Perceptron

(v. 1.0 February 2011)

From the old to the modern video feedback art

Contents

Introduction

Perceptron is a video feedback engine with a variety of extraordinary effects.

Home page: http://perceptron.sourceforge.net
Download: http://sourceforge.net/projects/perceptron
Gallery: http://perceptron.sourceforge.net/gallery/index.html

The classical *video feedback* is an artistic technique that uses a video camera and a TV screen. The camera is pointing at the screen that is simultaneously broadcasting the output from the camera. By using the camera alone, we can see a variety of fractal spirals, and a number of pixel-related or saturation-related effects that stem from the electronic components. Perceptron does not simulate the latter two, although they do occur in Perceptron-specific manner. With a single mirror located next to the TV screen, we can additionally create fractal trees, and by adding even more mirrors, we can create complex IFS fractals. [1]

Perceptron simulates the essence of a classical video-camera setup, although it does not try to match its output exactly. Instead, it uses a two-dimensional computer screen only, in order to morph its contents recursively. In the process, it applies any given complex function f(z, c) that produces a Julia fractal, a pullback effect (with rotation) that zooms in and out, a reflection transform that multiplies the resulting Julia fractal, and a set of coloring techniques responsible for the Julia contours, semi-transparency and visibility of different structures. The coloring techniques stem from the boundary condition check performed on the point z' = f(z, c). They are the outside coloring methods, the gradient functions, the fade-to color modes, and the color filters. The reflection transform returns the points mapped outside of the screen dimensions back within the screen. This yields complex IFS fractals comprised of Julia fractal fragments.

Perceptron is a free, open source program written in Java (www.java.com), and licensed under the GNU Public License GPL version 3. Primary author: Michael Everett Rule (mrule7404@gmail.com). Documentation and further development (v0.9 – v1.0): Predrag Bokšić (junkerade@gmail.com)

References:
[1] http://classes.yale.edu/fractals/Labs/VideofeedbackLab/VideofeedbackLab.html

Find out more online…
http://en.wikipedia.org/wiki/Video_feedback
http://en.wikipedia.org/wiki/Conformal_pictures
http://en.wikipedia.org/wiki/Fractal

