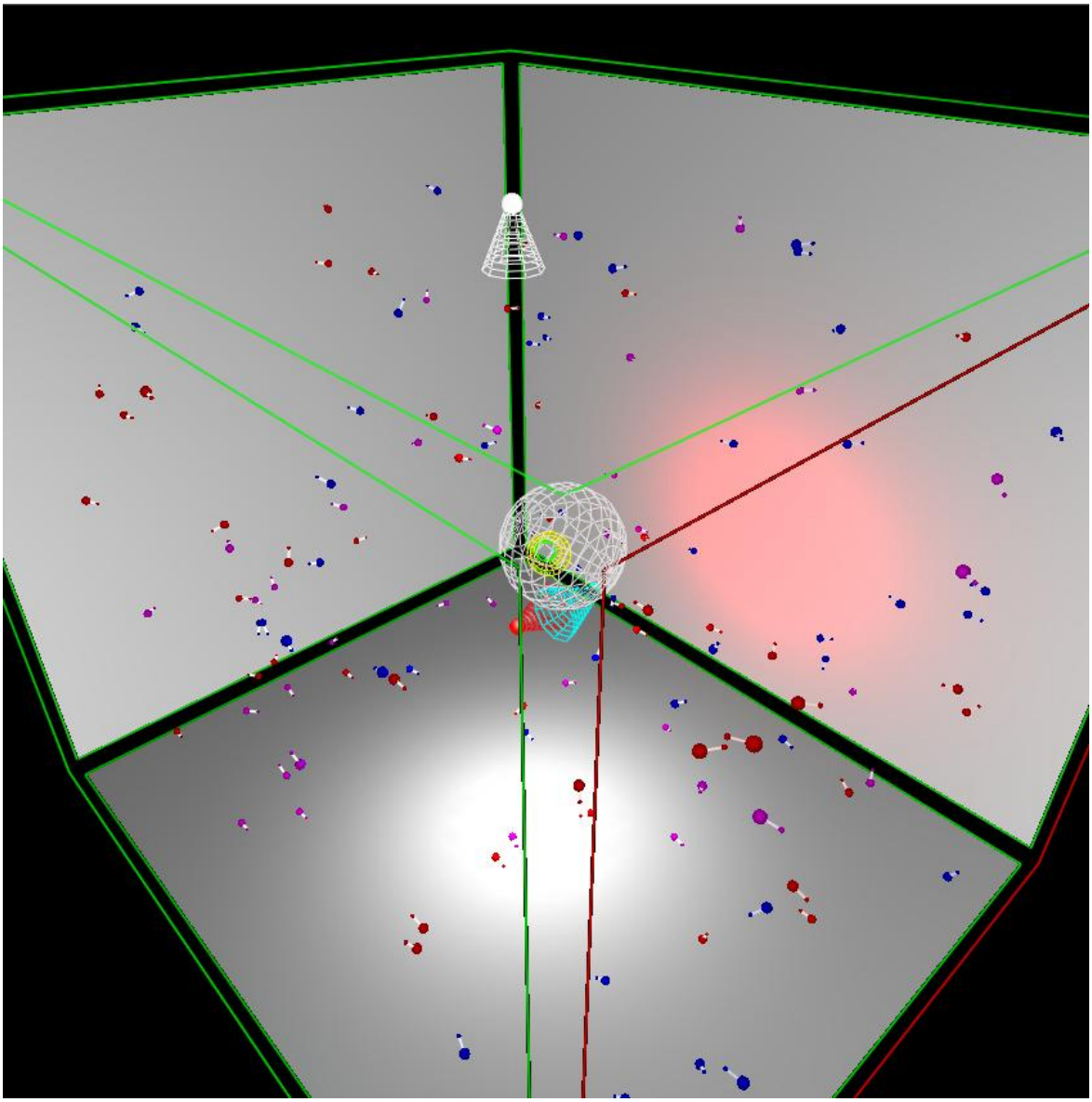
8/23/2010

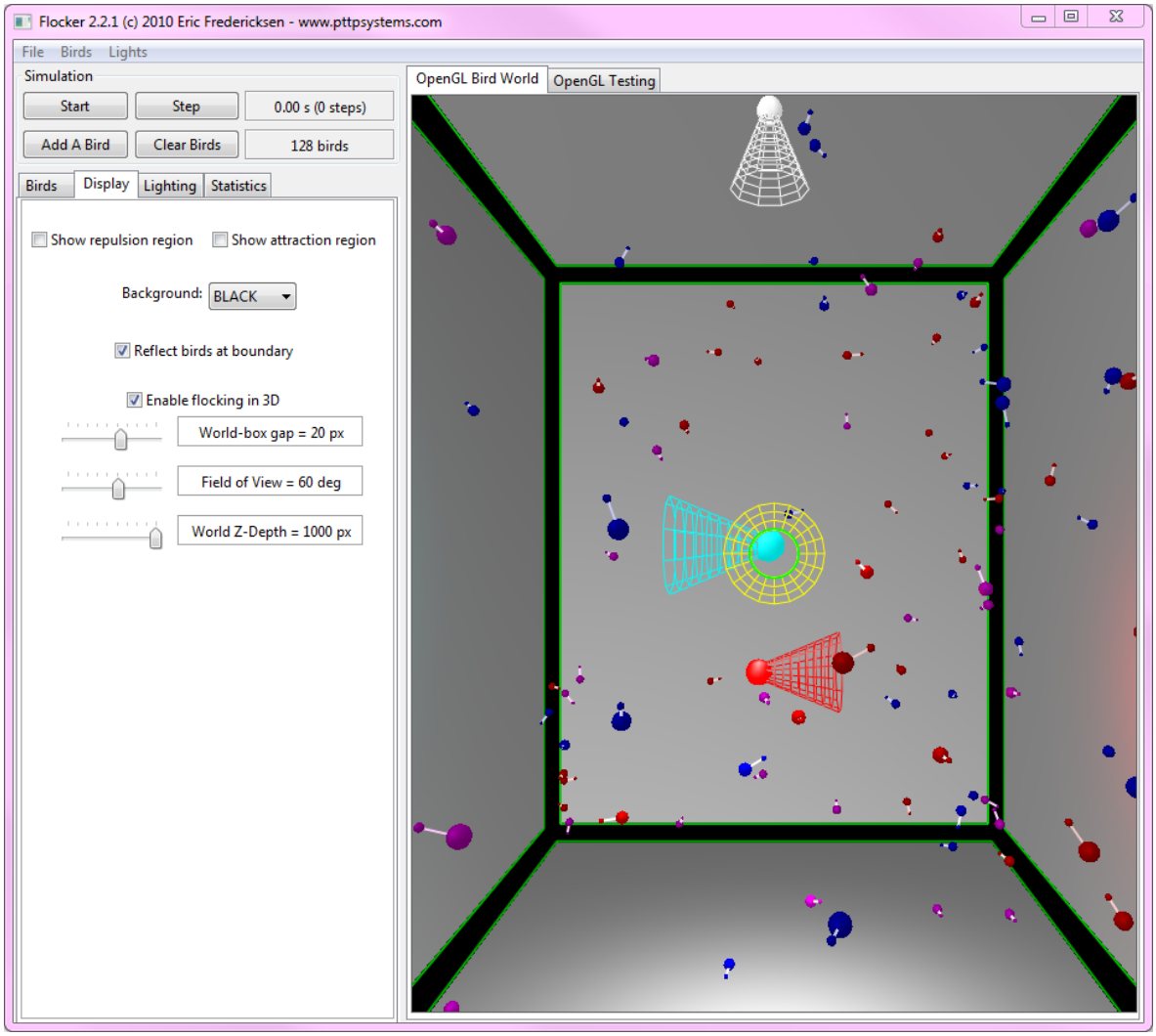
WWW.PTTPSYSTEMS.COM

FLOCKER 2.2.1



© Eric Fredericksen | eric@pttpsystems.com

Contents



Summary

Flocker started as a way for me to learn LUA (http://www.lua.org/). I thought, "Hey, I've never implemented that bird flocking thing, so why not try that while I learn a new language?" Things never stop changing when we play with software, and this little monster keeps growing.

