Práctica Final - Herramienta en C++ y C# para Unity

Middleware: Herramientas de Desarrollo (Programación)

Manual de Uso

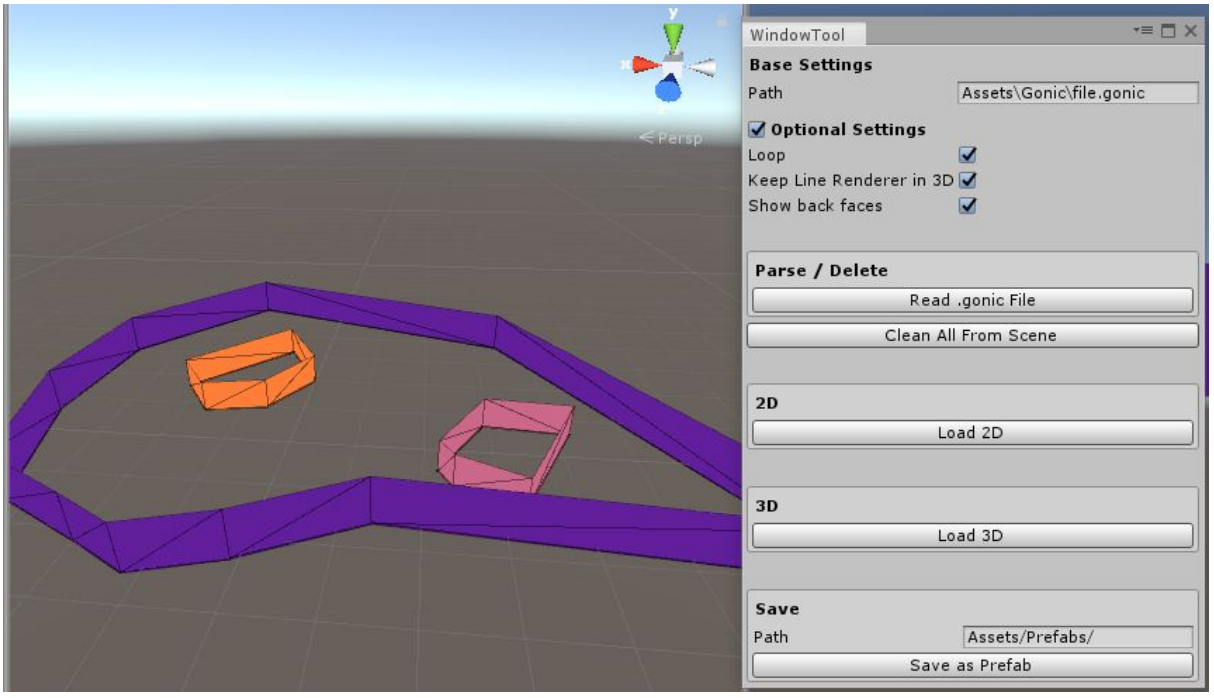
Nicolás Tapia Sanz
4.3 - DVJU
ESNE 2018/2019

Objetivo

El alumno debe implementar una librería escrita en C++ en Unity y desarrollar una extensión del editor que se apoye sobre esa librería. Se debe implementar un wrapper en C# que aúne toda la funcionalidad de la librería y que exponga de manera legible al nivel superior.

Se creará una extensión del editor que despliegue una ventana. En dicha interfaz podrán establecerse los parámetros que la librería soporte. La experiencia de un usuario debe ser sólida y se deberá evitar que el usuario use parámetros inválidos o se cuelgue el editor.

El software se debe poder ejecutar en Windows, debe haber un modelo de datos diferenciado que represente la estructura de los documentos sobre los que trabaja la herramienta. El modelo de datos debe ser controlado exclusivamente por código diferenciado e independiente de la interfaz de usuario (backend). Se debe poder configurar el funcionamiento de la herramienta. Se debe poder guardar de modo persistente algún tipo de información. La herramienta debe estar en forma de librería de enlace dinámico.



