IEAS

# Interactive Evolutionary Art System for ECJ

## Framework extensions, implementation, user guide

Michael Gircys

Mg12vp – 5269550

COSC 5P71 – Genetic Programming

Prof. Brian Ross

2015-05-01

## Purpose

The ECJ evolutionary system was expanded, providing new interfaces and prototypes to more easily allow for user-assigned fitness of evolved images, or other items requiring artificial selection. These interfaces were used to provide a prototype interactive evolutionary art system. The extensions and implementation created should provide a suitable base for expansion and further research into evolved art.

## Framework Adjustments and Extensions

Problems which require user-specified fitness in their evaluation should implement *IEASProblemForm*, which extends the *SimpleProblemForm* interface. The extended problem-implementable methods from the interface are used to

**setup** - as before, used to initialize problem state.

**prepareToUserEvaluate** - called after the generation has been bred, but before evaluation has been performed on the individuals. This is called on the single main thread, and can be used for setting up a user-visible window with a wait message or other placeholder.

**prepareToEvaluate** - called for each thread performing evaluation, this performs initialization independent of the individual being evaluated.

**evaluate** - called to evaluate each individual, possibly with some concurrency. While used in other problem forms to assign fitness, the IEASEvaluator leverages the concurrency ability of these calls to quickly render the phenotype images.

**finishEvaluating** - called for each thread performing evaluation after all individuals associated with the thread have been evaluated. While not made use of in the current implementation, it may be leveraged to update a user window with additional individuals to evaluate as they are rendered.

**userEvaluate** - called when all evaluation threads have completed. All individuals are ready for user evaluation, and the user-facing window can be updated. If other threads are used to obtain the user-assigned fitness (such as Swing UI threads in the provided implementation), care should be taken to ensure that the userEvaluate method waits, and does not return until the user has completed their evaluation.

