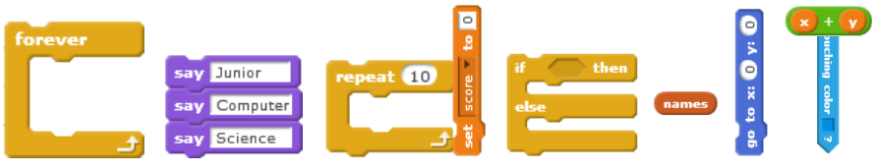



code-it.co.uk

# Exploring Regular 2D Shapes & Patterns

**Program Aim: Program regular 2D shapes using sequence, repetition, nested loops, simple and complex procedure.**

**Programming Concepts**

-Sequence

-Repetition

-Nested Loop

-Procedure

**Resources**

Masking tape

Whiteboards and pens

Space to draw shapes

**National Curriculum Programs of Study**

**Pupils should be taught to:**

-**design, write and debug programs that accomplish specific goals**, including controlling or simulating physical systems; **solve problems by decomposing them into smaller parts**

-**use sequence,** selection**, and repetition in programs;** work with variables and various forms of input and output

**-use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs**

| Creating Regular 2D shapes using sequence | More efficient shapes using loops | Shape patterns using nested loops | Simple procedures | Procedures using an input |
|---|---|---|---|---|

Learning Path →

**4th Draft Expanded Planning**

**Terminology**

**Sequence** –Instructions that follow after each other. Scratch works from top to bottom. Sometimes the order of the sequence is important sometimes it is not.

**Repetition** –Code that repeats. This might be a set number of times or forever while the program is running.

**Nested Loop** –A loop that is inside another loop.

**Procedure** –A section of code that can be run lots of times but which can have number or text inputted which changes the properties of how it runs. So if we had a procedure called Equilateral Triangle which had an input which determined how long each side was.

Equilateral Triangle 30 would be smaller than

Equilateral Triangle 60.

**Experience Needed**

No previous experience is necessary. However, I would recommend for upper KS2, Year 5+, 9 Years of age+

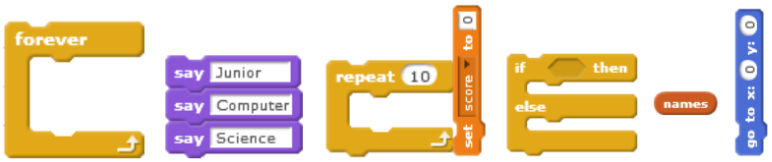
CC BY NC ND

You can follow me on Twitter

@baggiepr

You can find lots of free resources at code-it.co.uk

I enjoy sharing my journey with fellow educators around the world




# Exploring Regular 2D Shapes & Patterns P2

CC BY NC ND

## 1, Preparation before the Lesson

Print the sheet **Pupil shapes sequence sheet pdf** found with this planning.
Download **Shapes Programming PDF** slides and **Loops and nested loops (movement)** slides.
You will also want to print the **assessment booklet** which goes with this planning.
You could administer this as one summative assessment at the end of the unit or after each section.

## 2, Sharing the Aims of the Module

Looking for the best way to program Scratch to draw regular 2D shapes.

### *2a, Optional Tinkering/Exploring*

*You could give pupils time to explore how to do this themselves in Scratch first. Stress that this must be their own work as you are going to assess what they know and use this to determine their next steps.*

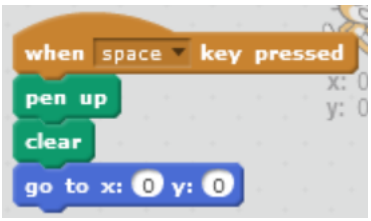
## 3, Creating Shape Algorithms

Explain that an algorithm is the precise set of instructions needed to solve a problem. Our problem is how to draw a regular shape using a computer program. Use **slide 2-5** of the **Shapes programming PDF** to illustrate how to do this. Now give pupils a few minutes to create a square and triangle algorithm on the **Pupils shapes sequence sheet PDF**. If pupils are really struggling with these ideas you can draw out shapes on the floor and walk over them. Exploring right angles planning may be more appropriate for your pupils at this time. http://code-it.co.uk/rightangles

