The Architecture of GlowScript VPython

Bruce Sherwood

Visiting Scholar, Department of Physics
University of North Texas

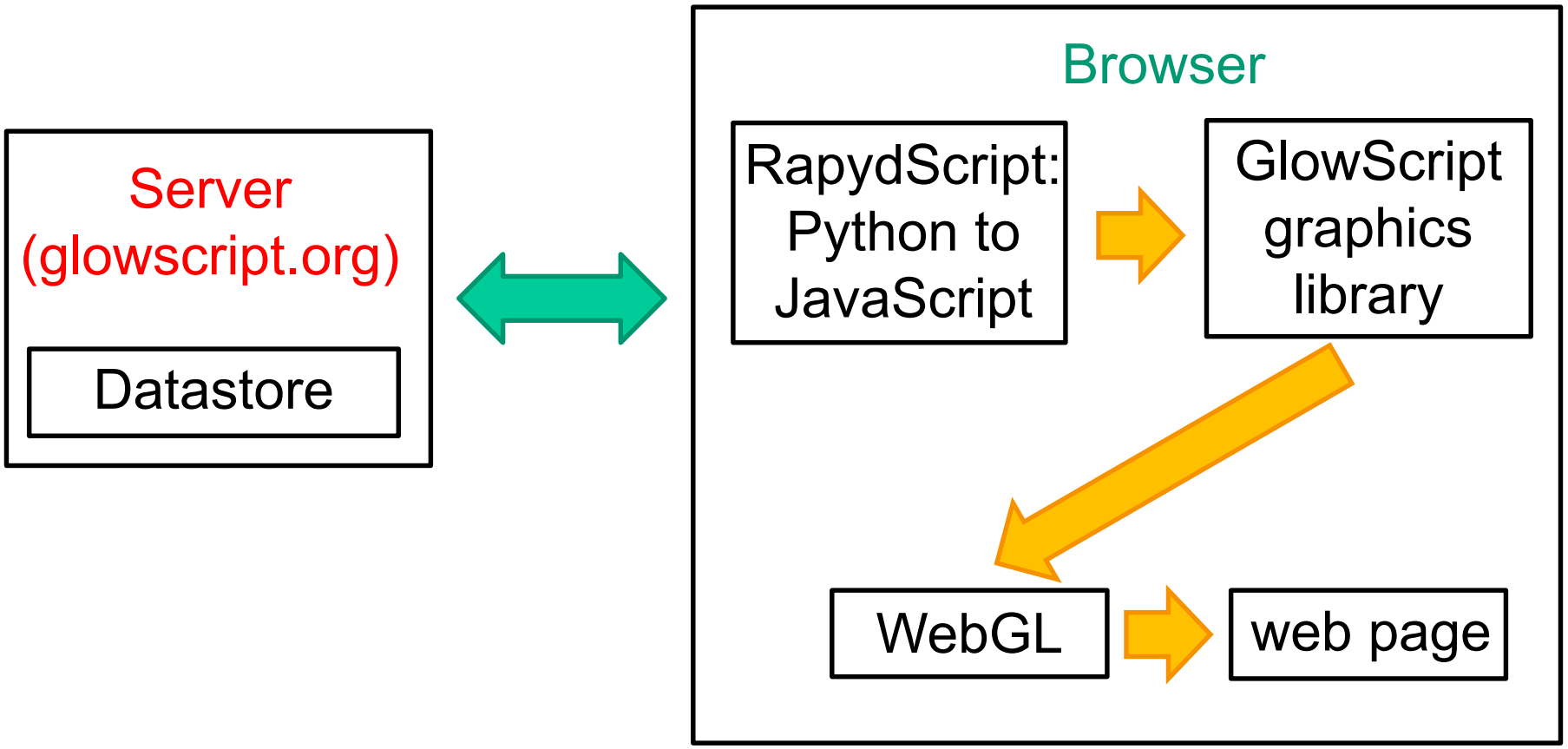
American Association of Physics Teachers, January 2020, Orlando Florida