Updates and additions in 2.2.1:

- Allowed manipulation of the world box to give a better view of teleportation vs. reflection.
- Saved field of view and world box gap to the bird configuration snapshot.
- Added a bit more geometry to bird tails.
- Fixed the "deader" bird icons (back-face culling was only showing them from one side)

Updates and additions in 2.2:

- Added a "Reset Light Defaults" menu item that resets all the lights to an off position with spotlights aimed at the center of the volume.
- Added viewpoint fly-through with simple avatar and mouse controls.
- Added a field-of-view control to control display.
- Refactored some internal code.
- Removed the testing window pane.

Note: the ***OpenGL Testing*** tab is just there for fun. Now you can fly around it. ☺

Possible Future Modifications

Future modifications that I am thinking about include

- Refactor the menu to allow saving of avatar/camera positions, and combinations of birds, lighting, etc.
- Geometry texturing, OpenGL fog, updated bird geometry
- Tag-a-long viewpoint flocking (pick a bird and auto-follow)
- Hunt-and-capture of birds with your avatar ☺
- Mouse-over statistics for a bird and graphing of statistics over time
- Environmental obstacles
- Add an eating state. For example, scavenging the dead or food spawned in random locations.
- Convert the display to a true torus.

Installation

All you need to do is extract the directory from the zip archive. Double-click ***startmeup.bat*** from the top-level directory. Alternatively, if you have LUA for Windows (http://code.google.com/p/luaforwindows/) installed then you can double-click the file named ***flocker.wlua*** in the ***Flocker*** directory.

Configuring Birds

Once the program is started you can go with the initial configuration, load one of the preconfigured simulations, or clear the birds out and start new ones using the ***Add A Bird*** button. While in 2D display mode you can drag birds around by left-clicking your mouse in the window to capture one or more at a time. Drag them where you

want. The resulting bird direction is determined by your drag direction. You can click-drag birds when the simulation is running or stopped, and you can start/stop the simulation as you like.

Viewpoint Motion in 3D Mode

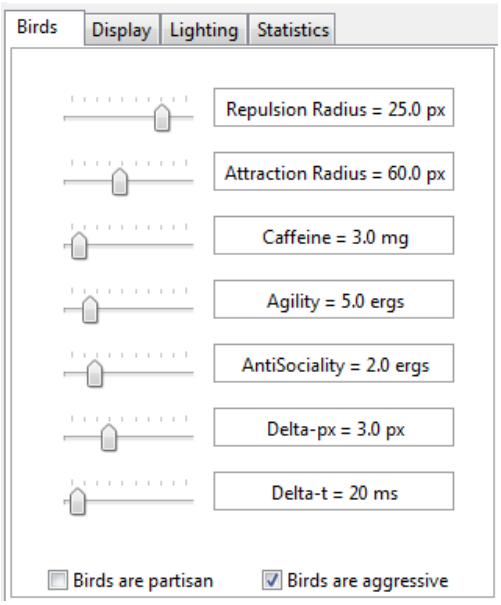
If you are familiar with the default World-of-Warcraft user interface then these actions will be familiar. ☺ You can move the viewpoint around in the 3D world after setting focus -- just click inside the display area. After you do the mouse wheel zooms your viewpoint toward and away from the wireframe avatar (the green and yellow tubular thingy). Right-click (and hold) in the display will change the cursor to a cross-hair. Your avatar will move forward as long as the mouse button is down; move the mouse to steer. Your avatar's speed will be a bit faster than whatever the bird's speed is set to (so you can catch up). A left-click (and hold) in the display allows you to move your viewpoint around the avatar by steering with the mouse. Up/down arrow is an 'autorun' toogle on/off.

