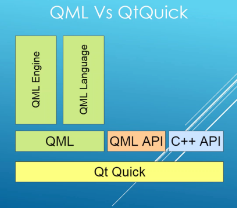
***Typedef struct vs struct***

1. Qt là gì: Qt là một Application Framework được dùng phổ biến để lập trình ứng dụng trên các hệ thống trên xe hơi và cho phép bạn phát triển ứng dụng đa nền tảng với C++ . Qt hiện nay đang được nhiều công ty lớn như Microsoft, Samsung, Bosch, FPT sử dụng để xây dựng ứng dụng

Qt Quick là một thành phần của Qt framework cho phép xây dựng giao diện ứng dụng nhanh chóng, hiệu ứng đẹp mắt. Qt Quick sử dụng ngôn ngữ kịch bản (script) gọi là QML (Qt Modeling Language).



2. Các bước biên dịch trong qt/qml

Preprocessing: Tiền xử lý mã nguồn Qt bằng cách sử dụng trình tiền xử lý của C++.

Meta Object Compiler (MOC): Tạo ra file meta object compiler (MOC): Để sử dụng các tính năng của Qt, bạn cần phải tạo ra file MOC. MOC là một trình biên dịch đặc biệt để tạo ra mã C++ cho các tính năng của Qt, chẳng hạn như QObject, Q\_PROPERTY, Q\_SIGNALS, Q\_SLOTS, vv.-> tạo ra các moc file

Compilation: Biên dịch mã nguồn C++ và mã nguồn Qt bằng trình biên dịch C->tạo ra các mã assemebly (file .obj)

Linking: Liên kết các tệp đối tượng: Liên kết các tệp đối tượng C++ và tệp đối tượng MOC để tạo ra một ứng dụng hoàn chỉnh.(tạo ra các file binary)

3. Cú pháp qml:

Rectangle {

id: topRect

width: Math.max(bottomRect.width, parent.width/2)

height: (parent.height / 3) + 10

color: "yellow"

}

- Khai báo một đối tượng QML bắt buộc bắt đầu bằng ký tự viết hoa

- Nên khgai báo trường id để cho phép đối tượng đó xác định và được gọi bởi các đối tượng khác.

- các dữ liệu cơ bản:

+ Number: Kiểu số để đại diện cho các giá trị số nguyên hoặc số thực.

+ String: Kiểu chuỗi để đại diện cho các giá trị dạng văn bản.

+ Boolean: Kiểu boolean để đại diện cho các giá trị true hoặc false.

+ Date: Kiểu ngày tháng để đại diện cho các giá trị ngày tháng.

+ Time: Kiểu thời gian để đại diện cho các giá trị thời gian.

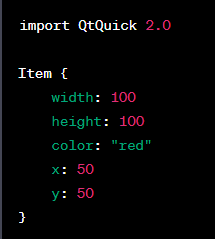
+ Color: Kiểu màu để đại diện cho các giá trị màu sắc.

+ Variant: Kiểu biến để đại diện cho một giá trị có thể có kiểu dữ liệu khác nhau.

- syntax của QML khá giống với các ngôn ngữ markup như HTML và XML, nhưng QML cũng có các đặc điểm riêng biệt để hỗ trợ việc thiết kế các giao diện người dùng tương tác phức tạp.

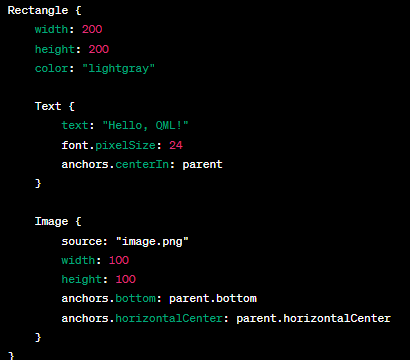
4. Các đối tượng cơ bản:

- Item là lớp base cho tất cả các đối tượng giao diện (visual item) trong Qt Quick. Mặc dù Item không có giao diện hiển thị nhưng nó lại chứa các thuộc tính cơ bản cho các đối tượng giao diện như toạ độ x, y, width, height, xử lý sự kiện với bàn phím. Trong thực tế Item hữu dụng cho việc gom nhóm các đối tượng trong QML.



