EcoDraw

A graphic ecosystem simulator

ecoNode.c was written using command line routines. I took its main routine and the supporting functions to create a threaded routine. This is joined by the display thread and a rudimentary UI thread. Now I can see what I have only been imagining. Growth patterns appear from the underlying algorithms. Showing the nutrient layer was very interesting; it displays the energy history of the ecosystem. I can see energy being extracted by swaths of plants. When they are eaten by herbivores the energy level rises. Then, when the carnivores attack the herd, more energy is added to the nutrient layer.

I Discuss new data structures, algorithms, or language features

One new thing is the display thread. It updates the ecosystem's bitmap inside of a scrolling window. The client window can be as large as the monitor will allow, but it is not large enough to display the entire 5000 x 5000 unit simulation space.

I added another bitmap to display a few lines of data. The ecoNode thread and the display thread are only tied together through the PL, HL, CL, and E buffers. ecoNode writes to them each cycle, display reads those buffers and draws to the background bitmap. If data == TRUE then population data is written to the data bitmap. A call to WM_PAINT copies the background bitmap onto the foreground display. Then, if data is TRUE, the data bitmap is copied on top of the current display. SRCINVERT is the most accurate ROP for this operation. During each write I need to blank the data bitmap so I fill it with black. Then draw the data text onto the data bitmap with my chosen color. Invert that onto the current display. That way the data is not overwritten at each cycle and the display seems to float on top of the current ecosystem display. Herbivores tend to 'eat' the data if they pass underneath due to the invert routine. But, I can live with that for now. Currently the display is in a color which is the inverse of the ecosystem display. It is yellow and the text is a shade of violet.

I tried to write the data onto the ecosystem bitmap. But, since they are not synchronized I got some oddly placed, and mistimed population information. So, I added a separate, smaller bitmap to hold the population data.

Trying to find the correct mono-font has proven difficult. I tried using "Courier" which helped but a generic FIXED_FONT, "" worked better. However, I have this application running on Leto, Loki, and Hephaestus and the font is not the same on any of them. Loki has the best output with larger, bold letters in a mono-font. Hephaestus has a mono-font which is about half the size of that on Loki. It is also offset farther from the right edge of the display. I think the latter is due to the smaller font size. On Leto the display is similar to that on Hephaestus but well it looks like those two may be the same except for the size of the display. I'm not sure why Loki has the larger font but Hephaestus and Loki to not. Dionysus is still working on the 250,000 generation ecoNode data file. Once that has completed some time today I will start ecoDraw running on it. I changed to "Liberation Mono" on Leto and got it looking great. This same scheme works great for Hephaestus too. But NOT for Loki. Ack! It does have Liberation Mono in its font folder. I am using Liberation Serif to write this on Loki :(I am confused.

II Describe functions and modules

```
   // Main functions  ////////////////////////
void initEcosystem( void );                  // Initialize ecosystem framework
void initPlants( void );
void initHerbivores( void );
void initCarnivores( void );
void moveHerbivores( void );                 // Call move() on each herbivore
void moveCarnivores( void );                 // Call move() on each carnivore
float metabolize( void );                    // metabolize()
void replenish( float );                     // replenish() nutrient layer in the E buffer
void eat( void );                            // Absorb nutrients, eat plants or herbivores
void reproduce( void );                      // reproduce() if you have the energy and age
void die( void );                            // die() from energy loss
void cull( void );                           // Bring out your dead
void addNew( void );                         // Catenate new organisms to their lists
void bufferSwap( void );                     // Swap even/odd buffers

   // Useful tools  //////////////////////////
struct vt vector( int, int );                // 2D integer vector for cell or pixel use
struct vt sum( struct vt, struct vt );       // Add 2D vectors
struct vt bound( struct vt );                // Place 2D vector into toroidal space
struct vt scale( int, struct vt );           // Find an open cell around current location
float fRan( void );                          // Floating point random number generator
void initDir( void );                        // Initialize the direction vector buffer
struct vt move( chromosome );                // Use motion genes to relocate animals
chromosome crossover( chromosome, chromosome );
chromosome mutation( chromosome );

   // Animal sensor array functions  //////////
void herbScan( void );                       // Local sensor scan for plants and carnivores
void carnScan( void );                       // Local sensor scan for herbivores
int maxHerb( int CS[] );                     // Find best feeding vector

   // Data Logging functions  /////////////////
void storeData( void );                      // Backup ecosystem to file
void loadData( void );                       // Restore ecosystem from file
void storeSamples( void );                   // Collect 1000 P,H,C samples
void loadSamples( void );                    // Read the samples into their own buffers
void loadSampTwo( void );                    // Read second sample into secondary buffers
void useSamples( void );                     // Build an ecosystem from those samples
void stamp( void );                          // Add timestamp to terminal window
```

