λ Lounge

OpenGL in Common Lisp

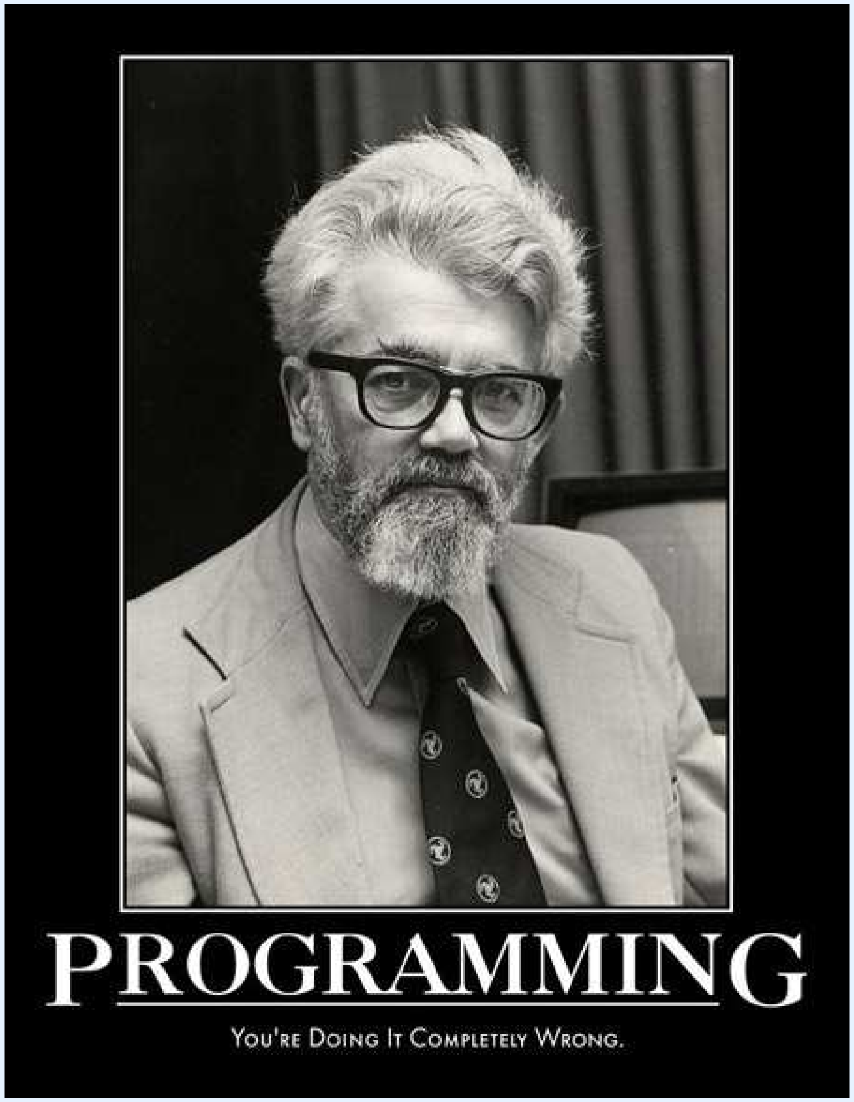
Christopher Mark Gore

`http://www.cgore.com`

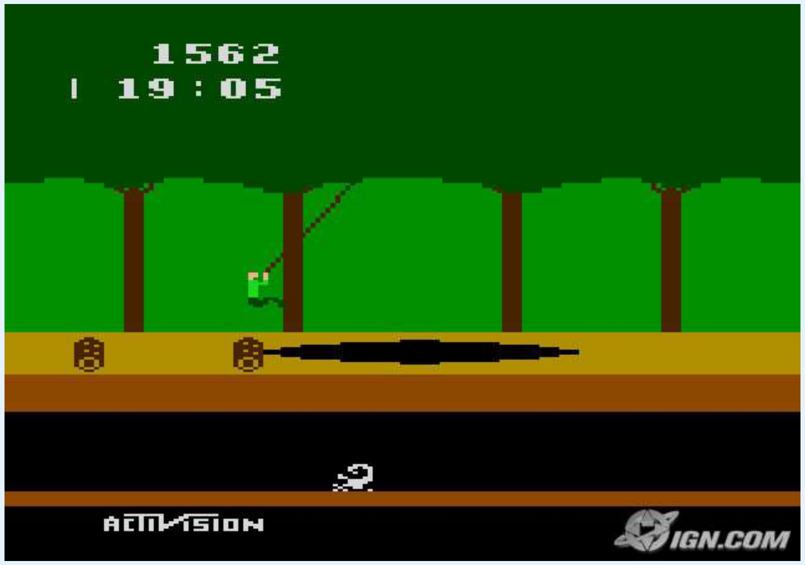
`cgore@cgore.com`

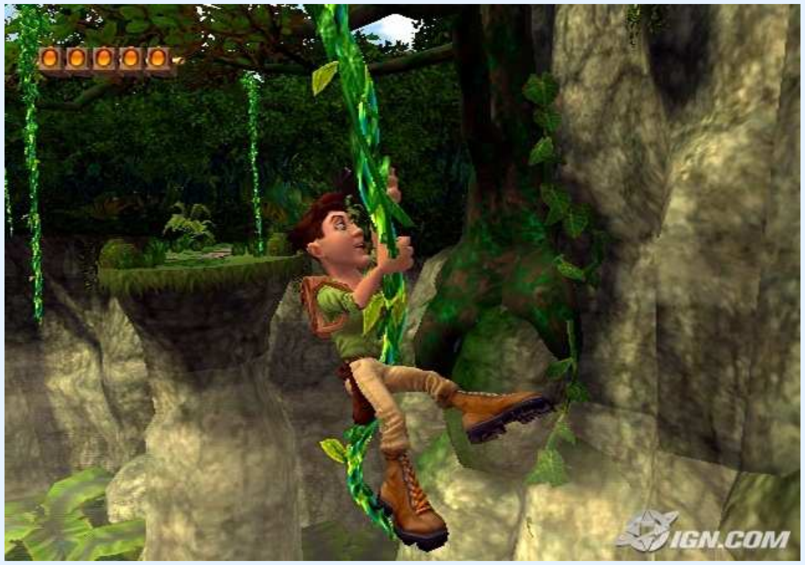
Thursday, November 7, AD 2013

Lisp is Cool!



3D is Cool!





Except for 3D Pitfall, which looks really lame.

Getting Started

1. Install Linux.

```
http://aptosid.com
```

2. Install SBCL and some libraries.

```
apt-get install sbcl{,-doc,-source} \
cl-{asdf,cffi}
```

3. Install Emacs and SLIME *(Not strictly required.)*

```
apt-get install emacs,-goodies-el cl-swank \
cl-swank slime common-lisp-controller
```

4. Install OpenGL.

```
apt-get install libgl1-mesa-dev \
libglu1-mesa{,-dev} libglut3{,-dev} ...
```

Extra Libraries

- Σ, my library of random useful things in Common Lisp.
  `https://github.com/cgore/sigma`
  (Almost completely undocumented.)

- `cl-opengl`, or else we need to do a lot more work.
  `https://github.com/3b/cl-opengl`
  This library provides `cl-glu` and `cl-glut`.

Getting Libraries via Quicklisp

Quicklisp is the less irritating way to get Common Lisp libraries. It is available at `http://www.quicklisp.org`.

```
curl -O http://beta.quicklisp.org/quicklisp.lisp
sbcl --load quicklisp.lisp
(quicklisp-quickstart:install)
(ql:quickload "cl-opengl")
```

