Escuela de Doctorado
y Estudios de Posgrado
Universidad de La Laguna

MÁSTER UNIVERSITARIO EN DESARROLLO DE VIDEOJUEGOS

Trabajo Fin de Máster

Proyecto de Mascota Virtual en Unity

Virtual Pet Project in Unity

Sergio González Guerra

D. **Rafael Arnay del Arco**, con N.I.F. 78.569.591-G profesor Contratado Doctor adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor.

C E R T I F I C A (N)

Que la presente memoria titulada:

“Proyecto de Mascota Virtual en Unity”

ha sido realizada bajo su dirección por D. Sergio González Guerra, con N.I.F. 79093820-V.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 13 de Mayo 2023.

Agradecimientos

Quiero agradecer a todo el profesorado Máster en desarrollo de videojuegos por su ayuda en mi formación durante la carrera.

Y a toda mi familia en especial a mis padres por todo su apoyo.

Resumen

El objetivo de este proyecto ha sido el desarrollo de un videojuego sobre el cuidado de una mascota virtual, usando el motor de videojuegos de Unity para móviles (Android).

El juego trata sobre el cuidado de una mascota virtual la cual debemos de responsabilizarnos de alimentar, lavar y jugar con ella para que se entretenga.

Todo esto sucede a tiempo real, en un determinado tiempo la mascota empezará a tener hambre, a ensuciarse o a aburrirse.

Palabras clave: Unity, C#, Mascota Virtual, Android, Pixelart.

Abstract

The objective of this project was the development of a videogame about the care of a virtual pet, using the Unity video game engine for mobile phones (Android).

The game is about taking care of a virtual pet that we must be responsible to feed, wash and play with so that it is entertained.

All this happens in real time, at a certain time the pet will start to get hungry, dirty or bored.

Keywords: Unity, C#, Virtual Pet, Android, Pixelart.

Índice general

Índice de figuras

Capítulo 1
Introducción

En esta sección explicaré los orígenes de las mascotas virtuales, los objetivos planteados para este proyecto y algunas mascotas virtuales que podemos encontrarnos a día de hoy en el mercado y las cuales han servido de inspiración en este proyecto.

1.1. Antecedentes

La primera mascota virtual [1] creada por Aki Maita y comercializada por Bandai en 1996 en Japón fue Tamagotchi [2] y es también la inspiración principal de este proyecto.

Figura 1.1: Tamagotchi Original

Tamagotchi tuvo un impacto enorme de popularidad, llegando a vender 82 millones de unidades.

Al igual que una mascota real, los tamagotchis tenían sus necesidades, desde su alimentación, entretenimiento con juegos simples, limpieza, cuidarlo cuando enferma, entre otras.

Según pasa el tiempo la mascota va creciendo y evoluciona dependiendo de lo bien que lo hayamos cuidado, en caso de no atenderla por mucho tiempo o por vejez, esta podría llegar a morir.

1.2. Objetivo General

El objetivo principal de este proyecto es el del desarrollo de un videojuego sobre el cuidado de una mascota virtual de la que tendremos que atender sus necesidades.

Se ha intentado abarcar las características más básicas desde un sistema de guardado, un control del tiempo real, una forma de alimentar, bañar y entretener a la mascota, una tienda para comprar comida y un sistema de evolución de la mascota.

1.3. Objetivos Específicos

Concretamente, los objetivos marcados para este proyecto son:

1. La mascota debe contener un nivel de amistad el cual va subiendo si tocamos la pantalla y la alimentamos. A un cierto nivel, la mascota evoluciona seleccionando al azar su siguiente forma.
2. Un inventario donde guardaremos diferentes tipos de comida para después alimentar a la mascota.
3. Una tienda para comprar diferentes tipos de comida con el dinero que obtengamos.
4. Minijuegos para entretener a la mascota y a la vez conseguir dinero.
5. Un baño donde se puede lavar a nuestra mascota.
6. Un sistema de guardado para cargar el estado actual de la mascota.
7. Y un sistema que controle el tiempo real que pasa desde la última vez que entramos al juego.

1.4. Estado Actual del Tema

Actualmente existen diferentes tipos de mascotas virtuales los cuales han servido de inspiración para este proyecto:

- **Tamagotchi pix** [3] **:** Existen más de 40 versiones de tamagotchis con diferentes características cada uno. La última versión que ha sacado Bandai es Tamagotchi pix, el cual cuenta con una cámara para hacer fotos y jugar con tu mascota.

Figura 1.2: Tamagotchi Pix

- **Kotodama Diary** [4] **:** Esta mascota virtual se centra más en la colección de diferentes mascotas virtuales, mezclando palabras con conceptos que la mascota irá aprendiendo para decidir la forma que va a adoptar al evolucionar.

Figura 1.3: Kotodama Diary

- **My Tamagotchi Forever** [5] **:** Una versión de tamagotchi para móviles, con los aspectos básicos del original pero en un entorno 3D.

Figura 1.4: My Tamagotchi Forever

Capítulo 2
Documento de Diseño de Videojuegos

A continuación se mostrará un documento del diseño

2.1. Concepto

Género: Simulador de mascota virtual.

Plataforma: Android.

Sinopsis de jugabilidad: Cuidaremos de una mascota virtual atendiendo a sus necesidades como su alimentación, higiene y entretenimiento.