Almost all of these functions are used in the ecoNode thread. So they were described in the ecoNode document. These new functions are all related to the graphics routines of Windows.

```
void initPalette( void );                    // Initialize Blue Red Yellow White palette
int colorTransform( int, int, double, double, double ); // Transform data range to color
void drawFrame( void );                      // Draw nutrients, plants, etc to background
void displayData( int );                     // Draw text to data bitmap
DWORD WINAPI display( LPVOID );              // Display thread
DWORD WINAPI ecoNode( LPVOID );              // Ecosystem thread
INT WINAPI WinMain( HINSTANCE, HINSTANCE, LPSTR, INT );
LRESULT CALLBACK WndProc( HWND, UINT, WPARAM, LPARAM);
```

III Debug and Profile the code

How about we change this section to testing and debugging? Such as all the routines I wrote in ecoNode to test functions as I went along. Just because I do not include error checking in my functions does not mean I did not perform error checking. I simply use external functions. I do include a lot of printf statements throughout the code as I build it. Then I remove them when the testing is done. I do so to let me read the code more easily. All the extraneous lines of code get in the way. So I have replaced those lines with external functions which perform the same tasks more efficiently and thoroughly.

IV Describe code line by line

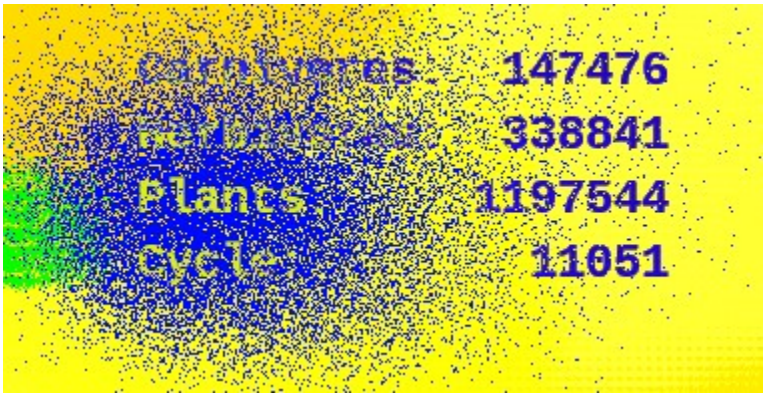
```
void initPalette( void );                    // Initialize Blue Red Yellow White palette
int colorTransform( int, int, double, double, double ); // Transform data range to color
void drawFrame( void );                      // Draw nutrients, plants, etc to background
void displayData( int );                     // Draw text to data bitmap
DWORD WINAPI display( LPVOID );              // Display thread
DWORD WINAPI ecoNode( LPVOID );              // Ecosystem thread
INT WINAPI WinMain( HINSTANCE, HINSTANCE, LPSTR, INT );
LRESULT CALLBACK WndProc( HWND, UINT, WPARAM, LPARAM);
```

These files were not used in ecoNode.c nor described in ecoNode.odt

VI Deeper

ROPs and bitmaps and data overlays.

Here is where I implemented and tested a number of raster operations (ROPs). I found SRCINVERT fit my needs for the floating data display. If the data is in the same thread I can use SRCAND to get the best results. I filled the data bitmap with black then wrote the data text. When I blitted it with the existing bitmap I used SRCINVERT. So the text will display as the color inverse of the current display. This works well most of the time. When a herd of herbivores passes underneath the numbers they get 'eaten' :) The effect is transitory but disconcerting.



```
BitBlt( hdc , 0, 0, cxClient, cyClient,
        bkhdc, iHscrollPos, iVscrollPos, SRCCOPY );

if( data )                                  // Display population data
   BitBlt( hdc, cxClient-350, 0, cxClient, cyClient,
   hdcdata, 0, 0, SRCINVERT);               // Notice source DC
// hdcdata, 0, 0, MERGEPAINT);
// hdcdata, 0, 0, NOTSRCCOPY);
// hdcdata, 0, 0, NOTSRCERASE);
// hdcdata, 0, 0, SRCAND);
// hdcdata, 0, 0, SRCCOPY);
```

There are three threads of execution in the application. A rudimentary, keyboard user interface, the ecoNode thread which runs the ecosystem, and the display thread which portrays four ecosystem layers on the screen.

```
wc.lpfnWndProc = (WNDPROC) WndProc;          // User Interface thread

DWORD displayThreadID;                       // Graphics thread
displayThreadHandle = CreateThread( 0, 0, display, 0, 0, &displayThreadID );

DWORD ecoThreadID;                           // Ecology thread
ecoThreadHandle = CreateThread( 0, 0, ecoNode, 0, 0, &ecoThreadID );
```

I did not coordinate the threads. The UI thread takes keyboard controls to scroll the display area around the bitmap (5000x5000) and a few other commands. The ecosystem thread runs ecoNode in a continuous loop. If any organism population drops below 2 then the simulation ends. It manipulates numbers which live in the PL, HL, CL, and E buffers. These numbers represent plants, herbivores, and carnivores living in the ecosystem framework. The display layer accesses the information held in the buffers and displays it. The nutrient layer is portrayed in a color palette ranging from a dusky orange to white. Brighter colors signify more energy. There is a noticeable darkening in color after a swath of plants eat their way across the nutrient layer. Then, when the herbivores eat plants or carnivores eat herbivores, the nutrient layer gets brighter reflecting the energy added by messy eating. The nutrient layer maintains this "history" for many cycles as new generations of organisms sweep across.

