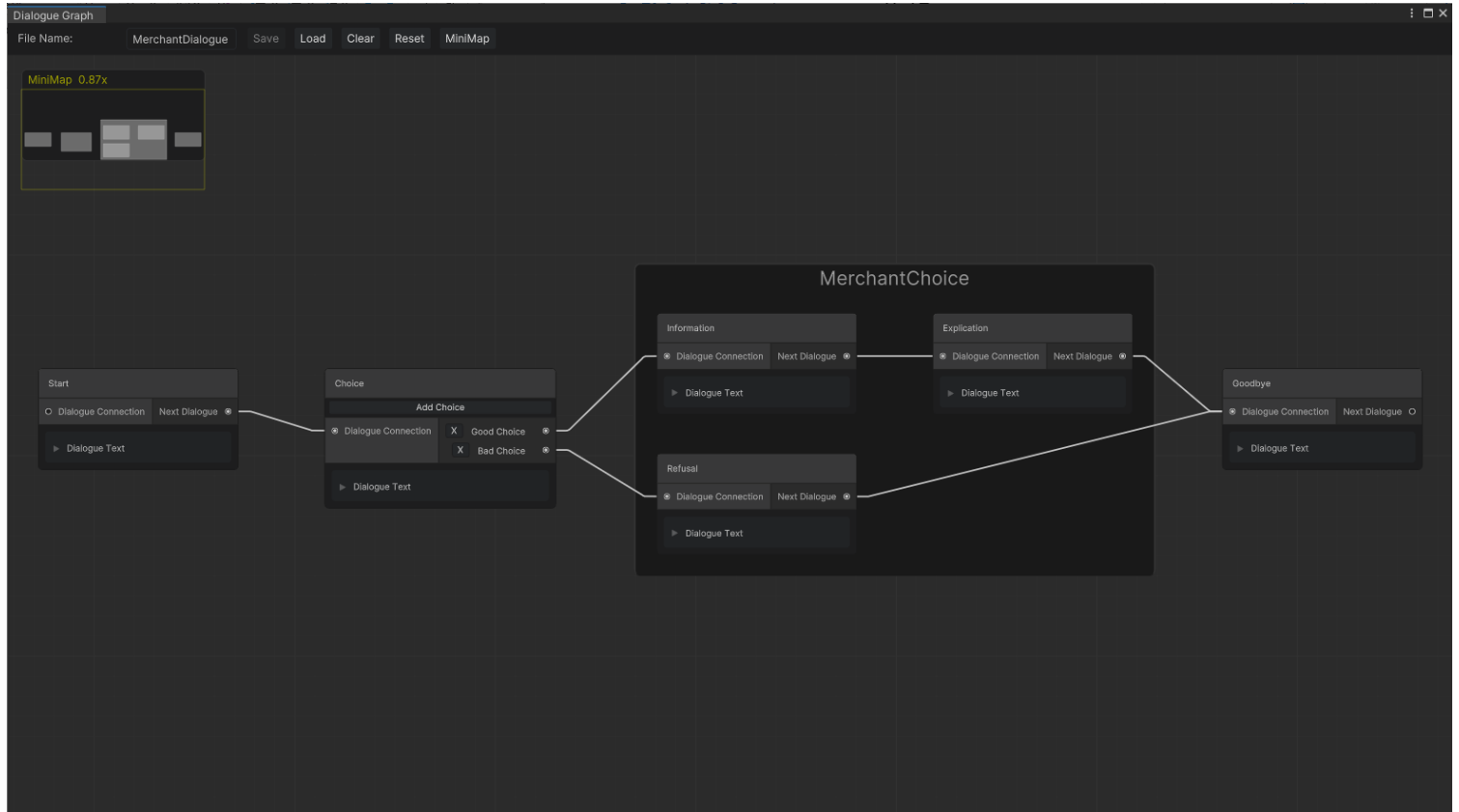




UNIVERSIDAD DE
DISEÑO, INNOVACIÓN
Y TECNOLOGÍA



Herramienta – Dialogue System

Middleware

Primera Práctica - GDDV 4.3

Martin Pérez Villabrille

Índice

Introducción.....	2
Requisitos Funcionales	2
Requisitos Técnicos.....	8
Diagrama de componentes	11
Modelo de datos.....	12
Diagrama de clases	13
Dialogue System.....	13
Elementos.....	14
Ventanas	14
Inspector Personalizado	16
Sistema de Guardado y Cargado	16

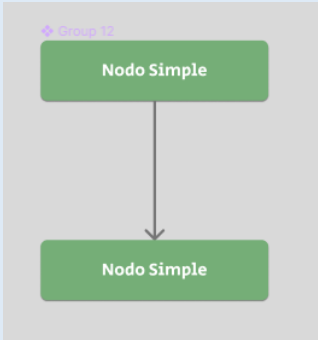
Introducción

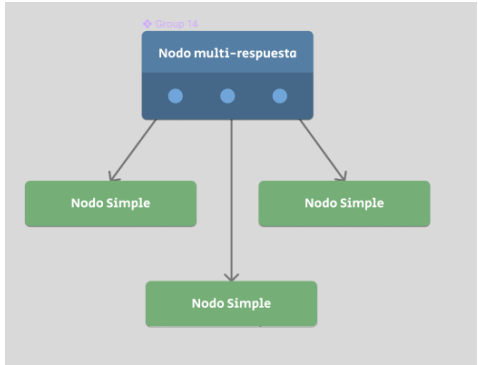
La herramienta propuesta servirá para crear sistemas de diálogos en el motor gráfico de Unity usando un sistema de Nodos y conexiones visuales. Se proporcionará al usuario nodos simples y nodos con múltiple respuestas. El sistema podrá cargar y guardar datos de manera permanente dentro del proyecto de Unity. Usará una DLL para conectar el backend del modelo de datos al frontend de la interfaz de Unity. Los requisitos, características y pertinentes diagramas han sido añadidos a continuación.

