Code de la composition Bass Drone générée par CSound

```
<CsoundSynthesizer>
<CsOptions>
</CsOptions>
<CsInstruments>

sr = 44100
ksmps = 10
nchnls = 2
0dbfs = 1

; initialize zak space  - one a-rate and one k-rate variable.
; We will only be using the a-rate variable.
zakinit   1, 1

giSine   ftgen   0, 0, 2^12, 10, 1 ; a sine wave

;------------------------------ instru drone_variation
-------------------------------------------------------
instr drone_variation

;controle de l'enveloppe du multi-oscillateur
iatt  = p9
idec  = p10
islev = p11
irel  = p12
kenv  adsr iatt, idec, islev, irel

kFreq expon p6,p8,p7 ;'expon' generates an exponential ramp, from p6 to p7 Hz,
taking p8 seconds
a1 poscil kenv*p4/8, p5*kFreq ; p4=amplitude du signal // kfreq =  frequence
fondamentale (pour memoire, evoluant de p6 à p7 en p8 secondes
a2 poscil kenv*p4/8, p5*kFreq*2     ; p4=amplitude du signal // p5=
multiplicateur de la frequence fondamentale
a3 poscil kenv*p4/8, p5*kFreq*3
a4 poscil kenv*p4/8, p5*kFreq*4
a5 poscil kenv*p4/8, p5*kFreq*5
a6 poscil kenv*p4/8, p5*kFreq*6
a7 poscil kenv*p4/8, p5*kFreq*7
a8 poscil kenv*p4/8, p5*kFreq*8


; on divise par 4 le parametre p4, pour repartir la charge audio entre les 4
oscillateur, evite de faire clipper

asig1 sum a1, a2, a3, a4, a5, a6, a7, a8
; equivalent -> a1+a2+a3+a4 etc.

outs  asig1, asig1

iRvbSendAmt = p13 ; reverb send amount (0 - 1)
; write to zak audio channel 1 with mixing
zawm asig1*iRvbSendAmt, 1


endin
```

```
;-------------------------------- instru drone_plateau
-------------------------------------------
instr drone_plateau

; NOTA: voir si on peut remplacer la courbe exponentielle par une sinusoide
lente (à voir pour la table de fonction qui devrait pouvoir moduler le signal,
pour l'instant on applique des coef fixes pour générer les harmoniques)
; par la suite on pourra faire évoluer l'amplitude des harmoniques pour passer
d'un instru riche à un instru au timbre pauvre

;controle de l'enveloppe du multi-oscillateur
iatt  = p9 ; tps d'attack
idec  = p10  ; tps du decay
islev = p11  ; amplitude du sustain
irel  = p12 ; tps de release
kenv  adsr iatt, idec, islev, irel

kFreq expon p6,p8,p7 ;'expon' generates an exponential ramp, from p6 to p7 Hz,
taking p8 seconds
a1 poscil kenv*p4/8, p5*kFreq ; p4=amplitude du signal // kfreq =  frequence
fondamentale (pour memoire, evoluant de p6 à p7 en p8 secondes
a2 poscil kenv*p4/8, p5*kFreq*2     ; p4=amplitude du signal // p5=
multiplicateur de la frequence fondamentale
a3 poscil kenv*p4/8, p5*kFreq*3
a4 poscil kenv*p4/8, p5*kFreq*4
a5 poscil kenv*p4/8, p5*kFreq*5
a6 poscil kenv*p4/8, p5*kFreq*6
a7 poscil kenv*p4/8, p5*kFreq*7
a8 poscil kenv*p4/8, p5*kFreq*8


; on divise par 4 le parametre p4, pour repartir la charge audio entre les 4
oscillateur, evite de faire clipper

asig1 sum a1, a2, a3, a4, a5, a6, a7, a8
; equivalent -> a1+a2+a3+a4 etc.

outs  asig1, asig1

iRvbSendAmt = p13 ; reverb send amount (0 - 1)
; write to zak audio channel 1 with mixing
zawm asig1*iRvbSendAmt, 1

endin

;------------------------------  instru bruit rose (impulsions)
delay----------------------------

instr noise_burst
kEnv loopseg   0.5,0, 0,1,0.003,1,0.0001,0,0.9969,0,0; amp. env.
