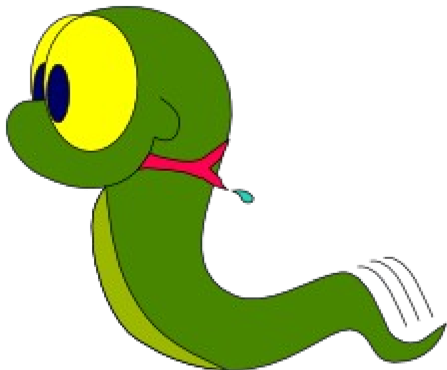
ctypes

Direct access to happiness.dll

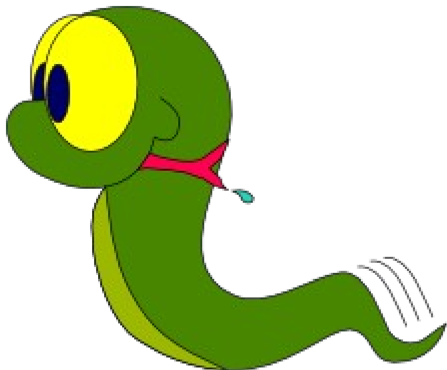
Mike C. Fletcher – VRPlumber Consulting Inc.



Who is this guy

PyOpenGL lead developer
(other stuff too)

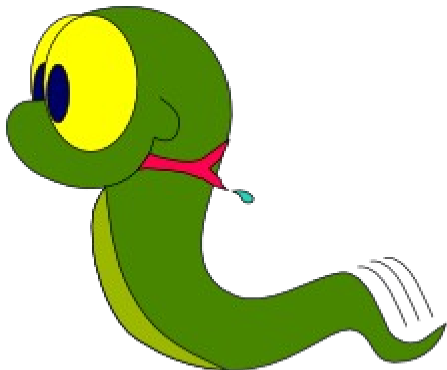
OpenGLContext, SimpleParse, StarPy, TTFQuery, BasicProperty, PyDispatcher
(and a consultant on VoIP
and other stuff)



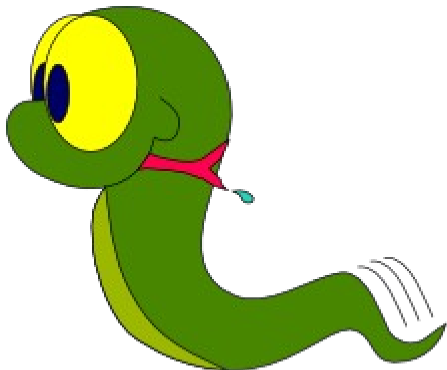
What I've done lately

Rewrote PyOpenGL from SWIG to ctypes

(Why?)



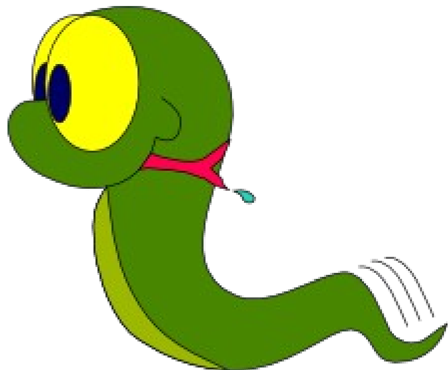
Python will be faster



Because it must

The language is fine

We need speed to expand into new areas (e.g. games)



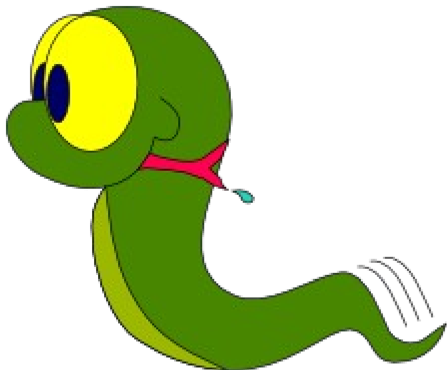
Or PyPy will replace it

Or IronPython

Or OLPC-Python

Or Parrot

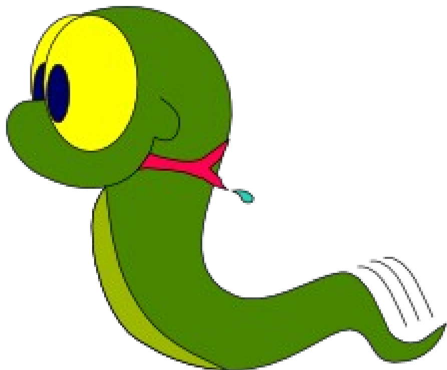
Or something else...



(And no...)

We really don't care

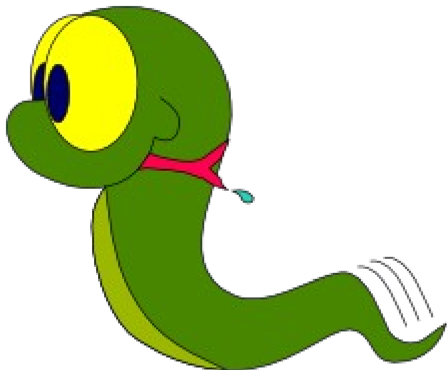
about a new print syntax



As Python accellerates

We can think about replacing C

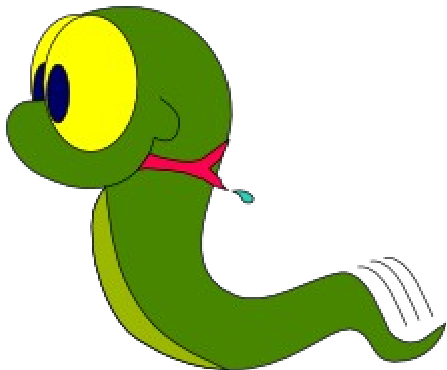
(Particularly C extensions)



