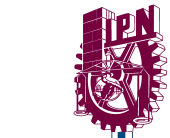


INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

ESCOM

Trabajo Terminal

“Yolotl: un videojuego para fomentar la cultura”

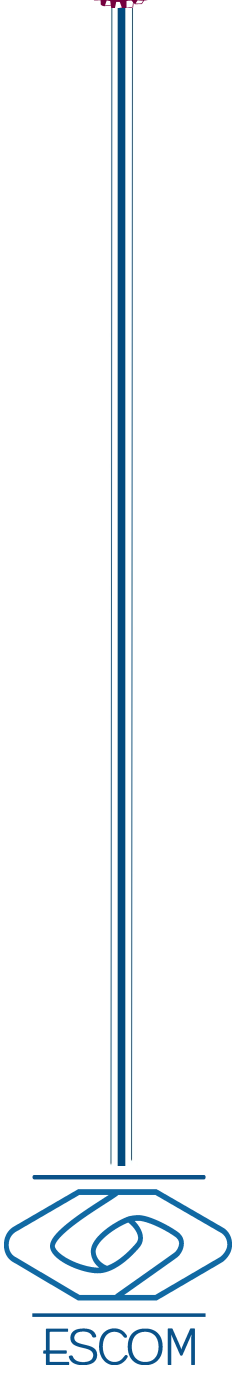
2017-A035

Presentan
Hernández Bautista Yasmine Pilar

Márquez Hernández Karla Rocío

Directores
M. en C. Rafael Norman Saucedo Delgado.

Lic. Ulises Vélez Saldaña.



Noviembre 2017

INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

No. de TT:2017-A035 17 de noviembre de 2017

Documento Técnico Parte A

"Yolotl: un videojuego para fomentar la cultura"

Presentan
Hernández Baustista Yasmine Pilar[1]

Márquez Hernández Karla Rocío[2]

Directores

M. en C. Rafael Norman Saucedo Delgado. ***Lic. Ulises Vélez Saldaña***.

RESUMEN

En México la industria de videojuegos tiene una alta demanda de consumo; sin embargo, existen pocos estudios que desarrollen videojuegos basados en la cultura mexicana. Actualmente en México existe un fuerte desinterés en la cultura nacional. El presente trabajo terminal consiste en el desarrollo de un videojuego que fomente la cultura con temática de la cultura mexica.

Palabras clave. – Cultura mexica, desarrollo tecnológico, ingeniería de software, videojuego.

[1]daughterofthewind10@gmail.com
[2]yolotl.escom@gmail.com

Advertencia

"Este documento contiene información desarrollada por la Escuela Superior de Cómputo del Instituto Politécnico Nacional, a partir de datos y documentos con derecho de propiedad y por lo tanto, su uso quedará restringido a las aplicaciones que explícitamente se convengan." La aplicación no convenida exime a la escuela su responsabilidad técnica y da lugar a las consecuencias legales que para tal efecto se determinen. Información adicional sobre este reporte técnico podrá obtenerse en: La Subdirección Académica de la Escuela Superior de Cómputo del Instituto Politécnico Nacional, situada en Av. Juan de Dios Bátiz s/n Teléfono: 57296000, extensión 52000.

Índice general

Capítulo 1

Introducción

Capítulo 2

Antecedentes

2.1. Propuesta

2.2. Ajuste en el plan de trabajo

En esta sección se definen todas las nuevas estrategias a seguir para agilizar y optimizar el desarrollo del juego y el desarrollo de los *sprites* faltantes del juego.

