Hallgatói kérdés

„Azt a hírt hallottam, hogy a tárgyhonlapon kiírtakkal ellentétben, nem jár az összes megoldott hftest-ért a +1jegy.

("Az **összes feladat** hibátlan **megoldása egy jeggyel jobb** félév végi jegyet eredményezhet, amennyiben az egyéb követelmények teljesülnek. A **2., 4., 6., és a 8. feladatot a Cporta** feladatbeadó rendszeren keresztül **is be kell adni, csak akkor jár érte pont."**)”

A félévközi jegy kiszámítása:

A 2 db nagy zárthelyivel és a három legjobban sikerült kis zárthelyivel összesen 110 pont szerezhető. A ténylegesen megszerzett pontokhoz adódnak hozzá az ún. EXTRA pontok.

EXTRA pontokat lehet gyűjteni az időben elkészített nagy HF részfeladataival, valamint a félév közben beadott és elfogadott szorgalmi feladatokkal. Összesen 1+1+2+2+8=14 pontot.

Pontszám=NZH1+NZH2+legjobb_3_kZH_pontszáma+EXTRA

Ponttáblázat

Pontszám	Jegy
0-51	1
52-66	2
67-81	3
82-96	4
97-110	5
>110	5

PótZH

- Pótolni a rosszabbul sikerült NZH-t lehet.
- Javítani, de rontani is lehet.
- kisZH nem pótolható, nem javítható
- PótZH ideje május 27. 08:15-10:00
- Neptun rendszerben jelentkezni kell.
- Határidő: május **26 08:00**
- Akit nem enged a Neptun tartozás miatt, az írjon e-mail. (tárgy: prog2 NZH)

Programozás alapjai II.
(12. ea) **C++**

Grafikus felületek és a C++

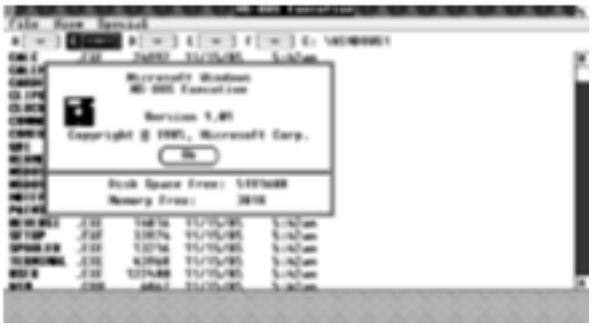
Szeberényi Imre
BME IIT
<szebi@iit.bme.hu>

Minden sokat változott...

Grafikus megjelenítő se volt mindig

- Annak ellenére, hogy a 60-as években már volt grafikus megjelenítő elterjedésükre még várni kellett. (egér: 1963)

- A mai értelemben vett grafikus felhasználói felületek (GUI) a 80-as években alakultak ki.
- X Window, MS Windows, Mac OS, ...



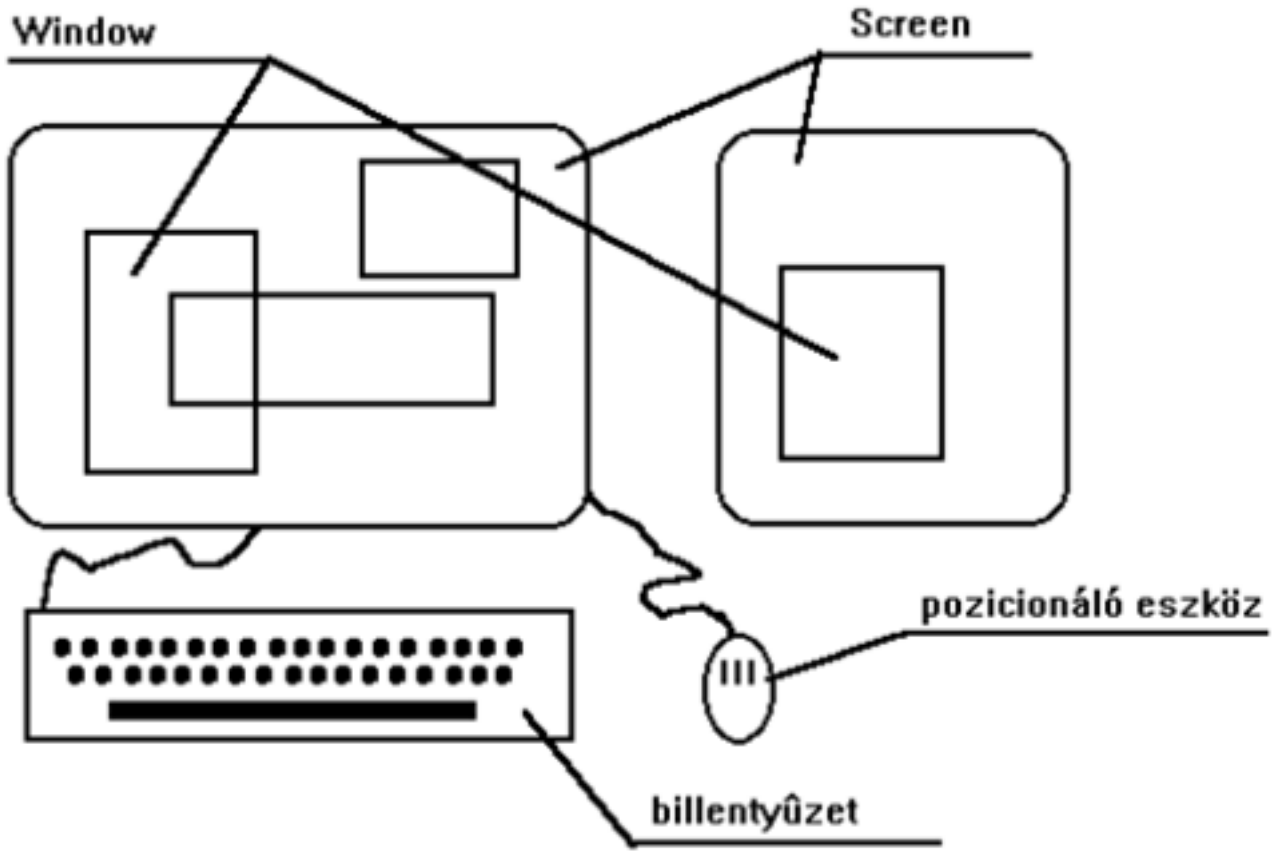
Hogyan működnek ?

- Objektum szemlélet (widget, gadget)
- Eseményvezérelt (objektumok eseményekkel kommunikálnak)
- A felhasználói felület tervezése és a program logikája gyakran elválik (külön módosítható)
- Első elterjedt grafikus rendszer a UNIX szabványos grafikus felülete az X Window rendszer, amit objektum szemléletű, de nem OO nyelven írták (C-ben)

