una herramienta llamada Animator. Animator ofrece diferentes estados para cada animación, y scripts específicos para cada estado. Esta herramienta puede ser utilizada como Finite State Machines (FSM), un sistema visual para gestionar los diferentes estados que tiene cada NPC de manera individual, pero unificada.

El sistema de sigilo y el sistema de rutinas son gestionados por este componente.:



Cada estado tiene scripts propios, así como condiciones que hacen posible navegar entre estados. El estado natural de los NPC es su Rutina; los NPC solo abandonan este estado si ven a Laura o esta utiliza sus poderes.

Capítulo 5: Diseño de niveles

El diseño del nivel es sencillo, pero ha sido creado con la idea de ofrecer un reto acorde al primer nivel de un juego de estas características. Laura está representada por la cápsula roja, mientras que las cápsulas azules son los guardias. Tal y como está planteado el nivel, es imposible pasar del vestíbulo sin parar el tiempo. El guardia de la entrada, no permite

entrar a Laura sin usar sus poderes. Este "muro" plantea el primer reto para el usuario, ya que requiere que paremos el tiempo para poder avanzar.

Si Laura es vista, será perseguida. Si los guardias se acercan demasiado a Laura (o viceversa), la misión fracasará. Podemos evitarlo parando el tiempo, o rebobinando antes de que nos capturen.

En algunas habitaciones, se encuentran ordenadores (representados con consolas) con los que se debe interaccionar durante varios segundos. Tan solo debemos acercarnos, presionar el botón de interacción, y permanecer cerca de la consola unos segundos. Cuando se han accionado los dos ordenadores de la primera zona, la puerta de seguridad central se abre.

Nuestro objetivo, es alcanzar el pasillo pasando la segunda puerta de seguridad. Por lo que en la segunda zona, tenemos tres ordenadores. Una vez más, tenemos que hackearlos antes de poder avanzar. En cada una de las habitaciones, se encuentra un ordenador. Uno debemos pasarlo parando el tiempo, mientras que los otros requieren estudiar las patrullas de los guardias y actuar cuando sea posible.

Capítulo 6: Entregables

Vídeo presentación

El vídeo muestra los aspectos principales del proyecto, que pueden revisarse en la escena "*LevelOne*". Las funcionalidades presentadas durante el video no incluyen el código, y muestran la escena con su configuración.

Vídeo presentación:
https://youtu.be/kydft5bab6g

Vídeo trailer

El trailer incluye un pequeño vídeo que muestra el gameplay del juego, sin revelar demasiado. Incluye un link para descargar el juego en la descripción.

Vídeo trailer:
https://youtu.be/t1JPu0kSZTk

Showcase

El **showcase** creado para Back On Time, se encuentra en:
https://connect.unity.com/p/back-on-time

Se aconseja descargar el juego, ya que el showcase no parece funcionar correctamente. Se puede acceder al juego para probarlo utilizando el siguiente link:
https://drive.google.com/open?id=1Msp96UfbimBcY3LQ1j3flRaA4Ns9zSES

Repositorio Git

Para poder descargar el proyecto, necesitamos una versión de Unity compatible. La **versión beta** actual, disponible aquí:
https://unity3d.com/unity/beta/2019.3

El **repositorio** con la versión final del proyecto se encuentra en:
https://github.com/Optimixto/ProjectBackOnTime

Capítulo 7: Manual de usuario

Requisitos mínimos

Los requerimientos mínimos del sistema (extraídos del manual de requisitos del sistema de Unity **[4]**):

System		Minimum requirements
Desktop		
	Operating system	Windows 7 SP1+ macOS 10.12+ Ubuntu 16.04+
	CPU	SSE2 instruction set support.
	GPU	Graphics card with DX10 (shader model 4.0) capabilities.
iOS		iOS 9.0 or higher.
Universal Windows Platform		Windows 10 and a graphics card with DX10 (shader model 4.0) capabilities.