Requisitos Funcionales

Requisitos Funcionales - Sistema de diálogos			
Número	Requerimiento	Descripción	Prioridad
RF1	Facilidad de uso	El usuario podrá abrir la herramienta desde su proyecto de Unity sin necesidad de aplicaciones externas.	Alta
RF2	Funcionar correctamente sin internet	Se podrá usar la herramienta sin necesidad de conexión a internet.	Alta
RF3	Creación de gráficos de diálogos	Se podrán crear archivos llamados diálogos que guarden de manera permanente la información del diálogo de un NPC, podrán ser cargados y guardados desde dentro del propio motor de Unity. Se les podrá dar un nombre concreto.	Alta
RF4	Permanencia en memoria	Al cerrar el proyecto los diálogos se guardarán de manera permanente en la memoria del ordenador del usuario y cuando se reabra el proyecto se volverán a cargar, manteniendo su información de manera permanente.	Baja

RF5	Representación de los diálogos	Los sistemas de diálogos estarán representados en formato de árbol de nodos, con cada nodo representando un estado y conteniendo una información concreta.	Alta
RF6	Inicio y final de los diálogos	Los sistemas de diálogos tendrán siempre al menos un nodo de entrada y uno de salida, marcando el inicio y el final de una conversación. Toda conversación comenzará por el nodo de inicio y terminará por uno de los nodos finales. Se proporcionará al usuario con el primer nodo que marcará la entrada al resto.	Alta
RF7	Nodos iniciales	Los nodos pueden ser de inicio si su puerto de conexión de input está vacío, lo cual los caracteriza como nodos iniciales. Son importantes para luego saber desde donde comenzar un diálogo.	Alta
RF8	Creación de nuevos nodos	Se podrán crear nuevos componentes como nodos dentro de un diálogo usando un sistema de ventanas emergentes al pulsar espacio donde podrás escoger la categoría del nuevo nodo, se creará en la posición del ratón de ese momento. También se podrán crear nuevos nodos mediante otra ventana emergente usando el click derecho .	Media
RF9	Destrucción de nodos	Si un usuario selecciona un nodo y pulsa la tecla de suprimir, este nodo se eliminará, borrando todas las conexiones que tenía previamente.	Alta
RF10	Destrucción de enlaces	Si un usuario selecciona un enlace entre nodos y pulsa la tecla de suprimir este enlace	Alta

		se eliminará, en ese momento el nodo inferior quedará desconectado y no habrá acceso a él desde el árbol principal.	
RF11	Selección de nodos	Un usuario deberá de poder seleccionar un nodo colocando el ratón sobre él y pulsando el click izquierdo. Cuando el usuario seleccione el nodo se le mostrará una ventana lateral que tendrá todos los detalles sobre el nodo en base a su tipo.	Alta
RF12	Nodos simples	<p>Los nodos simples tendrán una posible respuesta, estos guardarán la información que van a mostrar ellos por pantalla y una dirección al siguiente nodo al que pasar cuando el jugador acabe el diálogo actual.</p> 	Alta
RF13	Nodos Condicionales	Los nodos condicionales no contendrán información sobre diálogo, servirán como directores de flujo, pudiendo añadir una posible condición al diálogo con dos posibles salidas en base a la respuesta (true o false). Guardará una referencia a una función que retorne bool y dos posibles salidas a dos nodos en base al resultado de la función.	2º Versión
RF14	Nodos con eventos	Todos los nodos podrán tener una función para invocar un método de Unity al finalizar	2º Versión

		su diálogo, funcionarán como los eventos OnClick de los botones	
RF15	Nodos multi-respuesta	<p>Los nodos multi-respuesta tendrán tantas posibles respuestas como el usuario indique, con un mínimo de una.</p>  <p>En los nodos multi-respuesta se podrá añadir un texto de salida por respuesta y añadir o quitar opciones usando botones dentro del nodo.</p>	Alta
RF16	Unión de nodos	Los nodos se unirán automáticamente por medio de líneas, siendo el comienzo un nodo y el final otro.	Alta
RF17	Persistencia de datos	El usuario podrá guardar el conjunto de nodos de manera permanente en su memoria local usando un botón de guardar que se encontrará en la barra superior de la ventana. Se podrá especificar el nombre del archivo en el cual se guardarán los datos.	Media
RF18	Carga de datos	El usuario podrá cargar sistemas de nodos desde su memoria, abriendo un archivo vacío y usando el botón de cargar que se encontrará en la barra superior de la ventana. Se podrá especificar el archivo en memoria del cual se cargarán los datos.	Baja

