Creating Dot Paintings with Pencil Code

UON Coding & STEM 4 Schools – An Introduction to Coding and Computational Thinking

# Introduction

This document has all of the steps for the **Creating Dot Paintings with Pencil Code** activity. You can use this document as a guide when following along with the demonstration or you can also work through this document at your own pace during the session.

In this activity, you will create some patterns with dots, similar to [traditional dot paintings](https://www.google.com/search?tbm=isch&q=dot%20painting), in *Pencil Code*. An example of one of these patterns is shown in the image below:



To create this picture, you will combine Mathematics and Coding, as well as apply the following *Computational Concepts:*

* **Sequences**: instructions that a computer follows in order
* **Loops:** making a computer follow commands multiple times
* **Operators:** operations on numbers (such as addition and subtraction) and *variables*

You will also learn about the **function** block, which is in the **Operators** section of the *Blocks* view. **Function** blocks can be used to combine multiple instructions into a single block.

# Acknowledgements

This activity is an adaptation of an activity from UCL ScratchMaths. ScratchMaths is a resource for teaching Coding in Scratch alongside Mathematics at the Primary School level. This activity is an adaptation of a few different activities from the *Tiling Patterns* module of ScratchMaths. You can find out more about the ScratchMaths resources from [their website](http://www.ucl.ac.uk/ioe/research/projects/scratchmaths).

# Getting Started

The *Pencil Code* website is [pencilcode.net](http://pencilcode.net/). There is also a link to the website on the *From Blocks to Text* session page of the workshop website. You should be able to use any web browser to access and use *Pencil Code*, but we recommend that you use *Mozilla Firefox* or *Google Chrome* for this activity.

We recommend that you complete the **Drawing a** **Sky Full of Stars with Pencil Code** activity before starting this activity. The document for that activity is able to be downloaded from the *From Blocks to Text* session page on the workshop website.

The **Drawing a** **Sky Full of Stars with Pencil Code** activity document also includes some information about registering a *Pencil Code* account, which may be useful.

# Blocks and Text Views

*Pencil Code* allows you to switch between two views: *Blocks* and *Text*. You can switch between these two views by clicking the button that is highlighted by the green rectangle in the image below.

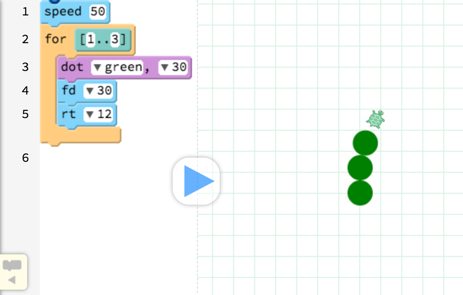


You can switch between the views as often as you would like. In this activity, the screenshots that we have included show the *Blocks* view of *Pencil Code*. However, if you would like to, you can use the *Text* view or switch between the two views when completing this activity.

In a couple of the steps, we recommend that you switch to the *Text* view, because you can copy and paste commands in the *Text* view, whereas you cannot copy and paste commands in the *Blocks* view. Additionally, unlike the *Text* view, the *Blocks* view shows you all of the different categories of commands (for example, *Move*, *Art* and *Text),* so you do not have to remember all of the commands.

# Drawing a Circle of Dots

The first step is to draw a circular pattern of dots. Create the code that is shown in the image below and run the program.



The Turtle should draw three green dots, like those that are shown in the image above. We would like to draw a full circle of green dots, as shown in the image below.



What do you think needs to change in the blocks above to make it draw a full circle of dots? Change those blocks so that the Turtle draws a circular pattern of dots.

Hint: How many degrees would the Turtle have to turn in total to draw the pattern in the image above? How many degrees does it turn when it draws the three green dots?

# Forwards and Backwards

As you learned about in the previous steps of the Turtle has to turn 360 degrees to draw the circle of dots.

We will now change the way that the Turtle moves around the *Canvas* and draws the circle of dots.

At the moment, the **algorithm** that the Turtle follows (30 times) is explained below:

* The Turtle draws a dot
* The Turtle moves forward
* The Turtle turns to the right by 12 degrees

We are going to change the **algorithm** so that the Turtle moves to the centre and back when it draws the circle of dots.

