Subject: Should RGBA have got 4 arguments constructor?
Posted by Klugier on Sat, 03 May 2014 18:59:07 GMT
View Forum Message <> Reply to Message

Hello,

It seems that RGBA structure hasn't got 4 arguments constructor. So, following code dosen't work:

RGBA rgba(255, 255, 0, 0); // <- alpha, red, green, blue;

BTW, Using IDE "insert color dialog" we can select and create RGBA in above form.

Alternatively we can write:

RGBA rgba = Color(255, 0, 0);
rgba.a = 255;

Sincerely,
Klugier

Subject: Re: Should RGBA have got 4 arguments constructor?
Posted by unodgs on Sat, 03 May 2014 19:44:54 GMT
View Forum Message <> Reply to Message

I think it should have such a constructor. It also reminds me old discussion about Color(r, g, b a) ;)

Subject: Re: Should RGBA have got 4 arguments constructor?
Posted by mirek on Thu, 29 May 2014 09:18:30 GMT
View Forum Message <> Reply to Message

According to C++03 rules, adding constructor would make RGBA non-POD, which has implications on what operations are legal with it.

Furthermore, the issue is tricky, because the standard format is premultiplied, so e.g.

RGBA(0, 255, 0, 128)

would be wrong if constructor would just assign numbers to respective fields.

RGBA rgba = Color(255, 0, 0);

rgba.a = 255;

No need to set .a here, it is automatically set to 255.

Note that there is also operator* to add alpha to Color:

RGBA x = 128 * Red(); // Sets Red with alpha = 128

Subject: Re: Should RGBA have got 4 arguments constructor?
Posted by mirek on Thu, 29 May 2014 09:32:45 GMT
View Forum Message <> Reply to Message

I have changed Insert color in the ide to avoid confusion...

Subject: Re: Should RGBA have got 4 arguments constructor?
Posted by Klugier on Thu, 24 Jul 2014 20:32:08 GMT
View Forum Message <> Reply to Message

Hello Mirek,

We can easily implement RGBA constructor in C++11 branch... This standard changes a little bit POD rules. Now we can define constructor, but firstly we need to tell compiler that default constructor exists. I enclose demonstrative code:

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

#include <type_traits>

using namespace Upp;

namespace UppCpp11 {
struct RGBA {
 RGBA() = default;
 RGBA(byte r, byte g, byte b, byte a) {
  this->r = r;
  this->g = g;
  this->b = b;
  this->a = a;
 }

 byte b, g, r, a;
};
```

```
struct NotPodRGBA {
 NotPodRGBA(byte r, byte g, byte b, byte a) {
  this->r = r;
  this->g = g;
  this->b = b;
  this->a = a;
 }

 byte b, g, r, a;
};
}

CONSOLE_APP_MAIN
{
 UppCpp11::RGBA color(255, 255, 255, 255);
 Cout() << "Is RGBA POD? " << (std::is_pod<UppCpp11::RGBA>::value ? "True" : "False") <<
"!\n";
 Cout() << "Is NotPodRGBA POD? " << (std::is_pod<UppCpp11::NotPodRGBA>::value ? "True" :
"False") << "!\n";
}
```

Result:

```
Is RGBA POD? True!
Is NotPodRGBA POD? False!
```

More information you can find on:
http://stackoverflow.com/questions/4178175/what-are-aggregat
es-and-pods-and-how-why-are-they-special/7189821#7189821 - second answer.

P.S.
Tested with g++ 4.8 using std=c++11 flag.

Sincerely,
Klugier

Subject: Re: Should RGBA have got 4 arguments constructor?
Posted by mirek on Fri, 25 Jul 2014 05:46:39 GMT
View Forum Message <> Reply to Message

Klugier wrote on Thu, 24 July 2014 22:32Hello Mirek,

We can easily implement RGBA constructor in C++11 branch... This standard changes a little bit

POD rules. Now we can define constructor, but firstly we need to tell compiler that default constructor exists. I enclose demonstrative code:

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

#include <type_traits>

using namespace Upp;

namespace UppCpp11 {
struct RGBA {
 RGBA() = default;
 RGBA(byte r, byte g, byte b, byte a) {
  this->r = r;
  this->g = g;
  this->b = b;
  this->a = a;
 }

 byte b, g, r, a;
};

struct NotPodRGBA {
 NotPodRGBA(byte r, byte g, byte b, byte a) {
  this->r = r;
  this->g = g;
  this->b = b;
  this->a = a;
 }

 byte b, g, r, a;
};
}

CONSOLE_APP_MAIN
{
 UppCpp11::RGBA color(255, 255, 255, 255);
 Cout() << "Is RGBA POD? " << (std::is_pod<UppCpp11::RGBA>::value ? "True" : "False") <<
"!\n";
 Cout() << "Is NotPodRGBA POD? " << (std::is_pod<UppCpp11::NotPodRGBA>::value ? "True" :
"False") << "!\n";
}
```

Result:

Is RGBA POD? True!