Mecánica:

- **Inventario:** El jugador tendrá un inventario para almacenar comida de diferentes tipos para seleccionar la que quiera dar a la mascota.
- **Evolución:** La mascota tendrá distintas etapas evolutivas a medida que sube su medidor de amistad y evolucionara en determinados niveles. Empezará por el estado “Huevo” seguido del estado “Bebé”, “Niño”, “Adolescente” y “Adulto”.
- **Higiene:** Cuando la mascota esté sucia, tendremos que limpiarla usando una pastilla de jabón y después una ducha.
- **Minijuegos:** Cuando la mascota se aburra, podremos jugar a diferentes minijuegos para entretenerla.
- **Tienda:** En los minijuegos o subiendo a la mascota de nivel obtendremos dinero para comprar comida a la mascota.
- **Sistema de guardado:** Después de cada cambio en el estado de la mascota, se guardará la partida en un archivo .json.
- **Control real del tiempo:** El juego mirará la hora real, según el tiempo que pase, bajarán las estadísticas de hambre, higiene y entretenimiento de la mascota.

Tecnología: Unity versión: 2022.1.17f1 , C#, Editor de imágenes (Pixelart): Aseprite [6].

Público: Es un videojuego casual y está dirigido a todos los públicos.

2.2. Mecánica del juego

Cámara: El juego usará una cámara en 2D.

Periféricos y Controles: El juego usará el control táctil para realizar distintas actividades como la alimentación (arrastrando la comida hacia la boca de la mascota).

Puntaje: A medida que acariciemos a la mascota (tocando la pantalla) subiremos su medidor de amistad, cuando este llegue al máximo, la mascota subirá de nivel recompensando al jugador con monedas para comprar más comida en la tienda.

También obtendremos más monedas jugando a los minijuegos.

Guardar/Cargar: La partida se guardará automáticamente cada vez que cambie el estado de la mascota. Es importante destacar que también se guardará en cada momento la hora real del dispositivo de modo que al volver a iniciar la sesión en el juego se calculará el tiempo transcurrido y se establecerán los valores de hambre, higiene y entretenimiento a los que correspondan.

2.3. Estados del juego

Habitación principal de la mascota: El juego empieza en una habitación con un huevo de la cual nacerá la mascota, tendremos que acariciarla tocando la pantalla para que nazca.
Una vez aparezca la mascota se desbloquearán los cuidados básicos de alimentación, higiene y entretenimiento.

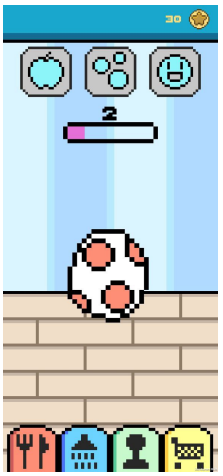

Figura 2.1: Habitación principal de la mascota al comenzar el juego

Menú de inventario: Tendremos diversas opciones de comida disponibles para alimentar a nuestra mascota, para ello accedemos a nuestro inventario de comida y seleccionamos la que le queramos dar. Después aparecerá un plato con la comida seleccionada para alimentar a la mascota.

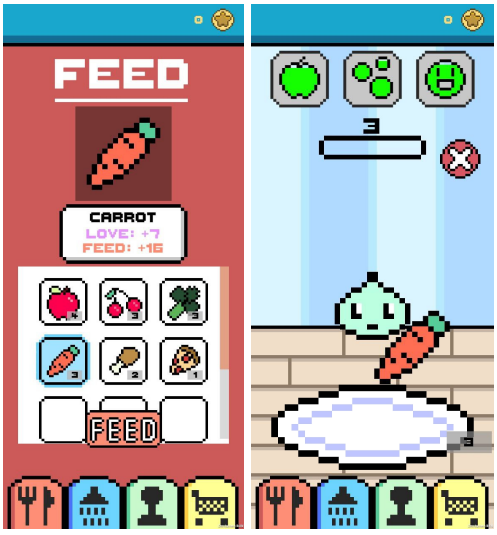

Figura 2.2: Menú de inventario y alimentación de la mascota

Cuarto de baño: Para poder limpiar a la mascota cuando esté sucia, tendremos que acceder al baño, primero tendremos que limpiarla con una pastilla de jabón. Después lo aclararemos con la ducha.

Figura 2.3: Cuarto de baño para lavar a la mascota

Minijuegos: Para entretener a la mascota, podremos acceder a diferentes minijuegos. “Fall Game” es el disponible en la demo de este proyecto, la mascota se lanzará al aire con un paracaídas y tendrá que esquivar los obstáculos que aparezcan, también podrá recoger las monedas que vea.

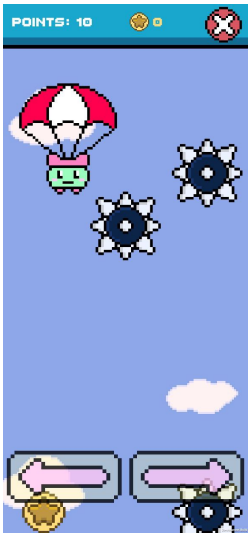

Figura 2.4: Minijuego “Fall Game”

Tienda: Con las monedas que ganemos, podremos comprar diferentes tipos de alimentos para nuestra mascota, en la parte superior veremos el precio, la cantidad de amistad y alimentación que nos proporciona la comida que seleccionemos.

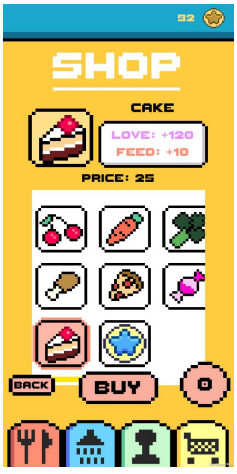

Figura 2.5: Menú de la tienda de alimentos

2.4. Progreso del juego

El objetivo principal del juego no es solo el cuidado de la mascota, sino de ver su evolución con el paso del tiempo. A medida que sube su nivel de amistad, en ciertos niveles cambiará de un estado evolutivo a otro.