Manual del usuario

Basado en la conceptualización

En Back On Time, controlaremos a Laura, una espía a sueldo con poderes sobrenaturales. Durante varios años ha estado trabajando para clientes de mayor o menor influencia, mientras aprendía a controlar mejor sus habilidades. Hace poco, Laura fue contactada por una persona desconocida, que ha conseguido su contacto a través de un aliado mutuo. La oferta de trabajo promete una cantidad de dinero muy elevada por una tarea bastante sencilla: recuperar un archivo en el antiguo ayuntamiento, el cual fue adquirido por una empresa privada hace varios años. Aunque tiene un mal presentimiento, la compensación económica es demasiado suculenta como para dejar pasar la oportunidad.

Laura es capaz de controlar el tiempo, aunque de forma limita. Si la situación lo requiere, Laura puede parar el tiempo, evitando así obstáculos que son imposibles de sortear para cualquier otro espía. Si se encuentra atrapada, o el fracaso es inminente, Laura puede volver atrás en el tiempo, regresando así a un punto seguro. Esta red de seguridad, permite a Laura aprovechar su conocimiento sobre el mundo para explorar zonas peligrosas, con cierto margen de error.

Back on Time puede ser jugado con mando o teclado. Los controles están disponibles en el menú de pausa.

Gracias por jugar Back On Time.

Capítulo 8: Conclusiones

Introducción a la conclusión

El proyecto Back On Time es el primer proyecto en Unity que he realizado. Durante el desarrollo de este juego, he aprendido a utilizar muchas de las herramientas de Unity: NavMesh, Animator, Editor… Mi experiencia previa con Unity fue escueta y en 2013, cuando Unity aún tenía dos versiones, la completa siendo de pago. Este verano (verano del 2019), tras descubrir la posibilidad de realizar el TFG en este área de estudio, utilicé las vacaciones de verano para familiarizarme con el software. Este proyecto ha sido un proyecto de aprendizaje, un proceso de *trial and error*, que ha concluido con un producto acorde a la experiencia adquirida.

Antes de trabajar en Back On Time, tampoco había programado en ningún lenguaje C. Aprender este lenguaje no supuso demasiado problema, pero sí supuso una carga de trabajo extra. Durante el proyecto, he aprendido sobre el proceso de planificación, gestión, y desarrollo de un videojuego.

Otros contratiempos en el camino que contribuyeron a este aprendizaje fueron:

- Git: Tuve problemas con Git y Source Tree, al no haber trabajado previamente con estas herramientas.
- Versión de Unity: Al intentar implementar el nuevo sistema de Input disponible en Unity, el proyecto se desarrolló en una versión Beta. Esto generó problemas a la hora de compartir el proyecto.
- Prefabs: Algunos de los componentes de los Prefabs creados (específicamente Action Collection) tienen problemas a la hora de incluirse en Prefabs, ya que son *Scriptable Objects*.

Retrospectiva de los logros y objetivos

Los objetivos planteados al inicio del proyecto hacían énfasis en el aprendizaje y en crear un *Minimum Viable Product* (MVP) funcional con la intención de añadirlo en un portfolio de desarrollador. En términos de los objetivos iniciales, valoro que han sido alcanzados con un nivel de éxito adecuado. Esto no quiere decir que el producto sea un producto final y definitivo, pero sí es un proyecto primerizo que sirve como primer contacto con una tecnología.

El desarrollo de este proyecto ha requerido de una investigación exhaustiva. Afortunadamente, hoy en día tenemos una gran variedad de recursos a nuestro alcance. Sin embargo, es común que esta información no esté centralizada. Aunque Unity Learn (https://unity.com/learn) ofrece una gran variedad de recursos, no todos son útiles. Algunos están obsoletos, otros no cubren el tema correctamente/con suficiente profundidad… por lo que el objetivo de familiarizarnos con Unity, requirió de más esfuerzo del esperado.

A pesar de los contratiempos, la complejidad del proyecto, y la limitada experiencia previa, los objetivos han sido cumplidos y el MVP es un producto con un mínimo de complejidad que permite mostrar los conocimientos obtenidos.

Seguimiento de la planificación y metodología

La planificación inicial del proyecto ha sido seguida exitósamente a grandes rasgos. Durante el desarrollo, la planificación ha tenido que ser adaptada para poder garantizar ciertos aspectos esenciales para el proyecto. Esto ha supuesto "sacrificar" los recursos de las fases tardías del proyecto, ya que requerían de un mayor presupuesto (en términos de horas de trabajo).

