

C++

Forelesning 11, vår 2014
Alfred Bratterud

I dag:

- * Prøve og Prosjektoppgave
- * Kort repetisjon, design patterns og idiomer
- * Qt
- * ...takk for i år! (så langt)

Avsluttende prøve!

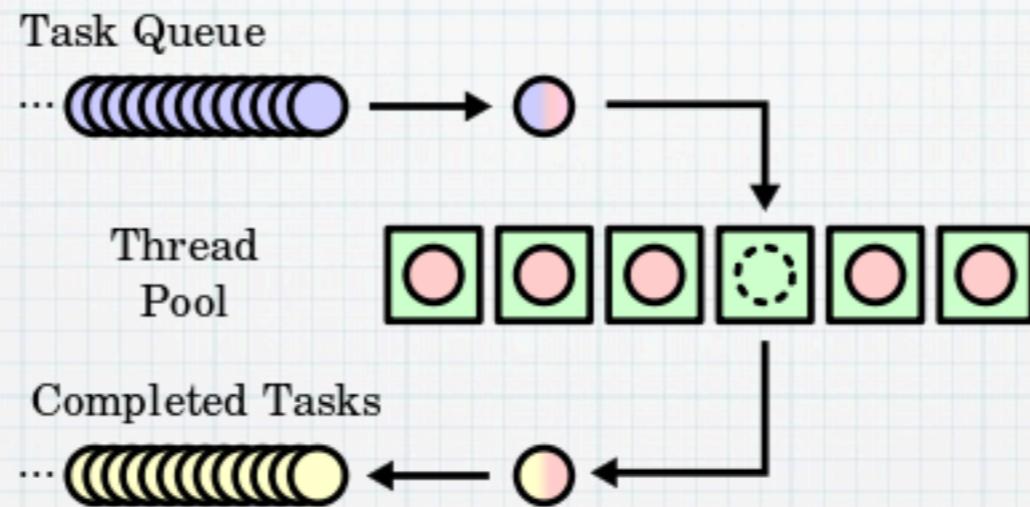
- * Prøve fredag 11.april (OBS: Feil dato i tidligere slides!)
- * Rom og klokkeslett; har bestilt PH422 og PH424, kl.08:30 - 10:15. (Venter endelig bekreftelse)
- * Pensum:
 - * Alt som er gått igjennom på forelesning - både slides, eksempelkode
 - * Bakgrunnsstoff fra læreboken:
 - * Særlig Appendix A, "Language summary"
 - * Alt som er gitt i oppgaver frem tom. 13.april
- * 90 min. - ingen pauser, ingen hjelpebidr. (ingen hjelpebidr.)
- * Noen flervalgsspørsmål, noen korte fritekstspørsmål
- * Tips: Gjør oppgaver! Hver gang du er usikker på noe - slå det opp, og prøv det ut isolert til du vet du skjønner det.

Prosjektoppgave

- * link til nytt, privat Github-repo for gruppen og meg med **oppgaveforslag** i, må leveres på fronter innen **søndag 06.april kl.23:59.** (senere innlevering spiser av tiden deres)
- * Prosjektbeskrivelsen må ha med:
 - * Beskrivelse av hovedfunksjonaliteten
 - * Tydelige designmål (Utvidbart? skalarbart? Effektivt? Integrerbart? Modulært?)
 - * Avgrensning: Noen minimumsmål, og noen hårete mål
- * Prosjektoppgaven må godkjennes av meg - gjøres i github
 - * Etter dette kan dere booke veiledning med meg
- * **Innleveringsfrist (hard): 04.mai 2014 kl.23:59,** da klones alle repoer.
- * Dere skal presentere prosjektet deres for meg, etter fristen.

Sist: Design patterns og idiomer

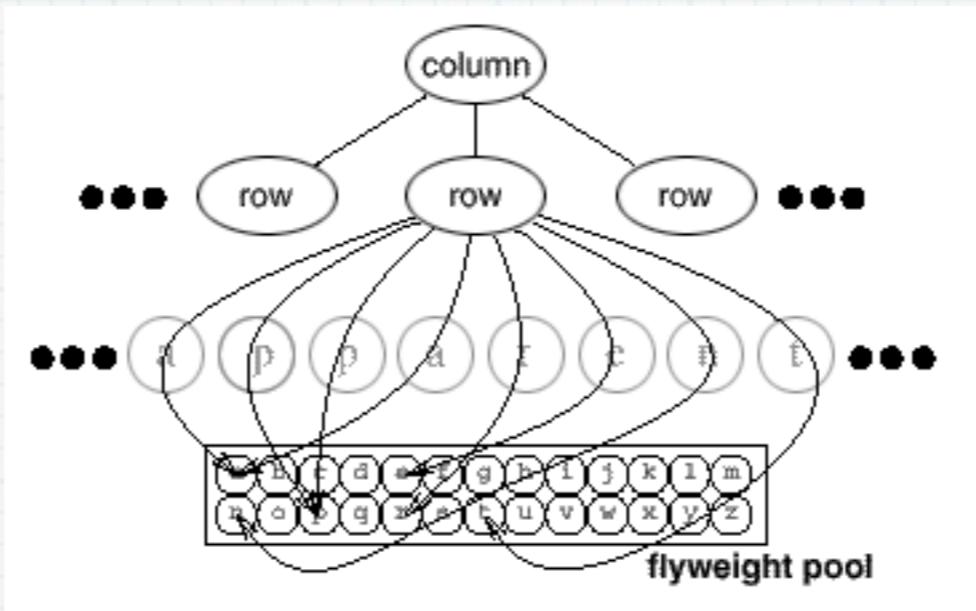
Patterns: ikke bare “convenience”



- * “Thread-pool” er ett eksempel på pattern som effektiviserer kode
- * Tråde er “dyre” å opprette, så vi tar vare på dem.

http://en.wikipedia.org/wiki/Thread_pool_pattern

Flyweight



- * Her har "row" pekere til de samme "bokstavene" mange ganger
- * Vi har kun 26 objekter, men vi kan ha millioner av bokstaver
 - * For alle får vi alle fordelene av objektorientering
- * Eks: Vi kan merke én gal bokstav i et ord, ved å kalle `letter(row,col)->mark_error()`
- * Brukes gjerne sammen med factory/multiton:
`letter_factory.get('a')` gir en "letter*", oppretter hvis den ikke finnes fra før.