Actualmente cuando la mascota llegue al nivel 3, pasará del estado “Huevo” al estado “Bebé”, al nivel 5 pasará al estado “Niño”, al 8 pasa al estado “Adolescente” y al 10 al estado “Adulto”. La intención es que en la transición de cada estado puedan llegar a pasar horas, pero por motivos de testeo, este se ha reducido en cuestión de segundos. En el juego final, el ciclo de vida de la mascota se cumplirá al cabo de una semana.

Aunque no se contempla en este proyecto, la idea sería crear una colección de todas las mascotas que el jugador puede encontrarse y repetir el ciclo de vida de varias mascotas hasta conseguir verlas todas.

Se han añadido un total de 12 mascotas en este proyecto:

Figura 2.6: Mascotas añadidas al proyecto y sus etapas evolutivas

Capítulo 3
Fases y desarrollo del proyecto

En este capítulo hablaremos del funcionamiento y del desarrollo de los objetivos de este proyecto.

3.1. Evolución de las mascotas

El elemento principal de este proyecto son las mascotas y su evolución.

El primer aspecto que se desarrolló fue la manera de evolucionar a la mascota, por ello se añadió una barra de progreso que determina el nivel de amistad de la mascota.

Figura 3.1: Barra de amistad y nivel de amistad con la mascota

Para aumentar la barra de amistad el jugador tendrá que pulsar la pantalla, esto se lleva a cabo con la clase “Touch” [7] de Unity que nos dejará saber la posición de los dedos en la pantalla y con “Input.touchCount” la cantidad que se está pulsando.

En el juego se establece un máximo de 2 dedos, recogeremos la posición de cada dedo en la pantalla y usaremos “Camera.main.ScreenToWorldPoint()” [8] para convertir la posición que nos da la pantalla respecto al mundo del juego.

Esto lo necesitamos ya que cuando el jugador pulsa la pantalla además de aumentar la barra de amistad, aparecerá un efecto de un corazón y un sonido.

En ciertos niveles de amistad, la mascota empezará a evolucionar, tendrá que esperar un tiempo y cambiará de estado.

Para esto se ha usado Scriptable Objects [9] que nos permite crear objetos reutilizables en el editor de Unity. En este caso se han usado para determinar ciertos datos de las mascotas y poder reutilizarlas en otras zonas del juego como los minijuegos.

Para este caso se guardan variables relacionadas con características únicas de la mascota como su Nombre, Estado, un Animator y una lista de las posibles evoluciones que puede tener. También proporciona de forma aleatoria la posible evolución de la mascota.

Con esto desde la interfaz de Unity, podremos crear un objeto de este tipo y rellenarlo con los datos correspondientes.

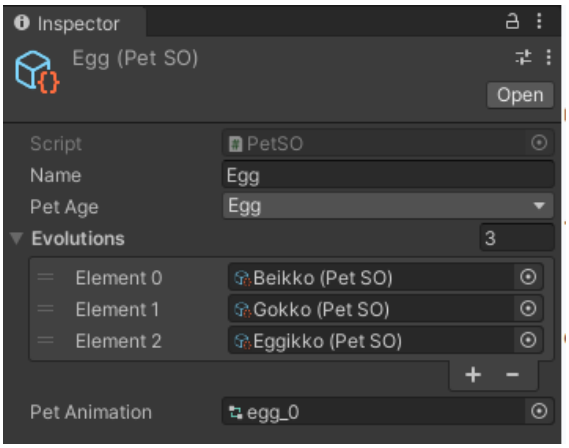

Figura 3.2: Ejemplo de Scriptable Object del huevo, podemos ver una lista con tres posibles evoluciones

3.2. Control real del tiempo

Para este proyecto he necesitado controlar exactamente el tiempo que pasa cuando el usuario sale de la aplicación y vuelve a entrar.

Para ello se utiliza principalmente DateTime.Now.Ticks [10] dentro de un Update() para guardar en cada momento la fecha exacta en la que se encuentra el jugador en ese momento.

Este dato lo guardamos utilizando PlayerPrefs [11]. Cuando el usuario regresa al juego, se carga la variable guardada y se compara con la fecha actual en la que se ha iniciado el juego para saber cuánto tiempo ha pasado.

Con esta información y mirando un determinado umbral (por ejemplo cada minuto que pase bajará en 1 la estadística del hambre) sabremos la cantidad de hambre, higiene o entretenimiento tiene en ese momento la mascota. También se actualiza el tiempo de espera necesario en caso de que la mascota esté en proceso de evolución.

Figura 3.3: Tiempo de espera hasta que la mascota evolucione

Mientras el jugador se encuentre en el juego, se usará Time.deltaTime [12] para saber el tiempo transcurrido durante la ejecución.

3.3. Inventario y alimentación de la mascota

Para alimentar a nuestra mascota, primero debemos acceder al inventario y ver que alimentos le podemos ofrecer.

El desarrollo del inventario se ha realizado en dos partes, primero la parte visual donde el usuario ve un grid con iconos representando los diferentes alimentos que tiene y una segunda parte que contiene todos los datos del inventario y de los alimentos.

3.3.1. Control visual del inventario

En la parte visual se ha combinado el Scroll Rect [13] de Unity y para ordenar el contenido se ha usado un Grid Layout Group [14].

Primero tendremos un script (UI_InventoryItem.cs) por cada “casilla” del inventario que determinará el icono que se representa del objeto y la cantidad que tenemos del mismo. En los huecos vacíos se desactiva el icono que representa el alimento de dicha casilla.

Figura 3.4: Grid del inventario con algunos alimentos, también muestra las cantidades de cada uno

Para detectar cuando el usuario selecciona una casilla, usamos el evento "OnPointerClick" [15] que activará un evento para saber qué alimento se está seleccionando y muestre al usuario el nombre y las estadísticas del mismo. Si la casilla no está vacía se mostrará además un botón para confirmar la selección del alimento que se quiere dar a la mascota.

