Universidad Rey Juan Carlos

ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

Curso Académico 2010/2011

Proyecto de Fin de Carrera

INTERFAZ GRÁFICA MULTIPLATAFORMA PARA LA SIMULACIÓN DE ECUACIONES FÍSICAS

Autor: Daniel Isaac Khan Ramiro
Tutores: Alexandre Wagemakers / Juan Sabuco

ÍNDICE

TABLA DE FIGURAS

1. INTRODUCCIÓN

Dynamics es un software que consiste en un conjunto de herramientas, paquetes y clases desarrolladas para ayudar a visualizar el comportamiento de los sistemas dinámicos mediante el uso de gráficos ilustrados. El programa de ordenador Dynamics ha sido creado para que lo puedan utilizar tanto gente experta en la materia como principiantes, ayudando a estos últimos a empezar rápidamente con la exploración dinámica de sistemas con una amplia gama de técnicas interactivas.

Estas herramientas son elementales en lo que respecta a alumnos universitarios que estudian dicho comportamiento.

Debido a la gran utilidad que ofrece este software para explorar y analizar la dinámica de sistemas de baja dimensión de ecuaciones diferenciales y mapas, surge la idea de facilitar un poco este trabajo con la creación de una interfaz gráfica intuitiva para el usuario y de fácil manejo que permita la generación de gráficos y diagramas a partir de una cierta ecuación y parámetros dados.

El objetivo principal de este proyecto es proporcionar un entorno visual sencillo para permitir la comunicación con el sistema Dynamics. A grandes rasgos una **GUI** (Graphical User Interface) es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones en la interfaz.

En este trabajo analizaremos los mecanismos básicos que ofrece la aplicación Qt Creator para el desarrollo de aplicaciones multiplataforma a través de las bibliotecas Qt. En la solución que proponemos se describen un conjunto de clases implementadas para la construcción de la interfaz de usuario.

1.1. Presentación del problema

El problema que trata de solucionar este proyecto es la ausencia de una interfaz gráfica sencilla e intuitiva para el usuario al ejecutar el programa Dynamics. En este sentido, cualquier operación física que pudiera hacer un usuario sobre Dynamics ahora será mucho más sencilla de hacer a través de la interfaz gráfica creada.

Por este motivo, el primer problema que se nos plantea es generar un diseño sobre la antigua interfaz de Dynamics y convertirlo en un formato más cómodo, creativo, potente, rápido e intuitivo. Una vez terminada la fase anterior, el segundo problema que se nos presenta es el hecho de que dicha interfaz se pueda ejecutar tanto en sistemas GNU/Linux, como Mac OS X y Windows.

1.2. Objetivos

Por los motivos expuestos anteriormente, los objetivos concretos que pretende alcanzar este proyecto son los siguientes de cara a la interfaz de usuario:

- Reunir y analizar la información del usuario.
- Diseñar la interfaz de usuario.
- Construir la interfaz de usuario.
- Validar la interfaz de usuario.

1.3. Método de trabajo

La metodología de trabajo para llevar a cabo este trabajo es la que se describe:

- **Fase de estudio previo:** Estudio de aspectos relacionados con la información que se trata y la forma con la que el usuario interactuará con el medio.

- **Fase de modelado de la interfaz:** Identificación de las partes de la interfaz y su posterior implementación.

- **Fase de implementación y desarrollo:** Durante esta fase se lleva a cabo todo el análisis previo a la construcción de la interfaz gráfica así como la aplicación de las herramientas que ofrece el Qt Creator y el diseño de las clases.

- **Pruebas de la aplicación:** Una vez terminadas todas las fases anteriores se procede al testeo de la aplicación para comprobar su correcto funcionamiento y en determinados casos, captar posibles fallos que pueda generar la aplicación.

1.4. Introducción a las GUI

En informática, una interfaz gráfica de usuario (Graphical User Interface) representa el medio por el cual el usuario podrá comunicarse con el ordenador o computadora. Una GUI representa la información y las acciones disponibles para el usuario a través de iconos gráficos e indicadores visuales, en contraposición a las interfaces basadas en texto, con tipo de comandos o etiquetas de navegación de texto.

Existen varios tipos de interfaces graficas de usuario:

Zooming User Interface

Es un entorno donde los usuarios pueden cambiar la escala de la zona para ver más detalles o menos, y navegar a través de diferentes documentos. Los elementos de información aparecen directamente en un escritorio virtual infinito, en lugar de en ventanas. Los usuarios pueden desplazarse por la superficie virtual en dos dimensiones y hacer zoom sobre los objetos según su interés.

Figura 1: ZoomingUser Seadragon

Toushscreen User Interface

Este tipo de interfaces son creadas para dispositivos con pantallas táctiles. Una pantalla táctil es una pantalla electrónica visual que puede detectar la presencia y la ubicación de un toque en el área de visualización. Este tipo de pantallas es común en dispositivos como tabletas y teléfonos inteligentes. Nos podemos encontrar este tipo de interfaces en muchos restaurantes alrededor del mundo y en tiendas autoservicio.

- Permite a un usuario interactuar directamente con lo que se muestra en pantalla.
- No se necesita de ningún dispositivo secundario para poder interactuar con la pantalla

También juegan un papel destacado en el diseño de los aparatos digitales el asistente personal digital (PDA), los dispositivos de navegación por satélite y los teléfonos móviles y videojuegos.

Figura 2: Toushscreen User Interface 1

Figura 3: Toushscreen User Interface 2

Interfaz natural de usuario

Un NUI se basa en que un usuario pueda pasar de forma rápida la transición desde el estado principiante al experto. Mientras que las interfaces de uso común requieren cierto tiempo de aprendizaje, en las NUI se facilita este proceso a través de su diseño que le da al usuario la sensación de formar parte de la interfaz interactuando de una forma natural sobre ella.

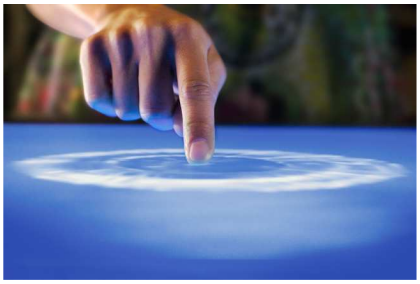


Figura 4: Natural User Interface

1.4.1. Importancia de la HCI (*Human-Computer Interaction*)

La interacción hombre-computadora comprende todo lo que ocurre cuando un hombre y un sistema computarizado realizan tareas juntas. Esto involucra tanto al rol de programador como al rol de usuario final, en un diálogo hombre-computadora en el que media una interfaz.

La interacción entre humano-computadora estudia el comportamiento del humano y la máquina en conjunto. Se basa en analizar tanto el conocimiento de la maquina como la del humano. En el lado de la máquina son redundantes las técnicas de computación gráfica, sistemas operativos, lenguajes de programación y entornos de desarrollo. Sin embargo, en el lado del humano, son relevantes la teoría de la comunicación, las disciplinas de diseño gráfico e industrial, la lingüística, las ciencias sociales, la psicología cognitiva y los factores humanos. Debido a la naturaleza multidisciplinaria de la HCI, personas con diferentes orígenes contribuyen a su éxito. La HCI también puede referirse como interacción hombre-máquina (Man-Machine Interaction MMI) o la interacción humano-computadora (Computer-Human Interaction CHI).

Las interacciones están relacionadas con una diversidad de aspectos, dentro de los que se incluyen: la realización de tareas por hombres y máquinas, la estructura de la comunicación hombre-máquina, las interacciones organizacionales y sociales, las capacidades humanas incluyendo el aprendizaje, los algoritmos y programación de la propia interfaz, las restricciones de la propia tecnología para el diseño y construcción de las interfaces.

La interfaz comprende a los canales de información que permiten que usuario y computadora se comuniquen. Toda interacción implica una relación emisor-receptor

donde cada una de las partes tiene una imagen de la otra según la cual estructuran e interpretan los *mensajes.*

Para que esta comunicación sea efectiva debe haber un conocimiento mutuo.

Por un lado el usuario debe conocer el funcionamiento de la interfaz a un nivel operativo, implicando además una situación de aprendizaje y el involucramiento de procesos cognitivos.
Por otra parte la interfaz debe estar diseñada en función de las características de ese usuario o adecuarse a distintos tipos de usuarios. Uno de los caminos para lograr esto es el Modelado de Usuarios.

1.4.2. Requisitos para GUI (Graphical User Interface).

Dynamics carecía de diseño, estructura y facilidad de manejo para el usuario. Por ello, desde el inicio del proyecto se han planteado dos fases importantes:

- Extracción de información necesaria para la creación de la interfaz.
- Conexión con las librerías de Dynamics.

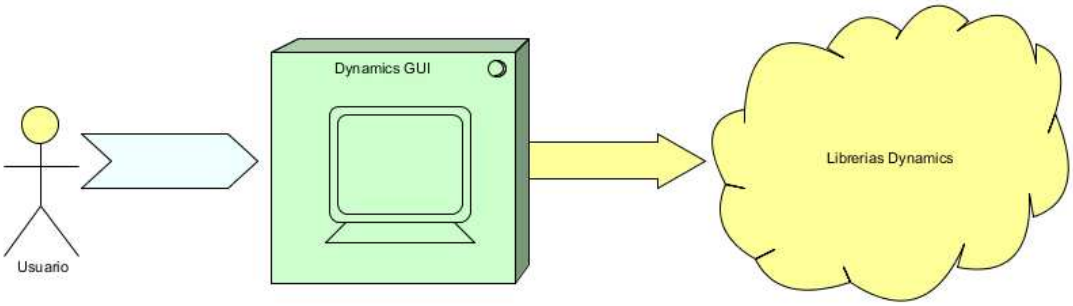


Figura 5: Requisitos Iniciales Dynamics

Las librerías de Dynamics, implementadas en el lenguaje de programación C, contienen toda la funcionalidad de la aplicación.

En un principio se pensó en crear la interfaz y que toda la funcionalidad de las ecuaciones se utilizara accediendo desde ella a las librerías ya implementadas. El problema que surgía era la falta de tiempo para implementar las librerías en C de Dynamics y pasarlas a C++. Por lo tanto, como solución alternativa se ha propuesto la creación de clases que implementen parte de la funcionalidad y que puedan ser reutilizables y actualizables en el futuro sin necesidad de realizar muchos cambios sobre el código.

2. ENTORNO DE DESARROLLO

2.1. Entorno de trabajo

El entorno de desarrollo utilizado para la creación del proyecto ha sido Qt Creator, IDE creado por Trolltech para el desarrollo de aplicaciones creadas a partir de las bibliotecas de Qt.

2.1.1. Qt Creator

Qt Creator es un entorno de desarrollo integrado (IDE) de aplicaciones multiplataforma el cual viene acompañado de un conjunto de herramientas para facilitar su uso. Posee un avanzado editor de código C++ ya que es el lenguaje que utiliza de forma nativa. Al utilizar el lenguaje de programación C++, hemos podido hacer uso del potencial de la programación orientada a objetos.

La siguiente imagen muestra la distribución interna de Qt Creator.