Bjarne Stroustrup's RAII

- * Husker dere "Exception leak?"
 - * Anta at vi allokerer minne, eller en hvilken som helst ressurs; en fil, et socket, en mutex
 - * Før vi rekker å slippe den (release) kastes en exception. Oops! Lekkasje!
- * Løsning: "Resource Acquisition Is Initialization"
 - * "Å anskaffe en ressurs er å initialisere".
 - * ...ikke av seg selv, men vi kan bruke et "pattern", eller et "idiom")
 - * Nøkkel: I C++ destrueres alle variabler når de går ut av skop
 - * Løsning: For hver ressurs du skal anskaffe, lag en klasse som anskaffer den i konstruktøren.
 - * Og la destruktoren "avskaffe" / deallokere / lukke.

Demo: f10 - VIKTIG

(exception_leak.cpp)
exception_leak2.cpp
raii.cpp

Hva er ?

- * Et portabelt C++ -rammeverk
 - * Fokus på GUI-utvikling, men også nettverk, tråder etc.
 - * MÅL: En kodebase over alt. Stor fokus på mobile enheter nå.
- * En velutstyrt IDE: Qt Creator
 - * Qt Creator er uavhengig av rammeverket
- * Ikke (helt) standard-compliant
 - * kan bruke "din kompilator" (gcc/clang etc.) men koden må preprosesseres ekstra pga. utvidelser av språket (mer om det)

Demo!

WYSIWIG Intro, Hello World
(ingen kode, bare gjennomgang)

Litt historie



- * En norsk oppfinnelse, laget i 1995 av norske "Trolltech"
- * På børs i 2006 (!)
- * Trolltech kjøpt av Nokia i 2008
- * Solgt til Digia i 2011, da Nokia inngikk strategisk partnerskap med Microsoft om windows phone

Qt i dag



- * Største multiplatform utviklingsmiljø for C++
 - * Støtte for windows og linux fra starten
 - * Native støtte for Mac i mange år
 - * Offisiell støtte for Android og iOS fra i år (!!)
- * Hovedfokus: En kodebase over alt
- * Brukes av:
 - * Skype, Ubuntu, Mathematica, KDE, Google Earth
+++
- * Bla. integrasjon med WebKit gjør det pinlig enkelt å lage en browser.

Qt Creator



- * En velutstyrt IDE
- * WYSIWYG drag'n drop GUI editor
- * Bruker ulike kompilatorer; Gcc, Clang, Visual Studio
- * Lager ferdige installere for alle støttede platfromer
- * Innebyget integrasjon med git

Qt quick v.s. Classic



- * Som "Standard" lages Qt desktop applikasjoner på klassisk "widget" måte
- * I februar i fjor kom Qt 5, og hovedfokus er nå Qt quick
 - * HTML-liknende språk, QML, for å skrive GUI-elementer, erstatter tidl. XML
 - * Alle GUI elementer styres med JavaScript (!)
 - * 100% lagdeling mellom GUI og kode
 - * API-kommunikasjon mellom gui og C++
 - * Støtte for multi-touch, OpenGL-effekter etc.

Qt Rammeverket



- * Grafikkbiblioteker som skriver all grafikk fra scratch, direkte til skjerm
- * Meget omfattende bibliotek, med alt fra QString til sockets, til webkit objekt
- * Kan fint brukes frittstående, utenfor Creator
- * Men, mye er basert på makro-utvidelser av språket



Slots & Signals

- * En løsning for å lage komponenter
- * Slot: en metode som kalles av et "Signal"
- * Signal: en metode som kaller alle slots - og som vi ikke implementerer direkte
- * Slots og Signals kobles sammen med `QObject::connect(sender, SIGNAL(sig), receiver, SLOT(slot))`:
- * Et signal som skal kobles til et slot må ta samme argumenter.
- * Argumentene fra "signalet" sendes til argumentene for "slot"
- * "Signalet" er egentlig bare et kall på alle slots det er koblet til



Slots & Signals

```
class MainWindow : public QMainWindow
{
    Q_OBJECT

public slots:
    void updateTextCount();

signals:
    void textCountChanged(int i);

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

private:
    Ui::MainWindow *ui;
};
```

Q_OBJECT er en macro som gir
klassen bla. signals & slots.
Forutsetter arv av QObject



Slots & Signals

```
class MainWindow : public QMainWindow
{
    Q_OBJECT

public slots:
    void updateTextCount();

signals:
    void textCountChanged(int i);

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

private:
    Ui::MainWindow *ui;
};
```

Q_OBJECT er en macro som gir klassen bla. signals & slots.
Forutsetter arv av QObject

"slots" ser ut som et C++-keyword og er en "utvidelse av språket"



Slots & Signals

```
class MainWindow : public QMainWindow
{
    Q_OBJECT

public slots:
    void updateTextCount();

signals:
    void textCountChanged(int i);

public:
    explicit MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private:
    Ui::MainWindow *ui;
};
```