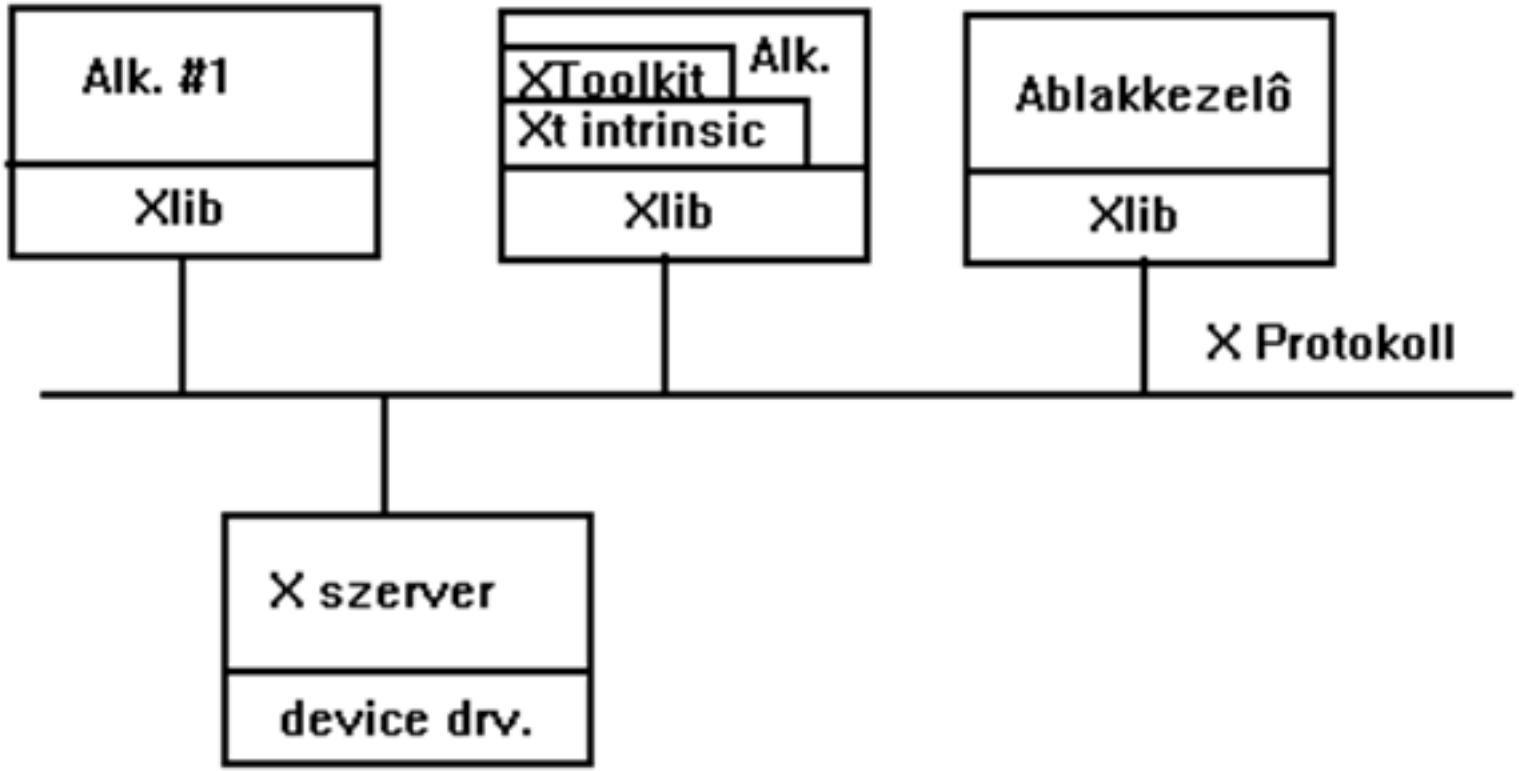
X window rendszer fogalmai

- Kliens = a szolgáltatást igénybe vevő.
- Egy adott berendezés (gép) egyszerre elláthat szerver funkciókat is, és kliens programokat is futtathat.
- X szerver = az X display-t működtető, a kliensek számára grafikus szolgáltatást nyújtó program.

X display = munkahely felépítése



X rendszer szoftver architektúrája



Eseményvezérelt alkalmazás

- **Inicializálás**:
  Kapacsolódás a szerverhez, window-k létrehozása, szerver erőforrások lefoglalása és attribútumaik beállítása.
- **Eseményhurok**:
  a programhoz érkező események feldolgozása.
  - Nem a program vezérli a felhasználót, hanem a felhasználó a programot

Kapcsolódás a szerverhez

HOST:SERVER.SCR

alakú azonosító stringgel történik, amit vagy explicit kap a megfelelő Xlib rutin, vagy az explicit megadás hiányában a **DISPLAY** környezeti változóból veszi.

- **HOST:** A szervert futtató számítógép hálózati azonosítója (név vagy cím).
- **SERVER:** Az adott hoston futó szerver azonosító száma (0. az első szerver).
- **SCR:** A kívánt screen sorszáma (0. az első).
  - Például: bubuka.iit.bme.hu:0.0

Egyszerű X program

```
int main() {
  Display *display;
  Window wMain;
  XEvent event;
  if ((display = XOpenDisplay(NULL)) == NULL) {
      fprintf(stderr, "Can't connect\n");exit(1);
  }
  wMain = XCreateSimpleWindow(display,
    DefaultRootWindow(display), 0, 0, width,
    height, borderWidth, border, bacgkground);
```

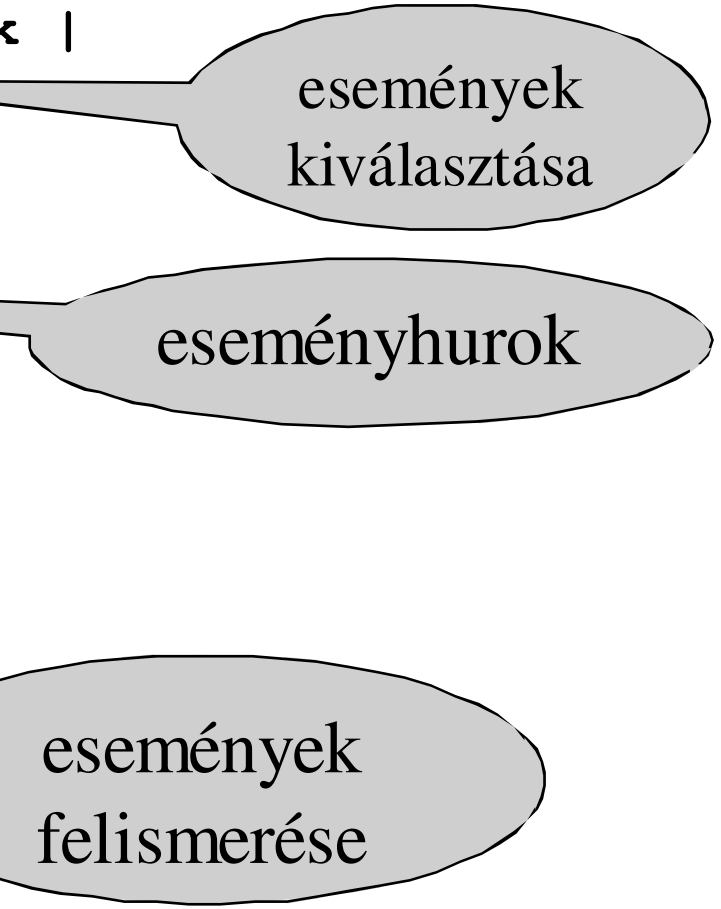
további inicializálások, gc, eseménymaszk, ablakok ...

```
  XMapWindow(display, wMain);
  while(1) {
    XNextEvent(display, &event); ........
  }
  XCloseDisplay(display);
}
```

események kezelése

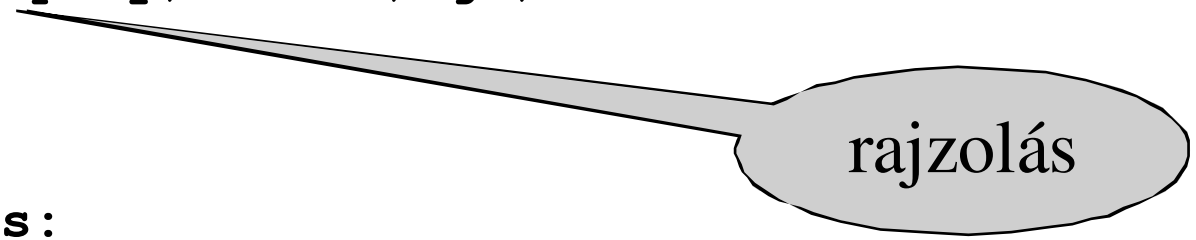
Események kezelése

```
XSelectInput(display, wMain, ExposureMask |
  KeyPressMask | ButtonPressMask |
  StructureNotifyMask);
XMapWindow(display, wMain);
while(1) {
  XEvent          event;
  XNextEvent(display, &event);
  switch (event.type) {
    case ConfigureNotify:
      ...
      break;
```



Események kezelése /2

```
   case Expose:
    ...
    XDrawString(display, wMain, gc, .....
    ...
   break;
   case KeyPress:
   case ButtonPress:
    ...
     if (...) {
        XUnloadFont(display, font_info->fid);
        XFreeGC(display,gc); XCloseDisplay(display);
        exit(1);
     }
   }
}
```



