Data compression

# Years 9-10

In many parts of the world, downloading images from the web can result in expensive data charges and greatly lengthen the time to load a webpage. Software engineers develop algorithms to reduce the size of images and text by looking at patterns in the data. Students will learn how the information in a pixel can be manipulated to change the image, and apply a bitmask filter to an image to remove some information and reduce the memory size of the file. Finally, students will modify the filter to find a balance between quality and small file size. This lesson employs **data representation**, **abstraction**, **pattern recognition**, **data analysis**, and **algorithm design.**

**You may also like to refer to the lesson** [**Seeing the big picture**](https://www.digitaltechnologieshub.edu.au/teachers/lesson-ideas/seeing-the-big-picture) **which complements this lesson.**

## Learning hook: Examples of compressing data using visuals and language (10 minutes)

In this activity, students will use [**data representation**](#_dnnkrn7kdbo5) and [**abstraction**](#_dnnkrn7kdbo5) to share examples of how humans are able to share information with each other as efficiently as possible.

|  |
| --- |
| Students respond to the following prompts:  **Prompt 1: Road signs, like a stop sign, are one example of conveying information visually. What are some other examples of pictures or symbols that are used to convey information?**  **Prompt 2: Abbreviations can be used to refer to something without having to say the entire word or phrase. What are some technical or scientific examples as well as those you might hear your friends say?**   1. After a couple of minutes, ask students to share their answers. 2. Write their responses on the board. 3. Use check marks to indicate and count duplicate responses. |

|  |
| --- |
| **Teaching Tips:**   * If students are struggling to come up with answers to the prompt you could share the following examples: * Visual examples: Road signs, logos, maps * Abbreviations: DNA, 3D, gov (for government), LOL (Internet slang for laughing out loud) |

## Learning input: Smallest amount of data necessary (20 minutes)

In this activity, students will use [**pattern recognition**](#_dnnkrn7kdbo5) and [**data analysis**](#_dnnkrn7kdbo5) to determine what is the smallest amount of data necessary to communicate to another person information such as a word, phrase, shape, or visual scene.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Activity:**   1. Pair students into groups of two. 2. Have one student think of a word or phrase and write down one of the letters from that word, wait a second, then write another letter from that word, omitting some of the letters. All of the letters should be placed in the order of where they belong e.g. If I were thinking of "HELLO WORLD," I might write down "L," then later write down "LL," and eventually it might look like "H LLO W R D.” 3. Have the second student try to guess the word or phrase as quickly as possible. 4. After the second student correctly guesses the word, students should switch roles. 5. After a couple of turns, ask students to try and do the same activity using a shape or a visual scene like "student walking to school," where each student draws the shape or visual one line or curve at a time.   **Q1: On average how much information (turns/steps) from the first person was necessary to guess**   |  |  |  |  | | --- | --- | --- | --- | | **Word** | **Phrase** | **Shape** | **Visual** | |  |  |  |  |   **Q2: Why do you think it required more information to guess a visual rather than a word?**  **Q3: What could have made each of these easier to guess, so that it would require fewer steps to solve?** |

|  |
| --- |
| **Assessment**:  **A1:** Answers will vary. On average, a word will require the fewest steps and a visual will require the most.  **A2:** Answers will vary. Responses could include that a visual is more complex than a word, and there is often more information in a visual (many lines) than a word (a few letters).  **A3:** Answers may include:   * Using placeholders for text to show how many letters were in the word or phrase.   + For example \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ for HELLO WORLD indicates how many letters are in a word and when letters are added it is easier to tell where they fit into the word e.g. H \_ LLO W \_ R \_ D * Color to help with identifying objects if a visual. |