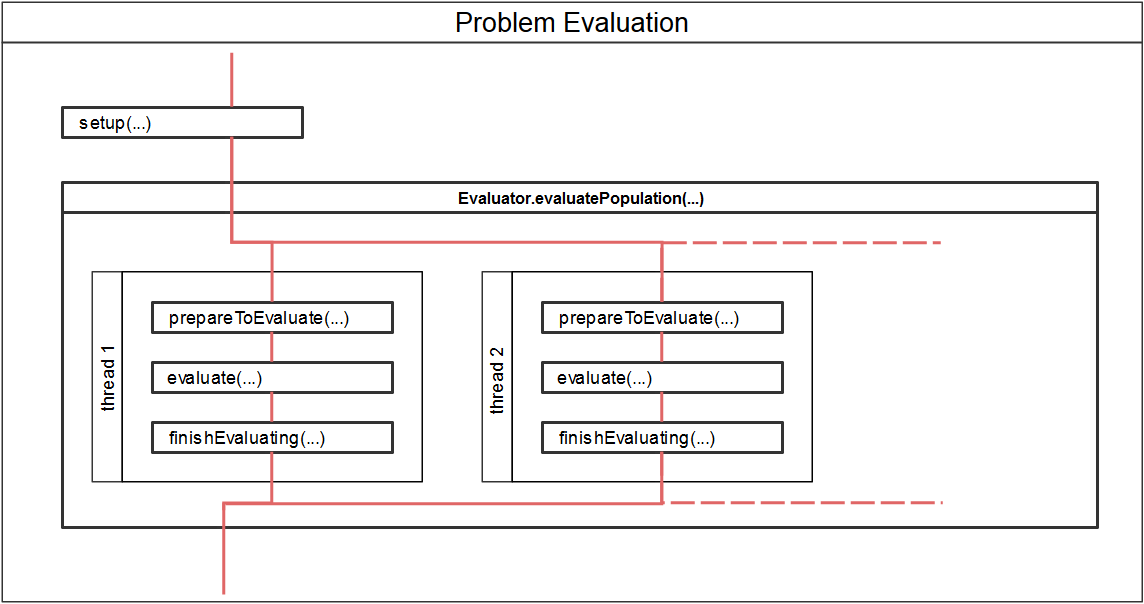


Figure 1 – SimpleEvaluator process flow using a SimlpeProblemForm interface

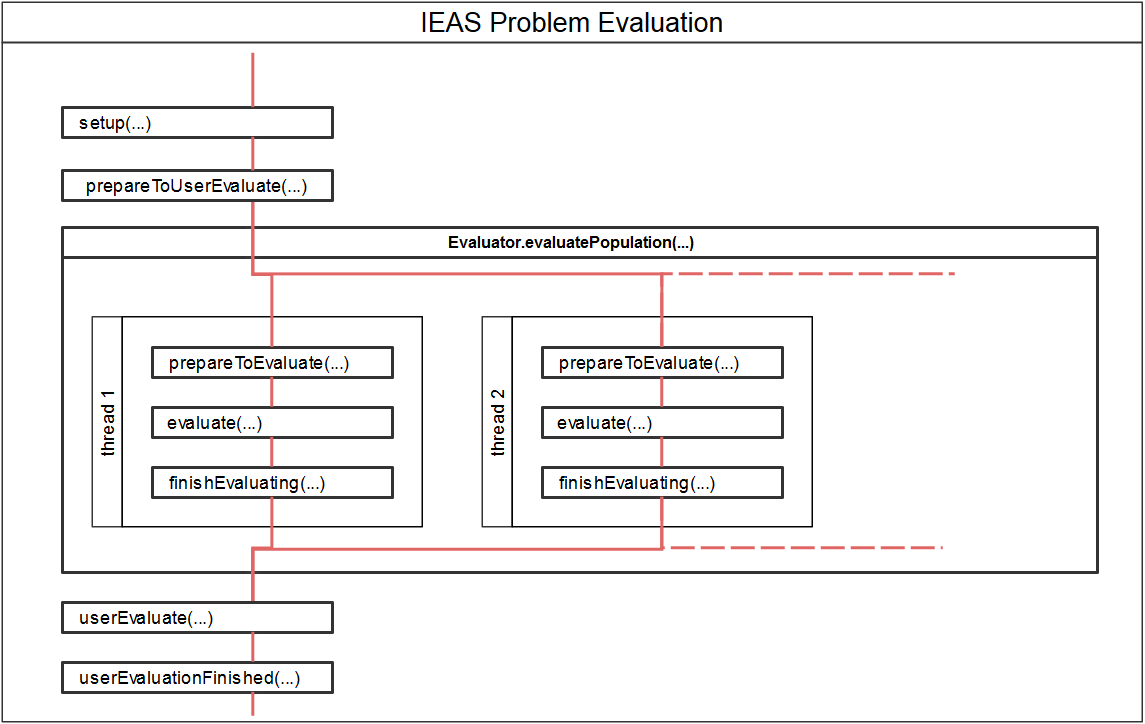


Figure 2 – IEASEvaluator process flow using the extended IEASProblemForm interface

**userEvaluationFinished** - called when *userEvaluate* completes. All individuals should have been assigned fitness, and scaling or other adjustments may be performed if needed (ie. adjustment for Roulette style selection).

**userExited** - replaces the check to see if any individuals have perfect fitness. As there are few discrete fitness values that can be assigned, this should check to see if a flag has been raised by the UI for explicit user exit.

As an IEAS is likely to have some sort of visual phenotype image rendered by the *evaluate* method, storage of these images are handled by using an **IEASIndividual** for individual representation. The IEASIndividual extends the GPIndividual class by including a BufferedImage member for quick association of the phenotype. The IEASEvaluator may operate independently, without usage of the IEASIndividual, though the provided IEAS problem and user-facing windows will not be as robust, and will require a custom implementation.

