Assignment #6

CS 245, Spring 2018

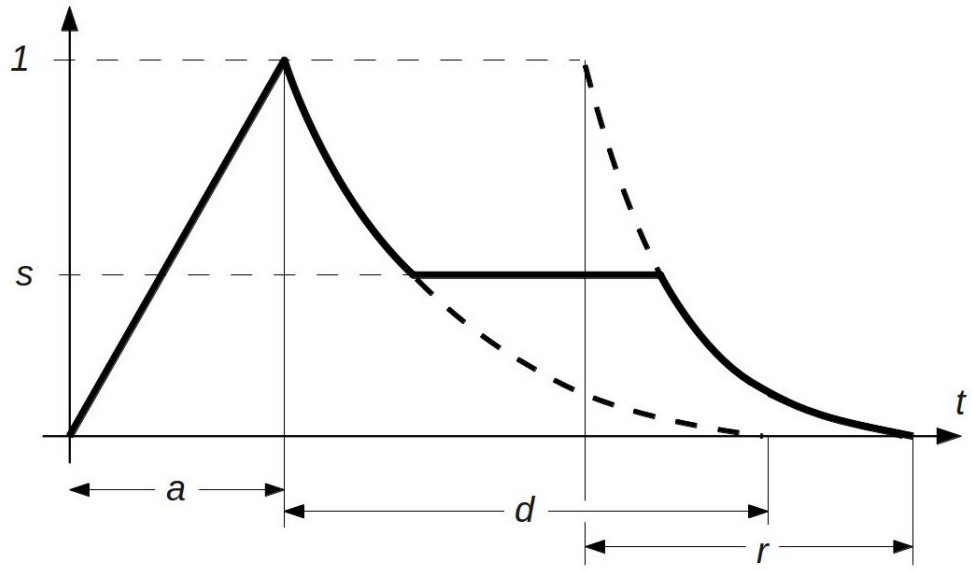
Due Thursday, March 8

DLS–style ADSR envelopes

A DLS–style ADSR envelope uses a linear ramp for the attack phase, but exponential decays for the decay and release phases. According to the DLS specification

`https://www.midi.org/specifications/item/dls-level-1-specification`

four envelope parameters are used: the attack time a, decay time d, sustain level s, and release time r. However, the decay and release times are *not* the same as the durations of the decay and release phases, but rather as the amount of time to go from maximum gain (0 dB) to effective zero (−96 dB). The details for each phase are given below.



Attack phase. This is a linear ramp from 0 gain to maximum amplitude (unity gain) that takes a time a seconds. The amplitude $y_n = y(n/R)$ of the envelope for the attack phase can be computed incrementally:

$$y_{n+1} = y_n + \tfrac{1}{aR}$$

where R is the sampling rate. That is, the amplitude of the envelope in the attack phase increases by the amount $1/aR$ with each successive sample, starting from 0 until unity gain is reached or exceeded.

Decay phase. This is an exponential decay envelope whose initial value is unity gain, and whose final value is the sustain level. The value d that governs the decay phase, is the *total time* in seconds to go from maximum amplitude of 0 dB to -96 dB (the effective zero value for 16 samples). The decay phase ends when the amplitude reaches the value s, the sustain level. The duration of the decay phase of the envelope depends on the exponential decay rate as well as the value s. In particular, if $s = 1$ (unity) gain, there is no decay phase (the decay phase duration is 0 seconds)!

The value of the ADSR envelope in the decay phase can be written as $y(t) = e^{-k_d(t-a)}$, where k_d is the decay rate for the decay phase (a is the attack time, which is also the duration of the attack phase). The value of k_d can be deduced from the decay phase parameter d. As with the attack phase, we can compute the envelope value incrementally, although not additively, but rather multiplicatively:

$$y_{n+1} = e^{-k_d/R}\, y_n \tag{1}$$

I.e., the amplitude of the envelope in the decay phase decreases by a factor of $e^{-k_d/R}$ with each successive sample, starting from unity gain until the envelope reaches the sustain level or below.

Sustain phase. The envelope has a constant value of s until the note is released. As the release of the note is an asynchronous event, the envelope remains in the sustain phase for an indefinite amount of time: the duration of the sustain phase cannot be determined beforehand.

Release phase. Just like the decay phase, this is an exponential decay envelope. At the start of the release phase, the envelope has a value of s. In principle, the release phase never ends, but in practice can be taken to end when the envelope amplitude reaches the effective zero of -96 dB. The parameter r which governs the release phase (and determines the decay rate) is the total time in seconds from *maximum* gain (note that this is <u>not</u> the sustain level) to -96 dB.

The value of the ADSR envelope in the release phase is given by $y(t) = se^{-k_r(t-t_s)}$, where k_r is the decay rate and t_s is the ending time of the sustain phase (i.e., the start time of the release phase). We can compute the amplitude of the ADSR envelope in the release phase incrementally using a formula similar to equation (1), with the envelope starting at s, and decreasing indefinitely.

Your task

In this assignment, you will be implementing a class for DLS–style ADSR envelopes. The class should be declared in the file `ADSR.h`, and have the following public and private interface.

```
class ADSR {
  public:
    ADSR(float a=0, float d=0, float s=1, float r=0, float R=44100);
    void sustainOff(void);
    float operator()(void);
  private:
    enum { ATTACK=0, DECAY=1, SUSTAIN=2, RELEASE=3 } mode;
    float envelope,
          attack_increment,
          decay_factor,
          sustain_level,
          release_factor;
};
```

The functions in the public interface are described below.

`ADSR(a,d,s,r,R)` — (constructor) creates a DLS–style ADSR envelope. Parameter a specifies the attack time of the envelope: the amount of time (in seconds) for the envelope to go from zero to unity gain. Parameters d and r give the decay and release times (in seconds) of the envelope. These parameters are described in more detail above; however, note that these values do *not* directly give the times spent in the decay and release phases of the ADSR envelope, but are used to specify the decay rates for these phases. The value of s gives the sustain level, and should be assumed to be a value in the range $[0, 1]$; i.e., a linear gain factor. As usual, R specifies the sampling rate.

Note that the envelope parameters a, d, s, r can be zero. You might need to take special action in these cases.

`sustainOff()` — start the release phase of the envelope. After the envelope passes through the attack and decay phases, it remains in the sustain phase until this function is called. Immediately after this function is called, the envelope enters the release phase, even if either the attack or the decay phase has not completed!

`operator()` — returns the current envelope amplitude. The first call to this function returns the initial value of the envelope. Subsequent calls return subsequent values of the envelope amplitude, with the time step between each call given by $T = 1/R$.

Your submission for this question should consist the single file `ADSR.cpp`. You may include only standard C++ header files (as well as `ADSR.h`). You may **not** alter the class interface above in any way. This includes the public as well as the private declarations.