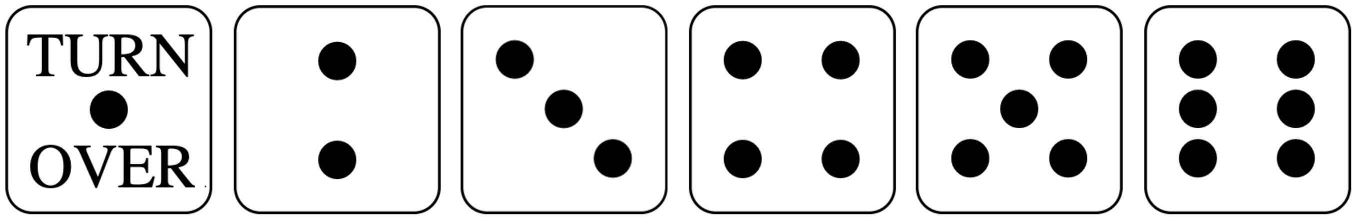
Optional Contest: Hog Dice Design

hog_dice_design.zip (hog_dice_design.zip)

Your true colors shine
Through the rolls of your pig dice
Like a code rainbow

> This contest is completely optional!

The game UI currently uses Western-style black and white dice:



We've made it possible for you to overwrite those dice with your own programmed graphics.

You could...

- Make much more colorful dice
- Play around with different ways to arrange the pips on each die
- Incorporate some piggies in the dice design
- Imitate the designs of dice from other cultures

The contest ends **Monday, September 20 at 11:59 PM**. Submit your dice before then!

Getting started

- Download the zip file linked at the top. That contains the Hog files, a submission script, and a new file `design.py` .
- Copy over your final `hog.py` into the folder.
- Open the `design.py` file.

Drawing dice

The `draw_dice` function takes a single argument `num` (the number rolled) and returns a Python object representing a vector graphic.

The provided starter code draws a 100x100 rectangle:

```python
def draw_dice(num):
    graphic = create_graphic(100, 100)
    draw_rect(graphic, 0, 0, 100, 100, fill="white", stroke="black")
    return graphic
```

The function `create_graphic` creates a vector graphic with the specified width and height. The `draw_rect` function adds a rectangle to the provided graphic (the first parameter).

Here are all the functions you can call to draw shapes on the graphic:

Function	**Description**
`draw_rect(graphic, x, y, width, height, stroke, fill)`	Draws a rectangle at the given `x` and `y` with the given `width` and `height`. The outline is colored with `stroke` and the shape is filled with `fill`.
`draw_circle(graphic, x, y, radius, stroke, fill)`	Draws a circle at the given `x` and `y` with the given `radius`. The outline is colored with `stroke` and the shape is filled with `fill`.
`draw_line(graphic, x1, y1, x2, y2, stroke)`	Draws a line that starts [`x1` , `y1`] and ends at [`x2` , `y2`]. The line is colored with `stroke`.
`draw_triangle(graphic, x1, y1, x2, y2, x3, y3, stroke, fill)`	Draws a `triangle` defined by three points: [`x1` , `y1`], [`x2` , `y2`], [`x3` , `y3`]. The outline is colored with `stroke` and the shape is filled with `fill`.
`write_text(graphic, x, y, str, stroke, fill)`	Writes the string `str` at the given `x` and `y`. The text outline is colored with `stroke` and the text is filled with `fill`.

The `fill` and `stroke` parameters are always optional and default to "black". You can specify other colors using color names (https://en.wikipedia.org/wiki/Web_colors) or hexadecimal notation.

Tips:

- You need to create a different design for each side, 1 through 6. One way to do that is to design each dice side separately and output the designs using an `if` statement. However, you wouldn't learn as much about programming that way. Instead, you should use a combination of loops and variables, with the variables tracking things like the spacing between pips, the number of columns needed, etc.
- The shapes are drawn sequentially on top of each other, so pay close attention to the order of the drawing commands.

- If you need inspiration, read about Dice on Wikipedia (https://en.wikipedia.org/wiki/Dice) or search Etsy for dice (https://www.etsy.com/search?q=dice).

Viewing dice

There are two ways that you can check your work as you go along:

- You can re-run `python3 hog_gui.py` to start the game locally and roll the dice. That's a good way to see how the dice all look together.
- You can also directly enter a URL in the browser to see the graphic for a particular die:
    - Dice 1: `http://localhost:31415/dice_graphic.svg?num=1` (http://localhost:31415/dice_graphic.svg?num=1)
    - Dice 2: `http://localhost:31415/dice_graphic.svg?num=2` (http://localhost:31415/dice_graphic.svg?num=2)
    - Dice 3: `http://localhost:31415/dice_graphic.svg?num=3` (http://localhost:31415/dice_graphic.svg?num=3)
    - Dice 4: `http://localhost:31415/dice_graphic.svg?num=4` (http://localhost:31415/dice_graphic.svg?num=4)
    - Dice 5: `http://localhost:31415/dice_graphic.svg?num=5` (http://localhost:31415/dice_graphic.svg?num=5)
    - Dice 6: `http://localhost:31415/dice_graphic.svg?num=6` (http://localhost:31415/dice_graphic.svg?num=6)

Contest rules

Teams can have one or two people. Each person can only be part of one team.

The course staff will vote on their favorite designs, and will look at both the design and the code.

The top three submissions will earn the following:

1. First place gets 3 points of extra credit.
2. Second place gets 2 points of extra credit.
3. Third place gets 1 point of extra credit.

Submit your designs

Before submission, please ensure that your designs abides by these guidelines:

- Designs should not contain anything discriminatory or offensive.

- Designs should not contain sexual or violent content, or otherwise objectionable content.
- Designs must not contain any personal information.

We reserve the right to hide designs that do not follow these guidelines.

Please also add a caption for your dice by replacing `DICE_CAPTION` in `design.py`:

```
DICE_CAPTION = "these dice taste good with rice"
```

Ready to submit? Run:

```
python3 ok --submit
```

> If you see an SSL error when trying to submit, that's an issue with your local installation of Python 3 on Mac OS X. Navigate to `Applications/Python 3.[Version]` and run `Install Certificates.command` by double-clicking it.

Once submitted, you should see your designs show up in the showcase (https://dice.cs61a.org)! 🎉 🎲 🎉