Subject: Pick Semantics: Passing a Vector to a method
Posted by NeilMonday on Thu, 20 Jun 2013 19:42:50 GMT
View Forum Message <> Reply to Message

I am learning about pick semantics now. Lets assume I wanted to print a Vector of Strings two times. Am I right in assuming that this will not work:

```
void PrintNames(Vector<String> names)
{
   for(int i=0; i<names.GetCount(); i++)
   {
      //print the name
   }
}

int main()
{
   Vector<String> myNames;
   //add a few Strings to myNames here...

   PrintNames(myNames);
   //myNames is not valid anymore?
   PrintNames(myNames);
}
```

Instead I will need to do this:

```
void PrintNames(Vector<String> names)
{
   for(int i=0; i<names.GetCount(); i++)
   {
      //print the name
   }
}

int main()
{
   Vector<String> myNames;
   //add a few Strings to myNames here...

   Vector<String> tempNames;
   tempNames <<= myNames;
   PrintNames(tempNames);

   tempNames <<= myNames;
   PrintNames(tempNames);
}
```

Subject: Re: Pick Semantics: Passing a Vector to a method
Posted by dolik.rce on Fri, 21 Jun 2013 04:24:54 GMT
View Forum Message <> Reply to Message

Hi Neil,

Welcome to the forum

NeilMonday wrote on Thu, 20 June 2013 21:42I am learning about pick semantics now. Lets assume I wanted to print a Vector of Strings two times. Am I right in assuming that this will not work:

```
void PrintNames(Vector<String> names)
{
   for(int i=0; i<names.GetCount(); i++)
   {
      //print the name
   }
}

int main()
{
   Vector<String> myNames;
   //add a few Strings to myNames here...

   PrintNames(myNames);
   //myNames is not valid anymore?
   PrintNames(myNames);
}
```

Yes, you understand that correctly. The first call would "pick" the contents of the original vector and the second call would show nothing or even thrown error.

NeilMonday wrote on Thu, 20 June 2013 21:42Instead I will need to do this:

```
void PrintNames(Vector<String> names)
{
   for(int i=0; i<names.GetCount(); i++)
   {
      //print the name
   }
}

int main()
{
   Vector<String> myNames;
   //add a few Strings to myNames here...

   Vector<String> tempNames;
   tempNames <<= myNames;
   PrintNames(tempNames);

   tempNames <<= myNames;
   PrintNames(tempNames);
}
```

This would work, but it has terrible overhead. It is not necessary to copy the vector at all, you can just to pass it to the PrintNames as a reference (in this case you can even use const reference):

```
// all you need to do is to add one little '&' ;-) the const is optional
void PrintNames(const Vector<String>& names)
{
   for(int i=0; i<names.GetCount(); i++)
   {
      //print the name
   }
}

int main()
{
   Vector<String> myNames;
   //add a few Strings to myNames here...

   PrintNames(myNames);
   PrintNames(myNames);
}
```

Since there is now no copying, there is no pick to worry about In general, you should use references as often as possible, it is a great boost to performance, especially for large containers.

Best regards,
Honza