> **Note: the IUP toolkit does not provide control over the mouse position, so *Flocker* can't keep it inside the window. Beware what you click on if you let the mouse get outside of the window!**

Control Tabs

I am including separate sections for each control group -- you can find those below. Use the File menu to exit or compact the SQLite database. The bird and lighting configurations can be loaded, saved, and deleted via the main menu. While in 3D mode you can click in the window and use the mouse wheel to zoom in and out (improvements TBD).

Bird Behavior Parameters



A bird's behavior follows configurable rules. If other birds are within detection range each one is considered to see if it is desirable to be close to, away from, or is perhaps a candidate for attack. The detection distance is configurable, as is the repulsion distance.

If another bird gets within the repulsion distance then the bird will try to move away from that other bird.

Birds can be curious, and the magnitude of their curiosity can be determined by the amount of coffee they drink.

The flight envelope of a bird can be modified by adjusting the agility parameter. The great the agility, the more tightly a bird can turn when flocking or fleeing or fighting.

When another bird gets within the repulsion radius, a bird has a configurable propensity (my neologism: antisociality) toward moving away from that other bird.

Bird flocking is essentially a simulation that approximates dynamics of motion in time slices. The time slice size and the per-slice distance travelled together determine a bird's flight speed.

Birds can flock with any other bird, or you can tell them to be partisan. Partisan birds only flock with birds of the same color.

Young adult birds (see description of states in this document) are potentially aggressive. If aggression is enabled then a bird will try to drive away other young adults, as determined by the partisan setting. They do this by attacking that bird, pursuing and biting it.

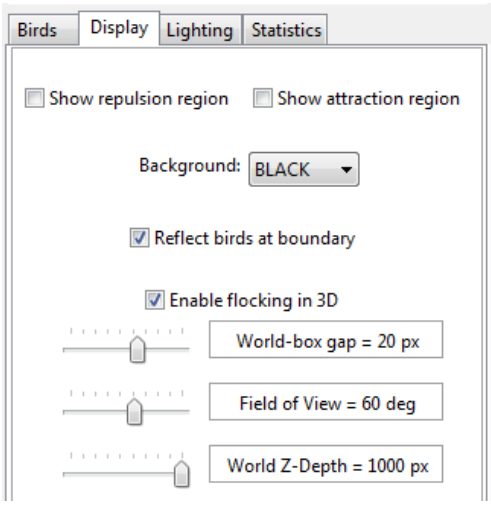
If a bird finds itself under attack before it can attack some other bird it will flee. If a young adult gets bitten enough it can die. Look for laser beams and dead birds during the simulation.

Repulsion Radius	The size of a bird's personal space
Attraction Radius	The distance a bird can see (or at least car about) other birds
Caffeine	How jittery, or curious, a bird is while flocking
Agility	How well a bird can turn and follow
Antisociality	How strongly a bird works to preserve its personal space
Delta-px	Bird movement distance per iteration of the simulation
Delta-t	Temporal step of the iteration (Delta-t also determines bird age change per iteration)
Birds are partisan	Toggle whether birds care or don't about other bird types
Birds are aggressive	Enables aggressive behavior when the bird is in a young **adult** state

Behavioral control description summary

Display Parameters

You can modify certain aspects of the bird world. For example, you can toggle the visual display of the repulsion and detection regions to make it easier to understand the flocking behavior.



You can configure the background color. This mostly affects the 2D display, as the 3D display includes a bounding box to delimit the flocking volume and give lights more things to brighten and darken.

You can decide whether birds teleport to another location when they cross the flocking boundary, or reflect/bounce off that boundary.