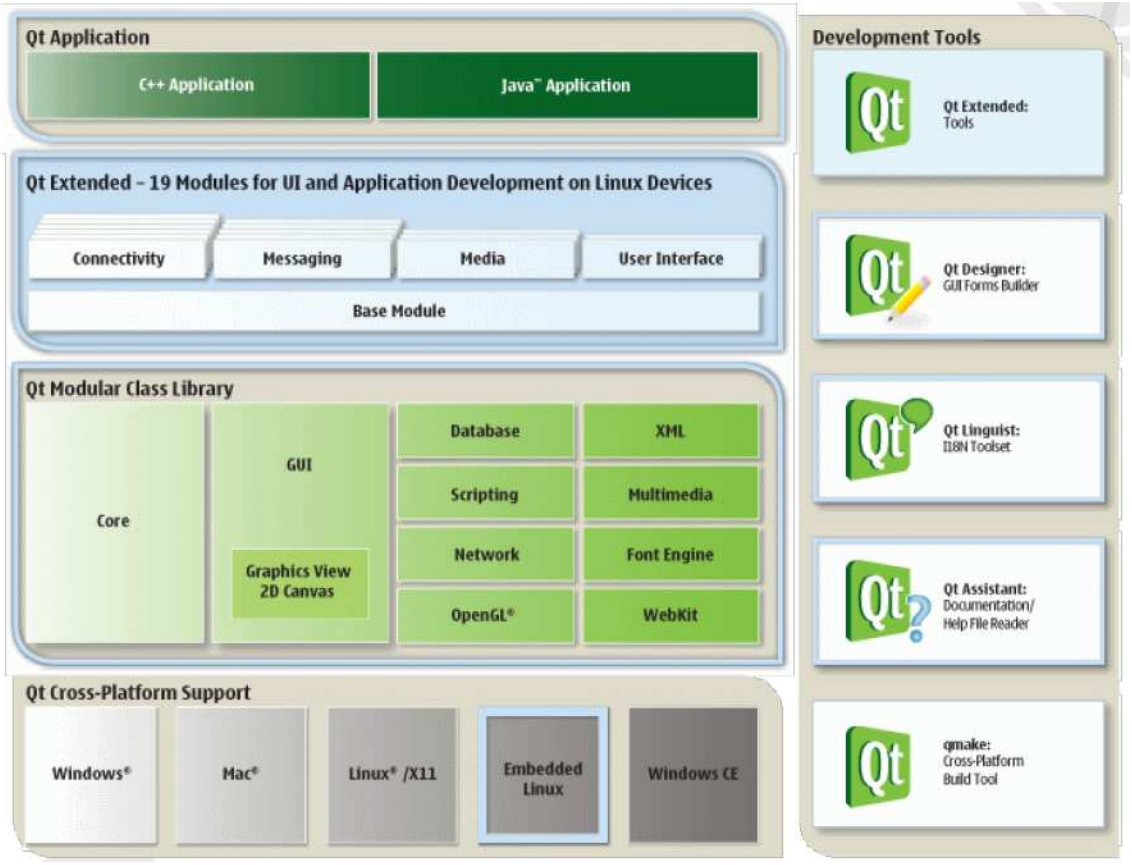


Figura 6: Distribución de paquetes Qt Creator.

- QtCore: Contiene el núcleo no gráfico de Qt.
- QtGui: Colección básica de componentes gráficos.
- QtNetwork: Clases para escribir clientes y servidores TCP/IP.
- QtOpenGL: Facilita el uso de openGL.
- QtScript: Expone las aplicaciones a scripting con un lenguaje ECMAscript.
- QtScriptTools: Es un depurador de QtScript.
- QtSQL: Integración de bases de datos.
- QtSVG: Soporte SVG.
- QtWebKit: Popular motor Web, con Qt.
- QtXml: Soporte básico de Xml.
- QtXmlPatterns: Es un motor de XQuery 1.0 y XPath 2.0 y parcialmente Xslt.
- Phonon: Framework multimedia.
- Qt3Support: Compatibilidad con Qt3.
- Otros: QtDessigner, QtUiTools, QtHelp, QtAssistant, QtTest, QtDBus (solo Unix), y a partir de Qt 4.6 QtOpenVG y QtMultimedia.

Para este proyecto se han utilizado en su mayoría las clases de las librerías QtCore y QtGUI.

2.2. Entorno de ejecución

Qt es una biblioteca multiplataforma, es decir, puede funcionar en diversas plataformas ya sean sistemas tipo Unix con el servidor grafico X Windows System (Linux, BSDs, Unix), para Apple Mac OS X, para sistemas de Microsoft Windows, para Linux Embebido e incluso para dispositivos que utilizan Windows CE.

Dependiendo del entorno donde se haga la compilación, el módulo qmake hará su función y generará los archivos necesarios para que el programa pueda ser ejecutado en el entorno donde se está trabajando sin necesidad de adaptar el código.

3. DESCRIPCIÓN INFORMÁTICA

La fase de análisis del proyecto se ha iniciado con el estudio del funcionamiento de Dynamics en modo consola. Durante este proceso se han separado las etapas necesarias para definir la nueva interfaz en varios planos.

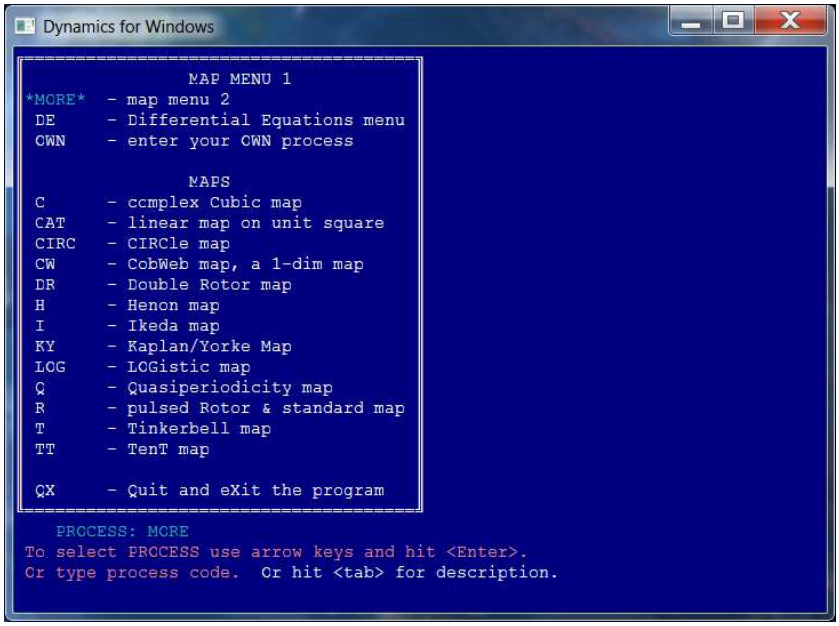


Figura 7: Interfaz en modo texto de Dynamics.

3.1. Etapas de desarrollo

Conforme se ha ido manejando Dynamics se han ido definiendo los requisitos necesarios para que a la hora de implementar nuestra interfaz queden fijados y bien estructurados.

Las etapas a seguir, se consideran en el siguiente diagrama.

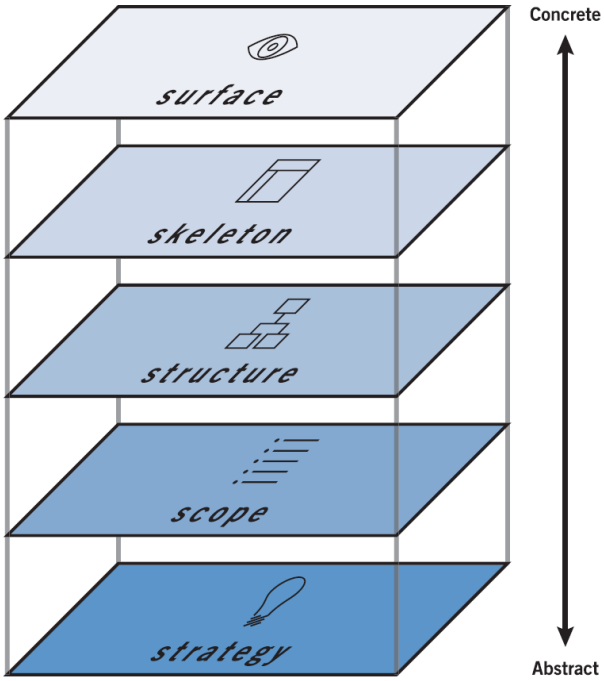
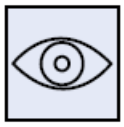


Figura 8: Diagrama de los elementos de la experiencia de usuario.

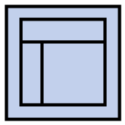
En cada uno de los planos el tema a tratar ha sido mas concreto y menos abstracto. Partiendo del mas lejano y mas abstracto, que fue la idea que queriamos plasmar a partir del Dynamics existente, se ha ido avanzando por cada una de las etapas hasta llegar a la mas elevada y concreta.

Al ir avanzando plano a plano, las decisiones que se han tenido que tomar han sido mas especificas, envolviendo con mas detalle la interfaz final que queriamos conseguir.



El plano de la superficie (Interfaz)

Es el plano más superficial del proyecto. En él es donde el usuario puede interactuar con la interfaz a través de una serie de objetos que atendiendo a una petición realizaran una acción determinada. En esta fase se ha prestado mucha importancia en la presentación de la interfaz. Su diseño y estilo marcan la claridad y una buena primera impresión hacia el usuario final.



El Plano esqueleto

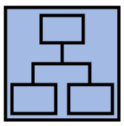
Debajo del plano superficie está el esqueleto de la aplicación. Durante esta etapa se ha diseñado la interfaz:

- Se han reflejado las posiciones que han tomando los elementos gráficos dentro de la interfaz, optimizando su posicionamiento para un rendimiento más óptimo y buscando siempre la eficiencia frente al usuario.

- La forma con la que el usuario actúa en la interfaz. Sobre los elementos de la interfaz el usuario podrá realizar acciones que tendrán una respuesta asociada.

- La información con la que se trabaja (Datos de entrada y salida).

Los principales objetivos a conseguir en esta etapa han sido diseñar el modo de interacción con el usuario y estructurar la arquitectura que compone la información.



El plano estructural

La fase de estructuración de la aplicación ha sido la más abstracta de todas, ya que en ella se ha llevado a cabo el análisis de las clases necesarias para desarrollar la aplicación. La estructura de la aplicación define la dirección en la que las características y las funciones de las clases trabajan juntas.

Los principales objetivos a conseguir en esta etapa han sido diseñar la especificación funcional de Dynamics y analizar los requisitos de contenido.

El plano alcance

El alcance de la aplicación depende en su totalidad del planteamiento elaborado en la fase de estrategia. En esta etapa se han especificado con más detalle los objetivos propuestos y las formas de poder conseguirlos.

Plano estratégico

La fase estratégica ha sido la base de todo el proyecto. En ella se han planteado los objetivos a conseguir y se han preparado las etapas por las que debe pasar el proyecto.

3.2. Análisis

Durante el inicio de esta etapa se ha prestado mucha atención a la distribución de la aplicación en cuanto a ventanas, como se muestran y que acciones puede realizar sobre ellas el propio usuario.

Analizando la estructura de la interfaz ya creada para Dynamics se ha ido descomponiendo y transformando cada vez más en un modelo donde la claridad de las ventanas (que no fuera difícil encontrar las herramientas), la simplicidad en el manejo de la aplicación y la eficiencia deben predominar.

3.2.1. Casos de uso

Se ha comenzado definiendo los posibles casos de uso que puede realizar el usuario. De esta forma se ha obtenido la información sobre la funcionalidad que requiere nuestro sistema.

El siguiente diagrama describe la funcionalidad del Sistema Dynamics de forma simplificada. Los casos de uso representados describen las posibles opciones que tendrá el usuario para interactuar con el sistema.

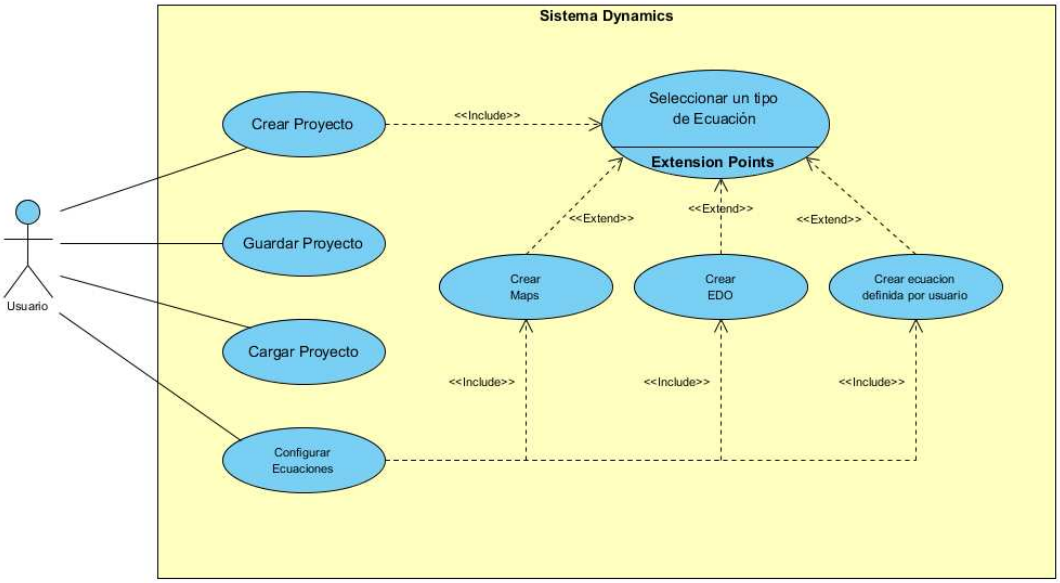


Figura 9: Diagrama de procesos Dynamics.