Luego tendremos otro script general (UI_InventoryPage.cs) que controla toda la interfaz visual del inventario, posee una lista con todas las casillas de inventario representado en el grid y es el que se encarga de manejar el evento cuando el usuario selecciona un alimento para mostrar los stats por pantalla.

Figura 3.5: Panel con las estadísticas del alimento. Se puede ver el icono, el nombre, la cantidad de amor y alimentación que da a la mascota

También se encarga de instanciar todas las casillas dentro del Grid Layout Group y de actualizar sus datos.

3.3.2. Control de datos del inventario

Al igual que con las mascotas, los alimentos funcionan con Scriptable Object.

Por un lado, para cada alimento se guardan variables como el nombre, el número máximo de objetos que podemos llevar, la cantidad de amistad y de alimentación que otorga a la mascota, el precio y el sprite con el icono que lo representa.
También contiene un ID que identifica todos los alimentos del juego.

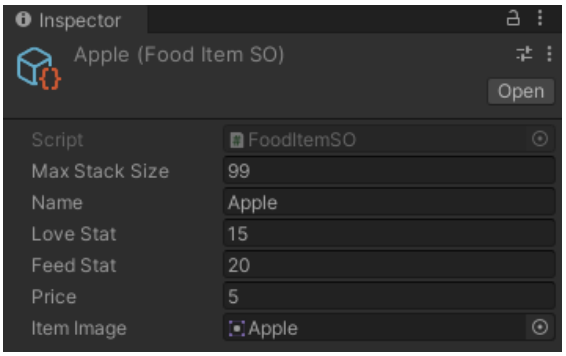

Figura 3.6: Ejemplo de Scriptable Object de una manzana

Por otro lado tenemos un Scriptable Object que representa los datos del inventario, contiene una lista con los alimentos que contienen, un tamaño máximo de objetos diferentes que se pueden añadir y varios métodos para añadir objetos, saber si el inventario está lleno o eliminar un objeto del inventario.

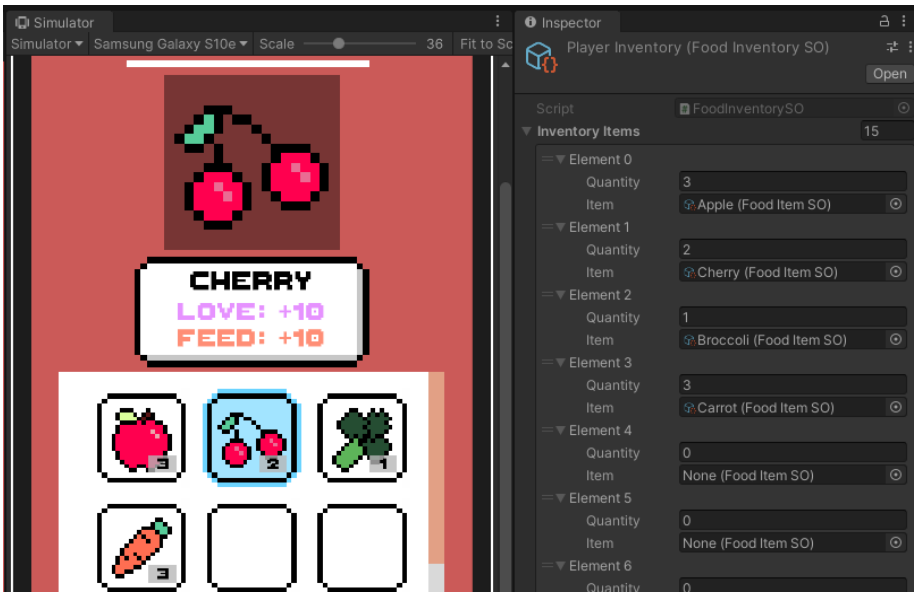

Figura 3.7: Ejemplo de Scriptable Object del inventario del jugador, podemos observar una lista con los alimentos y sus cantidades

Al añadir un alimento, el comportamiento es el siguiente:

- Si dicho objeto ya existe en el inventario, se suma la cantidad correspondiente. (Si se pasa, no se añade la cantidad).
- Si el objeto no existe, se busca un hueco vacío y se añade la cantidad a esta.
- Si el inventario está lleno, se impide añadir el alimento.

Al eliminar un objeto del inventario se comprueba que la cantidad a quitar no supere a la cantidad actual de dicho objeto.

3.3.3. Inventory Controller

Es el encargado de conectar la parte visual de la parte de los datos del inventario y los inicializa.

Cuando el usuario selecciona un alimento y pulsa el botón de alimentar, el Inventory Controller se encarga de pasar la información del Scriptable Object a la sección de alimentación de la mascota y automáticamente cierra la pestaña del inventario.

3.3.4. Alimentación de la mascota.

Una vez el jugador decide qué alimento ofrecer a la mascota, aparecerá un plato con la imagen de dicho alimento. El usuario tendrá que arrastrar con el dedo el alimento hasta la mascota y este recibe sus estadísticas de amistad y alimentación.

Para mover el alimento por la pantalla se creó el script “DragObject” que usa Physics.Raycast [16] y la posición del dedo con Touch, para saber si el dedo se encuentra en el área del collider del alimento.

Usando “TouchPhase”, podemos detectar cuando el usuario mueve el dedo por la pantalla, en cuyo caso se crea un vector indicando la posición a la que se tiene que mover el objeto respecto a la posición del dedo.

Existe una variable boolean que indica cuando el objeto está siendo arrastrado, de esta forma podemos saber si cuando el usuario suelta el dedo de la pantalla, el alimento se encuentra en el collider de la mascota o fuera.

Si está fuera, este regresa al plato, si no, se activa una animación para consumir el alimento y se suman las estadísticas correspondientes al estado de la mascota.