Requisitos

Para el desarrollo esta práctica se ha utilizado Unity 2019.4.1f1 y Visual Studio 2017. No se ha probado en ninguna otra versión de Unity y la herramienta podría fallar si no se utiliza en la misma versión mencionada anteriormente.
Enlace a descarga: https://unity3d.com/es/get-unity/download/archive

Descripción

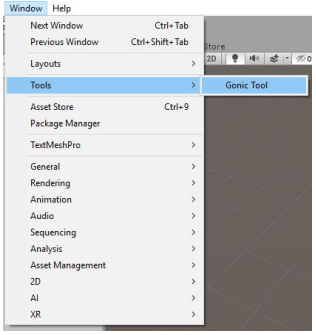
El objetivo inicial de esta práctica era el de generar una herramienta que permitiera la lectura de archivos DWG o CAD para la generación de casas en 3D a partir de planos en 2D. Más tarde, y dado la complejidad de estos archivos y siguiendo el consejo del profesor de la asignatura de Middleware, se decidió retrasar este objetivo y tratar de crear un archivo propio que contuviese por dentro un XML modificable y de fácil lectura y edición similar a los archivos SVG. De esta manera surgió la extensión GONIC, un formato que contiene capas con modelos en su interior que tienen un conjunto de puntos 2D. A su vez describen la altura que deben alcanzar estos modelos, así como el color con el que están representados.

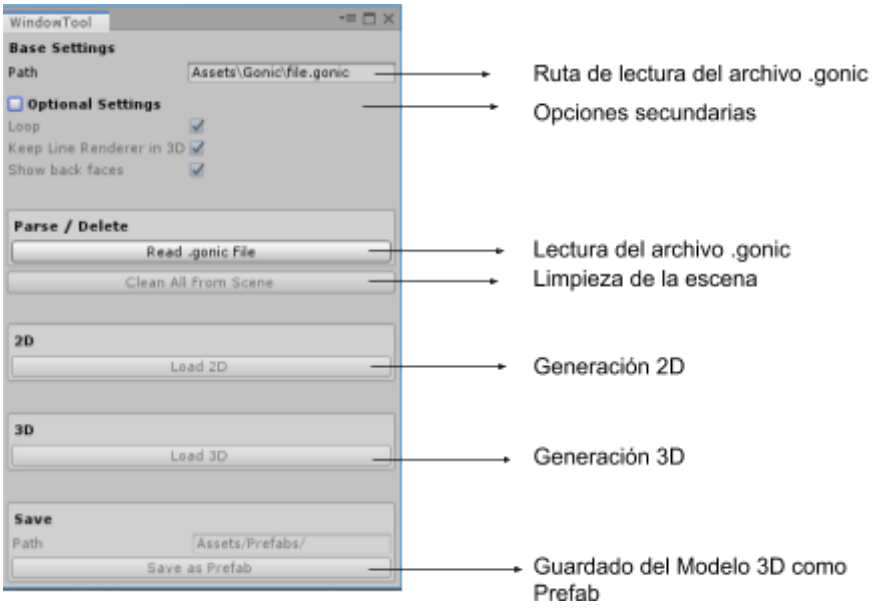
Se ha desarrollado una herramienta que permite abrir archivos .gonic, mostrarlos en un formato 2D, generar su modelo 3D y su guardado como Prefab, así como los Meshes y los Materiales (formatos de Unity). Para mayor entendimiento, el proyecto contiene un archivo .gonic ya creado que se puede encontrar en la carpeta *Assets* > *Gonic* > *file.gonic* y que se puede probar a modo de ejemplo.

Esta herramienta estaba planteada para unirse a la práctica del compañero Gonzalo Pérez Chamarro y poder crear una herramienta lo más completa posible. En el resultado entregado, esto no puede verse, ya que no está terminado (aunque no era necesario).

