Subject: ArrayCtrl cell color
Posted by hojtsy on Wed, 01 Feb 2006 15:29:26 GMT
View Forum Message <> Reply to Message

How do I set custom background or text color for an ArrayCtrl cell?

Subject: Re: ArrayCtrl cell color
Posted by mirek on Wed, 01 Feb 2006 16:59:54 GMT
View Forum Message <> Reply to Message

Using Display.

Mirek

Subject: Re: ArrayCtrl cell color
Posted by hojtsy on Thu, 02 Feb 2006 15:51:54 GMT
View Forum Message <> Reply to Message

I created a very simple Display class which displays the cell with red backround:

```
class RedDisplay : public Display
{
public:
  void Paint(Draw& w, const Rect& r, const Value& value,
             Color ink, Color paper, dword s) const
  {
    Display::Paint(w, r, value, ink, LtRed(), s);
  }
};
```

The problem is that the Paint method don't get the full cell rectangle, only a shrinked version. So every cell has a white border on the left and right side. Seems quite strange. Would it be possible to modify ArrayCtrl and the Display class so that Display gets the full cell rectangle?

File Attachments
1) upp_cell_color.png, downloaded 3373 times

Subject: Re: ArrayCtrl cell color
Posted by mirek on Thu, 02 Feb 2006 19:43:25 GMT
View Forum Message <> Reply to Message

ArrayCtrl::Column::Margin

- this sets the left/right margin of cell

HeaderCtrl::Column::SetMargin

- this sets the left/right margin of header cell margin and also arrayctrl margin if above Margin is -1 (which is default).

Means, to solve the trouble, use Margin(0) (but you will perhaps have to provide some margins in your Display).

Mirek

Subject: Re: ArrayCtrl cell color
Posted by hojtsy on Thu, 02 Feb 2006 20:46:43 GMT

View Forum Message <> Reply to Message

luzr wrote on Thu, 02 February 2006 14:43
Means, to solve the trouble, use Margin(0) (but you will perhaps have to provide some margins in your Display).Providing the margin inside the Display results in other problems: If the column contains cells with different Display-s (such as lots of StdDisplay cells) all of them should be subclassed to provide the margin. And there seems to be no easy way to make the margin size settable on a per-column basis if the margin is provided by the Display (any many other classes). It is inconvenient for the client code to create multiple subclasses (adding margin for every flavor of Display) just to modify the background color of a cell. I have some other alternative ideas for customizing the margin color withouth removing the margin, or definig several new classes.
Alternative 1: (new method)void Display::PaintMargin(Draw& w, const Rect& r, const Value& q, Color ink, Color paper, dword s) const

```
{
  w.DrawRect(r, paper);
}
```

Alternative 2: (new field in the Column and CellInfo):class Column : FormatConvert {

```
 ArrayCtrl           *arrayctrl;
 Mitor<int>           pos;
 const Convert        *convert;
 Ptr<Ctrl>            edit;
 const Display        *display;
 const Display        *marginDisplay;
        ....}
```

Obviously both of these alternatives need a small modification in the ArrayCtrl::Paint, to use the method for painting the margin area. And both alternatives have a really small performace hit. But the common benefit of them is that the client code could keep the settable margin in columns and mixed Display cells in the same column withouth subclassing every Display flavor. Would you consider adding one of these solutions to the library?

Subject: Re: ArrayCtrl cell color
Posted by mirek on Thu, 02 Feb 2006 21:10:55 GMT

View Forum Message <> Reply to Message

Note that if I undestand the issue well, alternative 1 does not solve the problem as you would have to subclass anyway.