aSig pinkish   kEnv       ; pink noise pulses
outs aSig, aSig ; send audio to outputs
iRvbSendAmt  = p4;send reverb
zawm      aSig*iRvbSendAmt, 1

; -- create a delay buffer --
aBufOut delayr   0.5                    ; read audio end buffer
aTap1   deltap   3.1373                 ; delay tap 1
aTap2   deltap   8.2197                 ; delay tap 2
aTap3   deltap   1.4139                 ; delay tap 3
```

```
        delayw   aSig + (aTap3*0.4)     ; write audio into buffer

; send audio to the output (mix the input signal with the delayed signals)
out aSig + ((aTap1+aTap2+aTap3)*0.4)
endin

;------------------------------ blip blip delay ----------------------------
instr blip
; -- create an input signal: short 'blip' sounds --
kEnv    loopseg  0.5,0,0,0,0.0005,1,0.1,0,1.9,0,0; repeating envelope
kCps    randomh  400, 1000, 0.5                 ; 'held' random values
aEnv    interp   kEnv                           ; a-rate envelope
aSig    poscil   aEnv, kCps, giSine             ; generate audio

iRvbSendAmt = p4 ; reverb send amount (0 - 1)
; write to zak audio channel 1 with mixing
zawm aSig*iRvbSendAmt, 1

; -- create a delay buffer --
aBufOut delayr   0.5                     ; read audio end buffer
aTap1   deltap   1.2                   ; delay tap 1
aTap2   deltap   2.6                   ; delay tap 2
aTap3   deltap   1.9                   ; delay tap 3
        delayw   aSig + (aTap3*0.4)     ; write audio into buffer

; send audio to the output (mix the input signal with the delayed signals)
out aSig + ((aTap1+aTap2+aTap3)*0.4)
endin

;------------------------------  instru bruit blanc (nappes diffuses)
----------------------------

instr white_noise

;controle de l'enveloppe du multi-oscillateur
iatt  = p4
idec  = p5
islev = p6
irel  = p7
kEnvWhiteNoise    adsr iatt, idec, islev, irel

;input of rand: amplitude, fixed seed (0.5), bit size
aNoise rand 1*kEnvWhiteNoise, 1, 0 ; syntaxe [, iseed] [, isel] [, ioffset]
aNoise reson aNoise, p9, 80, 1 ; bandpass filtered
;pan du bruit blanc

aPan lfo 0.5,10, 0      ; kamp / freq / type of lfo ici "sine"
aPan    =  aPan + 0.5     ; offset shifted +0.5
aPanL   =  sin((aPan + 0.5) * $M_PI_2)
aPanR   =  cos((aPan + 0.5) * $M_PI_2)

outs    aNoise*aPanL, aNoise*aPanR

iRvbSendAmt = p8 ; reverb send amount (0 - 1)
; write to zak audio channel 1 with mixing
zawm aNoise*iRvbSendAmt, 1

endin

;------------------------------  instru reverb ----------------------------
```

```
instr reverb ;always on
aInSig       zar       1    ; read first zak audio channel
kFblvl       init      0.88 ; feedback level - i.e. reverb time
kFco         init      8000 ; cutoff freq. of a filter within the reverb
aRvbL,aRvbR  reverbsc  aInSig, aInSig, kFblvl, kFco
outs aRvbL, aRvbR ; send audio to outputs
zacl 0, 1 ; clear zak audio channels
endin


</CsInstruments>
<CsScore>
;----------------------------------- demarrage reverb ----------------------
i"reverb" 0 205 ; demarre la reverb et l'arrete a 205 secondes (la piece en fait
200)



;-----------------------------------impulsions de bruit
rose-----------------------------------
;                        p2    p3    p4 (rev send)
i"noise_burst"           0     3     0.9
i"noise_burst"           3     8     0.9
i"noise_burst"           11    2     0.2
i"noise_burst"           15    2.5   0.3
i"noise_burst"           20    3     0.6
i"noise_burst"           26    8     0.7
i"noise_burst"           36    4     0.8
i"noise_burst"           42    8     0.9
i"noise_burst"           54    3     0.9
i"noise_burst"           60    8     0.9
i"noise_burst"           68    2     0.2
i"noise_burst"           72    5     0.001
i"noise_burst"           84    9     0.9
i"noise_burst"           94    1     0.1
i"noise_burst"           96    1     0.1
i"noise_burst"           98    1     0.1
i"noise_burst"           100   1     0.1
i"noise_burst"           106   2     0.6
i"noise_burst"           108   2     0.6
i"noise_burst"           112   8     0.2
i"noise_burst"           120   1     0.9
i"noise_burst"           158   1     0.1
i"noise_burst"           159   1     0.3
i"noise_burst"           162   1     0.5
i"noise_burst"           165   1     0.7
i"noise_burst"           168   1     0.9

;-----------------------------------blips-----------------------------------

i"blip"                  8     10    0.2
i"blip"                  27    10    0.3
i"blip"                  40    10    0.4
i"blip"                  55    10    0.2
i"blip"                  70    6     0.9
i"blip"                  91    4     0.2
i"blip"                  101   4     0.3
i"blip"                  111   4     0.4
i"blip"                  121   1     0.9
i"blip"                  135   20    0.4
```

```
i"blip"                 157   2     0.1
i"blip"                 160   2     0.3
i"blip"                 163   2     0.5
i"blip"                 166   2     0.7
i"blip"                 169   2     0.9

;---------------------------------- bruit
blanc-----------------------------------

;p1                     p2    p3    p4(tps_att) p5(tps_decay)     p6(level_sus)
p7(tps_release)   p8(rev_amount)    p9(center freq filtre passe-bande)