Q_OBJECT er en macro som gir klassen bla. signals & slots.
Forutsetter arv av QObject

"slots" ser ut som et C++-keyword og er en "utvidelse av språket"

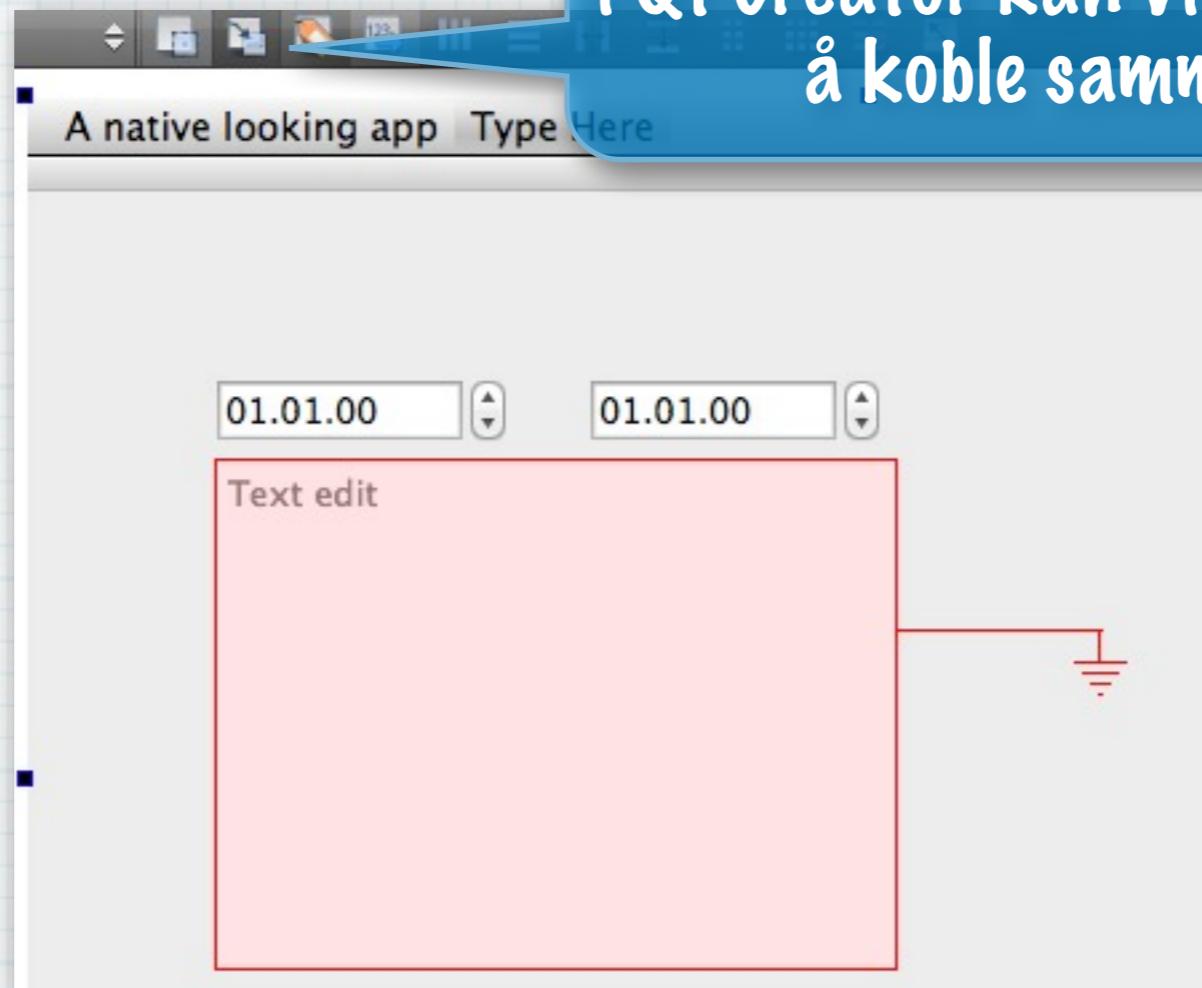
"signals" kan "kobles til" "slots" som tar kompatible argumenter

