**Additive Synthesis Examples**

**1. Resynthesis of a bell sound.**

This instrument was designed by the composer J C Risset. It generates a number of partials of a bell sound, whose frequencies where obtained by analysis. The envelope is basically a decaying exponential curve, and each partial has a different duration for this decay. **Freq** is a reference value (in Hz) for the bell partial frequencies.

| Partial Number | Frequency (Hz) | Amplitude (relative to a reference value) | Duration (secs) |
|---|---|---|---|
| 1 | 0.58*Freq | 1 | 1 |
| 2 | 0.58*Freq + 1 | 0.67 | 0.9 |
| 3 | 0.91*Freq | 10 | 0.65 |
| 4 | 0.91*Freq + 1.7 | 1.8 | 0.55 |
| 5 | 1.6*Freq | 1.67 | 0.35 |
| 6 | 1.2*Freq | 2.67 | 0.325 |
| 7 | 2*Freq | 1.46 | 0.25 |
| 8 | 2.7*Freq | 1.33 | 0.2 |
| 9 | 3*Freq | 1.33 | 0.15 |
| 10 | 3.75*Freq | 1 | 0.1 |
| 11 | 4.09*Freq | 1.33 | 0.075 |

With these values in hand we can try to re-synthesize the sound. We'll need 11 sinewave oscillators and 11 amplitude envelopes. The outputs of the oscillators are mixed together to generate the complex sound.

So, in csound we can define a number of initialisation variables

```
idur = p3    ; total duration
iamp = p4    ; amplitude reference
ifr =  p5    ; frequency reference
isine = 1    ; sinewave table number
iend = .0009 ; end amplitude for the exponential decay
```

The first and second oscillator-envelope pairs would look like this:

```
k1    expon     1, idur, iend
a1    oscil     k1*iamp, ifr*0.58, isine

k2    expon     1, idur*.9, iend
a2    oscil     k2*iamp*.67, ifr*0.58+1, isine
```

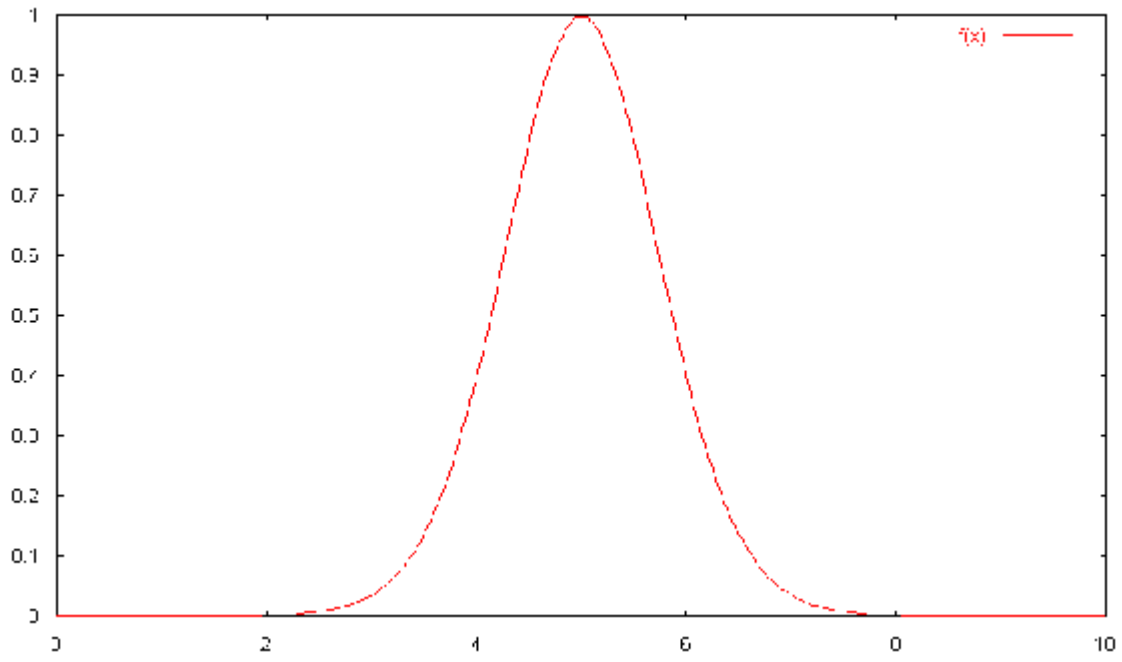
The remaining ones would just follow that format. I choose to generate the envelope with an exponential line generator, but it could be equally done with an oscillator reading a function table.