Los cambios han sido explicados en detalle anteriormente. A modo de resumen:

- La implementación de las mecánicas básicas del juego tuvieron que ser extendidas a la PEC 3, ya que requerían más trabajo tras su investigación.
- Muchos aspectos "secundarios" del juego, no tenían un horario previsto en la planificación inicial del proyecto. Por lo que parte de los recursos fueron desviados a este problema.

- La planificación inicial tenía suficiente holgura como para contener problemas e imprevistos. Esta previsión ha hecho que sea posible mantener un ritmo y un seguimiento adecuados, además de ayudar a visualizar qué aspectos del proyecto eran esenciales.

Futuro del producto

Estudio del estado actual

En la versión actual de Back On Time, algunas de las funcionalidades que se han descrito en este documento no han sido implementadas. La *sensación del juego* es aproximada; sin embargo, carece de la complejidad que ofrecería un producto acabado.

Esta versión es un prototipo que nos permite hacernos una idea de qué esperamos del juego. Las funcionalidades actuales son las básicas, que nos facilitan una idea que ofrecer para evaluar si se considera rentable invertir recursos en desarrollarla. Este sistema, el prototipado de juegos, es utilizado por muchas empresas. Algunas empresas, y otros grupos relacionados con el mercado, hospedan Game Jams. Estos Game Jams son eventos en los que los participantes diseñan e implementan pequeños videojuegos. Muchos juegos comienzan de esta forma, como por ejemplo "Baba is You" o "Untitled Goose Game", y acaban teniendo un gran éxito.

El producto actual requiere de más trabajo para poder considerarse un producto completo. Durante el Trabajo, he mantenido un listado de tareas esenciales, y otro con tareas ideales a realizar. Para poder implementar todo lo deseado, se necesitaría un mayor equipo, con una mayor variedad de talentos. Sin embargo, muchos aspectos de la versión actual, podrían mantenerse, o simplemente adaptarse.

Posible implementaciones futuras

Ciertas implementaciones que ayudarían a crear una experiencia más completa serían:

- NPC
  - Refinar el IA
  - Añadir mayor complejidad de acciones
  - Añadir diferentes tipos de NPC (civiles, empleados, cámaras…)
- Interacción Jugador-Entorno
  - Ampliar el repertorio de objetos interactivos
  - Recalcar los objetos interactivos
- *Cutscenes*
  - Añadir *Cutscenes* que permitan desarrollar la historia
- Animaciones
  - Añadir animaciones a objetos/HUD/personajes
- Sonido
  - Mejorar el diseño del sonido
  - Añadir sonidos a diferentes acciones (tanto de NPC como de jugador)
- Diseño gráfico

- Crear assets para obtener un ambiente cohesivo e inspirado en la conceptualización

Con estos cambios, podríamos considerar el juego una beta, e incluso considerar publicarlo en alguna plataforma como Early Access. Una vez realizadas estas implementaciones, el proyecto ofrece un producto sencillo, pero ampliable, que cubre los requisitos mínimos para un juego publicable.

Estos requisitos varían dependiendo de la plataforma de distribución que queramos utilizar. Hasta hace relativamente poco, Steam Green Lights era una de las pocas posibilidades para un juego indie (especialmente solo-developed) de entrar en el mercado. Una de las ventajas de la ruptura del monopolio de Steam es la posibilidad de publicar juegos en otras plataformas (Epic, GOG…).

Capítulo 9: Glosario, licencias, y misc.

Glosario

***Cooldown**:* Tiempo de espera necesario que requiere una acción antes de poder ser llevada a cabo de nuevo.

***Cutscenes**:* Escenas, similares a una película, en la que el usuario tiene control limitado sobre lo que ocurre.

Game loop: Término utilizado para referirse al ciclo que siguen algunos juegos. E.g. Recojo madera, construyo objetos, vendo objetos, consigo dinero, invierto en mi carpintería.

Low-poly: Una figura geométrica con un menor número de polígonos (que componen los objetos 3D de un juego), utilizados principalmente para simbolizar objetos más complejos.