i"white_noise"          23    4     2           0.2               0.6
1                 0.5               1400
i"white_noise"          34    6     0.2         0.2               0.4
4                 0.5               1300
i"white_noise"          42    11.5  10          0                 1
0.01              0.5               1600
i"white_noise"          42    11.5  0.01        0                 1
0.01              0.5               9000
i"white_noise"          42    11.5  0.01        0                 1
1                 0.5               750
i"white_noise"          60    4     2           0.2               0.6
1                 0.5               1400
i"white_noise"          75    14    3           5                 0.2
6                 0.9               8000
i"white_noise"          100   6     0.2         0.2               0.4
4                 0.5               1300
i"white_noise"          111   10    3           6                 0.2
1                 0.9               8000
i"white_noise"          120   14    12          0                 0.8
2                 0.5               1600
i"white_noise"          134   10    2           0.5               0.4
4                 0.9               10000
i"white_noise"          144.5 22    1           0                 1
6                 0.2               1500
i"white_noise"          167   4     0.001       0.5               0.2
2                 0.9               5000
i"white_noise"          172   28    10          10                1
8                 0.2               880

;---------------------------------- drone ----------------------------------
;argument p1= numéro de l'instrument (i1) // argument p2=nb de secondes avant de
jouer // argument p3 =durée du son // argument p4= amplitude du signal //
argument p5= coef multiplicateur de la frequence fondamentale kfreq // argument
p6 et p7 = bornes d'évolution de la frequence kfreq selon une rampe
exponentielle // argument p8 = durée d'évolution de kfreq selon la rampe p6->p7
;----------------------------------montee
;p1                     p2    p3    p4 p5 p6 p7 p8    p9att       p10dec
p11sus      p12rel      p13reverb
i"drone_variation"      60    12    .5 1. 10 39 10    1           0.
0.6         1           0.5
i"drone_variation"      60    12    .4 2. 10 25 10    1           0.
0.6         1           0.3
i"drone_variation"      60    12    .5 1. 10 37 11    1           0.
0.4         1           0.4
i"drone_variation"      60    12    .3 2. 10 30 11    1           0.
0.4         1           0.3

;----------------------------------etat plateau
```

```
;p1                    p2    p3    p4 p5 p6 p7 p8    p9att       p10dec
p11sus      p12rel     p13reverb
i"drone_plateau"  71    50    .5 1. 38 39 11    1.2         0.          0.6
1           0.5
i"drone_plateau"  71    50    .3 2. 29 31 11    1.2         0.          0.6
1           0.3
i"drone_plateau"  71    50    .5 1. 39 37 11    5           0.          0.4
1           0.4
i"drone_plateau"  71    50    .3 2. 31 30 11    8           0.          0.4
1           0.3

;-----------------------------------descente
;p1                    p2    p3    p4    p5 p6 p7    p8    p9att p10dec
p11sus      p12rel     p13reverb
i"drone_plateau"  121   11    .8    1. 39 1     10    1.    0.          0.6
10          0.5
i"drone_plateau"  121   11    .3    2. 38 1     7.3   1.    0.          0.6
10          0.3
i"drone_plateau"  121   11    .5    1. 37 1     5     0.9   0.          0.4
10          0.4
i"drone_plateau"  121   11    .3    2. 30 1     8     0.9   0.          0.4
10          0.3

;-----------------------------------secousses de fin
;p1                    p2    p3    p4    p5 p6 p7    p8    p9att p10dec
p11sus      p12rel     p13reverb
i"drone_variation"      120.2 2.5   .4    1. 10010    0.8   0.001 0.          1.
      2.49        0.2;secousse 1
i"drone_variation"      121.2 2.5   .35   1. 80 10    0.8   0.001 0.          1.
      2.49        0.2;secousse 1
i"drone_variation"      122.2 2.5   .3    1. 60 10    0.8   0.001 0.          1.
      2.49        0.2;secousse 1
i"drone_variation"      124.  4.    .2    1. 49 30    0.8   0.001 0.          1.
      3.99        0.2;secousse 4

;---------------------------------- retour de drone final
;p1                    p2    p3    p4 p5  p6         p7    p8    p9att
p10dec      p11sus     p12rel     p13reverb
i"drone_variation"      140   60    .5 0.7 40         38.5  20    5           20
      0.8         10          0.4
i"drone_variation"      145   45    .1 1. 18000       20000 1     5           25
      0.2         0.01        0.9
i"drone_variation"      145   45    .1 1. 20000       18000 1     5           25
      0.2         10          0.5
i"drone_variation"      150   40    .1 1. 20000       16000 1     5           20
      0.2         10          0.45
i"drone_variation"      155   35    .1 1. 20000       14000 1     5           15
      0.2         10          0.35
i"drone_variation"      160   30    .1 1. 20000       12000 1     5           10
      0.2         10          0.30

e
</CsScore>
</CsoundSynthesizer>
```