I have these ideas:

* Provide "MarginDisplay" class that has as attributes another Display and the margin(s).

and/or

* Provide "DisplayWithMargin" template

or

* Provide ArrayCtrl::Column::MarginColor

I must say I like MarginDisplay (with possible template) as that does not involve complicating already complicated ArrayCtrl

Mirek

Subject: Re: ArrayCtrl cell color
Posted by hojtsy on Thu, 02 Feb 2006 21:46:59 GMT
View Forum Message <> Reply to Message

luzr wrote on Thu, 02 February 2006 16:10Note that if I undestand the issue well, alternative 1 does not solve the problem as you would have to subclass anyway.You would only need to subclass the Display of the cell you would like to recolor, which you subclass anyway because you redefining the Paint method. You would just redefine the PainMargin too in the same class, which would use the same background color or pattern. But you would no longer need to subclass all other Display classes that are used by other cells in the same Column.

Quote:Provide ArrayCtrl::Column::MarginColorThis disables backround patters or gradients and value-dependent background color.

Quote:Provide "MarginDisplay" class that has as attributes another Display and the margin(s).My alternative 2 is simillar but withouth the indirection step. In my idea the ArrayCtrl would use the marginDisplay to paint the margin area, and then the ArrayCtrl would use the Display to paint the content area. The type of the marginDisplay could be simply pointer to Display, and the default value would be a pointer to a MarginDisplay, which would just draw the rectangle in the Paint. I am afraid I was not suffiently clear, so I am thinking about implementing the idea to show what I mean.

Quote:Provide "DisplayWithMargin" templateThis would just help in adding the margin to a choosen Display, but then the Column margin should be removed, and every other Display subclass should be provided with a margin too. Not really convenient.

Subject: Re: ArrayCtrl cell color
Posted by mirek on Thu, 02 Feb 2006 22:11:33 GMT
View Forum Message <> Reply to Message

Yes, you are right about alternative 1... Sorry, I should have investigated it more.

Funny part about you marginDisplay idea is that regular standard "Display" would be good enough there (you would be passing Null for the Value).

Well, I think I will play a little with possible impacts on client code of all variants.

Hmmm.. Now thinking about it.... What about using the same Display for margin, just with "Null" for the Value? Maybe would not cover absolutely all cases, but you could easily solve the rest using "Margin(0)" option. At it would work pretty "automagically"... uhm, but maybe it is not a good idea (sometimes, you would really want to display Null as distinct value).

Mirek

Subject: Re: ArrayCtrl cell color
Posted by hojtsy on Thu, 02 Feb 2006 23:06:00 GMT
View Forum Message <> Reply to Message

luzr wrote on Thu, 02 February 2006 17:11Hmmm.. Now thinking about it.... What about using the same Display for margin, just with "Null" for the Value? Maybe would not cover absolutely all cases, but you could easily solve the rest using "Margin(0)" option. At it would work pretty "automagically"... uhm, but maybe it is not a good idea (sometimes, you would really want to display Null as distinct value).I think Null is not a good idea, for the reasons you told, and also for this other reason: I would like the background color to depend on the value. See my screenshot a few posts ago: in my application background color should be
* red if cell value is "FAILED"
* green is cell value is "passed"
* default otherwise
So passing Null does not work.
I compare my two alternatives: My alternative 1 (PaintMargin method) needs no additional subclass for the margin if you are recoloring the whole cell background, but needs many subclasses if you are just putting some fancy thing into the margin of many different cells. Alternative 2 needs an additional subclass for the margin defined by the client code, but it can apply that MarginDisplay class for any kind of cell withouth further subclassing them.

Subject: Re: ArrayCtrl cell color
Posted by mirek on Fri, 03 Feb 2006 13:29:06 GMT
View Forum Message <> Reply to Message

Well, I think we should use 1 - add method to Display.

Note however that it will likely paint just single-color background (I think it should be given single Rect that has to be painted as margin - means 2 calls of this method to paint 2 margins in ArrayCtrl column).

Maybe it should be used by standard Display::Paint (and maybe others) to erase background too...