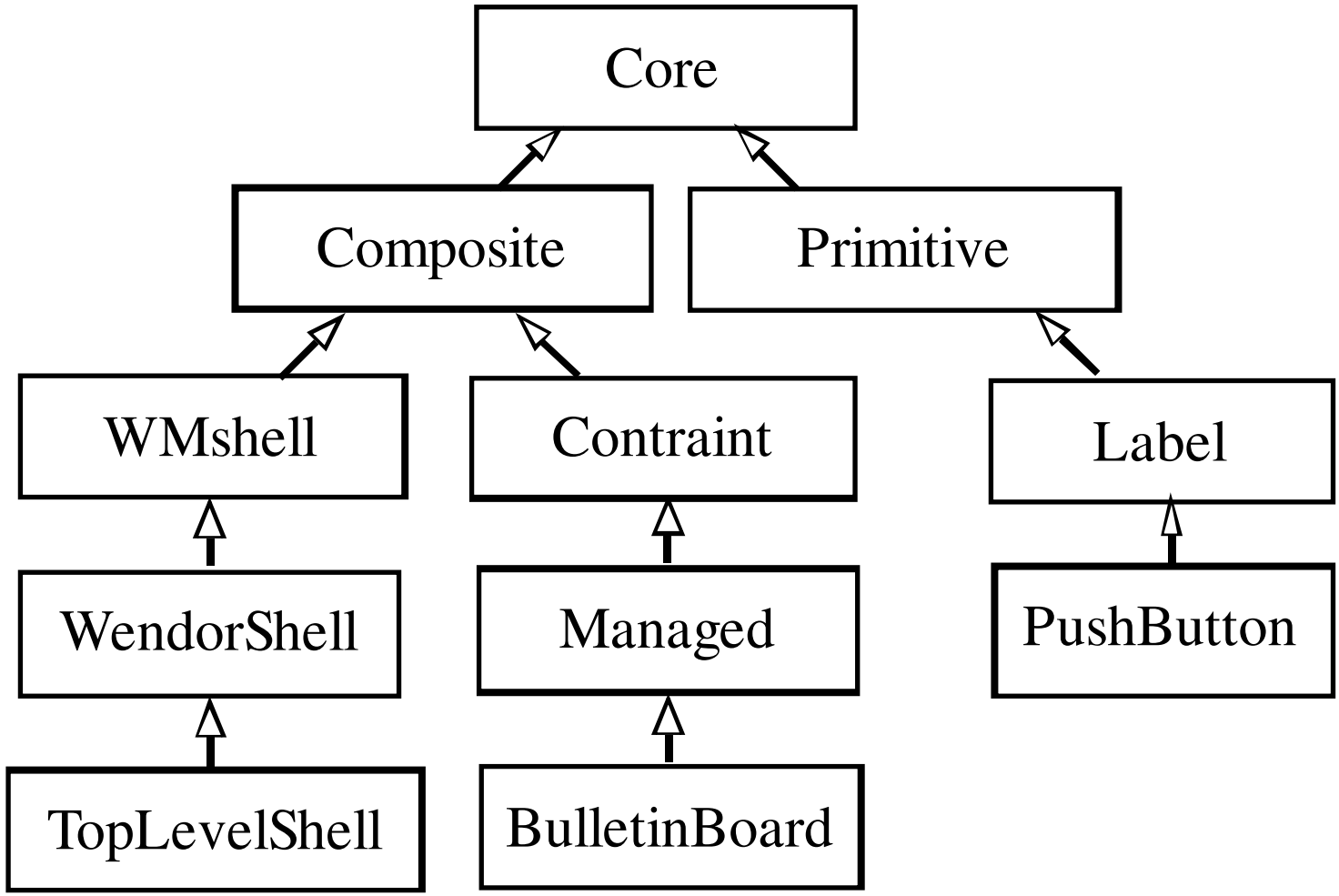
Lehet egyszerűbben ?

- Az X toolkit intrinsic, mint "OO motor" szolgáltatásaira épülő toolkit segítségével.
  - Athena, OpenLook, Motif, CDE, KDE ...
- Objektum orientált (pl. C++) nyelvhez kapcsolódó könyvtárak / toolkitek alkalmazásával
  - Agar, CEGUI, CLX, dlib C++, FLTK, FOX, GLUI, GTK+, IUP, Juce Lgi, Qt, Quinta, Tk, TnFOX, Ultimate++, VCF, wxWidgets, YAAF, XForms, XVT

Toolkit

- Objektum típusokat definiál, melyekkel megvalósíthatók a szokásos GUI elemek
  - label, button, radiobutton, checkbox, editbox, bulletinboard, scrollbar, stb.
- Az objektumok közös őssel rendelkeznek (widget, v. gadget).
- Származtatással újabb objektumok hozhatók létre.
- Az objektumok kommunikálnak az alkalmazással és az X szerverrel.

Motif toolkit hierarchia (részlet)



Motif hello

```
#include <stdio.h>
#include <X11/Intrinsic.h>
#include <Xm/Label.h>
main(int argc, char *argv[]) {
  Widget topLevel, hello;
  topLevel = XtInitialize(argv[0],
          "Motifhello", NULL, 0, &argc, argv);
  hello = XtCreateManagedWidget("hello",
          xmLabelWidgetClass,topLevel, NULL, 0);
  XtRealizeWidget(topLevel);
  XtMainLoop();
}
```

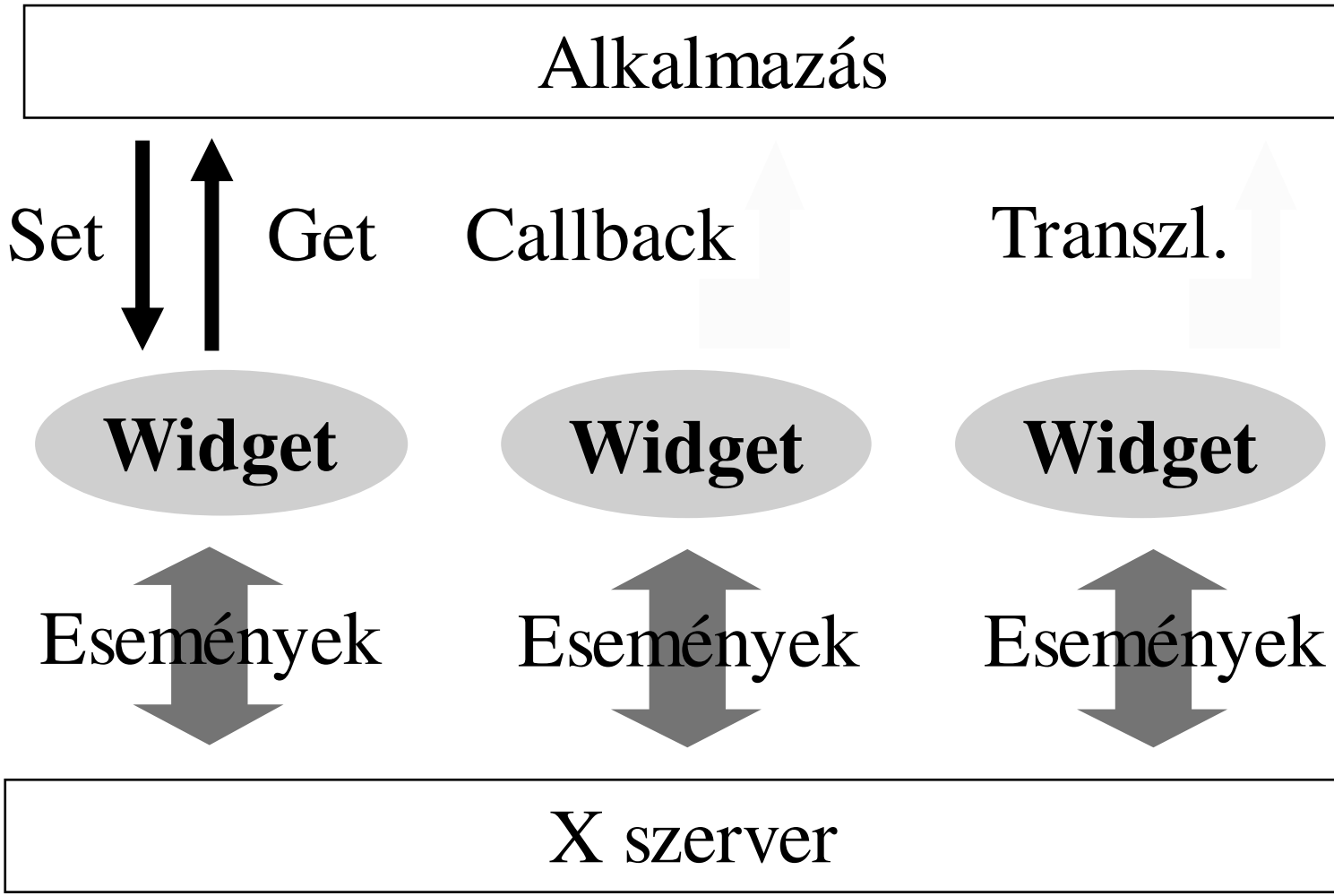
publikus header

new

osztály

eseményhurok

Komunikációs sémák



MS Windows

- Szintén a 80-as évek elején indult
- Hasonló alapelvek:
  - eseményvezérlés
  - raszter orientált grafika
  - objektum orientált szemlélet
- Fő különbségek:
  - az X nem része az OS-nek
  - az X hálózatorientált
  - az X szerver/kliens megközelítésű

