COLOR INTERFACE

View online: https://www.construct.net/en/make-games/manuals/addon-sdk/reference/geometry-interfaces/color

The `Color` interface represents a floating-point RGBA color in the SDK. It can also be constructed independently as a general-purpose color class. Each color component is normalized to the range [0, 1].

In the WebGL renderer, colors are normally required to have premultiplied alpha. Some APIs already return premultiplied colors, but others may not; check the documentation for any API methods returning colors to find out which are used. Wherever possible avoid using the `unpremultiply()` method, since it is lossy.

Constructor

```
new SDK.Color();
new SDK.Color(r, g, b, a);
```

A `Color` can be constructed with no parameters, which defaults all components to zero, or with given RGBA components.

Methods

setRgb(r, g, b)

Set the RGB components only, without affecting the alpha component, in a single call.

setRgba(r, g, b, a)

Set the RGBA components of the color in a single call.

copy(color)

Set the components of the color by copying another `SDK.Color`.

copyRgb(color)

Set the RGB components only, without affecting the alpha component, by copying another `SDK.Color`.

clone()

Return a new instance of an `SDK.Color` with an identical color to this one.

---

setR(r)

setG(g)

setB(b)

setA(a)

Set each component of the color individually. Note color components are floats in the range [0, 1].

---

getR()

getG()

getB()

getA()

Get each component of the color individually.

---

equals(color)

Return a boolean indicating if this color exactly matches another `SDK.Color`.

---

equalsIgnoringAlpha(color)

Return a boolean indicating if this color exactly matches the RGB components of another `SDK.Color`. The alpha component is ignored.

---

equalsRgb(r, g, b)

Return a boolean indicating if this color exactly matches the given RGB components.

---

equalsRgba(r, g, b, a)

Return a boolean indicating if this color exactly matches the given RGBA components.

---

premultiply()

Multiply the RGB components by the A component. This is usually required for rendering.

---

unpremultiply()

Divide the RGB components by the A component.

> *Avoid this method whenever possible, because it is lossy. (Unpremultiplying a premultiplied color will lose some precision in the RGB components and may not exactly match the original color.)*