National University of Singapore
School of Continuing and Lifelong Education

TIC1001: Introduction to Computing and Programming I
Semester I, 2019/2020

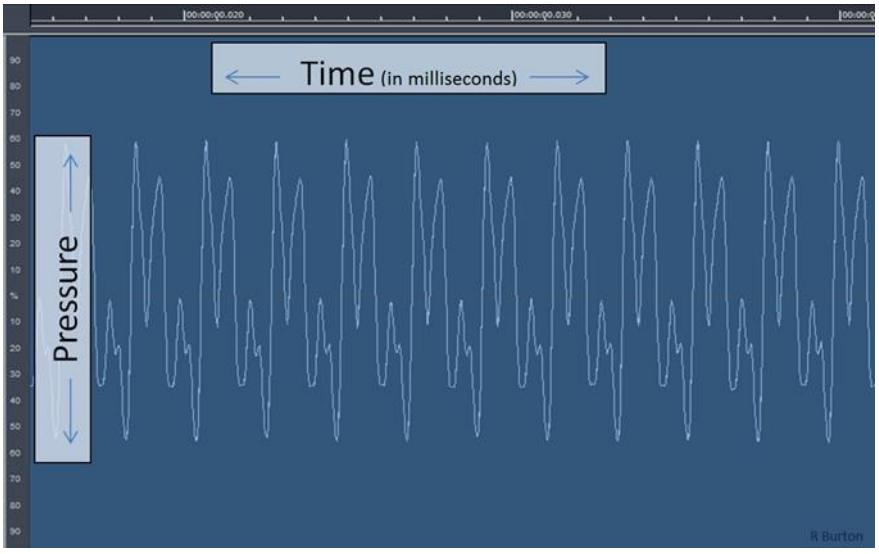
Tutorial 4
Compilation and Data

Note: The Coursemology **tutorial training** for **computing topics** usually consists of follow-up questions on the tutorial discussion. So, you should attempt the training only after trying out the tutorial questions and / or after attending the tutorial session.

We will use this tutorial to explore interesting idea in computing. The context may not be directly discussed in the lecture, but the principles and ideas can definitely be applied. DonâĂŹt feel daunted by the unfamiliar context

1. We mentioned that information is stored as just 0s and 1s on computer. For this question, we will explore ways to represent sound data like music, voice etc.

   Firstly, let us look at an example sound wave:

   We can see that the waveform is continuous in both time and pressure domains. **Discretization**, i.e., breaking up the continuous information into discrete (separate) chunk is one possible way to represent such analog information on computer.

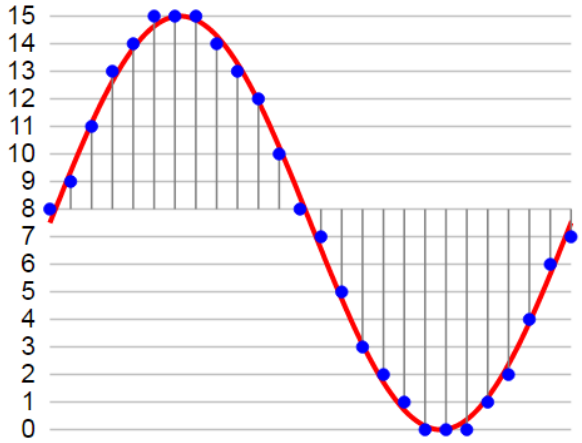
   One of the simplest discretization approaches is known as Pulse-Code Modulation (PCM):

   (a) Firstly, we break up the time domain by taking the pressure value only at certain **time intervals**. This process is known as **sampling**. For example, if we read 10 sound pressure values per second (i.e. every 100 milliseconds), then we have a **sampling rate** of **10 Hz**.

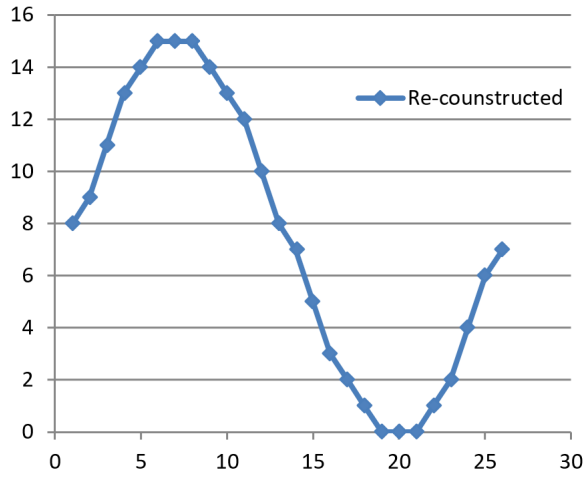
   (b) The pressure value is a continuous range too. We can round the decimal point values to a number of discrete values. This is known as **quantization** process. The number of discrete values (quantization levels) directly determines the amount of bits needed to represent each pressure value. For example, if we use only 16 values

(0–15), then each pressure value can be presented as a 4-bit value ($2^4 = 16$). We call this a **4-bit quantization**.