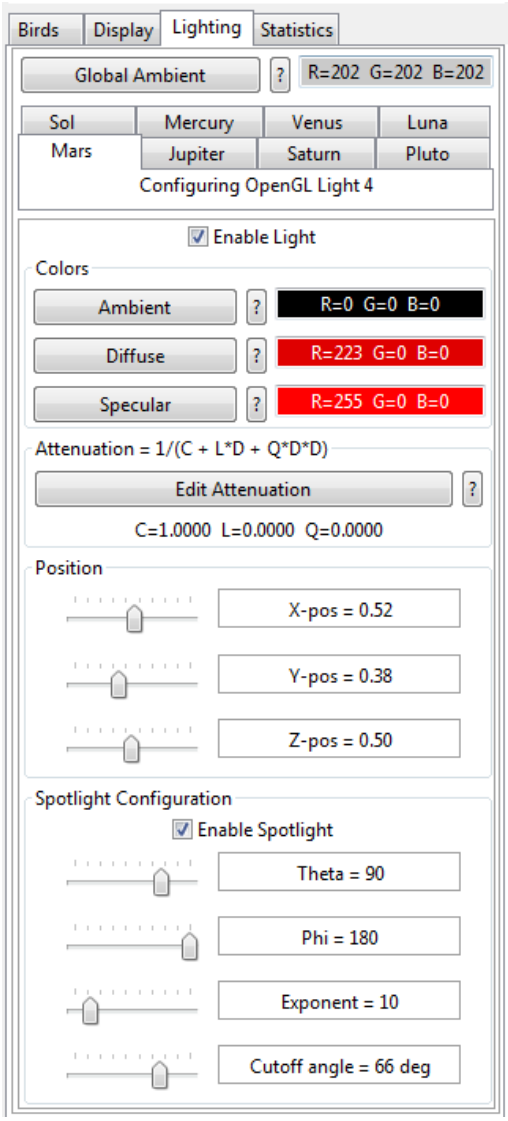
You can dynamically switch between 2D and 3D displays, adjust the world box boundary, and modify the field-of-view width and depth of the 3D world in real time.

Note that switching to 2D flocking from 3D will set the z-components of a bird's position and direction to 0. In other words, the z-component is lost. If you switch from 2D to 3D then birds will begin wandering in 3D again as long as they have some caffeine in their system.

Show repulsion region	Show personal space on the display - a circle around the bird
Show attraction region	Show "interest" space on the display - a circle around the bird
Background	Choose a background color for the world
Reflect birds at boundary	Toggle between torus "teleportation" at the edges and reflecting direction (bouncing)
Enable flocking in 3D	Toggle between 3D and 2D display and flocking
World Z-Depth	Choose the depth of the 3D volume

Display control description summary

Lighting Configuration



Lighting in OpenGL is a fairly involved topic. I won't try to explain how lighting works in this document because I don't have the space, I'm lazy, and it is has been well done in other places. ☺

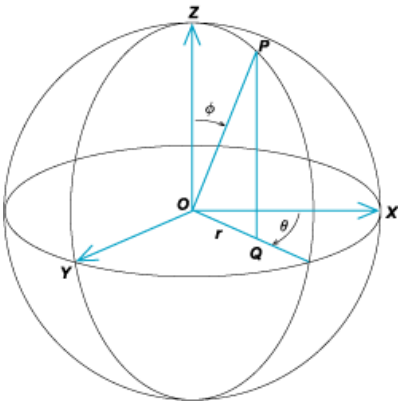
What I have done, however, is to provide configurability of nearly all of the OpenGL implemented lighting system parameters.

You can adjust the global ambient lighting model (but not yet choose/configure local viewer or two sided).

You can individually enable and configure 8 different lights -- all the lights provided by the LuaGL V1.4 package on my system.

