Tècniques de microsampling amb generadors de congruències lineals

Francesc Martí Pérez
Universitat Oberta de Catalunya
martifrancesc@uoc.edu

Resum

En el següent article l'autor descriu una nova tècnica per generar sons amb síntesi granular síncrona fent servir generadors de congruències lineals. S'estudia en particular el generador de números pseudoaleatoris de Lehmer i com aplicar-lo per generar i emetre grans de so en situacions de densitat granular baixa, que equival, en moltes situacions, a parlar de sons granulars amb caràcter rítmic. Finalment, mostra varies aplicacions del mètode relacionades amb la multimèdia, els videojocs i la creació artística.

I. Introducció

Síntesi granular sincrònica
La síntesi granular és una tècnica de síntesi sonora que permet crear esdeveniments sonors a partir de sons molt curts anomenats *grans* o *grans de so* (Gabor 1947; Roads 1988 i 2001). Un gra "típic" té una durada d'entre 1 i 100 mili-segons i pot ser un so sintetitzat o un fragment d'un *sample* (és a dir, d'un so pregravat).

El mètode de síntesi granular anomenat *síntesi granular sincrònica* (SGS, per les seves sigles en anglès) és un tipus de síntesi granular que es caracteritza per disposar o emetre els grans de so a una determinada freqüència o relació lineal d'aquesta.

Aquest tipus de tècnica s'ha fet servir per sintetitzar sons amb una o varies ressonàncies com poden ser alguns instruments reals o veus (Roads 2001). Un gra de so amb energia sobre 3 KHz que es repeteixi cada 10 mili-segons generarà un so amb harmònics múltiples de 100 Hz i amb un *formant* (una zona de ressonància) sobre els 3 KHz.

De totes formes, amb la mateixa tècnica, també es poden aconseguir altre tipus de sons molt diferents. A la síntesi granular, s'entén per *densitat* al número de grans de so emesos per segon. Al ser objectes de curta durada, densitats de centenars o milers de grans de so per segon són magnituds normals de treball. No obstant això, aquesta tècnica no limita la densitat i, en el cas de la síntesi granular sincrònica, una densitat molt baixa (menys de 20-30 grans per segon) pot generar sons rítmics.

És en aquesta variant de síntesi granular i amb densitats de gra baixes en la que l'autor es centrarà en aquest article.

La generació de números pseudoaleatoris
Actualment, la majoria de generadors de números pseudoaleatoris que s'utilitzen són generadors de congruències lineals (LCG, per les seves sigles en anglès). Aquests generadors es basen en la fórmula recursiva descoberta per D.H. Lehmer l'any 1949 (Lehmer 1951; Deng i Lin 2000) que, donat un valor inicial x_0, els successius valors $x_n, n \geq 1$ s'obtenen d'acord amb

$$x_n = (ax_{n-1} + c) \bmod m,$$

on x_0, a, c i m (la llavor, el multiplicador, la constant additiva i el mòdul) són enters i $m > 0$, $0 \leq a < m$, $0 \leq c < m$ i $0 \leq x_0 < m$.

Aquests generadors, que reben el nom de mixtes quan $c > 0$ i de multiplicatius quan $c = 0$, proporcionen números enters en $[0, m)$, que es converteixen en una successió de números uniformes al interval $[0, 1]$ amb l'operació $u_n = x_n/(m-1)$.

A pesar de la seva simplicitat i que la successió de nombres x_n no és en absolut aleatòria en estar completament determinada pels valors de x_0, a, c i m (Wehrwein 2007), una encertada selecció dels paràmetres a, c i m permet obtenir successions molt llargues i semblants a les de números completament aleatoris. S'ha de fer notar, també, que les successions de números obtingudes amb generadors congruencials són cícliques (o tendeixen a un valor constant), sent m el seu període màxim. I si bé l'elecció dels paràmetres s'acostuma a enfocar cap a l'obtenció de seqüencies de números pseudoaleatoris amb períodes molt grans (Hellekalek 1998), al parlar de les aplicacions del nou mètode de generació de sons granular que es presenta en aquest article es veurà que, de vegades, seqüències amb períodes relativament petits també seran molt vàlides en funció de la finalitat que si li vulgui atorgar.