Tilkobling gjøres enten i GUI-editor eller via "connect(...)" i QObject



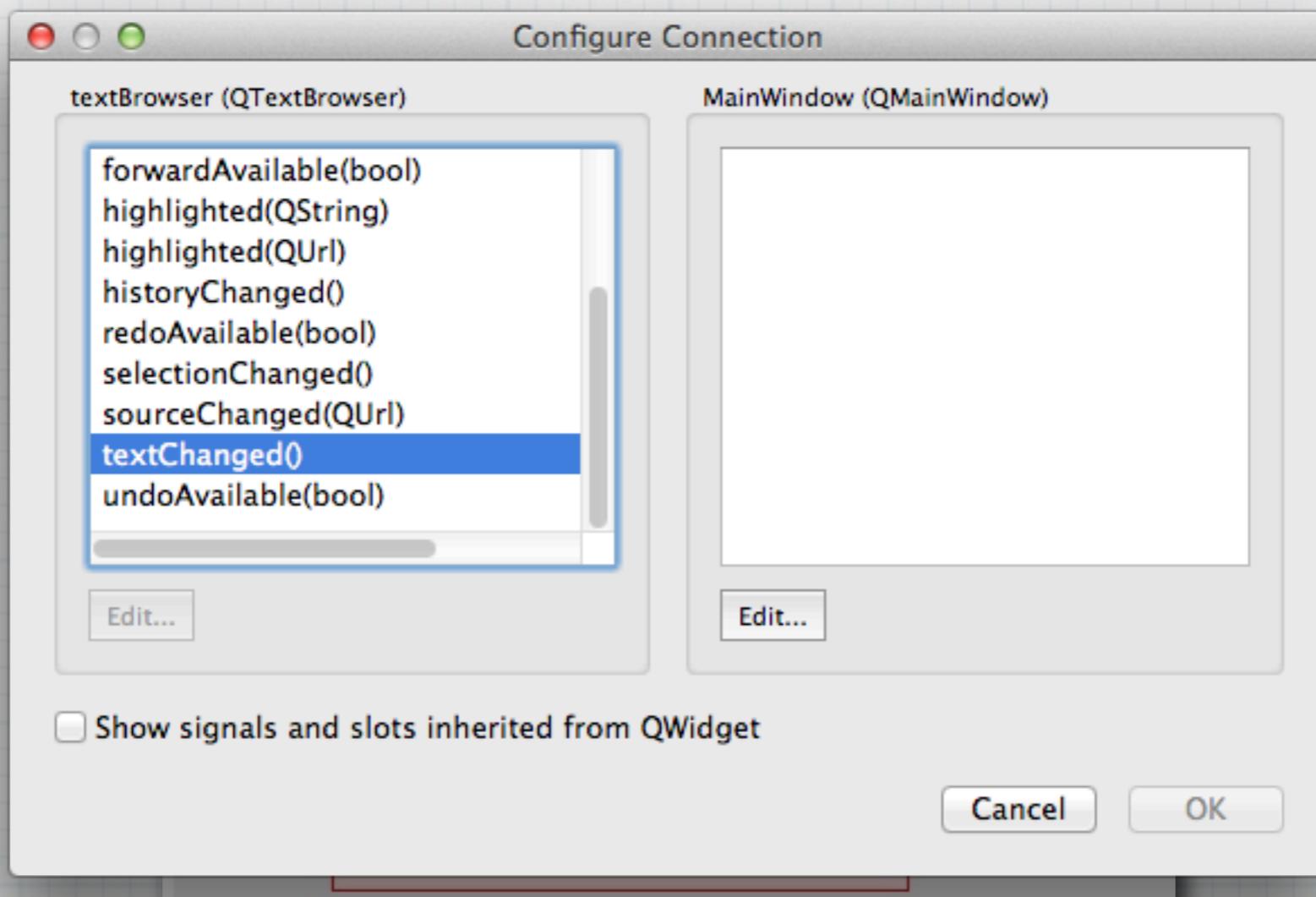
Slots & Signals: tilkobling

i Qt Creator kan vi bruke "click'n drag" for
å koble sammen slots/signals





