**Qt Remedial Class 정리**

ppt를 기본으로 해서 제작했습니다. 함께 순서대로 보면 될 것 같습니다. ppt 슬라이드의 부연설명이라도 생각해도 좋을 것 같습니다.

**<1번 자료>**

**새 프로젝트 생성**

* Qt Widgets Application -> kit 선택(하라는 대로 했으면 하나만 있을거에요 그거 선택) -> 클래스명 입력

**프로젝트 내에 생성된 파일**

* PROJECT\_NAME.pro: 프로젝트 설정 파일, 일종의 makefile. 새로운 클래스를 생성하게 되면 알아서 이 파일에 새로 생성된 내용이 추가된다.
* WINDOW\_CLASS.ui: 윈도우에 대한 UI

**main.cpp**

* 초기 생성 시 보이는 코드는 18번 슬라이드와 같다.
* QApplication a(argc, argv); <- 이건 건드리지 말자
* w.show(); <- 이건 w 라는 창을 UI로 보여주는 메소드.
* return a.exec(); <- main 함수의 return 0; 와 비슷하다.(프로그램 종료)

**mainwindow.h**

* Q\_OBJECT <- 매크로. 이게 있어야 slot을 인식한다고 한다.
* 클래스 선언이 이루어지는 부분이다. 19번 슬라이드에서 볼 수 있듯이 메인 창은 QMainWindow 클래스를 상속받는다.

**mainwindow.cpp**

* ui->setupUi(this); <- 이 함수를 constructor에 선언해야 만든 UI가 생성된다.

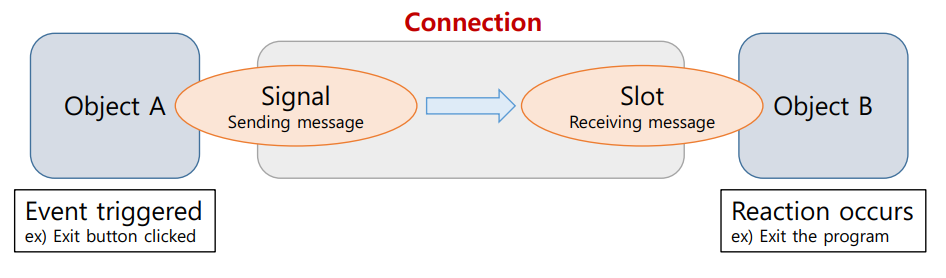
**ui 파일(여기서는 mainwindow.ui)**

* 드래그 앤 드롭 방법으로 버튼과 같은 위젯을 추가 가능.
* 맨 왼쪽의 Design 부분을 클릭하면 드래그 앤 드롭 방법으로 가능하나 Edit 부분을 선택하면 코드를 보는 것도 가능(수정은 Design 부분에서만 가능)

드래그 앤 드롭 방법으로 버튼을 넣고 버튼에 나타나는 text를 마음대로 바꿔서 좌측 하단의 초록 삼각형 버튼(run 버튼, ctrl+R 로도 수행 가능)을 눌러서 실행하면 버튼이 있는 창이 뜬다. 클릭해볼 수 있지만, 버튼을 클릭했을 때(signal) 무엇을 할 지 지정하지 않았기 때문에 아무런 일도 일어나지 않는다.

**중요! 23 페이지의 내용을 알아야 어싸인 진행에 있어서 수월합니다.**

23페이지의 내용은 Signal & Slot에 대해서 설명한다. 간단히 생각하면 신호와 행동을 이어주는 것이다. 예를 들어서 설명해보자면, mainwindow 객체에 있는 버튼이 클릭이라는 신호를 받으면, 해당 객체를 종료하는 행동을 할 수 있다. 혹은 다른 객체의 창을 띄우는 행동을 할 수도 있다.



**이미지에서 보듯 Object가 특정 조건이 만족되어(버튼의 클릭 등) signal을 보내면, slot이 그 signal을 receive하여 다른(같아도 된다) Object의 행동(프로그램의 종료 혹은 다른 객체의 창 띄우기 등)으로 이어지게 된다.**

**Slot**

* UI 창에서 위젯을 우클릭 후 go to slot을 선택, 반응하고자 하는 signal(예제에서는 클릭)을 선택한다. 그러면 헤더와 .cpp 파일에 해당 signal에 대한 함수가 생긴다. 무엇을 할 지 적어넣자.
* exit(0); <-프로그램 종료 함수
* exit 함수를 아까 말한 정의 부분에 넣고 프로그램을 실행하면, 버튼을 누르는 신호를 받으면 프로그램을 종료하는 행동을 하게 된다.

**새 창 띄우기**

* 먼저 새 창을 띄울 객체를 만든다. 죄측의 projects에서 sources 폴더를 우클릭, add new를 선택한다. Qt를 선택하고 Qt Designer Form Class 선택. 상황에 따라 적절한 템플릿을 선택한다. 예제에서는 without buttons 선택. Class 이름을 정하고 어느 project에 넣을지 정하면 끝.
* 새로운 class를 형성하면 .pro 파일에 자동으로 추가된 class에 대한 내용이 추가된다.