Extensions were about

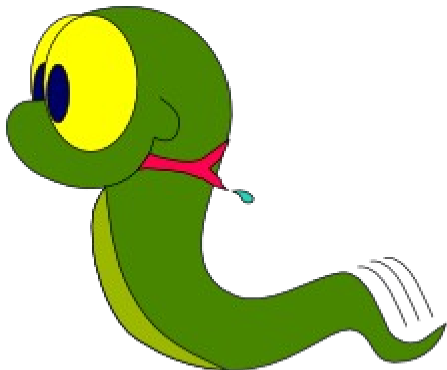
Speed

(zoom, zoom)



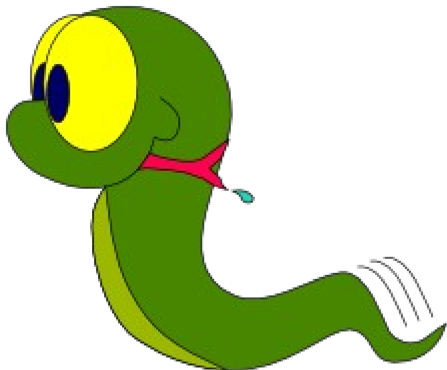
But also

Pre-written libraries of code



When $s_{\text{Python}} \sim= s_C$

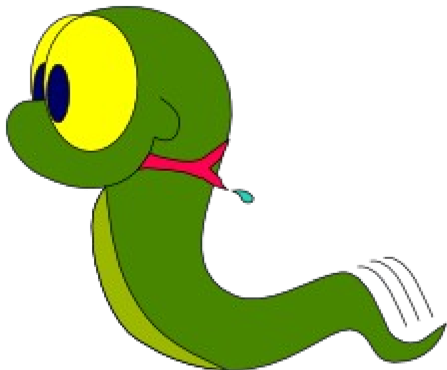
(Get working on this peoples)



We still need

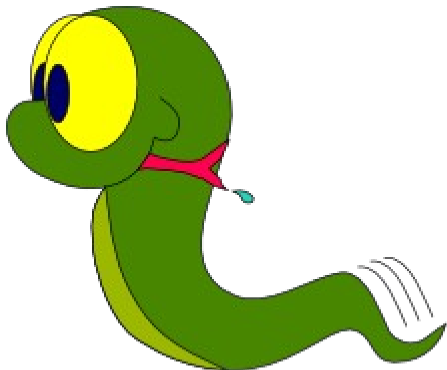
Access to pre-written libraries of code

(From pure Python)



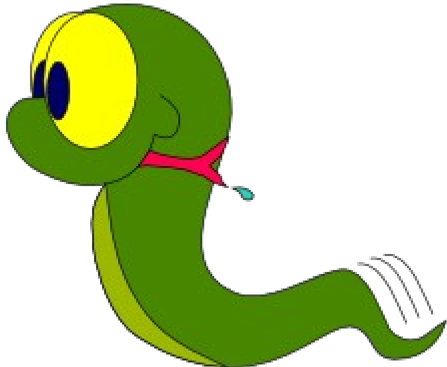
Lucky we already have it

It's been around for years



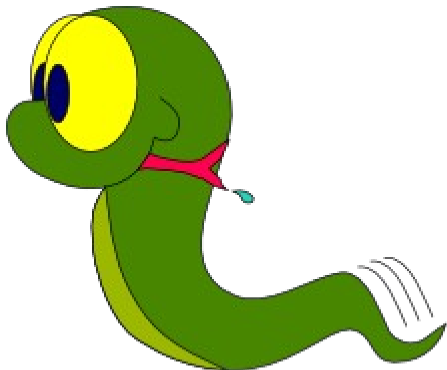
Hacker's ctypes (old school)

```
>>> import ctypes
>>> happiness = ctypes.cdll.LoadLibrary(
    './happiness.so'
)
>>> happiness.hello( 'Hello world %i\n', 42 )
Hello world 42
15  # return value
```



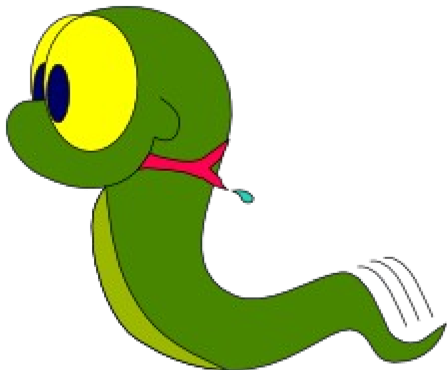
Yawn

(In a mind blowingly “cool” sort of way)



