Subject: Re: Arrys vs Vectors
Posted by kohait00 on Tue, 21 Dec 2010 10:20:44 GMT
View Forum Message <> Reply to Message

1) in general, no. if OTOH you use the containers in a 'high performance scenario, loops, creation, destruction, adding, it might be of interest to use Vector, especially for intrinsic types. but for GUI applications as storage for some user data structure, it has no perceptible difference.

2) the most important difference is, that Vector holds the mem space for the objects itself in a sequence and creates the objects (calls constructor) there, while Array creates the objects somewhere else on the heap, and stores only the pointers to them. thats why Adding to Vector can render any previous referenece to an object inside it as invalid, thats also why the objects need to be Moveable<>. see manual. Arrays dont need those precautions. in Vector, you cant Detach an element, and Attach() it in another Vector, in Array, you can, whithout even touching the element itself (handling over pointers in Array).
programming with both is quite the same, infact, Upp has MACROS which apply to a variety of Containers, because they have a huge subset of common Methods, like GetCount(), Add(), etc. debugging is not really easy, since Upp doesnt yet (AFAIK) support expressions, so things like looking at vec[2] are not possible. but you can look at v[0] indirectly. inspectiong them in terms of count, allocated mem etc is no problem.

3) explained above, they both lack the same. maybe try to hassle with the Vector<void*> vector inside the Array, '*(MyClass*)arr.vector[12]' or sth.

4) depends on use case and whether it really matters, best is to check the implementations, they are not too hard to understand, knowing the basics from manual.

5) correctly speaking, if you need references in which ever way to classes, beeing it pointers, or refs, that should not get invalid once you manipulate the container, (adding is dangerous, needs growing and 'moving' objects in Vector), Array is the only choice. it also grows, but only moves the pointers to the objects, which remain in same place.

6) Vector<> for intrinsic types, Buffers, 'oldschool' arrays (MyType ar[12] replacement, Arrays, if type is quite complex, or need to 'move' them around, by detaching/ataching somewhere else.