



**Eötvös Loránd Tudományegyetem
Informatikai Kar**

Eseményvezérelt alkalmazások fejlesztése I

1. előadás

Grafikus felület és eseményvezérlés, a Qt keretrendszer

Giachetta Roberto

<http://people.inf.elte.hu/groberto>

A Qt keretrendszer

Bemutatása

- A *Qt* egy alkalmazás-fejlesztési keretrendszer, amely számos platformot támogatja az asztali, mobil és beágyazott alkalmazások fejlesztését
 - elérhető a qt.io oldalról
 - támogatja a grafikus felületet, adatbázis-kezelést, multimédiát, 3D grafikát, hálózati és webes kommunikációt
 - rendelkezik nyílt forráskódú (LGPL) és kereskedelmi verzióval is
 - fejlesztésre elsősorban a C++-t támogatja, de más nyelvekre is elérhető, valamint rendelkezik saját leíró nyelvvel (Qt Quick)



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 myLabel("Hello, world!");
        // címke a felirattal
    myLabel.show(); // címke megjelenítése

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

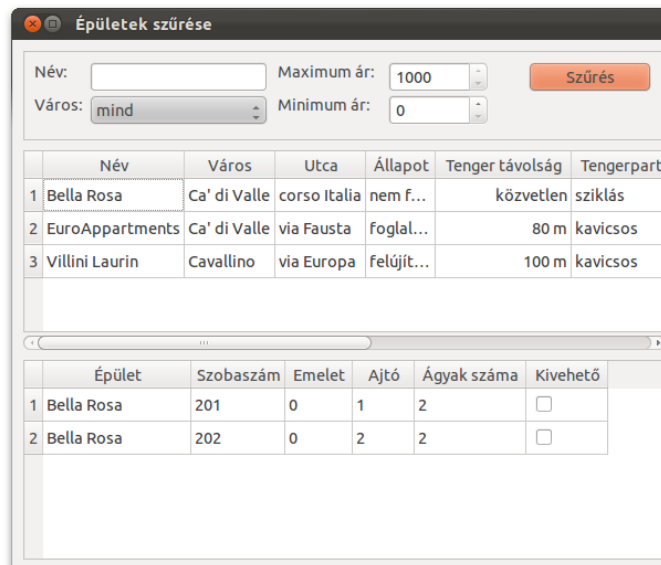
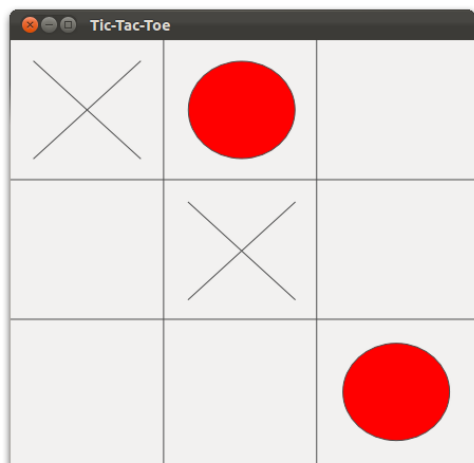
A Qt keretrendszer

A grafikus felületű alkalmazás

- *Grafikus felületű alkalmazásnak* 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

The diagram illustrates various Qt GUI components using two application windows. Red boxes and lines highlight specific elements, with labels in Hungarian explaining their function.

Épületek szűrése (Building Search Window):

- fejléc** (title bar): The top bar containing the window title and standard OS window controls.
- vezérlő (szövegmező)** (text control): A text input field for the search criteria.
- vezérlő (címke)** (label control): A label for the search criteria.
- fókuszált vezérlő (nyomógomb)** (focused control/button): A button labeled "Szűrés" (Filter).
- vezérlő (táblanézlet)** (table control): A table displaying search results with columns: Név, Város, Utca, Állapot, Tenger távolság, Tengerpart.