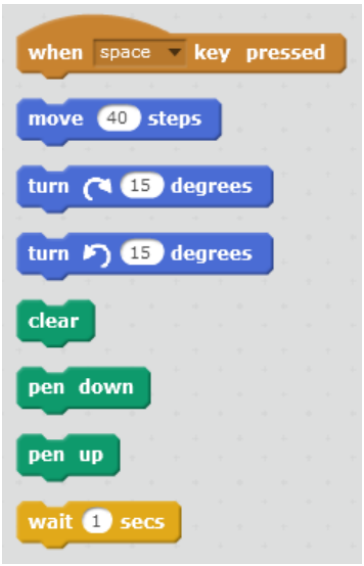
## 4, Turning Algorithms into Code

Shrink the sprite down to very small or draw a new sprite and centre it. Create the code shown on the right so that pupils have an emergency button to bring the sprite back. You may want to give them time to make and test this before moving on.

Show pupils where some of the code blocks they need are. Drag them out and leave them as shown. Indicate that each starting block should start with a different letter. Ask them to include a wait block after ever move and turn block. Give them time to create the code themselves. If pupils are struggling you could demonstrate how you would turn one of their algorithms into code. Add wait 1 second blocks between every move or turn to slow the program down so pupils can see where the bugs (errors are). Then give them time to turn their triangle and square algorithms into code.




**Emergency Button**




**Blocks Needed**

CC BY NC ND

If the code doesn't work they can;

- Walk over the algorithm again to see if their sequence was correct.
- Check to see they have recorded the correct angles.
- Drag out the code and run it one block at a time. Does it follow the whiteboard algorithm exactly?

**5, More Efficient Code with Loops**

Ask pupils if they can spot the pattern in their sequence. Some of them may have already started to use a loop to make the code more efficient.

Show them the repeat so many times loop. Can they draw the shape with the least possible code and the least possible sprite moves. Remind pupils that they must still include some wait blocks to slow the code down.

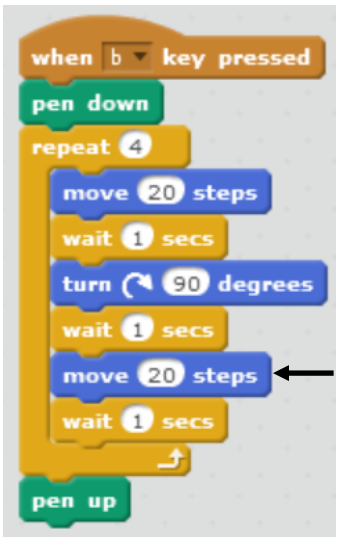
Give them plenty of time to convert some of their sequence code into repeat loop shapes.

Midway through this exercise stress the need for pupils to use the least blocks possible and the least sprite moves.

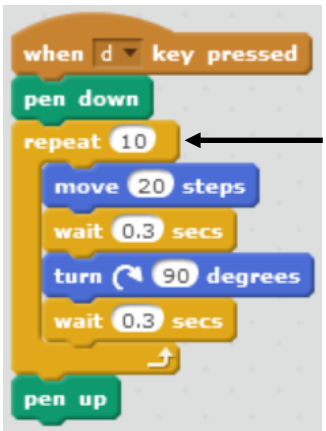
**Square within a repeat loop**

**Formative Assessment**

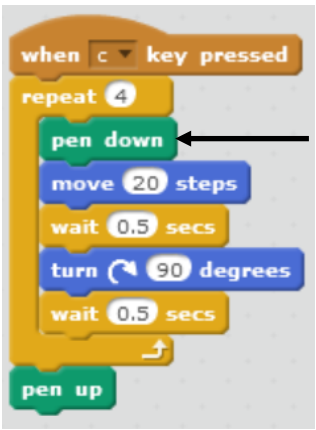
| **Formative Assessment** | **Corrective** | **Enhanced** |
|---|---|---|
| Pupils have one move and one turn inside a repeat loop with a pen down to start and a pen up to finish. Each shape should have its own separate keyboard input (letter or number) to trigger the program. See example above. | If they have too many code blocks but it still draws the shape. Praise what they have done so far but don't accept the partial solution.<br>If there are too many repeats get them to count the number of times the sprite moves or turns. Is it the least possible?<br>If the pen down is inside the loop ask if the program needs to put the pen down every time it repeats. Can it do this just once? | Can they create other 2D regular shapes? Point out that they are using part of the outer angle and that they can calculate the angle by dividing 360 degrees by the number of equal sides.<br>**Equilateral Triangle Angle**<br>360 divided by 3 =120 degrees. |




Extra code examples




Extra repeats




A code block running more times than it needs to

**5a, Loop Algorithm** Show pupil the simple loop algorithm on slides 6–9 of the Shapes programming PDF. This is to expose them to different algorithm design.