Now, the **algorithm** that the Turtle will follow (30 times, as well) is explained below:

* The Turtle moves forward from the centre
* The Turtle draws a dot
* The Turtle moves backwards to the centre
* The Turtle turns to the right by 12 degrees

You can see this **algorithm** implemented in *Pencil Code* in the image below. You should change the blocks that you have, so that they match the code shown below.

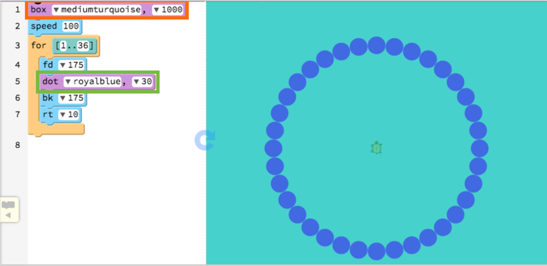


Once you have changed the code to match the image above, run the program. The Turtle will now draw the same circle of green dots as before. However, it will now finish up in the centre of the *Canvas*.

# Background Colour Change

Now you should change the colour of the *Canvas* background and the colour of the dots that the Turtle draws. Like the last activity, you can use a **box** command to fill in the background colour. A **box** command has been added to the scripts in the image below, as you can see highlighted by the orange rectangle.

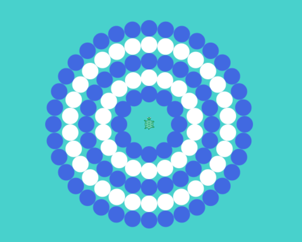
The colour in the **dot** command has also been changed to a different colour, as highlighted by the green rectangle in the image below.



We have used **mediumturqoise** for the background colour and **royalblue** for the colour of the dots. You can use different colours for your patterns, if you would like to. To see the different colours that you can use in *Pencil Code*, you can click on the **Colors Reference** link.

# Drawing the Other Circles

Next, you will add some more commands to draw more circles of dots, so that you end up with a pattern like the image below.



To make the Turtle draw more circles of dots, you can add more **for** blocks to the scripts that you already have. However, the degrees that the Turtle turns, repeats and the steps that it moves forwards and backwards, will be different for each of the circles.

In the table below, you can see the **degrees**, **times repeated** and **steps** for the different patterns. You have already draw Pattern 1, which is the outer-most circle of dots. You should use the table below to create the other 4 circles of dots, which you can see in the image of the pattern on the previous page.

|  |  |  |  |
| --- | --- | --- | --- |
| Pattern | Degrees in Each Turn | Times Repeated | Steps Forward and Backwards |
| 1 | 10 | 36 | 175 |
| 2 | 12 | 30 | 145 |
| 3 | 15 | 24 | 115 |
| 4 | 20 | 18 | 85 |
| 5 | 30 | 12 | 55 |

If you are using the *Blocks* view to complete the activity, you may want to switch to the *Text* view. That is because you can copy and paste commands in the *Text* view, which may be useful. Note that, you can also increase the number in the **speed** command to make the Turtle draw the circles quicker, if you are finding that it takes too long for the Turtle to finish following the instructions.

After adding the commands to draw the pattern on the previous page, you should have **5 for** blocks (one to draw each of the circles).

# Drawing an Extra 2 Dots

After adding the code that will make the Turtle draw the 5 circles of dots, you should add two more **dot** commands. The first **dot** command will draw a big dot, that will be behind the circles. We have added this **dot** command because, in later steps, the patterns may overlap each other.

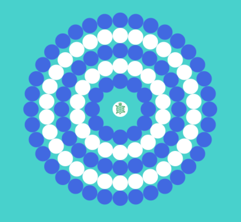
You should add a **dot** command that draws a circle with a diameter of 380 behind the others. You can choose any colour that you would like, although we recommend that you use the same colour as your background. That command should be at the top of your script, as highlighted by the green rectangle in the image below.



Next, at the end of the scripts you should add another **dot** command, which will draw the dot in the centre of the pattern. For our pattern, we used the block pictured below.



After adding these two **dot** commands, the resulting pattern will look similar to the image below (note that there is now a white dot under the Turtle).