RF19	Consola	El usuario recibirá feedback por parte del programa por medio de la consola del proyecto de Unity donde se especificarán si se han encontrado problemas en algún punto del uso de la herramienta como carga no completada, archivo corrupto, no se pudo acceder al archivo...	Baja
RF20	Guardado automático	Cada minuto de uso se guardarán automáticamente los datos del sistema en memoria si el archivo ha sido guardado previamente .	2º Versión
RF21	Grupos de nodos	El usuario podrá crear conjuntos de grupos de nodos para mejor organización del proyecto, esos grupos contendrán un nombre y se podrán ver al ser resaltados como ventanas con algo más de contraste del fondo	Media
RF22	Nodos duplicados	Dos nodos no podrán tener el mismo nombre, si dos nodos iguales tienen el mismo nombre su color cambiará ligeramente para especificar cuales 2 tienen el mismo nombre, uno de los dos tendrá que ser cambiado. Esto no se aplicará si por ejemplo uno de los nodos duplicados está dentro de un grupo.	Media
RF23	Grupos Duplicados	Dos grupos no podrán tener el mismo nombre, si dos grupos iguales tienen el mismo nombre su color cambiará ligeramente para especificar cuales 2 tienen el mismo nombre, uno de los dos tendrá que ser cambiado.	Media

RF24	Botón de Limpiar gráfico	El gráfico de nodos podrá ser limpiado usando un botón que se mostrará en la parte superior de la pantalla junto al de guardar y cargar. Este botón eliminará todos los elementos visuales que contenga el gráfico en ese momento, sin guardarlos anteriormente.	Media
RF25	Minimapa	Se le mostrará al usuario un minimapa que le indique la posición de los nodos dentro del gráfico global a modo de visión general de la ventana. Este minimapa estará situado en la esquina superior izquierda.	Baja
RF26	Botón Minimapa	Se mostrará un botón al lado de cargar, guardar o limpiar que activará o desactivará el minimapa con un click.	Baja
RF27	Nombre del Grafo	El usuario podrá añadir un nombre al gráfico de nodos que será con el que se guardará en las carpetas. El nombre por defecto será "DialogueFileName".	Media
RF28	Centrar gráfico	Usando la tecla F el usuario podrá centrar la vista del gráfico, haciendo que se haga un zoom generalizado de todos los componentes de la ventana.	Baja
RF29	Ventana de creación de nuevos nodos	Se creará una ventana usando el espacio que le dé al usuario la posibilidad de crear nuevos elementos para el gráfico de la misma manera que se puede usar el click derecho para usar un menú emergente. Esta ventana añadirá también un buscador por nombre de los componentes que el usuario puede añadir.	Baja

RF30	Movimiento de la ventana de gráficos	La ventana donde el usuario crea los gráficos ha de poder ser movida alrededor del motor, modificado su tamaño de manera que no afecte al gráfico general y colocada en las posiciones por defecto que el motor permite a las ventanas colocarse.	Media
-------------	--------------------------------------	---	-------

Requisitos Técnicos

Requisitos Técnicos - Sistema de diálogos			
Número	Requerimiento	Descripción	Prioridad
RT1	Sistema de uso	El sistema se desarrollará para Windows 10 que posean arquitectura de 64 bits.	Alta
RT2	Motor gráfico	La herramienta se desarrollará para el motor gráfico de Unity, versiones 2021 en adelante.	Alta
RT3	Lenguaje Front-End	El lenguaje del front-end y scripting será C# complementado con el scripting de Unity.	Alta
RT4	Lenguaje Back-End	El back-end de esta herramienta será programada usando C++ y se vinculará con el front-end por medio del uso de un dll (Dinamyc-Link Library o Biblioteca de enlace dinámico)	Alta
RT5	Conexión Front-Back	Se conectarán Back-end y Front-end usando una DLL la cual importará a Unity los métodos necesarios para el traspaso de datos simples entre visuales y controladores.	Alta
RT6	Exportación	La herramienta una vez finalizada se podrá exportar en un archivo comprimido donde irán la DLL y un Unity Package con el contenido del visor, junto con un manual de instrucciones de cómo implementarlo a tu proyecto.	Alta
RT7	Manual de instrucciones	Dentro de la herramienta se le proporcionará al usuario un manual de instrucciones donde se le explique cómo usar la herramienta, sus puntos clave,	Alta

		ejemplos de uso y como enlazar la herramienta con código propio. Se implementará en español.	
RT8	Idioma de la programación	El idioma de programación va a ser inglés estándar, para poder favorecer la comercialización o expansión de la herramienta.	Alta
RT9	Guardado de datos	Para guardar un archivo se abrirá una ventana de explorador de archivos donde el usuario podrá especificar la ruta donde se guardará el archivo junto con un nombre. El formato usado para el guardado será de txt. Contendrá por un lado la información relevante del sistema de nodos front-end y la información del back-end. Al ser txt se incluirá una marca al comienzo del archivo para especificar que es compatible con la herramienta. De no ser así se notificará al usuario con un mensaje por pantalla.	Baja
RT10	Carga de datos	Para la carga de datos el usuario tendrá que especificar que archivo de tipo txt quiere cargar, se leerá y se comprobará si contiene la marca del sistema. Una vez comprobado se pasará al script encargado de parsear su información y cargar primero los datos del back-end y posteriormente los del front-end de Unity.	Baja
RT11	Marca del sistema	Para la marca inicial se usará el formato: "M-P-T-O-O-L" indicando que todo archivo con estas primeras líneas será compatible con la herramienta.	Baja
RT12	Documentación del código	Todos los archivos de cabecera ya sean front-end o back-end han de ir acompañados de un comentario de cabecera que especifique: 1- Autor del archivo 2- Fecha de última modificación 3- Breve descripción del script 4- Copyright 5- Correo de contacto	Alta
RT13	Estándar de nombrado de variables y funciones	Para el nombrado de variables tanto en C# como en C++ se usarán camelCase para variables y snake_case para funciones.	Alta
RT14	Nombrado de funciones para exportación	Todas las funciones que su función sea ser exportadas en la DLL han de ser nombradas poniendo primero la clase a la que pertenecen y posteriormente su función: Game_controller_clear_view()	Media