- Rectangle là đối tượng giao diện cho phép tô màu đồng nhất hoặc không đồng nhất (gradients) cho một khu vực chữ nhật trên màn hình. Ngoài ra có thể vẽ thêm viền (border) hay bo góc (radius) cho hình chữ nhật. Việc sử dụng radius có thể tạo ra được hình tròn từ đối tượng Rectangle.

- Text là đối tượng cho phép hiển thị chữ viết lên màn hình, có thể lựa chọn font cũng như kích thước của chữ.

- Image là đối tượng hiển thị hình ảnh từ file hình ảnh có sẵn. Cho phép bạn cài đặt width, height để điều chỉnh kích thước của ảnh, nếu không cài đặt hai thông số trên thì ảnh sẽ hiển thị kích thước của ảnh gốc.

5. QML cung cấp cơ chế signal và slot:

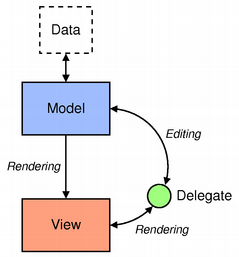
- Trong đó signal là sự kiện và slot là hàm được gọi khi sự kiện xảy ra. Khi một signal được phát ra, các slot kết nối với signal đó được thực thi. Phần này mình sẽ nói kỹ hơn trong phần sau

6. Model View and Delegates:

Trong Qt Quick người ta sử dụng kiến trúc Model View và Delegate để cho phép người lập trình tách biệt việc phát triển giao diện người dùng với xây dựng và xử lý dữ liệu.

là một mô hình thiết kế được sử dụng trong Qt để xây dựng các giao diện người dùng có khả năng mở rộng và linh hoạt. Nó dựa trên nguyên tắc phân tách quan tâm, trong đó các thành phần khác nhau đảm nhận việc quản lý mô hình dữ liệu, hiển thị và tương tác với dữ liệu

- Là mô hình rút gọn của MVC

Như trong MVC thì model: đại diện cho data. Trong Qt/QML, model thường được thực hiện bằng cách sử dụng framework Model-View của Qt(list view, path View, gird view, .....) hoặc tạo bằng C++. Là nơi xây dựng dữ liệu quản lý trạng thái của giao diện, thể hiện các dữ liệu cho việc xây dựng trạng thái của màn hình

view hiển thị dữ liêu người dùng. View được thực hiện bằng các thành phần QML, định nghĩa giao diện người dùng hiển thị và hành vi của nó, quản lý màn hình, các component

Delegate quản lý cách hiển thị và chỉnh sửa dữ liệu trong View. Delegates sẽ lấy từng dữ liệu trong model và đóng gói nó. Dữ liệu được truy cập thông qua các index.

Controller: hoạt động trung gian giữa model và view, đảm bảo rằng hai thành phần này là độc lập và có thể được sửa đổi độc lập nhau. được thực hiện bằng các xử lý tín hiệu và khe cắm QML, xử lý đầu vào từ người dùng và cập nhật model và view tương ứng.(thường phần này nằm dưới c++)

- Vì vậy, trong khi Qt/QML sử dụng mô hình MVD để xây dựng các giao diện người dùng, nó cũng tích hợp mô hình MVC để quản lý, dữ liệu và tính logic của ứng dụng.

7. Animation and Transform

Trong Qt, Animation là một cơ chế được cung cấp để tạo hiệu ứng chuyển động trên các đối tượng trong giao diện người dùng. Animation cho phép thay đổi các thuộc tính của đối tượng trong suốt thời gian nhất định để tạo ra các hiệu ứng chuyển động mượt mà và đẹp mắt.

Animation thường được sử dụng để tạo hiệu ứng chuyển động trong các ứng dụng desktop, mobile, hay các ứng dụng web. Ví dụ như khi một button được click, ta có thể sử dụng animation để làm cho button thay đổi kích thước hoặc màu sắc, tạo ra hiệu ứng chuyển động mượt mà.

