Eötvös Loránd Tudományegyetem
Informatikai Kar

Eseményvezérelt alkalmazások fejlesztése I

1. előadás

Grafikus felület és eseményvezérlés, a Qt keretrendszer

Giachetta Roberto

A jegyzet az ELTE Informatikai Karának
2014. évi Jegyzetpályázatának támogatásával készült

A Qt keretrendszer

Története

- A *Qt* egy alkalmazás-fejlesztési keretrendszer, amely egy komplett osztálykönyvtárat biztosít számos platformon történő alkalmazás-fejlesztésre elsősorban C++-ban
  - 1991-ben mutatta be a Trolltech, jelenleg a Digia tulajdonában van
  - 5.3 a legfrissebb változat

  - támogat grafikus felületet, multimédia lejátszást, adatbázis-kezelést, 3D grafikát, hálózati és webes kommunikációt
  - fejlesztőeszközök és dokumentáció a **`qt-project.org`** oldalon érhető el

A Qt keretrendszer

Felhasználása

- rendelkezik nyílt forráskódú (LGPL) és kereskedelmi verzióval is
- több grafikus környezet (*KDE*, *Unity*), számos alkalmazás (*AutoCAD*, *Skype*, *Adobe Photoshop*, *Maya*, *VLC*, *Mathematica*, …) és vállalat (*Google*, *HP*, *ESA*, *Panasonic*, *Samsung*, *Volvo*, …) használja
- többplatformos (*Windows*, *Linux*, *Mac OS X*, *Symbian*, *Maemo/MeeGo*, *Android*, *BlackBerry*, *iOS*, *OpenSolaris*, *webOS*, …)
- elsősorban a C++-t támogatja, de más nyelveken is biztosított a fejlesztés (*C#*, *Java*, *Ada*, *Haskell*, *PHP*, *Perl*, *Ruby*, …)

A Qt keretrendszer

A „Hello World” program

```
#include <QApplication>
#include <QLabel>
   // megfelelő Qt osztályok használata

int main(int argc, char *argv[]) {
  QApplication app(argc, argv);
     // alkalmazás osztály példányosítása
  QLabel label("Hello, world!");
     // címke a felirattal
  label.show();  // címke megjelenítése

  return app.exec();  // alkalmazás futtatása
}
```

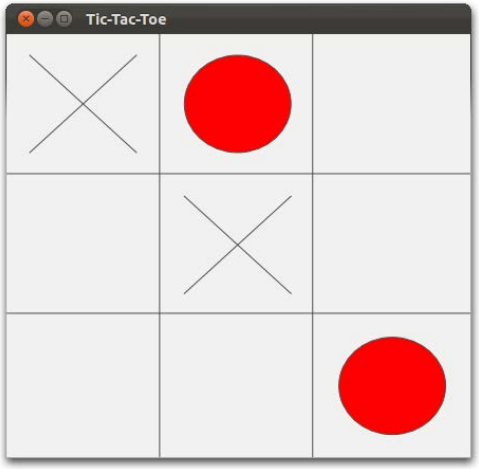
A Qt keretrendszer

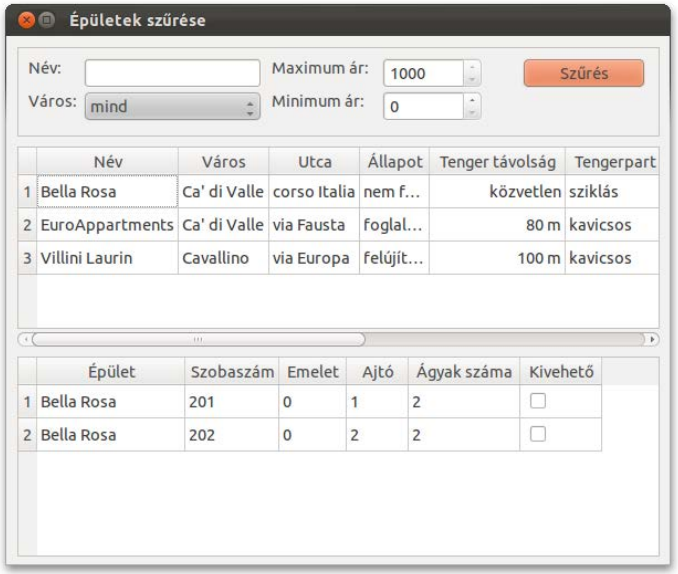
A grafikus felületű alkalmazás

- *Grafikus felületű alkalmazás*nak nevezzük azt a programot, amely 2D-s interaktív felhasználó felületen (*GUI*, *Graphical User Interface*) keresztül kommunikál a felhasználóval
  - gazdagabb interakció a konzol felületnél, számos módon beleavatkozhatunk a programfutásba
  - a működés jórészt várakozás a felhasználói interakcióra
  - a felület egy, vagy több ablakból (*form/window*) áll, amelyek vezérlőket (*control/widget*) tartalmaznak (pl.: nyomógombok, listák, menük, ...)
  - mindig van egy aktív ablak, és egy aktív vezérlő (ezen van a *fókusz*)

