**Core Reflection**

**Reason for selected sampling rate.**

The sampling rate I used is 44100 due to the Nyquist Theorem. This shows that a periodic signal must be sampled at least twice at the highest frequency of the signal. To have a high frequency in digital form you need to sample twice as fast as the high frequency on the signal. The human ear can hear acoustic signals up to 20,000 Hz which is fundamentally the highest frequency component. This is the reason why I have doubled it as this is the normal sampling rate in music also.

**How you calculated total number of samples**

Total number of samples *p.s* x number of seconds the wave runs for.

**Reasoning for selection of the types of all variables**

* Duration I put at 5 seconds. It is the core of the assignment, so I didn’t produce a long sample.
* Sample Rate I put at 44100 since it is the normal sampling rate for music.
* Int – “n\_samples” – number of samples is an integer.
* Double – “dt” – dt is a fractional number (very small)
* Dt – taken sound window of 1 second and divided it by number of samples (44100)
* Frequency is 6000hz. Comfortable frequency for hearing
* Volume is 5000hz. Easy volume to hear sound
* Waveform – utilizes air pressure as a function for the time equation. Exchanged suitable values into formula.

**Screenshot of waveform, does it look like what you intended?**

**Chart

Description automatically generated**

It looks like what I intended. I did not vary frequency within the code. This has shown to construct a vibrations on a repeating wave.

Down for Completion Reflection

**Completion Reflection**

Frequency remains constant depending on the amount of speed it is going at. If the speed were to slow down that is when the wavelength also alternates to fit, it. The high point of the sound wave takes time to adjust after change within the frequency. A good time to change the frequency is at the lower point so the sound and wave can then adjust as the wave increases upwards again.

Table

Description automatically generated with medium confidence

An example of an algorithm:

If((i\_sample>\_\_\_\_\_) &&(i\_sample<\_\_\_\_\_) {frequency: can change here as it reaches lower frequency}

If(i\_sample>\_\_\_\_\_) &&(i\_sample<\_\_\_\_) {frequency: can change here as per prior change in line above}

The code I used for completion is naïve (bunch of ifs). Another way to decide when to change the frequency is by modulo division. This returns the remainder of a division after one number is divided by another. The result is the remainder of integer division of the given numbers.

An example algorithm for the modulo division method:

35/3 = 11, remainder 2 🡪 35 mod 3 = 2

A formula to work this out is [dividend-{(dividend/divisor) \*divisor}]

🡪35 % 3

🡪35- {(35/3) \*3}

🡪35- {2\*3)

🡪=2

Modulo division is effective as we can determine an even result (the remainder) to construct frequencies used in our code.