## IEAS Implementation

The IEAS system implemented uses genetic programming representation to evolve 3 trees for each individual. These trees each correspond to a channel in RGB colour space. For each generation, the individuals are evolved, rendered to a RGB image, and evaluated by the user, before breeding pipeline configurations are adjusted, and the next generation is produced.

For evaluation, the user is presented with a window displaying a 128x128 sample of the rendering of each individual. The user may click on the image to obtain details about the individual. Details include a representation of the genotype trees, and a larger sized version of the image, which can be filtered to individual channels, or a channel overview. Users may also save individual images at a desired resolution, as well as the trees, to a file.

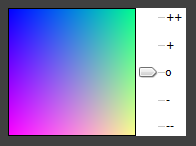


Figure 3 – Displayed individual with neutral evaluation

Once the image corresponding to the individual has been judged by the user, a slider next to the image may be adjusted to assign the individual a rank. The ranks are internally assigned as values from 1 to 5 before being normalized, but an alternate labeling has been used to better request whether the user wishes to see images more or less like the individual (as seen in figure 3). The default evaluation of an individual is neutral, represented by an O (value 3). Users can modify the rating to note that they greatly enjoy (++, rating 5) or dislike (--, rating 1) the image.

In addition to providing evaluations for individuals, the user may also adjust a number of simple breeder pipeline settings to be used in creating the next generation. The probabilities of all pipelines used for each subpopulation can be adjusted, as can the amount of elitism. A user wishing for more minute changes may assign reproduction and ephemeral mutation higher probabilities of being used to produce children, where a user wishing for more varied patterns may prefer higher amounts of crossover and (subtree) mutation.

## Implementation GP Language and Settings

A fixed number of generations must be provided to the system to ensure termination. As our implementation will require the user to explicitly notify the system to terminate, a fixed number of generations are not required, so a relatively high default of 216 has been selected.

While rendering could be done quickly for higher numbers (at our 128x128 resolution), a lower population size was required to aid with evaluation. One of the largest issues with interactive evolutionary systems is user fatigue (Li 1). A population size of 56 was chosen as all individuals (at 128x128) could be tiled in a 1920x1080 window and be visible at the same time, making it easier for a user to quickly identify positive and negative outliers.

Many of the settings used for the IEAS system borrow from default simple and Koza parameters included with ECJ. A notable exception is the reduction of the tournament selection size, which was far too strong at 10, especially in consideration of the reduced population size.

A default elitism value of 1 was hesitantly chosen. The few discrete ratings that an individual can be assigned raise issue with which top-scoring individual should be stored as elite. Further, the inclusion of reproduction as a breeder pipeline option, paired with a small population size could lead to very early convergence. However, should the user become attached to an image in an early generation, it may be disheartening to lose the favoured image on an off chance, or for evolutions of the image to be more poorly received. Both the probability of straight reproduction and amount of elitism are configurable from within the user-facing windows of the IEAS implementation.

In addition to the tree mutation, crossover, and reproduction pipelines, the inclusion of ephemeral constants permitted an additional mutation of the ephemeral constant values. For the provided IEAS implementation, ephemeral mutation would provide a fine amount of change, adjusting the value by of the constant’s max value, so long as the result would remain between 0 and the max value. Three ephemeral constant ranges were used, [0,1], [0,10], and [0,100] for various scales of constants.

Table 1 – GP Parameters

|  |  |
| --- | --- |
| *Parameter* | *Value* |
| Generations | ∞ (65535) |
| Population Size | 56 |
| *Generation 0* |  |
| Builder | Ramped Half & Half |
| New Node Depth | [2,6] |
| Grow Probability | 50% |
| Parent Selection |  |
| Elitism | 1 |
| Selection | Tournament |
| Selection Size | 3 |
| Node Selection |  |
| Terminals | 10% |
| Non-Terminals | 90% |
| Breeder Pipeline |  |
| Reproduction | 0% |
| Crossover | 80% |
| Mutation | 10% |
| Ephemeral Mutation | 10% |
| Crossover Settings |  |
| Max Depth | 17 |
| Attempts | 1 |
| Mutation Settings |  |
| Max Depth | 17 |
| Attempts | 1 |
| Builder | Grow |
| New Node Depth | [5,5] |
| Ephemeral Mutation Settings |  |
| Change Factor | [0%, 1%] Max |