A Qt keretrendszer

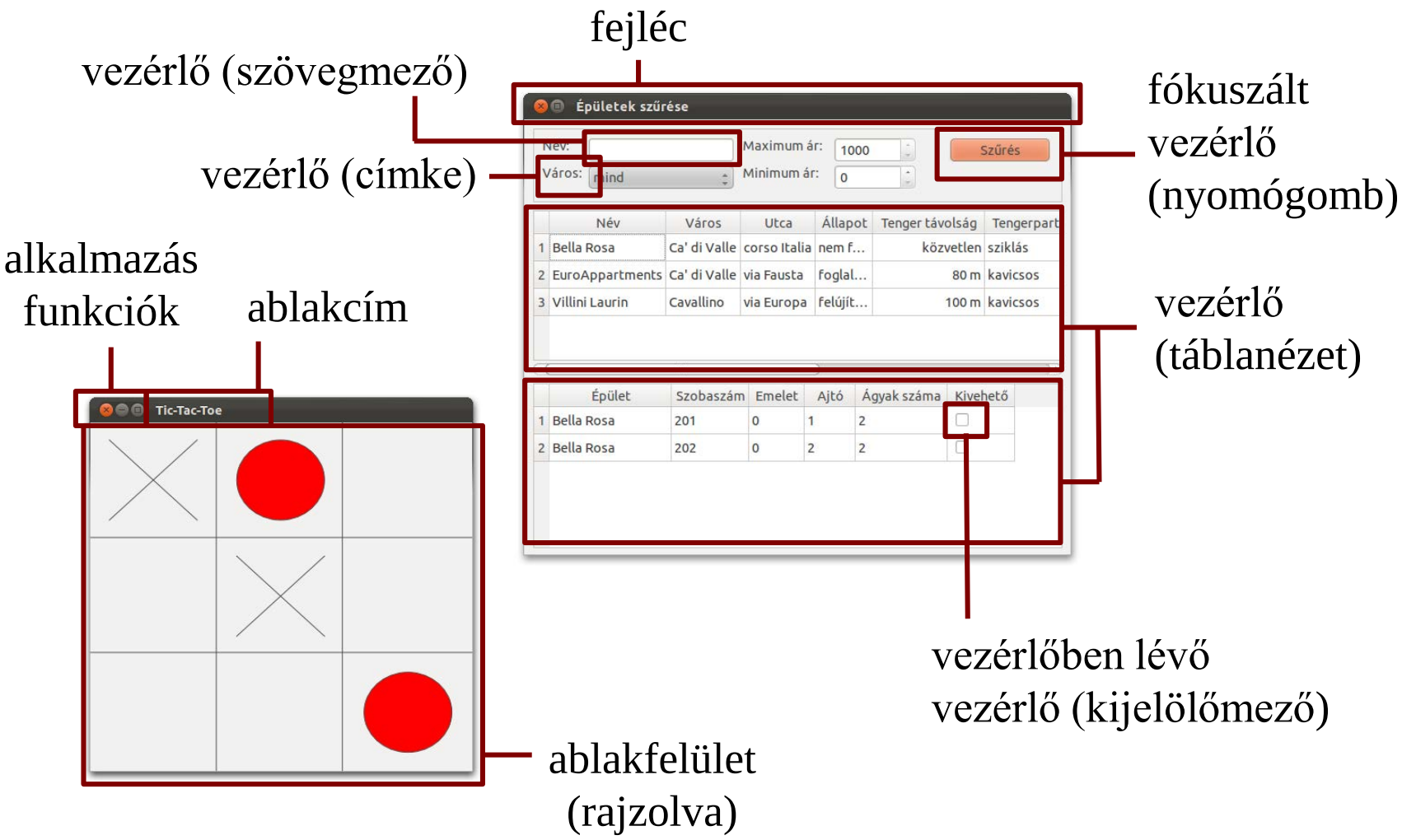
A grafikus felületű alkalmazás





A Qt keretrendszer

A grafikus felületű alkalmazás



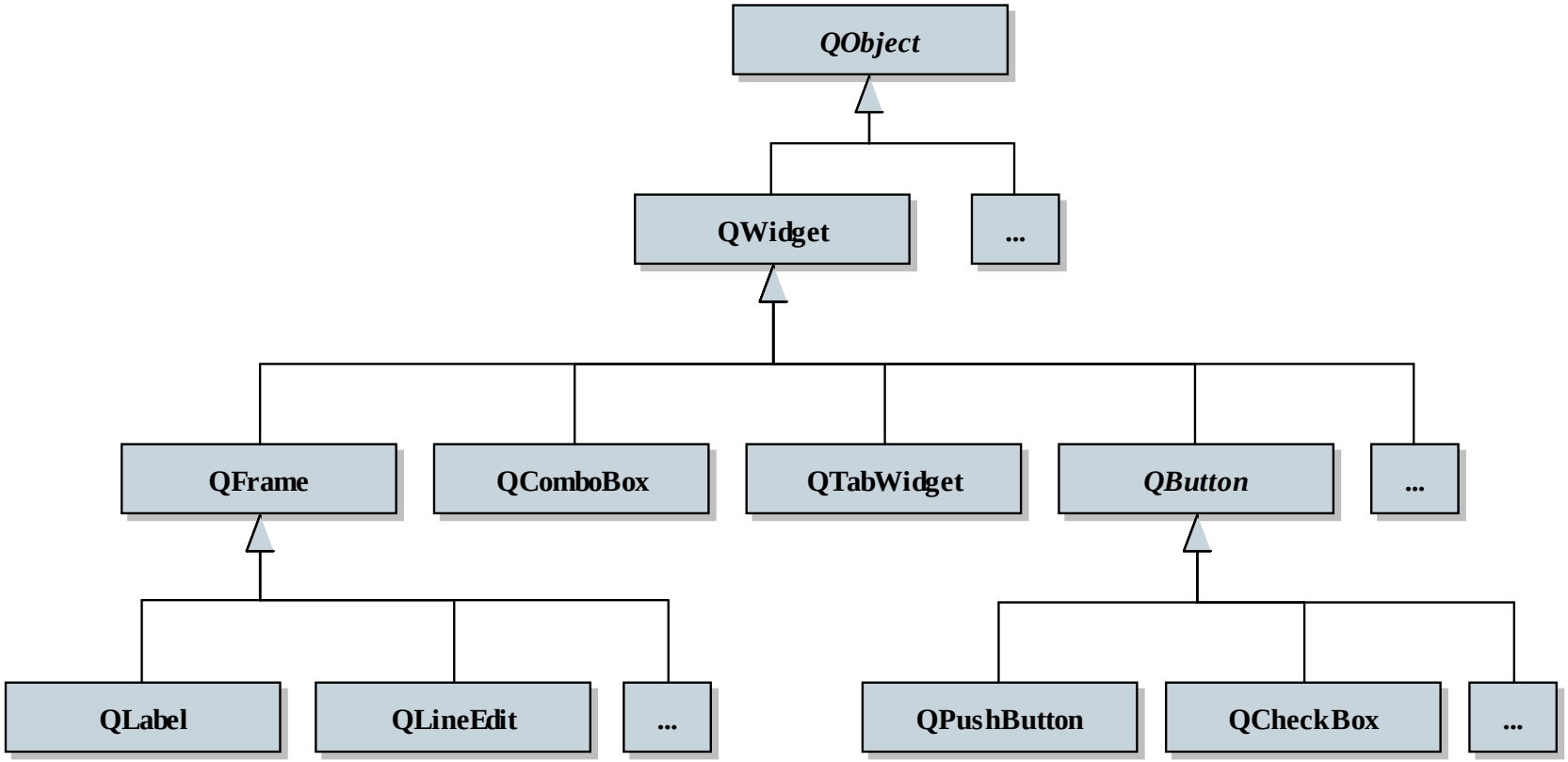
A Qt keretrendszer

A grafikus felület felépülése

- A grafikus felület *objektumorientáltan* épül fel
  - a vezérlőket osztályokként fogalmazzuk meg, megadjuk viselkedését (*metódusokkal*, pl. kattintás, megjelenés), illetve tulajdonságait (*mezőkkel*, pl. pozíció, méret, betűtípus)
  - a vezérlők sok hasonló tulajdonsággal bírnak, így könnyen öröklődési hierarchiába szervezhetőek
  - az öröklődési lánc legelején áll az általános vezérlő, új vezérlők származtatással definiálhatóak
  - a vezérlőket felhasználhatjuk más vezérlőkben, vagy használhatjuk önállóan, azaz *ablak*ként