Las nuevas estrategias de desarrollo

Para el sexto *sprint* y teniendo como base la experiencia de desarrollo los anteriores prototipos, quedaba claro que se necesitaba diseñar una nueva estrategia que permitiera agilizar el desarrollo del juego sin comprometer la calidad del mismo. La naturaleza iterativa de *Huddle* permite controlar la extensión del desarrollo y la asignación de tareas entre miembros del equipo, pero al estar orientada a niveles producía que la programación se tuviera que pausar cada vez que se iniciaba un nuevo nivel para diseñar la maqueta del nivel y para el desarrollo de los *sprites*. Por tal motivo se propuso un nuevo enfoque a *Huddle* orientar su naturaleza iterativa al desarrollo de los componentes del juego, una vez que estos esten completados y probado su funcionamiento de manera individual, se inicia la construcción de los niveles. Bajo este nuevo enfoque primeramente se crean las maquetas de todos los niveles, seguido de la creación de todos los *sprites*, para después programar los assets que correspondían a los actores, después se implementan las clases controladoras que comparten todos los niveles, como el controlador del personaje, el controlador de la barra de vida, etc.; con los controladores comunes funcionando se crea una escena base que incluye el personaje jugable y la interfaz de juego, sobre esta escena se construyen los niveles del juego. El anterior enfoque es una pequeña abstracción del Método *Centry* descrito en la sección **??**.

Ya definido el nuevo plan de trabajo, se procedió a diseñar las maquetas para los niveles. Para el diseño de maquetas se siguió utilizando la plantilla creada durante Trabajo Terminal 1. Si se desea ver las maquetas de los niveles, estas se encuentran en el anexo **??**.

Actualizando el *software* de desarrollo

Paralelamente al diseño de las maquetas de los niveles, fue liberada la versión 2017.3.1f de *Unity3D*. Esta versión incluye herramientas que agilizan la creación de niveles como el uso de:

- ***Tilemap*:** Herramienta para el mapeado de niveles. Esta herramienta facilita la creación de mapas al crear una malla sobre la que se arrastraran diferentes *Sprites* que se hayan importado previamente al tilemap (ver figura **??**). En la sección () se profundizará su funcionamiento.

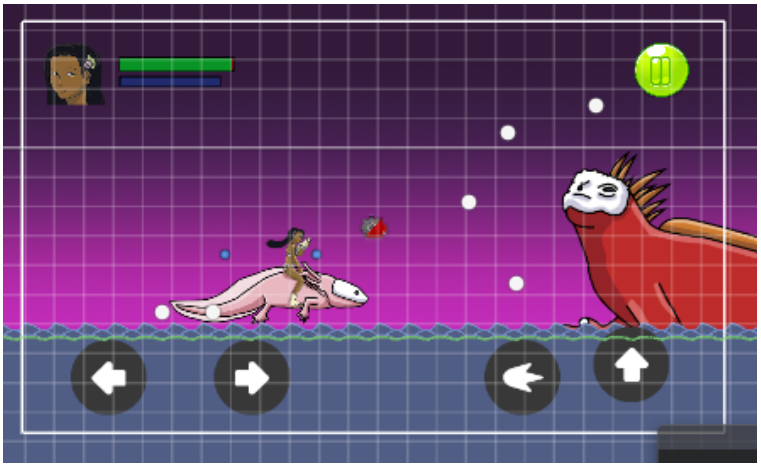


Figura 2.1: Vista de la escena cuando se tiene un *GameObject* de tipo *Tilemaps* para la construcción de niveles

- ***Cinemachine*:** *Asset* que permite controlar la cámara de la escena, con este *asset* se le puede indicar que objeto se desea que la cámara siga y se puede asignar un área que limitara el movimiento de la cámara (ver figura **??**). *Cinemachine* se descarga directamente desde la tienda de *assets* de *Unity* y fue desarrollado por los ingenieros de *Unity*, lo que significa que no genera conflictos o no requiere de configuraciones extras al proyecto para importar. En la sección () se profundizará su funcionamiento.

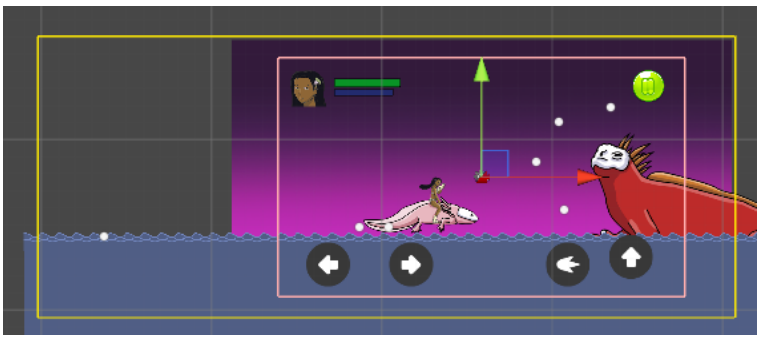


Figura 2.2: Vista de la escena cuando se tiene un *GameObject* de tipo *Tilemaps* para la construcción de niveles