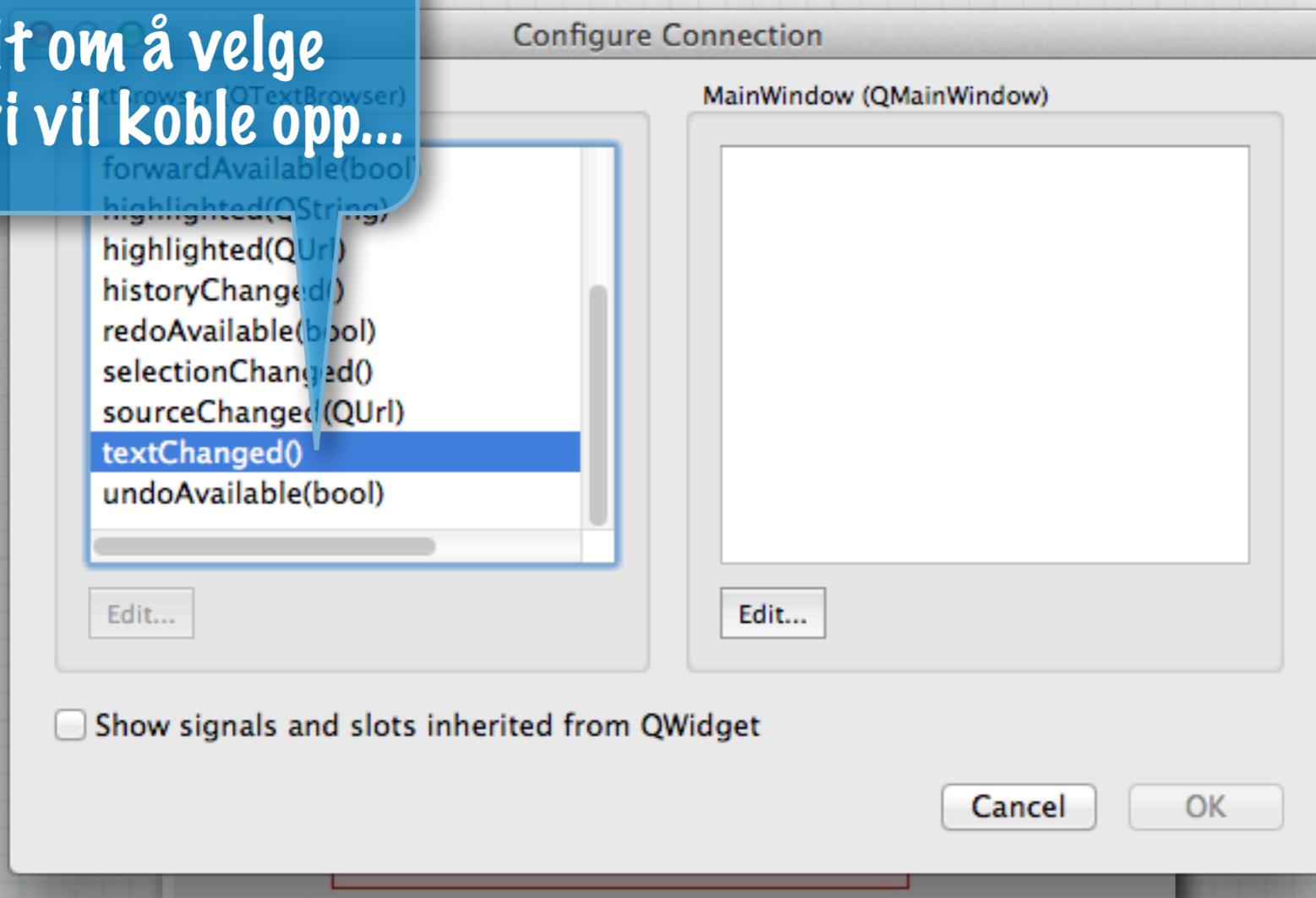
Slots & Signals: tilkobling





Slots & Signals: tilkobling

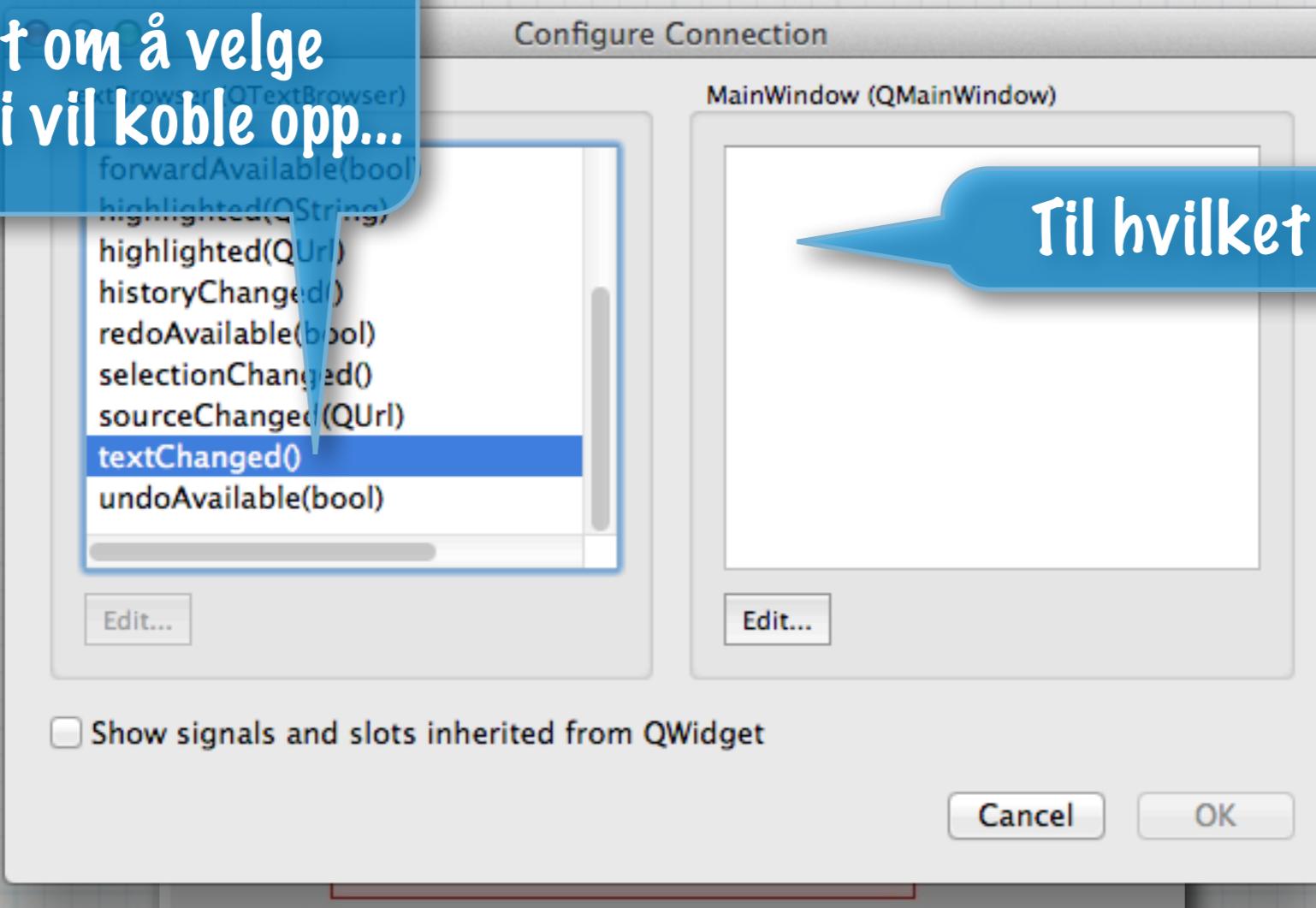
Vi blir så bedt om å velge
hvilket signal vi vil koble opp...