Brief History of GlowScript

- 2000: Classic VPython created by David Scherer, an undergraduate student at Carnegie Mellon University, in collaboration with Ruth Chabay and me; later major contributions by Jonathan Brandmeyer, David Scherer, Steve Spicklemire, and me
- 2011 GlowScript JavaScript begun by Scherer and me
- 2014 GlowScript VPython by me
- 2014 Jupyter VPython begun by John Coady
- 2016 VPython 7: Jupyter VPython made consistent with GlowScript VPython by Chabay and me, in collaboration with Coady; Classic VPython no longer supported
- 2017: Matt Craig builds installers for VPython 7 and later carries out an important restructuring of the VPython 7 code

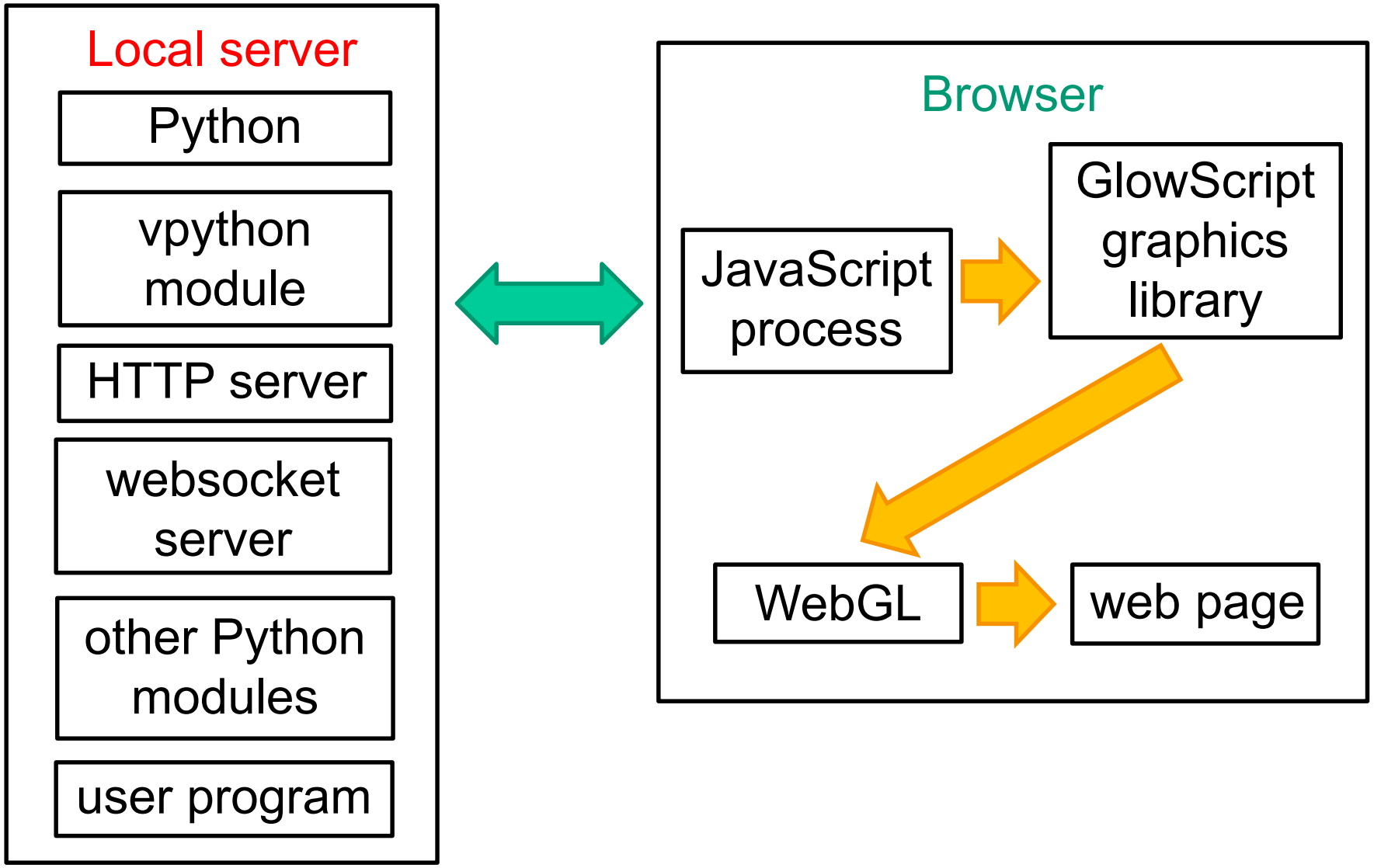
See **brucesherwood.net** for a detailed history