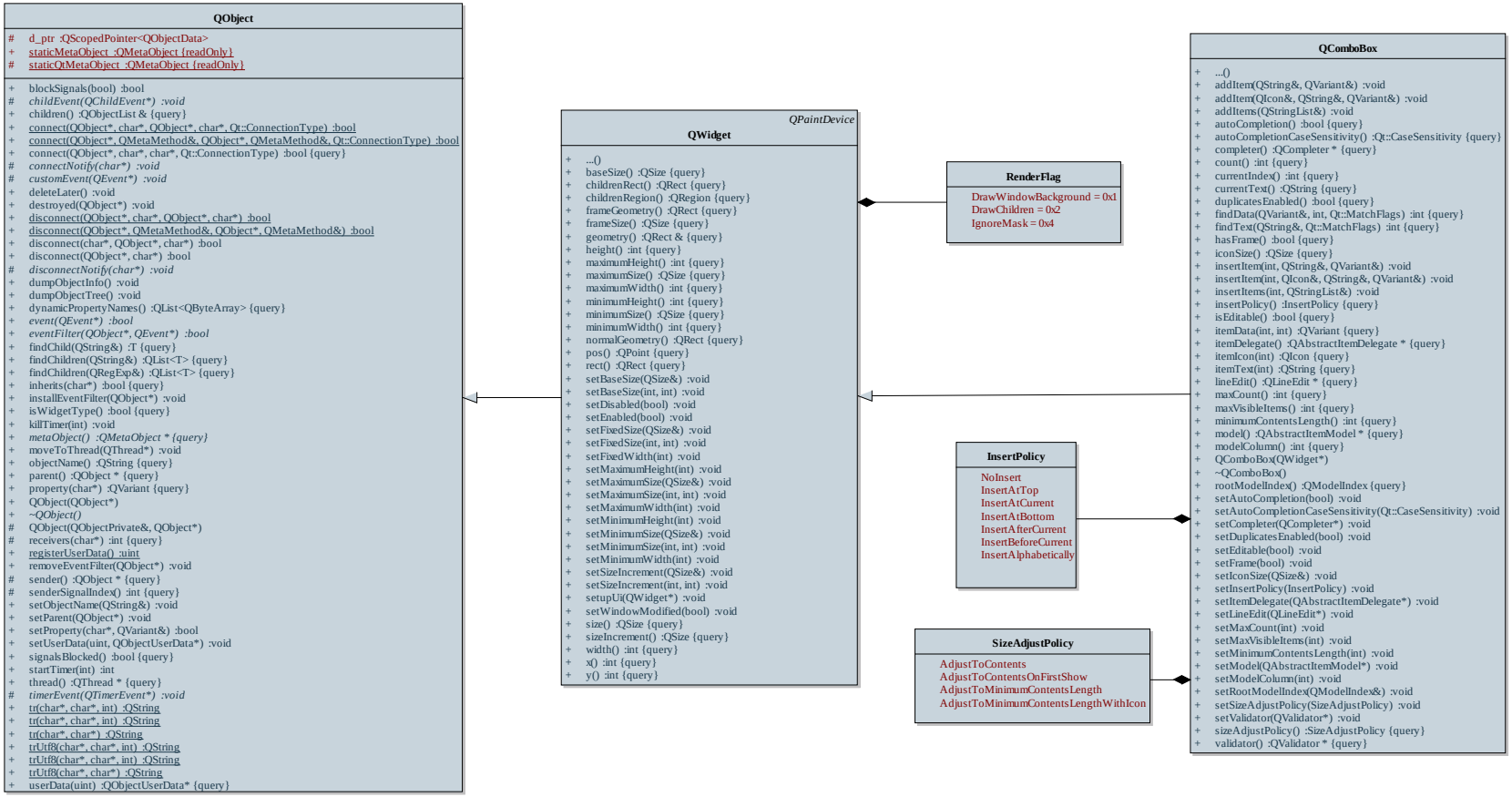
A Qt keretrendszer

A grafikus felület felépülése



A Qt keretrendszer

A grafikus felület felépülése



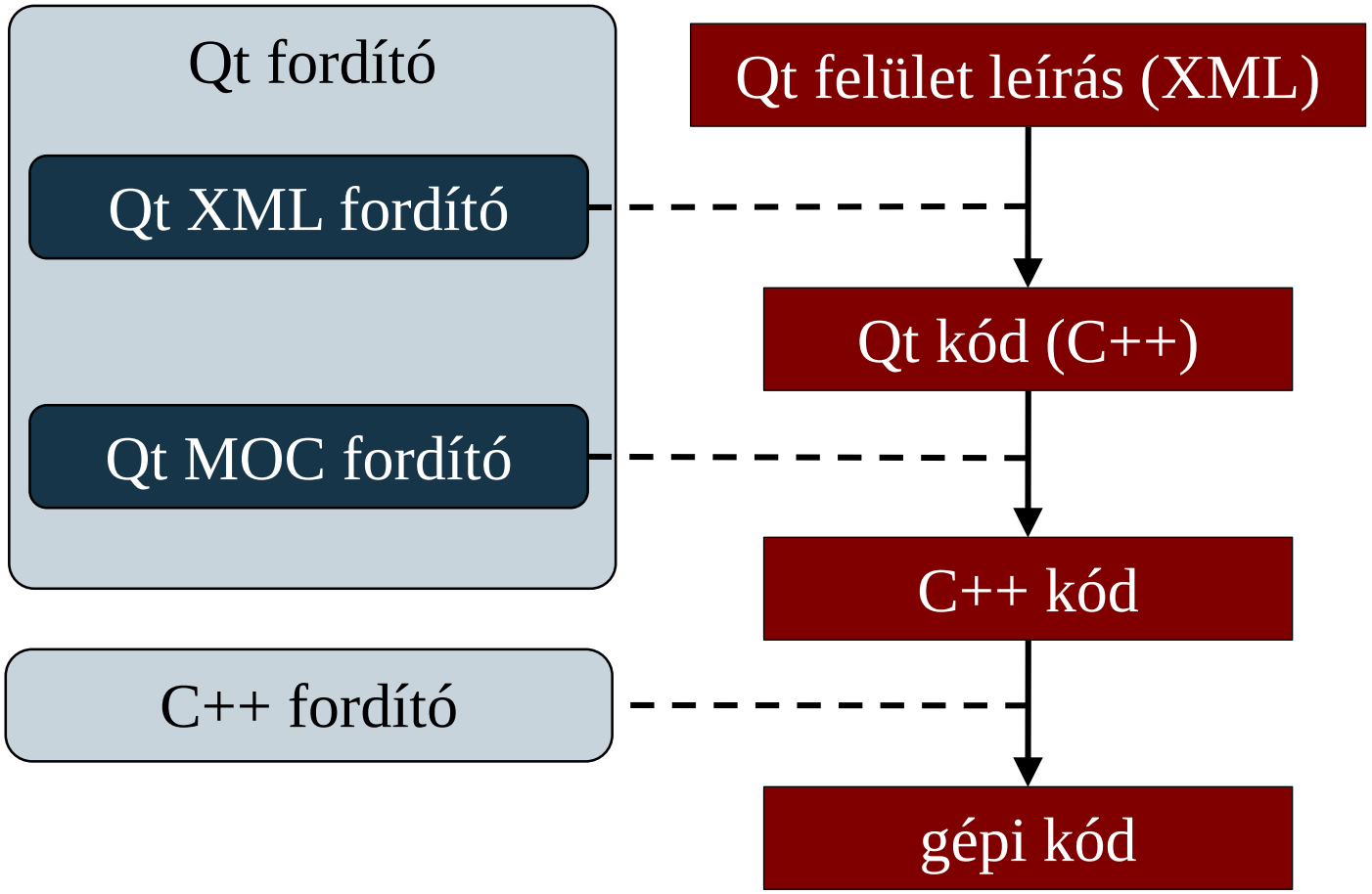
A Qt keretrendszer

Fejlesztés és fordítás

- A fejlesztés C++/Qt nyelven történik
  - elérhető a teljes C++ utasításkészlet, nyelvi könyvtár
  - a C++ nyelven felül további makrókat, kiegészítéseket tartalmaz, amelyeket a *Meta Object Compiler* (*MOC*) fordít le ISO C++ kódra
- Az alapértelmezett fejlesztőeszköz a Qt Creator, de más környezetekben is megjelent a Qt fejlesztés lehetősége (pl. *Code::Blocks*, *Visual Studio*)
- Külön tervezőprogram (*Qt Designer*) adott a grafikus felület létrehozására, amely XML nyelven írja la felület felépítését, ez automatikusan C++/Qt kódra fordul

