

```

/*          R I P P L E S   A L G O R I T H M          */

/*          BY DOUGLAS GEERS, 1997          */

/* NOTE: This C code constitutes the algorithm used to create the
   musical sounds of my composition Ripples, written in the
   spring of 1997.
*/

#include <stdio.h>

/* DECLARE SUB-FUNCTIONS */
float myrand(float lobound, float upbound);
void playcmixwtable(float start, float dur, float amp, float
pitch, float pan);
void open_out_scorefile(int argc);

/* DECLARE OTHER GLOBAL VARIABLES */
FILE *scorefile;
char outfilename[255];

/* ***** THE MAIN FUNCTION ***** */
void main (int argc, char *argv[])
{

/* ***** SETUP VALUES !!!! ***** */
float totaltime=60.0; /* <--SPECIFY TOTAL TIME TO PLAY IN SECONDS
*/
float peaktimel, peaktimel2, pan;
float earliestpeak1=10.0;
float latestpeak1=45.0;
float earliestpeak2=30.0;
float latestpeak2=50.0;
float midpoint, midtime;
/* SPECIFY TEMPO --IN BEATS/Minute */
float tempo = 60;
float beattime=60/tempo;
float amp=500.0, pitch=8.0;

```

```

/* DECLARE OTHER VARIABLES */
int octavenum, num, count, maxnotes, tuplet, note, timetogo, i;
int pandir;
float start=0, percentime, percentnotes, seed, temp;
float peakpoint1, peakpoint2, notedur,randnum;
char sndfilename[255];

float peakpoints[25][6];
float restprcnt[25][4]; /* EACH LINE WILL HAVE RANDOM % of RESTS
IN EACH SECTION */

/* for each of the 20 individual lines, this will record: peak
timel,
    peak amp1, peak density1, peak time2, peak amp2, peak
density2 */

float pitches[7]={0.0, 0.02, 0.03, 0.05, 0.07, 0.08, 0.10};
float subsetpitches[7]; /* these will be notes used by each motive
*/

/* PROGRAM'S USER-FRIENDLY GREETING: */
printf("\n\nRipple-maker 5, ver. 97.1\n\n");

    /* SET RANDOM SEED NUMBER */
    printf("\nInput a random (0-1) seed number:");
    scanf("%f", &seed);
    srand(seed);

open_out_scorefile(argc);

/* RANDOMLY SET UP ALL PEAK POINT VALUES */

for (count=0;count<23; count++)
{
    peakpoints[count][0]=myrand(earliestpeak1,latestpeak1);
    /* above is peak time 1 */
    peakpoints[count][1]=myrand(500,1000);
    /* the above is choosing a peak amplitude for peak
point1 */
    peakpoints[count][2]=myrand(11,17);
    /* the above chooses how many notes/beat at peak point1
*/
    peakpoints[count][3]=myrand(earliestpeak2,latestpeak2);
    /* above is peak time 2 */
    peakpoints[count][4]=myrand(750,1200);
    /* the above is choosing a peak amplitude for peak
point2 */

```

```

    peakpoints[count][5]=myrand(16,25);
    /* the above chooses how many notes/beat at peak point2
*/

    /* IF PEAK TIMES AREN'T IN CHRONOLOGICAL ORDER, SWAP THEM */
    if(peakpoints[count][3]<peakpoints[count][0])
    {
        temp=peakpoints[count][0];
        peakpoints[count][0]=peakpoints[count][3];
        peakpoints[count][3]=temp;
    }

    /* SET UP % OF NOTES THAT WILL BE RESTS */
    for(i=0;i<4;i++)
    {
        restprcnt[count][i]=myrand(0.0,.75); /* % of time notes
will be rests */
    }
}

/* WRITE TOP OF SCORE FOR CMIX WAVETABLE INSTRUMENT */

printf("What name do you want for the Cmix soundfile?\n");
printf("I'm assuming it will be in /sndh/dough/....just give
name: ");
scanf("%s", &sndfilename);
printf("\n\n");

fprintf(scorefile, /* p0 = start time\n");
fprintf(scorefile, "    p1 = duration\n");
fprintf(scorefile, "    p2 = amplitude\n");
fprintf(scorefile, "    p3 = pitch (oct.pc) */ \n\n");

fprintf(scorefile, "system(\"rm /sndh/dough/%s\")\n\n",
sndfilename);
fprintf(scorefile, "system(\"sffc2 /sndh/dough/%s\")\n",
sndfilename);
fprintf(scorefile, "output(\"/sndh/dough/%s\")\n\n",
sndfilename);

fprintf(scorefile, "makegen(1, 10, 1000, 1, 0.3, 0.2)\n");
fprintf(scorefile, "makegen(2, 24, 1000, 0, 0, 0.01,1, 0.1,0.2,
0.4, 0)\n");
/* fprintf(scorefile, "makegen(2, 24, 1000, 0, 0, 0.02,1,
0.1,0.2, 0.15,0.15, 0.4, 0)\n\n"); */
/* fprintf(scorefile, "/*\n"); */