El següent teorema (Wehrwein 2007) diu quines són les condicions suficients i necessàries que han de complir els paràmetres del generador congruencial per a que aquest tingui un període complet:

Teorema. Un generador congruencial té un període complet sí i sols si es compleixen les condicions:
1. $mcd\ (c, m) = 1$;
2. $a = 1\ mod\ p$ per a cada factor primer p de m;
3. $a = 1\ mod\ 4$ si 4 divideix a m.

Per exemple, és ben conegut que totes aquestes condicions es compleixen si $m = 2k$, $a = 4n + 1$ i c és senar, on n, c i k són enters positius (Wehrwein 2007). A la Figura 1, l'exemple c) és un exemple de generador congruencial amb període complet.

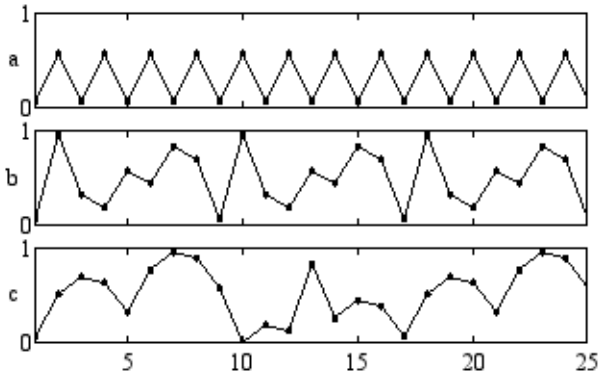

Figura 1. Tres representacions gràfiques del Generador Lineal de Congruencies de Lehmer amb llavor x_0= 1 i m =16. (a) a = 7, c = 2; (b) a = 13, c = 2; (c) a = 5, c = 3

Per últim, indicar que tot que el mètode presentat en aquest article és independent de la fórmula que s'utilitzi per generar els números pseudoaleatoris, però considerem que aquesta equació, per la seva senzillesa i possibilitats, és la millor opció per començar.

El sintetitzador granular Micro-X12

Per poder explicar millor alguns aspectes de la tècnica que presentem, parlarem en diverses ocasions del Micro-X12, un sintetitzador de síntesi granular sincrònica que porta incorporat un mòdul per aplicar aquest mètode de generació sonora, desenvolupat per l'autor amb el llenguatge de programació Processing i la llibreria d'àudio a temps real Beads (Merz 2011). Al Micro-X12, l'emissió de grans està en sincronia amb el *clock* (tempo) definit, que pot variar entre 30 i 300 BPM; i cada BPM està subdividit, al seu torn, en 32 *ticks*. Els grans de so s'emeten en sincronia amb aquests ticks, sent aleshores el número de ticks per segon la freqüència d'emissió de grans del sintetitzador. Un càlcul senzill mostra que amb un temps

màxim de 300 BPM es pot emetre un gra cada 6,25 mili-segons, el que dóna una densitat màxima de 160 grans de so per segon.

De forma molt succinta, les característiques fonamentals d'aquest programa són:

- És un sintetitzador en temps real de síntesi granular sincrònica.
- Els grans de so s'extrauen d'arxius d'àudio pregravats (*samples*), podent-se realitzar extraccions sobre diferents arxius d'àudio simultàniament.
- El programa conté tots els paràmetres de control típics de la síntesi granular. Breument:
    a) Ordre de selecció dels grans: seqüencial, semi-seqüencial i aleatòria
    b) Densitat de gra
    c) Duració de gra
    d) Volum i forma de l'envol-vent d'amplitud
    e) Transposició del to del gra
    f) Posició del gra a l'espai (*panner*)
- El programa incorpora un potent mòdul de generació de successions de números pseudoaleatoris aplicables als paràmetres.

Així doncs, si bé el mètode que s'exposarà a continuació sobre l'ús de generadors congruencials dins la síntesi granular síncrona és general i es pot fer servir en qualsevol altre sintetitzador, farem referència al Micro-X12 quan sigui necessari mostrar un exemple sobre programari real.

II. Generació i manipulació de grans sonors amb generadors de congruències lineals

Consideracions prèvies

Començarem detallant una mica el funcionament del Micro-X12. A la Figura 2 es descriu de forma esquemàtica el procés de generació i emissió d'un gra de so. Com ja s'ha indicat, la distància entre dos grans sempre serà múltiple de la distància entre dos *ticks* del rellotge del sintetitzador i anomenarem *trigger* a l'esdeveniment que provoca que s'emeti un gra de so en un tick determinat.