I have added genetic markers to some plants. Those markers are passed on by mothers discretely, there is no crossover for the GM chromosome. This lets me track plant populations I inject into the ecosystem with useSamples(). The function takes two sample populations and combines them 50/50. This could be under keyboard control too. Inject new organisms in the middle of a simulation run. Add organisms with specific characteristics and marker genes on GM. Sample sets are 1000 plants, herbivores, and carnivores chosen at random from a stable ecosystem. useSamples() should work with one or more sample sets. Currently it initializes a uniform nutrient layer in E. Then it adds Ppop, Hpop, and Cpop of each organism in turn. I changed that to use sample set one for the first half of the population, and the second sample set for the last half. I could interleave them if that makes a difference :) However, the second data set is tagged with a specific genetic marker on the GM chromosome. Since GM is passed only through the mother a lineage can be inferred.

The application can start in one of three ways. It can initialize E, PL, HL, and CL manually and run from there. It takes a number of attempts to achieve a 'stable' system. This is where adding energy would be useful. Then, once the system is stable, decrease the amount you add. The second way to start is by loading the 'eco.dat' and running from there. eco.dat is created at 500 cycles and every 500 cycles afterward. The third method is to use sample data files. They are created every 250 cycles. They are also much smaller than eco.dat which makes them easier to exchange. I want to pursue a keyboard control of the energy added to the nutrient layer. High at first, then drop to zero above the base values. Maybe a toggle? High or not high so try adding:

```
if( high )
   (E+i)->nutrient += energy / (SZ*SZ) + 2.3 + HIGH_VALUE;
else
   (E+i)->nutrient += energy / (SZ*SZ) + 2.3;
```

That way I can control energy added with one key. Either nothing is added to the base value, or a set high value of energy is added to each cell. I wonder if the over population caused by too much energy will be able to survive the crash when I cut the energy stipend? Oh the humanity!!

nota bene: I implemented this scheme. It works too well. I added far too much energy so I quickly toggled it off. But not before 3 million carnivores swept the land. Herbivore population dropped to 53 before it recovered. Now both herbivore and carnivore populations are doing fine. And the energy added is at its default setting. I cut the length of time of added energy to within the first 30 cycles and the extra amount to 12.5 units. I have not tested those settings yet. The acid test will be a #define FIRST run. Hopefully the energy boost will get a even a recalcitrant ecosystem to flourish. {kjr December 9th, 2024}

I had wondered about a good background color for the ecosystem. I began writing graphics code from E(i,j)->nutrient values. Starting with the nutrient layer was an awakening. Once I had its color palette in the 'right' range, it made a nicely varied background which contrasts against every organism color. I had to raise the lower limit on the color palette to increase the contrast. The color palette has 768 colors in it. I chose a dusky orange (472) as the bottom, and white (768) as the top of my color range. I was wondering about using the black body radiation color scheme {Black Red Yellow White}. However, since I started at 472 there is no difference from the {Blue Red Yellow White} scheme I have written. I could remove both blackbody.h and solarized.h from this project. But, I dither :)

There are three logging systems right now. Every cycle I add population data to dt.dat. Every 250 cycles I take 1000 samples and store them into sample.dat. Every 500 cycles I store an image of the ecosystem to ecoXX.dat. To repeat: data is appended to dt.dat every cycle, sample.dat is overwritten every 250 cycles, and the image file ecoXX.dat is created each 500 cycles.

I use gnuplot to chart dt.dat. I use the index value in column 1 as the x value. Then use columns 2, 3, and 4 as the y values for plants, herbivores, and carnivores. That gives me a population chart for all three organisms so I can monitor the health of the ecosystem. gnuplot is a command line tool which is quite powerful. I only use a tiny bit of its power to give me a decent semi-log chart. I have the y axis display logarithmic coordinate values. That covers the often large disparity between plant and

carnivore populations. A log-log chart is also useful. It shows how the system changed over time. Once an ecosystem becomes "stable" the chart becomes less interesting. However, the nutrient layer background grows very complex. I think the variations induce a more dispersed population of plants and critters.

I made many design decisions along the way. Such as the direction vector array mimicking the moves of the queen in chess. Or creating populations from samples as compared to restoring simulation images. Some decisions are easy when I just don't care. Such as synchronizing the display thread with the ecosystem thread. During startup, say the first 20 cycles, the display thread cannot keep up with the updates being sent by the ecosystem thread. I could slow things down by forcing each frame to draw before allowing the ecosystem to continue. Why? Normally I am more concerned with getting the system to survive past 100 cycles. However, the two threads do work well together once the system becomes more stable. After 1000 cycles there are no glitches. Rather there are none except during logging. Creating a full image of the ecosystem takes a while. More than a few frames are not displayed. But once that is done we are in order again.

