Logistic Chaos

Another application of the figure-ground reversed manner of drawing a function graph you saw in `sinewheel` is this introduction to logistic chaos. You may not be familiar with the mathematics behind this progam, so we shall start with that.

Consider a population of something, say mosquitos in a swamp, which grow at a rate ρ. That is, the next generation of mosquitos is directly proportional to the current generation, $y = \rho x$. By substitution you can calcuate that successive populations, starting with x_0, comprise the geometric sequence,

$$x_0,\ \alpha x_0,\ \alpha^2 x_0,\ ...$$

which has one of three equally dull outcomes. If $\alpha \neq 1$, the mosquito population either dies out or increases indefinitely.

Since the latter is unlikely, we improve the model a tiny bit. Let us agree that there is a maximum possible population determined by the food available in the swamp. This is called the carrying capacity of the environment. We take this to be 1, which automatically makes our population variable x a fraction of the *carrying capacity.* Thus normalized, we obtain a manageable geometric problem. We can express the effect of nearing the population limit by making the reproductive rate ρ itself proportional to how close we are to 1. Thus our equation becomes

$$y = 4 * \rho(1 - x)x\ .$$

This is called a logistic growth model. The factor 4 is chosen so that the proportionality constant α exactly measure the maximum altitude of this parabolic arch which touches the x-axis at 0 and 1.

You can discover some rudimentary properties of this system by drawing pictures accurately with pencil, paper and a right angled ruler. The trick is to plot the values of the logistic function not along the x or y-axes, but along the diagonal of the unit square. For example, starting from the input value $x = .33$ roughly a third of the way along the diagonal, find the point above or below on the parabola, and proceed horizontally to the output value $f(x)$. You are now ready to repeat the procedure forever, generating a discrete dynamical system based on the feedback loop $x \leftarrow f(x)$.

Exercise 15. With pencil, paper and ruler check that the population must become extinct for $\alpha < \frac{1}{4}$. For then the parabola remains below the diagonal, and every orbit steps its way to oblivion. Next convince yourself that for $\frac{1}{4} < \alpha < \frac{1}{2}$ the population converges to a single, positive, steady-state value, no matter where it begins. But past $\alpha > \frac{1}{2}$ life is not so simple.

basicBASIC

We can speed up our graphical investigations with a computer. First we describe the essential part of our visualization, in `basic`. Then we describe an interactive extension, but now in `basiCglut`, which faithfully emulates the way it was origi-

nally written in `basic`. This gives a glimpse of the evolution of computer graphics languages.[1]

```
5 REM CHAOS
10 CLS
20 DATA 63, 63, .5, .01
21 READ XM, YM, A, D
90 X = .9
100 REM ETERNAL LOOP
140 Y = 4*A*(1-X)*X
150 XX=X*XM : YY = Y*YM
155 XY=X*YM : YX = Y*XM
160 LINE (XX,XY)-(XX,YY),1
165 LINE (XX,YY)-(YX,YY),1
170 X=Y : REM NEW REPLACES OLD
180 GOTO 100 : REM FEEDBACK
199 END
```

In order to play with this program, i.e. to make it interactive, you might proceed as follows. The line

```
25 INPUT "A= "; A
```

asks you for a value of `A`.[2] Can you change line 90 to accept a user chosen initial value?

Exercise 16. Apply the skills you have gained by completing the exercise for the `sinewheel` program to add a subroutine to your chaos program which draws the graph of the parabola. This time you do not want a fancy way of graphing the function, the dynamical system already uses this device. Here is a hint. Enter these lines.

```
10
30 INPUT "GRAPH? (Y?N) "; A$
35 IF A$="Y" THEN GOSUB 200 ELSE CLS
200 REM GRAPHIT
210 CLS
299 RETURN
```

The blank line 10 erases the clear-screen command, saving it for when you really want it at 35 or 210. The line 299 returns the program to the place from where the subroutine starting at line 200 was called. Of course, you need to write the parabolic graphing routine and perhaps the unit box and diagonal in the 200's.

basiCglut

We now turn to an example of how a pocket program can mature into a fairly useful mathematical tool. The following program is affectionately called *Allerton* after a conference in Allerton Park.[3]

[1] The biological principle that "ontogeny recapulates philogeny" says that the stages an embryo of an animal of passes through mimics the stages of the animal's evolution. Perhaps this principle applies elsewhere too?