You can set the relative position of each light within the bird flocking area/volume as well as the attenuation parameters for the light. Lights remain in that relative position when you resize the volume.

> *Note: OpenGL also provides for lights with directional-at-infinity behavior, in which case the position vector is really the light's direction vector. Accommodating both types would make the interface even more crowded than it already is. I wanted omnidirectional lights with attenuation, so that's what I added for now. ☺*



Each light can be configured for its ambient, diffuse, and specular colors as shown in the background for the RGB value display.

Lights can be converted to a spot source. Spotlights are directional with direction configured in spherical coordinates.

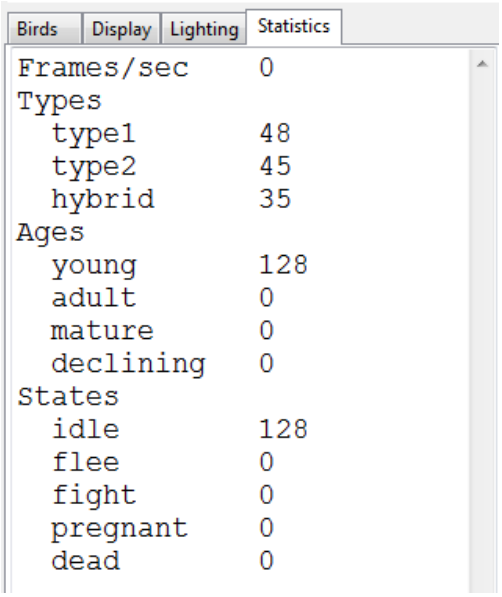
You can also configure the behavior of the falloff of intensity as provided by OpenGL. Light intensity will be curtailed as the ***cosine*** of the angle from the light's direction raised to the power of ***Exponent*** parameter. There is then a hard cutoff of the light at an angle from the light's direction as specified by the ***Cutoff*** parameter.

Global ambient	Configure the color of global ambient lighting
Light selection tabs	The lights are named after planets and moons. Yes, I'm still giving Pluto some props. Select a tab to configure that light.
Enable light	Turn a light on or off completely
Colors	Individually configure the ambient, diffuse, and specular colors emitted by the chosen light. Click the [?] help button for more detailed information via Wikipedia (which rocks, BTW).
Attenuation	Configure a light's attenuation properties. The default is no attenuation. The OpenGL attenuation model is a quadratic reduction with distance.
Position	Use the sliders to move the light around inside the viewing volume. Relative position is conserved across viewing volume resizing.
Enable spotlight	Turn the omnidirectional light into a spot source. The light's icon changes accordingly
Theta and Phi	Spotlight direction is configured in spherical coordinates. Theta rotates the light around the Y-axis and Phi rotates the light around the Z-axis. *Note: the Wikipedia entry for spherical coordinates uses the less-prevalent coordinates of Rho, Theta, Phi. My references use Rho, Phi, Theta so that is what I've implemented here.*
Exponent and Cutoff	Spotlight intensity falls off smoothly from the center following the cosine of the angle from the light direction raised to the power of ***Exponent***, $cos(\Phi)^{Exponent}$, and goes to zero at the angle given by ***Cutoff***.

Lighting control description summary

Statistics Display

Birds can multiply pretty quickly once they mature. This control provides some aggregated statistics of how many birds exist of each type, of each age, and in each state.



Bird Behavior is Driven by States

Bird behavior is determined by two parallel state-based systems: one set of states for aging and another set for behavior. The aging status serve as a trigger, or gate, for transitions to behavioral states. The state diagram with transitions is shown below. Age related actions are color coded by age.