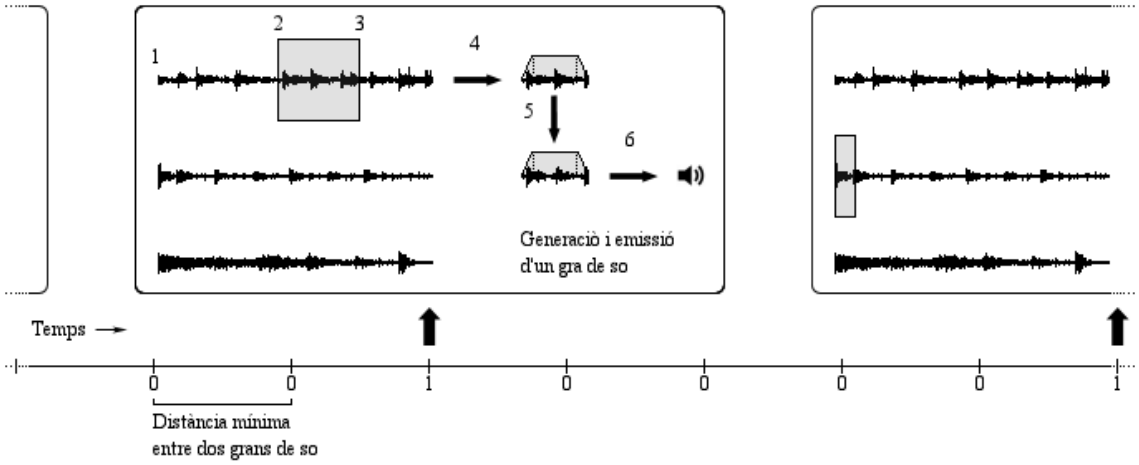

Figura 2: Esquema del funcionament del sintetitzador granular Micro-X12.

A cada tick generat pel rellotge, el programa avalua si s'ha produït un esdeveniment de tipus trigger i, per tant, si s'ha de crear i emetre un gra de so (de fet, d'emetre, ja que el gra podria estar ja 'preparat' per ser emès). La configuració d'aquests esdeveniments del programa es realitza gràficament amb unes barres de desplaçament (*sliders*), els valors dels quals s'aniran llegint seqüencialment: el programa llegeix els valors dels sliders activats d'esquerra a dreta, tornant al inici un cop llegits tots, és a dir, cada paràmetre té un seqüenciador per passos amb un número variable de passos entre 1 i 16. Un seqüenciador per passos es

pot interpretar com a una successió matemàtica de nombres. Per exemple, si tenim activats 3 sliders del paràmetre trigger i aquests marquen els valors 2, 3 i 4 (distància en ticks que hi haurà entre dos grans de so), podem interpretar aquesta configuració com la successió de nombres (2, 3, 4, 2, 3, 4, 2....) que provocarà que s'emeti un gra de so als 0 tikcs, als 2 ticsk, 5, 9, 11, 15, 18, etc.

Un cop determinat que s'ha d'emetre un gra de so, es posa en marxa un procés articulat en 6 tasques:

1. Selecció del sample del qual s'extraurà el gra de so
2. Selecció del lloc d'on s'extraurà (*Offset*)
3. Durada en temps del gra de so
4. Aplicació d'una envolvent d'amplitud
5. Processament del to (*pitch*) del gra de so
6. Distribució a l'espai del gra de so (*panner*)

Tots els valors numèrics necessaris per realitzar aquestes operacions també es generen amb la lectura de les barres de desplaçament de cada paràmetre de forma seqüencial. Per exemple, en relació al paràmetre que permet seleccionar l'arxiu d'àudio del que s'extraurà el gra de so (sample), la successió (7, 5) indicaria que al primer esdeveniment *trigger* s'hauria de crear el gra de so fent servir la mostra d'àudio número 7, amb el segon trigger la mostra 5, amb el tercer el 7, etc.