A Qt keretrendszer

Fejlesztés és fordítás



A Qt keretrendszer

Fejlesztés és fordítás

- A fordítás projektszinten történik, a szükséges információk *projektfájlokban* (`.pro`) tárolódnak, amely tartalmazza
  - a felhasznált modulokat, kapcsolókat
  - forrásfájlok, erőforrások (pl. kép, szöveg,) listáját
  - eredmény paramétereket
- A fordítás közvetlenül is elvégezhető a fordítóval:

```
qmake -project
      # projektfájl automatikus létrehozása
qmake # fordítófájl (makefile) előállítása
make  # projekt fájlnak megfelelő fordítás és
      # szerkesztés végrehajtása
```

A Qt keretrendszer

Modulok

- A keretrendszer felépítése modularizált, 15 központi és 28 kiegészítő modulból áll, a legfontosabbak:
  - központi modul (**`QtCore`**)
  - grafikus felület (**`QtGui`**), grafikus vezérlők (**`QtWidgets`**)
  - adatbázis-kezelés (**`QtSQL`**)
- A projektben használandó modulokat a projektfájlban kell megadnunk, pl.:

```
QT += core gui widgets
```

- Egy modul tartalmát osztályonként, illetve egyszerre is betölthetjük az aktuális fájlba (pl. **`#include <QtGui>`**)

A Qt keretrendszer

Osztályhierarchia

- A nyelvi könyvtár osztályainak jelentős része teljes származtatási hierarchiában helyezkedik el
  - minden egy ősosztály (**`QObject`**) leszármazottja
  - az ősosztály biztosítja az eseménykezelést (**`connect`**), a tulajdonságkezelést, az időzítést (**`timer`**), stb.
- Számos segédosztállyal rendelkezik, pl.:
  - adatszerkezetek (**`QVector`**, **`QStack`**, **`QLinkedList`**, …)
  - fájl és fájlrendszer kezelés (**`QFile`**, **`QTextStream`**, **`QDir`**, …)
  - párhuzamosság és aszinkron végrehajtás (**`QThread`**, **`QSemaphore`**, **`QFuture`**, …)

A Qt keretrendszer

Szövegkezelés

- Qt-ben a karakterek 16 bites Unicode (UTF8) kódolásúak
  - már a **`QObject`** típus biztosítja a kódolási konverziót egy osztályszintű művelettel (**`QObject::trUtf8`**)
- A karakterek kezelését a **`QChar`** típus biztosítja, míg szövegre a **`QString`** típus alkalmazható
  - kompatibilis a C++ standard könyvtár **`string`** típusával, pl.: **`QString::fromSTDString(stdstr)`**
  - megkülönbözteti az üres és a nem létező szöveget (**`isNull`**, **`isEmpty`**)
  - alkalmas típuskonverziókra, pl. **`QString::number(4)`**, **`str.toInt()`**

A Qt keretrendszer

Grafikus felületű alkalmazások vezérlése

- A konzol felületű alkalmazások csak billentyűzettől fogadnak bemenetet a programfutás meghatározott pontjain, a vezérlés módját mi szabályozzuk (pl. főprogram, menü)
- A grafikus felületű alkalmazások billentyűzettől és egértől (érintőképernyőtől, stb.) fogadnak bemenetet a programfutás szinte bármely pillanatában, a vezérlés módja előre definiált
- A grafikus felületű alkalmazás vezérlését az *alkalmazás osztály* (*application class*) látja el
  - kezeli a felhasználói bevitelt, a felület elemeit, beállítja az alkalmazástulajdonságokat (megjelenés, elérési útvonal, ...)
  - a Qt-ben az alkalmazást a **`QApplication`** típus biztosítja

A Qt keretrendszer

Eseménykezelés

- A grafikus felületen tehát számos módon és ponton kezdeményezhetünk interakciót a programmal
- A program által kezelhető, lereagálható interakciókat nevezzük *esemény*eknek (*event*/*signal*), az interakció kezdeményezését nevezzük az esemény *kiváltás*ának
  - pl.: gombra kattintás, egér húzás, listaelem kiválasztás