Leaves last and most difficult tast... find a name for the method

Mirek

Subject: Re: ArrayCtrl cell color
Posted by hojtsy on Fri, 03 Feb 2006 14:28:15 GMT

View Forum Message <> Reply to Message

It could be PaintBackground. It is a good idea to call this method from the Display::Paint too, because then I only need to redefine the PaintBackground method and not the Paint for recoloring the background.
I was also thinking about how to completely remove the need for client-defined new classes just for the simple purpose of customizing the cell background/text color or font. The default Display could look something like this:

```
void CellDisplay::Paint(Draw& w, const Rect& r, const Value& q,
    Color ink, Color paper, dword s) const
{
 WString txt;
 Font font = StdFont();

 if(IsType<ArrayCell>(q))
 {
   ArrayCell ac(q);
   txt = ac.txt;
   if(!IsNull(ac.foreground)) ink   = ac.foreground;
   if(!IsNull(ac.background)) paper = ac.background;
   if(!IsNull(ac.font))       font  = ac.font;
 }
 else {
   txt = IsString(q) ? q : StdConvert().Format(q);
 }

        PaintBackground(w, r, q, ink, paper, s);
 int tcy = GetTLTextHeight(w, txt, StdFont());
 DrawTLText(w, r.left, r.top + max((r.Height() - tcy) / 2, 0), r.Width(), txt, font, ink);
}
```

It's not that complicated or slow in the library and would really simplify the client code when colors/fonts are needed. The client would just use the ArrayCell value in the needed cells and immediately be able to assign colors for each cell withoth any subclassing. I understand that client code could also define this CellDisplay and ArrayCell class, but it seems a common need in many applications, so it could just be in the library.

Subject: Re: ArrayCtrl cell color
Posted by mirek on Fri, 03 Feb 2006 15:38:56 GMT
View Forum Message <> Reply to Message

Well, not a bad idea either, but I think we should keep that as separate class. And not call it CellDisplay, because it could be pretty useful elsewhere too.

Well, anyway, let us start with PaintBackground.

Mirek

Subject: Re: ArrayCtrl cell color
Posted by hojtsy on Fri, 03 Feb 2006 15:48:54 GMT
View Forum Message <> Reply to Message

Yes put it in a separate class. I was suggesting that use that separate class as the default value of the Display in the ArrayCtrl::Column. It is true that it can be used elsewhere so it should not be called CellDisplay. Maybe the ArrayCell type (which I also used in the example) could also be renamed to something like ColoredText.

Subject: Re: ArrayCtrl cell color
Posted by mirek on Fri, 03 Feb 2006 16:56:48 GMT
View Forum Message <> Reply to Message

OK, PaintBackground now part of U++.

(give me some time to absorb the "CellDisplay" idea...

Mirek

Subject: Re: ArrayCtrl cell color
Posted by fudadmin on Mon, 06 Feb 2006 05:20:27 GMT
View Forum Message <> Reply to Message

luzr wrote on Fri, 03 February 2006 11:56OK, PaintBackground now part of U++.

Mirek

Can you give an example how to use it?

Subject: Re: ArrayCtrl cell color
Posted by hojtsy on Tue, 07 Feb 2006 12:43:39 GMT
View Forum Message <> Reply to Message

Here is a dummy example

```
#include <CtrlLib/CtrlLib.h>

class NumbersOnRed : public Display
{
public:
  static bool numberString(String s);
 void PaintBackground(Draw& w, const Rect& r, const Value& q,
                     Color ink, Color paper, dword style) const;

};

bool NumbersOnRed::numberString(String s)
{
  if(s.GetCount() == 0)
    return false;

  int i = 0;
  if(s[0] == '+' || s[0] == '-')
    i++;

  while(i < s.GetCount() && s[i] >= '0' && s[i] <= '9')
    i++;

  return i == s.GetCount();
}

void NumbersOnRed::PaintBackground(Draw& w, const Rect& r, const Value& q,
                               Color ink, Color paper, dword style) const
{
 if(IsNumber(q) || (IsString(q) && numberString(AsString(q))))
   paper = Color(255, 150, 150);

 Display::PaintBackground(w, r, q, ink, paper, style);
}


GUI_APP_MAIN
{
  ArrayCtrl array;
  array.AddColumn("value").SetDisplay(Single<NumbersOnRed>());
  array.Add("test1");
  array.Add("2test");
  array.Add("3 test");
  array.Add("4");
  array.Add("test");
  array.Add("-99");
  TopWindow win;
 win.Zoomable().Sizeable();
 win.Add(array.SizePos());
 win.Run();
}
```