- ***Sprite Packer***: Si bien no es una herramienta para construcción de niveles o un *asset*, esta herramienta es una de las más útiles que se agregó a la nueva versión de *Unity* ya que, como su nombre lo indica, permite el empaquetado de *sprites* (ver figura). Empaquetar los *sprites* es una práctica que optimiza el renderizado de objetos, ya que el controlador de gráficos

de *Unity* realiza una sola llamada por paquete cuando renderiza los objetos y con esa única llamada renderiza todos los objetos de la escena que se encuentren en ese paquete; si los *sprites* no se encontraran dentro de un paquete el controlador de gráficos de *Unity* haría una llamada por cada *sprite*.

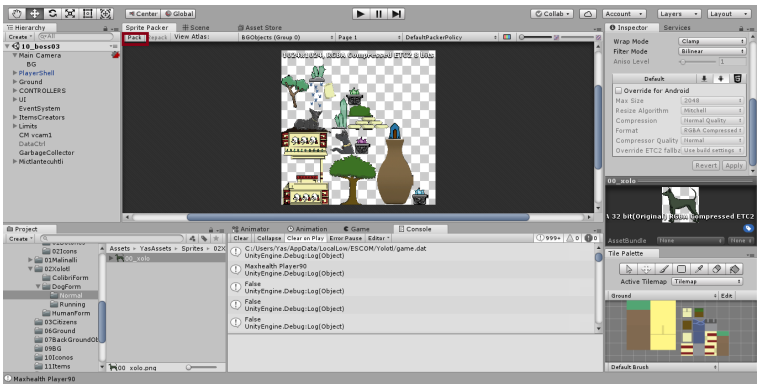


Figura 2.3: Vista de la pestaña del *Sprite Packer*.

Por el impacto que tendrían las nuevas herramientas de la versión de *Unity*, se propusó utilizarla en lugar de la versión 5.6.2f1. Antes de actualizar la versión de *Unity* se investigó si el proyecto sufriría algún impacto negativo como falta de compatibilidad de componentes por la diferencia de versiones. Al comprobar que existía una total compatibilidad entre ambas versiones en cuanto a trasladar un proyecto de la versión 5.6.1f a la versión 2017.3.1f. Se determinó que la nueva versión de *Unity* sería la que se emplearía para el resto del desarrollo del juego.

2.3. Contribuciones

Capítulo 3

Resultados obtenidos

3.1. Prueba

3.1.1. Objetivo de la prueba

3.1.2. Herramientas utilizadas durante la prueba

3.1.3. Aplicación de la prueba

3.1.4. Conclusiones de la prueba

Capítulo 4

Conclusiones

Capítulo 5

Anexos

En este capitulo se encuentran todos los anexos que se mencionaron en los capitulos anteriores.

5.1. Interfaces

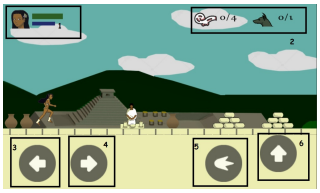


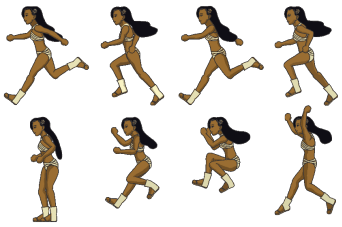
Figura 5.1: a nice plot

5.2. Diseño de Personajes

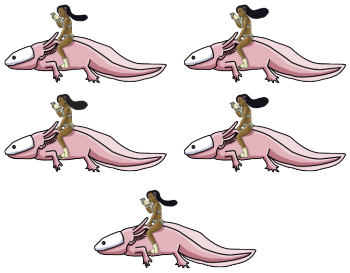
5.3. Modelo de Datos

5.4. Control de adicción en el jugador

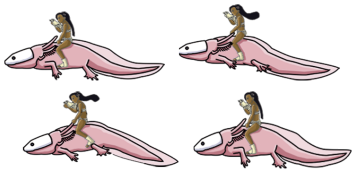
5.5. Maquetas de niveles



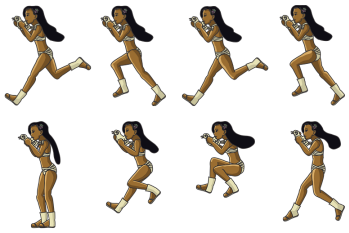
(a) Sprites Malinalli para el primer nivel.