# Exploring Regular 2D Shapes & Patterns P4

### 6, How many regular shapes can they code?

Show pupils how they can calculate the angles of the shape needed by dividing 360 degrees by the number of sides. So an equilateral triangle angle would be 360 divided by 3 =120 degrees. Pupils can use calculators built into most computer operating systems to do this.

Revise slide 2 which highlights that they are dealing with a part of the outer angle not the inner angle.

**NOTE** that shapes with a high number of sides might distort as the sprite bounces off the edge.

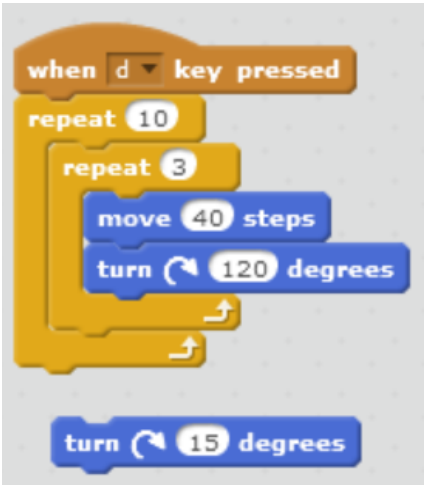
Give pupils time to duplicate and adapt their code to make lots of regular shapes.

**Section 6 assessment sheet goes with this activity.**

### 7, Drawing patterns using nested loops

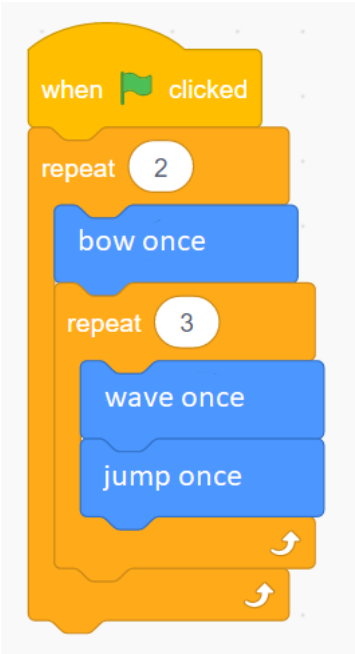
Programmers can use loops within other loops. We call these nested loops. Pupils may want to reduce the wait blocks to a tenth of a second or remove them altogether for this section.

Ask pupils to stand up and move away from their computers. Open the PDF **Loops & Nested Loops (Movement)**. Lead pupils through the physical activities. After a few simple loop activities they are introduced to a couple of nested loops. After they have completed this activity adapt their loop shape code to look like the code on the left. How can they use the turn right 15 degrees to create a repeated pattern?




How could they adapt the repeats, degrees or both to ensure a completely circular pattern, where the shape is repeated for 360 degrees?

Give pupils plenty of time to experiment with nested loops and multiple shapes.




**Example Nested Loop (Movement) Activity**

### 7a Nested Loop Algorithm

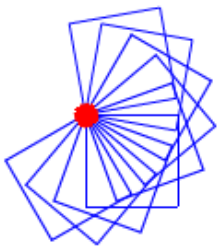
Show pupils the nested loop algorithms slides 11 and 12.

**Good questions are crucial at this point**

How can you make your pattern repeat for 360 degrees? *(make sure the outer loop repeat multiplied by the outer loop turn equals 360)*

What could you change to leave a space in the middle of the pattern? *(Add a move block into the outer loop)*

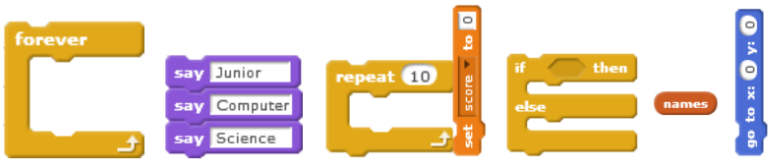
Once pupils have puzzled this out can they think of a rule for why it works?