Név	Város	Utca	Állapot	Tenger távolság	Tengerpart
1 Bella Rosa	Ca' di Valle	corso Italia	nem f...	közvetlen	sziklás
2 EuroApartments	Ca' di Valle	via Fausta	foglal...	80 m	kavicsos
3 Villini Laurin	Cavallino	via Europa	felújit...	100 m	kavicsos

Tic-Tac-Toe Window:

- ablakcím** (window title): The title bar text "Tic-Tac-Toe".
- alkalmazás funkciók** (application functions): The window's title bar and standard OS window controls.
- ablakfelület (rajzolva)** (drawn window surface): The 3x3 grid game board with red circles and grey X's.
- vezérlőben lévő vezérlő (kijelölőmező)** (control in control/checkbox): A checkbox in the "Kivehető" (Removable) column of a table below the game board.

Épület	Szobaszám	Emelet	Ajtó	Ágyak száma	Kivehető
1 Bella Rosa	201	0	1	2	<input type="checkbox"/>
2 Bella Rosa	202	0	2	2	<input type="checkbox"/>

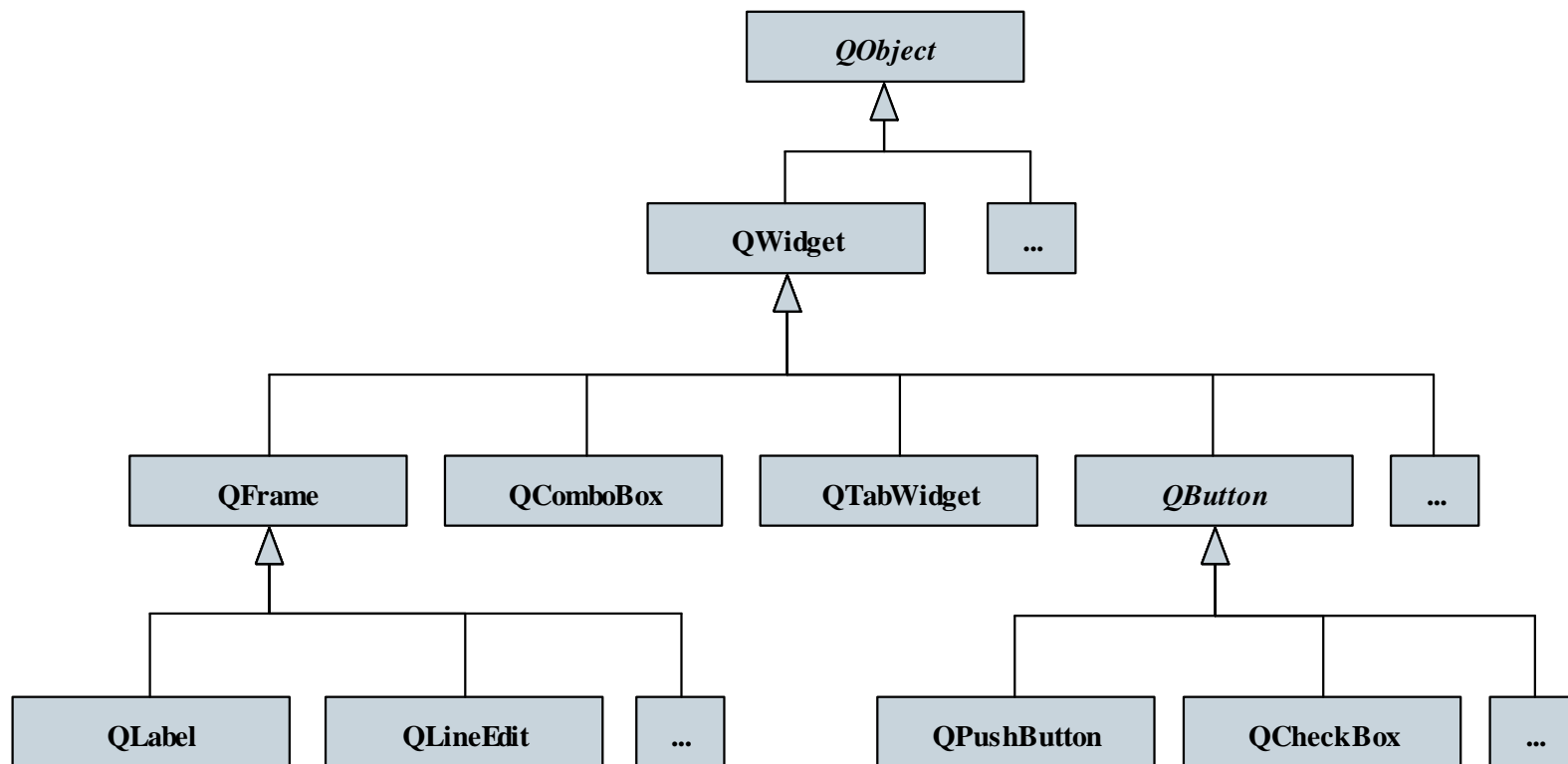
A Qt keretrendszer

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 *ablakként*

A Qt keretrendszer

A grafikus felület felépülése



A Qt keretrendszer

A grafikus felület felépülése

```
class QObject {
+ staticMetaObject :QMetaObject [readOnly]
+ # d_ptr :QScopedPointer<QObjectData>
+ # staticQMetaObject :QMetaObject [readOnly]
+ QObject(QObject*)
+ ~QObject()
+ event(QEvent*) :bool
+ eventFilter(QObject*, QEvent*) :bool
+ tr(char*, char*, int) :QString
+ trUtf8(char*, char*, int) :QString
+ metaObject() :QMetaObject * [query]
+ tr(char*, char*, int) :QString
+ tr(char*, char*) :QString
+ trUtf8(char*, char*, int) :QString
+ trUtf8(char*, char*) :QString
+ objectName() :QString [query]
+ setObjectName(QString&) :void
+ isWidgetType() :bool [query]
+ signalsBlocked() :bool [query]
+ blockSignals(bool) :bool
+ thread() :QThread * [query]
+ moveToThread(QThread*) :void
+ startTimer(int) :int
+ killTimer(int) :void
+ findChild(QString&, T) [query]
+ findChildren(QString&, QList<T>) [query]
+ findChildren(QRegExp&, QList<T>) [query]
+ children() :QObjectList & [query]
+ setParent(QObject*) :void
+ installEventFilter(QObject*) :void
+ removeEventFilter(QObject*) :void
+ connect(QObject*, char*, QObject*, Qt::ConnectionType) :bool