I wrote the eat() routines so they extract a percentage of what is available. That way the nutrient layer cannot be depleted below zero. After a plant, or a herbivore, is eaten there is excess energy. This returns directly to the nutrient layer. Reproduction is also messy; each participating organism loses some to the nutrient layer. However, when any organism dies, no energy is returned to the environment. I need to examine that assumption to see if plants have any energy left at senescence. nota bene { Yes, they do. It has now been added to the energy replenished to the nutrient layer. kjr December 15 }

I normally (almost always) run ecoDraw from the command line. Both ecoDraw and ecoNode, generate a log file to the screen as well as to the 'dt.dat' file. I display the cycle count and population levels for plants, herbivores, and carnivores. I display the maximum energy levels of each organism too. 'dt.dat' logs the number of plants and herbivores eaten, and the number of carnivores which died for each cycle instead of the energy levels. I use gnuplot to create a graph from 'dt.dat'. One terminal window is open running ecoDraw, while a second terminal window is open running gnuplot to make the graph. It is not automatic, I manually refresh the graph when necessary. Once again a don't care design decision. By using gnuplot I can graph my raw data quickly, with a minimum of extra code. Using two terminal windows, pointing at the very same place, lets me look at the data generated by a running application.

Compare ecoDraw with S4L5 Ecology

Differences between ecoDraw and S4L5_Ecology.

Lesson five used integer results, ecoDraw does not.

Much larger space. 5000 x 5000 as compared to 900 x 600

In eecology organisms could 'live' on the very same location. In ecoNode and ecoDraw I forced organisms to use an unpopulated location. I wrote this so two plants, or two herbivores, or two carnivores could not live in the same place. However, one plant, one herbivore, and one carnivore can live on the same spot. For a little while :) It depends upon the feeding cycle. The plant will draw energy from the nutrient layer. The plant is immediately eaten by the herbivore to increase its energy. The herbivore is immediately eaten by the carnivore. Eating is messy so some of the plant energy returns to the nutrient layer, as does some of the herbivore's energy when it is eaten. I wanted to make sure the organisms filled the space without overlap. Close neighbors feed and move to avoid each other. Their offspring are also placed where no others of their kind live.

I started writing my design with ecoNode, which had no display at all. I wanted to make sure my framework worked correctly before I added the graphics layer to it. ecoNode is a command line application. which can be profiled or debugged using standard gcc tools (gprof and gdb). This helped me find any bugs and also showed me why I was getting segmentation faults from large plant populations (> 8 million). I was asking for more space than I had allocated. I could increase the allocation but I put a hard limit on the population count instead. Recently, I have found little need of the limit since the system rarely grows over 2 million plants. Expressing more genes made life less harsh. I also added a little mutation to the mix which seems to stabilize the system too. The 250K+ run did not show any bad effects from prolonged mutation. I think crossover reinforces the system, while mutation explores more options. Using the two in concert keeps the system healthy.

I need to work on my sampling tools. Currently, loadSamples() adds a genetic marker to some plants. I could pause the simulation to examine the population. I could also use the pause to inject new organisms. If I mark them differently I could track population patterns. Did my new genes out perform the others or did they die out? Or, which of two populations is predominate after XX cycles.

Can I create an artificial population? One which I have 'hard wired' into existence? It would be interesting to inject populations of what I think are the best settings for all the genes on each chromosome or an organism. Then pause the run and sample the population. That begs for tools to display the data. A popup window display of graphs, or numeric data. Keyboard control to pause and toggle popups. Another thought: mark a specific 'herd' of herbivores or carnivores. Then track their existence. Use the lasso tool idea to capture them. Then mark the GM chromosome and change their display color. Currently the colors are:

```
for(int i=0; i<Ppop; i++)                  // Display plants
   SetPixel( bkhdc, (PL+i)->X.x, (PL+i)->X.y, RGB( 10, 250, 0 ));
for(int i=0; i<Hpop; i++)                  // Display herbivores
   SetPixel( bkhdc, (HL+i)->X.x, (HL+i)->X.y, RGB( 0, 10, 250 ));
for(int i=0; i<Cpop; i++)                  // Display carnivores
   SetPixel( bkhdc, (CL+i)->X.x, (CL+i)->X.y, RGB( 250, 0, 10 ));
```

I need a second red, green, and blue shade for the marked organisms. Then I can track them visually. The GM chromosome markings will help when I pause and sample. Need sample display tools for popup windows. A dark green would work. Cyan as compared to blue for herbivores? When the red and blue dots of carnivores and herbivores are close to one another they blur into violet. I don't think magenta would work for a second shade of red :(

One chart would be the percentage of marked organisms compared to unmarked. Or multiple values for a variety of markings. GM could hold more information than one simple integer. Look for certain genetic traits. Display the most energy rich organisms. Or during a pause display only those organisms with the chosen traits. Then the display can be your display :) Just draw your ideas to the background bitmap instead of the normal drawFrame() routines.

Keyboard display control. What can I modify or toggle?

Need an indication H has been pressed. Otherwise I will forget :)

I created a simulation bitmap much larger than any of my screens. I used code I had written years ago for showmap.cpp

Gene expression is more complicated. Integers, floating points, or factors.

Use chromosomes instead of genes. Chromosomes hold genes and I can create more of them.

Some of them are specific to plants, some to animals, some to all organisms.