# Creating a Function to Draw a Pattern

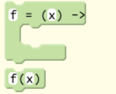
In the next steps, you will learn about **Functions**. **Functions** can be used to “encapsulate” commands, combining multiple instructions into a single command. For example, after completing the next steps you will have a block called **draw\_pattern,** which is pictured below. When you use the **draw\_pattern** block, the Turtle will draw a pattern similar to the one shown above.



If you have coded in *Scratch* before, you may have created a *Custom Block* with the *Make a block* button. *Custom Blocks* in *Scratch* are similar to **Functions** in *Pencil Code* and other languages.

# Functions in Pencil Code

**Function** blocks can be found in the *Operators* section of the *Blocks* view of *Pencil Code*. The blocks for creating and using **functions** from the *Operators* section are shown in the image below.



The block at the top is called a **function definition**. That block will tell the Turtle what to do when it follows that **function** (which is named *f* in the block above).

The block at the bottom Is called a **function call**. That block will tell the Turtle to follow the instructions in the *f* block.

In the below image, we have labelled the different parts of the **definition** and **call** blocks.



* Function Name:this is the name given to the **function**.When the Turtle sees this name (in a **function call** command), it will follow all of the instructions in the **Function Body.** You can change the text in that block to change the name of the block. Usually, it is ideal to name the **function** so that it is clear what it does. For example, you will call the **function** that you create that draws a circular pattern *draw\_pattern*.
* Input: **functions** can have *inputs,* which will have an effect on the instructions that the Turtle follows when the **function** is called. When the Turtle follows the instructions in the **function body**, when it sees *x* in the function body, that will be replaced by the value in the **function call**. For example, for *f(5)*, x will be replaced by 5 and the Turtle will move forward 5 times and draw 5 dots. If the **function call** is *f(10)*, the Turtle will move forward 10 times and draw 10 dots.
* Function Body**:** the **function body** contains all of the instructions that the Turtle will follow. When the **function** (*f* in the image above) Is called, these instructions will be followed by the Turtle.
* Function Calls: these are the commands that will make the Turtle follow the **function** (the *f* **function** in the image above).

# The Draw Pattern Function

In the next steps, you will create a **function** that will combine all of the instructions for drawing the pattern of 5 dotted circles into one command.

The first step is to drag the **function definition** block from the *Operators* section towards the top of your Turtle’s scripts, as shown by the green arrow in the image below.



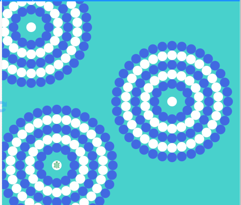
Next, give the **function** a name by changing the *f* to *draw\_pattern* in the spot highlighted by the green rectangle in the image above. Note that there are no *inputs* for the *draw\_pattern* **function**. You can remove the input by deleting the *x* from the spot that is highlighted by the pink rectangle in the image above.

The **function body** of the *draw\_pattern* **function** should contain all of the blocks that draw the pattern of 5 dotted circles. If you are in the *Blocks* view, you can drag all of the blocks inside the **function**, as illustrated in the image below. Once you have done that and run the program, you will notice that the Turtle doesn’t draw anything. You will have to add some **function calls** to make the Turtle draw the pattern again.



# Drawing the Pattern in Different Spots

Now that you have created a *draw\_painting* **function**, you can use this to draw the pattern of dotted circles in different spots on the *Canvas*. In the screenshot below, you can see the result of using our **draw\_pattern** and the **moveto** blocks in different positions on the *Canvas*.



To complete this activity, use the **draw\_pattern** and the **moveto** blocks to make the Turtle draw patterns similar to those above.An example of a stack of blocks that will draw the pattern in different spots on the *Canvas* is shown in the image below.



# Extension Ideas

Good work! You have completed the **Creating Dot Paintings with Pencil Code** activity. If you finish this activity before the end of the *From Blocks to Text* session, you can use some of these ideas to extend the scripts that you created in today’s session.

* Idea 1: You could make the circles of dots different colours or even randomly coloured
* Idea 2: You could add more patterns to the scripts, so that the *Canvas* is covered in circles of dots
* Idea 3: You could create a **function** called *draw\_circle*, which would have the following inputs: *repeat*, *steps* and *degrees*, and would draw a circle of dots for those inputs. Then, you could use that **function** to simply the Turtle’s scripts.
* Idea 4: You could investigate the use of **variables** in *Pencil Code* and experiment with ways of using them for the drawing of circular patterns.