Si tenim, per exemple, configurades les barres de desplaçament (recordar que cada paràmetre té el seu propi seqüenciador i que pot tenir un número diferent de passos activats) amb els valors Sample = (1, 3, 5, 7), Offset = (100, 300, 200), Durada del gra de so = (10, 50, 100), Envolvent d'amplitud = (10), Pitch = (1, -1) i Panner = (0), per a cada esdeveniment de tipus trigger podem representar la configuració del gra de so amb la matriu $M_j, j \geq 0$

$$M_0 = \begin{pmatrix} 1 \\ 100 \\ 10 \\ 10 \\ 1 \\ 0 \end{pmatrix}, M_1 = \begin{pmatrix} 3 \\ 300 \\ 50 \\ 10 \\ -1 \\ 0 \end{pmatrix}, M_2 = \begin{pmatrix} 5 \\ 200 \\ 100 \\ 10 \\ 1 \\ 0 \end{pmatrix}, M_3 = \begin{pmatrix} 7 \\ 100 \\ 10 \\ 10 \\ -1 \\ 0 \end{pmatrix}, ..., M_j = ...$$

De forma general, per a un esdeveniment trigger número j, la creació i emissió del gra de so ve determinat per la matriu

$$M_j = \begin{pmatrix} k^{(0)}x_j^{(0)} + t^{(0)} \\ k^{(1)}x_j^{(1)} + t^{(1)} \\ k^{(2)}x_j^{(2)} + t^{(2)} \\ k^{(3)}x_j^{(3)} + t^{(3)} \\ k^{(4)}x_j^{(4)} + t^{(4)} \\ k^{(5)}x_j^{(5)} + t^{(5)} \end{pmatrix}$$

on $x_j^{(i)}$, $0 \leq i \leq 5$, pertany al interval [0, 1] i $k^{(i)}$ i $t^{(i)}$, $0 \leq i \leq 5$, són les constants que converteixen els valors normalitzats $x_j^{(i)}$en el valors reals dels paràmetres. Per simplificar la notació, com els valors $k^{(i)}$ i $t^{(i)}$són constants per qualsevol i, els $x_j^{(i)}$ ja determinen completament el gra de so i, per tant, es pot considerar la matriu

$$M_j = \begin{pmatrix} x_j^{(0)} \\ x_j^{(1)} \\ x_j^{(2)} \\ x_j^{(3)} \\ x_j^{(4)} \\ x_j^{(5)} \end{pmatrix}$$

com la matriu M_j de valors dels paràmetres per al trigger número j.

Si tots el seqüenciadors dels paràmetres tinguessin els 16 passos activats, la successió de matrius M_j seria sempre una successió cíclica de període 16. Però com cada paràmetre pot tenir una configuració de seqüenciació diferent, un simple càlcul matemàtic demostra que el número de combinacions possibles per a 6 seqüenciadors amb un número de passos actius entre 1 i 16 es correspon amb el $mcm(16, 13, 11, 9, 7, 5) = 720720$. Per tant, la successió de matrius M_j és una successió cíclica de període inclòs al interval [1, 720720].

Així doncs, per resumir, en aquest apartat hem vist que la generació i configuració dels grans de so es pot representar amb dos successions: una successió de números cíclica de període màxim 16 per generar els esdeveniments trigger i una successió de matrius de dimensió $6x1$, també cíclica i amb període màxim 720720, amb els valors dels paràmetres dels grans.

Control dels paràmetres del sintetitzador amb generadors de congruències lineals
Un cop demostrat que per funcionar, el Micro-X12 simplement necessita dos successions -una de números i una altra de matrius- passarem a veure com podem generar aquestes successions amb un o varis generadors de congruències lineals de Lehmer (LLCG); i deixarem pel següent punt l'anàlisi dels resultats d'aquestes operacions.

Començarem amb la successió dels triggers. Com ja hem vist, al Micro-X12, els esdeveniments que provoquen que s'emeti un gra de so venen determinats per una successió de nombres cíclica de període màxim 16. Aquesta successió, que es genera amb els barres de desplaçament gràfiques del programa, també es pot generar amb un LLCG. Sense entrar en detalls, el sintetitzador disposa d'un mòdul de generació de successions de números pseudoaleatoris (MGNPA a partir d'ara) que permet definir un LLCG i associar-lo a un paràmetre.