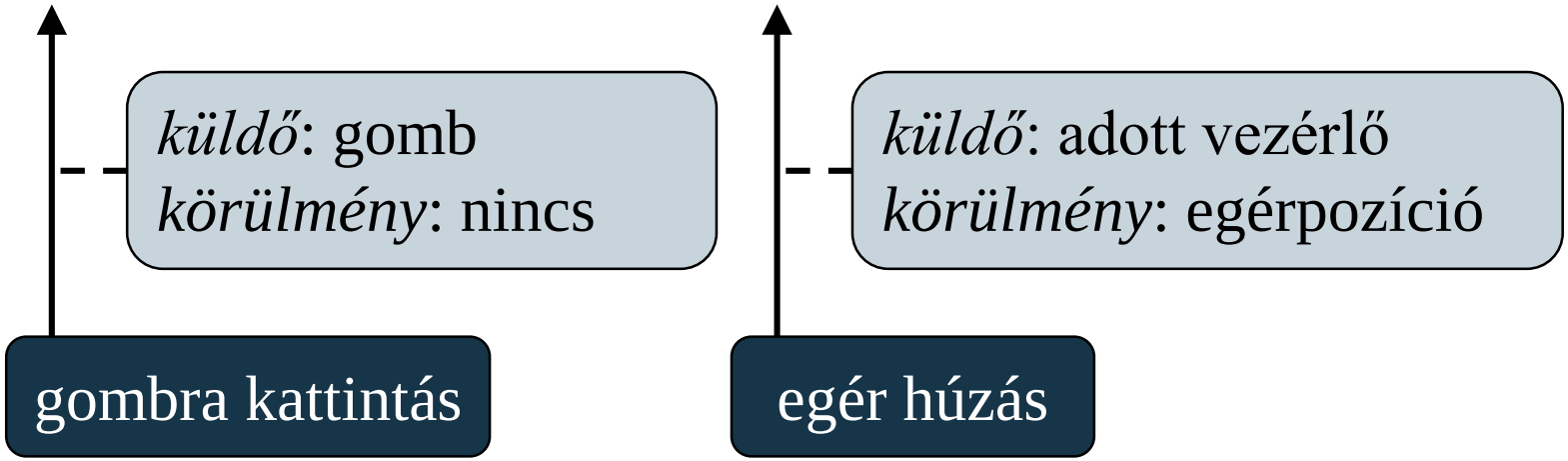
gombra kattintás

egér húzás

A Qt keretrendszer

Eseménykezelés

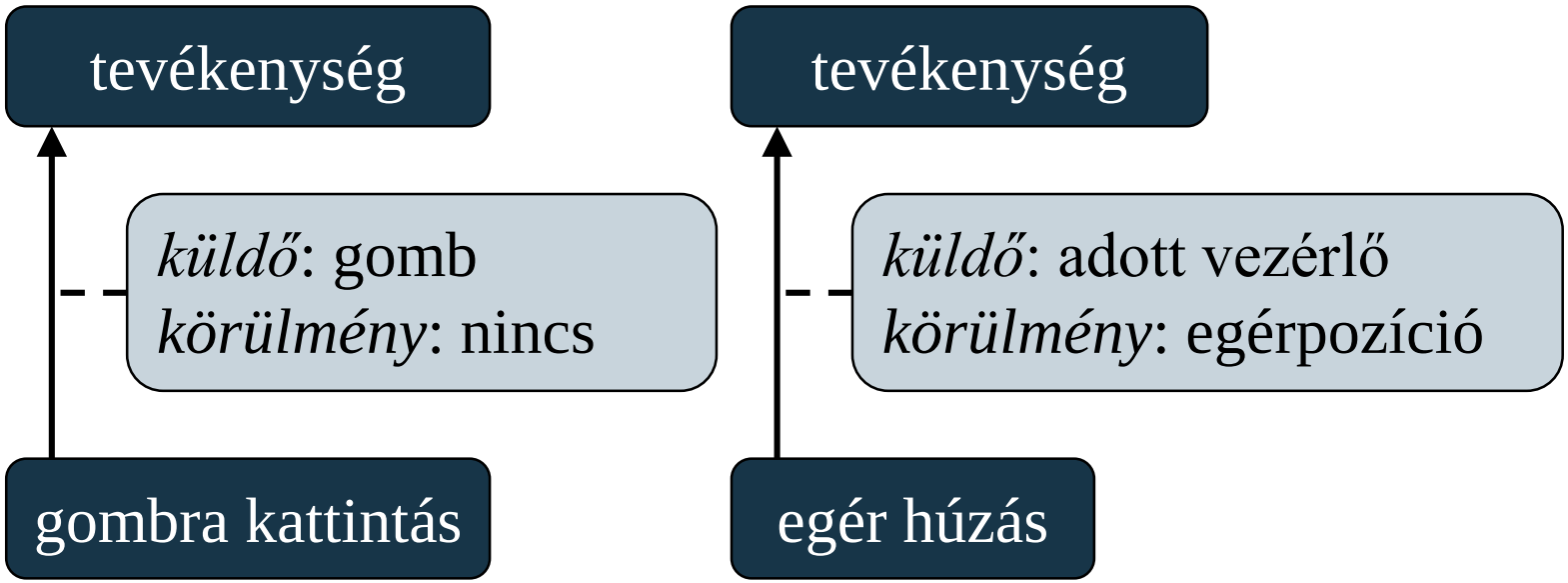
- Az eseménynek van:
  - *küldője* (*sender*): kiváltja az eseményt, pl. gomb, lista
  - *körülményei* (*arguments*): meghatározza az esemény paramétereit, pl. egér pozíciója a húzáskor, kiválasztott listaelem indexe