Intention of PaintBackground is to provide a way for client code to customize the cell background color, including the margin background color, by redefining a method of Display. Previously the margin color was not customizable.

Subject: Re: ArrayCtrl cell color
Posted by mirek on Sun, 19 Feb 2006 22:48:32 GMT
View Forum Message <> Reply to Message

hojtsy wrote on Fri, 03 February 2006 09:28It could be PaintBackground. It is a good idea to call this method from the Display::Paint too, because then I only need to redefine the PaintBackground method and not the Paint for recoloring the background.
I was also thinking about how to completely remove the need for client-defined new classes just for the simple purpose of customizing the cell background/text color or font. The default Display could look something like this:void CellDisplay::Paint(Draw& w, const Rect& r, const Value& q,

```
     Color ink, Color paper, dword s) const
{
 WString txt;
 Font font = StdFont();

 if(IsType<ArrayCell>(q))
 {
   ArrayCell ac(q);
   txt = ac.txt;
   if(!IsNull(ac.foreground)) ink   = ac.foreground;
   if(!IsNull(ac.background)) paper = ac.background;
   if(!IsNull(ac.font))       font  = ac.font;
 }
 else {
   txt = IsString(q) ? q : StdConvert().Format(q);
 }

        PaintBackground(w, r, q, ink, paper, s);
 int tcy = GetTLTextHeight(w, txt, StdFont());
 DrawTLText(w, r.left, r.top + max((r.Height() - tcy) / 2, 0), r.Width(), txt, font, ink);
}
```

It's not that complicated or slow in the library and would really simplify the client code when colors/fonts are needed. The client would just use the ArrayCell value in the needed cells and immediately be able to assign colors for each cell withoth any subclassing. I understand that client code could also define this CellDisplay and ArrayCell class, but it seems a common need in many applications, so it could just be in the library.

Implemented as AttrText at StdDisplay level.

Mirek

Subject: Re: ArrayCtrl cell color
Posted by forlano on Tue, 27 Jun 2006 18:13:06 GMT

View Forum Message <> Reply to Message

hojtsy wrote on Tue, 07 February 2006 13:43Here is a dummy example

```
#include <CtrlLib/CtrlLib.h>

class NumbersOnRed : public Display
{
public:
  static bool numberString(String s);
 void PaintBackground(Draw& w, const Rect& r, const Value& q,
                      Color ink, Color paper, dword style) const;

};

bool NumbersOnRed::numberString(String s)
{
  if(s.GetCount() == 0)
    return false;

  int i = 0;
  if(s[0] == '+' || s[0] == '-')
    i++;

  while(i < s.GetCount() && s[i] >= '0' && s[i] <= '9')
    i++;

  return i == s.GetCount();
}

void NumbersOnRed::PaintBackground(Draw& w, const Rect& r, const Value& q,
                                   Color ink, Color paper, dword style) const
{
 if(IsNumber(q) || (IsString(q) && numberString(AsString(q))))
   paper = Color(255, 150, 150);

 Display::PaintBackground(w, r, q, ink, paper, style);
}
```

```
GUI_APP_MAIN
{
  ArrayCtrl array;
  array.AddColumn("value").SetDisplay(Single<NumbersOnRed>());
  array.Add("test1");
  array.Add("2test");
  array.Add("3 test");
  array.Add("4");
  array.Add("test");
  array.Add("-99");
  TopWindow win;
 win.Zoomable().Sizeable();
 win.Add(array.SizePos());
 win.Run();
}
```

Intention of PaintBackground is to provide a way for client code to customize the cell background color, including the margin background color, by redefining a method of Display. Previously the margin color was not customizable.