Como actor del sistema se puede definir al usuario que podrá realizar algún tipo de interacción con la aplicación. En nuestro caso, ese usuario seguramente sea una persona con conocimientos de física y que podrá sacar el máximo partido a las funcionalidades de la aplicación.

A continuación se detallan los flujos de eventos que tendran lugar en cada uno de los casos de uso, teniendo en cuenta los siguientes puntos:

- Como comienza y termina el caso de uso.
- Como interactua con los actores.
- Objetos que se intercambian.

Flujo de Eventos
Nombre: Crear Proyecto
Descripción: Permite crear un nuevo proyecto.
Actores: Usuario de Dynamics.
Camino básico del caso de uso "Crear Proyecto" 1. El usuario pulsa sobre el botón para crear un nuevo proyecto. 2. El sistema muestra un asistente para que el usuario pueda seleccionar una ecuación. 3. El usuario selecciona un opción y pulsa "Aceptar" 4. El sistema comprueba que se han seleccionado bien las opciones 5. El sistema crea el nuevo proyecto.
Flujo Alternativo: 3. El usuario cancela la operación. 4. El sistema comprueba la opción seleccionada, si hay varias, el sistema notificará al usuario que los datos son erróneos y se deberá reiniciar el caso de uso.

Flujo de Eventos
Nombre: Guardar Proyecto
Descripción: Permite guardar la configuración actual del proyecto. De esta forma la situación de las ventanas, configuración de la ecuación, etc. queda registrado en el archivo de salvado.
Actores: Usuario de Dynamics.
Camino básico del caso de uso "Guardar Proyecto" 1. El usuario pulsa sobre el botón para guardar. 2. El sistema muestra una ventana de diálogo. 3. El usuario selecciona la dirección donde desea guardar el proyecto y le da un nombre. 4. El sistema guarda el proyecto. 5. El usuario puede seguir trabajando sobre el proyecto.
Flujo Alternativo: 5. El usuario cierra el proyecto.

Flujo de Eventos
Nombre: Cargar Proyecto
Descripción: Permite cargar la configuración de un proyecto. De esta forma la situación de las ventanas, configuración de la ecuación, etc. son cargadas en Dynamics.
Actores: Usuario de Dynamics.
Camino básico del caso de uso "Guardar Proyecto" 1. El usuario pulsa sobre el botón para cargar. 2. El sistema muestra una ventana de diálogo. 3. El usuario selecciona el proyecto Dynamics que desea cargar. 4. El sistema procede a la carga del proyecto creando una nueva ventana MDI. 5. El usuario puede trabajar sobre el proyecto.
Flujo Alternativo: 4. Si el proyecto ya ésta cargado en Dynamics, se activará la ventana asociada a él. 5. El usuario puede cerrar el proyecto.

Flujo de Eventos
Nombre: Configurar Ecuaciones
Descripción: Una vez creado el nuevo proyecto el usuario podrá configurar la ecuación seleccionada en el asistente (Wizard) con las herramientas que le ofrece Dynamics. Podrá establecer los parámetros de la ecuación, cambiar la forma de renderizado e incluso aplicar nuevo comportamiento a la ecuación con las funciones del menú Numerical Explorations.
Actores: Usuario de Dynamics.
Camino básico del caso de uso "Configurar Ecuaciones" 1. El usuario ha creado o cargado un nuevo proyecto 2. El sistema muestra el panel de renderizado así como el menú principal y los Dock para trabajar sobre la ecuación. 3. El usuario ejecuta una operación. 4. El sistema interpreta la operación y ofrece un resultado al usuario.
Flujo Alternativo:

3. El usuario cierra el proyecto.

3.2.2. Modelo de ejecución/navegación

Una vez descritos los posibles casos de uso que puede ejecutar el usuario junto con sus flujos de eventos vamos a describir brevemente la navegación que se puede realizar en la aplicación.

El siguiente diagrama muestra de forma genérica el conjunto de ventanas que se le mostrarán al usuario cuando ejecute la aplicación.

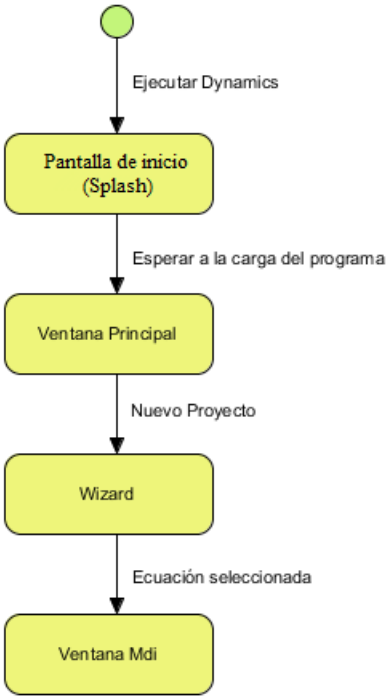


Figura 10: Diagrama de estados Dynamics

<u>Splash</u>

Es la ventana de inicio que se le muestra al usuario cuando inicia la aplicación. Muestra información sobre la carga del programa hasta que arranque la ventana principal.

Ventana Principal

Esta ventana es el contenedor de los proyectos que se crean, guardan y cargan en Dynamics. Está dividida en dos secciones las cuales se pueden diferenciar en:

- Barra de controles superior:

  La barra de menú y la barra de herramientas contienen los controles que el usuario puede ejecutar y que forman parte de la ventana padre contenedora. De tal forma, que estas acciones muestran los resultados en la sub-ventana hija, también llamada ventana MDI.

- Espacio de trabajo:

  Es donde el usuario trabaja con las ecuaciones. Las acciones ejecutadas en las barras de controles se ven reflejadas aquí.

Asistente (Wizard) para la creación de ecuaciones

Cuando el usuario quiera crear un nuevo proyecto tiene que rellenar los campos solicitados en el asistente de creación ecuaciones.
El asistente, no es más que un paso intermedio a la creación del nuevo proyecto. En él se selecciona el tipo de ecuación ya sea Maps o EDO. En el caso de que el usuario quiera crear su propia ecuación debe rellenar una serie de campos antes de finalizar el asistente.

Ventana MDI

Una vez finalizado el asistente se crea una nueva ventana conteniendo la configuración de la ecuación seleccionada. En la ventana MDI, que más adelante definiremos más detalladamente, se pueden realizar las acciones sobre el sistema elegido por el usuario.

Durante la planificación del proceso de navegación ya se empezó a pensar sobre que posibles clases se deberían implementar y la estructura de las mismas. Este proceso se llevaría a cabo en la etapa estructural.

3.3. Implementación

Tras la fase de desarrollo del esqueleto de la aplicación se han definido y concretado las clases que forman parte de la interfaz.
El primer paso ha sido la construcción de los diagramas de clases, en los cuales podemos ver las relaciones (asociativas, herencia, de uso y comportamiento) entre las clases que involucran el sistema.

Qt nos ha proporcionado la posibilidad de utilizar sus librerías ahorrándonos gran parte de código ya que mediante herencia hemos podido crear clases con la misma funcionalidad pero siendo más específicas.

El siguiente diagrama de clases define la estructura general de la interfaz gráfica. Está compuesto por:

- Clase: Mediante las clases representadas en el diagrama se ha encapsulado toda la información del objeto. A partir de ella hemos podido modelar el entorno que se estaba estudiando. Contemplan atributos, métodos y visibilidad de la propia clase.

- Relaciones: Mediante estas se explica cómo se comunican dos o más clases. Pueden ser de herencia, composición, agregación, asociación y uso.

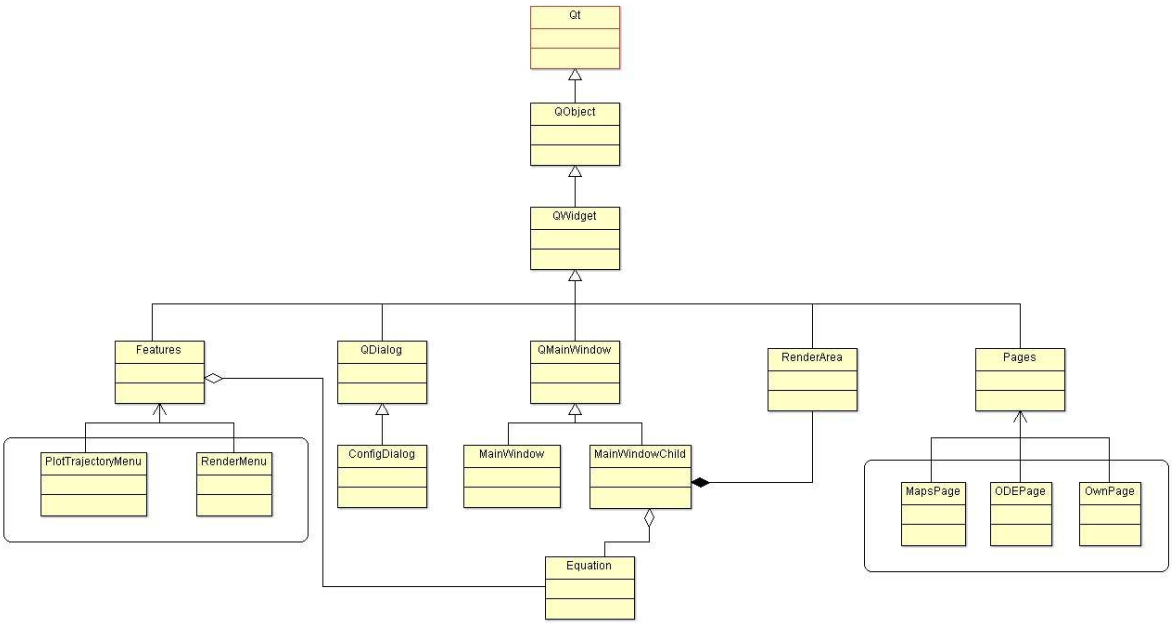


Figura 11: Diagrama de Clases Dynamics.

Como se puede observar en el diagrama anterior todas las clases implementadas en Dynamics heredan de la misma clase, QWidget. Esta clase ha jugado un papel fundamental para la construcción de la aplicación ya que implementa mucha de la funcionalidad que despues se ha usado en las clases heredadas.

3.3.1. Estructura interna de Dynamics

Debido a la longitud de las clases no se han incluido los metodos ni los atributos en el diagrama anterior. Estos quedan contemplados por separado a continuación.

Nombre de la Clase: MainWindow

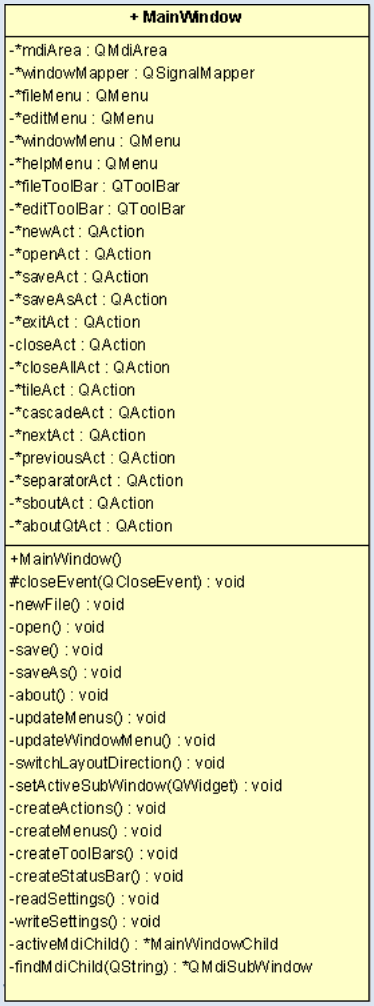


Figura 12: Clase MainWindow

Descripción:

Al ejecturar Dynamics se crea una instancia de esta clase y permanece en memoria hasta que se cierre el programa y por lo tanto, se libere. Desde el constructor de la misma se van haciendo llamadas a los diferentes métodos privados que configuran la ventana – createActions, createMenus, createToolBars, createStatusBar.