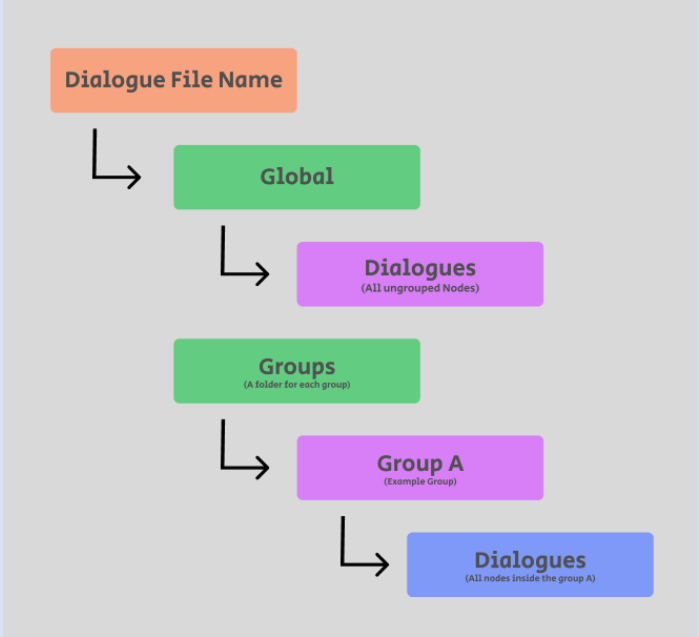
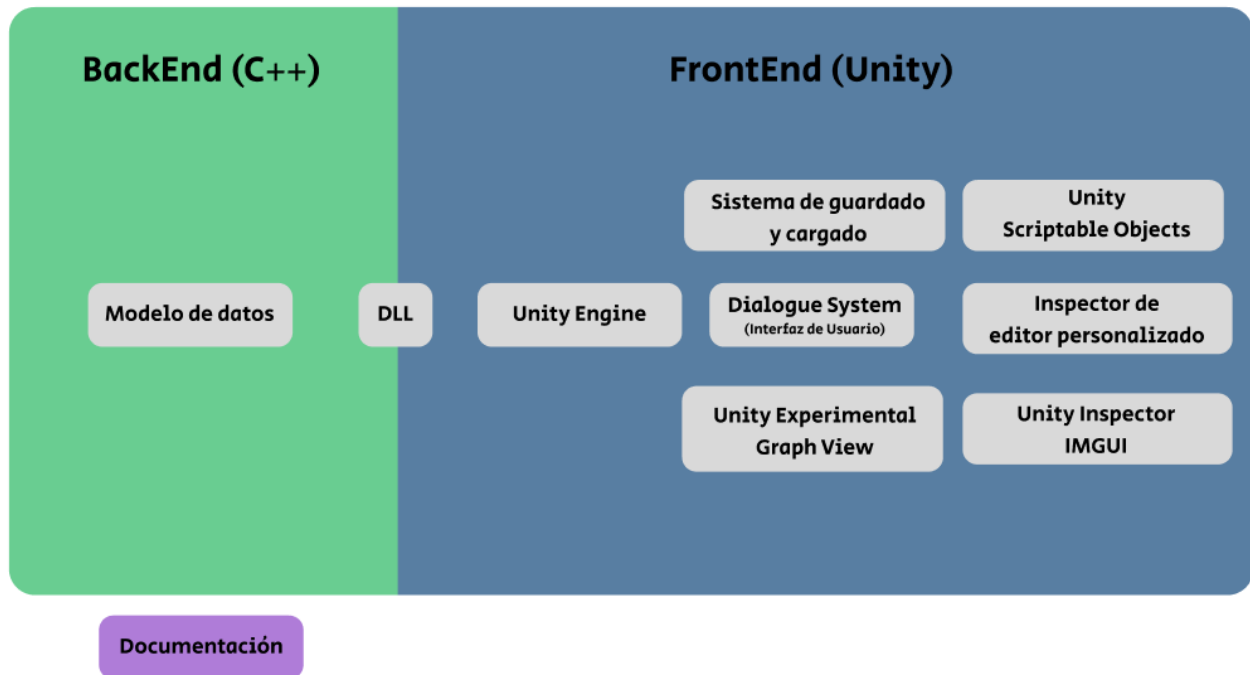
RT15	API para la programación visual	Se usarán las librerías de Unity de Visual Scripting para la creación del front-end visual en Unity, usando el mismo sistema que Shader Graph de nodos para los diálogos.	Alta
RT16	Identificación de nodos	Los nodos tendrán un id único que será su nombre, de esta manera se indicará al usuario cuando dos nodos tengan el mismo nombre que no puede ser, solo se aplica si los dos nodos están en el mismo espacio, si uno está dentro de un grupo y el otro en otro distinto no pasa nada.	Media
RT17	Identificación de grupos	Los grupos tendrán un id único que será su nombre, de esta manera se indicará al usuario cuando dos grupos tengan el mismo nombre que no puede ser.	Media
RT18	Modelo de datos para el guardado.	Para el sistema de guardado de datos se usarán Scriptable Objects de Unity. Y el tipo de dato final de estilo .asset.	Alta
RT19	Organización de nodos dentro del guardado	Los nodos se guardarán en dos grupos en base a su condición. Por un lado los nodos que no estén dentro de grupos se guardarán todos juntos en una lista de nodos desagrupados, los nodos que estén dentro de grupos se guardarán como hijos de ese grupo por otro lado.	Media
RT20	Sistema de carpetas en el guardado.	<p>Se usará el siguiente sistema para la organización de las carpetas a la hora de guardar los datos en memoria permanente:</p>  <pre> graph TD A[Dialogue File Name] --> B[Global] B --> C["Dialogues (All ungrouped Nodes)"] C --> D["Groups (A folder for each group)"] D --> E["Group A (Example Group)"] E --> F["Dialogues (All nodes inside the group A)"] </pre>	Media