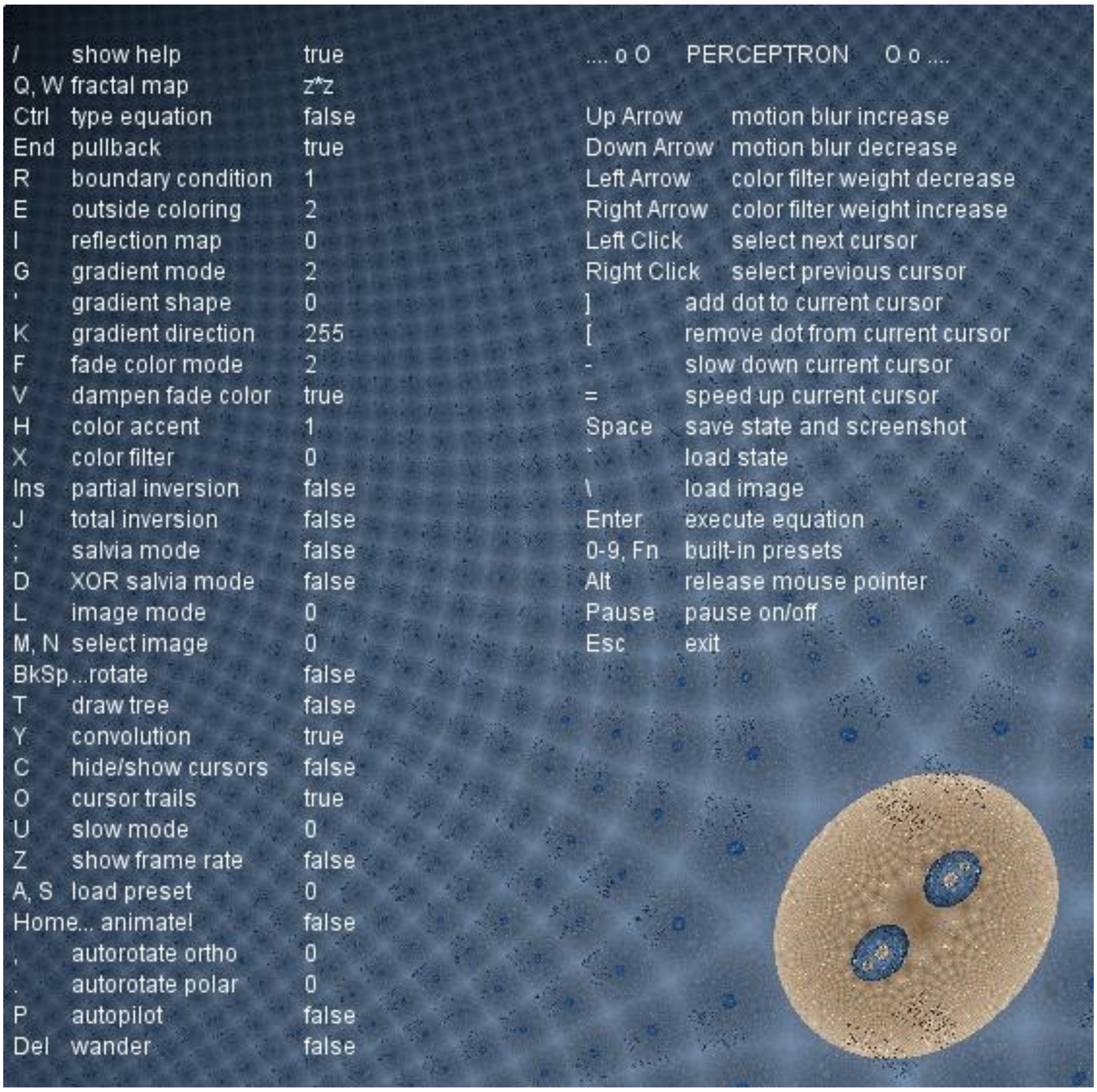
How to run

Download and unpack Perceptron zip archive to any folder, such as My Documents. Install Java from www.java.com. Enter the unpacked folder Perceptron and double-click **perceptron.jar**. It runs on all platforms. If you wish to alter the source code and compile the project, then download and install Net Beans from http://java.sun.com/javase/downloads/widget/jdk_netbeans.jsp.

Usage

Keyboard Controls

To start, press down arrow and left arrow a few times. The arrows control the screen update delay and the fog effects. Read the **on-screen help** by pressing the key **/** (slash).



/	show help	true
Q, W	fractal map	z*z
Ctrl	type equation	false
End	pullback	true
R	boundary condition	1
E	outside coloring	2
I	reflection map	0
G	gradient mode	2
'	gradient shape	0
K	gradient direction	255
F	fade color mode	2
V	dampen fade color	true
H	color accent	1
X	color filter	0
Ins	partial inversion	false
J	total inversion	false
;	salvia mode	false
D	XOR salvia mode	false
L	image mode	0
M, N	select image	0
BkSp	...rotate	false
T	draw tree	false
Y	convolution	true
C	hide/show cursors	false
O	cursor trails	true
U	slow mode	0
Z	show frame rate	false
A, S	load preset	0
Home	... animate!	false
,	autorotate ortho	0
.	autorotate polar	0
P	autopilot	false
Del	wander	false

.... o O PERCEPTRON O o ....

Up Arrow	motion blur increase
Down Arrow	motion blur decrease
Left Arrow	color filter weight decrease
Right Arrow	color filter weight increase
Left Click	select next cursor
Right Click	select previous cursor
]	add dot to current cursor
[	remove dot from current cursor
-	slow down current cursor
=	speed up current cursor
Space	save state and screenshot
`	load state
\	load image
Enter	execute equation
0-9, Fn	built-in presets
Alt	release mouse pointer
Pause	pause on/off
Esc	exit

Mouse Controls

The mouse controls any one of the four basic cursors. At startup (as well as upon loading a preset), neither cursor is selected. Click left or right mouse button to select a cursor.