**Video ví dụ:**

Qt có hỗ trợ các loại animation như, mình xin kể 1 vài loại mình hay sử dụng:

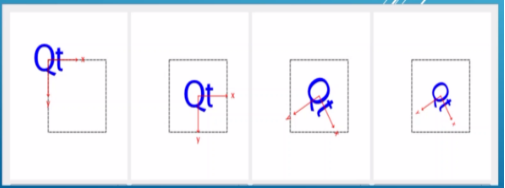
- RotationAnimation: Cho phép thay đổi góc quay của một đối tượng.

- SequentialAnimation: Cho phép chạy một chuỗi các animation theo thứ tự tuần tự.

- ParallelAnimation: Cho phép chạy một chuỗi các animation song song.

Ngoài ra còn nhiều loain animation khác nữa, các bạn có thể tham khảo trên documents của qt: **https://doc.qt.io/qt-6/qml-qtquick-animation.html**

- Trong Qt/QML, Transform là một lớp được sử dụng để thay đổi vị trí, kích thước, định hướng và hình dạng của các đối tượng trong giao diện người dùng. Transform được sử dụng để thực hiện các phép biến đổi hình học trên các đối tượng, bao gồm di chuyển, xoay, co giãn và bóp méo.



8. Lớp Qobject

Lớp QObjects là lớp cha (base class) của hầu hết các lớp đối tượng khác trong QT. Ngoại trừ các kiểu có giá trị như QColor, QString and QList.

Trong QT, QObjects được tổ chức theo mô hình object trees. Khi tạo ra một đối tượng QObject mới với một đối tượng khác là lớp cha, đối tượng mới sẽ được tự động thêm vào danh sách các children() của đối tượng cha.

Qobject là trái tim của Qt Qobject Model. Tính năng của mô hình này là cơ chế rất mạnh mẽ để giao tiếp với các đối tượng liền mạch được gọi là signal slot. Bạn có thể kết nối một signal slot bằng connect() và hủy kết nối bằng disconnect(). Để tránh vòng lặp thông báo vô hạn, bạn có thể tạm thời chặn tín hiệu bằng cách sử dụng hàm blockSignals (). Các hàm bảo vệ connectNotify () và disconnectNotify () cho phép theo dõi các kết nối.

9. Signals và slots

được dùng để giao tiếp giữa các Object với nhau. So với các framework khác thì đây là một cơ chế khác biệt mà QT cung cấp.

Signal được bắn ra bởi một item trên QML hoặc một đối tượng ở C++ khi có 1 sự kiện nào đó xảy ra. (Ví dụ như sự kiện click chuột hoặc một giá trị nào đó thay đổi.)

Slot là một hàm xử lý sự kiện mà Signal bắn ra. Các hàm này được khai báo dạng public ở C++

Để sử dụng Signal và Slot trong QT, ta phải tạo ra 1 class kế thừa từ QObject class. Class này phải bắt buộc phải định nghĩa macro Q\_OBJECT

Một signal có thể kết nối với nhiều slot và một slot cũng có thể kết nối với nhiều signal. Tuy nhiên không nên sử dụng cách kết nối một slot với nhiều signal.

Signal: chỉ có khai báo k có định nghĩa

Slot: có thân hàm

Để tạo kết nối giữa signal và slot có 2 cách: String-base và Functor-base

Cú pháp:

MyObject oject1, oject2;

- String-base:

QObject::connect(&oject1, SIGNAL(valueChanged(int)), &oject2, SLOT(slotValueChanged(int)));

- Functor-base:

QObject::connect(&oject1, &MyObject ::valueChanged, &oject2, &MyObject::slotValueChanged);

Ngoài ra Cách thứ ba để tạo kết nối giữa signal và slot trong Qt là sử dụng lambda expression:

QObject::connect(&object1, &MyObject::valueChanged, &object2, [](int value) {

qDebug() << "Value changed to" << value;

});

10. Iteractive c++ qml

- Kết nối signal và slot giữa các đối tượng C++: đã trình bày trước đó

- Kết nối signal và slot trên QML:

+ Để khai báo một signal trên QML bạn sử dụng cú pháp: signal <name>[([<type> <parameter name>[, ...]])]

Ví dụ: signal clickMyRectangle()

signal saveData(int a, int b)

+ Để tạo slot trên QML, bạn sử dụng cú pháp: on + <tên signal>.