Minimum Viable Product: El producto mínimo requerido para que se pueda considerar que todas las características básicas han sido implementadas.

NPC (Non-Playable Character): Personaje que el jugador no puede controlar directamente.

Save scumming: Técnica utilizada en el mundo de los videojuegos que consiste en recargar una partida guardada para volver a un estado anterior más favorable.

Sensación del juego (Game feel): Sensación intangible que experimenta el jugador cuando interacciona con un videojuego.

Scriptable Objects: Herramienta de Unity, similar a un Gameobject, pero sin tener representación física en la escena.

Trial and error: Método para resolver problemas que se basa en probar diferentes soluciones hasta que funcione o se desista.

Recursos utilizados

La mayoría de los recursos utilizados provienen de Unity, ya que se ha intentado mantener el proyecto lo más rentable posible, además de crear la gran mayoría de los scripts que gestionan la lógica del juego. Algunos assets de pago disponibles podrían haber facilitado el desarrollo, como es el caso de la herramienta de control temporal disponible en el Asset Store de Unity **Chronos** (https://assetstore.unity.com/packages/tools/particles-effects/chronos-31225).

Listado de recursos

2D:
Icono del juego:
https://www.iconfinder.com/icons/280827/clock_countdown_design_flow_game_gaming_hourglass_play_time_timer_wait_icon
Ordenador sin hackear: https://www.flaticon.com/free-icon/ban_2049959
Ordenador hackeando:
https://www.flaticon.com/free-icon/computer_869378?term=hacking&page=1&position=55
Ordenador hackeado: https://www.flaticon.com/free-icon/checkmark_1828710
3D:
SNAPS Sci-fi - prototipo:
https://assetstore.unity.com/packages/3d/environments/sci-fi/snaps-prototype-sci-fi-industrial-136759
SNAPS Office - prototipo:
https://assetstore.unity.com/packages/3d/environments/snaps-prototype-office-137490
Assets del Assets Store:
Quick Outline (free version):
https://assetstore.unity.com/packages/tools/particles-effects/quick-outline-115488

Licencias y copyright

ejemplares mediante alquiler y préstamo, sin la autorización escrita del autor o de los límites que autorice la Ley de Propiedad Intelectual.

Recursos de aprendizaje

Muchos de los recursos utilizados están obsoletos, pero siempre se puede aprender algo de ellos. Se han consultado muchas páginas web diferentes para encontrar soluciones. De entre estas cabe destacar:

- https://unity.com/learn
- https://stackoverflow.com/

Estos son los recursos más significativos utilizados para implementar el juego:

- http://www.gameprogrammingpatterns.com/contents.html - Contiene patrones específicos para el desarrollo de videojuegos.
- https://www.youtube.com/watch?v=raQ3iHhE_Kk - Vídeo de una conferencia que introduce el concepto de Scriptable Objects.
- https://medium.com/ironequal/unity-character-controller-vs-rigidbody-a1e243591483 - Blog que muestra las diferencias entre las herramientas Character Controller y Rigidbody, para desplazar un personaje.
- https://ornithoptergames.com/reactiverx-in-unity3d-part-1/ - Una de las páginas de investigación del UniRx y el lenguaje reactivo.
- https://gamedevelopment.tutsplus.com/articles/how-to-build-a-prince-of-persia-style-time-rewind-system-part-2--cms-26363 - Blog que presenta una versión obsoleta de la mecánica del control temporal.
- https://aiandgames.com/unity-pathfinding-part-3/ - Introducción al *pathfinding* en Unity

Bibliografía

[1] 2018 Global Games Market Values:
https://newzoo.com/insights/articles/global-games-market-reaches-137-9-billion-in-2018-mobile-games-take-half/ - 01/01/2020
[2] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides; **Design Patterns: Elements of reusable Object-Oriented Software**; 12ª edición, Editorial Pearson.
[3] Adventure game tutorial - Unity
https://learn.unity.com/project/adventure-game-tutorial - 03/01/2020
[4] Manual Unity - System Requirements:
https://docs.unity3d.com/Manual/system-requirements.html - 02/02/2020

Anexos

Requisitos del sistema:
https://docs.unity3d.com/2019.3/Documentation/Manual/system-requirements.html#editor