El objeto mas importante en esta clase es al que hemos llamado *mdiArea. Este objeto se crea en el constructor de la clase y es el que nos permite tener varios proyectos dentro de la misma ventana, actuando de esta forma como una interfaz multiple de documento MDI. Mediante los métodos setActiveSubwindow, activeMdiChild y findMdiChild podremos referenciar a los diferentes proyectos creados o abiertos.

Nombre de la Clase: MainWindowChild

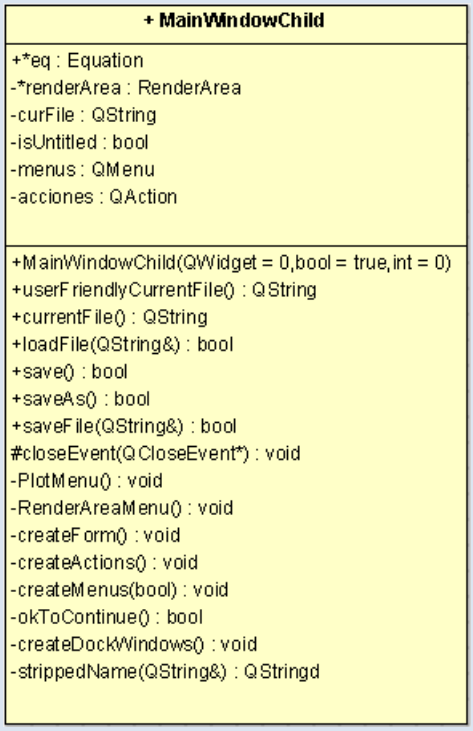


Figura 13: Clase MainWindowChid

Descripción:

Con esta clase creamos los proyectos que estan almacenados en el contenedor MDI. Como en la clase MainWindow, al llamar al constructor se configuran los elementos que formaran parte de la ventana. Pero esta vez, dependiendo del parametro que se le pase al constructor (si es Map o EDO), se mostraran los menus correspondientes al tipo de ecuación.
Esta clase contiene todas las funcionalidades de los proyectos nuevos que se crean, cargan y salvan.

Nombre de la Clase: ConfigDialog

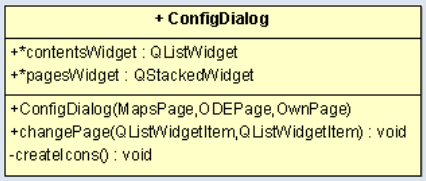


Figura 14: Clase ConfigDialog

Descripción:

La clase ConfigDialog se instancia cada vez que el usuario quiere crear un nuevo proyecto. Se pueden diferenciar dos partes:
Con el atributo *contentsWidget creamos el listado de iconos que se muestra en la parte izquierda del Wizard.
Con el atributo *pagesWidget añadimos las clases contenidas en la clase Pages, que son las posibles ecuaciones dentro de cada tipo de ecuación.
La relación entre las partes se ha implementado creando una conexión entre ambas. Cuando se hace click sobre los iconos de la parte izquierda del Wizard se lanza una señal para que se muestre en la parte derecha las ecuaciones del tipo de ecuación seleccionada.

Nombre de la Clase: Pages

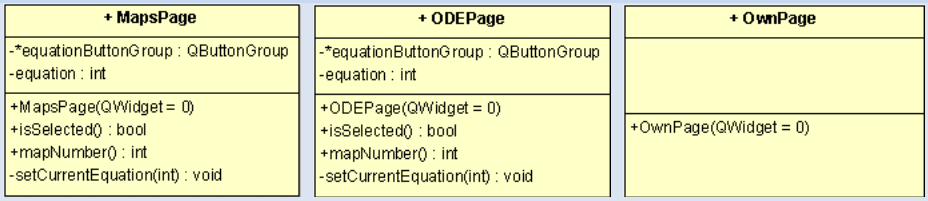


Figura 15: Clase contenedora Pages

Descripción:

Esta clase sirve como contenedor de clases. Es decir, dentro de la clase Pages tendremos las tres clases que se reflejan en la figura anterior. Se ha tomado la decision de introducirlas dentro de una clase única, Pages, para unificar las clases (MapsPage, ODEPage y OwnPage) que forman parte del dialogo de configuración (Wizard). De tal forma que a la hora de utilizar los objetos de la misma solo tengamos que hacer el <include> de la clase contenedora pudiendo utilizar las de dentro sín ningún problema.

Nombre de la Clase: Features

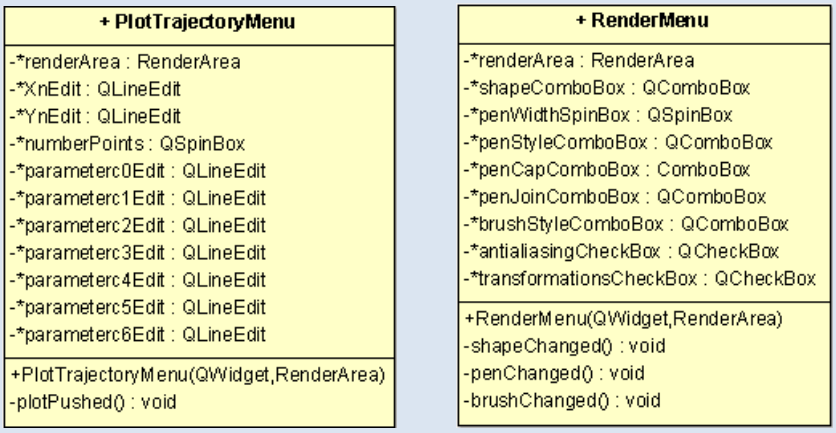


Figura 16: Clase contenedora Features

Descripción:

Como el caso anterior, la clase Features tambien sirve como contenedor de otras clases. En este caso, aquí se alojaran las clases que tengan que ver con las propiedades de la ecuación y las caracteristicas del panel de renderizado.
Hasta ahora solo se han implementado dos de estas clases, PlotTrajectoryMenu y RenderMenu. Ambas se utilizan como docks en la ventana MDI.
Con la clase PlotTrajectoryMenu podremos modificar las propiedades base de la ecuación, como son: puntos de partida xn e yn, número de puntos a representar y los parámetros.
Con la clase RenderMenu podremos modificar las propiedades del panel de renderizado pudiendo así representar la ecuación de varias formas, cambiar el grosor de los puntos e incluso activar la opción Antialiasing para una mejor representación.

Nombre de la Clase: Equation

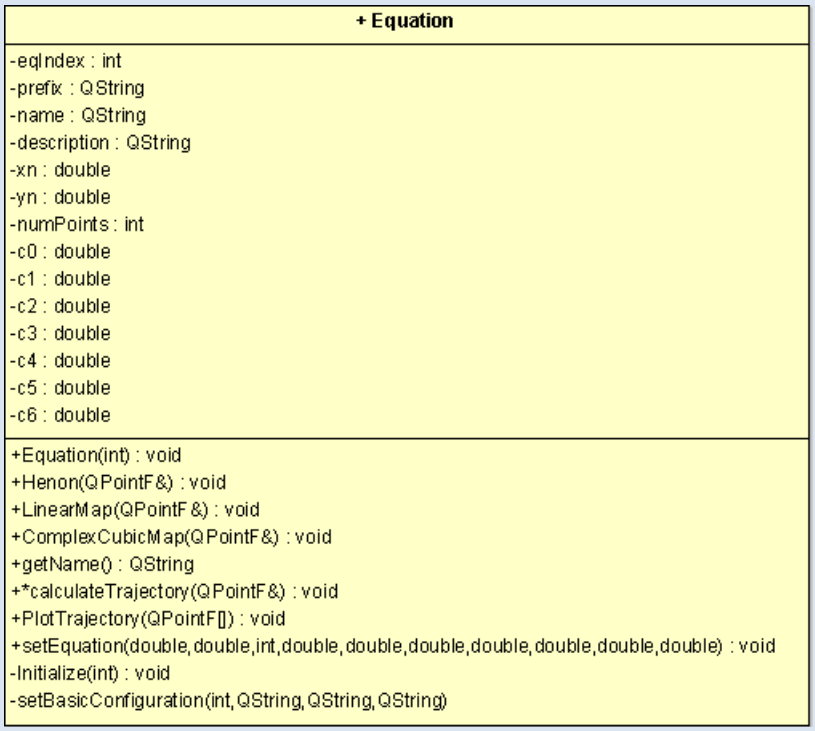


Figura 17: Clase Equation

Descripción:

Esta clase es una de las mas importantes del proyecto. Toma tanto protagonismo porque es la encargada de diferenciar cada tipo de ecuación y asociarlo a la opción seleccionada en el Wizard.
Todo esto se lleva a cabo a través de un "*puntero a función*" *calculateTrajectory(QPointF&):void. Cuando se instancia el objeto de tipo Equation se le pasa como parámetro el identificador de tipo *integer* asociado a cada ecuación. Dependiendo del parámetro el "*puntero a función*" apuntará a una ecuación u otra.
El método PlotTrajectory es el encargado de generar la trayectoria de la ecuación según el número de puntos que se quieran representar. Dentro de esta función se hace la llamada al "*puntero a función*" con fin de captar las trayectorias.

Nombre de la Clase: RenderArea

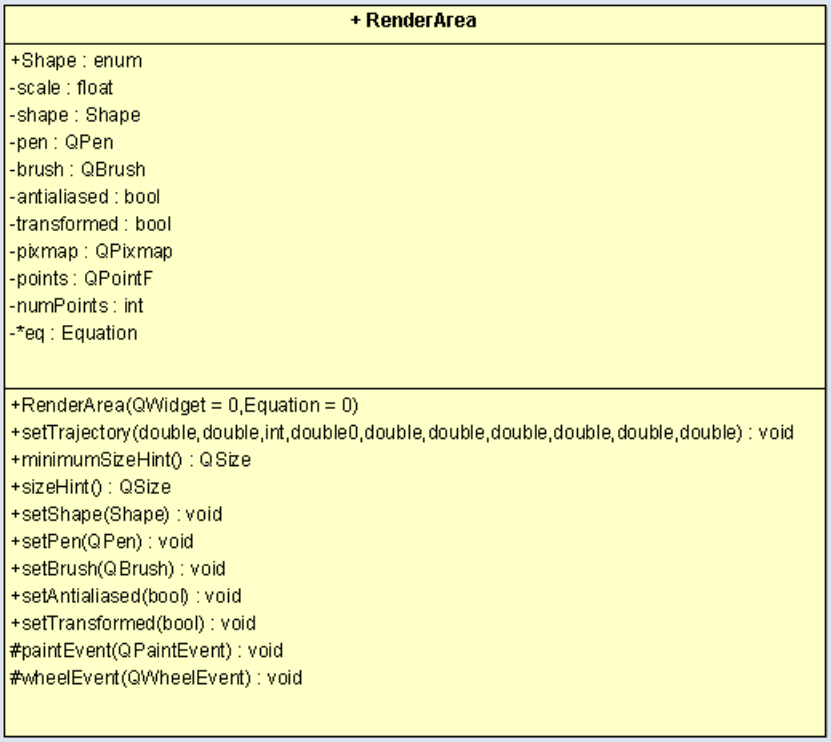


Figura 18. Clase RenderArea

Descripción:

Toda ecuación tiene que ser representada en alguna parte del programa. Pués bien, esta es la clase encargada de ello. Con ella podemos representar las trayectorias de las ecuaciones dependiendo del tipo de forma que haya seleccionado el usuario, establecer la pluma a la hora de pintar sobre el panel,etc.

El método donde reside la mayoria de la funcionalidad de la clase es #paintEvent(QPaintEvent). Este método responde a un evento de pintado cada vez que el usuario modifica algún parámetro del proyecto y ejecuta la acción.

Para mejorar el rendimiento de la clase el cálculo de la trayectoria no se hace cada vez que se ejecuta el evento “paintEvent” sino cuando el usuario realmente cambia los valores de la ecuación.

4. DISEÑO DE LA INTERFAZ GRÁFICA

El usuario final de Dynamics seguramente no sea un experto en informática, por ello se ha desarrollado una interfaz lo suficientemente sencilla para que sin tener mucho conocimiento sobre informática pueda desenvolverse con agilidad y de forma eficiente.