Figura 3.8: Alimentación de la mascota, se puede ver la cantidad que le queda del mismo alimento

3.4. Sistema de guardado

Es importante para comprobar el paso del tiempo en este proyecto tener un sistema de guardado.

Lo primero que ocurre en el sistema de guardado es cargar los datos, "Application.persistentDataPath" [17] nos devuelve una ruta al directorio persistente de la plataforma en la que se está ejecutando la aplicación, son datos que deben conservarse entre ejecuciones de la aplicación.

Lo primero que hacemos en el Awake(), es comprobar si este fichero existe, en caso contrario, debemos crear uno nuevo con una serie de datos por defecto.

El archivo json de este juego guarda los siguientes datos:

- **Love Level:** El nivel de amistad de la mascota.
- **Love Count:** El número del progreso actual de amistad de la mascota.
- **Max Love Count:** Número máximo del progreso de la amistad de la mascota, varía con el nivel por lo que debemos guardarlo también.
- **FoodCourt:** Estadística del hambre de la mascota.
- **Clean Count:** Estadística de la higiene de la mascota.
- **FunCount:** Estadística del entretenimiento de la mascota.
- **StarCoin Amount:** Cantidad de monedas que posee el jugador.
- **Current Pet:** Scriptable Object con los datos de la mascota.
- **Pet Evolving:** Booleano que indica si la mascota está en proceso de evolución o no.
- **Inventory List:** Scriptable Object con los datos del inventario del jugador.

Además, debido al cambio de escena para jugar a los minijuegos, también guardamos la siguiente variable:

- **Fun Recovered On MiniGame:** Estadística que recupera la mascota tras terminar de jugar a los minijuegos.

Si el fichero existe, cargamos los datos del archivo .json a una variable string y con "JsonUtility.FromJson" [18], se parsea el archivo para guardarlo en una clase especial con todos los datos.

Luego cargamos todos los datos a los elementos del juego que correspondan.

Para guardar los datos del juego, hacemos lo contrario, buscamos los datos necesarios en los diferentes elementos del juego, se guardan en una clase especial y con "JsonUtility.ToJson" pasamos todos los datos a un string con el texto de un archivo json y lo escribimos en el archivo de guardado generado anteriormente.

La partida se guardará en cada mínimo cambio que ocurra en el estado de juego. (Las estadísticas de hambre, higiene o entretenimiento cambia, sube de nivel de amistad, se añade o extrae objetos del inventario, etc.)

3.5. Tienda para comprar alimentos

El jugador podrá comprar alimentos con el dinero que obtenga de los minijuegos o al subir el nivel de amistad de la mascota.

La lógica de la interfaz visual de la tienda es la misma que la del inventario, solo que en este caso no es necesario extraer ni añadir ningún objeto, la tienda cuenta con un catálogo fijo de alimentos y el jugador tiene la capacidad de adquirir la cantidad de alimentos que prefiera siempre que tenga dinero suficiente.

Se ha reutilizado todo el código del inventario salvo por el "InventoryController", aquí uso un "ShopController" con las siguientes características:

- Al seleccionar un alimento, se mostrará en la interfaz además del nombre y las estadísticas, el precio.
- También mirará en el inventario del jugador para saber la cantidad de un determinado alimento y mostrará la cantidad disponible en la interfaz.
- Cuando el jugador selecciona un alimento, aparecerá un botón para comprarlo, si en su inventario ya tiene 99 alimentos del mismo tipo, se denegará la compra. O en caso de no tener el dinero necesario.

Figura 3.9: Interfaz de la tienda con la cantidad de objetos en el inventario del objeto seleccionado

3.6. Minijuegos

Los minijuegos desempeñan una doble función: entretener a la mascota y proporcionar dinero para comprar alimentos.

Los minijuegos tienen lugar en una escena diferente de Unity, hasta el momento solo existe un minijuego llamado “Fall Game” donde nuestra mascota caerá por el cielo con un paracaídas y tendrá que esquivar obstáculos y recoger monedas. Cuando el jugador choque con un obstáculo perderá la partida.

3.6.1. Característica general de los minijuegos

Cada minijuego poseerá un controlador que gestiona las siguientes características:

- Registra los puntos que gana el jugador. Por cada 50 puntos, el jugador recibe una moneda. El controlador también se encarga del conteo y la gestión de las monedas obtenidas.

- Administra la cantidad de puntos de entretenimiento que obtiene la mascota en función de su rendimiento en el minijuego.

- Controla los diferentes menús de la interfaz del minijuego, como la ventana emergente que muestra la puntuación y el dinero obtenido al perder la partida. Además, se encarga de manejar el regreso a la escena principal o la posibilidad de reiniciar el minijuego.

- Actualiza los valores de los puntos y las monedas obtenidas en la interfaz del juego.

Figura 3.10: Interfaz que muestra los puntos, monedas obtenidas y un botón para pausar el minijuego

Si el jugador pierde la partida, se guardan automáticamente las monedas conseguidas y el entretenimiento recuperado de la mascota.

Para los minijuegos se guardan sólo los datos relevantes del minijuego y al regresar a la escena principal, el entretenimiento recuperado se suma al que tenía en ese momento y se reinicia a cero.

Al cargar la escena del minijuego, se carga el Scriptable Object de la mascota actual, para acceder a su animator y representarlo en la escena del minijuego.