Red cursor: sets the parameter c of the complex function (e.g. $f(z) = z^2 + c$), (mimics the camera position).
Blue cursor: controls the pullback, (mimics the camera distance and rotation).
Yellow cursor: sets the color gradient parameters, offset and slope.
Dragonfly (black cursor): controls the amount and type of color filtering (contrast) when combined with the X switch.

The red cursor controls the **parameter c** with its position, c = (x, y). The center of the screen represents the coordinate zero, while the edges of a typical square screen are at about 1.1 units away.

The blue cursor controls the rotation and the **pullback**. If you rotate it around the coordinate zero, the whole fractal rotates, but at the same time, the parameter c rotates as well. If you move the blue cursor closer to the coordinate zero, you mimic the situation in which you move the camera away from the screen. This is analogous to the repeating images of screens within screens diminishing in size and contracting towards the screen's middle. If you move the blue cursor away from the coordinate zero, you zoom in practically, i.e. you simulate the camera approaching the screen. Press End to disable or enable the pullback.

It is best to see how the yellow cursor and the dragonfly cursor behave in practice. They control the application of the radial gradient, a matrix of shadow intensities that is used for the basic, **gray-scale coloring of fractals**.

Cursors are typically decorated with small circles (**dots**) that follow their motion. The **cursor motion** is slowed, because it smoothens the morphing of Julia fractal. It appears as though the cursor is dragging a weight on an elastic rope. Both features are optional and are adjustable by pressing =, - and], [.

The activation of the **rotating three-dimensional fractal Tree** will bring up an additional, temporary set of cursors that control the tree (its position and design). You can select any cursor by clicking left or right mouse button. The Tree is an IFS fractal for artistic purposes, which you can bring about when you press T. When you deactivate it, the tree-related cursors disappear. The **visibility of all active cursors** can be switched off and on by pressing C.

Operation

The screen resolution can be changed in the **settings file**, perceptron\resource\Settings.txt. (Double click the file to edit it manually.) Current default resolution is 640 × 640 pixels. This allows my computer (featuring the Intel E5200 processor and the ATI Radeon 2600 HD graphics card) to achieve 16 frames per second by using the default settings specified in the default preset, perceptron\resource\presets\1.state. If you achieve the frame rates far beyond the abilities of your monitor, you may wish to choose a higher resolution or slow the program down by pressing U. (Pressing U cycles through 4 degrees of speed moderation.)

Do not change the contents of the resource folder unless you know what you are doing. You need the settings file, at least one preset, one image file (usually the one denoted in the default preset), CrashLog.txt, cursors folder with the cursor images (icons), and possibly other files, especially if you are working with an experimental version.

The settings file contains parameters such as the **presets folder** and the **image folder,** which denote the locations from which Perceptron preloads all presets and images at startup. The **preloaded presets** are accessible by pressing S or A. The **images** are accessible in the **image modes** (L = 1 or 2) and in the **fractal mode** (L = 0) combined with certain outside coloring methods (such as E = 3 or 4). A **preloaded image** can be selected by pressing M or N. The loaded image morphs on the screen (together with the mouse cursors) according to the selected program options, namely the complex function (also known as the "fractal map") and the reflection transform. You can **load a single preset** later during operation by pressing `, or you can **load a single image** by pressing \. All presets and images loaded in the course of operation are held in program memory. They are added to the preloaded presets and images, and are available by pressing S, A or M, N respectively.