+ connect(QObject*, QMetaMethod&, QObject*, QMetaMethod&, Qt::ConnectionType) :bool
+ disconnect(QObject*, char*, char*, Qt::ConnectionType) :bool [query]
+ disconnect(QObject*, char*, QObject*, char*) :bool
+ disconnect(QObject*, QMetaMethod&, QObject*, QMetaMethod&) :bool
+ disconnect(char*, QObject*, char*) :bool
+ disconnect(QObject*, char*) :bool
+ dumpObjectTree() :void
+ dumpObjectInfo() :void
+ setProperty(char*, QVariant&) :bool
+ property(char*) :QVariant [query]
+ dynamicPropertyName() :QList<QByteArray> [query]
+ registerUserData() :uint
+ setUserData(uint, QObjectUserData*) :void
+ userData(uint) :QObjectUserData * [query]
+ destroyed(QObject*) :void
+ parent() :QObject * [query]
+ inherits(char*) :bool [query]
+ deleteLater() :void
+ # sender() :QObject * [query]
+ # senderSignalIndex() :int [query]
+ # receivers(char*) :int [query]
+ # timerEvent(TimerEvent*) :void
+ # childEvent(ChildEvent*) :void
+ # customEvent(QEvent*) :void
+ # connectNotify(char*) :void
+ # disconnectNotify(char*) :void
+ # QObject(QObjectPrivate&, QObject*)
}
```

```
class QWidget : QPaintDevice {
+ setEnabled(bool) :void
+ setDisabled(bool) :void
+ setWindowModified(bool) :void
+ frameGeometry() :QRect [query]
+ geometry() :QRect & [query]
+ nominalGeometry() :QRect [query]
+ x() :int [query]
+ y() :int [query]
+ pos() :QPoint [query]
+ frameSize() :QSize [query]
+ size() :QSize [query]
+ width() :int [query]
+ height() :int [query]
+ rect() :QRect [query]
+ childrenRect() :QRect [query]
+ childrenRegion() :QRegion [query]
+ minimumSize() :QSize [query]
+ maximumSize() :QSize [query]
+ minimumWidth() :int [query]
+ minimumHeight() :int [query]
+ maximumWidth() :int [query]
+ maximumHeight() :int [query]
+ setMinimumSize(QSize&) :void
+ setMaximumSize(QSize&, void) :void
+ setMinimumWidth(int) :void
+ setMaximumWidth(int) :void
+ setMinimumHeight(int) :void
+ setMaximumHeight(int) :void
+ setupUi(QWidget*) :void
+ sizeIncrement() :QSize [query]
+ setSizeIncrement(QSize&) :void
+ baseSize() :QSize [query]
+ setBaseSize(QSize&) :void
+ setFixedSize(QSize&) :void
+ setFixedSize(int, int) :void
+ setFixedWidth(int) :void
+ ...()
}
```

```
enum RenderFlag {
DrawWindowBackground = 0x1