Seguramente el usuario estará familiarizado con las interfaces que nos ofrecen los sistemas operativos y los programas en general de tal forma que ya conocerá los elementos comunes que ofrecen las interfaces. El principal reto que se presenta es conseguir responder a los siguientes objetivos:

- **Interfaz fácil de aprender y utilizar:** Tras una breve demostración tanto los usuarios que hayan utilizado la versión consola Dynamics como los que nunca la hayan utilizado deberán ser capaces de manejar sin problemas la aplicación.

- **Cumplir con las necesidades de interacción usuario-sistema:** La interfaz es el medio por el cual el usuario se podrá comunicar con el sistema y por lo tanto, tendrá que ser capaz de transmitir la información necesaria para que ambos interlocutores entiendan los datos que se intercambian.

- **Ofrecer los elementos apropiados para realizar las tareas requeridas:** Es importante tener buenos mecanismos para recopilar la información que ha de introducir el usuario y como ésta se le mostrará por pantalla. En todo momento se debe facilitar el trabajo al usuario final.

El cumplimiento de estos tres objetivos nos garantiza tener una herramienta potente en términos de usabilidad, eficiencia y rapidez. Además de lo comentado, al construir la interfaz se ha intentado que ésta mantenga un estilo propio y sea estéticamente atractiva.

4.1. Estilo de la interfaz

Partiendo de la interfaz ya creada para Dynamics en modo consola se ha ido extrayendo información en pequeños módulos para adaptarlos y convertirlos en partes de la interfaz gráfica.

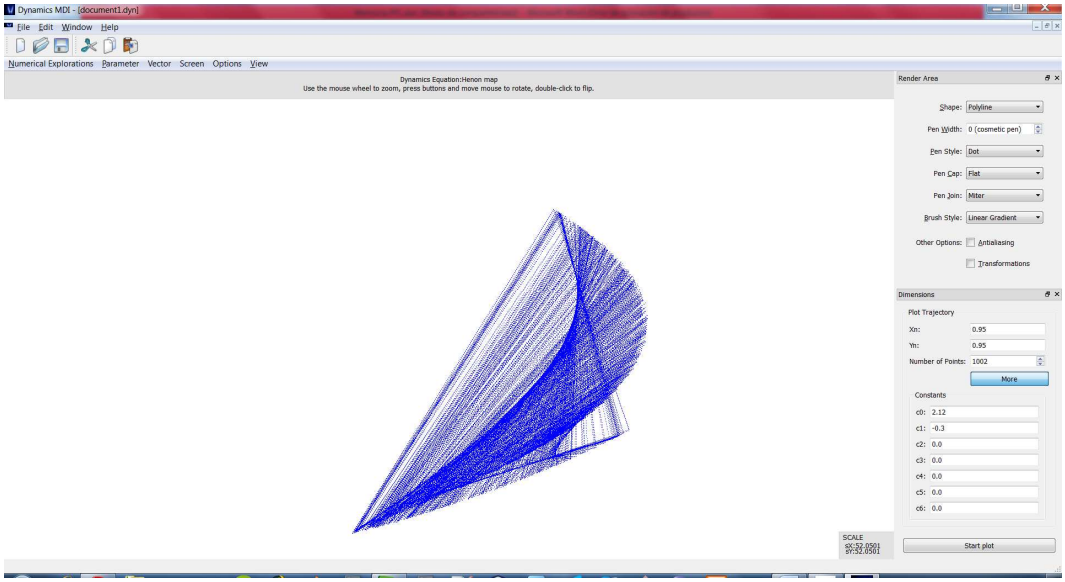


Figura 19: Interfaz de Dynamics

En un principio se pensó en crear una interfaz de documento único (SDI-Single Document Interface) por el cual la organización en ventanas de la interfaz gráfica seria manejada por el Administrador de Ventanas del Sistema Operativo (Operating System's Windows Manager) de forma separada. Se cumplirían los siguientes requisitos:

- Cada ventana se representaría como una entrada individual en la barra de tareas del sistema operativo o el gestor.

- Cada ventana contendría su propio menú o barra de herramientas, no teniendo un “fondo” de la ventana o ventana “padre” con su menú o barra de herramientas.

De esta forma, cada vez que ejecutáramos la opción de “Nuevo Proyecto” se abriría una nueva instancia del programa, teniendo por separado ambos proyectos.

Sin embargo, se llegó a la conclusión de que sería más práctico el hecho de tener una aplicación con múltiples interfaces de documento (MDI-Multiple Document Interface), de tal forma que los nuevos proyectos creados en Dynamics residieran bajo una misma ventana padre. Algunas de las ventajas que nos ofrece este sistema son las siguientes:

- Estos sistemas permiten que las ventanas hijas puedan incrustar otras sobre sí mismas, creando complejas jerarquías anidadas.

- La barra de menús individuales y/o la barra de herramientas se comparte entre todas las ventanas secundarias, lo que reduce el desorden y el aumento del uso eficiente de espacio de la pantalla.

- Posibilita un aumento en la rapidez y eficiencia de la memoria, ya que la aplicación es compartida. La velocidad de conmutación entre ventanas internas

es generalmente más rápido que hacer que el SO cambie entre las ventanas exteriores.

- Sin una ventana de marco MDI, las barras de herramientas flotantes de una aplicación pueden estorbar el espacio de trabajo de otras aplicaciones, lo que podría confundir a los usuarios con la mezcla de las interfaces.

- Las ventanas de una aplicación hija se pueden ocultar / mostrar / minimizar / maximizar como un conjunto.

- Características tales como “Mosaico” y “Cascada” se pueden aplicar a las ventanas secundarias pudiendo evitar el solapamiento de contenidos.

4.1.1. Iconos

El uso de iconos en una interfaz no solo conlleva una decoración extra en la misma, sino que son una parte fundamental de los mecanismos de interacción con el usuario por lo que deben ser diseñados cuidadosamente. Aunque su funcionalidad suele estar ligada a una determinada tarea, su uso adecuado aporta grandes ventajas.

Los iconos representan una acción, idea o concepto determinado.

A continuación detallo las ventajas que estos objetos pueden ofrecernos:

- Con ellos se pueden representar más funciones en un menor espacio. Por lo que en interfaces en las que se pretende obtener una funcionalidad máxima en un espacio reducido.

- El significado de estos debe ser aprendido. Una vez que se haya utilizado no le debe costar al usuario recordar para que servía.

- Los iconos suelen despertar la curiosidad del usuario, por lo tanto si se desea que el usuario investigue y descubra por sí mismo el funcionamiento del programa, los iconos juegan un gran papel.

En nuestra aplicación hemos incluido una serie de iconos en la barra de herramientas del programa principal. Para cualquier usuario ya habituado al uso de interfaces estos iconos le serán familiares y fácilmente reconocibles.

Figura 20: Iconos Barra Herramientas

También cabe destacar el uso de los iconos mostrados en el Wizard cuando ejecutamos la acción de Nuevo Proyecto.

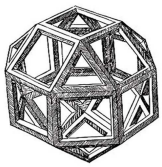


Figura 21: Icono Ecuacion Tipo Map

Figura 22: Icono Ecuacion Tipo EDO

Figura 23: Icono Ecuacion propia

4.1.2. Look&Feel

Debido a la inexistencia de una interfaz gráfica previa en la que basarnos no ha habido problema a la hora de marcar el estilo que esta ha ido tomando a lo largo del desarrollo del proyecto. En cuanto a los widgets, ventanas, docks, etc. dependiendo del Sistema Operativo donde se ejecute el programa tomará un estilo u otro nativo al propio sistema operativo.

Figura 24: Representación de widgets en los diferentes S.O

Todo esto se ha podido llevar a cabo gracias a las librerías de Qt, más concretamente a la clase QStyle. Esta clase está construida dentro de la librería QtGui la cual contiene las subclases de QStyle que emulan los diferentes aspectos de las plataformas soportadas por Qt (QWindowsStyle, QMacStyle, QMotifStyle, etc.). Con esta clase abstracta hemos podido encapsular el Look&Feel de la interfaz gráfica.

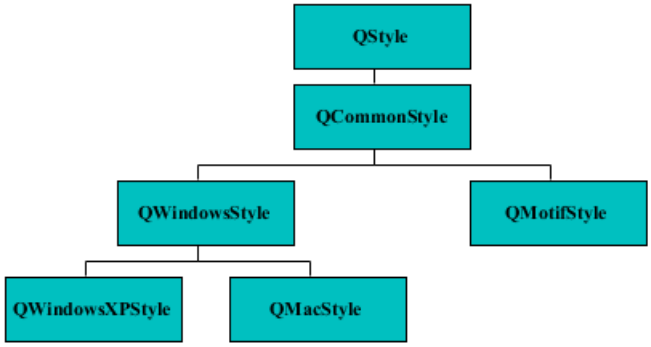


Figura 25: Diagrama de Clases Estilos Qt

4.2. Elementos básicos de la GUI

Para la realización de la interfaz gráfica se utilizaron muchas de las clases que contiene el módulo QtGui. Dicha librería extiende la funcionalidad de QtCore (la cual implementa la funcionalidad no gráfica) permitiéndonos construir interfaces gráficas para usuarios.

A continuación detallo los elementos y clases utilizadas para la creación de estos componentes y una breve descripción de su función.

4.2.1 Gestores de Organización (Layout Manager)

Los gestores de organización son componentes de software que sirven como contenedores de Widgets sin necesidad de utilizar unidades de distancia para situar a estos dentro del contenedor.

QGridLayout

Establece Widgets en una cuadricula. Este contenedor toma el espacio que se le ha prestado, lo divide en filas y columnas y coloca cada Widget que gestiona en la celda correcta.

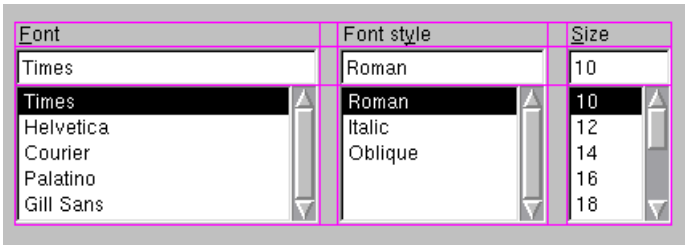


Figura 26: Gestor de Organización QGridLayout

QHBoxLayout

Este tipo de contenedor alinea los widgets que contiene de forma horizontal.

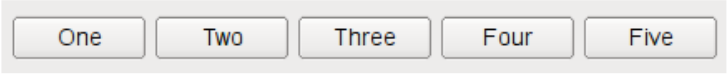


Figura 27: Gestor de Organización QHBoxLayout

QVBoxLayout

Este tipo de contenedor alinea los widgets que contiene de forma vertical.

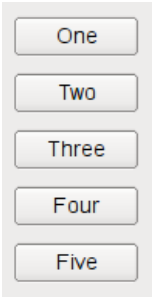


Figura 28: Gestor de Organización QVBoxLayout

4.2.2 Conectores (Signals/Slots)

Mediante los conectores Signal y Slot podemos establecer comunicaciones entre objetos. Es la manera con la que podemos notificar a otro Widget cuando este ha sido modificado.

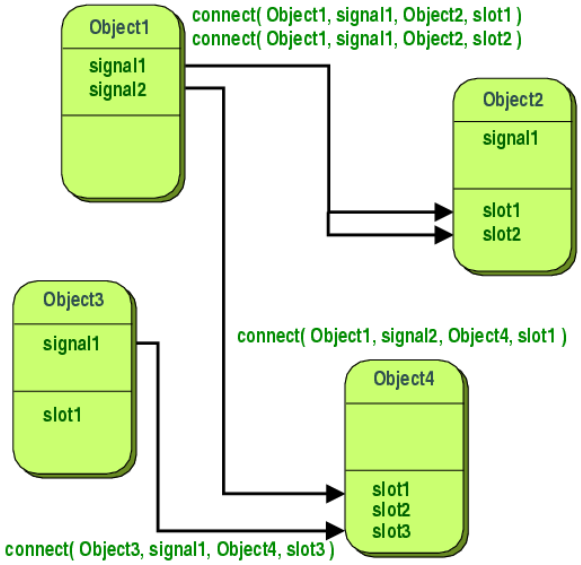


Figura 29: Ejemplo de Conectores Qt.

Los Signal (señales) son emitidos cuando un evento ha ocurrido. Y por lo tanto, un Slot (ranura) es una función que es llamada en respuesta de una determinada señal.

