

---

Subject: true dynamic dispatching with Upp?

Posted by [kohait00](#) on Thu, 18 Oct 2012 07:39:36 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

hi all...

i bumped into this one which costs me some head ache, because my curent solution is slow in performance.

is there any nice upp (or any) way to do this in c++?

i have a Elements, that would really like to keep untouched (so many design patterns don't go here, besides beeing not sufficient)..

the Elements are polymorph and should be represented in a View somewhere. For this pupose, i have some representation Editors (Display is not enough here). so i run a list of Elements checking all the Elements and try to figure out, which ElementEditor is fit for it. The check is currently done with `dynamic_cast<>`, but i want to avoid it (but i fear it is not possible). This is true dynamic type dispatching, so probably it's the only solution.

Things like Visitor pattern and Strategy pattern wont fit here as far as i can tell (because they imply extending the Element and expect Element to know all possible Editors)..

Strategy pattern would be sort of cached Info/Context stored in Element, of which Element is not aware of. Don't want that either.

So what are you guys telling me here? is a type Map the only Thing besides `dynamic_cast<>`? using typeid? for best performance?

```
#include <Core/Core.h>
using namespace Upp;
```

```
class Element
{
//some base class interface
};
```

```
class ElementA : public Element
{
//some concrete implementation of Element
};
```

```
class ElementB : public Element
{
//another concrete implementation of Element
};
```

```
// the above classes dont mustn't know anything of the following classes
```

```

class ElementEditor
{
// generic Element editing
};

class ElementAEditor : public ElementEditor
{
// specific/additional ElementA editing
};

class ElementBEditor : public ElementEditor
{
// specific/additional ElementB editing
};

//

void EditElement(const Element& e)
{
//how to dynamicly dispatch to the right Element Editor? truly, depending on runtime type of
Element?

//cant use Visitor pattern (implies extending Element with a Visitor interface, which knows of all
Editors, Bad!!!)
//cant use Strategy pattern (implies extending Element with additional hook to invoke a plugged in
editor, is more or less a cache of a once chosen editor dispatching)

//is dynamic_cast<> option the only one possible?

if(ElementA* p = dynamic_cast<ElementA*>(&e)) { /*invoke ElementAEditor*/ }
else
if(ElementB* p = dynamic_cast<ElementB*>(&e)) { /*invoke ElementBEditor*/ }
else { /* invoke ElementEditor as fallback */ }
}

```

---