

# Painter API

## About

Painter API allows you to draw things on the screen, both from server and directly from client. This can allow you to create widgets from server side or effects on screen or in world from client side.

Currently it doesn't support any input, but in future, in-game menus or even GUIs similar to Source engine ones will be supported.

Paintable objects are created from NBT/Json objects and all have an id. If id isn't provided, a random one will be generated. Objects x and z are absolute positions based on screen, but you can align elements in one of the corners of screen. You can bulk add multiple objects in one json object. All properties are optional, but obviously some you should almost always override like size and position for rectangles.

`paint({...})` is based on upsert principle - if object doesn't exist it will create it (if the object also contains valid `type`), otherwise, update existing:

- `event.player.paint({example: {type: 'rectangle', x: 10, y: 10, w: 20, h: 20}})` - New rectangle is created
- `event.player.paint({example: {x: 50}})` - Updates previous rectangle with partial data

You can bulk update/create multiple things in same object:

- `event.player.paint({a: {x: 10}, b: {x: 30}, c: {type: 'rectangle', x: 10}})`

You can remove object with `remove: true`, bulk remove multiple objects or remove all objects:

- `event.player.paint({a: {remove: true}})`
- `event.player.paint({a: {remove: true}, b: {remove: true}})`
- `event.player.paint({'*': {remove: true}})`

These methods have command alternatives:

- `/kubejs painter @p {example: {type: 'rectangle', x: 10, y: 10, w: 20, h: 20}}`

If the object is re-occurring, it's recommended to create objects at login with all of its static properties and `visible: false`, then update it later to unhide it. Painter objects will be cleared when players leave world/server, if its persistent, then it must be re-added at login every time.

## Currently available objects

Underlined objects are not something you can create

## Root

(available for all objects)

- Boolean visible
- Float x
- Float y
- Float z
- Float w
- Float h
- Enum alignX (one of 'left', 'center', 'right')
- Enum alignY (one of 'top', 'center', 'bottom')
- Enum draw (one of 'ingame', 'gui', 'always')
- Float moveX
- Float moveY
- Float expandW
- Float expandH

## rectangle

- Color color
- String texture
- Float u0
- Float v0
- Float u1
- Float v1

## gradient

- Color color
- Color colorT
- Color colorB
- Color colorL
- Color colorR
- Color colorTL
- Color colorTR
- Color colorBL
- Color colorBR
- String texture
- Float u0
- Float v0
- Float u1
- Float v1

## text

- Text text | Text[] textLines
- Boolean shadow
- Float scale
- Color color
- Boolean centered
- Float lineSpacing

## item

- ItemStack item (supports either 'itemid' or vanilla {id: 'item', Count: 4, tag: {...}} NBT syntax)
- Boolean overlay
- String customText
- Float rotation

## Properties

- Unit is a [Rhino Unit](#). It can be a number, boolean, color, equation. Every Float, Int, Boolean and Color are also Units, so you can use equations on them.
- Int is a int32 number, any whole value, e.g. `40`.
- Float is float64 floating point number, e.g. `2.35`.
- String is a string, e.g. `'example'`. Textures usually need resource location `'namespace:path/to/texture.png'`.
- Color can be either `0xRRGGBB`, `'#RRGGBB'`, `'#AARRGGBB'`, e.g. `'#58AD5B'` or chat colors `'red'`, `'dark_aqua'`, etc. RGBA `color(Float, Float, Float, Float)` is also supported where Float is any number between 0.0 and 1.0 (supports Units).
- Text can be a string `'Example'` or `Text.of('Red and italic string example').red().italic()` etc formatted string.

## Available Unit variables

- `$screenW` - Screen width
- `$screenH` - Screen height
- `$delta` - Render delta
- `$mouseX` - Mouse X position
- `$mouseY` - Mouse Y position

## Available Unit constants

- `true` - boolean true value, equal to 1.0
- `false` - boolean false value, equal to 0.0
- `PI` - number equal to 3.14159265358979323846
- `HALF_PI` - number equal to 1.57079632679
- `TWO_PI` - number equal to 6.28318530718

- E - number equal to 2.7182818284590452354

## Examples

```
onEvent('player.logged_in', event => {
  event.player.paint({
    example_rectangle: {
      type: 'rectangle',
      x: 10,
      y: 10,
      w: 50,
      h: 20,
      color: '#00FF00',
      draw: 'always'
    },
    last_message: {
      type: 'text',
      text: 'No last message',
      scale: 1.5,
      x: -4,
      y: -4,
      alignX: 'right',
      alignY: 'bottom',
      draw: 'always'
    }
  })
})
```

```
onEvent('player.chat', event => {
  // Updates example_rectangle x value and last_message text value to last message + contents from event
  event.player.paint({example_rectangle: {x: '(sin((time() * 1.1)) * (($screenW - 32) / 2))', w: 32, h: 32, alignX:
'center', texture: 'kubejs:textures/item/diamond_ore.png'}}})
  event.player.paint({last_message: {text: `Last message: ${event.message}`}}})
  // Bulk update, this is the same code as 2 lines above, you can use whichever you like better
  // event.player.paint({example_rectangle: {x: 120}, last_message: {text: `Last message:
${event.message}`}}})
  event.player.paint({lava: {type: 'atlas_texture', texture: 'minecraft:block/lava_flow'}}})
})
```



Image not found or type unknown

---

Revision #37

Created 12 August 2021 09:53:18 by Lat

Updated 14 October 2022 14:22:41 by Lat