Ví dụ: tên signal là clickMyRectangle() thì slot tương ứng là onClickMyRectangle

+ Nếu bạn có một property được khai báo dạng: property <type> <tên property> thì slot tương ứng với nó sẽ có dạng on<tên Property>Changed. Slot này sẽ được tự động xử lý khi giá trị của property thay đổi.

- Tạo một đối tượng QML bằng C++:

+ Tạo một đối tượng QML mới từ đối tượng C++ sử dụng qmlRegisterType.

+ Sử dụng setContextProperty để có thể đưa các dữ liệu của một đối tượng C++ lên QML và sử dụng.

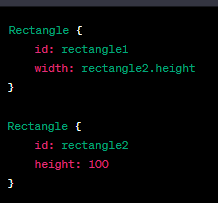
🡪 Mục đích của việc tạo một đối tượng QML bằng C++ là để mở rộng các tính năng của ứng dụng QML bằng cách sử dụng các thành phần được xây dựng bằng C++. Khi sử dụng C++ để tạo các đối tượng QML, bạn có thể tận dụng các tính năng mạnh mẽ của ngôn ngữ lập trình này để tạo ra các thành phần phức tạp và hiệu suất cao hơn so với việc sử dụng QML đơn thuần.

bạn có thể sử dụng C++ để tạo ra các đối tượng QML có tính năng xử lý video, đồ họa 3D hoặc chức năng liên quan đến dữ liệu lớn. Bằng cách sử dụng C++ và QML cùng với nhau, bạn có thể tận dụng những ưu điểm của cả hai ngôn ngữ để tạo ra các ứng dụng QML đa năng và hiệu quả.

hường thì việc tạo đối tượng bằng C++ sẽ nhanh hơn so với tạo đối tượng từ QML. Nguyên nhân chính là do C++ là ngôn ngữ lập trình được biên dịch, có khả năng tối ưu hóa mã máy và chạy nhanh hơn so với QML, một ngôn ngữ lập trình được thông dịch. Do đó, khi tạo đối tượng bằng C++, bạn có thể đạt được hiệu suất cao hơn trong một số trường hợp so với tạo đối tượng bằng QML.

- Cách binding trong QML và sử dụng:

binding là cách để khai báo một mối quan hệ giữa hai thuộc tính, sao cho khi một thuộc tính thay đổi, thuộc tính khác sẽ tự động cập nhật để phản ánh sự thay đổi đó. Điều này hữu ích khi bạn muốn giữ cho hai thuộc tính đồng bộ, mà không cần phải cập nhật chúng bằng tay mỗi khi một trong chúng thay đổi.



- Ngoài ra chung ta thường gặp các MACRO như:

+ Q\_OBJECT

+ Q\_INVOKABLE

+ Q\_PROPERTY

11. Tạo model C++ trên QT

- tạo model từ list View trên qml

+ Nếu mà model của mình của rất nhiều đối tượng, thì trên qml sẽ rất dài vì thế tốc độ xử lý sẽ rất chậm vì bản chất qml sẽ xử lí bằng java script. Và tốc độ qml thường chậm hơn tốc độ xử lí của c++. Do đó để cần tạo model xử lí nhanh và dẽ nhìn thì ta sẽ sang cách thứ 2 là tạo model từ c++

- tạo Model cho ListView từ C++ sử dụng QStringList và QVariantList.: danh sách Qstring, Qvariant là 1 kiểu dữ liệu dùng được với nhiều kiểu dữ liệu khác nhau được QT support

- Sử dụng QbjectList: 1 danh sách các đối tượng có thể tạo thành model từ C++

- Sử dụng lớp kế thừa từ QabstractListModel: Giới thiệu về QabstractListModel là mô hình trừu tượng có thể phân lớp để tạo mô hình danh sách 1 chiều, đc ké thừa từ QabstractItemModel. Khi tạo lớp kế thừa pahir ghi đè method rowCount() và data(). roleNmaes().

11. Multi Threading

Kể từ khi các hệ điều hành đa nhiệm ra đời, một máy tính có thể chạy đồng thời nhiều chương trình. Các chương trình cũng được tối ưu hơn nhờ cơ chế xử lý multi-processing và multi-thread. Hai cơ chế này tối ưu hóa được số processor mà CPU cung cấp.