In addition, you can add your own **complex functions** (equations) to the list in the settings file. They are available by pressing W or Q. You can **type them in** during runtime by pressing CTRL once, typing the equation and then pressing Enter. Press CTRL again to stop editing the equation. This way you will exit the **equation edit mode**, deactivate the text cursor and return the controls to keyboard (return the default key assignments.) Hint: you can play with the colorful letters by typing text anywhere on the screen without pressing Enter (as to avoid entering a new equation made of silly text). The colorful letters represent the **Salvia mode** (press ; to disable or enable, and D to change the color scheme). The successfully typed-in and executed equations are added to the list of equations in memory (available by pressing W or Q), but are not saved to the hard disk otherwise, except for the current equation that will be saved when you save a preset.

The equations in the settings file, those that you type in, as well as the ones stored in presets are in the form of f(z). For example, z*z gives the default Julia fractal. This is a well-known short form of $z_{new} = z_{old}^2 + c$.

The save option appears when you press Space. It will save the state (preset) and the screenshots (two consecutive frames) in the My Documents folder (or anywhere else as you select). Sometimes it is knowledgeable to study the evolution of the complex fractal displays by comparing the difference between the two consecutive frames. In some situations, the difference is not visible to the naked eye. The only noticeable difference may be the slight image softening (Gaussian blur) that the final image acquires. This image is named name.out.png. I recommend Picasa viewer for all your images, http://picasa.google.com.

The CapsLock is temporarily assigned as the save button during the equation editing mode (if for some reason, you wish to save a preset at that exact moment). Note that if you load a preset with the equation-editing mode enabled, the default key assignments will be temporarily unavailable until you press CTRL and exit the edit mode.

The saved presets are available by pressing ` (the button called BackQuote), which calls the Open File menu, or by placing the saved presets in the resource\presets folder before the program starts. Each preset takes note of the image file that was used at the time of operation, so make sure that the specified image file exists on your hard disk at the time of opening the preset. (Alternatively, a preloaded image from the resource\images folder is used.)

The animation mode starts when you press Home. Firstly, a menu will appear that requires you to select an empty folder. After that, Perceptron will save the finalized consecutive frames, denoted by numbers. It will erase any files with the same name in that folder without asking. The animation mode is slower than the normal mode of operation. Remember to perform your planned actions slowly.

Warning: the **animation recording** creates a very large number of PNG image files that you need to bind together into a motion picture by using Virtual Dub http://www.virtualdub.org. In **Virtual Dub**, click on File → Open, and then from the File Type drop down menu, select "Image Sequence". Select the first image (with the lowest number) in the folder where you saved the animation sequence and Virtual Dub will then automatically import all other pictures that follow in numerical order. On the main menu, click Video → Frame Rate, and choose a frame rate for your movie. The higher the frame rate, the more images are shown per second (25FPS will show 25 pictures per second). Effectively, you can produce a higher frame rate and a smoother (denser) video than the Perceptron managed to calculate and save, but the process requires time. Change video compression if needed. The video compression will be required for storing the final video, due to large file sizes. Finally, click File → Save as AVI. I recommend VideoLAN player (in case you need any), http://www.videolan.org.

If you create a very large video file and you need to compress it later, but your system does not contain any video encoders, try DivX http://www.divx.com. The encoder included in the package will allow a single kind of compression free of charge, available through a drag-and-drop method.

Algorithm

Perceptron upholds the infinite, circular flow of pixel colors or in other words, visual images. The visual images evolve quickly, slowly or endlessly depending from the geometric transformations that are occurring between the two stages along the flow, the screen and the buffer memory.

Perceptron uses 4 matrices of equivalent format: screen, buffer (temporary screen contents holder), lookup table and gradient matrix. All 4 matrices are used in coordination, that is, we focus on a single element from each of the 4 matrices at a given time in order to update a single pixel of the buffer. For speed purposes, all matrices are stored in the form of one-dimensional arrays.