**Exercise:**

Complete the instrument above and its score. Run it with different values for the reference frequency, use the value of 2000 for the reference amplitude. Create a second version using **oscil1** to read the decaying envelope shape from a function table.

## 2. Endless Glissandos

Another additive synthesis design uses 10 oscillators tuned in octaves, whose frequency continuously varies (creating a glissando). By carefully controlling the amplitude of each oscillator, an endless glissando is heard. The shape of the amplitude function is that of a bell (and it's often called the 'bell' function):




This is an example of a psychoacoustic illusion called **Sheppard tones**. Here's the instrument:

```
; Sheppard Tones,
; J C Risset, from Dodge & Jerse, pp. 95-6
; Victor Lazzarini, 1998

sr= 44100
kr =4410
ksmps = 10
nchnls= 1


instr 1

ifreq   =       p5
iamp    =       p4
```

```
k1  line    0, 120, ftlen(2)   ; generates a ramp from 0 to the table size
k2  line    0, 120, ftlen(3)   ; in 120 secs
                               ; used as an index for table lookup

kamp1   tablei  k1,2,0,0,1   ;reads table 2, indexed by k1("bell" )
kfreq1  tablei  k2,3,0,0,1   ;reads table 3, index by k2 (exponential)
aout1   oscili  kamp1*iamp,kfreq1*ifreq,1 ;generates one of the 10 glissandos

kamp2   tablei  k1,2,0,ftlen(2)*.1,1    ; tables read as before plus offset of
kfreq2  tablei  k2,3,0,ftlen(3)*.1,1    ; 1/10 table length
aout2   oscili  kamp2*iamp,kfreq2*ifreq,1

kamp3   tablei  k1,2,0,ftlen(2)*.2,1      ; etc plus 2/10 table length offset
kfreq3  tablei  k2,3,0,ftlen(3)*.2,1      ;
aout3   oscili  kamp3*iamp,kfreq3*ifreq,1

kamp4   tablei  k1,2,0,ftlen(2)*.3,1
kfreq4  tablei  k2,3,0,ftlen(3)*.3,1
aout4   oscili  kamp4*iamp,kfreq4*ifreq,1

kamp5   tablei  k1,2,0, ftlen(2)*.4,1
kfreq5  tablei  k2,3,0, ftlen(3)*.4,1
aout5   oscili  kamp5*iamp,kfreq5*ifreq,1

kamp6   tablei  k1,2,0,ftlen(2)*.5,1
kfreq6  tablei  k2,3,0,ftlen(3)*.5,1
aout6   oscili  kamp6*iamp,kfreq6*ifreq,1

kamp7   tablei  k1,2,0,ftlen(2)*.6,1
kfreq7  tablei  k2,3,0,ftlen(3)*.6,1
aout7   oscili  kamp7*iamp,kfreq7*ifreq,1

kamp8   tablei  k1,2,0,ftlen(2)*.7,1
kfreq8  tablei  k2,3,0,ftlen(3)*.7,1
aout8   oscili  kamp8*iamp,kfreq8*ifreq,1

kamp9   tablei  k1,2,0,ftlen(2)*.8,1
kfreq9  tablei  k2,3,0,ftlen(3)*.8,1
aout9   oscili  kamp9*iamp,kfreq9*ifreq,1

kamp10   tablei  k1,2,0,ftlen(2)*.9,1
kfreq10  tablei  k2,3,0,ftlen(3)*.9,1
aout10   oscili  kamp10*iamp,kfreq10*ifreq,1

         out     aout1+aout2+aout3+aout4+aout5+aout6+aout7+aout8+aout9+aout10


endin
```

Here's the score:

```
; sine
f1 0 1024 10 1
; bell function (Gaussian Window)
f2 0 512 20 6 1
; exponential from 1 to 2e-10
f3 0 512 5  1  511 .000976562
;     dur amp   freq_max
i1 0  20  10000 3900
```

**Notes:**

1. In order to understand the score, you will have to read up on the **tablei** (and **table**) opcode (please refer to the csound manual for that). Here's a quick word on what it does: it reads a value from a function table according to an **index**. The index determines the position at the table where the value will be extracted. The value is then sent to its output.
2. The function table 2 (the 'bell' function) uses GEN 20 which can generate different types of window shapes, ie. envelopes (please read the csound manual on GEN20)