4.2.3 Controles

La clase base utilizada para todos los objetos utilizados en la interfaz gráfica es QWidget. Esta clase es la encargada de recibir los eventos de ratón, teclado así como otros eventos producidos por el sistema de ventanas y pintar su representación en la pantalla.

Check Box

Con esta clase se ha implementado la funcionalidad de los checkbox.

Figura 30: Ejemplo de vista checkbox.

Spin Box

Con esta clase se ha implementado la funcionalidad de los spinbox.

Figura 31: Ejemplo de vista spinbox.

Combo Box

Nos permite crear listas emergentes en un botón combinado. Con esta clase se ha podido presentar listas de opciones al usuario de la forma más compacta posible reduciendo al máximo el espacio utilizado en la pantalla.

Radiobutton

Para crear este tipo de elementos hemos echo uso de la clase QRadioButton la cual implementa la funcionalidad de los mismos. Con ella hemos creado los botones de opción que se pueden ver en el Wizard al seleccionar la ecuación del proyecto.

Este tipo de botones por defecto son auto exclusivos, lo que significa que si se activa uno de ellos y después se activa otro el primero queda desactivado. Para identificar a cada “Radio Button” dentro de un mismo grupo los hemos introducido en un objeto de la clase QButtonGroup. De tal forma que cada “Radio Button” tiene asociado un identificador.

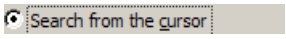


Figura 32: Ejemplo de vista radiobutton.

Push Button

Usando la clase QPushButton hemos podido fabricar botones de comando.

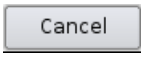


Figura 33: Ejemplo de vista pushbutton.

Label

La clase QLabel nos ha permitido crear etiquetas para insertar tanto texto como imágenes.

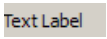


Figura 34: Ejemplo de vista etiquetas de texto.

Line Edit

Con el uso la clase QLineEdit hemos podido fabricar líneas de texto.

Figura 35: Ejemplo de vista lineas de texto.

Para impedir que el usuario indroduzca valores erroneos en los campos de texto hemos creado una expresion regular que debe seguir cualquier campo de texto. De tal forma que si el usuario en un campo donde se le pide un número intenta introducir una letra, el sistema no le dejará. Así evitamos errores al procesar las peticiones del usuario.

La expresion regular que sigue los campos de texto es la siguiente:

```
QRegExp regExp ("[-]{0,1}[0-9]*[.][0-9]*");
```

4.2.4 Ventanas y componentes

QDialog

Este clase es la base de las ventanas de dialogo. Se utiliza sobre todo para tareas de corto plazo y comunicaciones rápidas con el usuario.

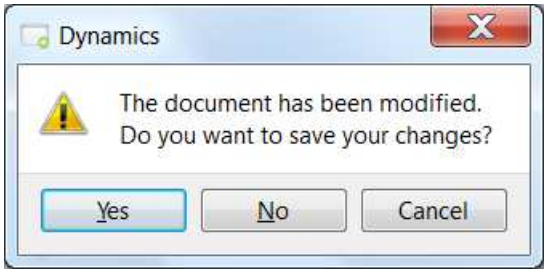


Figura 36: Ventana de dialogo

QFileDialog

Con esta clase se provee un dialogo que permite a los usuarios seleccionar archivos y carpetas. Un caso típico será cuando el usuario necesite guardar la configuración del proyecto actual o cargar la configuración de otro que guardo previamente.

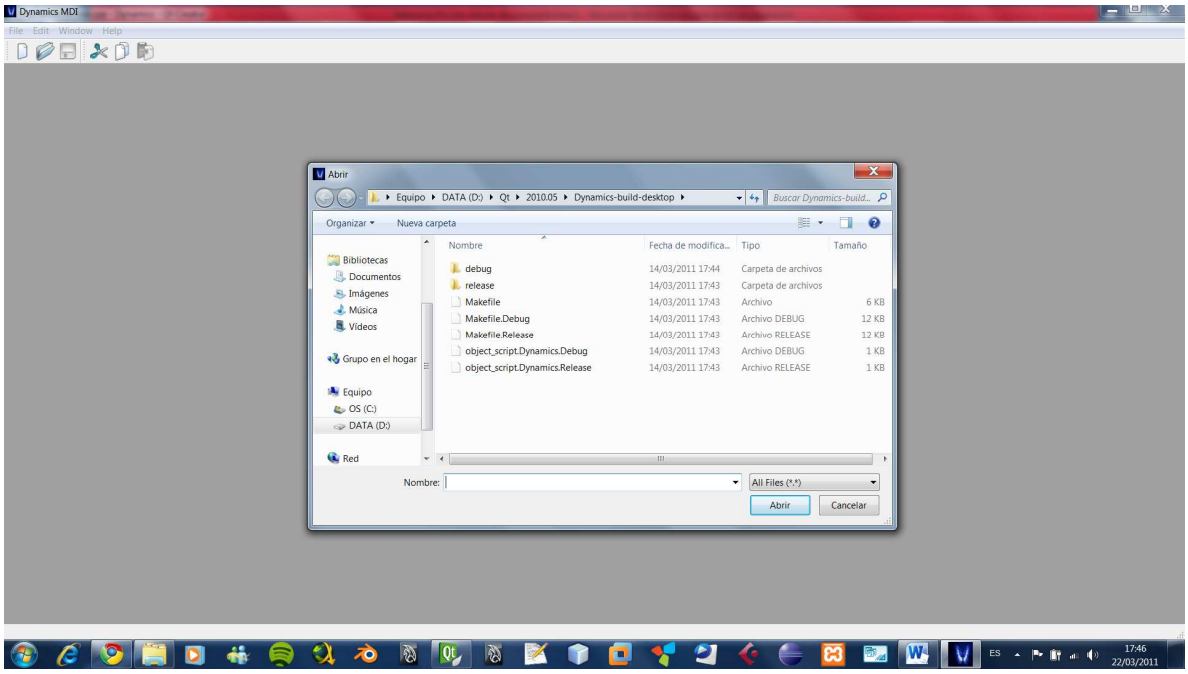


Figura 37: Ventana de dialogo para cargar/salvar proyectos

QMessageBox

Un cuadro de mensaje muestra un texto inicial para alertar al usuario sobre una situación, o un texto informativo para explicar con más detalle la descripción del problema o para mostrar al usuario una pregunta. También suelen incluir iconos para notificar la gravedad del mensaje.

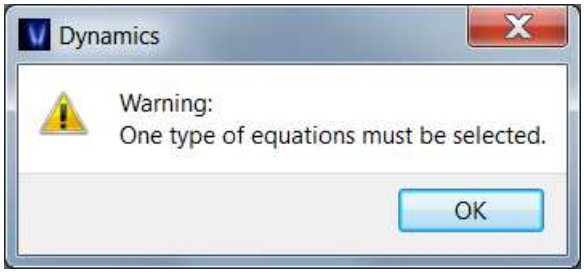


Figura 38: Ventana de dialogo para mostrar mensajes

	Pregunta	Para hacer una pregunta sobre las operaciones normales.
	Información	Reportar información sobre alguna operación
	Aviso	Para avisar sobre algún error no critico
	Critico	Para reportar errores críticos

SplashScreen

Durante el inicio de la aplicación se le ofrece al usuario una pantalla de presentación. Dicha pantalla sirve para informar al usuario que se está configurando la ventana principal, cargando los módulos, etc.

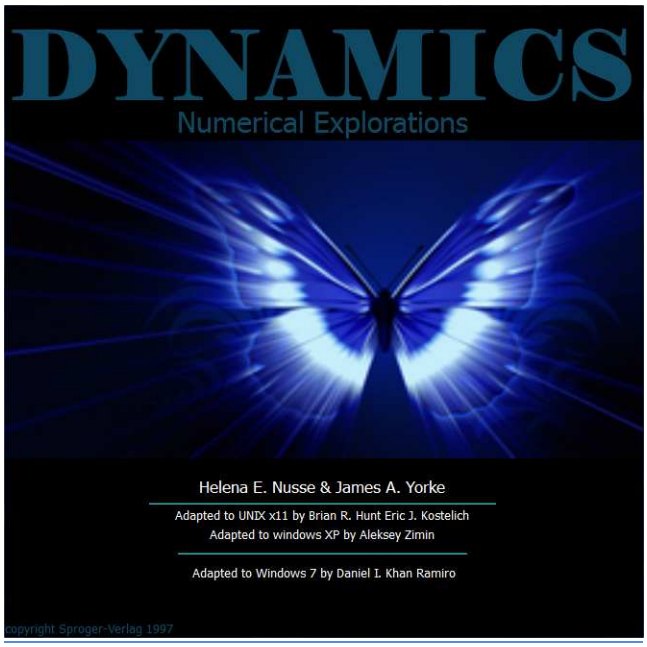


Figura 39: Splash Dynamics

4.2.5 Eventos

QPaintEvent

Esta clase contiene los parámetros de evento para los eventos de pintura. Cada vez que actualizamos los parámetros de una ecuación desde el dock “Dimensions”, modificamos el estilo del área de renderizado con el dock “Render Área” o incluso al minimizar y volver a abrir el proyecto se activa este evento.

QWheelEvent

Esta clase contiene los parámetros de evento para los eventos sobre la rueda del ratón. A través de este evento podemos capturar la intención del usuario si este desea hacer zoom sobre el área de renderizado.

4.3. Organización de elementos

De cara a la usabilidad de la interfaz es muy importante tener los elementos que la componen bien organizados dentro de la misma. De tal forma que el usuario no tenga que hacer mucho esfuerzo para poder navegar sobre ella.

4.3.1. Ventana Principal (Contenedor MDI)

QApplication

Con la clase QApplication manejamos el flujo de control de la aplicación GUI y los ajustes principales. Contiene el administrador de eventos principal donde se procesan y se envían los eventos del sistema de ventanas. También maneja la inicialización, finalización y provee la administración de la sesión.

QMainWindow

Esta clase provee el marco de trabajo para construir una aplicación con interfaz de usuario. QMainWindow tiene su propio contenedor de Widget en el que se podrán añadir barra de herramientas, dock widgets, barras de menú y barras de estado.

El contenedor dispone de un área central que puede ser ocupado por cualquier tipo de Widget. En nuestro caso, el área central está ocupada por el objeto MDI (Multiple Documnet Interface) permitiéndonos tener cuantos proyectos queramos dentro del programa.

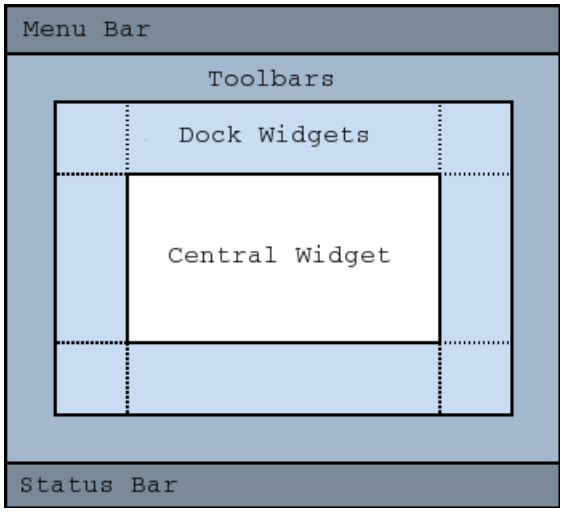


Figura 40: Estructura de la ventana principal

QMdiArea

Los Widget QMdiArea proporcionan un área en la que se pueden mostrar las ventanas MDI. Esta clase funciona básicamente como un gestor de ventanas para las ventanas MDI.

QMdiSubWindow

Proporciona una clase para la creación de ventanas dentro del contenedor MDI Área. Con esta clase cada vez que se cliquea sobre el botón "Nuevo Proyecto" se genera una nueva ventana que se incluirá en el gestor de ventanas del MDI Área.

4.3.2. Menú principal

Menú

Mediante la clase QMenu hemos podido incorporar un control de menú para usar en la barras de menús e incluso en los menús contextuales. Los menús desplegables se mostrarán al usuario cuando este haga clic sobre él.