That is so 2004

Just a hacker's toy

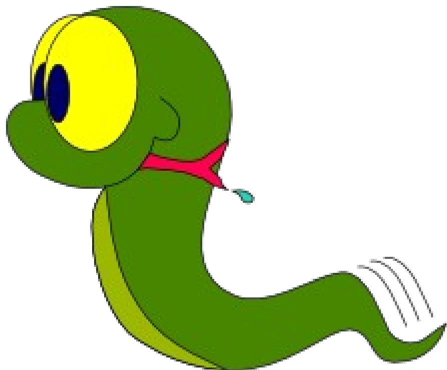


But it was COOL

We could poke deep in the machine

Twiddle random bits

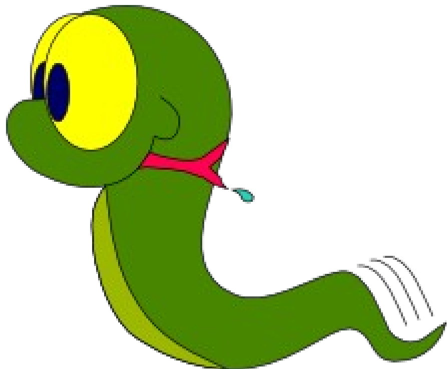
Make things happen



Um, we *said* “Yawn”

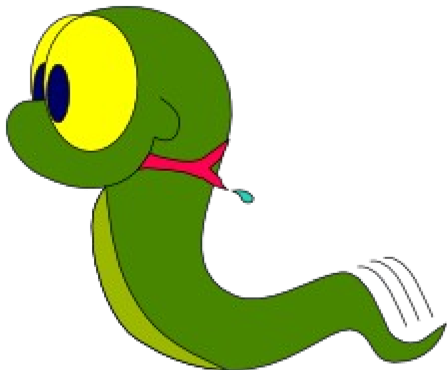
Hacker's backwater for years

(No-one really cared)



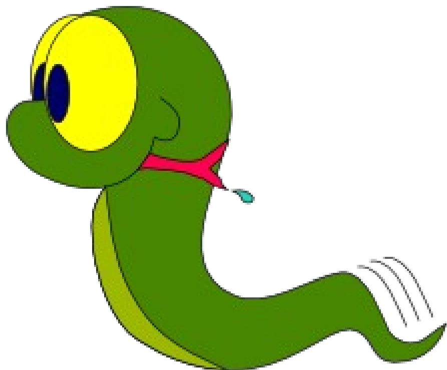
What's different?

Standard library inclusion



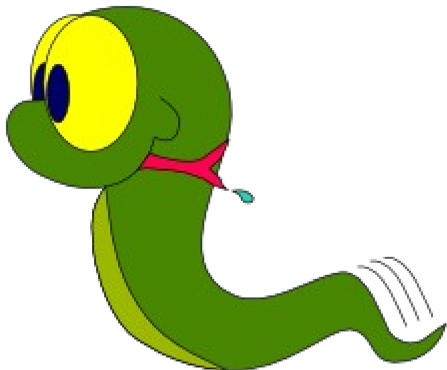
What's different?

Automated code generation



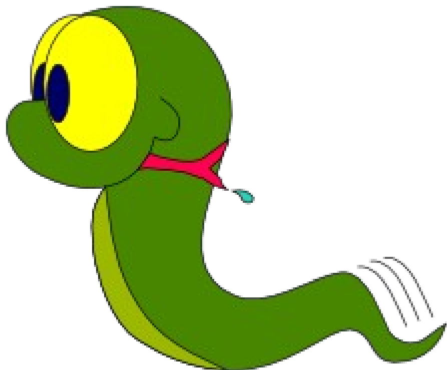
What's different?

PyPy support



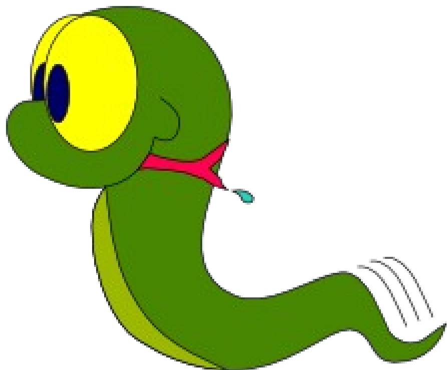
What's different?

Numpy Support



What's different?

Bigger projects possible



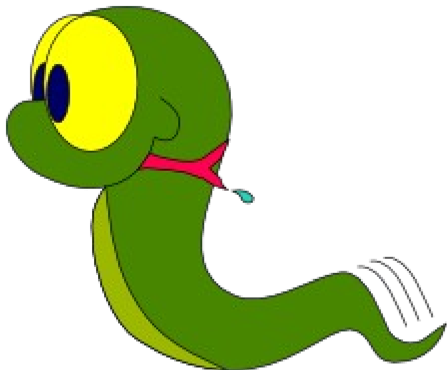
Bigger, you say?