DrawChildren = 0x2
IgnoreMask = 0x4
}
```

```
enum InsertPolicy {
NoInsert
InsertAtTop
InsertAtCurrent
InsertAtBottom
InsertAfterCurrent
InsertBeforeCurrent
InsertAlphabetically
}
```

```
enum SizeAdjustPolicy {
AdjustToContents
AdjustToContentsOnFirstShow
AdjustToMinimumContentsLength
AdjustToMinimumContentsLengthWithIcon
}
```

```
class QComboBox {
+ QComboBox(QWidget*)
+ ~QComboBox()
+ maxVisibleItems(int) :int [query]
+ setMaxVisibleItems(int) :void
+ count() :int [query]
+ setMaxCount(int) :void
+ maxCount() :int [query]
+ autoComplete() :bool [query]
+ setAutoCompletion(bool) :void
+ autoCompleteCaseSensitivity() :Qt::CaseSensitivity [query]
+ setAutoCompletionCaseSensitivity(Qt::CaseSensitivity) :void
+ duplicatesEnabled() :bool [query]
+ setDuplicatesEnabled(bool) :void
+ setFrame(bool) :void
+ hasFrame() :bool [query]
+ findText(QString&, Qt::MatchFlags) :int [query]
+ findData(QVariant&, int, Qt::MatchFlags) :int [query]
+ insertPolicy() :InsertPolicy [query]
+ setInsertPolicy(InsertPolicy) :void
+ sizeAdjustPolicy() :SizeAdjustPolicy [query]
+ setSizeAdjustPolicy(SizeAdjustPolicy) :void
+ setMinimumContentsLength(int) :void
+ iconSize() :QSize [query]
+ setIconSize(QSize&) :void
+ isEditable() :bool [query]
+ setEditable(bool) :void
+ setLineEdit(QLineEdit*) :void
+ lineEdit() :QLineEdit * [query]
+ setValidator(QValidator*) :void
+ validator() :QValidator * [query]
+ setCompleter(QCompleter*) :void
+ completer() :QCompleter * [query]
+ itemDelegate() :QAbstractItemDelegate * [query]
+ setItemDelegate(QAbstractItemDelegate*) :void
+ model() :QAbstractItemModel * [query]
+ setModel(QAbstractItemModel*) :void
+ rootModelIndex() :QModelIndex [query]
+ setRootModelIndex(QModelIndex) :void
+ modelColumn() :int [query]
+ setModelColumn(int) :void
+ currentIndex() :int [query]
+ currentText() :QString [query]
+ itemText(int) :QString [query]
+ itemIcon(int) :QIcon [query]
+ itemData(int, int) :QVariant [query]
+ addItems(QIcon&, QString&, QVariant&) :void
+ addItems(QStringList&) :void
+ insertItem(int, QString&, QVariant&) :void
+ insertItem(int, QIcon&, QString&, QVariant&) :void
+ insertItems(int, QStringList&) :void
+ ...()
}
```