Is NotPodRGBA POD? False!

More information you can find on:
http://stackoverflow.com/questions/4178175/what-are-aggregates-and-pods-and-how-why-are-they-special/7189821#7189821 - second answer.

P.S.
Tested with g++ 4.8 using std=c++11 flag.

Sincerely,
Klugier

OK, but premultiplied alpha issue is still there... Should we add ASSERT

```
 RGBA(byte r, byte g, byte b, byte a) {
  ASSERT(r <= a && g <= a && b <= a);
  this->r = r;
  this->g = g;
  this->b = b;
  this->a = a;
 }
```

or perform conversion?

Subject: Re: Should RGBA have got 4 arguments constructor?
Posted by Klugier on Fri, 25 Jul 2014 18:26:04 GMT
View Forum Message <> Reply to Message

Hello Mirek,

Quote:
OK, but premultiplied alpha issue is still there... Should we add ASSERT

```
RGBA(byte r, byte g, byte b, byte a) {
    ASSERT(r <= a && g <= a && b <= a);
    this->r = r;
    this->g = g;
    this->b = b;
    this->a = a;
}
```

or perform conversion?

I don't know anything about "premultiplied alpha issue", but personally I think that implicit conversion would be safer for U++ users.

Moreover, I would like to noticed that it will be fine if we will have unit tests for RGBA. Each time the test failed we will know that something isn't ok in RGBA.

Sincerely,
Klugier

Subject: Re: Should RGBA have got 4 arguments constructor?
Posted by mirek on Mon, 04 Aug 2014 12:21:38 GMT
View Forum Message <> Reply to Message

Quote:
I don't know anything about "premultiplied alpha issue", but personally I think that implicit conversion would be safer for U++ users.

Google "premultiplied alpha". I guess that alone is a reason not to rush to have constructor....

Quote:
Moreover, I would like to noticed that it will be fine if we will have unit tests for RGBA. Each time the test failed we will know that something isn't ok in RGBA.

I am not quite sure what you want to unittest about RGBA. But surely, it would be nice addition to have some RGBA *related* unittests, there is a couple of functions in Draw and in fact all Image processing is RGBA...

Mirek

Subject: Re: Should RGBA have got 4 arguments constructor?
Posted by Klugier on Tue, 05 Aug 2014 19:25:26 GMT
View Forum Message <> Reply to Message

Hello Mirek,

Quote:
Google "premultiplied alpha". I guess that alone is a reason not to rush to have constructor....

If you are sure about not to adding constructor to RGBA class. I will not protest, but personally I

think that RGBA constructor will be nice feature for common Upp user. I would like to notice that maybe simply conversion in constructor do the trick (This is your original proposition).

Quote:
I am not quite sure what you want to unittest about RGBA. But surely, it would be nice addition to have some RGBA *related* unittests, there is a couple of functions in Draw and in fact all Image processing is RGBA...

I mean if "premultiplied alpha" is Upp related issue it would be nice to have unittest for this. It will always remind us that something is wrong in this part of code. But if it isn't...

P.S.
I think that unit tests is good idea, but it drags additional cost in the form of time (writing tests) and tools that will execute this test periodically.

Sincerely,
Klugier

Subject: Re: Should RGBA have got 4 arguments constructor?
Posted by mirek on Wed, 06 Aug 2014 15:56:06 GMT
View Forum Message <> Reply to Message

Klugier wrote on Tue, 05 August 2014 21:25
I mean if "premultiplied alpha" is Upp related issue it would be nice to have unittest for this. It will always remind us that something is wrong in this part of code. But if it isn't...

Nope, premultiplied alpha is "computational reality related" issue. Have you googled it? :)

Mirek

Subject: Re: Should RGBA have got 4 arguments constructor?
Posted by Klugier on Sat, 09 Aug 2014 17:35:14 GMT
View Forum Message <> Reply to Message

Hello Mirek,

Quote:
Have you googled it? :)

Yep, I think I understand the basic idea of "premultiplied alpha". :) It is just better RGBA

representation.

Sincerely,
Klugier