MS windows program szerkezete

```
WinMain(.....) { //inicializálások
// ablak mint "objektum" regisztrálása
// menü, kurzor, icon, méret, szín, eseménykezelő, ...
    RegisterClass(.....);
// létrehozás
    InitInstance(.....) ;
// üzenetek feldolgozása:
    while (GetMessage(&msg, NULL, 0, 0))   {
          TranslateMessage(&msg);
          DispatchMessage(&msg);
    }
}
```

MS windows eseménykezelés

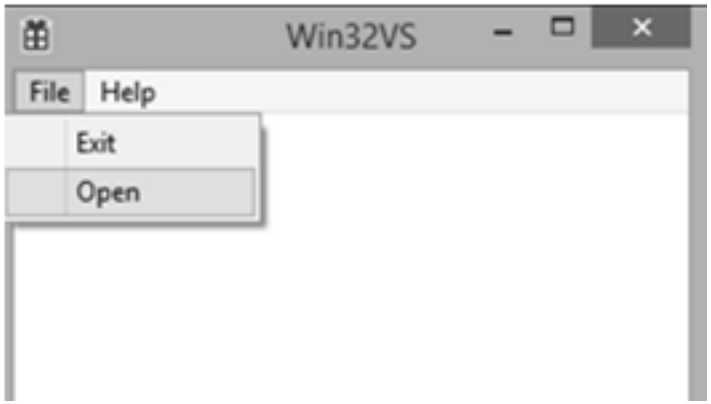
```
WndProc(...) {
    ....
   switch (message)          {
     case WM_PAINT:
            hdc = BeginPaint(hWnd, &ps);
        ....
            EndPaint(hWnd, &ps);
            break;
     case WM_DESTROY:
            PostQuitMessage(0);
            break;
      .....
}
```

V. Studio: new project → visual c++ → win32 project

new project → visual c++ → win32 proj

- Skeleton alkalmazás keletkezik.
- Fordítható, üres ablak, menüje is van.
- GUI elemeit külön szöveges fájlban írjuk le (resouce file, *.rc)
- Külön editorral, vagy szövegesen szerkeszthető.
- Resource compiler lefordítás után (*.res) belegyúrja az exe-be, így együtt hurcolható.

RC file példa




```
IDC_WIN32VS MENU
BEGIN
    POPUP "&File"
    BEGIN
      MENUITEM "E&xit",   IDM_EXIT
      MENUITEM "&Open",   IDM_OPEN
    END
    POPUP "&Help"
    BEGIN
      MENUITEM "&About ...",   IDM_ABOUT
    END
END
```

FOX toolkit

- C++ alapú
- átgondolt (...)
- kis méretű
- platform független (X, MS, Mac)
- számos ma szokásos megoldás
  - perzisztencia,
  - úszó dobozok,
  - buborék tippek

http://www.fox-toolkit.org/

FOX hello

```
int main(int argc, char *argv[]){
  FXApp application("Hello","FoxTest");
  application.init(argc,argv);
  FXMainWindow *main=new FXMainWindow(&application,
            "Hello",NULL,NULL,DECOR_ALL);
  new FXButton(main,"&Hello Fox!", NULL,
            &application, FXApp::ID_QUIT);
  application.create();
  main->show(PLACEMENT_SCREEN);
  return application.run();
}
```

üzenet

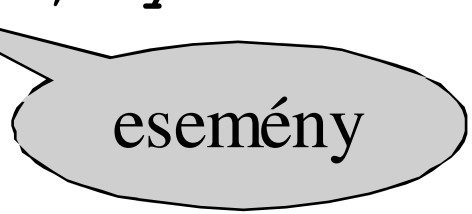
szerver oldalon

eseményhurok

FOX eseménykezelés

- Események kezelését makrókkal felépített táblák segítik. -> callBack függvények

```
FXDEFMAP(myWindow) myWindowMap[]={
  FXMAPFUNC(SEL_COMMAND, myWindow::ID_QUIT,
  myWindow::cbFv),
```



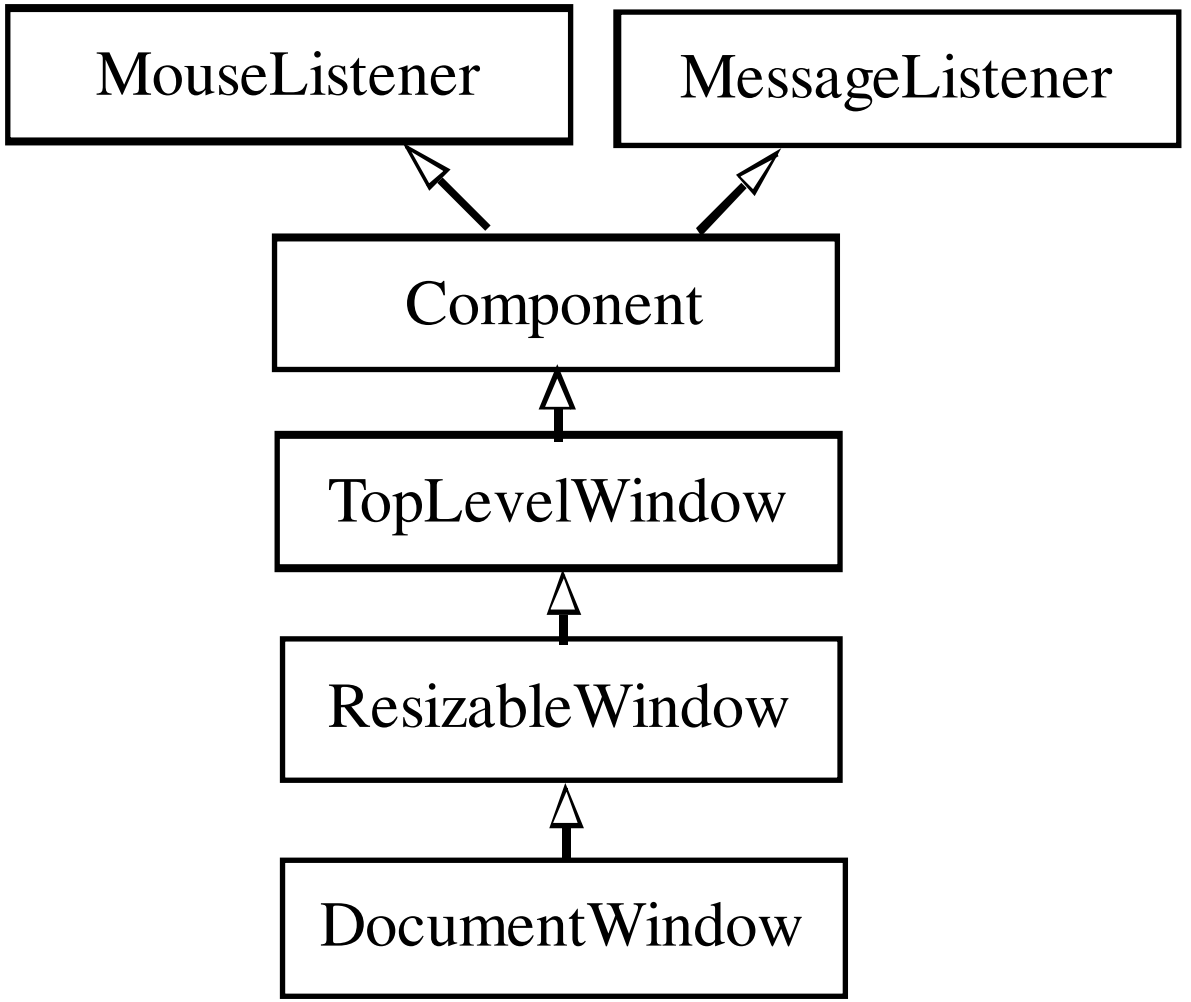
- Függvények egységes üzenetformátumot kapnak:
  - long cbFv(FXObject* sender, FXSelector sel, void *ptr);

JUCE toolkit

- C++ alapú, jobban kihasználja a C++ lehetőségeit
- átgondolt
- kis méretű
- platform független (X, MS, MAC)
- makróktól mentes
- OpenGL integráció
- GPL