## Learning construction: Representing Images with Pixels and Color (20 minutes)

In this activity, students will apply [**data representation**](#_dnnkrn7kdbo5) to convey visual information using binary.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Activity:**  Walk students through the following:   1. In the last activity, you were able to send a word or image to another student by writing letters and lines. How might you convey this information to a computer which has no idea what a letter or a line is? 2. A computer's screen is an array of points that can be turned on or off, also referred to as pixels. In the earliest versions of a computer screen there was only one color for outputting text.   **Q1: If you had a 5 x 5 grid that represented which pixels were on and off on your computer screen, which of the following boxes would you color in to tell the computer how to display a square?**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | 1 | 2 | 3 | 4 | 5 | | 6 | 7 | 8 | 9 | 10 | | 11 | 12 | 13 | 14 | 15 | | 16 | 17 | 18 | 19 | 20 | | 21 | 22 | 23 | 24 | 25 |  1. Computers transmit this information via electrical signals from the computer processor to the screen using binary numbers. The bi- prefix means there are only two options, zero (0) and one (1). Our decimal system has ten options for representing numbers, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.   **Q2: Tell the computer how to draw the box by indicating which pixels should be on and which should be off using 0 for off and 1 for on. Follow the numbering of the pixels from 1-25. For example if the first, second, and third boxes should be on and the fourth should be off you would write 1110.**   1. In groups of two, one person can draw a picture by shading boxes on graph paper while the other person translates this into binary 1s and 0s. Find another group to share your binary number with and see if they can recreate your drawing.Y1UncS 2. You were able to represent a square using only 25 numbers which the computer would store in its memory. Each of these numbers would take up a "bit" of memory. 8 bits = 1 Byte and 1000000 Bytes = 1 Megabyte which is approximately what is used to represent an image like the image on the right (<http://goo.gl/Y1UncS>). 3. Not all of those pixels are black, so some of the memory used to store this image on a computer will need to be used to store the color of a pixel. 4. One way of representing color is as a combination of the traditional primary colors in additive color mixing (<https://en.wikipedia.org/wiki/Additive_color>), **Red**, **Green**, and **Blue**. 5. Binary can represent a color by allocating some of the bits to each color. The amount of bits used will determine how many possible colors there are. 6. 8-16 bits were allocated for color in older video game consoles. Today's computer screens use 24 bits for color. By changing the values of the bits it is possible to mix the colors into every color we see. Here are some examples:  |  |  | | --- | --- | |  | **Red Green Blue** | | **Red** | **1111 1111 0000 0000 0000 0000** | | **Green** | **0000 0000 1111 1111 0000 0000** | | **Blue** | **0000 0000 0000 0000 1111 1111** |   **Q3: With 24 bits how many possible color options are available?** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Assessment**:  **A1:** A couple of options below.   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **1.**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | 1 | 2 | 3 | 4 | 5 | | 6 | 7 | 8 | 9 | 10 | | 11 | 12 | 13 | 14 | 15 | | 16 | 17 | 18 | 19 | 20 | | 21 | 22 | 23 | 24 | 25 | | **2.**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | 1 | 2 | 3 | 4 | 5 | | 6 | 7 | 8 | 9 | 10 | | 11 | 12 | 13 | 14 | 15 | | 16 | 17 | 18 | 19 | 20 | | 21 | 22 | 23 | 24 | 25 | |   **A2: A couple of options below**   1. 0 0 0 0 0 0 1 1 1 0 0 1 0 1 0 0 1 1 1 0 0 0 0 0 0 2. 1 1 1 1 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1 1 1 1 1 1   **A3:** Each bit has two possibilities, 0 and 1. Computers at a fundamental level use the binary system for representing bits. Binary is a base-2 number system and with 24 bits the possible colors are 224, or 16,777,216 colors. The human eye is unable to distinguish between all of these possible colors. To read more about the RGB relationship of additive color mixing, and the exponential relationship of 224, see <https://en.wikipedia.org/wiki/RGB_color_model>. |

## 

## Learning demo 1: Understanding bitmasks (20 minutes)

In this activity, students will apply [**pattern recognition**](#_dnnkrn7kdbo5) and [**abstraction**](#_dnnkrn7kdbo5) to understand how bitmasks can be used to add or remove information from a pixel.

