Assignment #2

CS 246, Fall 2018

Due Wednesday, September 19

Task #1: circular buffer

At the heart of the echo filter is a circular buffer, also called a ring buffer. We used a circular buffer before in CS 245 when we implemented an audio output class. However, the functionality that we require here is simpler: we only need to (1) store the last N audio samples, and (2) be able to retrieve the k–th previous sample (where $0 \leq k < N$). We encode this functionality in the class `RingBuffer`, declared in the file `RingBuffer.h` that I will provide.

```
class RingBuffer {
  public:
    RingBuffer(int N);
    void put(float x);
    float get(int k);
  private:
    std::vector<float> buffer;
    int current_index;
};
```

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

`RingBuffer(N)` — (constructor) creates a ring buffer object that stores a maximum of N audio samples. The samples should be initialized to 0.

`put(x)` — stores the value x in the ring buffer at the current index in the ring buffer. Note that you will need to increment the current index, wrapping back around if necessary, *before* you store the value.

`get(k)` — returns the value stored in the ring buffer at offset k; i.e., the value stored at the current index minus k, with possible wrapping. Thus an offset value of 0 retrieves the value at the current index. An offset value of 1 retrieves the value at the current index minus one (wrapped if necessary). Et cetera. The value of k is assumed to be in the range $0 \leq k < N$, where N is the size of the ring buffer. No error checking is performed.

Warning. The `C/C++` mod operator `%` will return a negative number if the dividend is negative. For instance, although mathematically we have $-22 \bmod 6 = 2$, the mod operator yields `-22 % 6 = -4`. In this case, to wrap -22 to the range $0, 1, \ldots, 5$, we need to add the divisor to get the desired result: `-4 + 6 = 2`.

For this task, your submission should consist a single file named `RingBuffer.cpp`. You may include any standard `C/C++` header file, as well as `RingBuffer.h`.

Task #2: echo filter

Simple echoes can be cast in the form of a digital filter. For a single echo, the transfer function would simply be

$$H(z) = 1 + \alpha z^{-k}$$

where α ($0 \leq \alpha \leq 1$) is the mixing factor of the echo, and k gives the index of the delay: for a delay of T seconds, we would take $k = \lfloor RT \rfloor$, where R is the sampling rate. A repeated echo can be obtained by feeding back the output into the input of the filter. This leads to the transfer function

$$H(z) = \frac{1 + (\alpha - \beta)z^{-k}}{1 - \beta z^{-k}} \tag{1}$$

where β ($0 \leq \beta \leq 1$) is the feedback factor. For this part of the assignment you are to implement an echo filter with feedback. I will give you the file `Echo.h` that declares the class

```
class Echo : public Filter {
  public:
    Echo(float Tmax=1, float R=44100.0f);
    void setDelay(float T);
    void setMix(float a);
    void setFeedback(float b);
    float operator()(float x);
  private:
    int max_samples;
    RingBuffer xvalues, yvalues;
    float rate, mix, feedback, offset;
  };
```

The `Filter.h` and `RingBuffer.h` header files have been included. The `Filter` interface class is described at the end of the handout.

`Echo(Tmax,R)` — (constructor) creates an echo filter object. The argument `Tmax` specifies the maximum delay value (in seconds), and the argument `R` is the sampling rate.

`setDelay(T)` — sets the delay of the filter (in seconds). The value of `T` is assumed to be between 0 and `Tmax` seconds, where `Tmax` is the value specified in the constructor (no error checking is performed).

`setMix(alpha)` — sets the mixing factor, the parameter α in equation (1).

`setFeedback(beta)` — sets the feedback factor, the parameter β in equation (1).

`operator(x)` — returns the output of the echo filter, given the input value `x`. Note that to compute the n–th output sample from the recurrence relation corresponding to (1):

$$y_n = x_n + (\alpha - \beta)x_{n-k} + \beta y_{n-k},$$

the values of $x_{n-1}, x_{n-2}, \ldots, x_{n-k}$ as well as $y_{n-1}, y_{n-2}, \ldots, y_{n-k}$ need to be stored. For this reason, you will need to maintain two circular buffers: one for the input values, and one for the output values.

Your submission for this part of the assignment should consists of the implementation file, named `Echo.cpp`. You may include any standard `C/C++` header file, as well as `Echo.h`.

For Full Credit

If you complete the above two tasks as stated, you will receive a *maximum* grade of 92%. To receive full credit (maximum of 100%), you will have to allow for a continuous range of delay times.

In equation (1), the parameter k is assumed to be an integer: it specifies a delay of k *samples*. Indeed, for a given delay time of T seconds, we take $k = \lfloor RT \rfloor$. In effect, this means that the delay time T is forced to be an integer multiple of the sampling period $1/R$. For many applications, this is sufficient. However, there are instances where we would like to allow k to have non–integral values; i.e., we take $k = RT$, without truncating to an integer. To implement this, you will have to use linear interpolation between adjacent samples in the circular buffers for both the input and output values.

Filter interface

For the echo filter and all of the filters that we will implement in the remaining homework assignments, we will derive from the following interface class contained in the file `Filter.h`.

```
class Filter {
  public:
    virtual ~Filter(void) { }
    virtual float operator()(float x) = 0;
};
```

That is, any filter that we implement will be guaranteed to provide an overloaded `()` operator. The initial (0–th) call to this function will return the output $y = y_0$ of the filter at time $t = 0$, given an input value of $x = x_0$; i.e., returns y_0, with x_0 as input. The n–th call to this function will return the output of the filter $y = y_n$ at time $t_n = n/R$, given an input value of $x = x_n$. That is, it returns y_n, with x_n as input. Here R is the sampling rate.