A Qt keretrendszer

Eseménykezelés

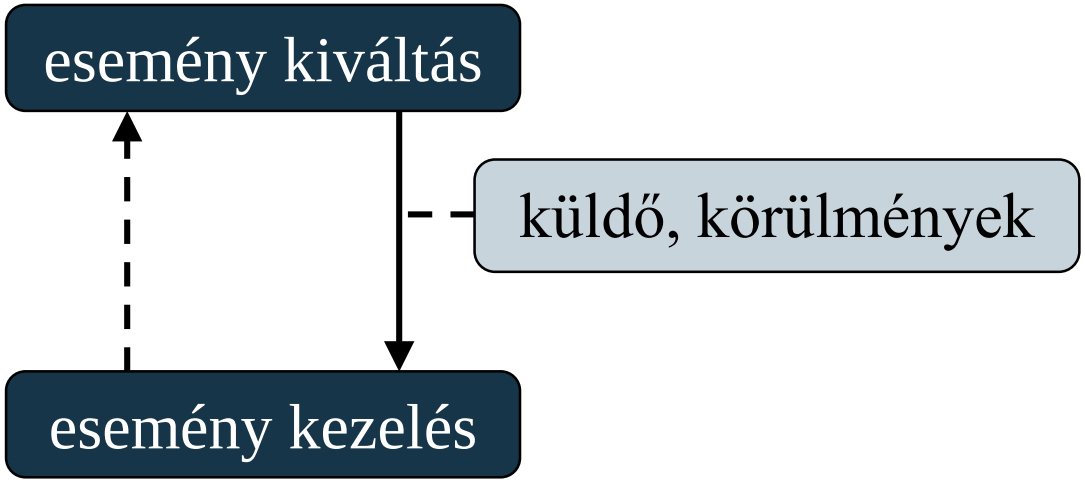
- Az eseményekre reagálva a program futtathat egy alprogramot, ezt nevezzük *eseménykezelő*nek (*event handler*/*slot*)
  - ha nem biztosítunk eseménykezelőt az eseményhez, akkor az *lekezeletlen* marad



A Qt keretrendszer

Eseménykezelés

- Az *összetett események* úgy valósulnak meg, hogy a program egy egyszerű eseményre kivált egy másikat
  - pl.: az egérrel kattintunk, és az egér a gombon van, akkor kiváltódik a gomb kattintása esemény
  - tehát az eseménykezelés egy több lépcsős, *ciklikus folyamat*



A Qt keretrendszer

Eseménykezelő társítás

- Az eseménykezeléshez összekapcsoljuk az eseményt az eseménykezelővel, ezt *társítás*nak nevezzük
  - Qt-ban ehhez a **`connect`** metódust használjuk, pl.:

```
connect(&button, SIGNAL(clicked()),
        &app, SLOT(quit()));
```

  - megadjuk, mely küldő objektum (*sender*) mely eseményére (**`SIGNAL`**) mely fogadó objektum (*receiver*) mely eseménykezelője (**`SLOT`**) kell, hogy fusson
  - mindig mutatókat adunk meg, bármely két alkalmas objektum összeköthető
  - a kötés elvégezhető **`QObject`** leszármazott típusban, illetve statikus metódus hivatkozással

A Qt keretrendszer

Eseménykezelő társítás

```
#include <QApplication>
#include <QPushButton>

int main(int argc, char *argv[]){
    QApplication app(argc, argv); // alkalmazás
    QPushButton quit("Quit"); // gomb
    quit.resize(75, 30); // méret
    quit.setFont(QFont("Times", 18, QFont::Bold));
       // betűtípus
    QObject::connect(&quit, SIGNAL(clicked()),
                     &app, SLOT(quit()));
    quit.show(); // gomb megjelenítése
    return app.exec();
}
```