Comtypes

Pygame-ctypes

Pyglet

PyOpenGL



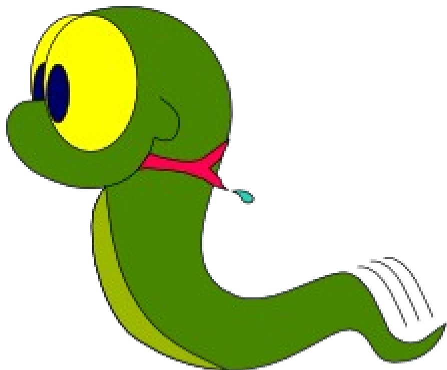
PyOpenGL Scale

2189 C functions

3475 constant definitions

262 extension modules

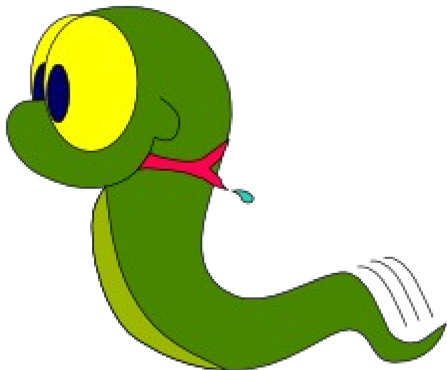
6 possible core versions



Where we came from

1.x Manual Wrapping

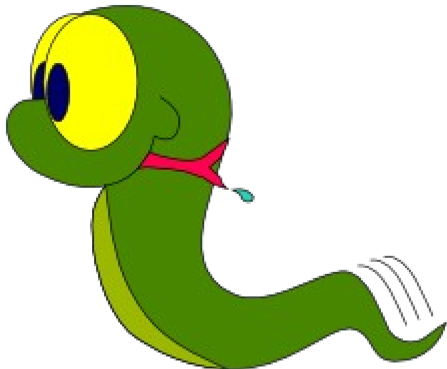
2.x SWIG Wrapping with custom distutils



1.x died years ago

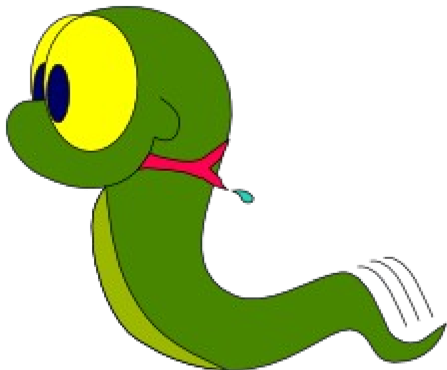
Manual wrapping way too time consuming

(This is where I came into the picture, life support for a dying project)



Really

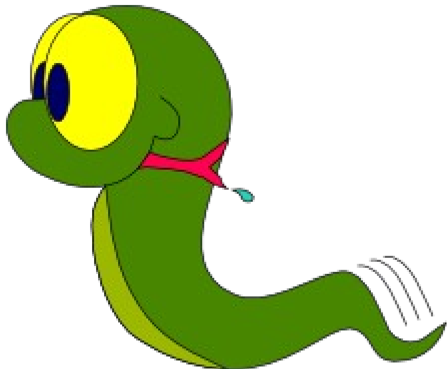
No one wants to code in C



The promised land

SWIG

(Tarn decided to use this to rewrite the project)

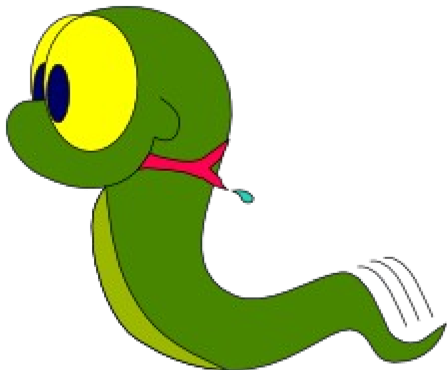


Macro headaches

Level upon levels of macro expansion

SWIG typemaps, SWIG macros, macros, macro-expanded utility libraries

(Easy to write, extremely difficult to maintain)



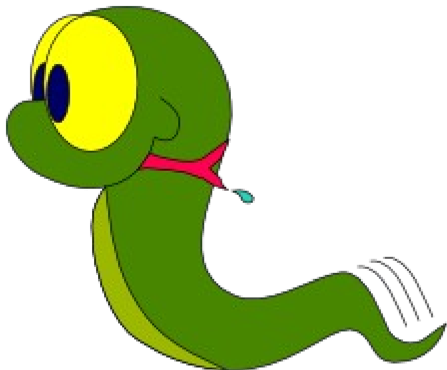
C compilation problems

Complex build process

Edit/compile/run cycle of 20 minutes+

Togl build procedure constantly broken

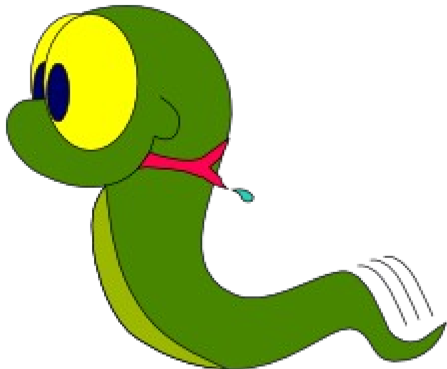
No compiler on one platform



Developer fatigue

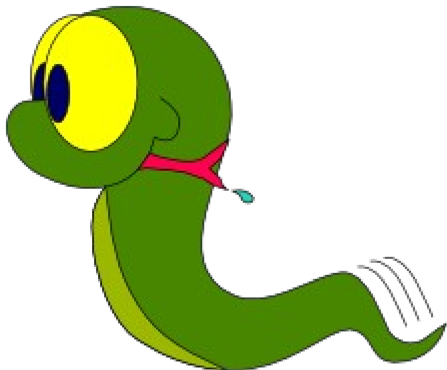
No one wanted to do the day-to-day stuff

(Because it was such a pain)



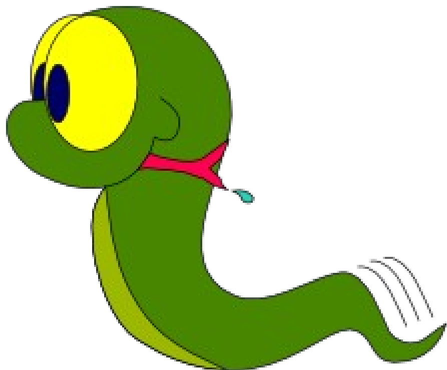
Few joined, none stayed

Figuring out how to start was way too hard



So I was going to dump PyOpenGL

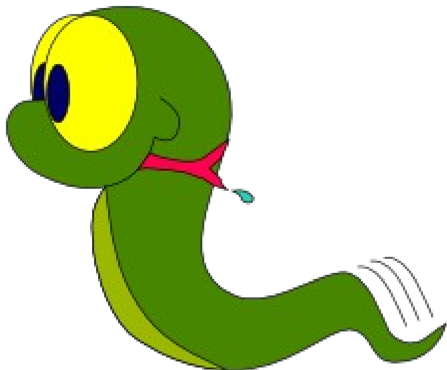
Wasn't enough fun to spend my free time on it



Won't someone think of the users?

