Solandra Cheat Sheet

For over 100 examples with source code, many tutorials and no-install starter projects see https://solandra.netlify.com.

Install

On NPM. Install with `npm i solandra` or `yarn add solandra`. There is a React wrapper for those using the most popular front end framework, install from NPM with `npm i solandra-react solandra react react-dom`or `yarn add solandra-react solandra react react-dom`.

Basics

A Solandra sketch is a function on `SCanvas` and we will just use `s` for this here.

```
const sketch = (s: SCanvas) => {
  // write code here
}
```

Key Ideas

- It uses TypeScript, so no need to memorise things, type `s.` then try to autocomplete (e.g. `Ctrl + Space`).
- The Solandra Canvas is always of width 1. The height depends on the aspect ratio. To get size and other metadata use `s.meta`.
- Colours always use `hsl(a)`.

Colours

- Fill background with `s.background(h,s,l,a?)`
- Set fill colour with `s.setFillColor(h,s,l,a?)`
- Set stroke colour with `s.setStrokeColor(h,s,l,a?)`
- Set fill gradient with `s.setFillGradient`
- Set stroke gradient with `s.setStrokeGradient`

There are some advanced things you can set like `lineWidth`, shadows and `lineStyle`.

Drawing

Will use your current colour state as set above.

- Draw (outline) a path with `s.draw`
- Fill a path with `s.draw`

Paths

Solandra ships with lots of ready to go shapes to draw. They are typically classes, created with a single object literal configuration, much of which is optional. You will typically `draw` or `fill` them. For example

```
s.fill(new Rect({ at: [0, 0], w: 0.1, h: 0.1 }))
```

Solandra uses concise configuration property names where they seem obvious (e.g. `w` for width). Other paths include (see TypeScript/autocomplete for configurations):

- `Arc`
- `Circle`
- `Ellipse`
- `Hatching`
- `HollowArc`
- `Line`
- `Rect`
- `Regular Polygon`
- `RoundedRect`
- `Square`
- `Star`

For custom paths there is:

- `Path` for simple straight lines between points
- `SimplePath` allows for curves between points

Many standard shapes can be converted to a primitive `Path` via `.path`. There are then operations which you can perform such as `chaiken` which smooths a path. See the examples.

Control Flow

You can use normal Javascript control flow. But Solandra adds some of its own APIs. Most take a configuration and callback arguments.

- `times` do something n times
- `downFrom` do something n times, but count goes down
- `range` cover a range in a number of steps
- `forGrid` cover a 2D grid
- `doProportion` do something a proportion of times

- `proportionately` supply a list of proportions and functions to call

~~Control~~ Canvas Flow

You can also have control flow over the canvas. Most take a rather fancy callback which it is often convenient to destructure:

```
s.forTiling({ n: 10 }, (point, delta, center, i) => {})
```

```
s.forTiling({ n: 10 }, ([x, y], [dX, dY], [cX, cY], i) =>
{})
```

- `forMargin`
- `forTiling`
- `forHorizontal`
- `forVertical`
- `aroundCircle`

You can get fancier these, which takes one of the above as an argument

- `build` which returns the result of the callbacks in an array
- `withRandomOrder` does things in a random order

Canvas operations

Each of these takes a configuration and callback. The configuration alters how things are drawn. These can be stacked.

- `withClipping` this clips all drawing in its callback to a path
- `withRotation` all drawing in the callback is with a rotation
- `withScale` draw with a scale
- `withTranslation` draw with a translation (move)
- `withTransform` fully custom

Randomness

Call to get pseudorandom values. Most have sensible default configurations

- `random` between 0 and 1
- `uniformRandomInt`
- `uniformGridPoint` a 2D grid point
- `randomPolarity` -1 or 1
- `sample` give it an array, it picks one each time
- `perturb` take a point and move
- `gaussian`
- `poisson`

Time

`s.t` gives you the current time. For sketches that are 'playing'.

Fancy

Solandra ships with even more functionality, for example hexagon grids, isometric grids and noise. See the examples.