Age states implemented include:

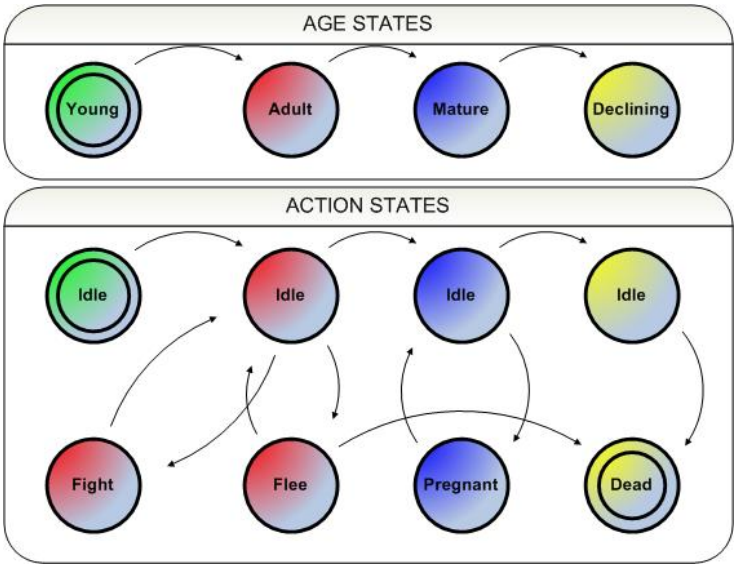
Young Health is increasing, and the bird is small.

Adult Health has maxed out, the bird has grown a bit, and may become aggressive toward other young (non mature) adults.

Mature The bird is full grown and gives up its delinquent ways, becoming non-aggressive. It is now eligible to mate with nearby birds.

Declining The bird is past its prime and starts to decrease in health. The bird has shrunk with age.

Behavioral states implemented include:



Idle Flocking, the idle state, is the default state when the bird isn't busy with something of higher priority.

Fight A young **adult** is vying for dominance by attacking some other young adult. After a bit of bad behavior the bird homes back to its mates.

Flee The goal is to get away from its attacker(s). This bird can only be attacked when it is a young **adult**, and by a young **adult**. When it feels safe again it homes back to its friends.

Pregnant Birds can give birth, but only when they reach a **mature** age state. If a red and blue bird mate their offspring is purple. Birds are hermaphroditic.

Dead Birds die when they grow old or get beaten up too badly by young adults. Keep your eyes peeled for young **adult** birds being pursued by an angry mob.

Files in the Package

startmeup.bat	DOS batch file to start the simulation (because I assume you don't have LUA installed)
Flocker\flocker.wlua	The UI code based on Portable User Interface and Canvas Draw (for now) from Tecgraf: http://www.tecgraf.puc-rio.br/iup/ http://www.tecgraf.puc-rio.br/cd/
Flocker\flockerengine.lua	A small "engine" that manages the simulation, asking the birds to step through their OODA loop (http://en.wikipedia.org/wiki/OODA)
Flocker\flockerbirds.lua	This is where bird behavior, through states, is defined.
Flocker\flockerutils.lua	Some tools used by everyone (e.g., 2D vectors and math)
Flocker\flockerdb.lua	The LUA code used by the UI to load, save, delete snapshots to the DB
Flocker\flockerogl.lua	The LUA code that manages the use of OpenGL (via LuaGL)
Flocker\flockerdb.lua	The LUA code used by the UI to load, save, delete things to the DB
Flocker\flocker.sqlite	The SQLite database for bird and lighting configuration snapshots and UI configuration saving. Includes sample bird and lighting configurations.
FLOCKER LICENSE.txt	The ***Flocker*** license description.
LUA LICENCSE.txt	A copy of the LUA license as contained in the Google-supplied package.
Flocker*.dll, *.txt	Files distributed from the LUA for Windows package.

License

Flocker code written by Eric Fredericksen uses the GNU AFFERO GENERAL PUBLIC LICENSE Version 3, 19 November 2007: see the included license file and blurbs in the code.

The LUA code and related libraries use BSD like and MIT licenses: see the included files for details. The files I distribute are extracted from the LUA for Windows package, which you can get from http://code.google.com/p/luaforwindows/.