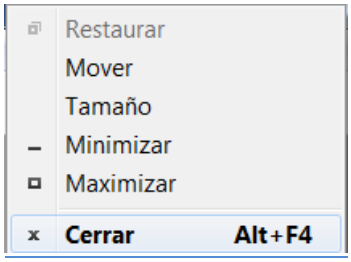


Figura 41: Menu desplegable

Barra de menú.

Para crear la barra de menús que se muestra en Dynamics hemos tenido que hacer uso de la clase QMenuBar la cual implementa la funcionalidad de las barras de herramientas horizontales. La barra de menú no es más que un conjunto de menús que el usuario tiene a disposición en todo momento y que al cliquear sobre él se despliegan una serie de acciones.

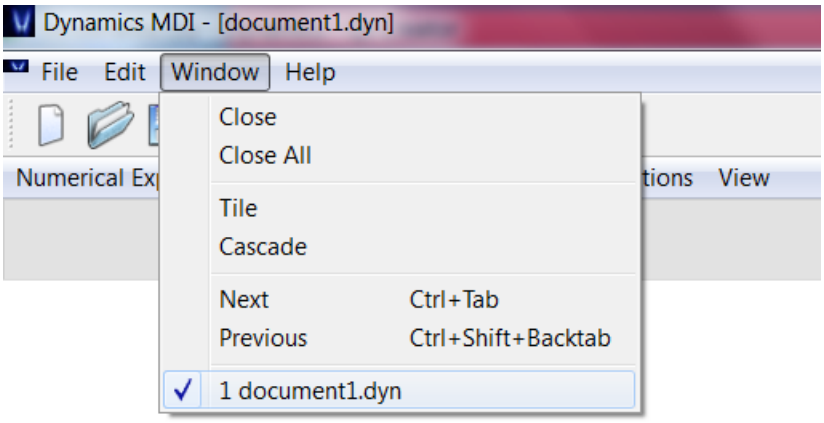


Figura 42: Barra de menu de Dynamics

Barra de estado

Con la barra de estado proporcionamos al usuario una barra horizontal para la presentación de información de estado. En Dynamics se utiliza, por ejemplo, para explicar brevemente la tarea a desempeñar cuando se posiciona el ratón encima de cualquier elemento de la barra de menús o la barra de herramientas.

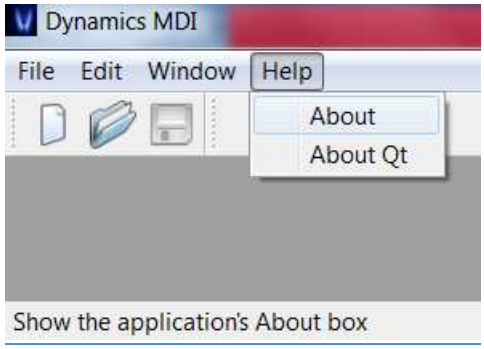


Figura 43: Barra de estado de Dynamics

4.3.3. Barra de herramientas

Barra de herramientas

Con la clase QToolBar creamos la barra de herramientas que se muestra en la figura. La barra de herramientas actúa como un panel móvil permitiendo al usuario moverlo donde más cómodo le resulte su utilización.

Figura 44: Barra de herramientas de Dynamics

De momento se han puesto los controles:

Crear un nuevo proyecto.
Abrir un proyecto existente.
Salvar el proyecto actual.

4.3.4. Asistente (Wizard)

Siempre que se quiera crear un nuevo proyecto se deberá rellenar el Wizard que se muestra en la siguiente figura.

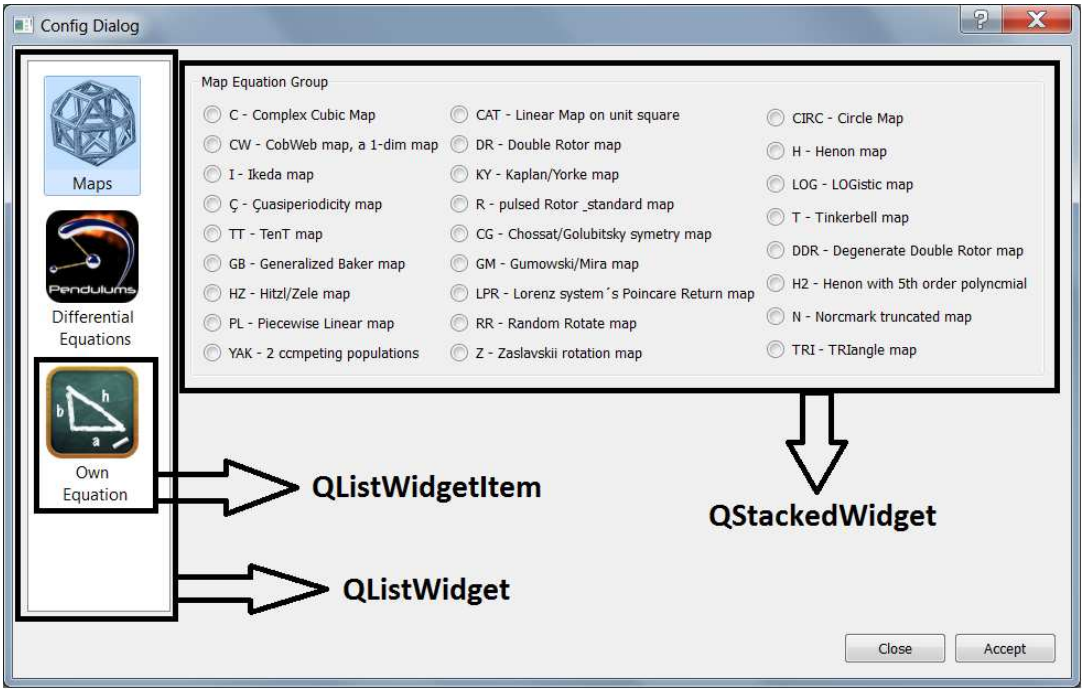


Figura 45: Ventana Wizard para la configuracion de un nuevo proyecto.

El Wizard está compuesto básicamente por dos tipos de Widget:

- QListWidget

  El Widget QListWidget sirve como contenedor para los elementos de tipo QListWidgetItem. Sobre estos últimos actuará el usuario para cambiar el tipo de configuración que desea.

- QStackedWidget:

  En dicho Widget introdujimos la configuración de cada una de las ecuaciones – Maps, Differential Equations y Own Equation. La clase QStackedWidget nos ha proporcionado la facilidad de poder introducir estos tres Widget en uno solo, donde solo uno de ellos podría ser visto a la vez.

La forma de comunicar los Widget QListWidget y QStackedWidget se hizo mediante señales.

4.3.5. Área de renderizado y Docks

QDockWidget

La clase QDockWidget proporciona un Widget que puede ser acoplado en un objeto de la clase QMainWindow o utilizado como una ventana de nivel superior en el escritorio. Las ventanas Dock son ventanas secundarias que se posicionan alrededor del área central del dock Widget área dentro del QMainWindow.

En nuestro caso, este tipo de Docks se han utilizado como pequeñas ventanas de servicio con las que podemos configurar la ecuación seleccionada en el Wizard. De tal forma, que si deseamos cambiar la configuración de la ecuación, cambiar el estilo del área de renderizado, etc. podremos llevarlo a cabo con estos Docks.

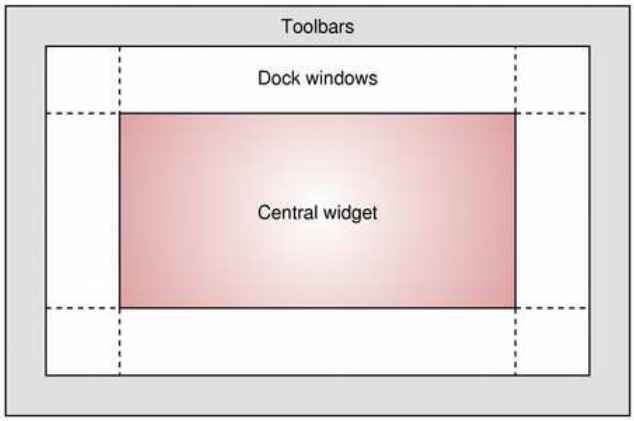


Figura 46: Estrucutura de los Docks en Dynamics

Espacio de Renderizado de Ecuaciones

Mediante la clase QPainter se ha podido representar las trayectorias de las ecuaciones. Dicha clase provee mucha de la funcionalidad que requieren las GUI de pintado, como es en nuestro caso, el panel de dibujado de la ecuación. Se ha optado por utilizar esta clase porque además de permitirnos pintar sobre un Widget nos ha proporcionado herramientas para personalizar la configuración y calidad de renderizado del panel

4.4. Funcionalidades de la interfaz.

Las librerías de Dynamics han sido creadas para visualizar el comportamiento de los sistemas dinámicos. Como funcionalidad básica, la interfaz de Dynamics cumple este requisito y permite al usuario representar cualquier tipo de ecuación así como una creada por él mismo.

Para dar un valor añadido a la interfaz se han implementado operaciones tales como crear, cargar y salvar proyectos además de las creadas en la ventana MDI.

Nuevo proyecto:

Mediante esta operación el usuario puede crear cuantos proyectos quiera. Al pulsar sobre este icono se despliega el Wizard con el que se puede configurar el tipo de ecuación deseado.
El atajo de teclado para esta acción es: Ctrl + N
El mensaje de estado en la barra de estado será el siguiente: Create a new file

Abrir proyecto:

Mediante esta operación el usuario puede cargar algún proyecto Dynamics. Al pulsar sobre este icono se le muestra al usuario una ventana modal en la cual podrá elegir el archivo con extensión .dyn que cargará la aplicación.
El atajo de teclado para esta acción es: Ctrl + O
El mensaje de estado en la barra de estado será el siguiente: Open an existing file

Guardar proyecto:

Mediante esta operación el usuario puede guardar la configuración del proyecto actual. Si este no había sido guardado se le pregunta al usuario en donde desea guardarlo y con qué nombre, sino guarda el proyecto actualizando el existente.
El atajo de teclado para esta acción es: Ctrl + S
El mensaje de estado en la barra de estado será el siguiente: Save the document to disk

Modificar el área de Renderizado:

Con el dock Render Área el usuario puede modificar la forma de renderizado de la ecuación. De esta forma, una vez generada la ecuación el usuario dispone de diferentes opciones para su representación.

En la siguiente figura se muestra el dock encargado de tal tarea:

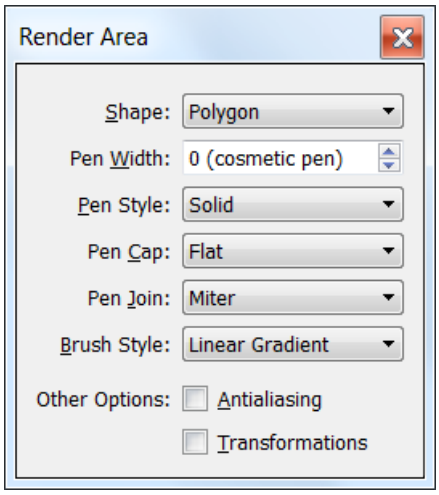
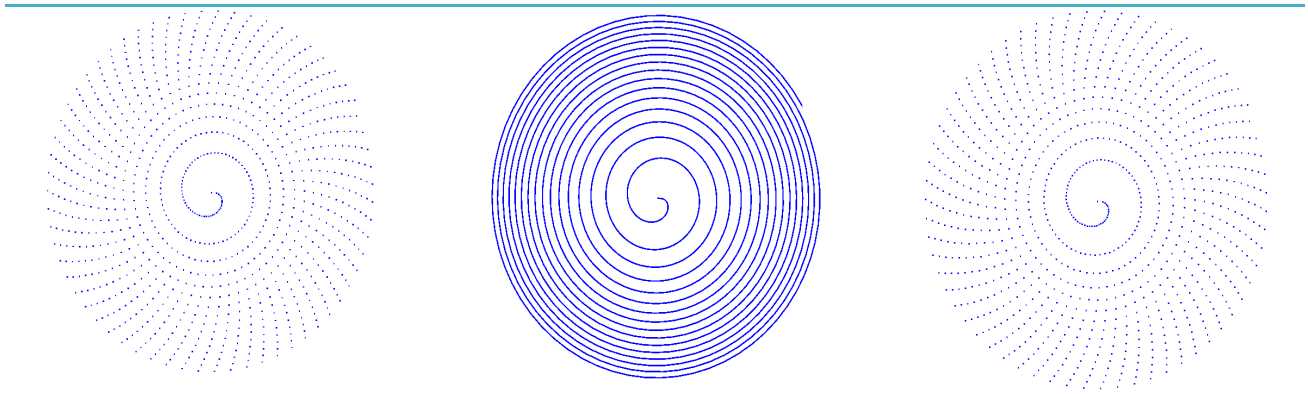