UI

Para abrir la herramienta, el usuario debe abrir el proyecto de Unity (*projects* > *unity* > *Assets* > *Scenes* > *SampleScene)* adjunto a este archivo. La herramienta funciona en cualquier escena. Para abrir la venta, el usuario debe ir a *Window* > *Tools* > *Gonic Tool.*





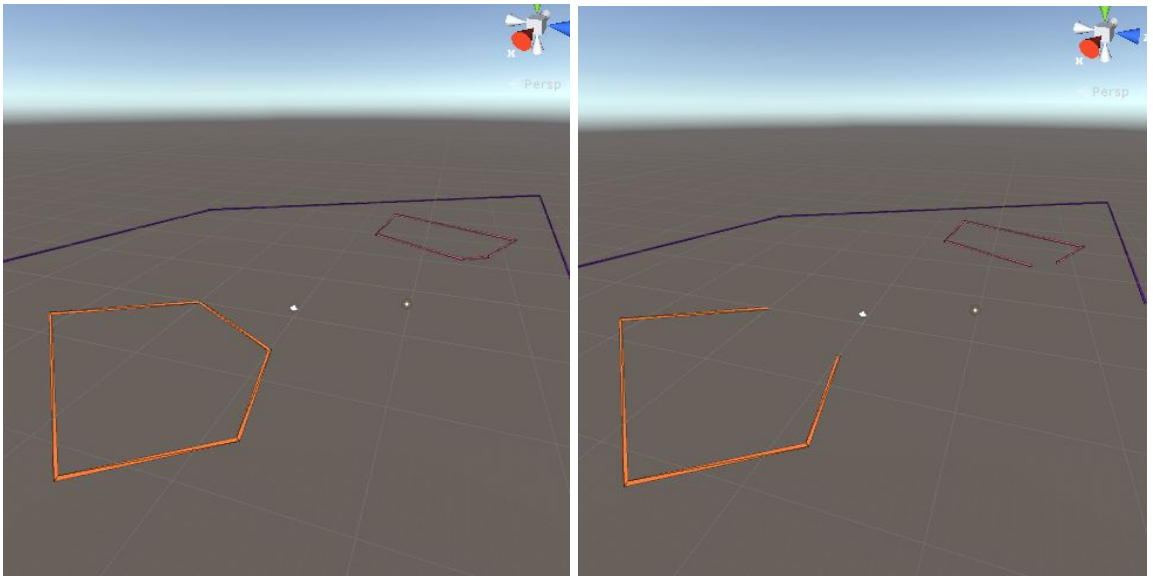
Documentos

En la carpeta *documentation,* donde está alojado este mismo archivo, se puede encontrar la documentación Doxygen del proyecto de C# y del proyecto de C++. Ambas documentaciones se ha hecho por separado al considerarse independientes.

