# flext

## C++ layer for Pure Data & Max/MSP externals

Thomas Grill Vienna, Austria

http://www.parasitaere-kapazitaeten.net

#### Pure Data & Max/MSP

- Modular real-time frameworks for audio (and video)
- Pure Data for Linux, IRIX, Mac OSX and Windows
- Max/MSP for Mac OS9/X and Windows
- Asynchronous (messages) and synchronous (DSP) processing
- Functionality made up of loadable modules ("externals")
- Both systems originally developed by Miller Puckette
- APIs have a lot in common

## flext

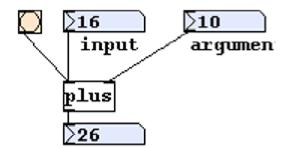
- Library with C++-interface to the PD or Max API
- External objects represented as classes
- Wrappers for various other functions
- Covers nearly all features of the respective real-time framework
- Cross-platform (PD & Max, OS independence)
- All major compilers supported

## Simple message object – PD

```
#include <m pd.h>
                                                                          ≥16
                                                                                      >10
typedef struct {
                                                      input
                                                                           input
                                                                                      argument
    t object x obj;
    t float x f1,x f2;
                                                                          plus
                                                     plus 3
} t_plus;
static t class *plus class;
                                                                          ∑26
                                                     ∑12
static void *plus new(t floatarg f) {
    t plus *x = (t plus *)pd new(plus class);
    outlet new(&x->x obj, &s float);
    floatinlet new(&x->x obj, &x->x f2);
    x->x f1 = 0;
    x->x f2 = f;
    return (x);
}
static void plus bang(t plus *x) { outlet float(x->x obj.ob outlet, x->x f1 + x->x f2); }
static void plus float(t plus *x, t float f) {
    outlet float(x->x obj.ob outlet, (x->x f1 = f) + x->x f2);
}
void plus setup(void) {
    plus class = class new(gensym("plus"), (t newmethod)plus new, 0, sizeof(t plus), 0,
                 A DEFFLOAT, 0);
    class addbang(plus class, plus bang);
    class addfloat(plus class, (t method)plus float);
}
```

## Simple message object – flext

```
#include <flext.h>
                                                         input
class plus: public flext base
    FLEXT HEADER(plus,flext base)
                                                       plus 3
    float f1,f2;
                                                        ∑12
    plus(float f): f1(0),f2(f) // constructor
         AddInAnything("Bang or Number");
         AddInFloat("Argument");
         AddOutFloat("Result");
         FLEXT ADDBANG(0,m bang);
                                      // add methods
         FLEXT ADDMETHOD(0,m float);
         FLEXT ADDMETHOD(1,m arg);
    }
    inline void m bang() { ToOutFloat(0,f1+f2); }
    inline void m float(float f) { ToOutFloat(0,(f1 = f)+f2); }
    inline void m arg(float f) { f2 = f; }
    FLEXT CALLBACK(m bang) // callback wrappers
    FLEXT CALLBACK F(m float)
    FLEXT CALLBACK F(m arg)
};
FLEXT NEW 1("plus", plus, float0) // set up class
```



#### **Pros and Cons**

- Easy to read, easy to write
- Hardly any system-dependent code necessary
- Use advantages of C++ (esp. polymorphism)
- Convenience functions
- Extensions (attributes, multi-threading etc.)
- Processing overhead
- Memory consumption
- File size of statically linked binaries
- Building problems

## **Building parts**

- Detection of platform and compiler
- Two base classes
  - flext\_base for pure message objects
  - flext\_dsp for DSP objects
- Static flext class for general system functionality
  - Buffer class (access to sampled audio)
  - Atom list classes
  - Threading classes
  - Timer class and functions
  - Some SIMD functions
  - Messaging functions