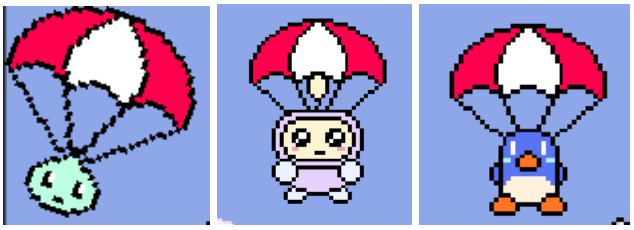

Figura 3.11: Ejemplo con tres mascotas para mostrar la carga del Scriptable Object

3.6.2. Minijuego Fall Game

En este minijuego, la mascota puede moverse horizontalmente utilizando los botones de la interfaz para desplazarse a la izquierda o derecha. Para evitar conflictos entre los botones, cada botón tiene una variable booleana que se establece en verdadero cuando se presiona, mientras que el botón contrario se establece en falso de inmediato. Esto evita problemas cuando el jugador pulsa ambos botones simultáneamente y garantiza un movimiento correcto del personaje.

La mascota se mueve usando su rigidbody2D, modificando la variable “velocity”, además también rota modificando transform.rotation con Quaternion.Lerp [19] para crear un efecto al movimiento.

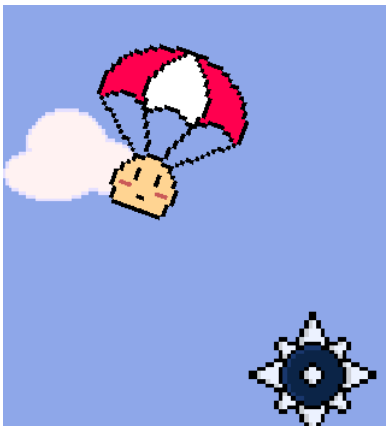

Figura 3.12: Minijuego "Fall Game" se muestra como rota la mascota

En el juego, hay un spawner que genera los obstáculos y las monedas. Estos elementos están predefinidos en prefabs con diferentes cantidades y posiciones. Hay prefabs con monedas y prefabs sin monedas. Cada 3 rondas, aparece un prefab con una moneda y cada 10 rondas aparece un prefab solo con monedas.

Figura 3.13: Ejemplo de varios set que aparecen en el minijuego

Estos prefabs siguen la lógica del Object Pooling [20]. Se instancian de antemano y se seleccionan de forma aleatoria según sea necesario.

Cada oleada que pase (cuando un prefab pase por encima del jugador) aumentará la dificultad del juego haciendo que la velocidad aumente y el tiempo que tardan en aparecer se reduzca hasta cierto umbral.

Los obstáculos y las monedas contienen un circle collider2D y el personaje tiene otro con "IsTrigger" activado y un Rigidbody2D de tipo Kinematic. En el evento OnTriggerEnter2D [21] detectamos con etiquetas cuando el jugador choca con un obstáculo o con una moneda.

3.7. Higiene de la mascota

Cuando la estadística de higiene de la mascota disminuye, esta mostrará signos de mal olor mediante un efecto visual.

Figura 3.14: Efecto visual del mal olor de la mascota

Cuando el jugador acceda al modo de higiene de la mascota, se desplazará a otra zona de la escena, el cuarto de baño. Para esto desplazamos la cámara y a la mascota a la posición correcta.

La rutina de higiene de la mascota consta de dos partes: el lavado con una pastilla de jabón y el aclarado con una ducha.

3.7.1. Lavado de la mascota

Primero el jugador tendrá a su disposición una pastilla de jabón a su izquierda que podrá arrastrar hacia la mascota y moverla para lavarla.

Se reutiliza el mismo script usado anteriormente para mover la comida y alimentar a la mascota, además en otro script se calcula la distancia recorrida con la pastilla de jabón y pasada una distancia determinada se genera una burbuja y se recupera una porción de la estadística de higiene de la mascota (Las burbujas son instanciadas previamente con el método de Object Pooling).

Figura 3.15: Burbujas que aparecen después de lavar a la mascota

Una vez la mascota recupere toda su higiene, cambiamos a la ducha, se le retira la pastilla de jabón, se mueve fuera de la escena y luego aparecerá una ducha.

3.7.2. Aclarar a la mascota con agua.

La ducha seguirá la posición horizontal del dedo del jugador gracias al Touch de Unity que nos dará dicha posición en la pantalla, con ScreenToWorldPoint lo pasamos a una posición del mundo del juego que nos sirva para saber dónde situar la ducha.

Mientras el jugador pulsa la pantalla para mover la ducha, además irán cayendo gotas de agua instanciadas con el método de object pooling y posicionadas en un rango aleatorio debajo de la ducha para hacer el efecto del agua cayendo.

Las gotas de agua se moverán constantemente hacia abajo, contienen un box collider y las burbujas contienen otro box collider y un rigidbody cinemático. Si la burbuja detecta una gota de agua, la hace desaparecer, cuando choque con 3 gotas de agua se desactiva a sí misma.

Figura 3.16: Burbujas desapareciendo al hacer contacto con las gotas de agua de la ducha

Una vez se eliminan todas las burbujas, el juego recompensa al jugador con monedas ya que el proceso de aclarar con agua a la mascota es opcional.

3.8. Interfaz del jugador

Una de las partes más complicadas del desarrollo ha sido la interfaz, muchos errores del juego se producían por no gestionar bien la interfaz y algunas animaciones.

En la parte superior podemos ver un banner con el dinero actual del jugador, debajo aparece el estado actual de la mascota con medidores que indican su hambre, higiene y entretenimiento, estos cambian de color según va bajando la cantidad de cada uno.

Figura 3.17: Interfaz que muestra el estado de la mascota

En la parte inferior de la pantalla, se encuentran cuatro pestañas que permiten al jugador acceder a diferentes acciones del juego.

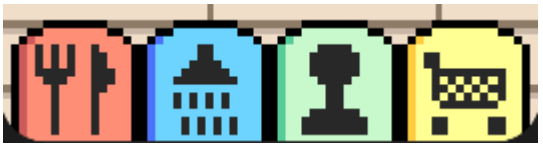

Figura 3.18: Pestañas de la interfaz para acceder a las acciones del juego, en orden, estas pestañas son: Alimentación, Higiene, Minijuegos y Tienda.