Dialog\* dialog = new Dialog(this);

dialog->show();

* Dialog라는 객체를 동적할당으로 할당하고, show 메소드를 이용해 창을 띄운다.
* 조금 전 구현한 함수에서 exit 부분에 위의 코드를 넣으면 버튼을 누르면 새로운 창이 뜨게 된다. 새로운 창을 띄우고 기존의 창을 숨기고 싶다면 hide() 메소드를 사용한다.

**QTimer**

* 일정 시간 간격으로 timeout() 시그널을 보낼 수 있는 클래스

QTimer\* timer = new QTimer(); <- 이 내용 선언은 주로 class의 constructor에서 한다.

connect (객체 A (timer), SIGNAL(timeout()), 객체 B, SLOT(FUNTION()));

timer->start(2000); <-ms 단위

* 예시 코드 : connect(timer, SIGNAL(timeout()), this, SLOT(Timeout()));
* 헤더 파일의 Class 정의에 public slots에 함수의 형태로 슬롯 정의. 예시 코드 참고.

public slots:

void FUNCTION();

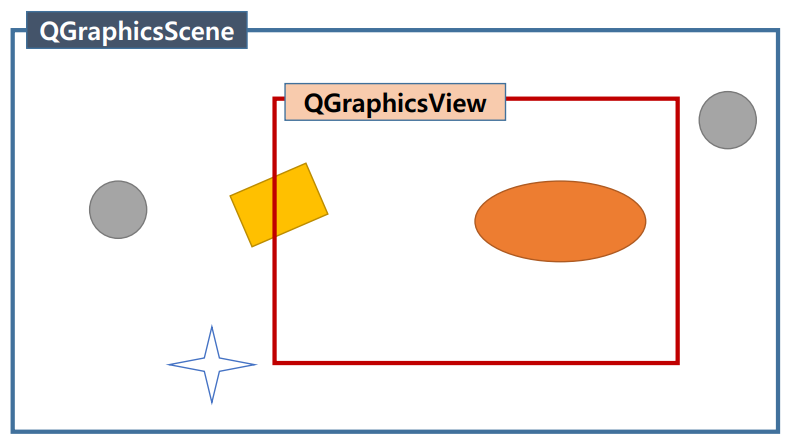
* 이번 예제에서는 Timeout()이 FUNCTION에 해당한다.
* 혼동될 수 있는데, timeout(소문자 t)은 기본 제공 함수고, Timeout(대문자 T)은 사용자가 헤더 파일에서 정의한 함수이다. 즉, 객체 A(여기서는 timer)가 timeout()이라는 signal을 보내면, 객체 B(여기서는 this, 즉 MainWindow)가 그것을 인식해 Timeout 함수를 수행한다.

이 이후(QMessageBox 부터)로는 실습을 하지 않아 ppt를 그냥 보시길 추천드려요~

**<2번 자료>**

**QGraphicsScene, QGraphicsView**

* QGraphicsScene: 그래픽 요소들이 배치된 하나의 공간
* QGraphicsView: Scene 에서 실제로 유저의 화면에 그려지는 부분 (유저가 확인할 수 있는 View)



**QGraphicsItem**

* QGraphicsScene에 들어갈 수 있는 객체 클래스이다.
* QGraphicsRectItem, QGraphicsEllipseItem 등의 하위 클래스 존재하고, 우리는 예제에서 직사각형을 이용하기 위해 QGraphicsRectItem을 사용했다.

**5 슬라이드의 예시 코드**

* 붉게 표시된 부분은 순서대로 QGraphicsScene 객체를 scene이라는 이름으로 동적할당-> QGraphicsRectItem 객체를 item이라는 이름으로 동적할당 -> item 의 크기 지정(두 꼭짓점의 위치로 크기를 정하며 좌상단이 (0, 0) 이다.) -> scene 인스턴스에 item 이라는 Item(대소문자 구별 잘 하자)을 addItem이라는 메소드로 추가 -> 이후 view를 동적할당하고 show() 메소드로 창을 띄운다.
* 프로그램을 실행하면 UI 화면에는 한 변이 100인 (아마 단위는 pixel일거에요) 정사각형이 보일 것이다.

**캐릭터 클래스 생성**

* 앞서 언급한 방법을 통해 새로운 class를 생성하되, 단 이번에는 Qt에서 선택하는 것이 아니고 C++ 템플릿에서 C++ Class 를 선택한다. 생성 시 이름은 Player로 하자.
* 클래스가 생성되면 헤더 파일에서 클래스를 정의할 때 public으로 class의 모양에 맞는 QGraphicsItem의 하위 클래스를 상송받는다. 이번에는 직사각형 모양의 player를 만들 것이므로 QGraphicsRectItem을 상속받는다.
* 이후, main.cpp에서 Player 클래스에 대해서 새로운 인스턴스를 player라는 이름으로 동적할당 받는다. 그러면 앞서 작성한 예시 코드와 거의 같게 된다. item을 player로 바꾸면 된다.

