

# Mission 1: Rune Trials

Start date: 23 August 2017

**Due: 25 August 2017, 11:59AM**

Readings:

- Textbook Sections 1.1.1 to 1.1.4

Every year, the Source Academy holds a trial for worthy candidates to be trained in the Way of the Source. This will be their first step on a lifelong journey of bringing peace and justice to the galaxy.

After years of anticipation, it is finally your turn. The instructors will give you some basic training before you are tested with a set of challenges. If you pass these challenges, you will be initiated into the Academy.

## Mission Overview

In Lecture 1B, we discussed how to generate runes. Today we will practice drawing them in Task 2, 3 and 4.

For this mission, the Playground loads the `webGLrune.js` script library. This provides the four runes discussed in class `rcross_bb`, `sail_bb`, `heart_bb`, and `nova_bb`. For example, you can display `rcross_bb` in the Canvas with the following command in the Console:

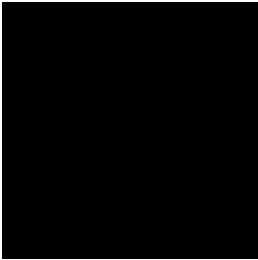


```
show(rcross_bb);
```

NOTE: To be able to draw runes ensure you have changed `source_week_3` to `rune_2d` on the playground (the dropdown tab on the top-right corner of the playground).

Also defined in `webGLrune.js` are the following functions as discussed in class:

- `stack`
- `stack_frac`
- `quarter_turn_right`
- `flip_horiz`
- `flip_vert`
- `turn_upside_down`
- `quarter_turn_left`
- `beside`
- `make_cross`
- `repeat_pattern`
- `stackn`

In addition to the ones you saw in Lecture, we have also defined a few more basic runes that you can use:

Rune	Reference Image
<code>black_bb</code>	
<code>blank_bb</code>	
<code>circle_bb</code>	
<code>pentagram_bb</code>	

In writing rather large, complex programs, one does not often understand (or even see) every single line of the program, since such programs are usually written by several people, each in charge of smaller components. The key idea in functional abstraction is that as the programmer, you need not understand how each function works. All you need to know is what each function does and its signature (such as what parameters to pass). More specifically, **you need not read the source for the predefined functions listed above**. You may refer to the lecture slides to understand what arguments each function requires and its corresponding effect. Now we will use these functions as primitives building blocks to draw our own pictures.

This mission consists of **four** tasks. **Click here for the link to the template**. Use the template provided in the link to complete the tasks.

NOTE: For now always refresh the page before testing out your code. Also only have one image to be shown at a time when testing your code.

## Task 1:

Your first task shall be to practising writing comments. This comment will include a short introduction and one interesting fact about yourself!

You can use either single line comments or multi-line ones.

```
// Example of a single Line comment
```

```
// I am Luke Skywalker
```

```
// Example of a multi-Line comment
```

```
/*
```

```
One Interesting fact about myself is my father was Darth Vader!
```

```
I am one with the force!
```

```
*/
```

```
***
```

## Task 2:

Write a function `mosaic` that takes four runes as arguments and arranges them in a 2 by 2 square, starting with the top-right corner, going clockwise. In particular, the command

```
show(mosaic(rcross_bb, sail_bb, corner_bb, nova_bb));
```

should draw the following:



NOTE: Take note of the position of each of the images in the mosaic.

### Task 3:

Write a function `upside_down_mosaic` that takes four runes as arguments and creates a mosaic that is rotated a 180 degrees.

```
show(upside_down_mosaic(rcross_bb, sail_bb, corner_bb, nova_bb));
```

should draw the following:



NOTE: Its the whole mosaic turned 180 degrees, not individual pieces rotated and then joined into a mosaic.

### Task 4:

Suppose we now want to apply a different transformation to the mosaic pattern. It would be tedious to create a new one for every transformation function now, wouldn't it? Hence why we are going to use the power of abstraction!

Now, write a function that takes four runes and a transformation function as arguments and creates a mosaic, after which you apply the transformation function to the mosaic and return the resultant image. In particular, the command

```
show(your_function(rcross_bb, sail_bb, corner_bb, nova_bb, make_cross));
```

should draw the following:



Also give your function an appropriate name.(i.e. replace `your_function` with your own name)

NOTE: Take note of the position of each of the images in the resulting image.

## Submission

To submit your work for this mission, copy the url on your browser and email it to your respective Avengers. Strictly follow to the deadlines set at the start of this file.

IMPORTANT: Make sure that everything for your programs to work is on the left hand side (source code) and not in the interpreter! This is because only that code is preserved when opening the url you have emailed to us.