ANSWERS

F8.7 Images

Data representation · GCSE · OCR J277 1.2.4, AQA 8525 3.3.6, Edexcel 1CP2 2.2.2 · about 15 min

BugBotLab

What this lesson is about

Pixels, resolution, colour depth, metadata and file size, from the robot's camera.

- A picture is a grid of numbers
- Seeing it in the console
- Resolution and colour depth
- File size
- Metadata

Questions

5 marks in all

1. An image is 32 × 24 pixels with a colour depth of 8 bits. How many bits is it? [1 mark]

> **Answer:** 6144. 32 × 24 × 8.

2. How many colours can a pixel with a colour depth of 4 bits have? [1 mark]

> **Answer:** 16. 2 to the power 4.

3. Doubling both the width and height of an image does what to its file size? [1 mark]

- ☐ **A** Multiplies it by 4
- ☐ **B** Doubles it
- ☐ **C** Leaves it the same
- ☐ **D** Halves it

> **Answer:** A. There are twice as many pixels across and down: four times as many pixels.

4. What is metadata in an image file? [1 mark]

- ☐ **A** Data about the image, such as its width, height and colour depth
- ☐ **B** The brightest pixels
- ☐ **C** The compressed pixels
- ☐ **D** A copy of the image

> **Answer:** A. Without the width, the pixels could not be arranged back into rows.

5. What is the effect of increasing colour depth? [1 mark]

- ☐ **A** More colours and a larger file
- ☐ **B** Fewer pixels
- ☐ **C** A smaller file
- ☐ **D** Lower resolution

> **Answer:** A. More bits per pixel means more possible colours and more bits to store.

The task: camera pixels

Capture `camera_image(32, 24)`. Print it as 24 lines of 32 characters, `#` for a pixel whose average of red, green and blue is below 128, `.` otherwise. Then print `1-bit size: <bits>` and `24-bit size: <bits>` for a 32 × 24 image, worked out with a calculation.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

img = camera_image(32, 24)
```

The hint students can ask for: For each row, for r, g, b in row: add '#' if (r + g + b) / 3 < 128 else '.'. Size in bits is width * height * depth.

A solution

```
from bugbot import *
connect()
img = camera_image(32, 24)
picture = []
for row in img:
    line = ""
    for r, g, b in row:
        line = line + ("#" if (r + g + b) / 3 < 128 else ".")
    picture.append(line)
for line in picture:
    print(line)
print('1-bit size:', 32 * 24 * 1, 'bits')
print('24-bit size:', 32 * 24 * 24, 'bits')
```

Any program that meets the task's checks is marked correct in the simulator; this is one way, not the only way.