**QKeyEvent**

* 키보드의 입력을 받는 클래스이다.
* 10 슬라이드의 예시 코드를 보면 **keyPressEvent** 라는 메소드가 보일 것이다. 이는 QGraphicsItem의 virtual 함수이고 키 입력에 대한 이벤트 처리를 진행한다.
* 예시에서는 아무 키든 입력을 받으면 메시지를 콘솔 창에 출력하게 된다.
* qDebug() 함수가 cout 역할을 한다고 생각할 수 있다.
* 하지만 아직은 아무 일도 일어나지 않는데, 이는 바로 뒤에 설명할 Focusable과 관련이 있다.

**Focusable**

* Focus 설정을 하지 않으면 키를 입력해도 아무 일이 일어나지 않는다. 즉, Focus 설정을 해야 키를 입력했을 때 반응을 출력할 수 있다. Focus 설정을 하면 키보드를 누를 때마다 메시지를 출력할 수 있다.

player->setFlag(QGraphicsItem::ItemIsFocusable);

player->setFocus();

* 위의 예문 코드를 보자. 처음의 setFlag는 키를 입력받을 후보를 정하는 메소드이다. 즉, player->setFlag를 진행함으로서 player 객체를 키를 입력받을 수 있게 하는 후보군에 추가하는 것이다.
* 하지만 아직은 후보에 추가하는 것에 지나지 않았다. setFocus 메소드를 이용하면 앞서 추가한 후보군 중 키를 실제로 입력받을 객체를 정할 수 있다. 즉, 이 과정을 통해 player 객체를 드디어 키를 입력받을 수 있게 한 것이다.

**특정 키가 눌렸는지 확인**

* QKeyEvent에는 key() 라는 메소드가 있다. 이를 통해 현재 눌린 key에 대한 정보를 알 수 있다. key() 메소드는 입력받은 키의 정보를 가진다.
* 12 슬라이드의 예제 코드를 보면 <- 키와 -> 키에 대해 행동을 수행한다.
* 예제의 **setPos** 는 Item의 좌표를 설정하는 함수이다. 함수 안에 x()와 y()가 있는데, 이들은 객체의 x 좌표와 y 좌표를 의미한다. 기본적으로 초기화를 해주지 않으면 (0, 0)으로 설정된다.

이후 내용은 지금까지 설명한 내용의 보충입니다. Bullet 객체를 새로 생성해서 역시 QGraphicsRectItem을 상속한다. 단, constructor에서 setRect 메소드를 이용해 생성과 동시에 그려지게 한다. **(슬라이드 14)**

이후 스페이스바를 누르면 총알이 화면에 나타나게 한다. 예제 코드를 보면 player.cpp 에서 Bullet 객체를 bullet 이라는 이름으로 새로 동적할당한 뒤 addItem을 이용해 scene에 추가하는 모습을 볼 수 있다. **(슬라이드 15)**

여기까지는 총알이 새롭게 화면에 나타나기만 하지만, 앞서 배운 QTimer를 이용하면 총알이 움직이게 할 수 있다. Bullet 클래스를 QObject라는 클래스를 하나 더 상속받아 다중상속을 받게 한다. Q\_OBJECT라는 매크로를 헤더에 선언한다. 앞서 말했듯이 이 매크로가 있어야 slot을 인식한다. 예제 코드를 보면 알 수 있듯이 timeout 이라는 signal에 대해 move라는 행동을 진행할 것이기 때문에 헤더의 slots에 move 메소드를 선언하고 cpp 파일에서 정의한다. 참고로 QGraphics에서는 y 감소는 위쪽 방향이다. **(슬라이드 16)**

컴파일 과정 중 vtable 에러가 생기면 .pro 파일의 bullet과 player의 순서를 바꿔보자. **(슬라이드 17)**

여태까지는 스크롤바가 생성되고 화면 크기를 고정할 수 없었지만, 수정할 수 있다. QGraphicsView 클래스의 메소드인 setHorizontalScrollBarPolicy(Qt::ScrollBarAlwaysOff) 와 setVerticalScrollBarPolicy(Qt::ScrollBarAlwaysOff) 를 이용하면 각각 가로 / 세로 스크롤바를 제거할 수 있다. 또한 setFixedSize 메소드와 setSceneRect 메소드를 이용하면 view와 scene의 크기를 지정할 수 있다. **(슬라이드 19)**

Bullet 객체가 화면(view) 밖으로 이탈하는 경우, 소멸시킬 수 있다. Bullet 클래스의 move 메소드를 선언할 때, y()가 0보다 작으면 removeItem 메소드와 delete를 이용해 인스턴스를 소멸시키는 것으로 구현할 수 있다. **(슬라이드 21)**

이 이후(Enemy 생성부터)는 실습 없이 설명만 했기 때문에 내용 추가는 하지 않겠습니다.