Introducción a Qt

Tzvir Darío

UTNHAEDO
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL HAEDO

¿Qué es Qt?

Qt es un framework para C++ que permite programar aplicaciones para entornos gráficos para tanto para los tres principales sistemas operativos (Windows, Linux y macOS) como para ciertos sistemas embebidos.

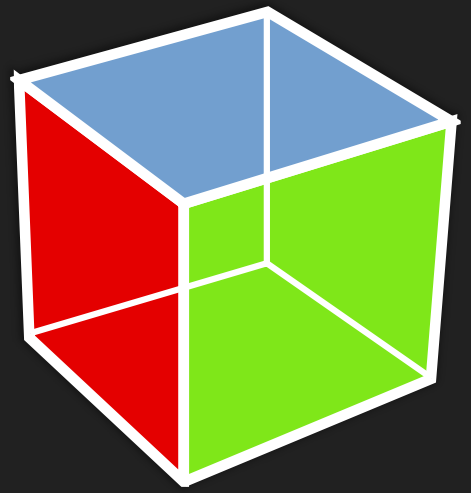
Tiene tanto una versión Open Source para uso no comercial y una versión licenciada para proyectos comerciales.

Qt se separa principalmente en QtWidgets y QtQuick.

Otros frameworks de C/C++

- JUCE: también es licenciada con una opción gratuita no comercial, está más orientada a la música y MacOS.

- GTK: tiene soporte en varios lenguajes(C, Python, JavaScript) , es completamente gratuito y es usado para Linux principalmente.



Aplicación ejemplo:

Buscaremos armar una aplicación para comunicarnos con un dispositivo MIDI, en este caso un: Arturia Minilab MK II.

Queremos:

- Ver los mensajes MIDI que nos envía el teclado, con algunas indicaciones para identificarlo correctamente.
- Cambiar las luces de los pads a cualquiera de los que el teclado tiene disponible, esto en cada memoria del mismo.

Herramientas usadas:

- Como IDE se utilizará tanto CLion como QtCreator.
- El sistema de generación será CMake.
- Usaremos los bloques básicos de la STL.
- RtMidi para generar la comunicación entre el teclado y la aplicación.
- Usaremos la librería Json de Boost para leer archivos del formato.

Nociones básicas de MIDI:

MIDI o Musical Instrument Digital Interface es un protocolo serial de 8bits que permite comunicarse con dispositivos MIDI y con programas adecuados crear música sin necesidad de que nuestro dispositivo MIDI genere ningún sonido.

El protocolo DMX para controlar luces y cañones de escenarios se basa en MIDI, (sin chequear).

Nociones básicas de MIDI:

Los mensajes son de un byte de largo y hay de dos tipos:

- Status bytes: el bit más significativo es 1 e indican que tipo de parámetro es el que enviaremos luego la información.
- Data bytes: el bit más significativo es 0 y llevan la información que comunicamos.

Nociones básicas de MIDI:

Status bytes		Data bytes	Descripción
8n	1000nnnn	0kkkkkkk 0vvvvvvv	Nota apagada
9n	1001nnnn	0kkkkkkk 0vvvvvvv	Nota encendida
An	1010nnnn	0kkkkkkk 0vvvvvvv	Presión de tecla
Bn	1011nnnn	0ccccccc 0vvvvvvv	Cambio de control
Cn	1100nnnn	0ppppppp	Cambio de programa
Dn	1101nnnn	0vvvvvvv	Presión de canal
En	1110nnnn	0vvvvvvv 0vvvvvvv	Pitch bend
F0	1111nnnn		System exclusive

¿Cómo funciona Qt?

Cuando trabajamos con una aplicación de Qt no tendremos un while(1) donde trabajo, sino que crearemos una objeto Qapplication que controlará la ejecución del programa en función a los objetos de GUI que hayamos creado.

Luego de entregar control al objeto todo el programa se basará en eventos.

¿Cómo funciona Qt?

Cosas a tener en cuenta al trabajar con Qt:

- La sección de Core nos da soporte para leer archivos, conectarse al internet, crear archivos de configuración, etc.
- Parentesco de los objetos.
- Cualquier objeto que pertenezca a la librería empezará con una Q como prefijo.
- Qstyles y Qpalette nos permite junto al sistema de herencia seleccionar colores y propiedades a toda nuestra aplicación sin cambiar cada uno de nuestros widgets.

¿Cómo funciona Qt?