(b) Sprites Malinalli de nado para el segundo nivel.



(c) Sprites Malinalli de salto para el segundo nivel.



(d) Sprites Malinalli para los niveles posteriores al segundo nivel.

Figura 5.2: Sprites del personaje jugable (Autoria propia)

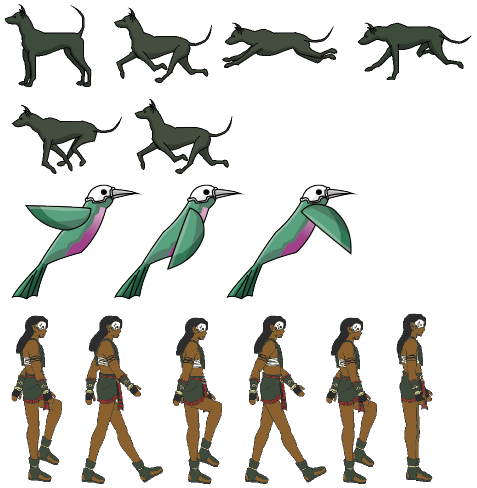


Figura 5.3: Sprites para las diferentes formas que toma Xólotl a lo largo del juego.

Figura 5.4: Sprites para los enemigos normales del juego.

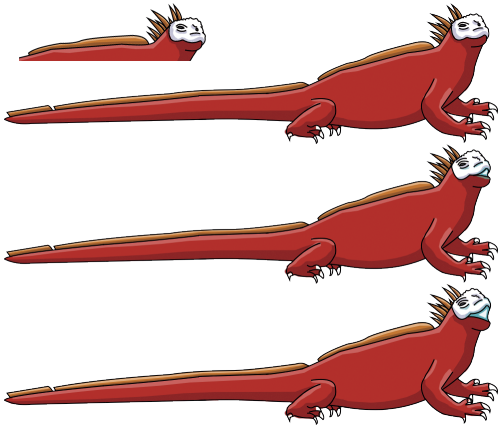


Figura 5.5: Sprites de Xochitonal.

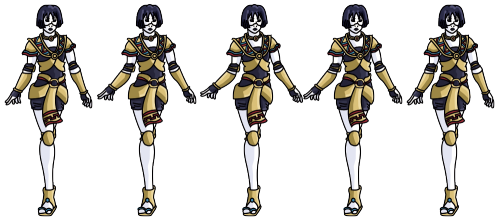


Figura 5.6: Sprites de Itzpapálotl.

Figura 5.7: Sprites de Tlazolteotl.

Figura 5.8: Sprites de Tepeyollotl.

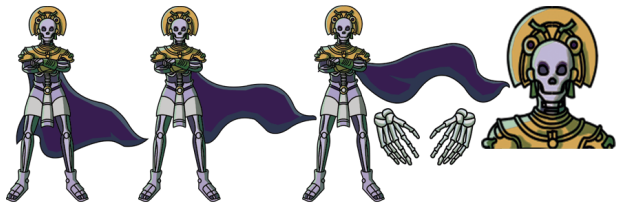


Figura 5.9: Sprites de Mictlantecuhtli.

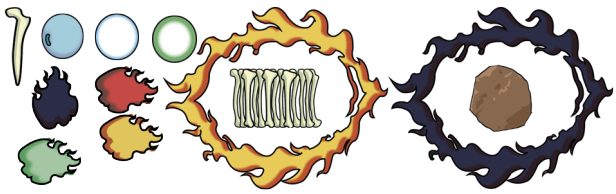


Figura 5.10: Sprites de los ataques de los personajes.