More genes are expressed during eating, moving, breeding, dying, etc.

Energy flow

Display the nutrient level. I had no idea how much information is held here.

At least a two to three orders of magnitude larger population in ecoNode/Draw than in S4L5

Space is larger too. S4L5 lived in 2% of what it takes for ecoDraw.

eecology topped out at 166907 plants. ecoDraw currently has 1,059,045 plants. There is a hard limit set at 6 million plants. However, I have seen the plant population over 10 million without crashing. The herbivores take advantage of the feast while their population explodes. This provides plenty of energy to the carnivores which feast on the blue herds. All this feasting recycles energy to the nutrient layer so the plants can keep growing. I stopped ecoNode at over 250,000 cycles. I have archived the data from that run and use it to create new ecological systems.

Marked all PFA numbers as such. Now I can search on PFA when I want to tweak them, or when I find a way to remove the PFA label with facts, or reasonable assumptions.

Future Work List

Think about a start/stop mechanism. It sure would be nice to find out why I cannot restart an image (eco.dat) successfully. I would like to have extended runs of countless cycles. What is not being saved?

However, if we start the application before either the ecosystem or the display threads begin, we can add a few menu settings from dialog boxes. Let the user change the number of plants, herbivores, or carnivores. Tweak a few of the feeding parameters. Add more to the keyboard interface too. Currently I have D and H controlling the data display and added energy respectively. What else can be under user control? How many of my tweaks to get the system stable can be relaxed after 1000 cycles? I could add a pause button but I don't know why. Work on StretchBlt() a little more; it would be nice to have a larger picture of the ecosystem. Save the nutrient layer as a separate file. Restore it, then build a new ecology on top of it with useSamples(). I think a mature nutrient layer provides a better growth matrix. The plants are somewhat more plentiful but spread out more thinly. The herds of both herbivores and carnivores are also spread widely but with fewer members in each. I would like to test this hypothesis.

I also need to work on the sampling techniques. Currently I save 1000 randomly chosen members of the PL, HL, and CL buffers. Then fill the new ecosystem with Ppop, Hpop, and Cpop organisms. Unlike loadData(), useSamples() places the organisms randomly on a uniform nutrient layer. It also has its own Ppop, Hpop, and Cpop values.

There may be a way to write a few differential equations to mimic the behavior for this simulation. Replenishing energy is an example.

```
float metabolize( void )                        // Sum energy lost by all organisms
   {
   float energy = 0;                            // Balance global energy loss and gain

   for(int i=0; i<Ppop; i++)                    // EN::G5 energy use per cycle
      {
      (PL+i)->age += 1;                         //   EN::G5  7.5 - 14.5
      (PL+i)->energy -= (float) ((PL+i)->EN & 7) + 7.5;
      energy += (float) ((PL+i)->EN & 7) + 7.5;
      }
   for(int i=0; i<Hpop; i++)
      {
      (HL+i)->age += 1;                         //   EN::G5  4.2 - 11.2
      (HL+i)->energy -= (float) ((HL+i)->EN & 7) + 4.2;
      energy += (float) ((HL+i)->EN & 7) + 4.2;
      }
   for(int i=0; i<Cpop; i++)
      {
      (CL+i)->age += 1;                         //   EN::G5  5.3 - 20.3
      (CL+i)->energy -= (float) ((CL+i)->EN & 7) + 5.3;
      energy += (float) ((CL+i)->EN & 7) + 5.3;
      }
   return energy;                               // Pass energy used to replenish()
   }
```

```
void replenish( float energy )                  // Balance energy lost during metabolism
   {
   emin = 1e5;      emax = -1e5;                // Guard the endpoints

   for(int i=0; i<SZ*SZ; i++)
      {
      if( high )                                // Use the H key to toggle this
         (E+i)->nutrient += energy / (SZ*SZ) + 2.3 + HIGH_VALUE;
      else                                      // Spread energy evenly across ecosystem
         (E+i)->nutrient += energy / (SZ*SZ) + 2.3;

                                                // Determine extrema
      if( (E+i)->nutrient < emin ) emin = (E+i)->nutrient;
      if( (E+i)->nutrient > emax ) emax = (E+i)->nutrient;
      }
   }
```

metabolize() tells me how much energy was used by all of the organisms. This was easy since I was already collecting the high and low values of each organism. I just summed the data as I was collecting it. Passing it to replenish() was trivial.

```
replenish( metabolize() );                      // Balance energy out with energy in
```

So energy added is the sum of the sum of energy used by each plant, and the sum of the energy used by each herbivore, and by the sum of the energy used by each herbivore. I need to determine how much energy is being thrown away when an organism dies due to senescence. nota bene: I just implemented energy left over at senescence now flows to the nutrient layer.

I now have a partial equation for metabolic energy flow. Now to account for the senescence addition.