**A repeated square pattern**

**Section 7 assessment sheet goes with this activity.**

CC BY NC ND

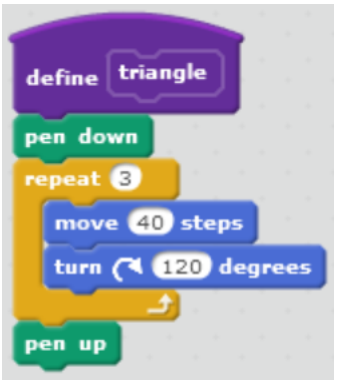



# Exploring Regular 2D Shapes & Patterns P5

## 8, Creating simple shape procedures

Explain that procedures are sections of code that can be written and then used lots of times. They help programmers concentrate on the bigger picture because they don't have to re-write the same code lots and lots of times.

After you have shown pupils how to create a simple shape procedure attach the code underneath as shown.

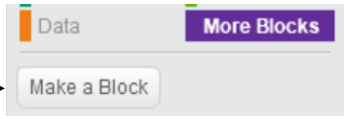



Encourage pupils to create procedures for each shape.

These can be used to create complex geometric patterns.

**To create a procedure**

Click on make a block.




Give the block (procedure) a name.

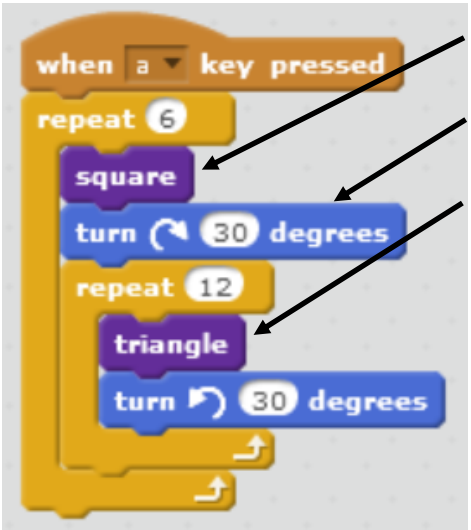
This one has been called triangle.

You can add more advanced options here. We will use this in the next section.

The procedure will look like this.

It can be used/run/activated by using the triangle block that goes with it.

Currently it has no code so will do nothing.




This code starts a procedure called square which is similar to the triangle procedure above so it draws a square.

It then turns right 30 degrees.

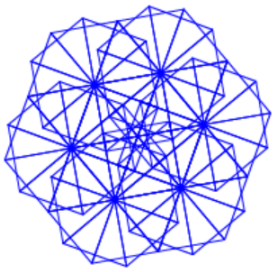
It then uses the triangle procedure (above) to draw a triangle twelve times interspaced with 30 degree turns.

It then uses the square procedure again to draw a square.

It then turns right 30 degrees.

And so on.

The pattern looks like this.



Give pupils lots of time to experiment with this. After general experimentation insist that patterns stop after 360 degree rotations to encourage them to think through the geometric rules.

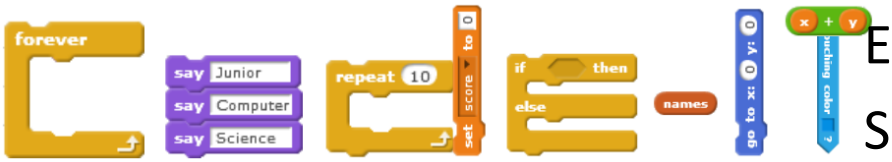
**Section 8 assessment sheet goes with this activity.**

## 8a Share the simple procedure algorithm slide 13

**Messy Code** [1]

Leaving code that is unfinished or only partially working inside a project. Allowing solutions that only partly meet the original goals/algorithm. Occasionally pupils can have an earlier code attempt hinder current code. Remove starting blocks and label parts that are unfinished but pupils want to come back to.

