Basics

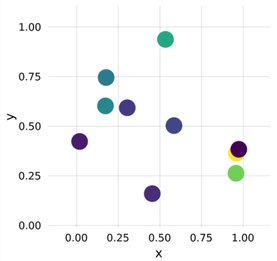
Set the Scene

The Scene object holds everything in a plot
Initializing: scene = Scene()

Basic plotting

You can put your mouse in the plot window and scroll to zoom. Right click and drag lets you pan around the scene, and left click and drag lets you do selection zoom (in 2D plots), or orbit around the scene (in 3D plots).

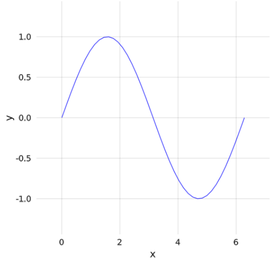
It is worth noting initally that if you run a Makie.jl example and nothing shows up, you likely need to do display(scene) to render the example on screen.



Scatter plot

```
using Makie

x = rand(10)
y = rand(10)
colors = rand(10)
scene = scatter
(x, y, color = colors)
```



Line plot

```
using Makie

x = range(0, stop = 2pi,
length = 40)
f(x) = sin.(x)
y = f(x)
scene = lines
(x, y, color = :blue)
```

Animation

Makie.jl saves to: .mkv, .mp4, .webm, .gif

All you need to do is wrap your changes in the record function.

```
using Makie
scene = lines(rand(10); linewidth=10)
record(scene, "line_changing_colour.mp4", 1:255;
framerate = 60) do i
  scene.plots[2][:color] = RGBf0(i/255, (255 - i)/255, 0)
  # animate scene
end
```

Using time

time = Node(0.0)
lift is using to set up a pipeline to access its value.
Whenever the Node time is updated (e.g. when you push! to it), the plot will also be updated.

```
push!(time, Base.time())
```

Appending data to a plot

If you're planning to append to a plot, like a lines or scatter plot, you will want to pass an Observable Array of Points to the plotting function, instead of passing x, y as separate Arrays. This will mean that you won't run into dimension mismatch issues.

Animating in a loop

for loop:

```
for i = 1:length(r)
   s[:markersize] = r[i]
   sleep(1/24)
end
```

You don't need to use AbstractPlotting.force_update!() in a loop

If you want to animate a plot while interacting, use async_latest

Interaction

Node interaction pipeline

A Node is a Julia structure that allows its value to be updated interactively.

```
x = Node(0.0)
```

The value of the x can be changed simply using push!

to_value to get the value of a Node

Nodes depending on other Nodes

You can create a node depending on another node using lift:

```
f(a) = a^2
y = lift(a -> f(a), x)
```

Updating the value of x will also update the value of y!

Event triggering

Often it is the case that you want an event to be triggered each time a Node has its value updated. This is done using the on-do block from Observables.

```
on(x) do val
   println("x just got the value $val")
end
push!(x, 5.0);
```

Functions

text

text(string)

Plots a text.

meshscatter

meshscatter(positions)
meshscatter(x, y)
meshscatter(x, y, z)

Plots a mesh for each element (x, y, z), (x, y), or positions. markersize is a scaling applied to the primitive passed as marker.

scatter

scatter(positions)
scatter(x, y)
scatter(x, y, z)

Plots a marker for each element in (x, y, z), (x, y), or positions.

mesh

mesh(x, y, z)
mesh(mesh_object)
mesh(x, y, z, faces)
mesh(xyz, faces)

Plots a 3D mesh.

lines

mesh(x, y, z)
mesh(mesh_object)
mesh(x, y, z, faces)
mesh(xyz, faces)

Creates a connected line plot for each element in (x, y, z), (x, y) or positions.

volume

volume(volume_data)

Plots a volume.

image

image(x, y, image)
image(image)

Plots an image on range x, y

contour

contour(x, y, z)

Creates a contour plot of the plane spanning x::Vector, y::Vector, z::Matrix

surface

surface(x, y, z)

Plots a surface, where (x, y) define a grid whose heights are the entries in z.

Check Makie.jl docs for more informations