|  |
| --- |
| **Notes to the Teacher:**  Students should have a basic understanding of the binary and hexadecimal number systems before using this activity. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Activity:**  Walk students through the following:   1. You may have used an application on your phone to give an image a different look, perhaps sepia, or black and white. This used to be accomplished through filters attached to the phone or the development process used in processing the film. For digital photos these effects can be created by manipulating the bits in a pixel. 2. There are many ways to modify the bits, for this activity a bitmask will be applied to the image. A bitmask is a term in computer science where one applies a logical test to bits. This example adds green to a pixel:  |  |  | | --- | --- | |  | **Red Green Blue** | | **Red** | 1111 1111 0000 0000 0000 0000 | | **Bitmask** | **OR** 0000 0000 1111 1111 0000 0000 | | **Yellow** | 1111 1111 1111 1111 0000 0000 |  1. In the example above, the red pixel became a yellow pixel when the OR bitmask was applied. OR is a logical test where the result is True or 1 if either of the bits are 1. So 1 OR 1 is 1, 1 OR 0 is 1, and 0 OR 0 is 0. OR bitmasks are used to turn on/add bits.   240px-AdditiveColor.svg.png  **Q1: Using the additive color mixing diagram on the right from Wikipedia, what do you predict will be the resulting binary and color of applying a bitmask of blue to red?**   1. Binary is useful for a computer, but it's easier for us to write 1234 in decimal form rather than writing 10011010010 in binary. For websites you may notice that hexadecimal is used for color.    1. Hexadecimal is similar to the decimal system you are used to, but instead of 10 digits, 0-9, hexadecimal has 16 digits, which includes 0-9 as well as A,B,C,D,E,F. Using this system, 12 becomes C and 1234 in decimal becomes 4D2 in hexadecimal.       1. If you are unfamiliar with hexadecimal, it may be helpful to see how 4D2 is translated back into decimal to understand the notation.   4D216 = (**4** \* 162) + (**13** \* 161) + (**2** \* 160) = (**4** \* 256) + (**13** \* 16) + (**2** \* 1) = 123410  Note: 13 is used to represent D in the calculation because D is the 13th digit in the hexadecimal system (0-9,A,B,C,D)   1. To represent Red in hexadecimal, convert 111111110000000000000000 into FF0000. You can verify this on the Wikipedia entry for red (<https://en.wikipedia.org/wiki/Red>). 2. Another way to manipulate the bits in a pixel is the AND bitmask. Unlike the result of an OR bitmask, which is True or 1 if either of the bits are 1, the result of an AND bitmask is only True if both bits are 1. Modifying the example from earlier:  |  |  | | --- | --- | |  | **Red Green Blue** | | **Yellow** | 1111 1111 1111 1111 0000 0000 | | **Bitmask** | **AND** 0000 0000 1111 1111 0000 0000 | | **Green** | 0000 0000 1111 1111 0000 0000 |  1. This bitmask filters out every color but green.   **Q2: What would be the resulting binary and color if the same AND bitmask was applied to Red?**   1. By using different bitmask operations and different bit combinations it is possible to add or remove information from the pixels. |

|  |
| --- |
| **Teaching Tips:** Screen Shot 2015-10-20 at 8.51.57 AM.png   * If students have not worked with different number systems like binary and hexadecimal before, you could search online for [1234 to hexadecimal], [binary to hexadecimal converter], or search for [yellow in hexadecimal]. * Students can explore the hexadecimal values for colors in Chrome and other Internet browsers by right clicking on a color inside a webpage and selecting "Inspect Element". The developer console shown will allow them to see what color is used and modify it if they choose. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Assessment**:  **A1:** 1111 1111 0000 0000 1111 1111 Purple   |  |  | | --- | --- | |  | **Red Green Blue** | | **Red** | 1111 1111 0000 0000 0000 0000 | | **Bitmask** | **OR** 0000 0000 0000 0000 1111 1111 | | **Purple** | 1111 1111 0000 0000 1111 1111 |   **A2:** 0000 0000 0000 0000 0000 0000 White   |  |  | | --- | --- | |  | **Red Green Blue** | | **Red** | 1111 1111 0000 0000 0000 0000 | | **Bitmask** | **AND** 0000 0000 1111 1111 0000 0000 | | **White** | 0000 0000 0000 0000 0000 0000 | |

## Learning demo 2: Applying bitmasks to an image (30 minutes)

**Activity Overview:** In this activity, students will use [**abstraction**](#_dnnkrn7kdbo5) to implement a bitmask in [**algorithm**](#_dnnkrn7kdbo5) to modify an image.

