Assignment #8

CS 245, Spring 2018

Due Thursday, March 29

Compared to addition and multiplication of floating point numbers, the C/C++ function `sin` is relatively expensive. For applications that use additive synthesis or FM synthesis, this can lead to an excessive CPU load. In such cases, we can use a look–up table to make the computations less expensive.

In this assignment, you will implement a class to create and use a look–up table for computing values of the sine function. The interface to this class should be

```
class Sine {
  public:
    Sine(unsigned R);
    float operator()(float x);
  private:
    std::vector<float> sine_table;
    double scale;
};
```

(the standard header file `vector` has been included).

`Sine(R)` — (constructor) creates a look–up table (array) with R samples of one period of the sine function. Specifically, the n–th entry in the table will have the value

$$y_n = \sin\left(\frac{2\pi n}{R}\right)$$

for $0 \leq n < R$. In other words, we sample the sine function at points $t_n = \frac{2\pi n}{R}$. To generate the values in the table efficiently, we can make use of the recurrence relation for values of the sine function that we discussed previously in class:

$$y_0 = 0, \quad y_1 = \sin\left(\frac{2\pi}{R}\right), \quad y_n = \beta y_{n-1} - y_{n-2} \quad \text{where } \beta \doteq 2\cos\left(\frac{2\pi}{R}\right)$$

for $n \geq 2$. Here $y_n \doteq y(t_n)$. *Note:* y_n is computed by accumulating floating point values, so we should use double precision arithmetic. This is a general rule of thumb: double precision variables should be used when computing by accumulation of floating point values.

`operator()(t)` — returns the approximate value of $\sin(t)$, obtained by using the look–up table in conjunction with linear interpolation. Note that since $t_n = \frac{2\pi n}{R}$, the fractional index into the look–up table is

$$x = \frac{Rt}{2\pi}.$$

> Thus we must interpolate values in the table between index k and index $k + 1$, where $k = \lfloor x \rfloor$. However, k is not necessarily in the range $0 \leq k < R$. So that we will need to use modular arithmetic to get an index that is within this range. *Warning:* if the value of k is negative (as happens when $t < 0$), then the C/C++ expression
>
> ```
> k % R
> ```
>
> will result in a negative value.

Your submission for this assignment will consist the implementation file `Sine.cpp`. You may include only the `Sine.h` header file and any standard C++ header file. However, the usage of the `Sine` class must guarantee that only a **single** call to `sin` and a single call to `cos` is ever made; i.e., these functions should not appear inside of any loop, and should not be called by the `operator()` function.