```

```

/* printf("I'm starting output to scorefile.....\n\n"); */
for (octavenum=1; octavenum<16; octavenum++)
{
num=octavenum-2; /* THIS IS ACTUAL ITERATION NUMBER IN LOOP
*/
start=0;

/* OUTPUT SIGNIFICANT VALUES TO SCOREFILE */

fprintf(scorefile, "\n\n/* Octavenum=%d\n", octavenum);
fprintf(scorefile, "peak point one at time %f\n",
peakpoints[num][0]);
fprintf(scorefile, "peak point one max amp= %f\n",
peakpoints[num][1]);
fprintf(scorefile, "peak point one max density= %f\n",
peakpoints[num][2]);
fprintf(scorefile, "peak point two at time %f */ \n\n",
peakpoints[num][3]);

maxnotes=(int)(peakpoints[num][2]); /* # notes at peak 1*/
peaktimel=peakpoints[num][0];
peaktimel2=peakpoints[num][3];

percentnotes=peaktimel/maxnotes;
/* THIS VALUE IS HOW OFTEN TO ADD TO CURRENT TUPLET
LEVEL */

subsetpitches[0]=pitches[(int)myrand(0,6)];
subsetpitches[1]=pitches[(int)myrand(0, 6)];
/* printf("We are using pitches %.2f and %.2f
\n",subsetpitches[0], subsetpitches[1]);
for now, each octave will use only
2 pitches of the 7 possible */

tuplet=1; /* start with "quarter notes" */

/* PUT IN DELAYED START IN SOME VOICES */
if(myrand(0,1)>.5) start=start+(.5*beattime);

/* GIVE EACH VOICE A RANDOM PANNING LOCATION TO BEGIN AT */
pan=myrand(0,1);
/* RANDOMLY SET A PANNING DIRECTION */
temp=myrand(0,1);

```

```

if (temp<.49)
{
    pandir=1;
}
else
{
    pandir=-1;
}

/* MAKE NOTES LEADING UP TO FIRST PEAK POINT */

while (start<peaktimel)
{

    percentime=start/peaktimel;
    /* printf("percentime=%f\n", percentime); */

    if (percentime>1.0)
    {
        printf("Error in percent calc at peaktimel!\n");
        exit(1);
    }

    amp=peakpoints[num][1]*percentime;

    if (amp<100) amp=100;

    if (start>percentnotes)
    {
        percentnotes+=percentnotes;
        tuplet++;
    }

    notedur=beattime/tuplet;
    /* DUR TIME FOR EACH NOTE OF THE TUPLET */

    for (note=0;note<tuplet;note++)
    {
        temp=restprcnt[num][0];
        if (myrand(0,1)>temp) /* if <temp, it is a rest */
        {

            /* CHOOSE PITCHES AT RANDOM */
            randnum=myrand(0,2);
            pitch=subsetpitches[(int)randnum];
            pitch=pitch+octavenum;
        }
    }
}