While determining which functions to provide for the GP language, a subset of the language employed by Li et al. in their work on evolutionary systems learning aesthetic judgements was used. The language consists of a number of arithmetic and trigonometric functions, various ranges of ephemeral constants, variables for the current image position, and a conditional operator. Noise functions were not implemented in the current implementation, where a focus on primitive operators was preferred for initial testing.

While the functions used within the GP language were restricted to returning double floating-point precision scalar values, a number of other works made use of a hybrid of scalar and vector functions, which could provide an area for additional research.

Table 2 – GP Language

|  |  |  |
| --- | --- | --- |
| Category | Sign | Description |
| Variables | X | Current horizontal position (in a [0,1] window) |
|  | Y | Current vertical position (in a [0,1] window) |
| Ephemeral Constants | E[#] | Random constant in [0,1] |
|  | E[#] | Random constant in [0,10] |
|  | E[#] | Random constant in [0,100] |
| Math, Unary | sin | Trigonometric cosine (expecting radian angles) |
|  | cos | Trigonometric sine (expecting radian angles) |
|  | exp | e (Euler’s number) raised to the parameter value |
|  | - | Sign change |
| Math, Binary | + | Arithmetic addition |
|  | - | Arithmetic subtraction |
|  | \* | Arithmetic multiplication |
|  | / | Safe division ( divisor of 0 🡺 0) |
|  | max | Greater of the two parameters |
|  | min | Lesser of the two parameters |
| Conditionals | IfGT | If the first parameter if greater than the second, returns the third parameter, otherwise returns the fourth |

## Further Research Topics

The IEAS implemented was intended to provide a base from which additional research into evolutionary art could be explored. While designing and creating the system, a number of further research topics were identified.

The system implemented makes exclusive use of RGB colour space, though work has been done making use of HSV and other colour systems. One idea noted by Li et al. would be to allow an additional mutation operator which toggles the colour space that the individual employs.

*Learning Aesthetic Judgements in Evolutionary Art Systems,* and *Gentropy: Evolving 2D Textures* present many ideas on using metrics to assist with, or provide unguided evolution of textures, and could be integrated well with the current system.

A number of papers have noted GP languages that include a number of vector functions. Experimentation and determination of notable differences presented by vector and/or different noise functions could be of interest.

Providing options for granularity of tree and ephemeral mutation might provide for better control of fineness/coarseness of structure mutation.

The domain on which the implementation renders images is confined to [0,1], though a more varied range may allow for different functions to better express their inherent behaviour. (Evolved) adjustment to the range that variables X and Y could take could permit for more interesting behaviour, and/or permit images rendered with variable aspect ratios

The details window for an individual presents an intuitive tree representation of the individual. An editable tree view which could render individuals tweaked by users could be neat.

## References

Camick, Rob. “Wrap Layout”. *Java Tips Weblog.* 2008, Accessed 2015-04-25. <https://tips4java.wordpress.com/2008/11/06/wrap-layout/>

Li, Yang et al. “Learning Aesthetic Judgements in Evolutionary Art Systems”. *Genetic Programming and Evolvable Machines* 14.3 (2013): 315-337.

Romero, Juan and Penousal Machado, eds. “The Art of Artificial Evolution – A Handbook on Evolutionary Art and Music”. Berlin: Springer, 2008. Print.

Ross B.J., and Wiens A. “Gentropy: Evolutionary 2D Texture Generation”. *Computers and Graphics Journal* 26.1 (2002): 75-88.