Glyphs

In the context of data visualization, a glyph is any marker, such as an arrow or similar marking, used to specify part of a scientific visualization. This is a representation to visualize data where the data set is presented as a collection of visual objects. These visual objects are collectively called a Glyph.

Glyphs are the basic visual building blocks of Bokeh plots, e.g. lines, rectangles, squares, wedges, patches, etc. The `bokeh.plotting` interface provides a convenient way to create plots centered around glyphs.

- To style the fill, line, or text properties of a glyph, it is first necessary to obtain a specific `GlyphRenderer`.

- When using the `bokeh.plotting` interface, the glyph functions return the renderer:

```
>>> r = p.circle([1,2,3,4,5], [2,5,8,2,7])
>>> r
<bokeh.models.renderers.GlyphRenderer at 0x106a4c810>
```

Then, the glyph itself is obtained from the `.glyph` attribute of a `GlyphRenderer`:
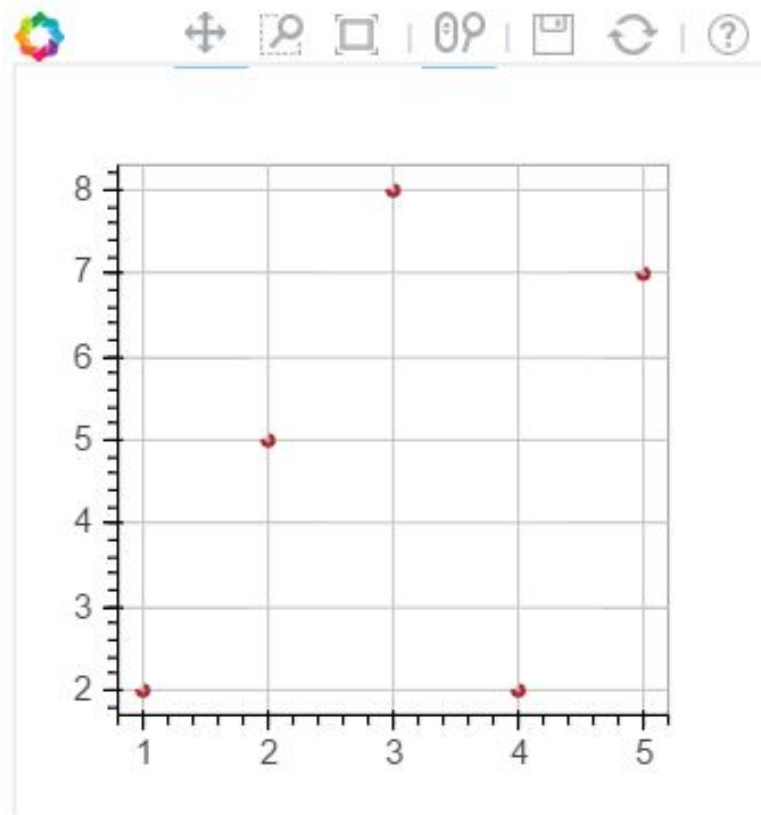
```
>>> r.glyph
<bokeh.models.markers.Circle at 0x10799ba10>
```

This is the object to set fill, line, or text property values for:

```
from bokeh.plotting import figure, output_file, show

output_file("axes.html")

p = figure(plot_width=400, plot_height=400)
r = p.circle([1,2,3,4,5], [2,5,8,2,7])
show(p)
```
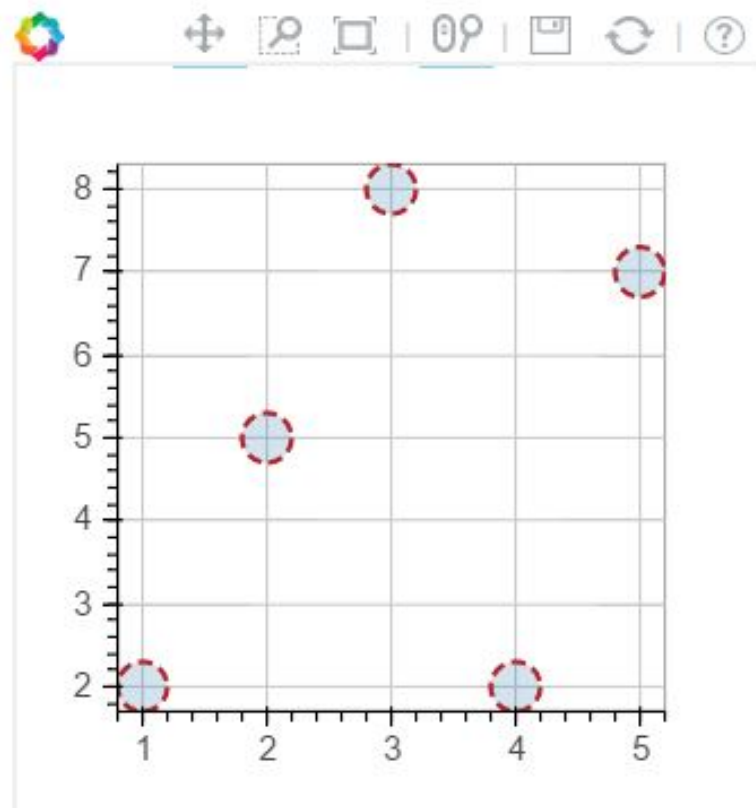
```
glyph = r.glyph
glyph.size = 60
glyph.fill_alpha = 0.2
glyph.line_color = "firebrick"
glyph.line_dash = [6, 3]
glyph.line_width = 2

show(p)
```

Selected & Unselected Glyphs

The styling of selected and non-selected glyphs can be customized by setting the selection_glyph and/or nonselection_glyph attributes of the GlyphRenderer either manually or by passing them to add_glyph().

```python
from bokeh.io import output_file, show
from bokeh.plotting import figure
from bokeh.models import Circle

output_file("styling_selections.html")

p = figure(plot_width=400, plot_height=400,
        tools="tap", title="Select a circle")

p.circle([1, 2, 3, 4, 5], [2, 5, 8, 2, 7],
        size=50, name="mycircle")

selected_circle = Circle(fill_alpha=1,
        fill_color="firebrick",
        line_color=None)

nonselected_circle = Circle(fill_alpha=0.2,
        fill_color="blue",
        line_color="firebrick")

renderer = p.select(name="mycircle")
renderer.selection_glyph = selected_circle
renderer.nonselection_glyph = nonselected_circle

show(p)
```

- Click/Tap to select circles on the plot above to see the effect on the nonselected glyphs.

- Click in the plot, but not on a circle, to see their original state

(this is set by the original call `p.circle()`).

The same could be achieved with the models interface as follows:

```
p = Plot()
source = ColumnDataSource(dict(x=[1, 2, 3], y=[1, 2, 3]))

initial_circle = Circle(x='x', y='y',
fill_color='blue', size=50)

selected_circle = Circle(fill_alpha=1,
fill_color="firebrick", line_color=None)

nonselected_circle = Circle(fill_alpha=0.2,
fill_color="blue", line_color="firebrick")

p.add_glyph(
  source,
  initial_circle,
  selection_glyph=selected_circle,
  nonselection_glyph=nonselected_circle
 )
```

Notes

- Only the visual properties of `selection_glyph` and `nonselection_glyph` are considered when rendering.

- Changing positions, sizes, etc. will have no effect.