CC BY NC ND




# Exploring Regular 2D Shapes & Patterns P6

## 9, Creating procedures that have inputs

When you create a procedure there are options to add inputs. These allow you to make one procedure but vary its properties by inputting different data. We are going to stick to number inputs.

Ask pupils if they would like to create one procedure that will draw **any** regular 2D shape?

Ask pupils what properties every regular 2D shape has.

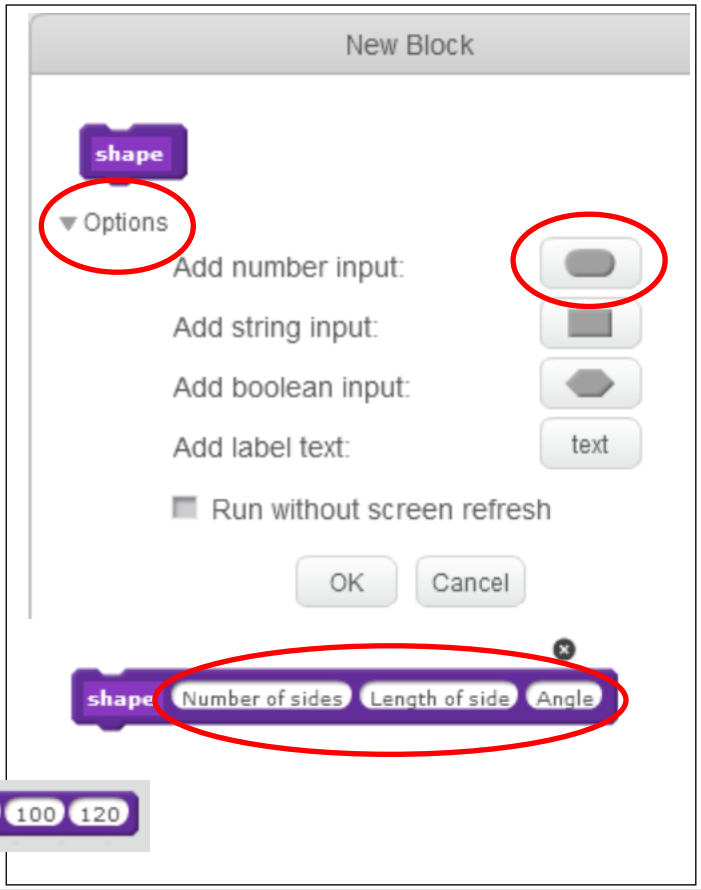
**Length of side**

**Number of sides**

**Angle of Turn**

Create a new procedure called shape.

Add three number inputs from the options as shown on the right. Name them after the properties of a regular 2D shape.




Ask pupils which number in the basic loop program corresponds to which property?

**Length of side** **Move x steps**

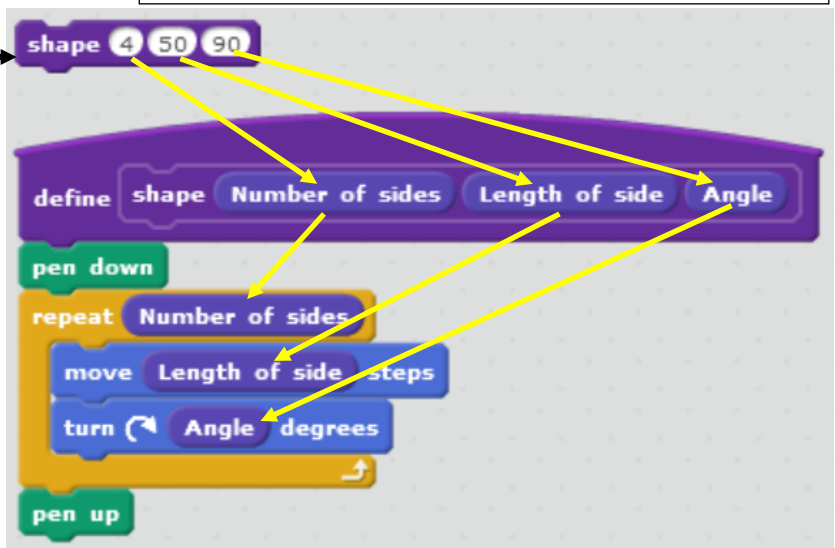
**Number of sides** **Repeat x**

**Angle of Turn** **Turn right or left x degrees**

Drag the correct block into each number.