[2] Incidentally, the keypress CONTROL-C will interrupt your program, and `RUN` will run it again. Of course, there are more elegant ways of controlling your program, even in BASIC.

[3] The conference on using computers in mathematics instruction was in the early eighties.

We shall analyse this code in the order it is written in the program. You should be running the program on a computer while reading this. As an exercise, you could design a tutorial which teaches the student the mathematicasl story *using* Allerton without going into the computational details.

```
/* Logistic Chaos in GLUT, gkf 22jan2K on linux, 10aug on windows */
/* translated into basiCglut form the original Applesoft BASIC */
/* revised 26nov2K and 5jan01 */
#include <stdlib.h>
#include <stdio.h>
#include <gl\glut.h>
#include <windows.h>
#include <math.h>
#define  WIN      640           /* size of the square window */
#define  NAP      100           /* microseconds for ell() only */
#define  SLEEP(u) usleep(u)     /* usleep() - inux, sginap() - irix */
#define  FOR(i,a,b) for(i=a;i<b;i++)
int ii, jj, kk; float tmp, temp;
float xmax = WIN , ymax = WIN ,      /* screen size */
      x =.9, y = .9,  A = .99 ,      /* world variables */
      xx, yy, xy, yx,                /* screen variables */
      alt = .96,                     /* altitude of parabola */
      pnt = .9 ,                     /* starting point */
      del = .01 ;                     /* graph step size */
int clr =0,                           /* color index */
    til = 10,                         /* run ells til */
    nth = 1;                          /* iterate */
float rainbow[8][3]={ {1.0, 0.0, 0.0},        /* red */
                      {1.0, 0.5, 0.0},        /* orange */
                      {0.8, 0.8, 0.0},        /* yellow */
                      {0.0, 1.0, 0.0},        /* green */
                      {0.0, 8.0, 0.8},        /* (blue) cyan */
                      {0.0, 5.0, 1.0},        /* indigo */
                      {1.0, 0.0, 1.0},        /* (violet) magenta */
                      {1.0, 1.0, 1.0} };      /* white */
int hasnumber, number, decimal, sign ;  /* SLevy  bump gadget */
/*********************************************************************/
void usleep(int nap){int ii; for(ii=0;ii< nap; ii++);} //a bad kludge
/*********************************************************************/
```

This was the time when the University of Illinois began to switch form its pioneering PLATO system to microcomputers. Although I had translated this example from PLATO to Applesoft, I presented only the transparencies. A colleague brought an Apple computer to the conference. He convinced me once and forever that a single live computer demo is worth a hundred slides and transparencies.

The preamble of this program is pretty much self-explanatory. The abbreviation `FOR(i,a,b)` (starting a counted loop) is easier to type than its more versatile counterpart in C. It is also less likely to be mistyped. The `rainbow` contains an approximation to those colors. You can tweak these RGB values to make the colors look better.

This program introduces Stuart Levy's gadget for writing into the graphics window and reading entries made there. It mimics the command lines at the bottom of the Applesoft high resolution screen. We will see how it works later.

From the backslash, `<gl\glut.h>`, and the `windows.h` header file in the includes you can see this is the VC98 version of Allerton. Because nobody could find a counterpart to the Linux pause function, `usleep()` , it was redefined as a longish loop.[4] it can be quickly replaced or removed.