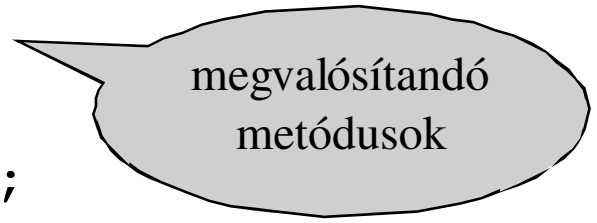
http://www.rawmaterialsoftware.com

Objektum hierarchia példa



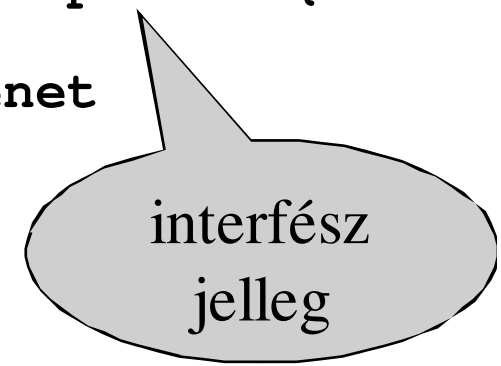
JUCE hello

```
class JUCEHelloApplication :public JUCEApplication {
  HelloWorldWindow* helloWorldWindow;
public:

  void initialise (const String& commandLine) {
    helloWorldWindow = new HelloWorldWindow();
  }

  void shutdown() {
    delete helloWorldWindow;
  }
};

START_JUCE_APPLICATION (JUCEHellodApplication)
```



JUCE hello /2

```
class HelloWorldContentComponent :public Component {
public:
  void paint (Graphics& g) {  // paint üzenet
    g.fillAll (Colours::white);
    g.setColour (Colours::black);
    g.setFont (20.0f, Font::bold);
    g.drawText (T("Hello World!"),
       0, 0, getWidth(), getHeight(),
                    Justification::centred, false);
    }
};
class HelloWorldWindow :public DocumentWindow {
public:
   HelloWorldWindow() :DocumentWindow (
     T("Hello World"), Colours::yellow,
     DocumentWindow::allButtons, true ) {
        setContentComponent (
         new HelloWorldContentComponent());
        setVisible (true);
};
```



FLTK toolkit

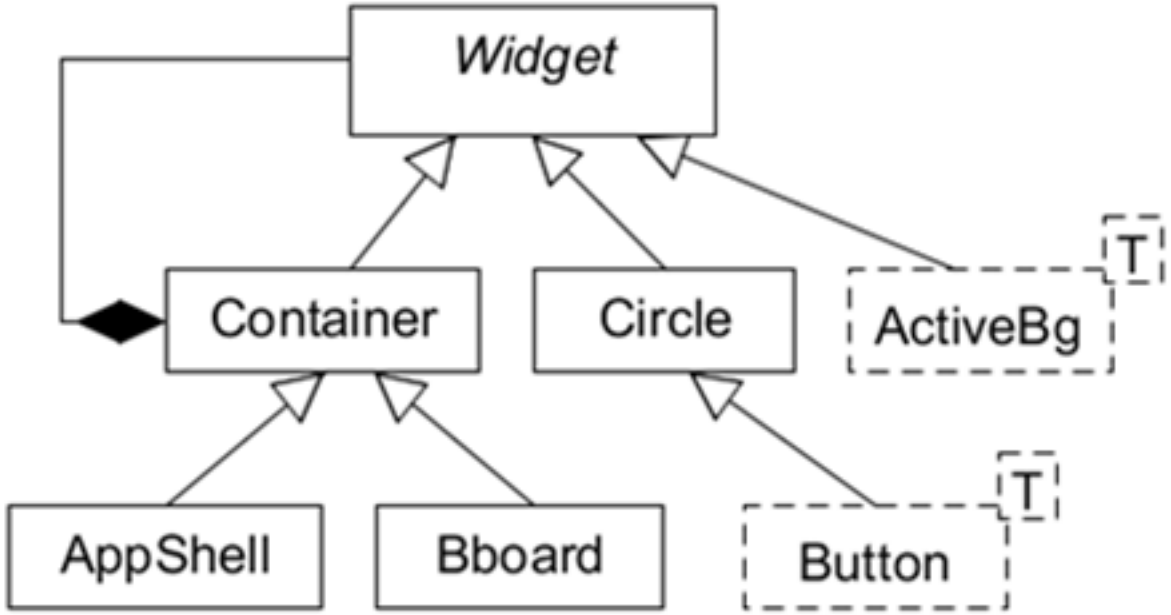
- C++ alapú, kihasználja a C++ lehetőségeit
- kis méretű
- FLUID (Fast Light User-Interface Designer)
- platform független (X, MS, MAC)
- makróktól mentes
- OpenGL integráció, GLUT kompatibilis
- GNU
- uCFLTK mikrokontrollerekhez

http://www.fltk.org

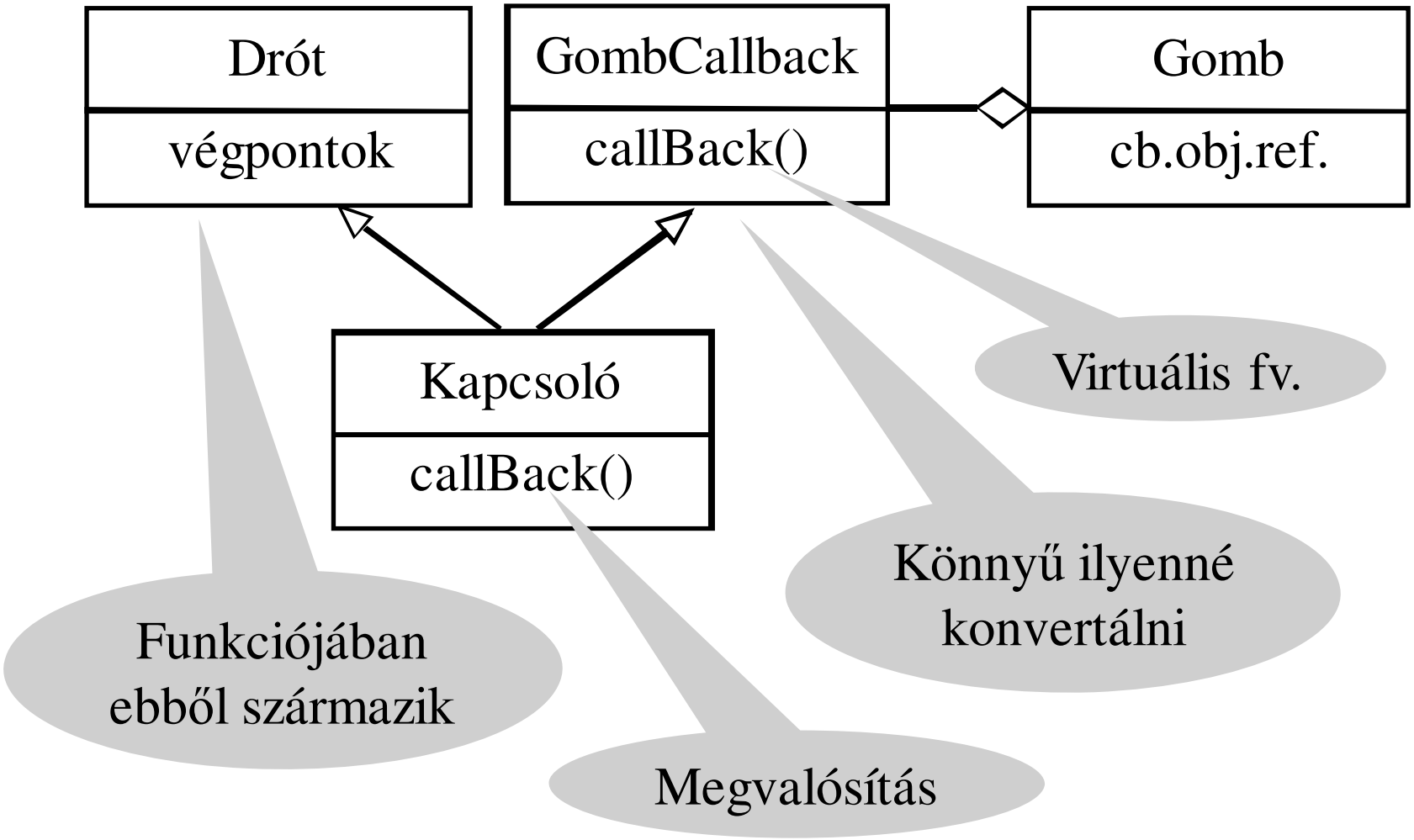
FLTK hello

```
class Hello :public Fl_Window {
  static void quit(Fl_Widget*,void*) {// privát m.
    exit(0);
}
public:
  Hello(int w, int h, const char *n=0)
                              :Fl_Window(w, h, n) {
    Fl_Button *bt =
         new Fl_Button(10, 10, 100, 25, "Exit");
    bt->callback(quit); // cb. fv. összerendelés
    callback(quit);
    show();                 // megjelenítés
  }
};
int main() {
    Hello hel(400, 200, "Hello");
    return Fl::run();    // eseménykezelés indul
}
```