Slots & Signals: tilkobling

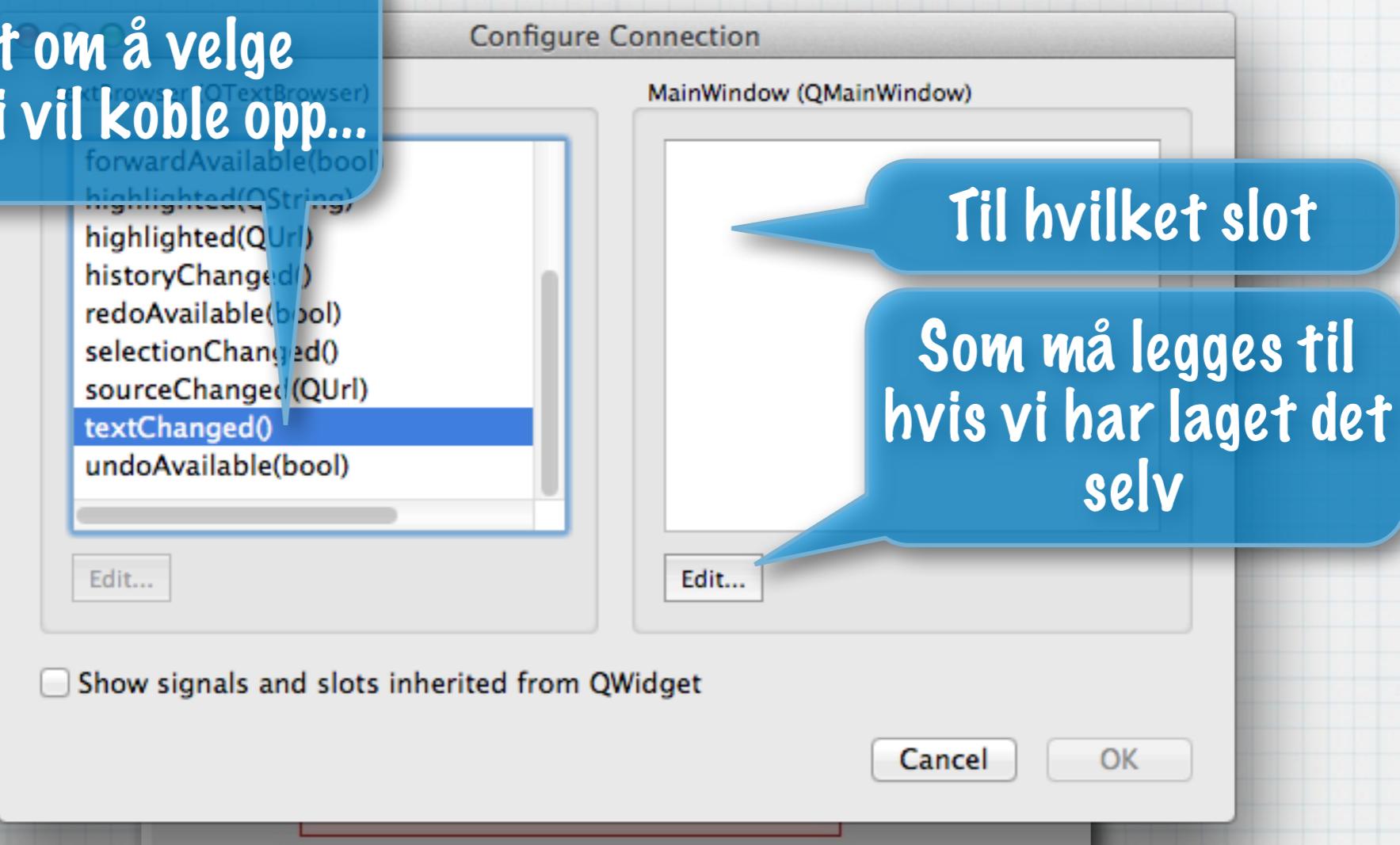
Vi blir så bedt om å velge
hvilket signal vi vil koble opp...





Slots & Signals: tilkobling

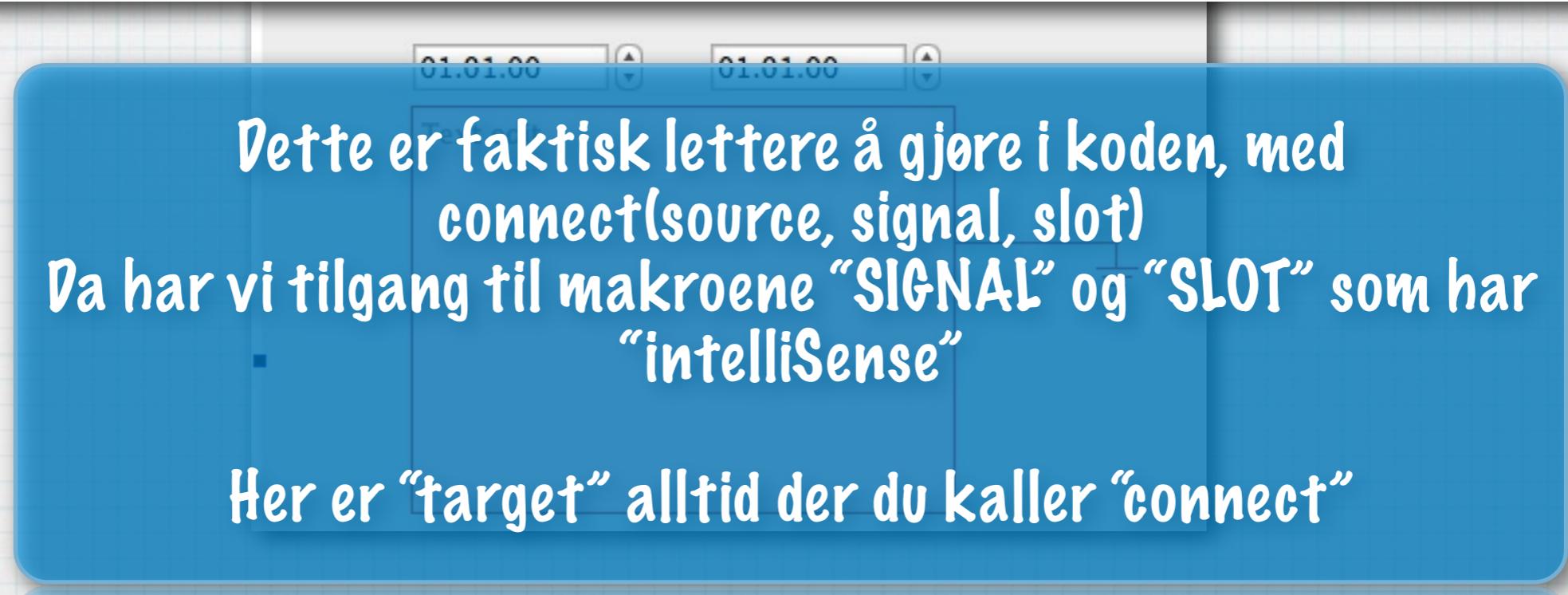
Vi blir så bedt om å velge
hvilket signal vi vil koble opp...