GlowScript VPython

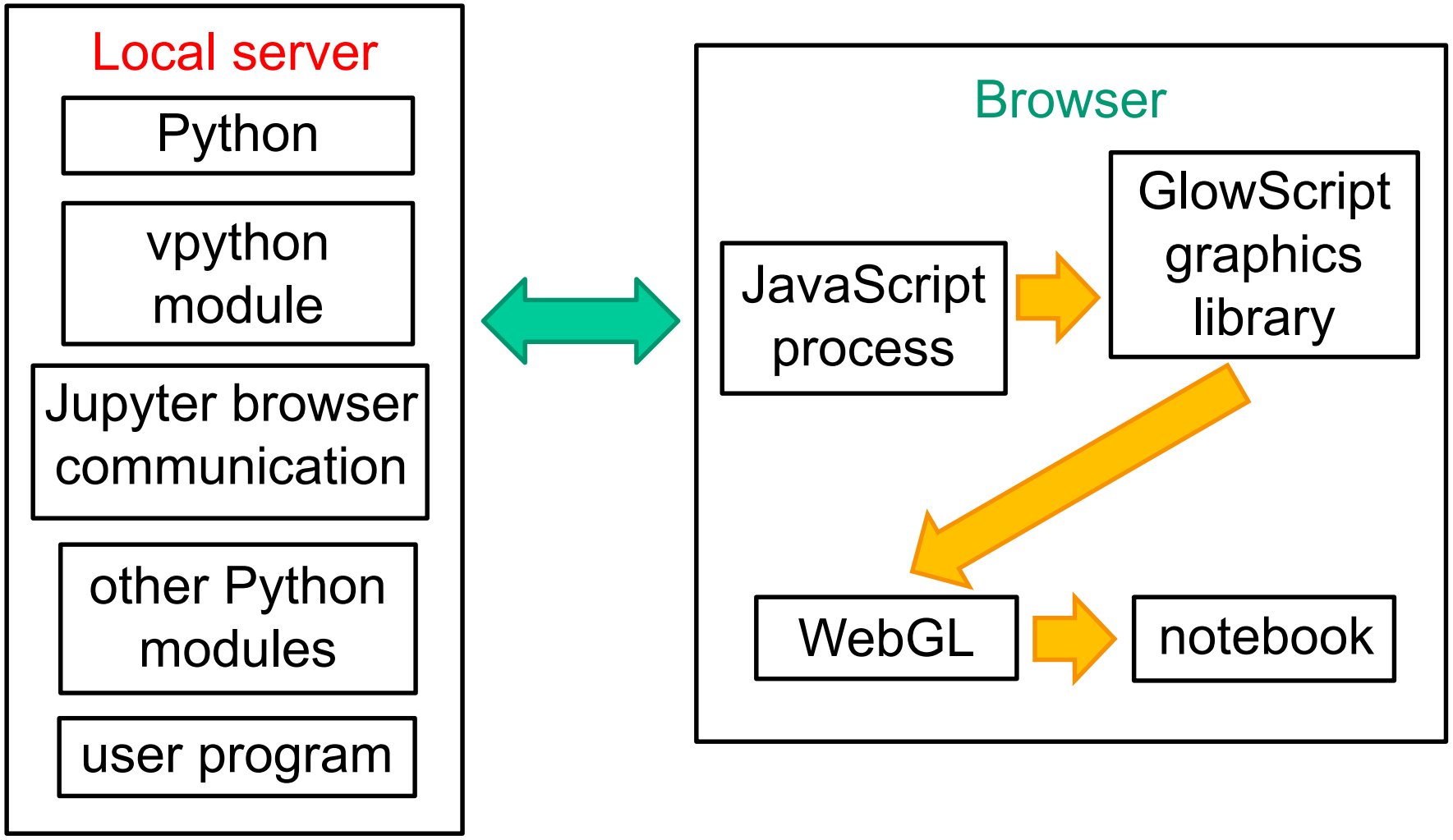


This architecture is also used by trinket.io

VPython 7


Local server
Python
vpython module
HTTP server
websocket server
other Python modules
user program
Browser
JavaScript process
GlowScript graphics library
WebGL
web page

VPython 7 with Jupyter Notebook



Compilation (GScompiler.js)

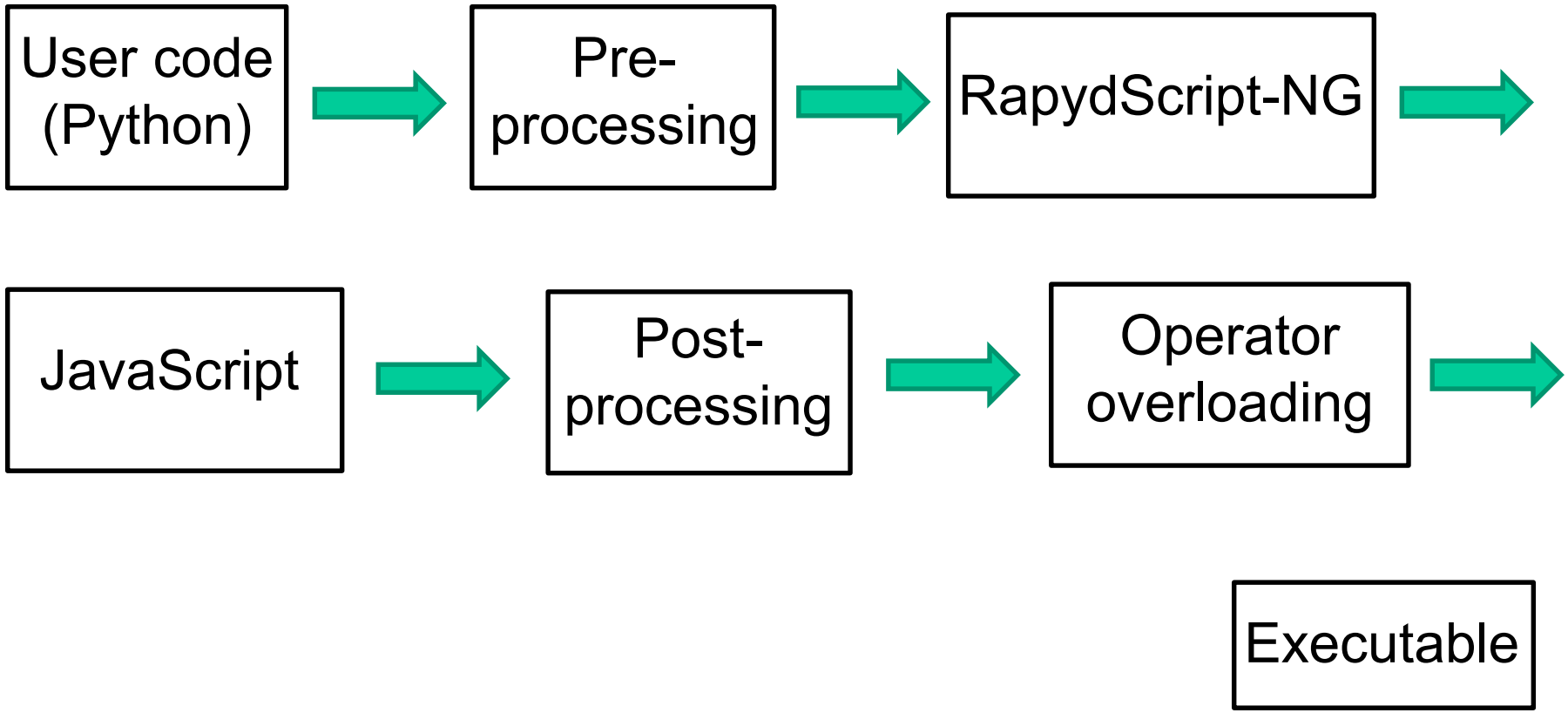
Python → JavaScript

```
1  GlowScript 2.9 VPython
2  a = 5*3
3  b = box(pos=vec(1,0,0), color=color.red)
4  b.pos = b.pos + vec(0,2,0)
5
6  def f(x):
7      print('Click to continue, x =', x)
8      scene.pause()
9   f(35)
```

```
"3";
a = 5["*"](3);
"4";
b = RS_interpolate_kwargs.call(this, box,
        [RS_desugar_kwargs({
         pos: vec(1, 0, 0), color: color.red})]);
"5";
b.pos = b.pos["+"](vec(0, 2, 0));
"7";
async function f(x) {
   "8";
   print("Click to continue, x =", x);
   "9";
   ;(await scene.pause());
};
if (!f.__argnames__)
  Object.defineProperties(f, {
     __argnames__ : {value: ["x"]}
});

"10";
;(await f(35));
```