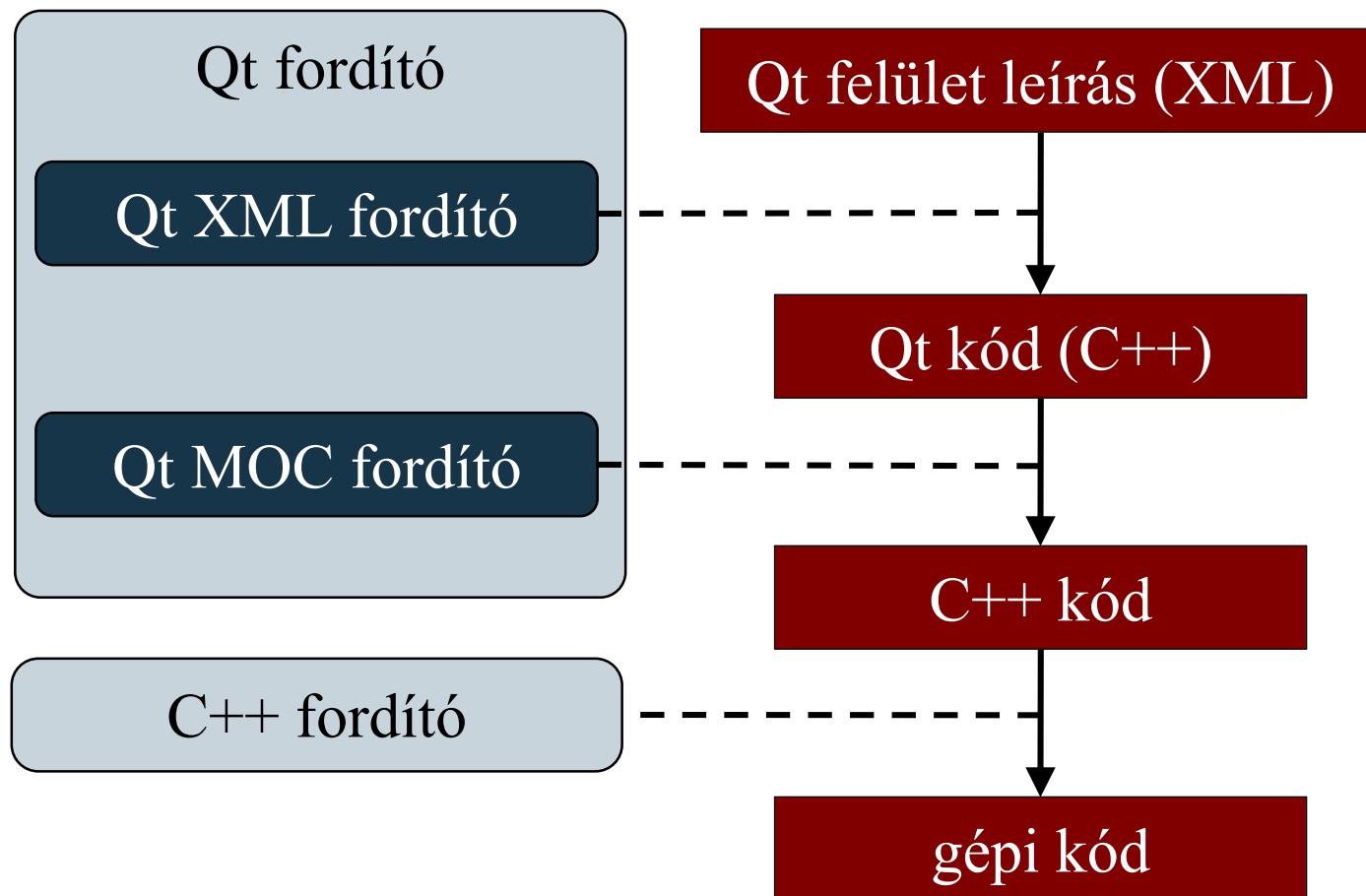
A Qt keretrendszer

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 le a 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) előállítás
```