The flow of visual images starts with the screen matrix and follows to the buffer matrix. The screen represents the portion of the complex plane around the coordinate zero, no bigger than the circle of radius $r = 3$. The first or initial visual image (frame 0) needs to be copied ("mapped") from the screen to the buffer. The buffer will fill its element (pixel) at the position $z = (x, y)$ by reading the position $z' = f(z)$ at the screen. The number z' is precalculated and stored in the lookup table at the position z.

Next, we test to see whether the point z' is within the limit circle of radius $r = 1$ or beyond (depending from the boundary condition). If it is within, the color is simply copied from the screen to the buffer. If it is beyond, some color is added to the existing color found at the screen location z'. The added color depends from various coloring functions. Primarily, it is a value (shadow intensity) taken from the gradient matrix at the position z. The entire flow steps into the color inversion step next. When all pixels of the buffer are filled, the buffer is copied to the screen directly. Over several cycles, this creates a gray-scale Julia fractal with black and white contours.

Let us step back to the situation right after the boundary condition test. If the point z' is beyond the screen matrix size, the reflection transform moves it inside by the application of somewhat nonsensical linear functions. The color is obtained from the screen after all and passed on to the buffer. This creates multiple reflections, simple copies of Julia fractal portions (fragments) that form IFS fractals. The linear functions are equivalent to the affine transformations used to create IFS fractals in the ordinary fractal generators. The fact that their values are repeating within the screen limits makes their effect similar to the effect of branch cuts of some complex functions when they are used in the conventional fractal generators (such as Fractint and UltraFractal).

A single cycle is complete when the buffer sends its contents to the screen as-is. This initiates a new cycle of geometric transformation from the screen to the buffer.

Over the course of many cycles, the exquisite combination of Julia and IFS fractals emerges.

Program options

Boundary Condition (R) – the boundary condition for points z'.

0	Rectangular Window (as the “limit circle”)
1	Limit Circle (for classical Julia sets)
2	Elastic Limit Circle (the limit circle size is tied with the blue cursor)
3	Horizontal Window
4	Vertical Window
5	Inverse Oval Window (the inverse check on oval window)
6	No Window (move on to outside coloring)
7	Framed Window (with a colored frame)
8	Convergent bailout condition (for Newton fractals)

Outside Coloring Method (E) – what to do with the z' points.

0	Fill With Fade-to Color
1	Edge Extend (pass on the color for points within a large screen)
2	Just Pass on the Color
3	Paint With Image (image appears on fractal contours)
4	Paint With Image II (image with soft edges)
5	Fuzzy (a type of color average around the location z')

Reflection Transform (I) – the contraction of points z' back within the screen limits

Let $z' = (x, y)$, while Width and Height denote the screen dimensions (W, H). The point z' can reach the positions between 0 and $W – 1$, and 0 and $H – 1$. Make x and y positive.

0	Shrink z' to $x = W – 1 – x$ modulo W, and $y = H – 1 – y$ modulo H. Perform the pixel interpolation (rounding of pixels and enhancement of image quality).
1	If x / W is even, $x = x$ modulo W, otherwise $x = W – 1 – x$ modulo W. If y / H is even, $y = y$ modulo H, otherwise $x = W – 1 – x$ modulo W. Without the interpolation, but in use by the convolution procedure that adds a layer over images (for softening).
2	Same as 1, but with pixel interpolation.
3	$x = x$ modulo $W / 2$, $y = y$ modulo $H / 2$. With interpolation.
4	If $x > W – 1$, $x = x$ modulo W, otherwise $x = W – 1 – x$ modulo W. If $y > H – 1$, $y = y$ modulo H, otherwise $y = H – 1 – y$ modulo H. With interpolation.
5	If $x < W$, $x = x$ modulo $W / 2$, otherwise $x = W – 1 – x$ modulo W. If $y < H$, $y = y$ modulo $H / 2$, otherwise $y = H – 1 – y$ modulo H. With interpolation.
6	If x / W is even, $x = x$ modulo $W / 2$, otherwise $x = W – 1 – x$ modulo $W / 2$. If y / H is even, $y = y$ modulo $H / 2$, otherwise $x = W – 1 – x$ modulo $W / 2$. Performs the additional operation of making x, y positive, because they could become so large (>2147483647) that they wrap around and come up as negative numbers. With interpolation.
7	If x / W is even, $x = x$ modulo $W / 2$, otherwise $x = (W – 1) / 2 – x$ modulo $W / 2$.