SDL_bboard demo



callBack újra



Kapcsoló megvalósítása

```
class GombCallback {            // callback funkcióhoz
public:
   virtual void callBack() = 0; // virtuális cb. függvény
};

class Gomb {     // felhasználói felület objektuma
   GombCallback &cb;          // objektum referencia
public:
   Gomb (GombCallback &t) :cb(t) {}// referencia inic.
   void Nyom() { cb.callBack(); }        // megnyomták
   ....
};
```

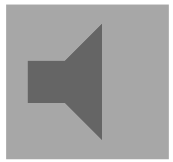
Kapcsoló megvalósítása/2

```
class Kapcsolo :public Drot, public GombCallback {
    int be;                              // állapot
public:
    void ki();
    void be();
   void callBack() { if (be) ki(); else be(); } // callback
};
...
Kapcsolo k1;
Gomb g1(k1); // kapcsoló és a callBack fv. összerendelése
```

signal/slot mechanizmus

- Az örökléssel megvalósított callback mechanizmus nagyon szoros csatolást jelent a két objektum között, ráadásul nem típusbiztos.
- signal/slot lényegesen lazább csatolást jelent.

 → signal

slot → 

```
Boost:
struct Hello {
   void operator()() { ..... }
};
boost::signal<void ()> sig;
Hello hello;
sig.connect(hello);
sig();
```

signal/slot mechanizmus/2

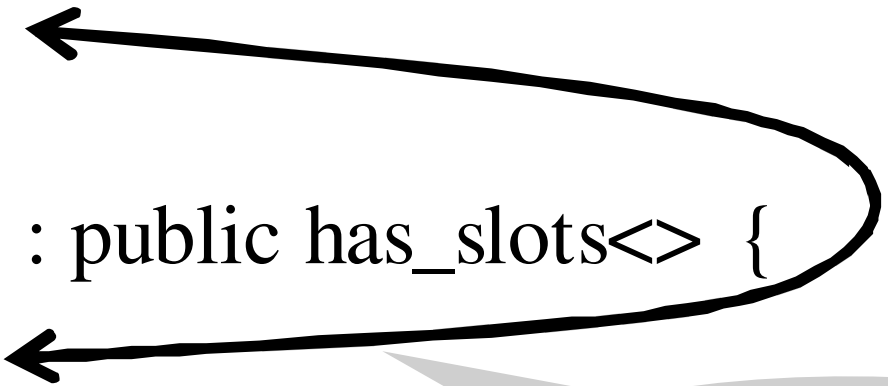
- Ez a mechanizmus világosabban lehetővé teszi a callback függvények összerendelését.
- Kevésbé szoros csatolás ad, ugyanakkor paraméterezhető és típusos.

```
boost::signal<float (int, int)> sig;
cout << sig(1, 2);
```

- Megvalósítás:
  - template-tel (boost, sigslot)
  - preprocesszorral (Qt)

Példa: sigslot lib

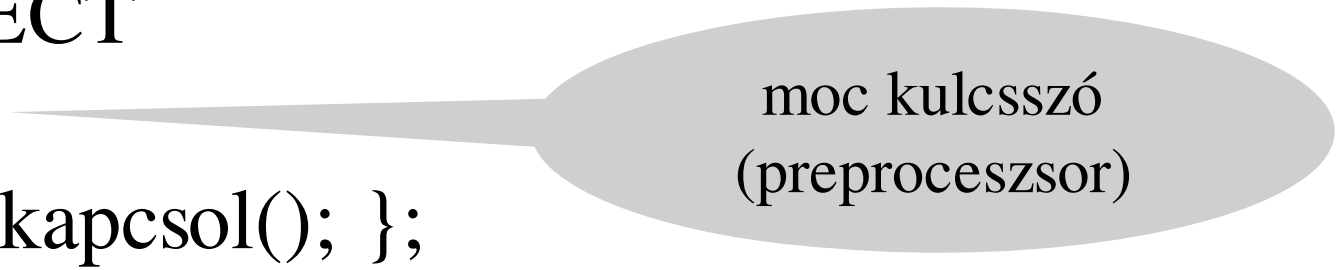
```
struct Gomb {
   signal0<> kapcsol;
    …kapcsol();
};
struct Kapcsolo : public has_slots<> {
      void be();
};
Gomb g1; Kapcsolo k1;
g1.Kapcsolo.connect(&k1, &kapcsolo::be);
```



A kapcsol fv. hívás operátorát összeköti

Példa: Qt

```
struct Gomb : public QObject{
   Q_OBJECT
   signals:
        void kapcsol(); };
struct Kapcsolo :  public QObject{
   Q_OBJECT
   public slots:
        void be();
};  // moc (Meta-object compiler)
```



Qt toolkit

- C++ alapú
- kiegészítő utasítások → előfeldolgozó
- platform független (X, MS, MAC)
- OpenGL integráció, GLUT kompatibilis
- 2008: Nokia
- 2009-től LGPL, QPL, és üzleti licenc
- 2011: Digia
- számos további nyelv:
  - Phyton, C#, Ruby, Ada, Perl, PHP, Haskell
- Migrációs lehetőségek (MFC, Motif)

http://qt-project.org, http://qt.digia.com/

Qt hello

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

int main(int argc, char *argv[]) {
  QApplication app(argc, argv);

  QPushButton hello("Hello world!");

  connect(&hello, SIGNAL(clicked()),&app,
                                  SLOT(quit()));
  hello.resize(100, 30);

  hello.show();
  return app.exec();
}
```

Qt platformok

- Android
- Blackberry
- iOS
- Linux/X11
- Mac OS X
- ~~Simbian~~
- Windows
- Windows CE
- Rasberry Pi

Wt toolkit

- Egy érdekes példa a Wt (Web Toolkit)
- Segítségével teljesen C++-ban írhatunk meg egy web alkalmazást, ami minden szokásos dolgot tartalmazhat pl. sessionkezelést is.
- Nem kell ismerni egyéb technológiát mint pl: HTML, CSS, Java, php, stb.

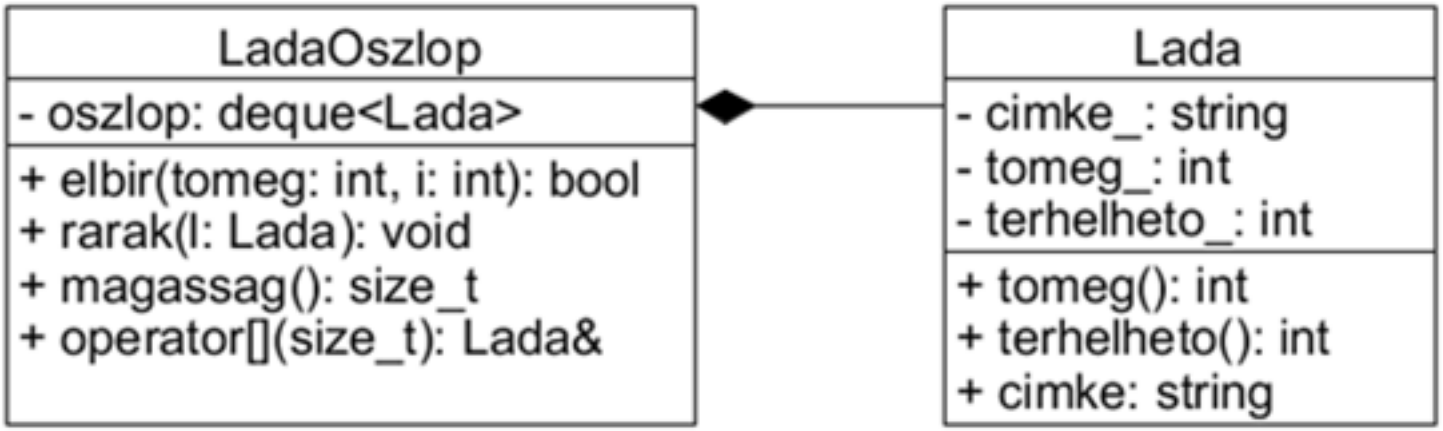
  http://www.webtoolkit.eu

Wt hello