```

```

        /* INCREMENT PANNING */
        if(pandir==1)
        {
            pan=pan+.1;
            if(pan>.9) pandir=-1;
        }
        else
        {
            pan=pan-.1;
            if(pan<.1) pandir=1;
        }

        playcmixwtable(start, notedur, amp, pitch, pan);
        }
        else
        {
            fprintf(scorefile,"/* rest */\n");
        }

        start=start+notedur;
    }

} /* END OF START 1 LOOP */

/* MAKE NOTES FALLING OFF FROM FIRST PEAK POINT */
midtime=peakttime2-peakttime1; /* HOW MUCH TIME UNTIL PEAK #2
*/
midpoint=peakttime1+(midtime/2); /* 1/2 way through midtime */
/* printf("\n midpoint 1 = %f \n\n", midpoint); */

percentnotes=(midtime/2)/tuplet;
    /* THIS VALUE IS HOW OFTEN TO SUTRACT FROM CURRENT
        TUPLET LEVEL */

while(start<midpoint)
{
    percentime=start/midpoint;
    /* printf("percentime=%f\n", percentime); */

    if(percentime>1.0)
    {
        printf("Error in percent calc at midpoint 1!\n");
        exit(1);
    }
}

```

```

    amp=peakpoints[num][1]-
((peakpoints[num][1]*.66)*percentime);

    if(amp<100) amp=100;

    if((start-peaktime1)>percentnotes)
    {
        percentnotes+=percentnotes;
        tuplet--;
        if(tuplet<1)
        {
            printf("tuplet error in midpoint 1 loop!\n");
            exit(1);
        }
    }

    notedur=beattime/tuplet;
    /* DUR TIME FOR EACH NOTE OF THE TUPLET */

    for(note=0;note<tuplet;note++)
    {
        temp=restprcnt[num][0];
        if(myrand(0,1)>temp) /* if <temp, it is a rest */
        {
            /* CHOOSE PITCHES AT RANDOM
            NOTE: in this section, it picks from ALL 7 PITCHES! */
            randnum=myrand(0,7);
            pitch=pitches[(int)randnum];
            pitch=pitch+octavenum;
            /* fprintf(scorefile, "/* pitch=%f \n", pitch); */

            /* INCREMENT PANNING */
            if(pandir==1)
            {
                pan=pan+.1;
                if(pan>.9) pandir=-1;
            }
            else
            {
                pan=pan-.1;
                if(pan<.1) pandir=1;
            }

            playcmixwtable(start, notedur, amp, pitch, pan);
        }
    }

```

```

        else
        {
            fprintf(scorefile,"/* rest */\n");
        }

        start=start+notedur;
    }

} /* END OF START 2 LOOP */

/* RESET VALUES FOR CLIMB TO SECOND PEAK POINT */
maxnotes=(int)(peakpoints[num][5]); /* # notes at peak 2*/

percentnotes=(midtime/2)/maxnotes;
/* THIS VALUE IS HOW OFTEN TO ADD TO CURRENT TUPLET
LEVEL */

subsetpitches[0]=pitches[(int)myrand(0,6)];
subsetpitches[1]=pitches[(int)myrand(0, 6)];
subsetpitches[2]=pitches[(int)myrand(0, 6)];
/* printf("We are using pitches %f and %f \n"); */
/* for now, each octave will use only
   3 pitches of the 7 possible */

tuplet=1; /* start with "quarter notes" */

/* MAKE NOTES LEADING UP TO SECOND PEAK POINT */

while(start<peakttime2)
{
    percenttime=start/peakttime2;
    /* printf("percenttime=%f\n", percenttime); */

    if(percenttime>1.0)
    {
        printf("Error in percent calc at peakttime 1!\n");
        exit(1);
    }

    amp=peakpoints[num][4]*percenttime;

    if(amp<100) amp=100;

    if((start-midpoint)>percentnotes)