Today I used this example to color some cells of my ArrayCtrl. It worked very nice OK and there were no problems... except one detail: why it does work?
I know that is not important to understand to let the things to work... in contrast it is better very often do not know. But I'm very curious. Please forgive me for my silly questions but it is stronger than me to ask:

1) This line is very important:

```
  array.AddColumn("value").SetDisplay(Single<NumbersOnRed>());
```

Now, SetDisplay is the method with arguments:
SetDisplay(int i&cedil; int j&cedil; const Display& d)
Instead I see "Single<NumbersOnRed>()" that does the job... but HOW? What is it?

2) The last stupid question. Is "q" the cell value? Where it has been defined?

Thank you for your patience,
Luigi

PS: if nobody answer I'll not complain

Subject: Re: ArrayCtrl cell color
Posted by hojtsy on Tue, 27 Jun 2006 20:06:41 GMT

View Forum Message <> Reply to Message

forlano wrote on Tue, 27 June 2006 14:13
1) This line is very important:

array.AddColumn("value").SetDisplay(Single<NumbersOnRed>());

Now, SetDisplay is the method with arguments:
SetDisplay(int i&cedil; int j&cedil; const Display& d)
Instead I see "Single<NumbersOnRed>()" that does the job... but HOW? What is it?

A different SetDisplay is called there. This one:

ArrayCtrl::Column& ArrayCtrl::Column::SetDisplay(const Display& d);

The reason for that is that AddColumn does not return a reference to the array, but to the newly added column. You are setting the Display object for the newly added column, and not the whole array.
So this trick is not done by the Single template method. What it does instead is that it returns a reference to an internally stored "singleton" instance of a given type. Search for "singleton pattern" on google for more info about singletons.

forlano wrote on Tue, 27 June 2006 14:13
2) The last stupid question. Is "q" the cell value? Where it has been defined?
The "q" is the cell value, it is defined as a function parameter to the PaintBackground, and Paint methods of Display. The name of the parameter is freely choosen, I put q, because Paint method had it originally this way. The type of q is Value, which is the magical joker type in U++, able to store values of practically any other type. This way the fields of the ArrayCtrl could be Strings, ints, doubles, or even instances of some complex class type.

And: there are no stupid questions.

Subject: Re: ArrayCtrl cell color
Posted by forlano on Tue, 27 Jun 2006 22:49:38 GMT
View Forum Message <> Reply to Message

hojtsy wrote on Tue, 27 June 2006 22:06forlano wrote on Tue, 27 June 2006 14:13
1) This line is very important:

array.AddColumn("value").SetDisplay(Single<NumbersOnRed>());

Now, SetDisplay is the method with arguments:
SetDisplay(int i&cedil; int j&cedil; const Display& d)
Instead I see "Single<NumbersOnRed>()" that does the job... but HOW? What is it?

A different SetDisplay is called there. This one:

ArrayCtrl::Column& ArrayCtrl::Column::SetDisplay(const Display& d);

The reason for that is that AddColumn does not return a reference to the array, but to the newly added column. You are setting the Display object for the newly added column, and not the whole array.

So this trick is not done by the Single template method. What it does instead is that it returns a reference to an internally stored "singleton" instance of a given type. Search for "singleton pattern" on google for more info about singletons.