70+ downloads a day

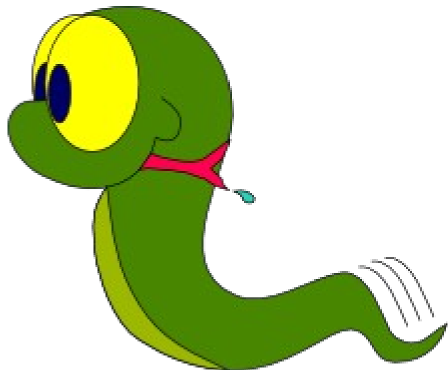
(outside the distributions or applications)



Won't someone think of the users?

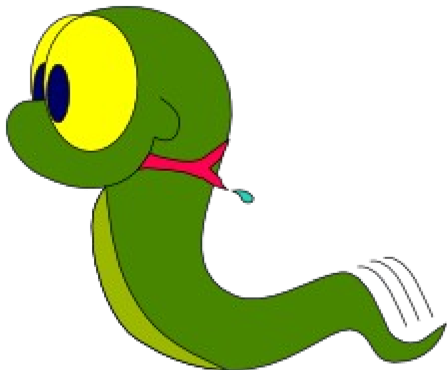
Hundreds, maybe thousands of applications

(Science, extension systems, graphic libraries)



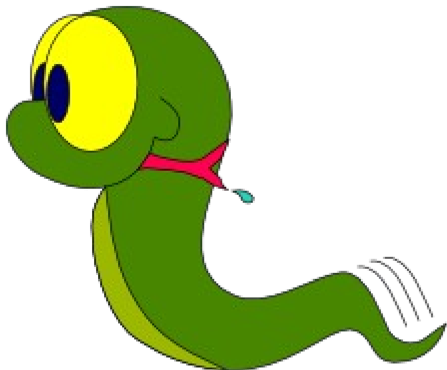
We should try ctypes

(Rene suggested it IIRC)



First tests

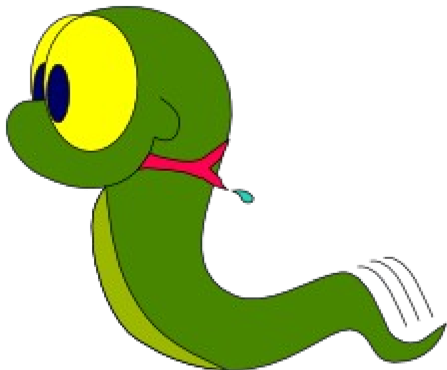
Could create C-like API easily
with custom (hacky) auto-generation



But it wasn't compatible

So I ignored it for a while

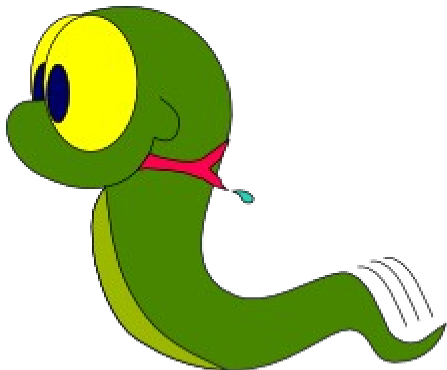
(And things got worse)



Second tests

Not "can we wrap OpenGL"

"Can we create PyOpenGL"

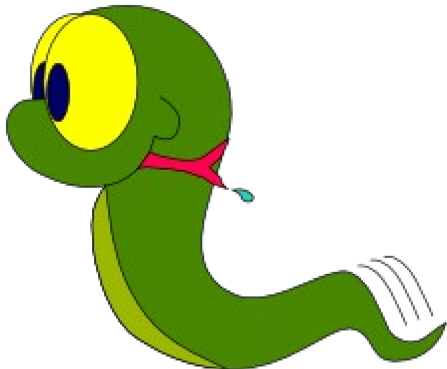


The goal

Fully compatible with PyOpenGL 2.x
(reasonably compatible)