This code would draw a triangle whose sides are 100 pixels long.

This code draws a square whose sides are 50 pixels long.



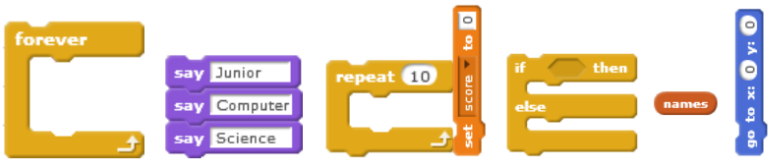

There is an info and prediction sheet that has the diagram on the right to help pupils understand how this works. Give this to pupils now but ask them to not answer the questions until they have made and used this procedure.

Give pupils time to create lots of geometric patterns using this shape procedure.

**Section 9 assessment sheet goes with this activity.**

**9a, Procedure with input algorithm** Show the pupils slide 14 & 15

CC BY NC ND



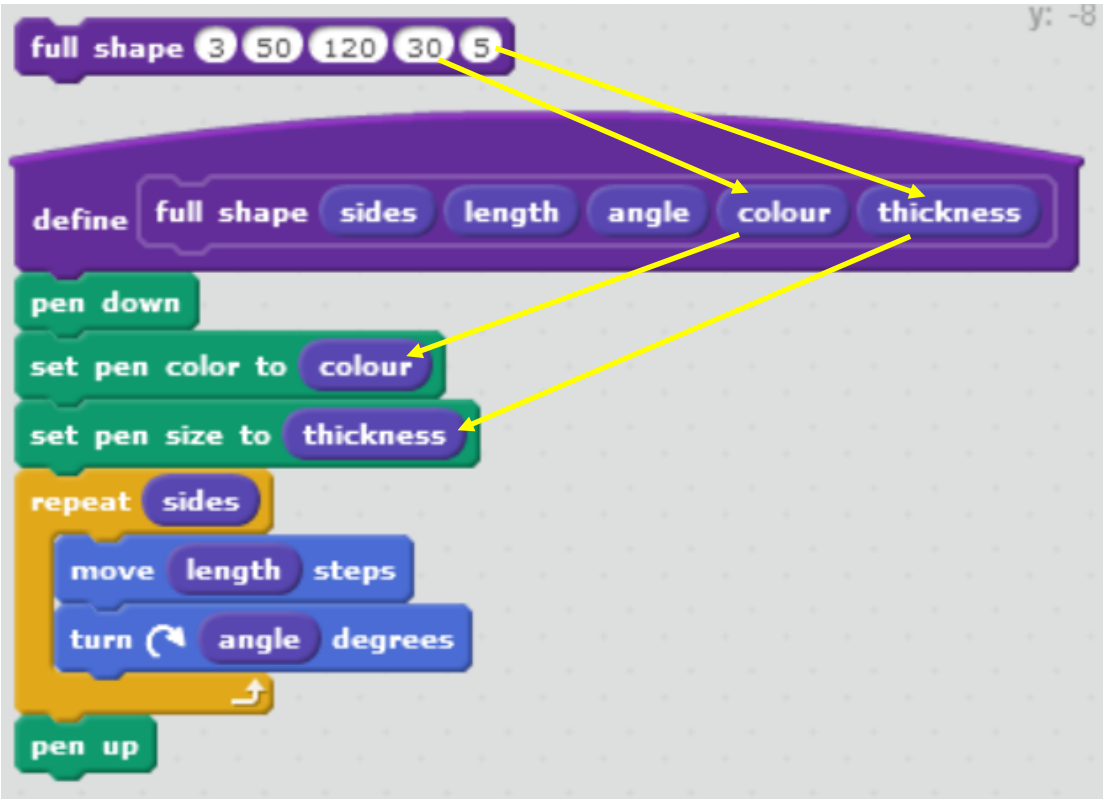

# Exploring Regular 2D Shapes & Patterns P7

## 10, Optional Extension (Other properties as inputs)

Ask pupils if there are other properties of a shape that haven't been included?