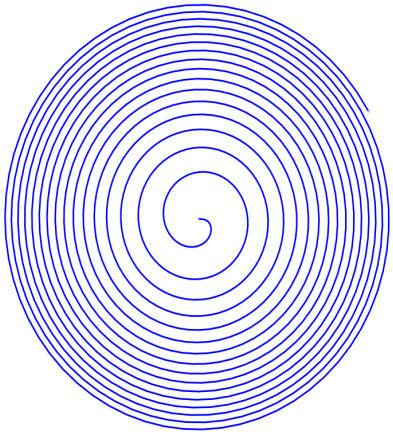
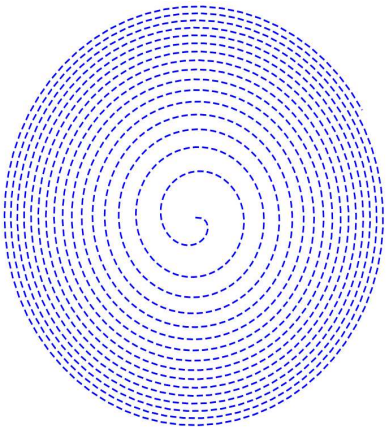
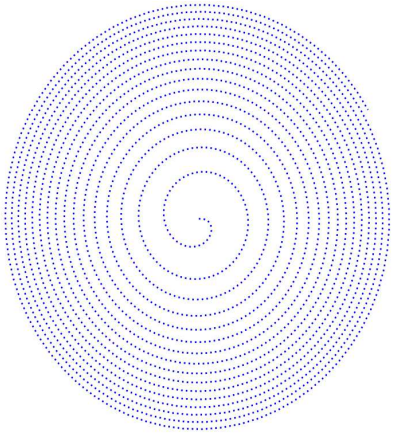
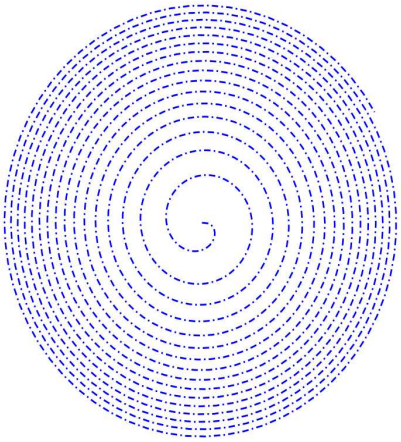
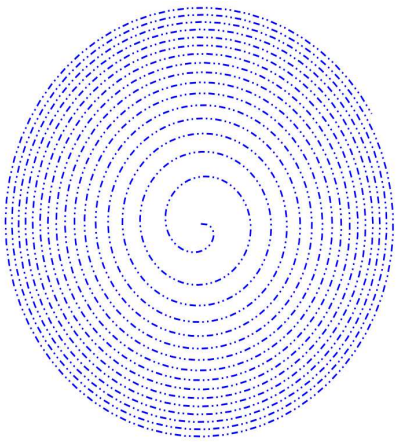
Figura 47: Dock para el Renderizado



Polygon	Polyline	Points

- Shape: Son los tipos de figuras con las que podemos representa la ecuación.
- Pen Width: Define el ancho con el que se dibujan los puntos. Por defecto esta puesto cosmetic pen.

- Pen Style: Define el color o el punteado que se utiliza para dibujar líneas o límites.

		
Solid	**Dash**	**Dot**
		
Dash Dot	**Dash Dot Dot**	**None**

- Pen Cap: Define el estilo de capuchon de la pluma con el que se pintan las ecuaciones.
- Pen Join: Define la union de estilos de la pluma.
- Brush Style: Define el color o patrón que se utiliza para el llenado de formas.
- Other options:
  - Antialiasing: El motor de pintado minimizará el aliasing cuando intente representar una imagen de alta resolucion en un sustrato de mas baja resolución.
  - Transformations: Se representará la imagen de forma transformada trasladando, rotando y escalando su posición.

Modificar los parámetros de la ecuación:

Es el Dock mas importante a la hora de trabajar sobre las ecuaciones en los proyectos Dynamics. En el se definen los parámetros de la ecuación asi como el número de puntos con el que queremos representarla.

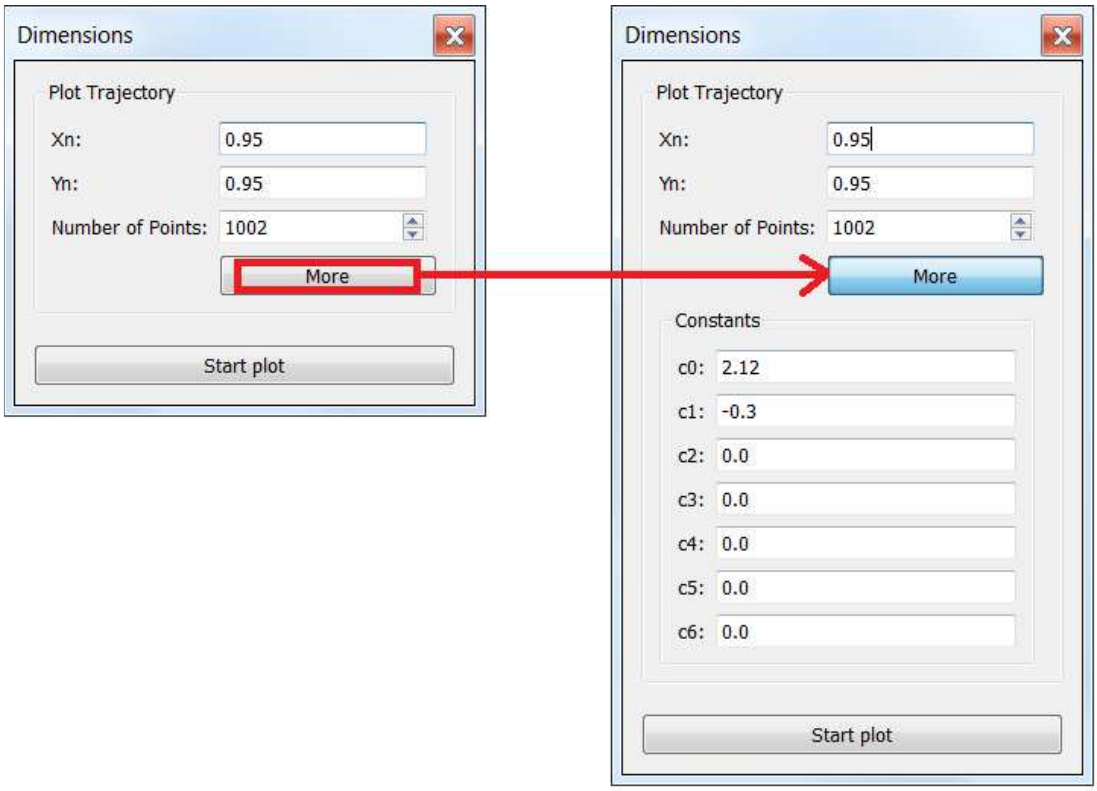


Figura 48: Dock para la configuracion de los parametros de la ecuación

- Xn: Marca la coordenada inicial del eje x.
- Yn: Marca la coordenada inicial del eje y.
- Number of Points: Marca el número de puntos con el que queremos representar la ecuación
- Constants: Definen los parámetros que tendrá en consideración la ecuación a la hora de calcular la trayectoria.

5. FUTURO DE DYNAMICS

Con la creación de esta interfaz gráfica se ha dado un gran paso en cuanto al desarrollo futuro y posible ampliación de la aplicación para una mayor funcionalidad. El diseño de la interfaz gráfica no ha sido tarea fácil pero al final se ha conseguido cumplir con las

funcionalidades básicas que se plantearon al principio del proyecto. Siendo así, y pensando en que este proyecto será la base para otros, se ha respetado una estructura clara abierta a posibles mejoras ayudando así a que la aplicación pueda ser ampliada por otros desarrolladores.

Como posibles mejoras a realizar sobre la interfaz Dynamics se encuentran:

- La utilización de las librerías OpenGL para la representación de las ecuaciones tanto en 2D como en 3D. De esta forma la interfaz incrementaría muchísimo su funcionalidad.
- Implementación de las funciones creadas en Dynamics modo texto en la nueva interfaz gráfica.

6. CONCLUSIONES

Los resultados del presente trabajo permiten comprobar el incremento de la facilidad de uso de una aplicación a la hora de utilizar herramientas con una clara e intuitiva interfaz gráfica. El procedimiento utilizado capta la funcionalidad que había en la interfaz Dynamics modo texto, puesto que se partió de la misma a la hora de realizar la interfaz gráfica, y la convierte en una aplicación multiplataforma dispuesta para ser ejecutada en varios sistemas operativos.

Los objetivos propuestos se han cumplido de forma satisfactoria, ya que tal y como se pretendía desde un primer momento, se ha creado la interfaz gráfica de usuario así como el módulo de operación de las ecuaciones donde se llevarán a cabo las instrucciones y operaciones necesarias para su ejecución. Además, para que el usuario final tenga más libertad de movimiento a la hora de tratar con las ecuaciones, se ha implementado variedad de funcionalidad extra como es la configuración del área de renderizado y parametrización de las ecuaciones.

La cantidad de técnicas y conceptos aprendidos, así como el saber que esta aplicación puede resultar muy útil a la comunidad científica para observar el comportamiento de ecuaciones tipo Maps, Edo o incluso las creadas por uno mismo ha sido muy gratificante y muy enriquecedor. Por lo tanto, estoy seguro de que gran parte del conocimiento adquirido durante el desarrollo del proyecto me ayudarán a lo largo de mi carrera profesional.

Cabe destacar, que durante el desarrollo del proyecto se han presentado algunos problemas tales como:

- Análisis y captura de requisitos de una aplicación ya creada como era Dynamics en modo texto. Las librerías estaban implementadas en código C y, por lo tanto, no las podíamos utilizar directamente desde nuestra interfaz. Así que se ha tenido que idear un procedimiento para su implantación en la interfaz gráfica.

- El proceso de dibujado de las ecuaciones ha sido complicado hasta que finalmente se ha conseguido llevar a cabo. En un principio se pensó en utilizar las librerías de OpenGL ya que incrementaba mucho la funcionalidad de la aplicación, pero debido a la falta de tiempo se ha procedido a utilizar la librería QPaint que ofrece Qt.

El conocimiento que se adquiere en el día a día al solucionar cada dificultad que se va presentado en el desarrollo del proyecto, tiene un valor incalculable que repercute directamente en el desarrollo profesional presente y futuro. A pesar de las dificultades, analizando todo el trabajo que ha llevado consigo el proyecto fin de carrera y la cantidad de tiempo invertido, me siento orgulloso de la aplicación creada. El balance ha sido muy positivo y el conocimiento aplicado ha sido inmenso.

7. BIBLIOGRAFIA

[1] *Wikipedia, la enciclopedia Libre* [en línea] [] Disponible en Web http://es.wikipedia.org

[2] Jasmin Blanchette y Mark Summerfield. C++ GUI Programming with Qt 4 [en línea] Universidad Rey Juan Carlos

[3] *Qt Reference Documentation* [en línea] [] Disponible en Web http://doc.qt.nokia.com/4.6/index.html

[4] Deitel. How to program C [en línea] Universidad Rey Juan Carlos

[5] Patrick Ward. Qt programming for Windows and Linux [en línea]

[6] Dalheimer, Matthias Kalle. Programming with Qt [en línea]

[7] http://www.usolab.com/wl/2008/01/el-iceberg-de-la-experiencia-d.php

[8] http://www.monografias.com/trabajos6/inus/inus.shtml#cara

[9] http://www.chr5.com/investigacion/investiga_igu/igu_aproximacion_semio-cognitiva_by_chr5.pdf

[10] http://stackoverflow.com/questions/1947229/how-to-deploy-my-application-using-qt-creator