Another flow is from eating of plants or herbivores. EN::G3 is the satiety gene and EN::G4 is the absorption gene. That gives us a few equations for energy flow through plants, herbivores, and carnivores. Plants and herbivores can get eaten. They each metabolize and die of senescence if not from other means.

```
for(int i=0; i<Cpop; i++)
   {                                            // 39 - 70
   senescence = (int) (((CL+i)->EN & (31 << 22)) >> 22) + 39;
   if( (CL+i)->energy < 0 )
      (CL+i)->alive = false;
   if( (CL+i)->age > senescence )
      {
      (E +(CL+i)->X.y*SZ + (CL+i)->X.x)->nutrient = (CL+i)->energy;
      (CL+i)->alive = false;
      }
```

Here is how I prevent two of the same kind of organism in any cell. First during initialization, second from moving, and the third example is from the reproduction() routine.

```
for(int i=0; i<Cpop; i++)
   {
   do {
      X = rand()%SZ;                          // Generate prospective points
      Y = rand()%SZ;
      }  while( (E +Y*SZ +X)->C );            // Find your own lair

   (CL+i)->X.x = X;                           // Remember where I live
   (CL+i)->X.y = Y;
   (E +Y*SZ +X)->C = (CL+i);                  // Point at carnivore
```

How I implemented it during carnivore movement:

```
for(int i=0; i<Cpop; i++)                     // Scan carnivore list
   {
   ch = 0;                                    // Choice counter
   min = 3;                                   // { min <= speed <= 10 }
   speed = (int) (((CL+i)->CN & (7 << 28)) >> 28) + min;

   A = (CL+i)->X;                             // Current location
   do {                                       // Find an open spot within range
      ring = min + rand()%speed;              // {min <= ring < speed}
      B = scale( ring, move( (CL+i)->DR ) );  // Motion vector from DR chromosome
      C = bound( sum( A, B ) );               // Search location

   if( ch > 8 ) break;  else  ch++;           // All your choice are us
   }  while( (E +C.y*SZ +C.x)->C );           // Carnivore already lives here

(E +A.y*SZ +A.x)->C = NULL;                   // Blank old spot
(CL+i)->X = C;                                // Remember where I live
(E +C.y*SZ +C.x)->C = (CL+i);                 // Inhabit new spot
```

And how I implemented my avoidance routine during reproduction of plants:

```
for(int i=0; i<Ppop; i++)
   {
   maturity = (int) (PL+i)->EN & 7 + 3;       // EN::G4
   satiety = (float) (((PL+i)->EN & (7 << 10)) >> 10) + 1.0;   EN::G3  1 - 512
// if( ( (PL+i)->age > maturity ) && ((PL+i)->energy > satiety ) )
   if( ((PL+i)->age > 3) && ((PL+i)->energy > 47.5) )
      {                                       // && pollen > 0.25???
      A = (PL+i)->X;                          // Location of parent plant.
      int r = rand()%Ppop;                    // Select random mate

      if( ((PL+r)->age > 3) && ((PL+r)->energy > 47.5) )      // PFA number
         {
            // Need to find an empty spot in E
         int j = 1;                           // Skip center spot
         int ring = 5;                        // Pollination inner boundary

         do   {
              B = scale( ring, dir[j] );      // Motion vector
              C = bound( sum( A, B ) );       // Prospective new location

              j++;                            // Walk around direction vector
              if( j > 8 )                     // End of dir[]
                 {
                 j = 1;                       // Search next
```

```
                  ring++;                       // outer ring
                  }
               if( ring > 20 )  break;          // The Outer Limit!!
               }    while( (E +C.y*SZ +C.x)->P );   // Occupied

                    // New plant needs to be filled
            (NP+nw)->X = C;                     // Put plant at new location
            (NP+nw)->energy = (PL+i)->energy * 0.125;    // Natal energy
            (NP+nw)->energy += (PL+r)->energy * 0.125;   // 1/8 from each parent

            (PL+i)->energy -= (PL+i)->energy * 0.125;    // Parent loses 1/8 of its energy
            (PL+r)->energy -= (PL+r)->energy * 0.125;    // Other parent loses energy

            (NP+nw)->EN = crossover( (PL+i)->EN, (PL+r)->EN );   // Crossover both plant's
            (NP+nw)->PL = crossover( (PL+i)->PL, (PL+r)->PL );   // Chromosomes
            (NP+nw)->GM = (PL+i)->GM;           // Pass Mom's genetic marker on
/*          if( (NP+nw)->GM == 3855 )
               if( fRan() > 0.5 )               // Mutate 50% of marked plants
                  {
                  (NP+nw)->EN = mutation( (NP+nw)->PL );
                  (NP+nw)->PL = mutation( (NP+nw)->PL );
                  }
*/                                              // My name is 905
                  (NP+nw)->alive = true;        // and I've just become alive
                  (NP+nw)->age = 1;             // I'm the newest populator
                                                // of the planet called Earth.
                  (E +C.y*SZ +C.x)->P = (NP+nw);  // Point to plant data
                  nw++;                         // Add new plant to count
                  }
               }
            }
```

An energy flow pattern. Nutrients feed plants. They feed herbivores and return energy to the nutrient layer. Herbivores feed carnivores and return energy to the nutrient layer. Carnivores return energy to the nutrient layer via senescence. Energy flows through metabolism, eating, reproduction, and death. I need to account for each path. I think I have metabolism, eating, and death worked out. Now to do the same analysis of energy flow during reproduction.