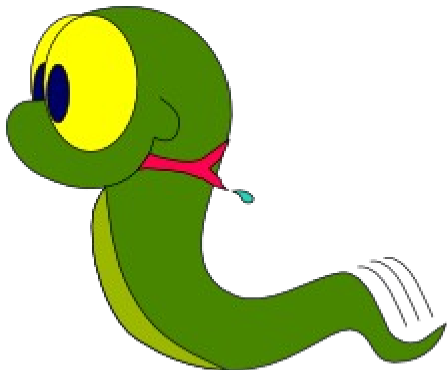
With half a dozen new fixes/features

Full extension coverage



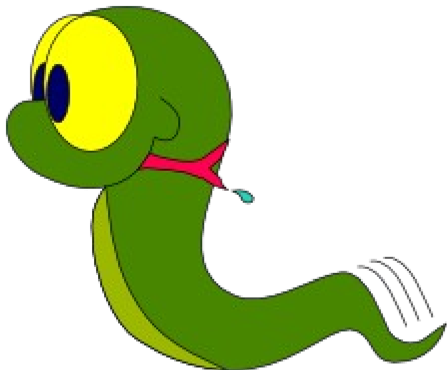
A few bugs early on

AMD64 platform issues, mostly



But we could make it work

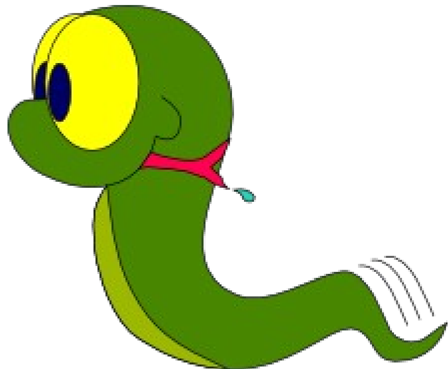
So we did



Array handling

3 different array systems (now pluggable)

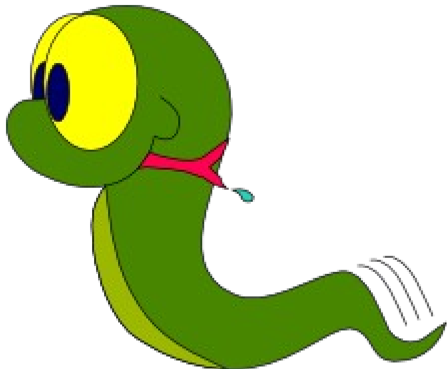
Fairly trivial, it turns out
(Even easier now)



No high-level automation

No automated type/name matching

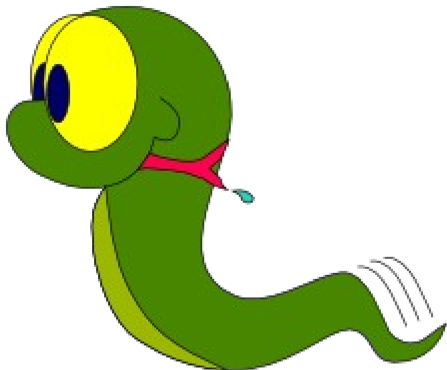
(Yay, wrote it in Python)



Library loading

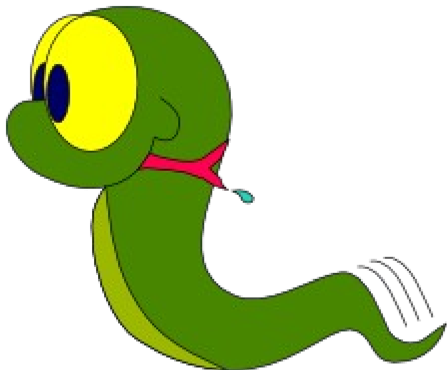
Platform specific (no big deal)

Needs to be available (hmm)
Needs to be dynamic library



No C++

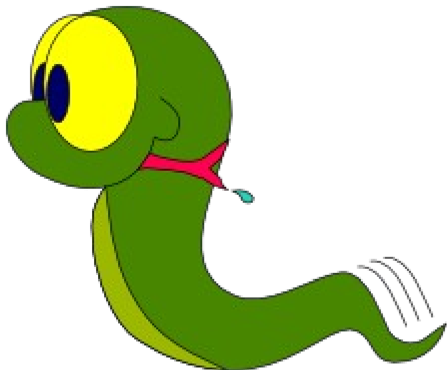
We don't care, others will



Macro problems

We don't have a lot of them

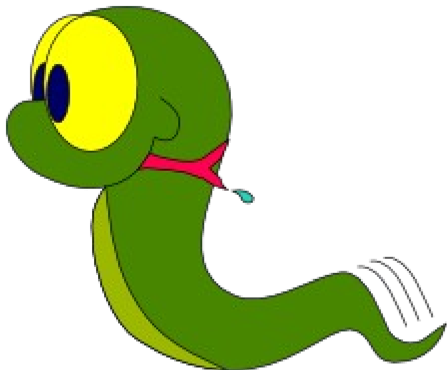
Just hacked around them in Python



Speed problems

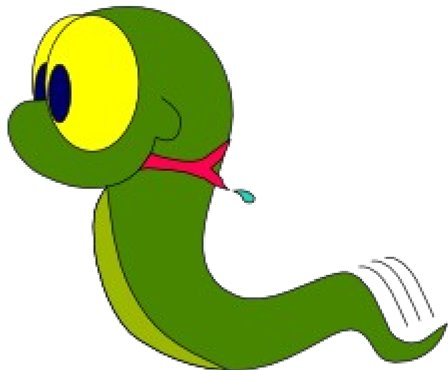
2-5x slower

(Delicious irony of slowing down to speed up)