If y / H is even, y = y modulo H / 2, otherwise x = (W – 1) / 2 – x modulo W / 2. Performs the additional operation of making x, y positive, because they could become so large (>2147483647) that they wrap around and come up as negative numbers. (Java specific.) With interpolation.

Color Gradient (G) – the color added to the points z' beyond the limit circle.

0	No Gradient (do nothing)
1	Simple Gradient (gradient affects the color)
2	Accented Gradient (accent color plays an additional role)

Gradient Shape (') – the shape of the two-dimensional shadow used for basic gray-scale coloring.

The gradient shape can be visible on the contours of a typical Julia fractal, which in turn can be uniformly colored (without any indication of the gradient shape), or varied (which shows the gradient shape). Each of 10 gradients was made by the calculation of various functions of two variables. The default gradient (0) is the radial shadow, black in the center and gradually whiter towards the perimeter.

Suppose that the Julia set is a circle. When you change the proportions of screen (by adjusting the resolution), the Julia fractal becomes an ellipse. The radial shadow also becomes elliptic. This behavior is considered more natural for a video feedback system, because its displays depend from the shape of the screen. The contours of the Julia fractal behave similarly to the images of screens within screens in the classical video feedback setup.

Gradient Direction (K) – whether the gradient shade is going from black to white or from white to black.

Fade Color Mode (F) – the fade-to color mixed with the existing color of the point.

0	Black
1	White
2	Mid-Screen Pixel Hue (select the color from the middle of the screen)
3	Not Mid-Screen Pixel Hue (select the negated color value of the mid-screen pixel)
4	Mid-Screen Pixel Hue Rotate (rotate the hue through HSV color spectrum and combine it with the mid-screen pixel color)
5	Hue Rotate (rotate the hue through HSV color spectrum)

Dampen Fade-to Color (V) – a small dampening of fade-to color.

Color Accent (H) – choose the accent color.

Color Filters (X) – the contrast enhancements.

0	None (do nothing)
1	RGB (various filters and color functions play a role)
2	Mush (various filters and color functions play a role)

Partial Inversion (Insert) – partial inversion of colors affects the fractal contours; black can become white and white black.

Thanks to the fundamental color inversion that affects the entire flow of visual data in each cycle, we have the interchanging contours, but consequentially the Julia lake flickers. The partial inversion does not affect the flicker, but acts as a practical (conventional) color inversion of the current screen contents.

Total Inversion (J) – disables the fundamental color inversion.

The flickering of the Julia lake stops, but the contours disappear as well. The visibility of fractals depends from the fractal map (since some Julia fractals do not possess any lakes), and the presence of other images on the screen (notably the screen cursors).

Convolution (Y) – adds a layer that softens and blurs the image.

Even though it is technically a blur, this improves the quality of image greatly. The default reflection transform is located in the same function, so it combines with other reflection transforms. When I disable the convolution, I achieve a speed increase from 16 to 20 frames per second with default settings.

Autorotation, Autopilot and Wanderer (,, ., p, Delete) – automatically change parameters.

Fun () – for speed purposes, we do not time your usage of Perceptron.

Inserting the images into the flow

One of the fascinating research options is to insert different images, live feed from a web camera, or geometric patterns produced in various processes into the video feedback flow. The rotating three-dimensional fractal Tree is one such insertion. It enables one to understand the morphing of images under the given geometric transformations. On small scale, the insertions are called the “color seeding”. They allow us to understand the dynamics of the given fractals. If you have any idea what to insert into the flow and how, or wish to expand the theory of video feedback fractals, please send us an email.

Be Intuitive!