Cuando seleccionamos una pestaña, hay una transición con una animación hasta que aparece el menú de dicha pestaña.

Figura 3.19: Animación de transición de la interfaz para la higiene

En el momento de la animación es importante eliminar la posibilidad de que el jugador seleccione otra pestaña diferente ya que da lugar a errores, por lo que se deniega su uso con una variable booleana.

Cuando la animación ha terminado y el jugador selecciona otra pestaña con un menú ya abierto, la anterior desaparece automáticamente con otra animación.

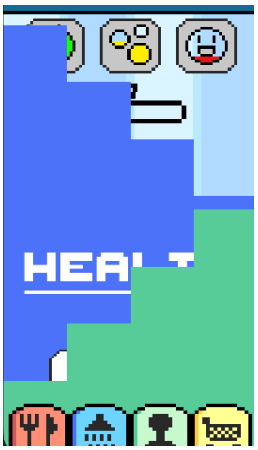

Figura 3.20: Animación de la transición de una pestaña a otra

A excepción del inventario, todos los menús ofrecen la posibilidad de utilizar varios botones, lo que permite añadir contenido adicional en un futuro.

Presupuesto

Este es el presupuesto aproximado para este proyecto suponiendo que el coste de cada hora invertida sea de 30€ /hora.

Para esto se ha dividido el trabajo invertido en:

- Programación: Tiempo invertido en escribir el código de los Scripts en C#

- Arte: Tiempo empleado en crear los sprites del juego con un estilo pixelart.

- Diseño: Tiempo para idear las mecánicas jugables y todos los elementos necesarios para conseguir realizar un prototipo.

- Producción: Tiempo que se ha necesitado para pensar en cómo estructurar y desarrollar el prototipo.

Sección	Tiempo / Dinero invertido
Programación	250 horas -> 7500€
Arte	80 horas -> 2400€
Diseño	30 horas -> 900€
Producción	30 horas -> 900€

Total de horas: 390 horas.

Total de presupuesto: 11700 €

Conclusiones y Líneas futuras

Con todas las características descritas en el documento, se ha conseguido desarrollar un prototipo de un videojuego sobre una mascota virtual que el jugador debe cuidar en su alimentación, higiene y entretenimiento.

Este proyecto aborda los aspectos más básicos del cuidado de una mascota virtual, pero se pensaron en más contenidos para añadir.

La idea del juego completo incluiría los siguientes contenidos:

- El progreso principal del juego sería la colección de las mascotas, se incluiría una sección con el registro de todas las mascotas disponibles en el juego y aparecerán las que hemos desbloqueado.
- Existirían diferentes tipos de huevos que incluirían diferentes tipos de mascotas por temáticas (Mascotas con temática acuática, mascotas con temática de frutas, etc.)
- Una vez se completa el ciclo de vida de una mascota, aparecerá una animación que lleva la mascota al cielo y se convierte en una estrella, más adelante se nos proporciona otro huevo y empezamos de cero con una nueva mascota. Existiría un registro con todas las mascotas que hemos cuidado.
- Las mascotas tendrían una probabilidad de tener un color especial, lo que incentiva la colección.
- En el juego podremos decorar la habitación con una colección de muebles, suelos, alfombras y paredes. Estos contenidos se pueden comprar en la tienda.
- Podemos darle sombreros a la mascota, los cuales también compraremos en la tienda.
- Cuando se termina el ciclo de vida de la mascota, este nos suelta un objeto especial que podremos intercambiar en la tienda o usar para expandir el inventario, etc.
- Si alimentamos a la mascota con muchos dulces puede llegar a enfermar, tendremos que acceder al apartado de cuidados de la mascota para curarla.
- Existe un tipo de caramelo especial que permite reducir el tiempo de espera cuando la mascota está en proceso de evolución.
- El nivel de jugador serviría para desbloquear poco a poco el contenido del juego como los minijuegos, los alimentos, sombreros y muebles.
- La mascota también necesitará descansar, podremos elegir su horario de sueño y no podremos interactuar con él en ese tiempo.
- Diferentes animaciones y estados de ánimo que representen cuando la mascota está feliz, enfadada, etc.

- Misiones diarias que nos proporcionarán dinero y experiencia para subir el nivel del jugador.

La intención es poder actualizar el juego fácilmente con ciertos contenidos ya programados como con más mascotas, sombreros y muebles. Además se pueden incluir diferentes tipos de minijuegos.

En el proceso de este trabajo he aprendido información importante sobre el uso de Scriptable Objects, sistemas de guardado con archivos .json y formas de interactuar en el juego con la pantalla táctil.

También he aprendido a gestionar bien los scripts para poder reutilizarlos fácilmente como por ejemplo en el uso de arrastrar un objeto del juego con la pantalla táctil. Se utilizó el mismo código tanto en el alimento como en la pastilla de jabón, luego otro script se encargaba de las características únicas de cada funcionalidad.

Durante el transcurso de este Máster Universitario en Desarrollo en Videojuegos he aprendido muchos contenidos útiles para este proyecto.

La asignatura de Fundamentos del Desarrollo de Videojuegos me ha servido para aprender sobre videojuegos en 2D, físicas y otras técnicas como el Object Pooling en Unity.

Y en la asignatura de Temas Avanzados de la Tecnología de los Videojuegos he aprendido muchos estándares en el diseño de interfaces que he aplicado para este proyecto.

En general en este proyecto he aprendido muchas cosas sobre el desarrollo de aplicaciones móviles en Unity

He intentado seleccionar un proyecto de videojuego lo más sencillo posible con intención de poder completarlo y aun así he aprendido que el desarrollo de videojuegos es costoso y duradero.