VII Assignments

- Run ecoNode with the compile directive #define FIRST active.
- Rename one of the ecoXX.dat files to eco.dat
- Comment out #define FIRST and run ecoNode again to use the eco.dat snapshot.
-

VII Links

https://www.geeksforgeeks.org/genetic-algorithms/
https://learn.microsoft.com/en-us/windows/win32/api/wingdi/nf-wingdi-createfonta
https://learn.microsoft.com/en-us/windows/win32/controls/scroll-a-bitmap-in-scroll-bars
https://learn.microsoft.com/en-us/windows/win32/gdi/binary-raster-operations
https://learn.microsoft.com/en-us/windows/win32/gdi/ternary-raster-operations
https://learn.microsoft.com/en-us/windows/win32/api/wingdi/nf-wingdi-patblt
https://learn.microsoft.com/en-us/windows/win32/api/wingdi/nf-wingdi-stretchblt

VIII Appendices

Miscellaneous Notes

gnuplot

set logscale y gives me a semi-log graph to show millions and hundreds on the same chart.

set logscale gives me a log,log chart to show change over time

plot 'dt.dat' using 1:4 title 'Carnivore' with line, 'dt.dat' using 1:3 title 'Herbivore' with line, 'dt.dat' using 1:2 title 'Plant' with line

This script is not as complicated as it appears. The commas delimit the three organism scripts.

plot tells gnuplot this is a 2D graph. 'dt.dat' tells gnuplot where the data lives. using 1:4 is a bit more involved. It is telling gnuplot to use the values in the first column for the X coordinate and the values in column four as the Y coordinate. The first column is the cycle index and the fourth column is the carnivore population. title 'Carnivore' with line tells gnuplot to use Carnivore in the legend and use a line instead of a dot for display. This scheme is repeated for herbivores and plants. The topmost line is magenta while the second is green and the third is blue.

Organism energy modification.

How do I test each module for accuracy?

When I modified die() I sent the energy left in senescent organisms to the nutrient layer. This caused the system to gain too much energy and crash around 1500 cycles or so. My modification of reproduce() don't seem to have such a strong effect. There I returned energy to the nutrient layer which had been ignored. I thought about it but found I had not written that into the code!! Ack! Only writing this document got me to see this error. I should have been writing about my code for a long time. It really helps me think about what I am writing :)

I now have senescence and reproduction and replenish tweaked. Energy from senescent organisms returns to nutrient layer. Energy from reproduction returns to nutrient layer. And replenish does NOT add an offset amount of energy to each cell. Currently I have two simulations running which are telling me the system will collapse before 1500 generations. I may need to add the offset back to replenish() again. Hmm... this is more sensitive than I had known.

I have been doing more tweaking. Currently I am adding 0.3 units of energy beyond what is expended by metabolism(). The last simulation crashed around 6500 cycles. I have reset senescence to return energy to the nutrient layer. Reproduction is tweaked too. Plants expend 18.75% of each parent's energy. 1/8 from each parent to their offspring, 1/16 from each parent to the environment. Herbivores and carnivores expend a little more. But only when the organism is the mother. She gives 1/8 to the offspring, as does the father, and 1/16 to the environment, as does the father. However, she expels afterbirth at the offspring's location. That costs her an extra 1/16 of her energy. Thus the male expends 18.75% of its energy during reproduction. The female expends 6.25 % more for a total of 25%.

The extra drain on the total energy has me chasing parameters to achieve stable, long term choices.

User tweakable parameters in ecoDraw.c

Tweak data storage routines. Since useSamples() employs 'sample.dat'

and 'sample9.dat' ecoDraw should write to a separate 'sampleXX.dat'

Then you can maintain 'sample.dat' as a fixed starting point.

'eco.dat' is overwritten every 500 cycles. I use these snapshots

to start a new simulation. Test that mode again.

But make sure to limit the amount of disk space it requires.

It was OK to keep a series of them during ecoNode's long

run of over 250,000 cycles. Now I have some historic data.

I need to do this with 'sample.dat' too.

A menu structure for data logging would be a good start.

Then create a catalog of choices for either start method:

Need to test all three of them. You are only using one right now.

A) Start from scratch with initP, initH, initC routines.

B) Start from a previously stored snapshot 'eco.dat'.

C) Start from samples saved in 'sample.dat' and 'sample9.dat'

You need a few historic copies of: 'eco.dat', 'sample.dat', and 'sample9.dat'

Samples should be save to a NEW name, not 'sample.dat' which destroys
your legacy file.

storeSamples() now writes to 'sampleN.dat' instead of overwriting
'sample.dat'.

Archive in 'eco.dat', 'sample.dat', and 'sample9.dat'
Create into 'ecoN.dat' and 'sampleN.dat'.

Toggle storage of both snapshots and samples.
Turn them on during interesting cycles.
However, they do provide you with stored genetic history
to apply in later simulations.