The function definitions in Allerton begin with the logistic equation. Here you can write different functions, perhaps using a switch-case statement and a key-controlled choosing gadged between different functions.

```
/*********************************************************************/
float func(int nth, float x){
      FOR(ii,0,nth)x= 4 * alt*(1-x)*x; return x;
}
/*********************************************************************/
void  drawell(void){
      y=func(nth,x);     /* evaluate */
      xx = x*xmax ; yy = y*ymax;    /* world to screen coords */
      xy = x*ymax ; yx = y*xmax;
      glBegin(GL_LINE_STRIP);
      glVertex2f(xx,xy); glVertex2f(xx,yy); glVertex2f(yx,yy);
      glEnd();
      x = y;              /*feedback*/
      SLEEP(NAP);
}
/*********************************************************************/
void frame(void){  /* box with diagonal */
    int bot=1, top=WIN;
    glBegin(GL_LINE_STRIP);
    glVertex2f(top,top); glVertex2f(bot,top); glVertex2f(bot,bot);
    glVertex2f(top,top); glVertex2f(top,bot); glVertex2f(bot,bot);
    glEnd();
}
/*********************************************************************/
void wipe(void){glClear(GL_COLOR_BUFFER_BIT); glClearColor(0.,0.,0.,0.);
 }
```

[4] This is an example of a "hack" involving an inelegant solution, or "kludge".

```
/*********************************************************************/
void graph(void){ float x;   glColor3fv(rainbow[clr]);
     glBegin(GL_LINE_STRIP);
     for(x=0; x < 1+del; x += del)
         glVertex2f(x*xmax, func(nth,x)*ymax);
     glEnd();
}
/*********************************************************************/
void run(void){ glColor3fv(rainbow[clr]); FOR(jj,0,10){drawell();} }
/*********************************************************************/
```

Type the keys for (W)ipe, (F)rame, (G)raph, el(L), and (R)un to see what the next four functions do. Look at the code to see why they do it. These four are also marked on the (H)elp button.

The (J) is next to the (H) and brings up the six gadgets that accept your input to change the value of their parameters. To make this work from *inside* the graphic window takes some explaining. Some functions we describe next are *advanced* and some *elementary*.

```
/*********************************************************************/
float getnumber(float dflt){      /* return new or default number  */
      if(!hasnumber)return dflt;
      tmp = sign ? -number : number;
      return decimal>0 ? tmp/(float)decimal : tmp ;
}
/*********************************************************************/
void graffiti(char strng[128], float par){  /* from avn by SLevy */
     char buf[128], *p ;
     glColor3f(0,0,0); glRecti(5,3,WIN,20);  /* erase old graffiti */
     glColor3fv(rainbow[clr]);  /* create and draw new graffiti */
     sprintf(buf, strng, par);
     glRasterPos2i(5,5);
     for(p=buf; *p; p++)glutBitmapCharacter(GLUT_BITMAP_8_BY_13, *p);
}
/*********************************************************************/
void altit(void) {
      alt= getnumber(alt); graffiti("%0.3f=(A)lt", alt); }
void point(void){
   x=pnt = getnumber(pnt); graffiti("%0.3f=(P)nt",pnt); }
void delta(void){
     del = getnumber(del); graffiti("%0.3f=(D)el",del); }
void until(void){
     tmp = getnumber((float)til); graffiti("%f=(T)il",tmp); til=(int)tmp; }
void power(void){
```

```
   tmp = getnumber((float)nth); graffiti("%f=(N)th",tmp); nth=(int)tmp;}
void color(void){
   tmp = getnumber((float)clr); graffiti("%f=clr",tmp); clr = (int)tmp; }
#if 0
void help(void){ fprintf(stdout,
     "(ESC)ape (w)ipe (f)rame (g)raph el(l) (r)un (h)elp \n \
      (a)ltitude (p)oint (d)elta (n)th (t)il (c)olor \n ",0);
#endif
void helpH(void){
     graffiti("(ESC)ape (W)ipe (F)rame (G)raph el(L) (R)un (H)(J)",0); }
void helpJ(void){
     graffiti("(A)ltitude (P)oint (D)elta (N)th (T)il (C)olor",0);}

/*********************************************************************/
```

Both `getnumber()`and `graffiti()` are advanced.[5] They are factors of the next six gadgets, which have similar structure. For example, `alt` is the altitude of the parabolic arch. Being a parameter, it is a global variable, used by several functions independently. First, `alt = getnumber(alt);` assigns a new value to the altitude if you typed one in, and otherway keeps the old value by default. Secondly, put some graffiti on the wall.

