a *few* IDEAS for elaborating your DrScheme scenes
(NOT a complete list! just to get you started...)

* it is helpful to define `WIDTH` and `HEIGHT` for your scenes, so you can change their size more conveniently later if you want/need to;

```
(define WIDTH 300)   ; or whatever value you want for width
(define HEIGHT 200)  ; or whatever value you want for height
```

* in your **define** expression for your **create-scene** operator...
  * write an expression for place-image's x-coordinate based on **coord** to make something go right/left
  * write an expression for place-image's y-coordinate based on **coord** to make something go down/up
  * write expressions for both based on **coord** to make something go diagonally
  * write expressions for a shape's size based on **coord** to make it grow/shrink

* use **integer division remainders** (the **`modulo`** operator) to keep your image in-scene

  * (**`modulo`** *value divisor*) produces the integer remainder when you divide *value* by *divisor.*

  * why is this useful? because the integer remainder when you divide anything by *divisor* can never exceed *divisor* --- it must be between 0 and (- *divisor* 1)

  * so, for example, an x-coordinate of `(modulo coord WIDTH)` would always be between 0 and (- WIDTH 1) --
    and a y-coordinate of (modulo coord HEIGHT) would always be between 0 and (- HEIGHT 1)

* a few more image operations (read about them, and how to change pinholes, and more, in the **universe.ss** documentation):
  (note that *solid-or-outline* is always either the string `"outline"` or the string `"solid"`)

  * (**`circle`** *radius solid-or-outline color*) -> image
    produces a circle image of radius *radius* pixels, either "solid" or "outline", and of color *color*

```
(circle 15 "outline" "green")
```

  * (**`make-color`** *red-value green-value blue-value*) -> color
    expects *red-value*, *green-value*, and *blue-value* expressions each between 0-255 inclusive, and produces a color value. Note that this can be used for a color expression in MOST image operations (but NOT the fabric-teachpack operations...)

```
(circle 25 "solid" (make-color 200 100 100))
```

  * (**`star`** *num-points outer inner solid-or-outline color*) -> image
    produces a star image, with *num-points* points, an outer radius (distance from center to point) of *outer* pixels, an inner radius (minimum distance from center to base of points) of *inner* pixels, either "solid" or "outline", and of color *color.* It must have at least 2 points...!

```
(star 6 100 30 "solid" "darkblue")
```

* (**rectangle** *width height solid-or-outline color*) -> image
produces a rectangle image of *width* by *height* pixels, either "solid" or "outline", and of color *color*

```
(rectangle 35 20 "outline" "darkgreen")
```

* (**nw:rectangle** *width height solid-or-outline color*) -> image
similar to **rectangle**, except the resulting rectangle image here has a pinhole of (0, 0), making it especially nice for placing on an empty scene (hint!). Sadly, you cannot use **make-color** for nw:rectangle's color, though --- you need to express it as a string.

```
(nw:rectangle WIDTH HEIGHT "solid" "pink")
```

* (**triangle** *side solid-or-outline color*) -> image
produces an upward-pointing equilateral triangle image whose side is *side* pixels long, either "solid" or "outline", and of color *color.*

```
(triangle 40 "outline" "gray")
```

* **(ellipse** *width height solid-or-outline color*) -> image
produces a *width* by *height* ellipse image, either "solid" or "outline", and of color *color*

```
(ellipse 70 30 "solid" "black")
```

* (**regular-polygon** *num-sides side-length solid-or-outline color [angle])* -> image
produces a regular polygon image, with *num-sides* sides each of length *side-length* pixels, either "solid" or "outline", and of color *color.* If an *angle* is specified, the polygon image is rotated by that *angle degrees.*

```
(regular-polygon 5 13 "outline" "orange")
(regular-polygon 5 20 "solid" "brown" 55)
```

* (**text** *string font-size color*) -> image
produces an image of the text *string* (expressed as a string expression!) of point size *font-size* of color *color.*

```
(text "CS 131" 40 "purple")
```

* (**add-line** *image start-x start-y end-x end-y color*) -> image
produces an image by adding a line (colored *color*) from (*start-x, start-y*) to (*end-x*, *end-y*) to image *image* (based on *image*'s pinhole, note...!)

```
(add-line (rectangle 100 75 "outline" "darkgreen") 0 0 30 50 "black")
(add-line (nw:rectangle 100 75 "outline" "darkgreen") 0 0 30 50 "black")
```

* you can define a backdrop containing images that aren't changed in a scene, and then use that backdrop instead of an empty-scene expression inside create-scene;

```
; example
(define BACKDROP (place-image
                     (nw:rectangle WIDTH HEIGHT "solid" "blue") 0 0
                     (empty-scene WIDTH HEIGHT)))

(define (create-scene time-counter)
   (place-image bird 30 (modulo time-counter HEIGHT) BACKDROP)
)
```

  * want more "static" unmoving images in a scene? nest **`place-image`** or **`overlay`** expressions appropriately within BACKDROP's define

  * want more "moving" images in a scene? nest **`place-image`** or **`overlay`** expressions appropriately using **`time-counter`** within a create-scene function's body

* check out DrScheme's **universe.ss** documentation (the F1 key can help you reach it, remember, or the function key and F1 on a Mac) to read about more provided image operations - getting and changing pinholes, shrinking images, and more...!