Creating Your Own Shapes

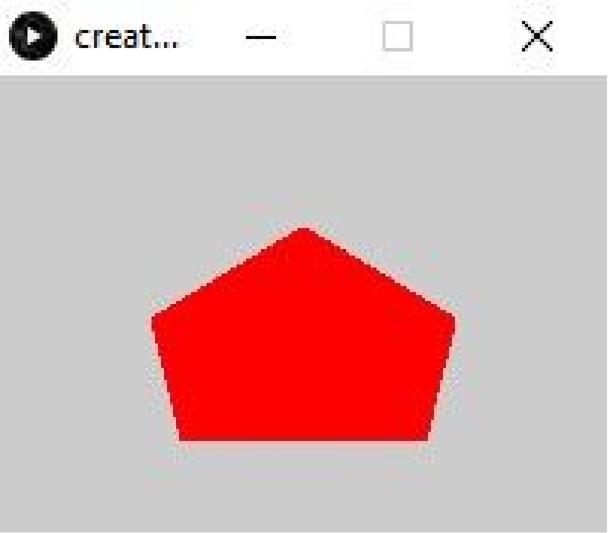
So far in this course we have encountered ellipses, rectangles, line, points and maybe triangles. But all of these functions were given to us and we could only do so much with them. If we want to create our own unique shapes we need a few more commands.

The first of these is PShape. PShape is an object. An object can be used like a variable in this context but has some extra features. PImage is also an object that you have used before. Like PImage we declare this object at the top of our sketch.

```
PShape  pentagon;
```

We then need to initialise the object. This is slightly more tricky when it come to PShape because we need to create the shape here, where we initialise it. We create the shape using what is called a vertex (or “vertices” if we have more than one). A vertex is just another name for a point on a shape, so you can just think of it as a point. The shape is then created by drawing a line from vertex to vertex (or from point to point). (**Remember** you initialise the variable in void **setup**().)

```
pentagon = createShape();
pentagon.beginShape();
pentagon.fill(255, 0, 0);
pentagon.noStroke();
pentagon.vertex(0, 0);
pentagon.vertex(50, 30);
pentagon.vertex(40, 70);
pentagon.vertex(-40, 70);
pentagon.vertex(-50, 30);
pentagon.endShape(CLOSE);
```

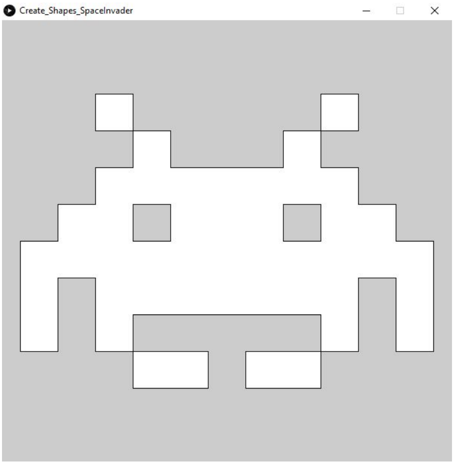


When you have done this you can now use your custom shape. To do this you use the shape function.

```
shape(pentagon, 300, 300);
```

This will draw the pentagon shape at the position 300 across and 300 down. What you get should be similar to the pentagon in the picture above. (**N.B** you need to be careful where you set as (0,0) or the origin when creating your shape as it can make future uses easier or harder, depending on what you use it for. If you set the center of a shape as the origin collision detection becomes easier, whereas if you set the bottom right hand corner of the shape as the origin collision detection becomes a lot harder.)

Using the same steps as above you can create your own custom shapes like the space invader below and use them in your future projects.



To create the eyes for the space invader you will need one further set of functions, beginContour() and endContour(). These are used in much the same way as beginShape() and endShape(). The only big difference is that if the points of the main shape are going clockwise the points of the contour must go anticlockwise or vice versa. (There is some more detail in the reference guide which might help here).

Add in the code below just before pentagon.endShape(CLOSE), this will create a triangular hole in the pentagon.

```
pentagon.beginContour();
pentagon.vertex(-20, 30);
pentagon.vertex(0, 60);
pentagon.vertex(20, 30);
pentagon.endContour();
```

Challenge

- Now that you have learnt to create your own shapes your task is to create a star like the one on the right.
- If you think that is too easy you can try making the space invader on the previous page or a design of your own.