```
WApplication *createAppl(const WEnvironment& env){
  WApplication *appl = new WApplication(env);
  appl->setTitle("Hello world!");
  appl->root()->addWidget(
      new WText(L"<h1>Hello, World!</h1>"));
  WPushButton *Button = new
      WPushButton(L"Quit", appl->root());
  Button->clicked.connect(SLOT(appl,
      WApplication::quit));
  return appl;
}
int main(int argc, char **argv){
  return WRun(argc, argv, &createApplication);
}
```

13. heti labor példa

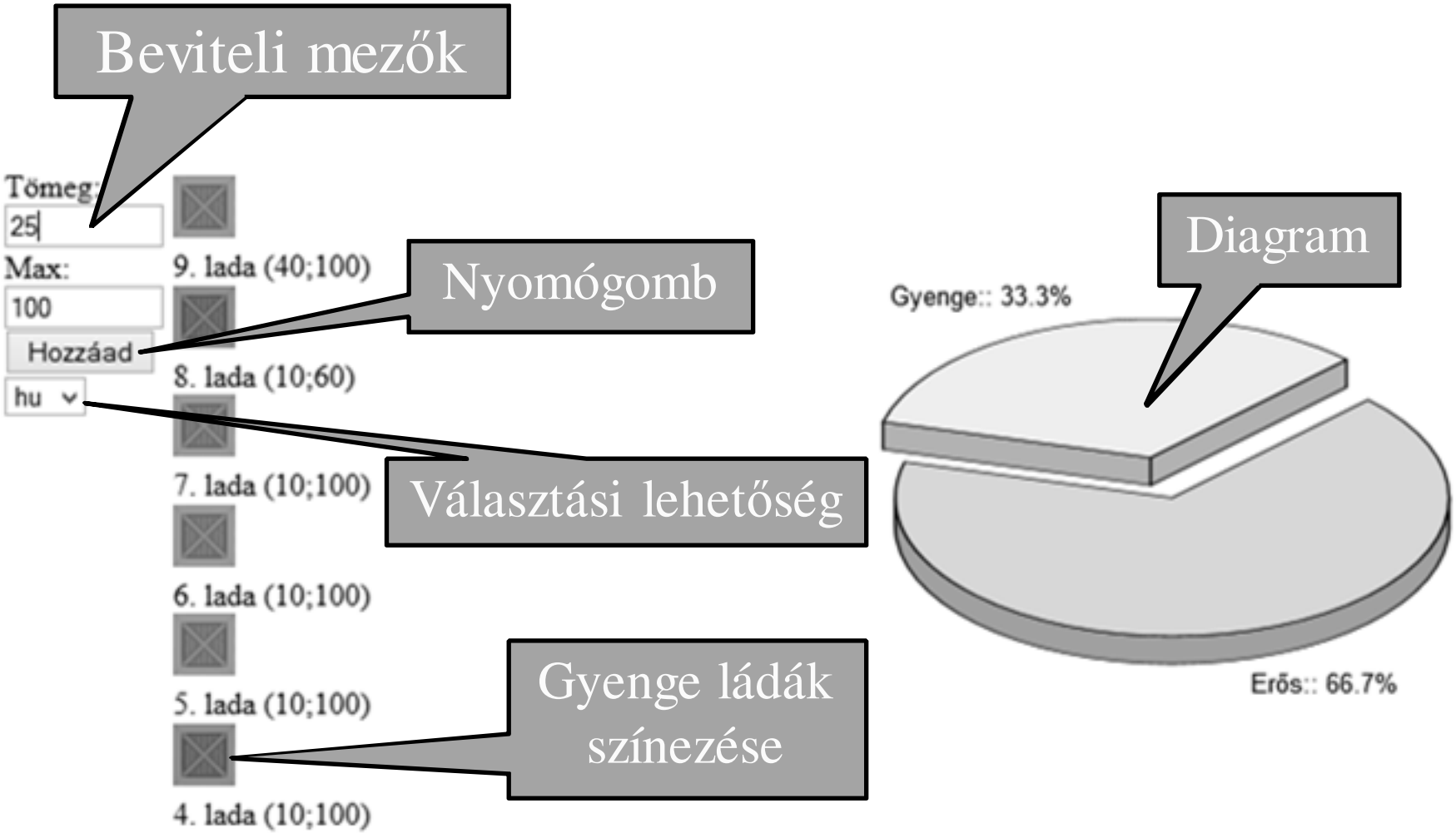
- Ládákat modellezünk. Minden ládának van felirata, teherbírása és tömege.
- A ládákból ládaoszlopokat építünk. A teherbírás túllépésekor a láda összetörik.



LadaOszlop

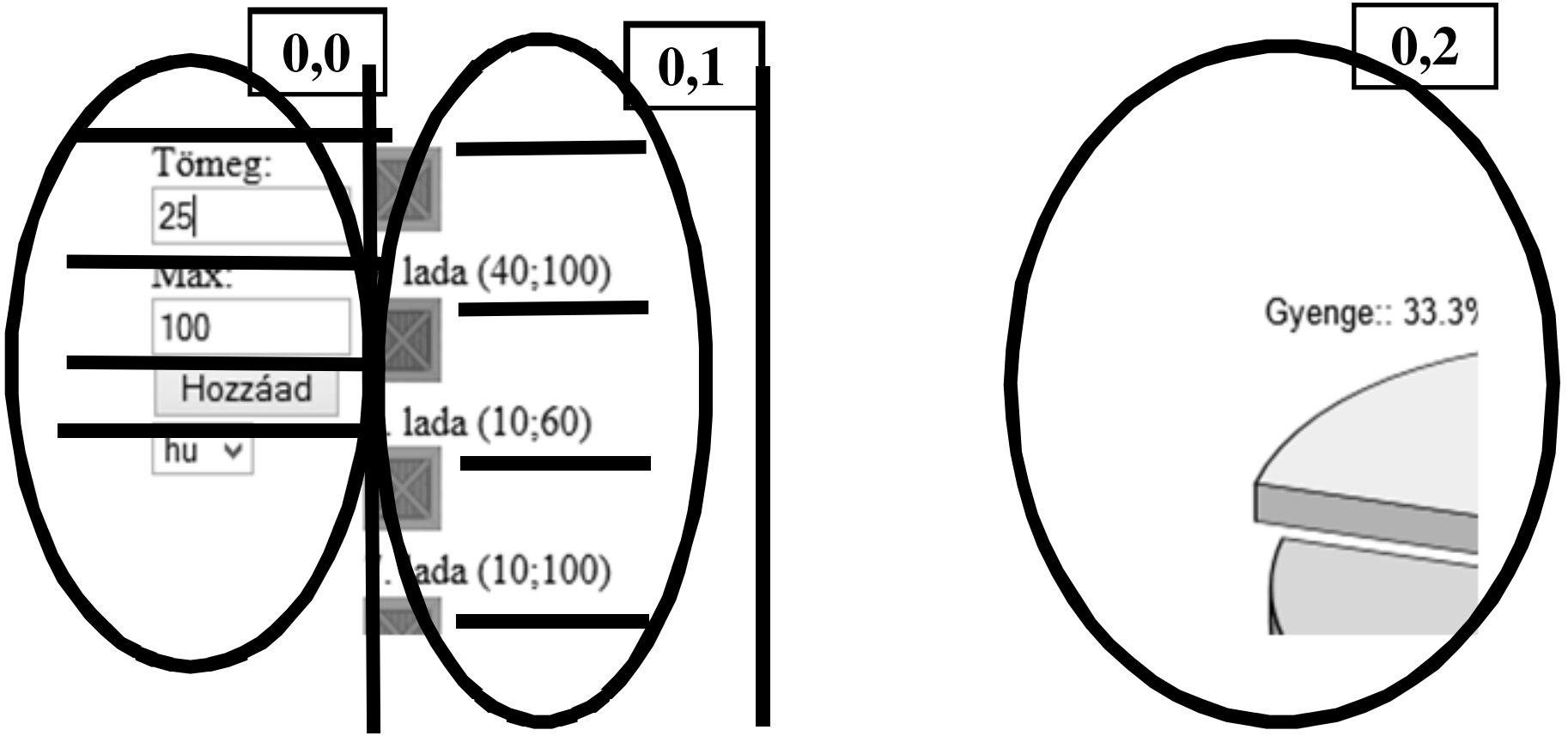
```
bool LadaOszlop::elbir(int t, size_t i) const {
  for (size_t j = 0; j < i; j++)
    t += oszlop[j].tomeg();
  return oszlop[i].terhelheto()>=t;
}

void LadaOszlop::rarak(Lada l) {
  for (int i=oszlop.size()-1; i>=0; i--)
    if (!elbir(l.tomeg(),i))
      oszlop.erase(oszlop.begin()+i);//random iter
  oszlop.push_front(l);
}