Thank you for your answer.
I googled "Singleton pattern" and at the end I decided that it is better to use the code without to understand . The topic seems for advanced C++ users while I've started a few monthes ago.
I am a simple user that need widgets that can dialog among them easily and that their properties can be set the same easily. After few monthes U++ thought me to think in the easier, lazier and non verbose way to reach the goal. For this reason sometimes I feel unconfortable and a bit angry, , when some operation need to be done in a too elegant, powerfull and smart way but that I do not understand.
In our case, for example, I would instinctively think about some method like array.SetBgColor(int i, int j) or array.SetPaper(int i, int j) to set the background color of a cell.
Of course I do not know all the environment and my point of view is simply faulty. Nevertheless I am sure that such a direct methods one day not to far will appear .

Luigi

Subject: Re: ArrayCtrl cell color
Posted by fudadmin on Wed, 28 Jun 2006 12:06:49 GMT
View Forum Message <> Reply to Message

forlano wrote on Tue, 27 June 2006 23:49
I googled "Singleton pattern" and at the end I decided that it is better to use the code without to understand . The topic seems for advanced C++ users while I've started a few monthes ago.
I am a simple user that need widgets that can dialog among them easily and that their properties can be set the same easily. After few monthes U++ thought me to think in the easier, lazier and non verbose way to reach the goal. For this reason sometimes I feel unconfortable and a bit angry, , when some operation need to be done in a too elegant, powerfull and smart way but that I do not understand.
In our case, for example, I would instinctively think about some method like array.SetBgColor(int i, int j) or array.SetPaper(int i, int j) to set the background color of a cell.
Of course I do not know all the environment and my point of view is simply faulty. Nevertheless I am sure that such a direct methods one day not to far will appear .

Luigi

array.SetBgColor(int i, int j) ??? OR:
array.AddColumn("value").SetDisplay(Single<NumbersOnRed>());
I see that as an analogue between plain HTML and CSS.
While I haven't studied to much the Display (or sigleton patterns at all) and my C++ experience is not too much advanced but I think the idea of Display is super great because of flexibility and huge memory savings esp. for big ArrayCtrls? And you can "attach" one property to range of cells.

Each cell (or her "master" -column), instead of keeping and reading properties as members, goes

(by following pointers?) to a memory place where sharable property is kept. You could use only SetDisplay(NumbersOnRed) if NumbersOnRed meets the requirements.
But, as I understand, Single ensures (by adding some extra "shape" to that memory place) that there is only one (single) such memory place created when needed.
Is my understanding correct?

Subject: Re: ArrayCtrl cell color
Posted by mirek on Wed, 28 Jun 2006 12:13:04 GMT
View Forum Message <> Reply to Message

BTW, ignoring some subtle MT issues for a moment,

```
Single<Foo>()
```

is equivalent to

```
Foo& SingleFoo()
{
   static Foo foo;
   return foo;
}
```

Two things to consider:

- it is one shared instance (object) for all uses

- placing static inside function ensures lazy initialization, in other words, Foo constructor gets called during the first call to SingleFoo.

Mirek

Subject: Re: ArrayCtrl cell color
Posted by forlano on Wed, 28 Jun 2006 14:26:01 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 28 June 2006 14:13BTW, ignoring some subtle MT issues for a moment,

```
Single<Foo>()
```

is equivalent to

```
Foo& SingleFoo()
{
   static Foo foo;
   return foo;
}
```

Two things to consider:

- it is one shared instance (object) for all uses

- placing static inside function ensures lazy initialization, in other words, Foo constructor gets called during the first call to SingleFoo.

Mirek
Thank you for all your explanations.
I do not doubt it is a great feature, but I need some time to digest it.
BTW, can it be used to set the font of, say, two columns of the ArrayCtrl?

Luigi

Subject: Re: ArrayCtrl cell color
Posted by fudadmin on Wed, 28 Jun 2006 14:58:49 GMT

View Forum Message <> Reply to Message

forlano wrote on Wed, 28 June 2006 15:26
BTW, can it be used to set the font of, say, two columns of the ArrayCtrl?

Luigi

Please, start a new topic...