Diagrama de componentes



Se ha dividido el sistema en dos apartados, por un lado el backend, el cual será la parte del proyecto que esté destinada a usar el lenguaje de C++ y la cual su principal componente será mantener el modelo de datos.

El modelo de datos se encargará de recibir, almacenar y devolver información relevante para la interfaz de usuario, de esta manera su contraparte solo tiene que preocuparse de manejar la experiencia del usuario y no la información recibida.

Todos los componentes se detallarán más adelante en el diagrama de clases.

La documentación será todo lo referido a la explicación del uso y funcionamiento de la herramienta, este documento sirve en parte como ella junto con los comentarios del código y el documento de "Read Me" de instalación.

Para unir el backend y el frontend existe la DLL, una librería de enlace dinámico que compila el modelo de datos y permite al frontend interactuar con él, enviando y recibiendo datos de manera continua. Es un nexo entre los dos apartados.

Para el apartado de frontend se divide en varios componentes, por un lado tenemos los dados ya de manera directa por el motor de Unity, usado para crear la herramienta, y aquellos creados internamente.

Por el lado de Unity tenemos los siguientes apartados:

- **Unity Engine:** La librería principal del motor gráfico, usada para MonoBehaviours e instanciación de objetos en la escena.
- **Unity Experimental Graphview:** La librería usada para poder acceder a todos los elementos visuales mostrados en la herramienta, da al creador la posibilidad de

tener elementos pre-hechos ya como ventanas, toolbars, botones, nodos, conexiones, minimapas...

- **Unity Scriptable Objects:** La librería especial de Unity para crear archivos de datos que se pueden guardar como archivos de proyecto pudiendo usarlos para almacenar memoria de manera permanente.
- **Unity Editor IMGUI (Immediate Mode GUI):** La librería que permite modificar editores de monobehaviours de manera personalizada, añadiendo campos y parámetros propios. IMGUI es el sistema inmediato de Unity para extender editores. Se ha usado este método en vez de UI Elements (Variante más actual y que ressemble un sistema de HTML más visual para la creación de editores) ya que la segunda opción aun siendo algo más moderna todavía no está del todo bien configurada y no permitía de manera simple la modificación de partes del editor que se requerían para este apartado.

Por el lado de la herramienta:

- **Dialogue System:** El componente principal de la herramienta, crea la ventana, los nodos, las uniones, permite la modificación de los parámetros, agrupar los nodos, moverte por la ventana...
- **Sistema de guardado y cargado:** Permite al usuario desde la herramienta guardar y cargar gráficos completos, usa Scriptable Objects para generar archivos de datos que mantienen la información de todos los nodos del gráfico y su información.
- **Inspector personalizado:** Para que el usuario pueda hacer uso de la herramienta dentro de proyectos de Unity se dota de una clase con monobehaviour que permite añadir un gráfico en modo de SO y navegar por él. La forma en la que se navega se deja a disposición del usuario para que la complete él de la manera que mejor complemente su proyecto. Solo se otorga el inspector que ayudará al usuario a navegar mejor los gráficos.

Modelo de datos

La herramienta propuesta permite la creación de diálogos utilizando un sistema basado en nodos para permitir la creación de conversaciones dinámicas y fáciles de diseñar. Cada nodo representa una línea de diálogo y las respuestas permiten al diseñador alternar el curso del diálogo.

Se parte de una clase base de Node, que permite acceder y modificar sus parámetros de manera interna. Contiene la información básica de un nodo, ID, Texto, Nombre y Tipo. Luego se emplea una clase "puente" entre C++ y C#, que exporta las funciones a la DLL para que se puedan usar desde Unity, contiene funciones que hacen referencia a las de la clase Node y aceptan parámetros del mismo tipo.