Per exemple, si activo un generador pel paràmetre trigger i el configuro amb els valors $a = 17$, $c = 13$, $m = 31$ i valor inicial $x_0 = 0$, internament el programa ja genera una successió de números pseudo-aleatoris al interval [0, 1]: (0.03, 1.0, 0.9, 0.23, 0.26, 0.83,...). I, suposant que el programa tingui definit un interval de valors de trigger entre [8, 64] i un factor de quantització de 8, amb un senzill càlcul matemàtic s'obté que els valors de la nova successió que passa a controlar els triggers del sintetitzador serà (8, 32, 40, 56, 8, 64, 64, 16, 24, 56, 16,...). És a dir, hem substituir la successió que es generava amb les barres de desplaçament per una que el programa genera internament en base a una fórmula LLCG.

De forma anàloga, per a cada paràmetre, cada successió de valors pot ser creada amb un LLCG i, per extensió, la successió de matrius M_j pot ser generada amb 6 LLCG. Per fer això, al MGNPA s'activen i configuren 6 LLCG ($f^{(0)}$, $f^{(1)}$, $f^{(2)}$, $f^{(3)}$, $f^{(4)}$, $f^{(5)}$) i s'assignen als 6 paràmetres del sintetitzador. La matriu M_{j+1} associada al esdeveniment trigger $j + 1$ es pot representar ara per com una matriu de funcions LLCG

$$M_{j+1} = \begin{pmatrix} f^{(0)}(x_j^{(0)}) \\ f^{(1)}(x_j^{(1)}) \\ f^{(2)}(x_j^{(2)}) \\ f^{(3)}(x_j^{(3)}) \\ f^{(4)}(x_j^{(4)}) \\ f^{(5)}(x_j^{(5)}) \end{pmatrix} = \begin{pmatrix} (a^{(0)}x_j^{(0)} + c^{(0)})\ mod\ m^{(0)} \\ (a^{(1)}x_j^{(1)} + c^{(1)})\ mod\ m^{(1)} \\ (a^{(2)}x_j^{(2)} + c^{(2)})\ mod\ m^{(2)} \\ (a^{(3)}x_j^{(3)} + c^{(3)})\ mod\ m^{(3)} \\ (a^{(4)}x_j^{(4)} + c^{(4)})\ mod\ m^{(4)} \\ (a^{(5)}x_j^{(5)} + c^{(5)})\ mod\ m^{(5)} \end{pmatrix}$$

on els termes $a^{(t)}, c^{(t)}, m^{(t)}$ i $x_0^{(t)}, 0 \leq t \leq 5$, han de complir les condicions anteriorment exposades.

I aquesta seria la idea central de la primera part de l'article: la part gràfica del sintetitzador, les barres de desplaçament i els seus seqüenciadors es poden substituir per 7 LLCG (una pel trigger i 6 pels paràmetres). Podem passar d'utilitzar el programa directament a través dels controls gràfics a donar-li el control als generadors congruencials que, al ser fórmules recursives, es van executant sense parar.

Generadors en paral·lel

Anant una mica més al fons de les possibilitats del programa Micro-X12, no era del tot exacte el que hem comentat abans de que el MGNPA permet assignar-li un LLCG a cada paràmetre. En realitat va més enllà: es pot assignar un LLCG a cada barra de desplaçament, amb el que un paràmetre pot arribar a ser controlat ja no per una única LLCG sinó per tota una successió de LLCG. Això es pot entendre fàcilment amb el següent exemple. Suposem que el seqüenciador del paràmetre *sample* tingui 4 passos i que els seus valors siguin (1, 9, 4, 2). En aquest paràmetre podem definir un LLCG per al pas/barra de desplaçament 2 i un altre LLCG per al pas 4. Cada un d'aquests generadors anirà canviant els valors de les posicions 2 i 4, de forma que la successió resultant podria ser alguna cosa com 1, 3, 4, 6, 1, 8, 4, 5, 1, etc. En el cas d'assignar un LLCG a cada un dels passos actius, la successió (1,9,4,2) es podria considerar una matriu de valors inicials.

Evidentment, tot l'explicat abans continua sent vàlid, ja que si es vol controlar completament un paràmetre amb un única LLCG només tenim que activar una barra de desplaçament i assignar-li una LLCG.

III. Síntesi granular i aleatorietat

Fins aquí hem vist com un sintetitzador de síntesi granular síncrona pot ser controlat per successions de números i matrius, i com aquestes successions poden ser generades amb LLCGs. Aquesta és una nova forma de controlar un sintetitzador granular, però que està limitada a que -sobretot- aquest treballi amb densitats de gra molt baixes, doncs el control particularitzat que aquest mètode realitza sobre cada gra de so no tindria sentit amb densitats de milers de grans per segon. Quan diem una nova forma de controlar un sintetitzador granular no estem dient que fins ara no s'hagi experimentat amb procediments aleatoris o estadístics dins la síntesi granular. Tot el contrari, forma part de la seva essència, però sempre orientats a grups de grans, no al control de grans particulars.