|  |
| --- |
| **Notes to the Teacher:**  The Pencil Code program used in this activity has two primary points where students can easily modify the code:   1. Students can replace the link to the image used. It must be hosted from a publically available website. 2. This is where the bitmask is applied and the focus of this activity. Students can modify the bitmask itself 0xFFFFFFFF as well as the operator, changing it from & (and) to | (or) |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Activity:**  In the previous activity the idea of a bitmask was introduced and how it could modify the bits in a color. In this activity students will modify an image using a bitmask. Walk students through the following:   1. Open the Pencil Code [Data Compression](https://example.pencilcode.net/edit/data-compression#blocks=0) program. 2. This program has three main parts:    1. Draw the pixels of the image on the screen (lines 5-8)    2. Draw a second image on the screen to be modified by the bitmask (lines 12-22)    3. Apply the bitmask (lines 25-26) 3. By default on line 26, the bitmask is set to apply & 0xFFFFFFFF.   & → refers to the bitwise AND operation  0x → indicates that the number is a hexadecimal  FFFFFFFF → is the bitmask.  The first two digits set the picture opacity from 00 = invisible to FF = 100% visible.  The next 6 digits set the color. FFFFFFFF will keep the image as it is.   1. Have students try the following: change the bitmask on line 26 from 0xFFFFFFFF to the values below and press the play button to run the program again.button.png    1. 0xFFFFFF00 (remove blue from all pixels)    2. 0xFFFF88FF (remove 50% of red from all pixels)    3. 0xAAFFFFFF (make the image 40% less opaque)    4. Change & to | and try various combinations of the bitmask to see the impact of OR vs AND operation. 2. Applying the bitmask causes the image to change its appearance. If students look at the bytes used, the amount of information stored in the image is also changing. It is possible to reduce or compress the amount of data used without dramatically changing the appearance of the image.   **Q1:** **Complete the following chart using the & bitmask. Add your opinion of the quality of the new image compared to the original.**   |  |  |  | | --- | --- | --- | | **Bitmask value (base16)** | **Memory Size (bytes)** | **Quality (1-10)** | | **0xFFFFFFFF** | **142845 (original size)** | **10** | | **0xFF000000** | **1389** | **1** | | **0xFFAAAAAA** |  |  | | **0xFFCCCCCC** |  |  | | **0xFFEEEEEE** |  |  | | **0xFFA0A0A0** |  |  | | **0xFFC0C0C0** |  |  | | **0xFFF0F0F0** |  |  | |

|  |
| --- |
| **Notes to the Teacher:**  The primary concern of previous versions of smartphones was the amount of storage available. Now, with so much information coming from the Internet, developers are constantly thinking of how they can reduce the size of the data as small as possible to reduce expensive network charges for users.  Discuss with students that if they were developing software for sending photos to friends versus software for photo editing, they might be willing to sacrifice a little bit of quality in order to make the photos load more quickly. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Assessment**:  A1: Answers are based on the default image (<http://goo.gl/Y1UncS>).   |  |  |  | | --- | --- | --- | | **Bitmask value (base16)** | **Memory Size (bytes)** | **Quality (1-10)** | | **0xFFFFFFFF** | **142845 (original size)** | **10** | | **0xFF000000** | **1389** | **1** | | **0xFFAAAAAA** | **113200** |  | | **0xFFCCCCCC** | **92758** |  | | **0xFFEEEEEE** | **124374** |  | | **0xFFA0A0A0** | **50779** |  | | **0xFFC0C0C0** | **35283** |  | | **0xFFF0F0F0** | **64899** |  | |

## Wrap-up activity: Developing ideas for compressing data (10 minutes)

Ask students what understanding they have gained through this lesson about compressing data.

**Ask students, What additional ideas do you have for compressing an image?**

|  |
| --- |
| **Assessment**:  Ideas might include:   * Removing pixels from the image * Merging similar adjacent pixels * Reducing the size of the image |

# Additional Information and Resources

## Curriculum Links

| Links with Digital Technologies Curriculum Area | | |
| --- | --- | --- |
| **Year** | **Strand** | **Content Description** |
| **7-8** | **Knowledge and Understanding** | Explain how and why digital systems represent integers in binary [(AC9TDI8K04)](https://v9.australiancurriculum.edu.au/f-10-curriculum.html/learning-areas/digital-technologies/year-7_year-8/content-description?subject-identifier=TECTDIY78&content-description-code=AC9TDI8K04&detailed-content-descriptions=0&hide-ccp=0&hide-gc=0&side-by-side=1&strands-start-index=0&subjects-start-index=0&view=quick). |
| **9-10** | **Knowledge and Understanding** | Investigate simple data compression techniques [(AC9TDI10K03)](https://v9.australiancurriculum.edu.au/f-10-curriculum.html/learning-areas/digital-technologies/year-9_year-10/content-description?subject-identifier=TECTDIY910&content-description-code=AC9TDI10K03&detailed-content-descriptions=0&hide-ccp=0&hide-gc=0&side-by-side=1&strands-start-index=0&subjects-start-index=0&view=quick). |