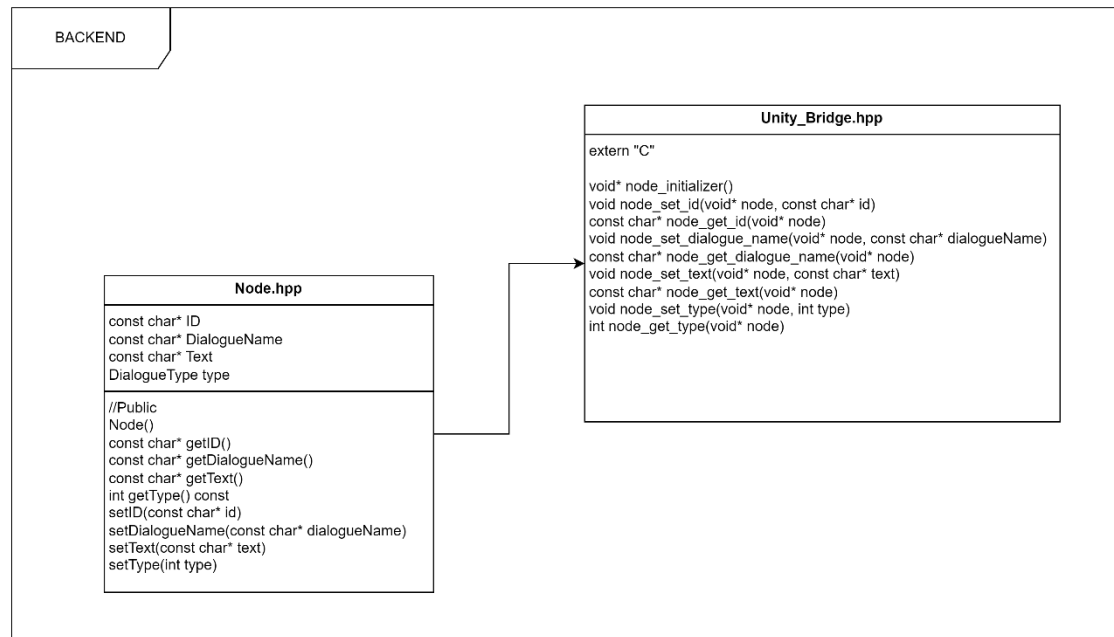


Diagrama de clases

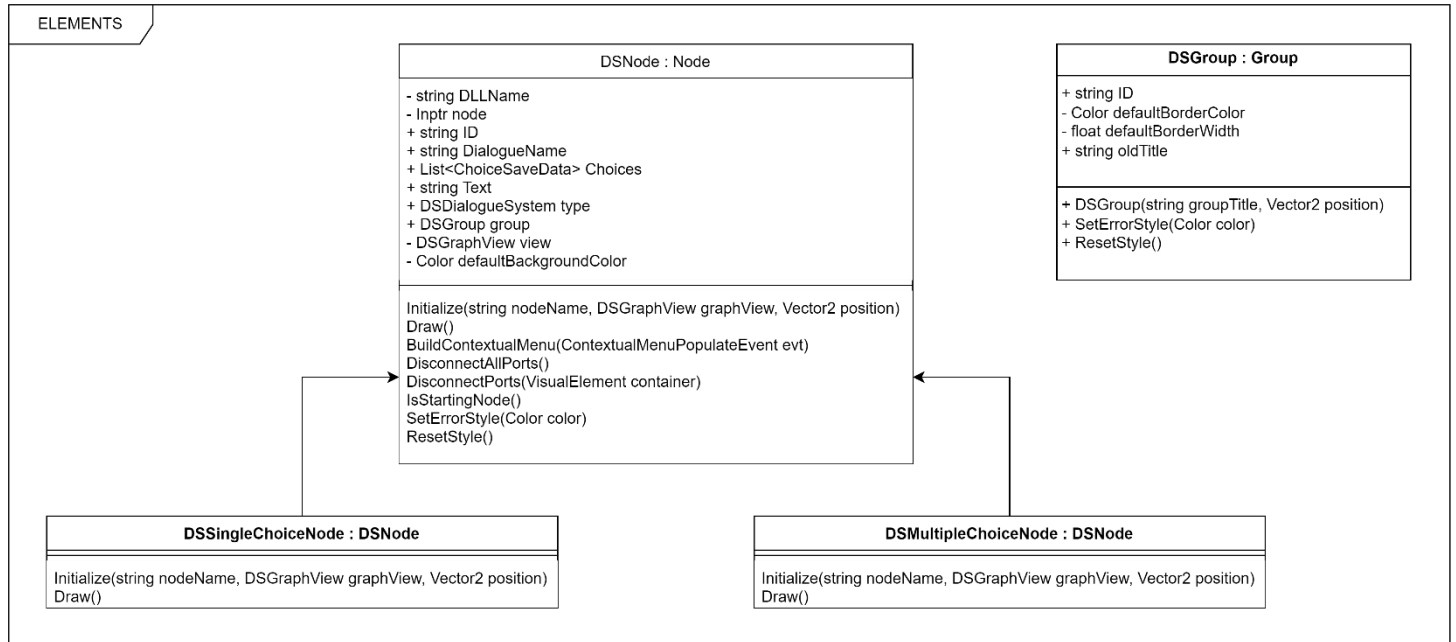
En el diagrama de clases se comentarán las clases propias de la herramienta, aquellos componentes que son propios de Unity, al ya existir suficiente documentación, no se consideran necesarios de volver a explicar.

Dialogue System

El sistema de diálogos está compuesto por una serie de sub-componentes o grupos de código que contienen lógica propia.