Slots & Signals: tilkobling

```
MainWindow::MainWindow(QWidget *parent) : QMainWindow(parent), ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    connect(ui->textBrowser, SIGNAL(textChanged()), SLOT(updateTextCount()));
}
```

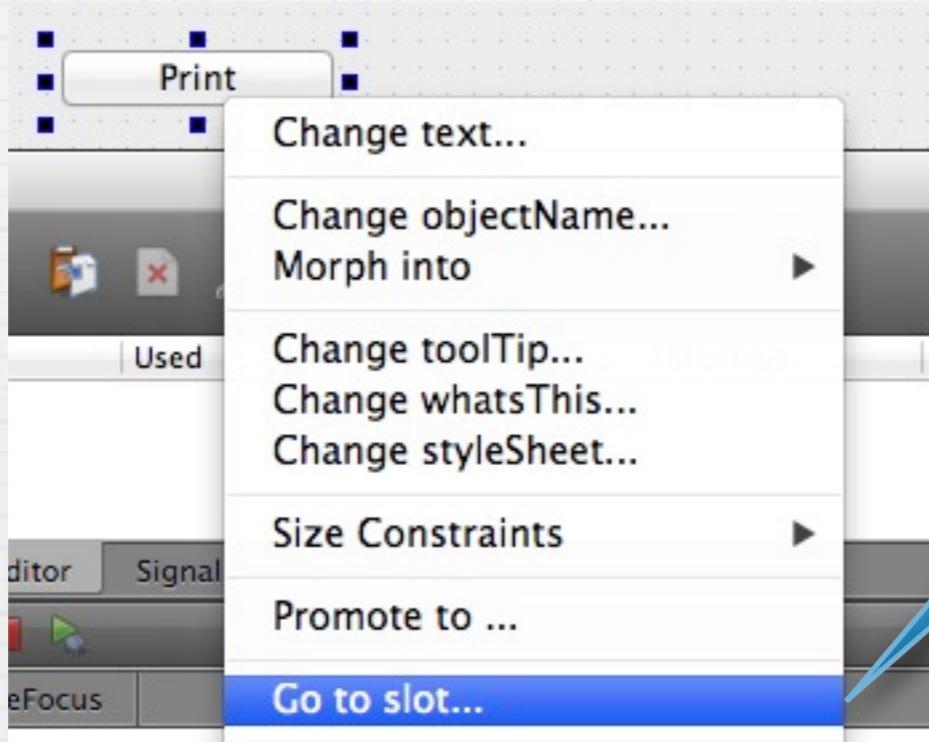


Her er "target" alltid der du kaller "connect"



Slots & Signals

- * For hvert signal kan man også lage en "hurtigtilkobling" ved å hoyreklikke på objektet med signalet (her knappen)



Dette tar oppretter en ny
"private slot" for signalet i
"Parent"



Slots & Signals

- * For hvert signal kan man også lage en "hurtigtilkobling" ved å hoyreklikke på objektet med signalet (her knappen)

```
private slots:  
    void on_pushButton_clicked();
```

Lages automatisk i
"mainwindow.h"

Implementasjonen lages automatisk i
"mainwindow.cpp" og du "hopper dit"
og kan skrive koden

```
void MainWindow::on_pushButton_clicked()  
{  
    //Whatever happens on click  
}
```

Demo!

Qt_demol.pro
(Oppdatert med progressbar siden forelesningen)



Slots & Signals

- * For å bruke slots og signals som ikke ligger i klassen fra før, kan man "promote" klassen (høyreklikk på UI-elementet)
- * Subklassen må man naturligvis skrive
- * Man kan da bruke de nye "slots" og "signals" direkte i GUI-editoren



Slots & Signals

- * Slots & Signals kan kun brukes av objekter som arver QOBJECT
 - * Alle QWidgets gjør det
 - * Men det koster
- * For rask grafikk med ekstremt mange elementer anbefales ikke dette
- * Da har Qt egne widgets man kan tegne på, veldig raskt