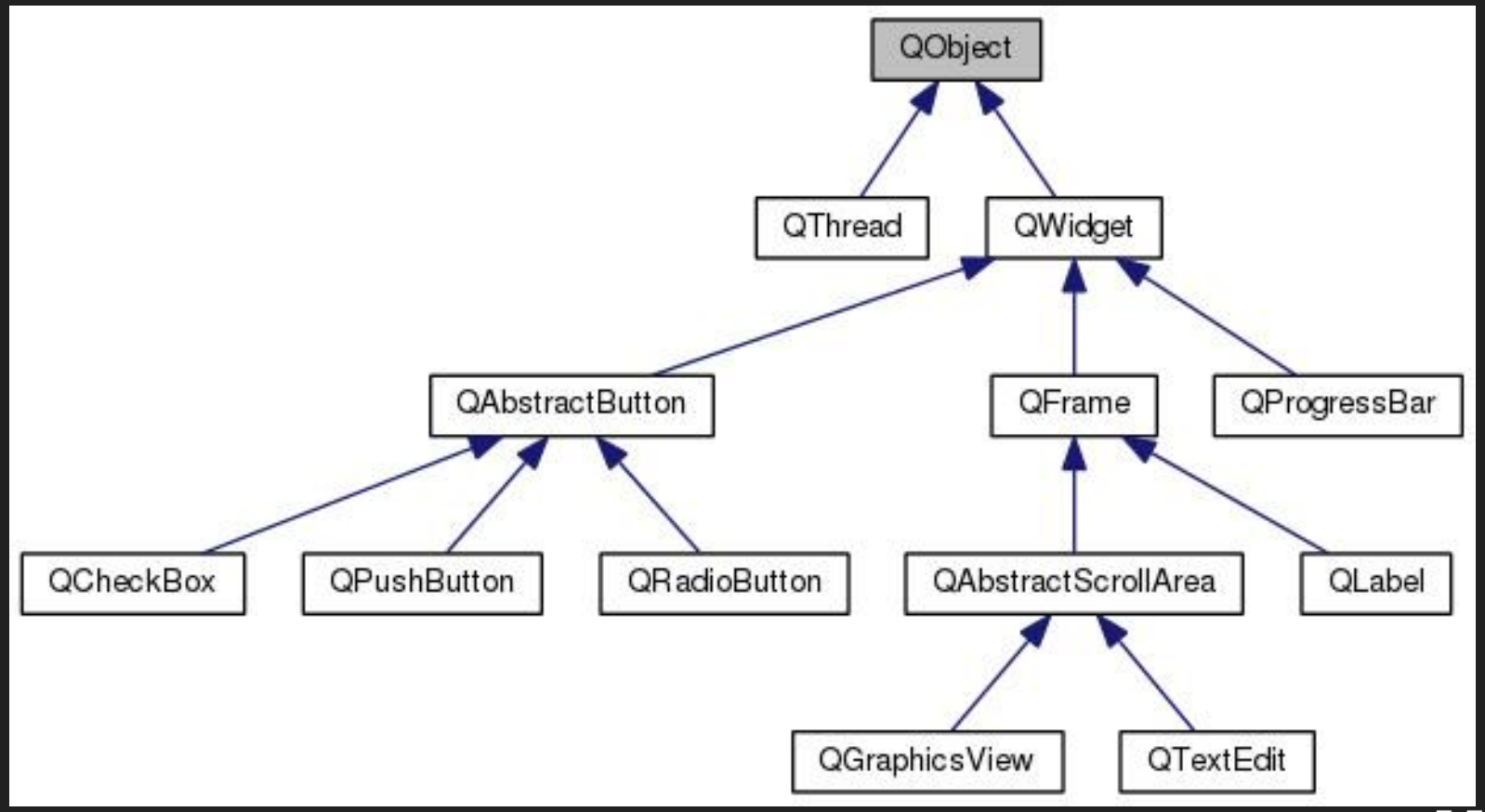
```cpp
#include <QApplication>
#include <QWidget>
#include <QPushButton>

int main(int argc, char **argv)
{
 QApplication app (argc, argv);

 QWidget window;
 window.setFixedSize(100, 50);

 QPushButton *button = new QPushButton("Hello World", &window);
 button->setGeometry(10, 10, 80, 30);

 window.show();
 return app.exec();
}
```

QtCreator:

Esta IDE nos permitirá tanto editar el código de C++ como editar con una interfaz gráfica los archivos de .UI de una manera más amigable, pudiendo conectar señales, propiedades de los objetos y atajos de teclado.

El proceso de compilación tomará nuestros archivos .UI y los convertirá en una clase que nuestro código incluirá de forma que podamos instanciarla y trabajar con ella.

Gerarquía de Qwidgets:



Señales y slots:

Las señales y slots nos permitirán comunicar dos objetos distintos.

Todos los objetos de Qt tienen señales y slots propios que nos permitirán conocer sus estados más importantes y actuar enbase a ellos.

También podemos crear señales y slots como miembros de cualquier clase que herede algún tipo de Qt, QObject por ejemplo.

Creando señales y slots:

- Una señal solo se definirá como una función de la clase y su argumento será el dato que queremos pasar. (No se necesita declarar como función, solo prototipo).
- El slot también será una función miembro y en este caso si habrá que escribir una función que manipulará el dato.

```
public slots:
    void process();
    void enviar_comando(Formatear_Midi::Mensaje msj);
    void finalizar();
signals:
    void finished();
    void error(const QString err);
    void msj_midi(Formatear_Midi::Mensaje msj);
```

Conectando una señal y un slot:

En este caso obj es un puntero a objeto, generalmente usaremos la ventana principal para conectar señales.

```
obj->connect(&obj_emisor, &tipo_obj_emisor::señal,
             &obj_receptor, &tipo_obj_receptor::slot);
```

La señal debe tener la misma o mayor cantidad de argumentos que el slot, este desestimará los argumentos sobrantes.

Trabajando con hilos:

Los hilos nos permiten tener dos funciones corriendo al mismo tiempo en el programa, esto nos habilita a separar partes del código que pueden trabar a las demás al ponerlas en sus propias tareas y tener mejor rendimiento en nuestro programa.

El problema que los hilos traen es el de sincronizar variables y la condiciones de carrera, para solucionar estos problemas se pueden usar estructuras tales como las colas o señales y slots.