## Lesson Vocabulary

|  |  |  |
| --- | --- | --- |
| **Term** | **Definition** | **For Additional Information** |
| **Binary** | A number system with two possible digits 0 and 1. So the number 30 in decimal is written as 11110 (1 \* 25 + 1\* 24 + 1\* 23 + 1\* 22 + 1\* 21 + 0\* 20) | <https://en.wikipedia.org/wiki/Binary_number> |
| **Bitmask** | Applies a bitwise operation to multiple bits at once. | <https://en.wikipedia.org/wiki/Mask_(computing)> |
| **Bit** | A **b**inary dig**it**, the smallest representation of information in a computer or digital communications. | <https://en.wikipedia.org/wiki/Bit> |
| **Bitwise operation** | Manipulates a bit by comparing it to another bit (e.g. 1 AND 0 = 0 but 1 OR 0 = 1) | <https://en.wikipedia.org/wiki/Bitwise_operation> |
| **Hexadecimal** | A number system with 16 possible digits 0-F (0-9, A, B, C, D, E, F). So the number 30 in decimal is written as 1E (1 \* 161 + 14 \* 160) | <https://en.wikipedia.org/wiki/Hexadecimal> |
| **Pixel** | The smallest controllable physical part of a computer screen. | <https://en.wikipedia.org/wiki/Pixel> |

## Computational Thinking concepts

|  |  |
| --- | --- |
| **Concept** | **Definition** |
| **Abstraction** | Identifying and extracting relevant information to define main idea(s) |
| **Algorithm Design** | Creating an ordered series of instructions for solving similar problems or for doing a task |
| **Data Analysis** | Making sense of data by finding patterns or developing insights |
| **Data Representation** | Depicting and organizing data in appropriate graphs, charts, words or images |
| **Pattern Recognition** | Observing patterns, trends, and regularities in data |

## Additional Resource Links

* CS Unplugged - Image Representation (<http://csunplugged.org/image-representation/>)
* CS Unplugged - Text Compression (<http://csunplugged.org/text-compression/>)
* Code.org Unit 1 (<https://code.org/educate/csp/unit1>)
  + Number systems
  + Binary Numbers
  + Bytes and file sizes
  + Text Compression
  + Encoding images
* Visit <http://pencilcode.net/> to explore the Pencil Code development environment
* See Pencil Code: A Programming Primer (<http://book.pencilcode.net/>) for more than 100 example programs written in CoffeeScript (<http://coffeescript.org/>)

## Extension activities

* studio.code.org Text Compression (<https://studio.code.org/s/allthethings/stage/16/puzzle/1>)

## Administrative Details

|  |  |
| --- | --- |
| **Contact info** | For more info about Exploring Computational Thinking (ECT), visit the ECT website ([g.co/exploringCT](http://g.co/exploringCT)) |
| **Credits** | Developed by the Exploring Computational Thinking team at Google and reviewed by K-12 educators from around the world. |
| **Last updated on** | 22/01/2020 |
| **Copyright info** | Except as otherwise [noted](http://www.google.com/edu/policies.html#restrictions), the content of this page is licensed under the [Creative Commons Attribution 3.0 License](http://creativecommons.org/licenses/by/3.0/), and code samples are licensed under the [Apache 2.0 License](http://www.apache.org/licenses/LICENSE-2.0). |

Nat can we add as a pop up?

Add this code into pencilcode.net

speed Infinity

picture = picture = 'http://goo.gl/Y1UncS' # from Wikimedia

# First sprite

c1 = new Sprite

c1.fd 150

c1.wear picture

await c1.done defer() # Load image

# Get raw image data bits and convert data into

# 32-bit integer array.

d = c1.imagedata()

source = new Uint32Array(d.data.buffer)

# Second Sprite, empty to start.

c2 = new Sprite # fold

width: d.width

height: d.height

c2.bk 150

n = c2.imagedata()

target = new Uint32Array(n.data.buffer)

# Apply the bitmask.

for j in [0...source.length]

target[j] = source[j] & 0xFFFFFF00

# Set the altered imagedata to sprite 2.

c2.imagedata(n)

# Finally we encode each image using PNG

# compression and then label each image

# with its number of bytes as a PNG file.

labelBytes = (c) -> # fold

b64 = c.canvas().toDataURL().replace /^.\*,/, ''

bytecount = atob(b64).length

c.label bytecount + ' bytes',

font: 'bold 32px Arial', color: yellow

textShadow: '0 0 7px black'

labelBytes c1

labelBytes c2

ht()