```

```

    {
        percentnotes+=percentnotes;
        tuplet++;
    }

    notedur=beattime/tuplet;
    /* DUR TIME FOR EACH NOTE OF THE TUPLET */

    for(note=0;note<tuplet;note++)
    {
        temp=restprcnt[num][0];
        if(myrand(0,1)>temp) /* if <temp, it is a rest */
        {
            /* CHOOSE PITCHES AT RANDOM */
            randnum=myrand(0,2.9999);
            pitch=subsetpitches[(int)randnum];
            pitch=pitch+octavenum;
            /* fprintf(scorefile, "/* pitch=%f \n", pitch); */

            /* INCREMENT PANNING */
            if(pandir==1)
            {
                pan=pan+.1;
                if(pan>.9) pandir=-1;
            }
            else
            {
                pan=pan-.1;
                if(pan<.1) pandir=1;
            }

            playcmixwtable(start, notedur, amp, pitch, pan);
        }
        else
        {
            fprintf(scorefile,"/* rest */\n");
        }

        start=start+notedur;
    }

} /* END OF START 3 LOOP */

/* MAKE NOTES FALLING OFF FROM SECOND PEAK POINT */
timetogo=totaltime-peakttime2; /* HOW MUCH TIME UNTIL END */

```

```

percentnotes=timetogo/tuplet;
    /* THIS VALUE IS HOW OFTEN TO SUTRACT FROM CURRENT
        TUPLET LEVEL */

while(start<totaltime)
{
    percenttime=start/totaltime;
    /* printf("percenttime=%f\n", percenttime); */

    if(percenttime>1.0)
    {
        printf("Error in percent calc after 2nd peak
point!\n");
        exit(1);
    }

    if((start-peaktime2)>percentnotes)
    {
        percentnotes+=percentnotes;
        tuplet--;
        if(tuplet<1)
        {
            printf("tuplet error in loop after peak 2 at time
%f\n", start);
            tuplet=1;
        }
    }

    notedur=beattime/tuplet;
    /* DUR TIME FOR EACH NOTE OF THE TUPLET */

    for(note=0;note<tuplet;note++)
    {
        temp=restprcnt[num][0];
        if(myrand(0,1)>temp) /* if <temp, it is a rest */
        {
            /* CHOOSE PITCHES AT RANDOM
NOTE: in this section, it picks from ALL 7 PITCHES! */
            randnum=myrand(0,7);
            pitch=pitches[(int)randnum];
            pitch=pitch+octavenum;
            /* fprintf(scorefile, "/* pitch=%f \n", pitch); */

            /* INCREMENT PANNING */
            if(pandir==1)
            {

```

```

        pan=pan+.1;
        if(pan>.9) pandir=-1;
    }
    else
    {
        pan=pan-.1;
        if(pan<.1) pandir=1;
    }

    playcmixwtable(start, notedur, amp, pitch, pan);
    }
    else
    {
        fprintf(scorefile,"/* rest */\n");
    }

    start=start+notedur;
}

} /* END OF START 4 LOOP */

} /* END of OCTAVENUM loop */

/* CLOSE OUTPUT FILE */
fclose(scorefile);

} /* end of MAIN */

/* THIS FUNCTION WRITES A CMIX WAVETABLE NOTE TO THE SCOREFILE */
void playcmixwtable(float start, float dur, float amp, float
pitch, float pan)
{
    if(amp>0.0) fprintf(scorefile,"wavetable(%.4f, %.4f, %.4f,
%.4f, %.4f)\n",start,dur,amp, pitch, pan);

    if(amp==0.0) fprintf(scorefile, "/* (rest) */ \n");
}

```

```

/* THIS ROUTINE IS THE RANDOM
NUM□                                BER GENERATOR */

myrand(float lobound, float upbound)
{
    float thenum;
    float range;

    range = upbound - lobound;
    thenum = (((float)random()/2147483646.0) * range) + lobound;
    /* NOTE: this I put in this if statement to prevent any
negative
return values --doug. */
    if(thenum<0)
        thenum=0-thenum;
    return(thenum);
}

/* FUNCTION TO OPEN OUTPUT TO SCOREFILE */

void open_out_scorefile(int argc)
{
    char answer2[1];

    /* GET NAME FOR OUTPUT SCORE FILE */

    if(argc<2)
    {
        printf("\nWhat name do you want for the output score
file?");
        scanf("%s", &outfilename);
    }

    /* FIRST, OPEN OUTPUT SCORE FILE */

    if((scorefile=fopen(outfilename, "r"))==NULL)
    {
        scorefile=fopen(outfilename, "w");
    }
    else
    {
        printf("The output score file you have specified already
exists--overwrite?");
    }
}

```

