Genetic art installation documentation:

Programs:

drawall.pl:	Sequentially call *cgdraw* on all files in a directory
exptocg.pl:	Compile all of the expressions in a .exp file to separate .cg files
crossover:	Do crossover on the expressions in a .exp file
evaluate.pl:	Take the output of *cgtile* and change the fitness in a .exp file
mutate.pl:	Do mutation on the expressions in a .exp file
populate.pl:	Create expressions for a population
select.pl:	Perform selection based on the fitness of expressions in a .exp file
cgdraw:	Use Cg to draw an expression animated with time
cgeval:	Use Cg to create a snapshot of all cg files in a directory
apply.pl:	Compare the snapshots from cgeval to another image to create fitness
cgtile:	Display interactive demo designed for touch screen using Cg

Files:

gatemp.cg:	Template file used by *exptocg.pl* to create Cg programs
cg.exp:	Expression file
cg.grm:	Grammar file, referenced by *cg.exp*
genetic.pm:	Module required by most of the perl programs

drawall.pl

`drawall.pl [`*`count`*`]`

Simply calls *cgdraw* in preview mode on all of the .cg files in a directory. Displays *count* at a time before pausing. *count* defaults to 5 if not specified.

exptocg.pl

`exptocg.pl` *`filebase desc`* `[`*`expfile`*`]`

Compile all of the expressions in *expfile* (or STDIN if not specified), to separate Cg programs. Expects to find *gatemp.cg* in the same directory as *exptocg.pl* as a template for the Cg programs. Filenames for Cg programs are constucted by taking *filebase*, appending the number of the expression, starting with 1, and appending ".cg". The first program is named "*filebase*.cg". *desc* must be a data description file describing the data that will be fed to *cgtile* to execute the Cg programs.

crossover

`crossover` *`expfile`*

Perform crossovers on the expressions in *expfile*.

evaluate.pl

`evaluate.pl` *`sysexp userexp`* `[`*`runfile`*`]`

Change the fitness of the expressions in *sysexp* and *userexp* according to the votes from *runfile*. If *runfile* is not specified, use STDIN.

Requires *genetic.pm* in the current working directory.

mutate.pl

`mutate.pl` *`expfile`*

Perform mutation on every expression in the file *expfile*. Amount of mutation is set by variables at the top of the expression file.

Requires *genetic.pm* in the current working directory.

populate.pl

`populate.pl [rc]` *`expfile`*

Populate the expression file *expfile* with new random expressions according to the parameters at the top of the expression file. In normal mode, all expressions in the file are removed and an entire new population is created. If "rc" is specified on the command line, expressions from the file are removed randomly according to the random creation parameter at the top of the expression file, and the population is regrown to full size with new random expressions.

Requires *genetic.pm* in the current working directory.

select.pl

`select.pl` *`expfile`* `[query]`

Remove all but the best expressions in *expfile* as specified in the parameters at the top of the expression file, and then replicate the best expressions to refill the population. If "query" is specified on the command line, prints out statistics about the population and fitness without modifying the expression file.

Requires *genetic.pm* in the current working directory.

cgdraw

`cgdraw` *`cgprogram datafile descfile`* `[preview]`

Draws an animated Cg program feeding it data from a file. The program to draw is specified by *cgprogram*. The data must be described by the file *descfile* and must be contained in the file *datafile*. If "preview" is specified on the command line, the programs are rendered at 320x240. Otherwise they are rendered at 1280x1024.

Requires libSDL, libCg, and OpenGL.

cgeval

`cgeval`

Renders one frame of every Cg program in the current working directory and saves it in ppm format under the same filename as the Cg program with the extension changed from .cg to .ppm.

Requires libSDL, libCg, and OpenGL.

apply.pl

`apply.pl` *`expfile testppm`*

Calculates the fitness of each expression in *expfile* for which there is a ppm file in the current working directory. Fitness is calculated by comparing against *testppm* using *pnmpsnr* which calculates the peak signal to noise ratio.

Requires *genetic.pm* in the current working directory, and netpbm.

Cgtile

`cgtile` *`tilesize votes sysdir userdir datafile descfile`*

Runs the genetic art interactive touch screen demo. *tilesize*^{2} random Cg programs from either *sysdir* or *userdir* will be shown and the user will be given *votes* votes for them. The programs will be rendered using data from *datafile* as described by *descfile*.

Requires libSDL, libCg, libpthread, and OpenGL.

Example:

Here I present an example of using this genetic system with the interactive touch screen demo. We must begin by setting up an appropriate directory in which to run the system. For this example, I will call that directory *run*. Start with the following unmodified files found with the genetic programs:

run/

- *confirms.bmp*
- *genetic.pm*
- *icon.bmp*
- *introtext.bmp*
- *load.bmp*
- *sysadmin.bmp*
- *thankyou.bmp*
- *votes.bmp*

Now you'll need a grammar file. Copy the *cg.grm* file into this directory and change the weights in it if desired. You will need to reference this file in the expression files.

Now copy the expression file *cg.exp* into this directory twice. Once as *user.exp* and once as *sysadmin.exp*. Modify the settings in the files as desired, making sure that the grammar in the expressions points to the grammar file you're using (*cg.grm*).

Now you'll need data, and a description file for the data. Either copy or create a symbolic link to the data file at *data.txt* and copy or link the data description file as *desc.txt*.

Finally you'll need to create two subdirectories, *user/* and *sysadmin/*. Now the directory contents should be as follows:

run/

- *cg.grm*
- *confirms.bmp*
- *data.txt*
- *desc.txt*
- *genetic.pm*

icon.bmp
introtext.bmp
load.bmp
sysadmin/
sysadmin.bmp
sysadmin.exp
thankyou.bmp
user/
user.exp
votes.bmp

Now you're ready to begin running the system. The following steps constitute a normal run of the system assuming that all of the programs are in *../bin/* and that the current working directory is the *run/* directory above:

1) `../bin/populate.pl user.exp`
2) `../bin/populate.pl sysadmin.exp`
3) `../bin/exptocg.pl user/e desc.txt user.exp`
4) `../bin/exptocg.pl sysadmin/e desc.txt sysadmin.exp`
5) `../bin/cgtile 3 5 sysadmin user data.txt desc.txt | ../bin/evaluate.pl sysadmin.exp user.exp`
6) Run the demo until enough votes have been cast. Vote counts can be checked with the following:
   a) `../bin/select.pl user.exp query`
   b) `../bin/select.pl sysadmin.exp query`
7) `../bin/select.pl user.exp`
8) `../bin/select.pl sysadmin.exp`
9) `../bin/populate.pl rc user.exp`
10) `../bin/populate.pl rc sysadmin.exp`
11) `../bin/mutate.pl user.exp`
12) `../bin/mutate.pl sysadmin.exp`
13) `../bin/crossover user.exp`
14) `../bin/crossover sysadmin.exp`
15) Repeat from step 3