**Thickness of the line**

**Colour of the line**



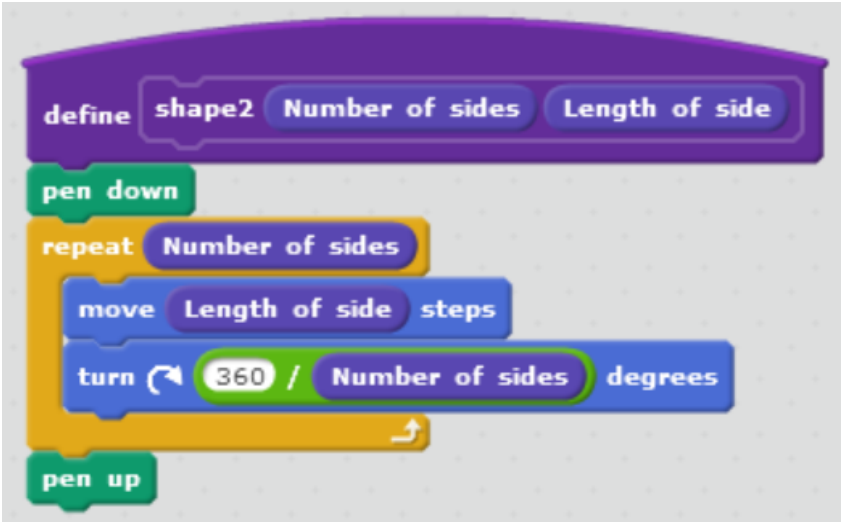

Can pupils write a new procedure that includes these as well?

Let them puzzle this out but here is one working method

CC BY NC ND

## 11, Optional Extensions (More efficient procedure)

Can pupils use the information that each shapes outer angle is number of sides divided by 360 degrees to automate the shape procedure so that the user only has to type in the number of sides and the length of the sides to draw the shape.



It can help to point out that

**They won't need an angle input.**

**They might need a divide block.**

This can be a real wow moment for some pupils.

**12, Optional extension (Using variables with procedure blocks)**

What happens when you change a variable inside a loop and use it with a procedure?

Show pupils the thickness example.

Can they trace what is happening in every repeat?

**Thickness**

**Length of Side**

Draw triangle thickness 1

Turn 36 degrees

Draw triangle thickness 2

Turn 36 degrees

Etc

Can they create more variable to investigate how this might affect other number inputs?

What might happen if you combine a variable with extension 11?

**13, Assess Computational Attitudes**

Use the chart on the next page to draw out key computational attitudes in this module. Pupils can also look for these attitudes in their own work and that of their fellow classmates.

There are some stickers to go with these on code-it.

http://code-it.co.uk/attitudes/

If you don't wish to print the stickers there is a non sticker recording sheet that can be

# Exploring Regular 2D Shapes & Patterns P8

## Computational Attitudes

Repeated patterns all sections from paragraph 5 onwards

This planning is all about multiple methods to achieve the same thing

What fantastic patterns can they make

*I recognise there is more than one way to solve/describe a problem*

*I don't just accept the first solution*

*I look for a range of solution to the same problem*

*I look for how a project can be extended*

*I can evaluate my solutions against a set criteria*

*Handles Ambiguity*

*Open Ended Problem Solver*

*I can break complex problems into parts*

*I can design criteria to evaluate my creations*

*Evaluates*

Insert picture of your students here

*Copes with Complexity*

*I can discover / concentrate on the most important part of a problem*

*I can identify patterns in problems & solutions*

*I can contribute useful ideas to a partner or group*

*I can encourage others to share their ideas*

*Communicates*

*Adapts*

*I can adapt existing ideas to solve new problems*

*I can develop, test and debug until a product is refined*

*I lead using all the people talent in my group*

*Investigates*

*I make predictions about what will happen*

*I learn from setbacks and don't let them put me off*

*I can persevere even if the solution is not obvious*

*Perseveres*

*I repeatedly experiment through predicting, making, testing & debugging*

@baggiepr

Inspired by Behaviour Rubric Created with @MarkDorling and linked at http://code-it.co.uk/attitudes/

If you are paired programming or working with a response partner

Bugs in computing are normal

Have a debugging wall of fame for all those who fix a bug themselves.

See the assessment activities

The whole project is about building on prior knowledge to adapt an idea

CC BY NC ND