```
make # projekt fájlak megfelelő fordítás és  
# szerkesztés végrehajtása
```

A Qt keretrendszer

Modulok

- A keretrendszer felépítése modularizált, a legfontosabb modulok:
 - 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 őosztály (`QObject`) leszármazottja
 - az őosztály 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 programfutas meghatározott pontjain, a vezérlés módját mi szabályozzuk (pl. főprogram, menü)
- A grafikus felületű alkalmazások billentyűzettől és egértől (érintőképernyőtől, stb.) fogadnak bemenetet a programfutas 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 *eseményeknek (event/signal)*, az interakció kezdeményezését nevezzük az esemény *kiváltásának*
 - pl.: gombra kattintás, egér húzás, listaelem kiválasztás

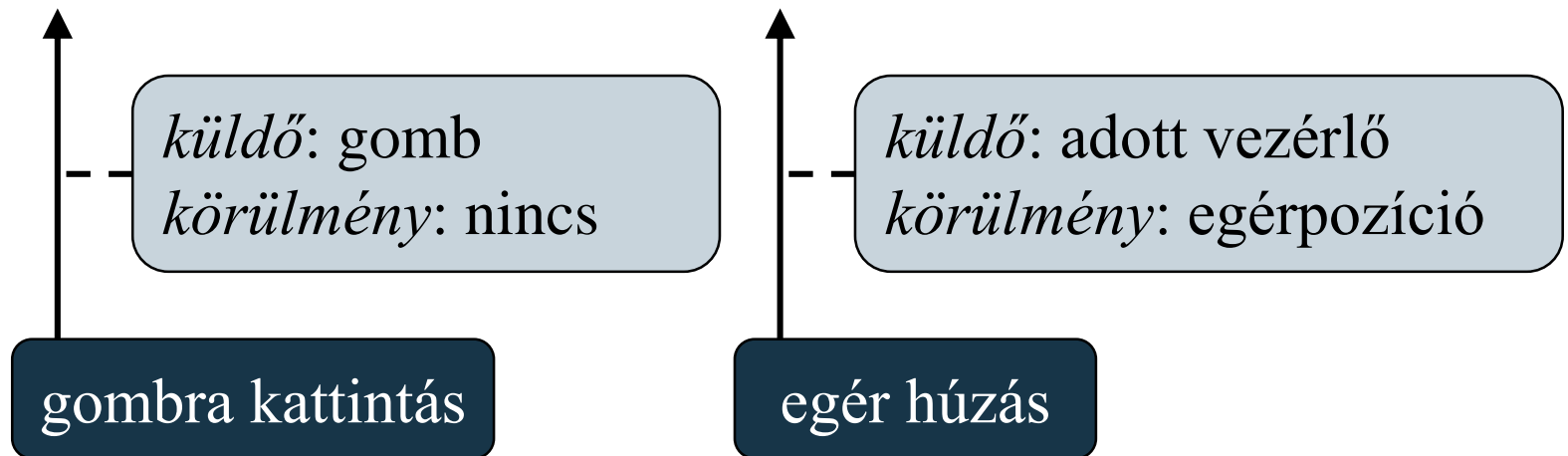
gombra kattintás

egér húzás

A Qt keretrendszer

Eseménykezelés

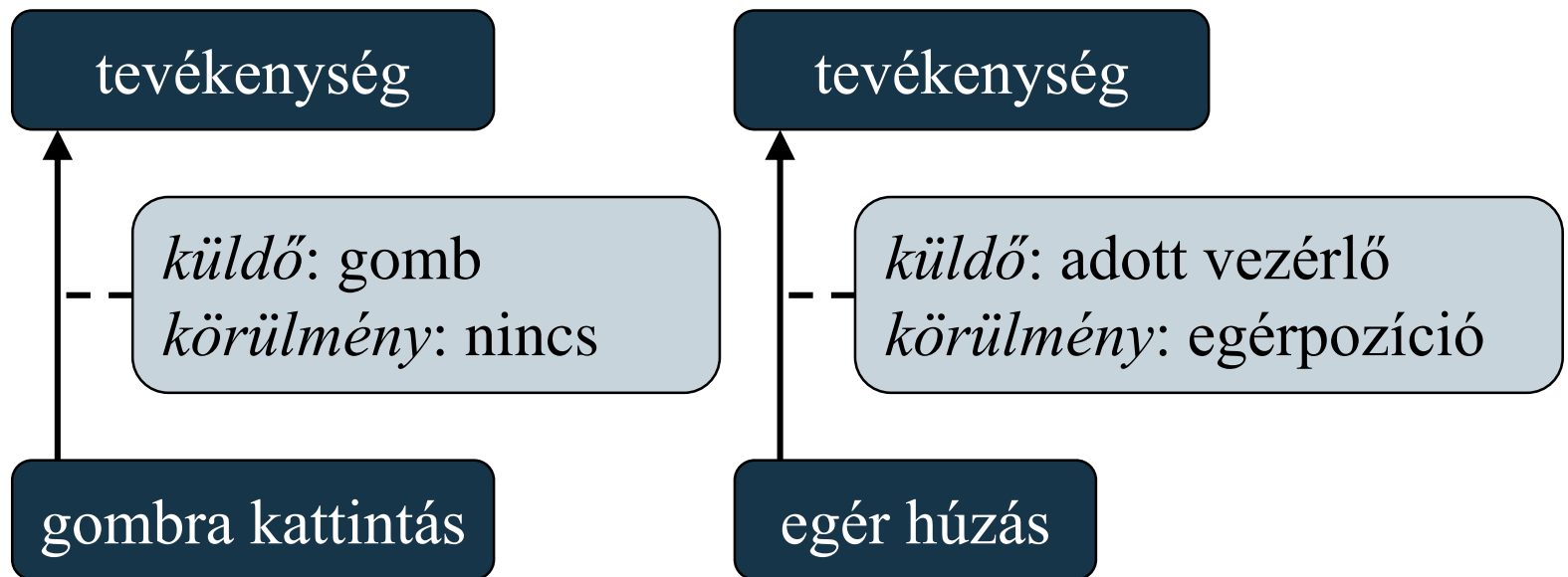
- Az eseménynek van:
 - *küldője (sender)*: kiváltja az eseményt, pl. gomb, lista
 - *körülményei (arguments)*: meghatározza az esemény paramétereit, pl. egér pozíciója a húzáskor, kiválasztott listaelem indexe



A Qt keretrendszer

Eseménykezelés

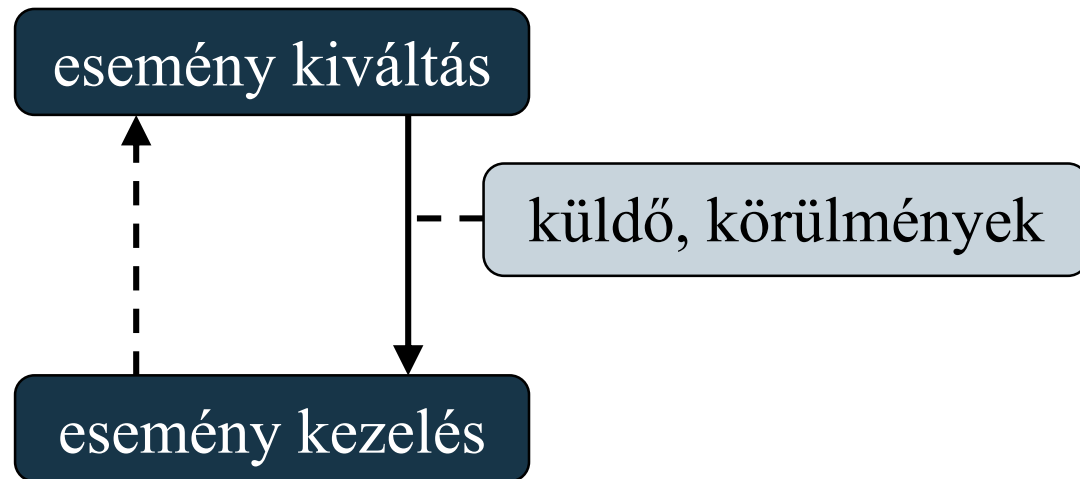
- Az eseményekre reagálva a program futtathat egy alprogramot, ezt nevezzük *eseménykezelőnek (event handler/slot)*
 - ha nem biztosítunk eseménykezelőt az eseményhez, akkor az *lekezeletlen* marad



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 *társításnak* 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();
}
```