També, en el cas que es treballés (com es comentarà en alguna de les aplicacions següents) amb longituds de gra més grans de les típiques seria preferible parlar de *microsampling* abans que de síntesi granular, ja que l'emissió de pocs grans de so i que aquests tinguin durades superiors a les normals no casa molt amb la filosofia d'aquest tipus de síntesi.

Una altra particularitat d'aquesta tècnica, que hereta de forma natural la característica dels generadors congruencials, és la de poder tornar repetir de forma senzilla les condicions en les que s'havia efectuat un experiment. Com es veurà més endavant en una de les aplicacions, aquest mètode permet generar sons de

varis minuts de durada que es poden "guardar" en uns quants bytes (els valors de les variables dels generadors).

Anem a mostrar ara algunes aplicacions d'aquesta nova tècnica i resultats que l'autor ja ha obtingut amb ella.

IV. Aplicacions i resultats

Artístiques: composició algorítmica, disseny d'instal·lacions sonores i experimentació sonora.
Les aplicacions musicals relacionades amb aquesta tècnica són moltes. Per exemple, l'autor fa anys que explora amb èxit aquest concepte de crear sons rítmics fent servir síntesi granular síncrona i procediments aleatoris. La utilització de generadors congruencials per a la creació d'obres musicals no és una novetat, ha sigut una constant des de fa molts anys (Battey 2004), però ara ja no estem parlant de generar notes o esdeveniments MIDI amb aquests generadors, ara parlem d'utilitzar eines de composició algorítmica directament sobre mostres d'àudio.

Un altre camp on l'autor ha experimentat aquest mètode és en el disseny d'instal·lacions sonores. Bàsicament, era una instal·lació sonora multicanal amb diferents sons que s'emetien de forma aleatòria per diferents altaveus. Una opció hagués sigut fer servir les típiques funcions random() per emetre de forma aleatòria pels diferents altaveus els diferents sons, a diferents volums, durades, etc. Però els problemes eren varis, a més de que un comportament absolutament aleatori sense cap control tampoc era el desitjat. Una de les avantatges de fer servir LLCGs per realitzar aquest tipus de tasques és que les successions que s'obtenen, tot i que semblin valors aleatoris, en realitat no ho són. Aquestes successions de números pseudoaleatoris poden ser prèviament testejades i, en cas d'èxit, tornar-se a reproduir (¡a reproduir milers de números!) guardant en memòria només 4.

Per escenificar aquesta idea, definim tres generadors $f^{(0)}$, $f^{(1)}$i $f^{(2)}$ assignats al paràmetres de síntesi granular volum, sample i panner, on

$f^{(0)}$: controla el volum de les mostres. A la imatge que generarem, el volum es visualitza amb el radi de la circumferència: major radi = major volum.
$f^{(1)}$: controla quina de les mostres sonores s'emetrà. Per simplificar, en aquest exemple farem servir només 3 mostres que vindran determinades pel color de la circumferència.
$f^{(2)}$: aquest LLCG distribueix les mostres per l'espai, indicant per quins dels altaveus s'ha d'emetre.

Ara, podem fer diferents proves, amb diferents configuracions, fins a trobar uns resultats satisfactoris que podrem guardar en uns quants bytes de memòria i que podran ser tornats a executar les vegades que siguin necessàries.

A les següents configuracions utilitzem $m = 128$ i $x_o = 1$, amb

Configuració 1: $f^{(0)}: a = 9, c = 3$; $f^{(1)}: a = 13, c = 11$; $f^{(2)}: a = 17, c = 31$
Configuració 2: $f^{(2)}: a = 87, c = 111$ i mateixos valor per $f^{(0)}$i $f^{(1)}$.
Configuració 3: $f^{(2)}: a = 15, c = 31$ i mateixos valor per $f^{(0)}$i $f^{(1)}$.