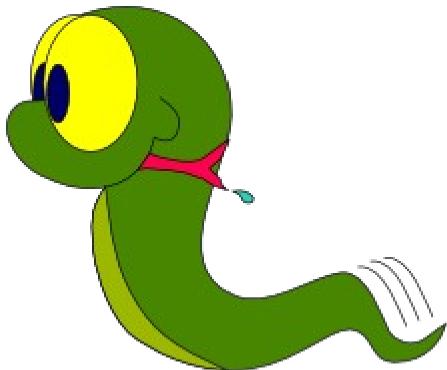


Documentation

Not as extensive as you'd want

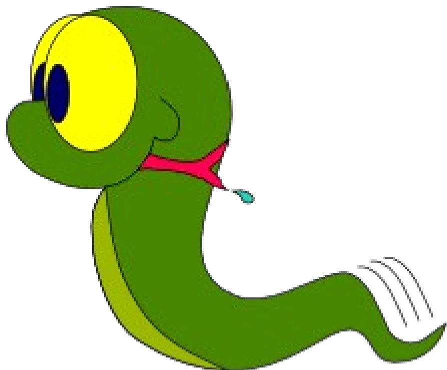


What we got



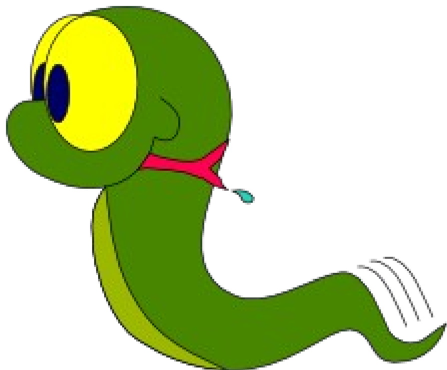
No C coding

No one wants to code in C



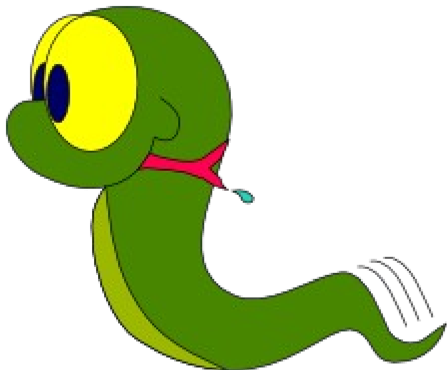
Lower barrier to entry

Hack from day one



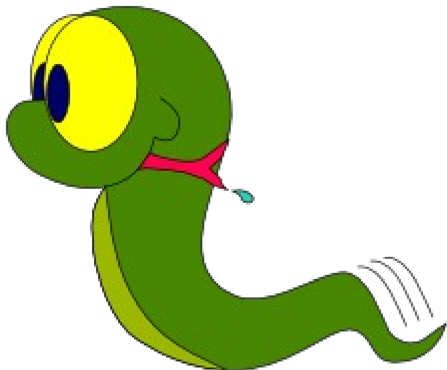
Easier to contribute

Developer's “hacks” integrate easily



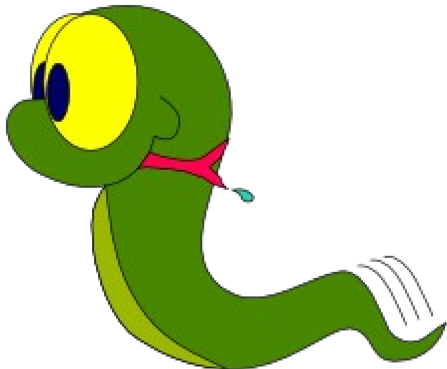
Easier to install

easy_install PyOpenGL



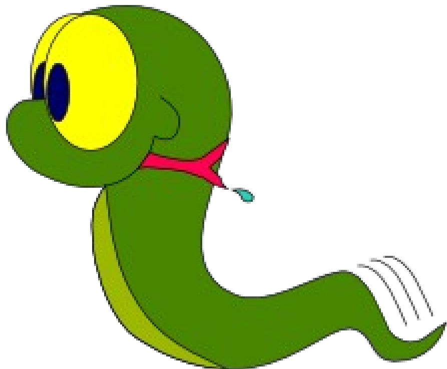
Easier to build

(do nothing)



Easier to debug

Walk through the whole wrapping process

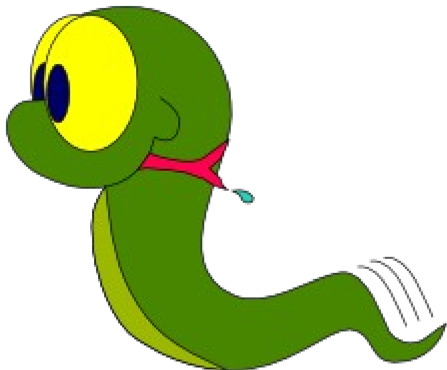


More coverage

Core library 1.3 through 2.0 (automatic)

All registered extensions

Silly little regex script creates them
(Pyglet guys have a
more advanced wrapper)

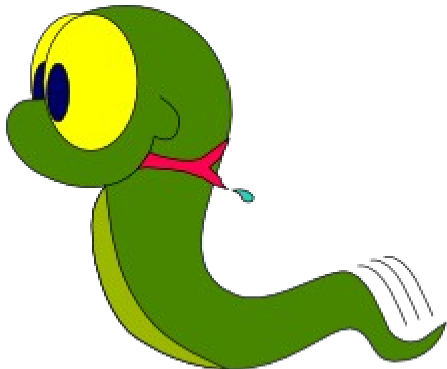


New features

Pluggable data-format support

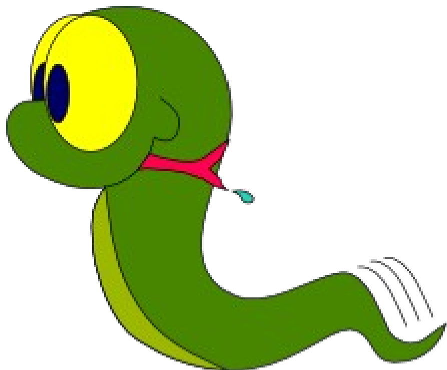
Optional logged operation

Run-time binding (dll substitution)



