



A7.7 Analogue, digital and graphics

Data representation · A level · AQA 7517 4.5.6.1, Eduqas A500QS 2.3 ·
about 30 min

Name _____

Class _____

Date _____

What this lesson is about

Analogue and digital signals, ADCs and DACs, bitmapped and vector graphics and when to use each.

- Bit patterns
- Analogue and digital
- ADC and DAC
- Bitmapped graphics
- Vector graphics
- Vector or bitmap?

Questions

6 marks in all

1. How many bytes does an 800 x 600 bitmap with a colour depth of 24 bits take, ignoring metadata? [1 mark]

2. How many different levels can a 10-bit ADC output? [1 mark]

3. A company logo must look sharp on a business card and on a large banner. Which is better, and why? [1 mark]

- ☐ A A vector graphic, because the shapes are recalculated at any size with no loss
- ☐ B A bitmap, because it stores every pixel
- ☐ C A bitmap, because it has a colour depth
- ☐ D A vector graphic, because it is always a smaller file

4. Which of these would a vector graphic store for a circle? [1 mark]

Tick every answer that is true.

- ☐ A The centre coordinates
- ☐ B The radius
- ☐ C The fill colour
- ☐ D The colour of every pixel inside it

5. Where would a DAC be used? [1 mark]

- ☐ A Between a computer's digital audio and a loudspeaker
- ☐ B Between a microphone and a computer
- ☐ C Between a light sensor and a microcontroller
- ☐ D Inside a hard disk to store bits

6. The byte 01000001 is stored in memory. What does it represent? [1 mark]

- ☐ A It depends on how the program interprets it
- ☐ B The letter A
- ☐ C The number 65
- ☐ D A machine code instruction

The task: vector to bitmap

`shapes` is a vector drawing of two objects. A rectangle has `x` and `y` (its top-left corner) and `w` and `h`; a circle has a centre `cx`, `cy` and a radius `r`. All are in pixels. Rasterise it onto a bitmap `WIDTH` 32 pixels wide and `HEIGHT` 16 pixels tall, 1 bit per pixel. 1. The pixel in column `col` (0 to 31) and row `row` (0 to 15) has its centre at `(col + 0.5, row + 0.5)`. It is on when that centre is inside any shape: for a rectangle, `x <= px < x + w` and `y <= py < y + h`; for a circle, the squared distance from the centre is no more than `r` squared. 2. Print the 16 rows, top row first, each as 32 characters: `#` for on and `.` for off. 3. Then print, calculating each number from `WIDTH` and `HEIGHT`: 1-bit bitmap: `<n>` bits, 24-bit bitmap: `<n>` bits, and 1-bit bitmap at 4 times the size: `<n>` bits (4 times the width and 4 times the height, still 1 bit per pixel).

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

WIDTH = 32
HEIGHT = 16
shapes = [
    {"type": "rect", "x": 2, "y": 2, "w": 10, "h": 6},
    {"type": "circle", "cx": 22, "cy": 8, "r": 5},
]

def inside(shape, px, py):
    return False
```

Plan your program here, then type it in and press Run.



Do it on the robot

www.bugbotlab.com/learn/a7-7-analogue-digital-and-graphics/

The simulator checks it and tells you when it passes. Nothing to install, no account.

Challenges

1. Rasterise the drawing at 4 times the size by scaling the shapes, not the bitmap. Compare it with scaling the 32×16 bitmap up.
2. Add a `line` shape with two end points and a thickness.
3. Estimate how many bytes the vector drawing takes as text, and compare it with the 24-bit bitmap.