#### **Attributes**

- Consistent way to initialize, set and query state
- Originally conceived by J.K.Clayton for Max/Jitter

```
class plus: public flext base
                                                                    argument get/set
                                                                               >10
                                                         input
    FLEXT HEADER(plus,flext base)
                                                                               arq $1
                                                         >1.5
                                                                      getard
    float f1,f2;
    plus(float f): f1(0),f2(f)
                                                                  plus @arq
         AddInAnything("Bang or Number");
                                                                                 attribute dum
         AddInFloat("Argument");
         AddOutFloat("Result");
                                                                                 prepend set
                                                         >25
         FLEXT ADDBANG(0,m bang);
                                                                                 arg 10
                                                         result
         FLEXT ADDMETHOD(0,m float);
         FLEXT ADDMETHOD(1,m arg);
         FLEXT ADDATTR_VAR1("arg",f2); // add attribute
    }
    FLEXT ATTRVAR F(f2) // wrapper for attribute variable
    // ... more methods and wrappers
};
FLEXT_NEW_1("plus",plus,float0) // set up class
```

## **DSP** objects

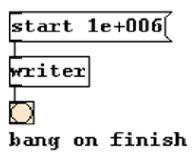
Virtual methods for DSP processing

```
signal
                                                                                       gain
class gain: public flext dsp
                                                                            noise~
                                                                                       >-20
    FLEXT HEADER(gain,flext dsp)
    float factor;
                                                                               gain~
    gain(): factor(0)
         AddInSignal("In"); // add signal inlet
         AddInFloat("Gain");
                                                                              dac~ output
         AddOutSignal("Out"); // add signal outlet
         FLEXT ADDMETHOD(1,m gain);
    }
    virtual void m signal(int n,t sample *const *insigs,t sample *const *outsigs)
         const t sample *in = insigs[0];
         t sample *out = outsigs[0];
         while (n--) * (out++) = *(in++)*factor; // or: MulSamples (out, in, factor, n);
    }
    void m gain(float db) { factor = pow(10.f,db*0.05f); }
    FLEXT CALLBACK F(m gain);
};
FLEXT NEW DSP("gain~",gain) // set up DSP class
```

## **Threading**

- Transparent threading in real-time environment
- Threads are running at lower priority
- Several threading models supported (pthreads as default)

```
class writer: public flext base
    FLEXT HEADER(writer, flext base)
    writer() {
         AddInAnything("Bang to start");
         AddOutBang("Bang on finish");
         FLEXT ADDMETHOD I(0, "start", m start); // register method
    }
    void m start(int loops) {
         // slowly write some megabytes to disk
         std::ofstream file("/temp/bigfile.txt");
         for(int i = 0; i < loops; ++i) file << i << ' ';
         file << std::endl;</pre>
         ToOutBang(0); // indicate end of work
    }
    FLEXT THREAD I(m start) // special wrapper
};
FLEXT NEW("writer", writer)
```



#### **Timers**

- Timer class using class methods or callback functions
- Additional high-precision clock functions

```
class bangs: public flext base
    FLEXT HEADER(bangs, flext base)
    Timer timer;
    bangs() {
         AddInAnything("In");
         AddOutBang("Out");
         FLEXT ADDMETHOD(0,m interval);
         FLEXT ADDTIMER(timer, m bang);
    }
    inline void m interval(float intv) { timer.Periodic(intv*0.001); }
    inline void m bang(void *) { ToOutBang(0); }
    FLEXT CALLBACK F(m interval);
    FLEXT CALLBACK T(m bang); // timer callback
};
FLEXT NEW("bangs", bangs) // set up class
```

1000

bangs

#### More features

- Binding to symbols (named senders & receivers)
- Libraries of externals
- Interface classes to SndObj and STK frameworks

#### **Future**

- flext as a shared library
- System-wide inspectors for flext-based objects
- Better integration into Pure Data?
- Better support for Max messaging/timing
- Integrated make system for *flext* externals

#### **Download**

- Release versions from the flext website
- Fresh stuff from Sourceforge CVS

## **Prerequisites**

- Real-time system SDK
- Compiler (free: gcc, cygwin, MinGW)

## **Support**

- PD and Max mailing lists
- flext online documentation and tutorials
- Open source externals

## **Applications**

- fluid~ soundfont external by Frank Barknecht
- tbext object collection by Tim Blechmann
- readanysf~ sample/stream-player by August Black
- xsample a collection of sample objects
- vasp modular vector assembling signal processor
- py/pyext Python-based external objects
- pool a hierarchical database object
- dyn~ dynamic object management
- fftease library of spectral processors