Now we should have a working OpenGL in Common Lisp.

Hello Cube

The simplest thing to do in 3D is a plain cube. This is a good test to see if the libraries and dependencies are all okay. Cf. `source/hello-cube.lisp`, **run** `(hello-cube)`.

Handling Keypresses

It would be nice if we could quit the program just by pressing **Esc**. Cf. `source/quit-button.lisp`, **run** `(quit-button)`.

```
(defmethod glut:keyboard
           ((w quit-button-window) key x y)
  (declare (ignore x y))
  (when (eql key #\Esc)
    (glut:destroy-current-window)))
```

Changing Colors

We would like to be able to change the colors of the cube. Cf. `source/colors.lisp`, run `(colors)`.

We need new accessors on the window class:

```
((red :accessor red :initform 1)
 (green :accessor green :initform 1)
 (blue :accessor blue :initform 1))
```

We change the color definition:

```
(gl:color (red w) (green w) (blue w))
```

We call an update function:

```
(glut:post-redisplay)
```

Moving the Camera

We would like to be able to move around the camera within the scene. Cf. `source/movement.lisp`, run `(movement)`.

We make class attributes and keyboard code like with the colors, and update the camera like this:

```
(glu:look-at (eye-x w) (eye-y w) (eye-z w)
             0 0 0  ; look pos
             0 1 0) ; up vector
```

TODO ...

Questions?