Uso

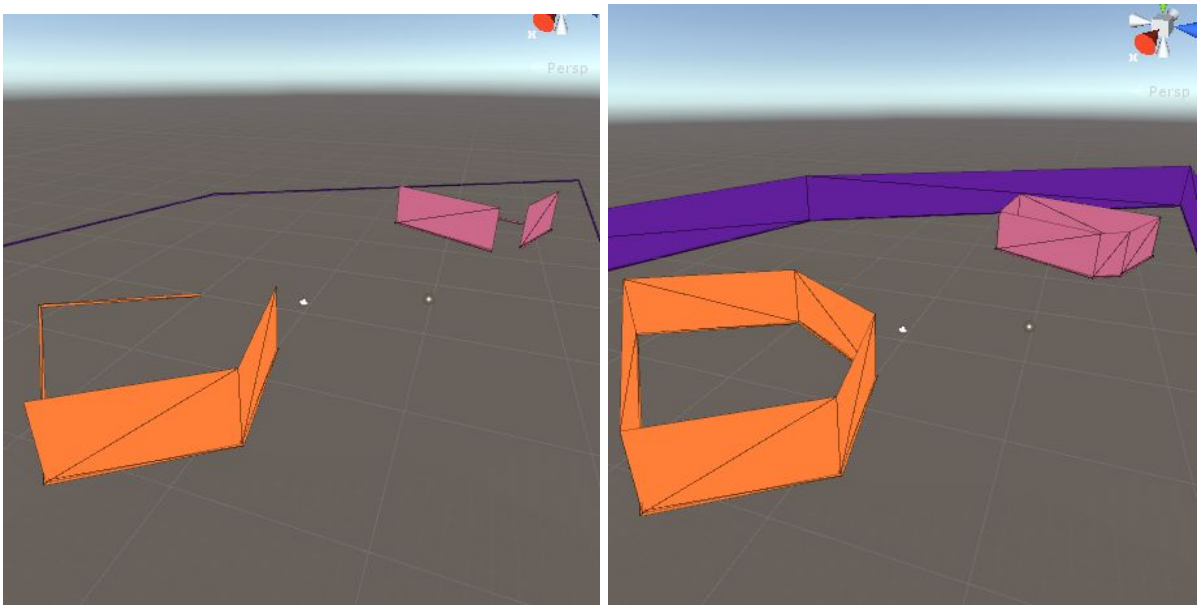
Para la generación de estos modelos, lo primero que debe hacer el usuario es introducir una ruta válida. En caso contrario, si la ruta no es válida, propondrá crear una carpeta para que lo sea. Si no encuentra un archivo .gonic con ese nombre, propondrá la creación de un archivo .gonic con una pequeña plantilla de base en su interior. Los archivos .gonic pueden modificarse mediante cualquier editor de textos (Bloc de Notas, Notepad++...).

Si la ruta es totalmente válida, leerá los datos del modelo y la opción de generar modelo 2D se desbloqueará. En este caso, la opción secundaria *Looped* permite que, si el modelo descrito en el archivo .gonic no es un modelo cerrado, se pueda cerrar y crea una cara adicional.



Con los modelos 2D ya generados, se puede utilizar el botón que borra los modelos, aunque este se desbloquee desde el momento en que se lee el archivo para la eliminación de posibles objetos generados en otras ocasiones en la misma escena.

Ahora se puede generar los modelos 3D. El usuario puede decidir si mantener las líneas base que se muestran en la escena o no al ser generados, así como ver las caras traseras de los modelos 3D o no.



Una vez que ya se han generado los modelos 3D, el usuario puede guardar los modelos como prefabs en la ruta que él defina. Así mismo, se guardan los meshes en su carpeta correspondiente (*Assets* > *Meshes*) y los materiales de las mallas y los Line Renderer (si estuvieran activados) (*Assets* > *Materials*).

Características Funcionales de la Herramienta

Funcionalidades

- **Lectura / Escritura de archivos con extensión .gonic**: Estos archivos contienen en su interior un XML que describen un conjunto de capas que a su vez contiene un conjunto de modelos 2D. Estos modelos 2D vienen descritos por un nombre, una altura, un color y una serie de puntos que son los vértices de estos modelos. en 2 dimensiones. En caso de que no encuentre un archivo .gonic en la ruta que se le proporcione, la herramienta genera un archivo .gonic con una pequeña plantilla en su interior.

- **Eliminación de Modelos 2D/3D generados con anterioridad**: En caso de que se hayan generado una serie de modelos, independientemente de su forma, el usuario podrá borrarlos de la escena mediante un único botón.

- **Generación de Modelo 2D:** Una vez que el archivo ha sido leído, el usuario puede generar los Modelos 2D en la escena de Unity. Estos modelos viene representados mediante un Line Renderer de Unity. Sus coordenadas son modificadas de 2 dimensiones a 3, ya que así lo exige el componente de Unity, utilizando las dimensiones X y Z e ignorando la Y.

- **Generación de Modelo 3D**: Una vez que se ha generado una versión 2D del modelo, el usuario puede generar los Modelos 3D de la escena. Para ello, la herramienta internamente calcula las coordenadas generadas mediante la altura de todos los vértices y la disposición de los mismos para la generación de los triángulos necesarios.