Now, please experiment with the (A)ltitude, the initial (P)oint, the stepsize (D)elta for the graph, the (N)th power the logistic is iterated before the next position is drawn. Check out the interplay of (N) and (G). There is another loop variable affecting a (R)un of el(L)s un(T)il you said s(T)op. The (C)olor gadget is even trickier. Not only can you tell it which color to use, but pressing the key repeatedly cycles the colors to help you tell a story with Allerton.

Finally, there are three helpers. The original `help()`is commented out in a peculiar way. The compiler is directed to ignore the code between `#if` and `#endif` because the "0" is always false.[6] If you don't like graffiti and want to write to the command window instead of the graphics window then use this instead of the other helpers. In Unix the `fprintf(stdout ...` works normally. In Windows you'll have to experiment.

You have already met the `keyboard()` in prior pocket programs. But this one is a step more sophisticated. For one, it has the rest of Stuart Levy's gadget. For another it chooses the display callback function for the GLUT library at your command.

```
/*********************************************************************/
void cycle(int *par, int bas){
```

[5] Their operation but not their C-structure is discussed anon.

[6] This is a useful abuse of the compiler directives.

```
     if(hasnumber){*par = getnumber(0); return;} /* the 0 is a herring */
     *par = (*par + 1)%bas ;
}
/*****************************************************************/
void keyboard(unsigned char key, int x, int y){
#define PLOT(foo) glutDisplayFunc(foo); glutPostRedisplay();
  switch(key){
     case  27: fprintf(stderr," Thanks for using GLUT ! \n"); exit(0); break;
     case 's': {exit(0); break;}; /* stop */
     case 'w': {wipe(); break;};
     case 'f': {PLOT(frame); break;};
     case 'g': {PLOT(graph); break;};
     case 'l': { glColor3fv(rainbow[clr=(clr++)%7]);  /* cycle colors */
                 PLOT(drawell); break;};
     case 'r': {PLOT(run); break;};
     case 'h': {helpH(); break;};
     case 'j': {helpJ(); break;};
     case 'a': {altit(); break;};
     case 'p': {point(); break;};
     case 'd': {delta(); break;};
     case 'n': {power(); break;};
     case 't': {until(); break;};
     case 'c': {cycle(&clr,7); color(); break;};
   }
   glFlush(); /* superstition ? */
   /* Stuart Levy's gadget parser from avn.c 1998 */
   if(key >= '0' && key <= '9'){ hasnumber = 1;
         number = 10*number + (key - '0'); decimal *= 10;}
   else if(key == '.'){ decimal = 1;}
   else if(key == '-'){ sign = -1;}
   else {hasnumber = number = decimal = sign = 0;}
}
/*****************************************************************/
```

The first function is again advanced and is used for the (C)olor cycler. In the keyboard function we define a macro `PLOT()`which does two things. It hands the GLUT-library to a new display funcation to "call back" when appropriate. And it tells the GLUT-library to update itself, in case its fallen asleep.

The switch-case construction which turns your key-presses into action, can pretty much be puzzled out. Note the two ways of writing a cycling gadget.[7] The rest is advanced wizardry.

[7] In computer graphics there is a well-defined notion of a *widget* in a graphical user interface (GUI). A gadget is a poor man's widget. It gets the job done with a minimum of fuss, not a minumum of inconvenience. We will discuss gadgets at greater lengths in the second part of the course.

We now come the part that uses the GLUT-library.

```
/*********************************************************************/
void nothing(void){  } /* Glut3 forbids glutDisplayFunc(NULL); */
/*********************************************************************/
int main(int argc, char **argv){
  glutInitWindowSize(WIN, WIN);
  glutInitWindowPosition(10,10);
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_RGB);
  glutCreateWindow("<< Logistic Chaos in GLUT >>");
  glutDisplayFunc(nothing);
  glutKeyboardFunc(keyboard);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glOrtho(0,WIN,0,WIN,-10.0,10.0);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  glutMainLoop();
  return 0;             /* ANSI C requires main to return int. */
}
```