Below is an illustration of the PCM process on a waveform:



The vertical lines are the sampling time intervals and the blue dots are the digitized value of the wave form at those intervals. So, the above "curve" are quantized as a series of integer values: $8, 9, 11, 13, 14, 15, 15, 15, 14, \ldots$. If we use these discretized values to re-construct the curve, it looks like:



You can see that it looks pretty close to the original continuous wave! Due to its simplicity, the PCM approach is used to encode sound data and used in Audio-CD, .WAV file, signal from some of the radio stations, etc.

Let us now answer a few related questions:

(a) Is higher sampling rate desirable? Why? What is the drawback of high sampling rate?

(b) Similarly, is larger number quantization desirable? Why? What is the draw back?

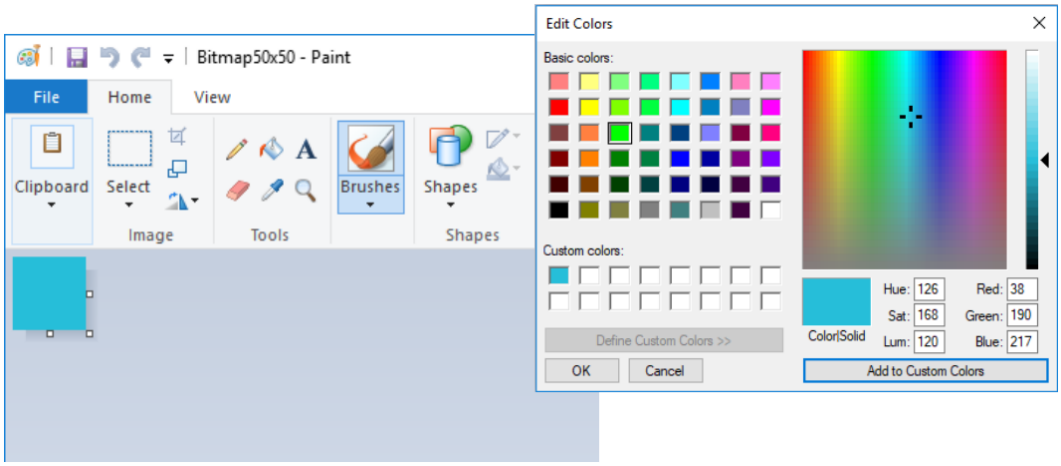
(c) Suppose we use **8 kHz (8,000 Hz) sampling rate** with **4-bit quantization**, what is the storage size needed for a **1-minute sound data**?

(d) Audio CD has two sound channels (stereo) where the left and right sound channel are encoded separately. A high-end audio CD usually uses 44.1 kHz sampling rate and 16-bit quantization levels. What is the storage size required for a **1-minute sound data** in this case?

(e) MP3 is much more commonly used nowadays as compared to .WAV file. Take a look at one of your own .mp3 file and estimate the storage size required for a 1-minute sound data. How does it compared with (d)?

2. In the lecture, we discussed the idea of **encoding**, which deals with representation of data using just 0s and 1s. For this question, we will take a look at two type of data that you may be familiar with: i) a picture (in BMP format) and ii) a text file.

   First, open up a simple image editor on your system (e.g. paint on Windows). Create a new file with height of 50 pixels and width of 50 pixels, then fill the 50 x 50 pixels with a special color where the RGB values are **R=38, G=190 and B=217**. Below is an example of how you can do it in paint (find out how to do it on your platform if you are not on windows):



   Save the file as **`24-bit BMP`** file which stores each pixel as a RGB value and each of the R, G and B values using 8 bits (i.e. 1 byte).

   (a) What do you think is the file size of this BMP file? Verify the file size by using file explorer or similar.

   (b) Use a hex editor, which allows you to see the byte values stored in a file and open the BMP file from (a). You may use this online tool: `https://www.onlinehexeditor.com/`.

   Can you find the pixel color R=38, G=190 and B=217 in the file? (Recall that each of the 2,500 pixels is saved individually in a BMP file) (hint: you need number base knowledge)

   (c) Let us now try to understand the different between a text file and a BMP file.

   Open a new text file, insert 2,500 copies of the RGB value you found in (b).

   Hint: there is no need to copy-and-paste 2,500 times. You can first make 10 copies of the values, copy that 5 times giving you 50 copies, copy these 50 copies 10 times to get 5000 copies, then copy these 500 copies 5 times.

   Save the file.

   What do you think is the file size? Verify your guess using file explorer. Can you explain the difference you see between this and (a)?