I genero una imatge per visualitzar com es comporta cada una de les configuracions testejades (Figura 3). Es pot veure que amb la Configuració 1, les circumferències estan més distribuïdes per l'espai. Això és degut a que el cicle de $f^{(0)}$ és el màxim (128), mentre que les altres dos tenen períodes més petits.

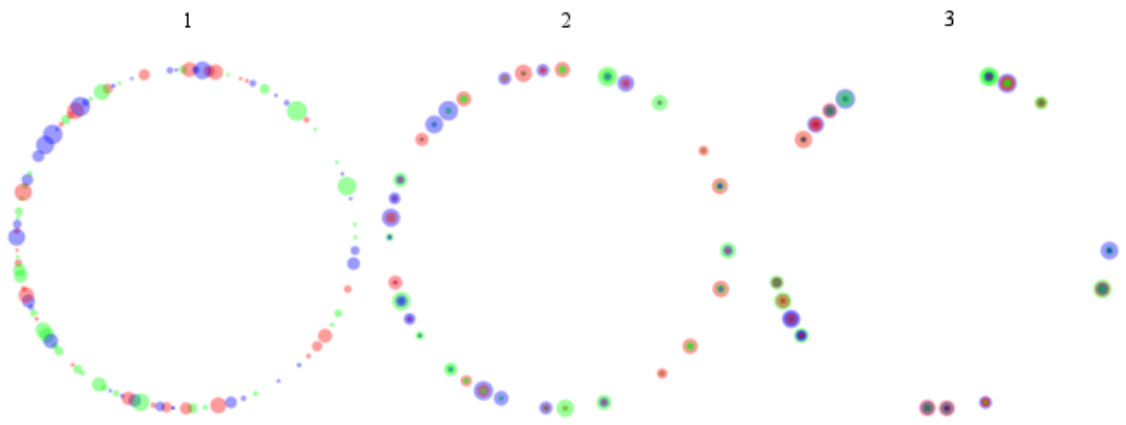

Figura 3.

Quan en un punt anterior es comentava que en alguns casos no sempre el que es buscaria seria trobar la successió pseudoaleatòria més llarga i més perfecta en termes d'aleatorietat, pensàvem en aquests exemples amb un componen artístic que pot fer que una certa periodicitat dels esdeveniments sigui necessària.

Generació de "textures rítmiques"

La síntesi de textures sonores tipus pluja, vent o soroll de tràfic és una important aplicació pel cinema, videojocs, instal·lacions i creacions multimèdia en general (Schwarz 2011). Si bé el mètode aquí presentat no pot donar solució a cap d'aquests problemes, sí que pot ser una tècnica adequada per a la generació de textures que podríem anomenar "rítmiques". Per exemple, generar els sons d'uns animals en una jungla: sons que van apareixen i desapareixen de forma aleatòria en un videojoc o en una aplicació multimèdia i que poden generar un problema si aquests han d'estar sonant molta estona. Amb el mètode presentat en aquest article, amb necessitat de poca memòria per guardar els sons pregravats i els valors de les LLCGs, es podrien generar hores de successions de sons d'aquest tipus sense arribar a repetir la successió en cap moment.

A la Figura 4 es mostra de forma molt esquemàtica un exemple amb aquesta idea. Partiríem inicialment d'una llibreria de sons d'ocells cantant, i configuraríem -per exemple- 3 LLCG amb $m = 32$ i

$$f^{(0)}: x_0 = 1, a = 9, c = 3;\ f^{(1)}: x_0 = 21, a = 13, c = 11;\ f^{(2)}: x_0 = 16, a = 17, c = 31$$

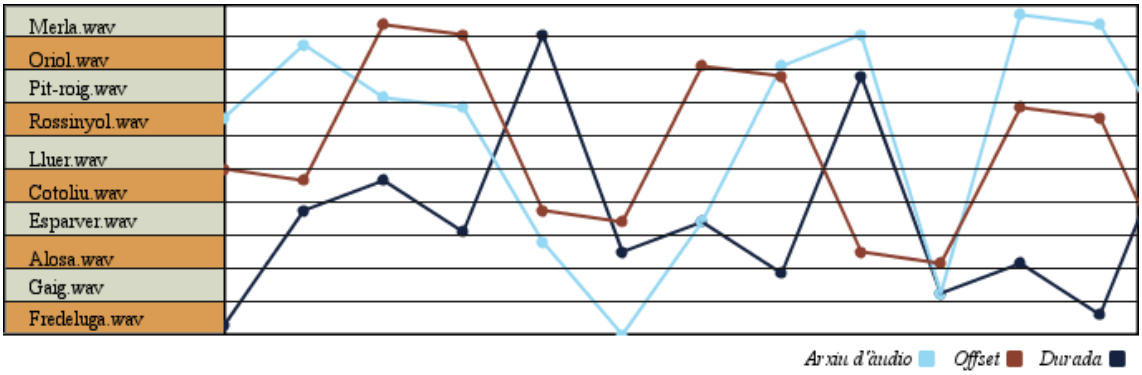

Figura 4.