Elementos

Esto serán todo aquello que podamos ver en el gráfico, bien sean nodos o grupos. Los nodos parten de una clase base llamada **DSNode** y los hijos hacen override a los métodos del padre de inicialización y dibujado. El otro componente es el **DSGroup**, el cual contiene la información necesaria para guardar nodos dentro de él.

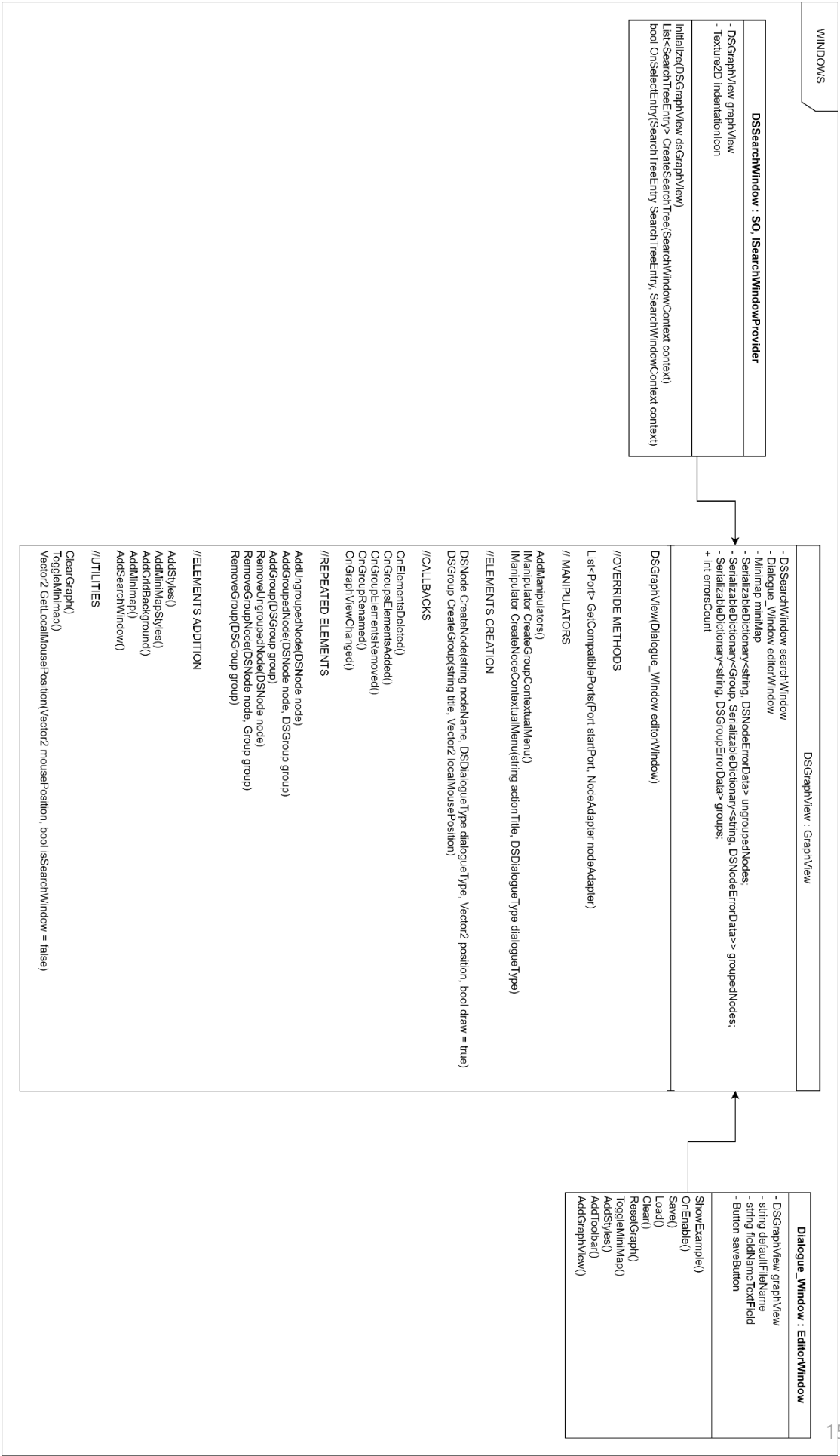


Ventanas

Las ventanas con aquellos componentes que nos permiten ver lo que está ocurriendo dentro de la herramienta, ya sea la ventana extensible que aparece cuando abrimos la herramienta por primera vez o la ventana "interior" que contiene toda la información del gráfico. La clase más importante dentro del sistema es **DSGraphView**, es la que se encarga de manejar todo lo que ocurre dentro de la herramienta.