I don't think I should modify the logging of 'dt.dat'.
That should be a standard feature to archive the simulation history.
We probably should kill printing to a terminal window.
Normal people find it disconcerting :)

There is a 'glitch' of sorts. WM_CREATE initializes the system.
However, CreateFontA() does not make the same font for all
of my computers. Leto, Dionysus, and Hephaestus like
20 a point font while Loki likes a 14 point font.
They are all running the same version of the same OS
their font libraries should be exactly the same.
What is going on here? Argh!!!

There is also a glitch on Hephaestus.
The top few lines on the display get filled with

garbage. If I scroll away and then back again they are gone.

This does not occur on any of the three other computers.

Once again: ARGH!!!

Fixed VirtualBox on both Dionysus and Hephaestus. I reinstalled both of them with a fresh VirtualBox VMs folder. Snapshots and images were not being stored correctly. I think it was a permissions issue. Recreating the system worked just fine.

Examine memory use. Do all of the buffers need to be so large?

Reproduction energy flows:

```
       (NP+nw)->energy =  (PL+i)->energy * 0.125;
       (NP+nw)->energy += (PL+r)->energy * 0.125;
       (E +(PL+i)->X.y*SZ + (PL+i)->X.x)->nutrient += (PL+i)->energy * 0.0625;
       (E +(PL+r)->X.y*SZ + (PL+r)->X.x)->nutrient += (PL+r)->energy * 0.0625;
       (PL+i)->energy -= (PL+i)->energy * 0.1875;
       (PL+r)->energy -= (PL+r)->energy * 0.1875;

       (NH+nw)->energy =  (HL+i)->energy * 0.125;
       (NH+nw)->energy += (HL+r)->energy * 0.125;
       (E +(PL+i)->X.y*SZ + (HL+i)->X.x)->nutrient += (HL+i)->energy * 0.0625;
       (E +(PL+r)->X.y*SZ + (HL+r)->X.x)->nutrient += (HL+r)->energy * 0.0625;
       (E +C.y*SZ +C.x)->nutrient += (HL+i)->energy * 0.0625;
       (HL+i)->energy -= (HL+i)->energy * 0.250;
       (HL+r)->energy -= (HL+r)->energy * 0.1875;

       (NC+nw)->energy =  (CL+i)->energy * 0.125;
       (NC+nw)->energy += (CL+r)->energy * 0.125;
       (E +(CL+i)->X.y*SZ + (CL+i)->X.x)->nutrient += (CL+i)->energy * 0.0625;
       (E +(CL+r)->X.y*SZ + (CL+r)->X.x)->nutrient += (CL+r)->energy * 0.0625;
       (E +C.y*SZ +C.x)->nutrient += (CL+i)->energy * 0.0625;
       (CL+i)->energy -= (CL+i)->energy * 0.250;
       (CL+r)->energy -= (CL+r)->energy * 0.1875;
```

useSamples()

has its own copy of:

Ppop = 500000; Hpop = 230000; Cpop = 50000;

It also uses initEcosystem()

We could save a mature nutrient layer saved from a long run.

Work with genetic markers.

Add them to injected populations.

Find them during run for special display.

Count them for comparison studies.

Display more information about them. ???

Tweakable parameters: 36 right now

```
int Ppop = 3900000, Hpop = 900000, Cpop = 80000; in main()
    Ppop = 500000;  Hpop = 230000; Cpop = 50000; in useSamples()

initEcosystem()
       E->nutrient = 250
initPlants()
       PL->energy = 80
initHerbivores()
       HL->energy = 120
initCarnivores()
       CL->energy = 300
-------------------------------------------------------------------------

metabolize()
       PL->energy = EN::G5 + 7.5
       HL->energy = EN::G5 + 4.2  ||
       CL->energy = EN::G5 + 5.3  \/

replenish()
       E(i,j)->nutrient += energy / (SZ*SZ) + 0.7

eat()
       PL satiety = PL->EN::G3 + 19.1
       PL absorb = (PL->EN::G4 + 1.0) / 15.0
       PL spine = PL->PL::G2 / 94.0
       PL strength = PL->PL::G3 / 94.0

       HL satiety = HL->EN::G3 + 56.1
       HL absorb = (HL->EN::G4 + 1.0) / 10.0
       HL speed = HL->HB::G1 + 12
       HL withstand = HL->HB::G3 / 31.0
       HL eat = HL->HB::G4 / 31.0
       HL camo = HL->HB::G5 / 50.0
              ch > 29  plant search count limit

       CL satiety = CL->EN::G3 + 8.0
       CL absorb = (CL->EN::G4 + 1.0) / 8.0
       CL speed = CL->CN::G1 + 12
       CL see = CL->SN::G5 / 255.0
              ch > 57  herbivore search count limit
This a search limit and an eating limit.  It eats what it finds 57 times.

reproduce()
       PL->age > 2         hard wired
        HL maturity = HL->EN::G4 + 1
       CL maturity = CL->EN::G4 + 3
       PL->energy > 12.5   hard wired
       HL->energy > 65.0   hard wired
       CL->energy > 35.5   hard wired

die()
       PL senescence = EN::G1 + 3
       HL senescence = EN::G1 + 19
       CL senescence = EN::G1 + 39
```