La interpretació del gràfic de la Figura 4 pot ajudar a entendre la idea de generar aquesta textura rítmica agafant contínuament diferents mostres, amb diferents offsets, diferents durades, volums, situació a l'espai, etc.

Creació de *Adaptive music* per videojocs.
La música adaptada (*adaptive music* en anglès) per videojocs és una música que canvia en funció de les condicions de joc (Young 2012). És un problema obert i de difícil solució.

Una de les possibles línies d'investigació que podria seguir la tècnica de síntesi presentada en aquest article seria com a aplicació per generar música adaptada. Si més no, quan la seva component rítmica sigui destacable. Suposem que tenim una llibreria de sons rítmics i tot un conjunt de LLCGs definides $f^{(k)}$ sobre aquesta llibreria. I que cada una d'elles hagi estat definida de forma que el ritme que creï tingui un caràcter ben definit (no seria complicat configurant amb cura la LLCG que selecciona els samples). Aleshores, un algoritme pot relacionar les condicions del joc amb alguna d'aquestes$f^{(k)}$, canviant de funció $f^{(k)}$, i de rítme per tant, cada cop que les condicions del joc canviïn.

V. Conclusions

En aquest article s'ha presentat una nova tècnica, flexible i potent per generar sons de caràcter rítmic amb síntesi granular síncrona i el generador de congruències lineals de Lehmer. Relacionat amb això s'ha presentat també el sintetitzador, desenvolupat amb Processing i Beads, Micro-X12, una invitació a tots els compositors interessats en la composició algorítmica, però també interessats en la manipulació de sons digitalitzats. Finalment, s'han analitzat alguns resultats satisfactoris de la tècnica i s'han anomenat algunes possibles línies futures d'investigació com la creació de textures rítmiques aleatòries i la generació de música adaptada per videojocs.

Referències

[1] Gabor, D. 1947. "Acoustical quanta and the theory of hearing." *Nature* 159(4044): 591-594.
[2] Roads, C. 1988. “Introduction to Granular Synthesis.” *Computer Music Journal* 12(2): 11-13.
[3] Roads, C. 2001. *Microsound*. Cambridge, Massachusetts: The MIT Press.
[4] López, S., Martí, F., and Resina, E. 1998. "Vocem: an application for real-time granular synthesis." In B. Garau and R. Loureiro, eds. *Proceedings 98 Digital Audio Effects Workshop*. Barcelona: Pompeu Fabra University. pp. 219-222.
[5] Lehmer, D. H. 1951. "Mathematical methods in large-scale computing units". In *Proceedings of the Second Symposium on Large-Scale Digital Computing Machinery*. Cambridge, Massachusetts: Harvard University Press. pp. 141– 146.
[6] Deng, L.-Y., Lin, D. K. J. 2000. "Random Numbers Generation for the new Century". *The American Statistician* 54(2): 145-150.
[7] Wehrwein, J. 2007. *Random Number Generation*. Senior Thesis. Middlebury College.
[8] Hellekalek, P. 1998. "Good random number generators are (not so) easy to find". *Mathematics and Computers in Simulation* 46: 485-505.
[9] Merz, E. 2011. "Sonifying Processing: The Beads Tutorial". Internet: http://www.computermusicblog.com/SonifyingProcessing/
[10] Schwarz, D. 2011. "State of the art in sound texture synthesis". Proc. of the 14th Int. Conference on Digital Audio Effects (DAFx-11). Paris.
[11] Young.D. M. 2012. "Adaptive game music: the evolution and future of dynamic music systems in video games". Ph.D. thesis. Ohio: Ohio University.
[12] Battey, B. 2004. “Musical Pattern Generation with Variable-coupled Iterated Map Networks”. *Organised Sound* 9(2): 137–150.