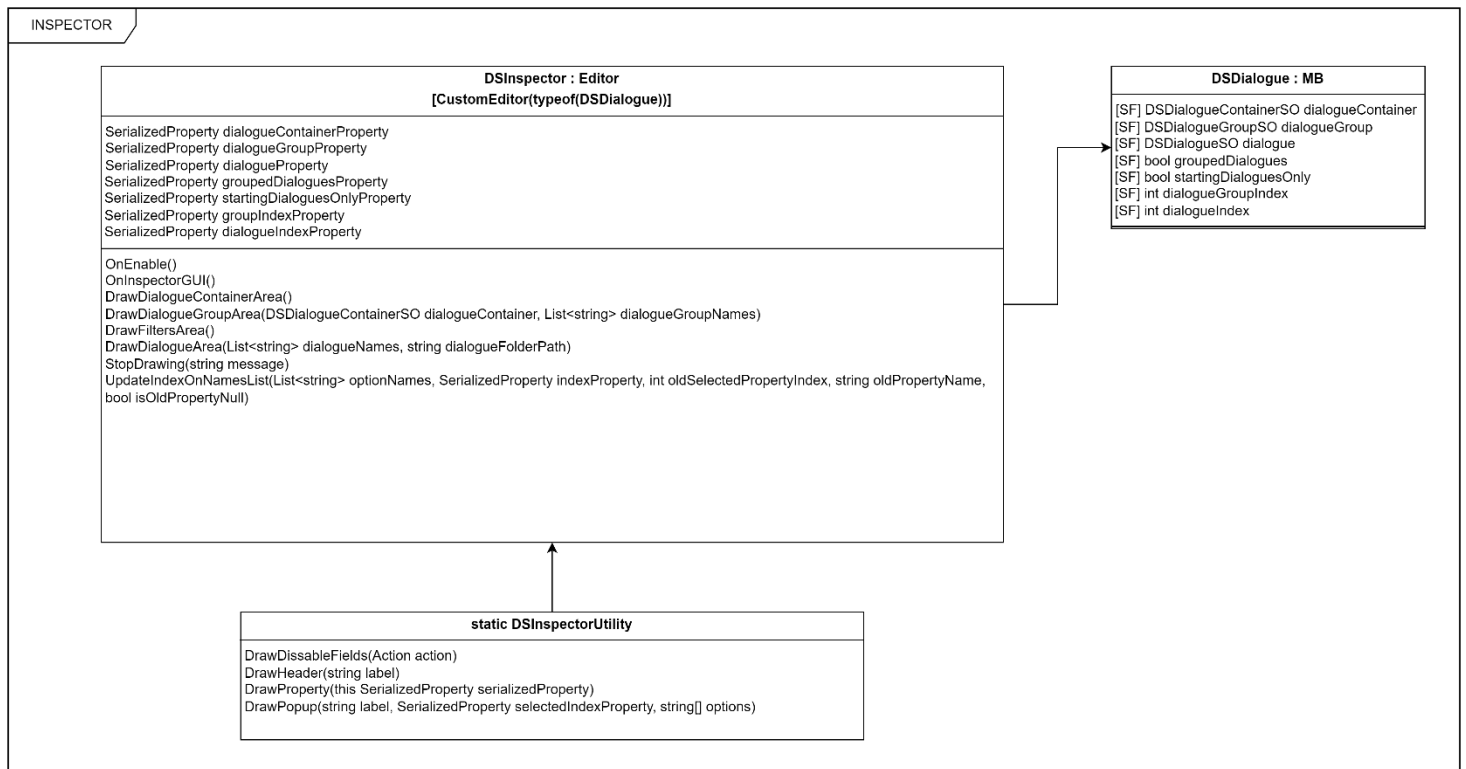
DSGraphView hereda de la clase base de **GraphView**, proporcionada por la librería de Unity Experimental **GraphView**. Da la posibilidad de crear espacios donde poder instanciar elementos visuales. Desde esta ventana podemos añadir nodos o grupos al gráfico con ventanas secundarias como la de búsqueda, **DSSearchWindow**, la cual hace la función de herramienta de comandos al ser esto una herramienta de editor y no de aplicación, ha sido lo más cercano que he podido encontrar.

Permite esta clase añadir manipuladores al gráfico y callbacks. Los manipuladores son comportamientos que un gráfico puede tener, zooms, drag and drop, selección múltiple... Y los callbacks sirven para poder acceder a métodos concretos como **OnElementDelete**, **OnGraphChanged**, **OnElementsDeleted**... De esta manera podemos configurar esos comportamientos con nuestros propios métodos y actualizar correctamente la información según vaya ocurriendo y sin necesidad de generar referencias múltiples y cruzadas.



Inspector Personalizado

Otro de los componentes principales de la herramienta es el inspector personalizado. Se ha dejado un como base un inspector personalizado al usuario para que una vez tenga el grafico listo y guardado pueda usar ese Scriptable Object para generar conversaciones dentro del juego. El inspector genera un editor personalizado para la clase **DSDialogue**, la cual está vacía de métodos, solo contiene las variables necesarias para poder generar el inspector.



Sistema de Guardado y Cargado

Como se comentó anteriormente la herramienta provee un sistema de guardado y cargado de gráficos, los cuales se almacenan en la memoria permanente de Unity usando Scriptable Objects. El sistema se gestiona desde la clase de **DSIOUtility**, una clase estática que permite recorrer todos los elementos del gráfico, categorizarlos y crear Assets con su información en base a su tipo para el guardado. Para la carga se selecciona el archivo principal SO que guarda todos los datos y se recorre, creando elementos en base a sus componentes.

Los componentes usan SO adaptables a la información que tienen que guardar, por lo que se crean este tipo de componentes que son los que almacena el SO general del gráfico.

La estructura de carpetas ya se define dentro de los requisitos técnicos como se han de crear las carpetas y los contenidos de cada una de ellas.

SAVING/LOADING SYSTEM