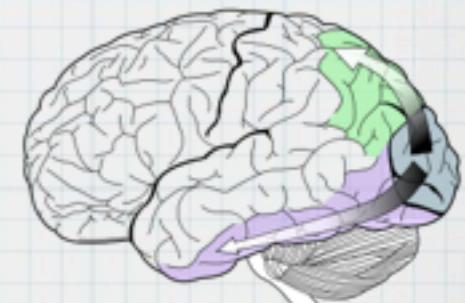
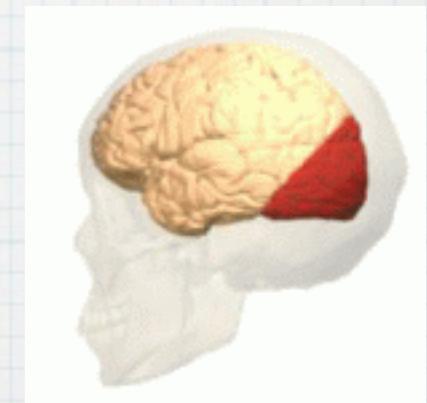
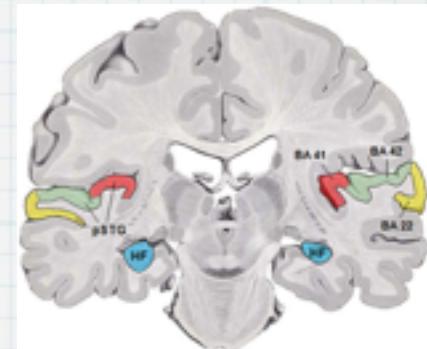


QML og QtQuick

- * En ny måte å lage GUI på, som er lettere å integrere med ny hardware
- * Fungerer nå for både Android og iOS
- * HTML-liknende kode og Javascript
- * Anbefaler å lære dette for reelle prosjekter, men ikke i kurset (det blir for mye javascript)

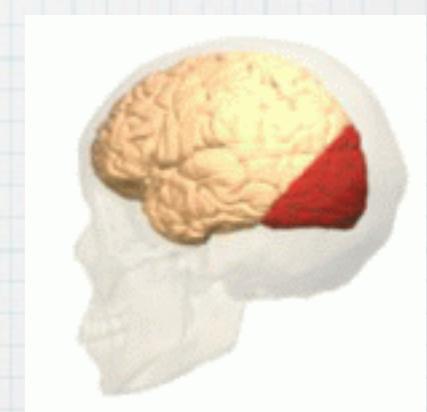
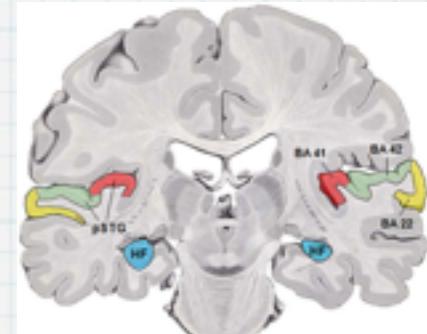
GUI-programmering er komplekst

- * Litt av hjernen er satt av til lyd
- * Hele bakhodet er GPU
- * Å visualisere noe gir tilgang til enormt mye ressurser
- * Men det er også veldig komplekst å gi hele GPU'en noe fornuftig å gjøre



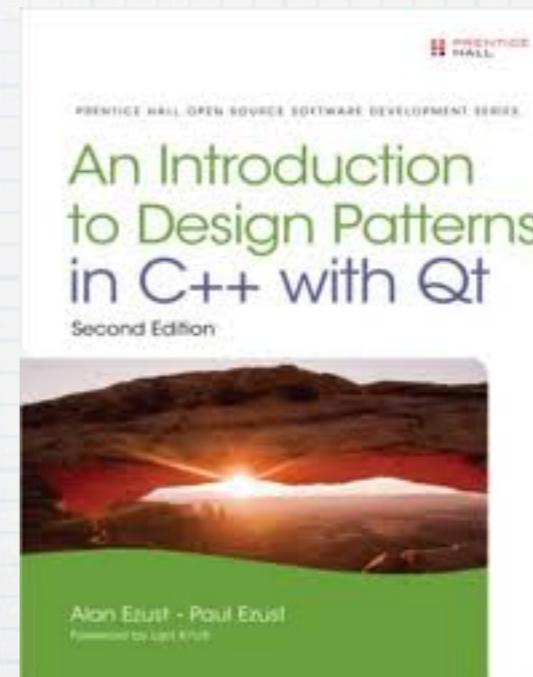
Kompleksitet: HD GUI vs. Terminal

- * En "Normal" terminal skjerm for MS Dos var $640 \times 480 = 0.37$ Mill. Piksler
 - * Men $80 \times 30 = 2400$ tegn terminal grafikk
 - * Med 16 bit farger, $2400 \times (2^{16}) = 150$ mill. mulige bilder
- * En "normal" HD-skjerm er $1920 \times 1080 = 2.1$ Mill. Piksler
 - * Og hver piksel er 32 bit stor (24 bit + alpha)
 - * 2.1 mill. * 4.2 mrd. > 9×10^{15} mulige bilder (millioner av milliarder - billiard)



GUI er komplekst

- * Derfor er svært mange design patterns blitt til for å håndtere GUI
- * Factory, Composite, chain of responsibility, observer...
- * Egen bok om Qt og design-patterns



Qt oppsummering

- * Det utvider C++ med nye "keywords" og konsepter.
 - * Det blir borte før kompilering, men krever Qmake
- * Bjarne Stroustrup liker ikke at de har er "non-standard"
 - * Det gjør deg lett avhengig av Qt hvis du først begynner med det
 - * ...Anta du bare vil ha et Qt-vindu
 - * ...men det har en event-loop som bruker signals/slots
 - * ...Og alle widgets du vil putte inn må arve QObject
- * Robust, rikt, "convenient" og godt dokumentert
 - * Bruk det gjerne i prosjektoppgaven

Det var det...

- * Det var siste forelesning (snufts)
- * Fra nå: lab'er med stud.ass fra 10:30
- * Jeg kommer ved behov
- * Dere kan booke veiledning, og som alltid, sende e-post.
- * Håper kurset har vært nyttig for dere - om dere blir C++-utviklere eller ikke!

Takk for meg!

Og lykke til med prosjektoppgave og karriere :-)

PS: Vi sees ved prosjektpresentasjonen