Lada& LadaOszlop::operator[](size_t i) {
    return oszlop[i];
}
```

Megjelenítés



Fő grafikus elemek helye

Táblázatokkal elhelyezzük a widgeteket.
A ládaoszlop belül egy újabb táblázat.



App widget

```
class LadaApplication : public WApplication {... };
LadaApplication::LadaApplication(...) {
  WTable* layout = new WTable();
  root()->addWidget(layout);
  WTable* panel = new WTable(layout->elementAt(0,0));
  ladak = new LadaWidget(layout->elementAt(0,1));
  layout->elementAt(0,1)->setPadding(5);
  WLabel* label = new WLabel(WString::tr("TOMEG"),
                             panel->elementAt(0,0));
  tomegLine = new WLineEdit("2",
                             panel->elementAt(0,0));
  tomegLine->setValidator(new WIntValidator(0, 200));
  label->setBuddy(tomegLine); // a felirat melle
  WPushButton* button=new WPushButton(WString::tr("AD"),
                             panel->elementAt(2,0));
```

LadaWidget

```
class LadaWidget : public Wt::WTable,
                           public LadaOszlop {...};
void LadaWidget::hozzaad(Lada l) {
                  rarak(l); rajzol(l.tomeg()); }
void LadaWidget::rajzol(int t) {
  WTable::clear();    // töröljük ládákat
  for (size_t i = 0; i < magassag(); i++) {
    WLabel *l = new WLabel((*this)[i].cimke(),
                                    elementAt(i,0));
    WImage *kep;
    if (elbir(t,i))
      kep = new WImage("img/chest.png");
    else
      kep = new WImage("img/chest-red.png");
    l->setImage(kep);
}
```

Felhasználói input

Egysoros editbox

```
tomegLine = new WLineEdit("2",
                    panel->elementAt(0,0));
```

Nyomógomb esemény bekötése:

```
button->clicked().connect(this,
                  &LadaApplication::hozzaad);
```

Billentyű felengedése az editboxban:

```
tomegLine->keyWentUp().connect(this,
        &LadaApplication::tomegValtozott);
```

Internationalization & Localization

- i18n = internatilozation
- l10n = localization (kulturális beágyazás)

Különböző jelek, formátumok

- dátum, idő,
- pénznem, jelek, mértékegységek

Különböző nyelveken

i18n és l10n támogatása

- GNU gettext
- Wt::WMessageResourceBundle
- OASIS XLIFF
- POSIX catalogs
- Qt ts/tm
- Java properties
- Windows resources
- …

WMessageResourceBundle

- A Wstring::tr metódusa egy belső azonosító alapján a nyelvi környetenek megfelelő xml formátumú fájlból olvassa ki szöveget. pl:

```
WPushButton(WString::tr("HOZZAAD"),...
```

app.xml

```
<messages>
  <message id='TOMEG'>Tömeg:</message>
  <message id='HOZZAAD'>Hozzáad</message>
</messages>
```

app_en.xml

```
<messages>
  <message id='TOMEG'>Mass:</message>
  <message id='HOZZAAD'>Add</message>
</messages>
```

Fejlesztés támogatása

- Sok állomány → egy alkalmazás
- Hogyan és melyiket kell lefordítani?
- Melyik változat az legutolsó?
- IDE (integrált fejlesztő eszköz)
  - fordítást belső eszközzel támogatja (nyelvet ismeri)
  - verziókövetést külső eszközzel
- Önálló univerzális eszközök
  - nem csak az adott nyelvhez

make

- Egy szöveges leírás *(Makefile), és az állományok módosítási ideje* alapján végrehajtja cél (program, dokumntáció stb.) előálításához szükséges parancsokat.
- Makefile:
  - makró definíciók,
  - függőségi információk (szabályok és implicit szabályok)
  - végrehajtható parancsok
  - megjegyzések

Makefile szerkezete

Makródefiníció:

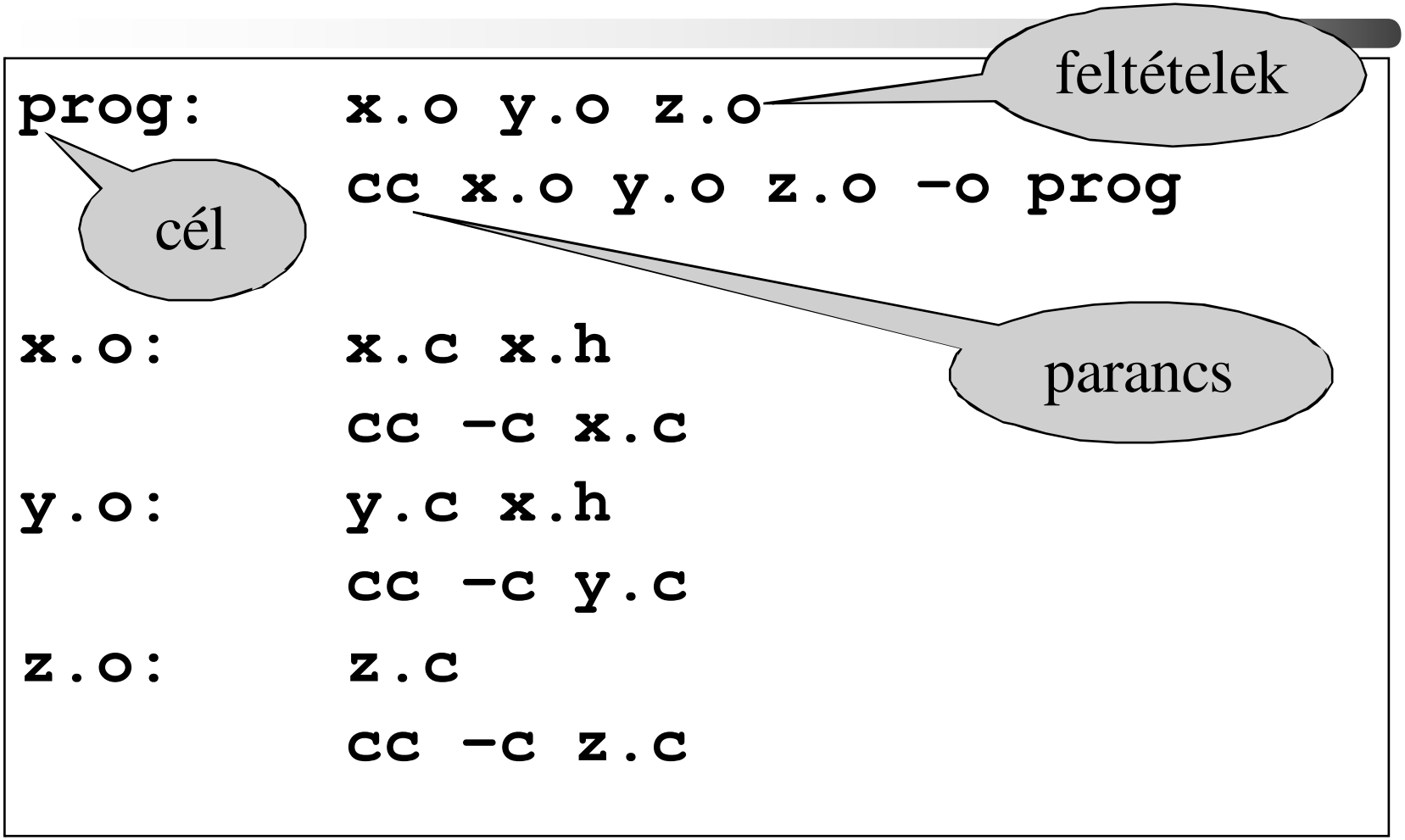
```
makró_név = string
```

Szabályok:

cél1 [cél2] :[:] [feltétel1...] [;parancsok] [#...]

[<TAB>parancsok][#...]

make példa



!!!!!! A parancsok előtt <TAB> van !!!!!

make példa/2

```
OBJECTS = x.o y.o z.o
HEADS = x.h
prog:    $(OBJECTS)
         $(CC) $(OBJECTS) -o prog
$(OBJETS): $(HEADS)
```

```
.c.o:
    $(CC) -c $<
```

implicit szabály miatt a **.o**-k egyértelműen előállíthatók.

make változatok

Számos változata és kiegészítése van:

- make (eredeti)
- BSD make
- GNU make
- nmake (Microsoft)
- CMake – cross platform make