- **Opciones secundarias**: Además de lo anterior, el usuario cuenta con un conjunto de opciones secundarias que le permitirán generar formas cerradas, la visualización de las caras traseras de los modelos y mantener la visibilidad de las líneas 2D en los modelos 3D. Todas estas opciones son opcionales.

Modelo de Datos

Gonic
Contiene todas las capas del modelo, tanto en su versión 2D como en su versión 3D.

2D

Layer2D
Contiene los modelos 2D de una capa concreta. Tiene un nombre determinado.

Model2D
Es un modelo 2D que contiene los vértices 2D que lo forman, un nombre y un color con el que ser representado

Vector2D
Es un vector con dos componentes: X e Y.

3D

Layer3D
Contiene los modelos 3D de una capa concreta. Tiene un nombre determinado. Se crea a partir de una Layer2D

Model3D
Es un modelo 3D que contiene los vértices 3D que lo forman, un nombre y un color con el que ser representado. Se crea a partir de un Model2D.

Vector3D
Es un vector con tres componentes: X, Y y Z.

Funciones del Controlador / Wrapper

- Creación de la herramienta en C++. Envío del puntero de la herramienta en C+ a C#.
- Parseo del archivo .gonic.
- Obtención del número de capas 2D para la generación correcta de elementos en la escena.
- Obtención del nombre de una determinada capa 2D para la correcta generación de elementos en la escena.
- Obtención del número de modelos en una determinada capa 2D.
- Obtención del nombre de un determinado modelo 2D de una determinada capa 2D para la correcta generación de elementos en la escena.
- Obtención de la cantidad de vectores de un determinado modelo 2D de una determinada capa.
- Relleno de un array con los valores de los vectores 2D de un determinado modelo 2D de una determinada capa.
- Relleno de un array con los valores de los vectores 3D de un determinado modelo 3D de una determinada capa.
- Obtención del color de un determinado modelo 2D.
- Generación de un capa 3D a partir de una capa 2D
- Conversión de un modelo 2D a un Modelo 3D
- Cálculo de los triángulos de un modelo 3D.

Características de la práctica Implementadas

- La herramienta no genera ningún tipo de error ni warning en su compilación y tampoco durante su ejecución en Unity.
- Se ha tratado de desarrollar el código según la explicaciones recibidas en clase. Se ha tratado de dejar sencillo y claro.
- Se ha generado la documentación de los lenguajes C++ y C# así como este mismo documento explicativo.
- Se ha generado dos modelos de datos: 2D y 3D tal y como el profesor me indicó en clase.
- Se ha tratado de generar un backend totalmente independiente de Unity, de manera que la DLL se puede llevar a otros medios, engines, etc.
- Se ha generado una interfaz lo má atractiva visualmente para el usuario y tratando de que fuera clara y sencilla aprovechando la facilidad que proporciona Unity para su generación.
- Se lee archivos .gonic y se pueden guardar dichos modelos en la misma escena o en los Assets del proyecto.
- Se ha utilizado C# como lenguaje de scripting
- Se ha creado una librería de enlace dinámico mediante C++.

Comentarios Extra

Me gustaría añadir a toda esta documentación, que nos hubiera gustado a Gonzalo y a mi mezclar ambas herramientas, pero por tiempo nos ha sido imposible. Aun así, ha resultado una herramienta totalmente independiente y que funciona correctamente.

Así mismo, me hubiera gustado poder añadir formas geométricas concretas en el archivo .gonic, tales como cuadrados, esferas, conos, etc. sin tener que describir todos los puntos. No hubiera sido demasiado complicado pero, de nuevo, por tiempo ha sido imposible.

La implementación de los archivos .CAD ha sido totalmente imposible por falta de tiempo y de conocimientos claros para el uso de la librería de terceros correspondiente.

Link a repositorio

https://github.com/Nikoooo95/Tool-in-Unity/tree/develop