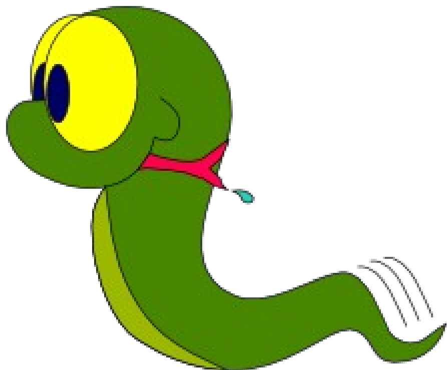
Compatible

Some users don't even realise it's a new technology, it's just “3.0”



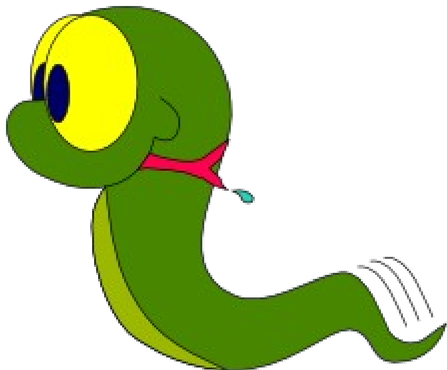
We're future-facing again

Type inferencing ready



We're future-facing again

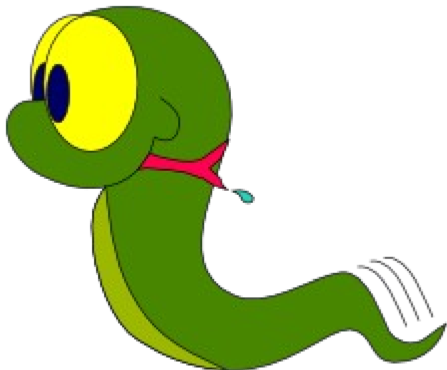
PyPy ready



We're future-facing again

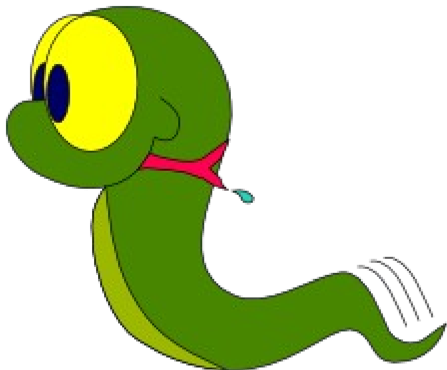
Numpy compatible

(pluggable data-types throughout)



I'm motivated again

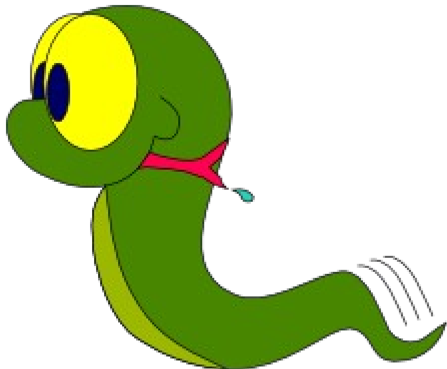
Development is fun



Why ctypes?

You have libraries in C you want to use

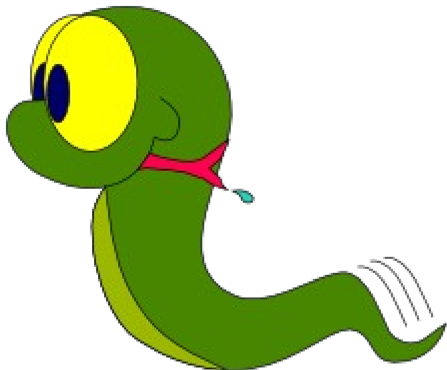
(Loadable libraries without many macros)



Why ctypes?

You want to code in Python

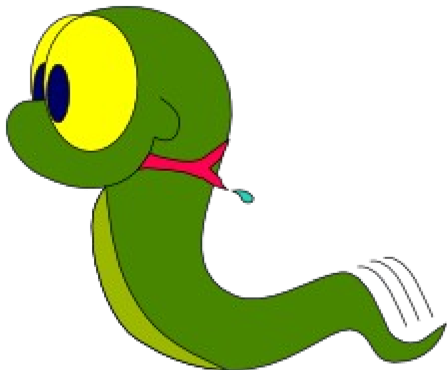
(No-one wants to code in C)



Why ctypes?

You want to avoid users having to compile your code

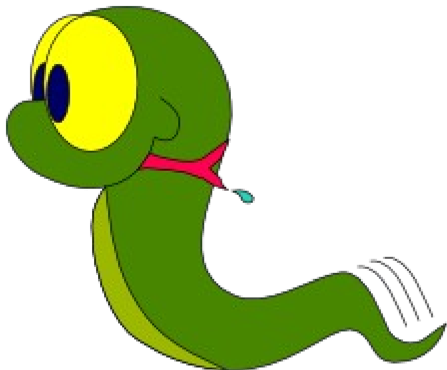
(Drop a pure-Python .egg)



You want to code Python

Because Python is fun

(and no-one wants to code in C)



Mike C. Fletcher – VRPlumber Consulting Inc.

Why ctypes?

Because you want to
load happiness.dll
into your namespace