Tengo intención de continuar este proyecto e intentar publicarlo algún día.

Summary and Conclusions

Based on the described features, a prototype of a virtual pet video game focusing on feeding, hygiene, and entertainment has been successfully developed. While this project covers the fundamental aspects of virtual pet care, there are additional contents that could be incorporated.

The idea of the complete game would include the following contents:

- The main progress of the game would be the collection of pets, a section with the record of all the pets available in the game would be included and the ones we have unlocked would appear.
- There would be different types of eggs that would include different types of pets by themes (aquatic-themed pets, fruit-themed pets, etc.).
- Once the life cycle of a pet is completed, an animation will appear that takes the pet to the sky and it becomes a star, later we are provided with another egg and we start from scratch with a new pet. There would be a record of all the pets we have taken care of.
- The pets would have a chance to have a special color, which encourages the collection.
- In the game we will be able to decorate the room with a collection of furniture, floors, carpets and walls. These contents can be purchased in the store.
- We can give hats to the pet, which we can also buy in the store.
- When the pet's life cycle is over, it releases a special item that we can exchange in the store or use to expand the inventory, etc.
- If we feed the pet with too much candy it can get sick, we will have to access the pet care section to cure it.
- There is a special type of candy that allows to reduce the waiting time when the pet is in the process of evolution.
- The player level would serve to gradually unlock game content such as mini-games, food, hats and furniture.
- The pet will also need to rest, we will be able to choose its sleep schedule and we will not be able to interact with it during that time.
- Different animations and moods to represent when the pet is happy, angry, etc.
- Daily missions that will provide us with money and experience to level up the player.

The intention is to be able to update the game easily with certain contents already programmed as with more pets, hats and furniture. In addition, different types of mini-games can be included.

In the process of this work I have learned important information about the use of Scriptable Objects, saving systems with .json files and ways to interact in the game with the touch screen.

I have also learned how to manage scripts well so that I can easily reuse them as for example in the use of dragging an object in the game with the touch screen. The same code was used in both the food and the soap bar, then another script handled the unique features of each functionality.

During the course of this Master's Degree in Video Game Development I have learned a lot of useful content for this project.

The subject of Fundamentals of Video Game Development has helped me to learn about 2D video games, physics and other techniques such as Object Pooling in Unity.

And in the Advanced Topics in Video Game Technology course I have learned many standards in interface design that I have applied for this project.

Overall in this project I have learned many things about mobile application development in Unity.

I have tried to select as simple a video game project as possible with the intention of being able to complete it and yet I have learned that video game development is expensive and long lasting.

I intend to continue this project and try to publish it someday.

Bibliografía

- [1] Historia de la mascota virtual. Revista Central. https://www.revistacentral.com.mx/actualidad/tamagotchi-historia-mascota-virtual
- [2] Bandai. Tamagotchi original. Wikipedia. https://es.wikipedia.org/wiki/Tamagotchi
- [3] Bandai Namco. Tamagotchi Pix. Tamagotchi. https://tamagotchi.com/es/productos/tamagotchi-pix/
- [4] Polaris-x Inc. Kotodama Diary. Google Play. https://play.google.com/store/apps/details?id=com.ske6bento.kotodama&hl=es&gl=US
- [5] Bandai Namco. My Tamagotchi Forever. Google Play. https://play.google.com/store/apps/details?id=eu.bandainamcoent.mytamagotchiforever&hl=es&gl=US
- [6] Igara Studio S.A. Aseprite. Aseprite. https://www.aseprite.org/
- [7] Unity Technologies. Touch. Unity Manual. https://docs.unity3d.com/ScriptReference/Touch.html
- [8] Unity Technologies. Camera.ScreenToWorldPoint. Unity Manual. https://docs.unity3d.com/ScriptReference/Camera.ScreenToWorldPoint.html
- [9] Unity Technologies. Scriptable Object. Unity Manual. https://docs.unity3d.com/Manual/class-ScriptableObject.html
- [10] C#. DateTime.Now.Ticks. Learn Microsoft. https://learn.microsoft.com/en-us/dotnet/api/system.datetime.ticks?view=net-8.0
- [11] Unity Technologies. PlayerPrefs. Unity Manual. https://docs.unity3d.com/ScriptReference/PlayerPrefs.html
- [12] Unity Technologies. Time.deltaTime. Unity Manual. https://docs.unity3d.com/ScriptReference/Time-deltaTime.html
- [13] Unity Technologies. ScrollRect. Unity Manual. https://docs.unity3d.com/es/2018.4/Manual/script-ScrollRect.html
- [14] Unity Technologies. Grid Layout Group. Unity Manual. https://docs.unity3d.com/es/2019.4/Manual/script-GridLayoutGroup.html
- [15] Unity Technologies. OnPointerClick. Unity Manual. https://docs.unity3d.com/2019.1/Documentation/ScriptReference/UI.Button.OnPointerClick.html
- [16] Unity Technologies. Physics.Raycast. Unity Manual. https://docs.unity3d.com/ScriptReference/Physics.Raycast.html

- [17] Unity Technologies. Application.persistentDataPath. Unity Manual. https://docs.unity3d.com/ScriptReference/Application-persistentDataPath.html
- [18] Unity Technologies. JsonUtility. Unity Manual. https://docs.unity3d.com/ScriptReference/JsonUtility.html
- [19] Unity Technologies. Quaternion.Lerp. Unity Manual. https://docs.unity3d.com/ScriptReference/Quaternion.Lerp.html
- [20] Unity Technologies. Object Pooling. Unity Learn. https://learn.unity.com/tutorial/introduction-to-object-pooling#
- [21] Unity Technologies. OnTriggerEnter2D. Unity Manual. https://docs.unity3d.com/ScriptReference/MonoBehaviour.OnTriggerEnter2D.html