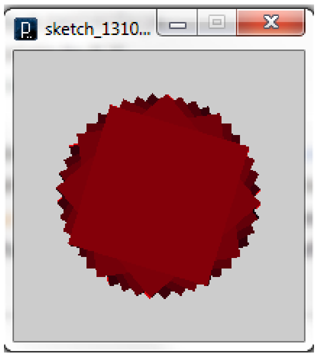
Interactive TBA

Art is cool! But, its even cooler when it isn't static and changes over time. Time based art does just that, it reacts to time as a phenomena. Think of TBA as an abstract clock in a lot of ways!



PROCESSING
SKILL REQUIREMENTS
DIFFICULTY 1-5

ELECTRICAL PROTOTYPING	○○○○○
ROBOTICS	○○○○○
SOLDERING	○○○○○
PROGRAMMING	●●○○○
DIY	○○○○○

MATERIALS LIST

- Computer
- Processing
- Graph Paper
- Colored Pencils
- Super fine tip sharpie

STEP 1: Draw Something!

We highly recommend that you spend some time coming up with an analog version of what you plan to create in Processing. Sometimes its hard enough to come up with something when you are just drawing it let alone drawing it in code. You can then note origin points of shapes, their color, and widths and heights. This step is not required, but highly recommended.

STEP 2: Patterns, Shapes and colors oh my!

To start our interactive time based project we are first need a set of shapes to manipulate and use. Create a few shapes and/or lines and use the `fill(R,G,B);` or `stroke(R,G,B);` function to give a line color. If you want to get up and running here are a few lines of code you can change to make your own.

```
fill(255,0,0);
rect(50,50,100,100);
```

STEP 3: Background Where?!

Remember back in project 0 we talked about the `setup()` code only runs once at the very beginning of your sketch and then the `draw()` runs over and over 30 times a second. If we take any function and place it in the `setup()` it will only run once right at the beginning. A good example of this is putting the `background()` in the `setup()` and that will cause the background to only be drawn once. After that as shapes move they leave tracers behind them as they are drawn over and over again. So if you want a shape to move but not leave behind previously drawn shapes then you put your `background()` in your `draw()`. If you want tracers, put background in `setup()`.

STEP 4: First comes the Push, then the Pop!

We can now easily control each shape in different ways using the built in variables like `mouseY` and `mouseX` as parameters. But the question arises...How do I move a group of shape or lines? If I have drawn an awesome spider, how do I move all of the shapes that make up that spider all at once?

Enter the matrix! A matrix in Processing is taking a group of functions and putting them on their own invisible background. We then can move the entire matrix, rotate, and scale it. To create a matrix we use push and pop; we start a matrix with `pushMatrix();` and end the matrix with `popMatrix();` you can see an example of a couple of shapes in a matrix in the image to the right.

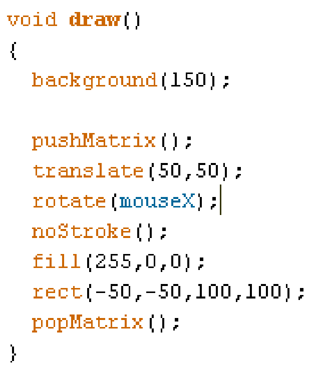
```
void draw()
{
  background(150);

  pushMatrix();
  noStroke();
  fill(255,0,0);
  rect(50,50,100,100);
  popMatrix();
}
```

STEP 5: Translation...

Once we can group shapes within a matrix we can then manipulate each group independently of one another. The manipulations are called translations. The basic translations we can use for a matrix are as follows:

scale(%); - Scales whatever is in the matrix by a percentage
translate(X,Y); -Translate is moving a matrix origin to X,Y
rotate(R); -Rotates a matrix by R radians (3.14 Radians= 180 Degrees)

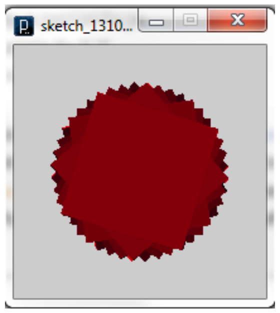
So if we want to rotate something `mouseX` Radians we add `rotate(mouseX);` to our code above. Try using the example code here, and once you get rotate working try swapping it out for `scale();, or rotate();`. Once you have played around with translating a matrix create a second with other shape in it and have it do something.



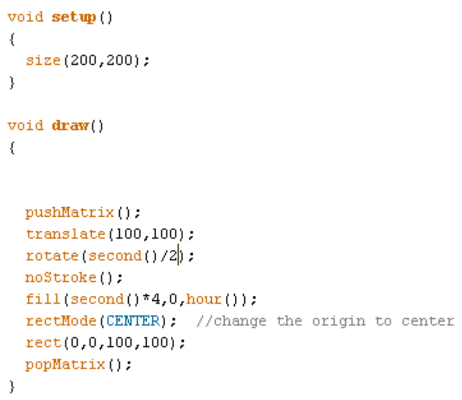
```
void draw()
{
  background(150);

  pushMatrix();
  translate(50,50);
  rotate(mouseX);
  noStroke();
  fill(255,0,0);
  rect(-50,-50,100,100);
  popMatrix();
}
```

STEP 6: Adding Time

Once you feel comfortable with using a matrix and translating it use time based variables and leave your `background()` in your `setup()` code. This will cause the background to only be drawn once and you will get some pretty cool looking art. You can now add time based variables and your mouse variables to your matrix translation. Here are a few time based variables to try and some example code of a TBA project.



`millis()`	-number of milliseconds since the sketch started
`second()`	-The second hand taken from the time set to your computer
`minute()`	-The value taken from the time set to your computer
`hour()`	-The hour taken from the time set to your computer
`day()`	-The current day
`month()`	- The current month
`year()`	-The current year



```
void setup()
{
  size(200,200);
}

void draw()
{


  pushMatrix();
  translate(100,100);
  rotate(second()/2);
  noStroke();
  fill(second()*4,0,hour());
  rectMode(CENTER);  //change the origin to center
  rect(0,0,100,100);
  popMatrix();
}
```

STEP 7: Share your work!

If you would like to share your work...which you do! You can sign up at openprocessing.org. Open Processing allows you to share your code, view the drawing online, as well as allows others to fork or borrow your code to manipulate and change it on their own while still keeping your sketch intact for others to check out.

OpenProcessing

TAKING IT FURTHER

- try using math and variables to make interaction more intricate
- make a snowman wave!
- play around with `text("text",x,y);`