Compilation Steps



Preprocessing

- Check for missing or extra parens, brackets, braces
- Insert Python source line numbers
- Handle import statements
- Locate functions and function calls
- Pass code to RapydScript-NG

Postprocessing

- Insert preamble to JavaScript code
- Redefine objects: box = vp_box
- Insert async and await where needed
- Miscellaneous small adjustments

Operator Overloading

- Using the PaperScript library with the Acorn parser of JavaScript, convert

  a + b => a[‘+’](b)

- JavaScript permits changing the behavior even of built-in classes such as Number and String (see vector.js)

- String.prototype['+'] = function(r) { return this + r }

- Number.prototype['+'] = function(r) {
  return (r instanceof vec) ? add_error() : this + r }

- vec.prototype['+'] = function(v) {
  if (!(v instanceof vec)) add_error()
  return new vec(this.x + v.x, this.y + v.y, this.z + v.z) }

Constructing vp_box()

- lib/glow/primitives.js handles creation of 3D objects
- function vp_box(args)
  { return initObject(this, vp_box, args) }
  subclass(vp_box, box)": inherit from JavaScript box;
  see function subclass(sub,base)
- box inherits from function Primitives(), which includes pos, color, visible, rotate(), etc.
- vp_box adds width, height, etc.
- See lib/glow/property.js for definition of mechanisms used to simplify attribute handling

Special Attribute Vectors

- obj.pos is a special “attributeVectorPos” vector
- When you change obj.pos (or obj.pos.x), code in vectors.js calls obj.__change() in primitives.js, marking the object as having changed
- For Python programs (not JavaScript programs), changing axis or size for most objects affects size or axis
- Also, change of axis or up calls adjust_axis() in vectors.js to make sure axis and up always remain perpendicular to each other

Updating Render Data

- For an object whose “changed” flag is set and is visible, at render time its current attributes are repackaged from 64 bits to 32 bits by the render function calling obj.__update()
- Five 4-vectors: pos, axis, size, color, up, with texture, opacity, shininess, emissive packed into 4th slots; total of 80 bytes per object instance

obj.opacity vs. obj.__opacity

Many attributes and methods used internally have "__" prepended to the name, to emphasize that user programs don't have direct access.

The basic opacity data is held in obj.__opacity. When referring to obj.opacity there are "getters" and "setters" to read and write the basic data and set the changed flag.

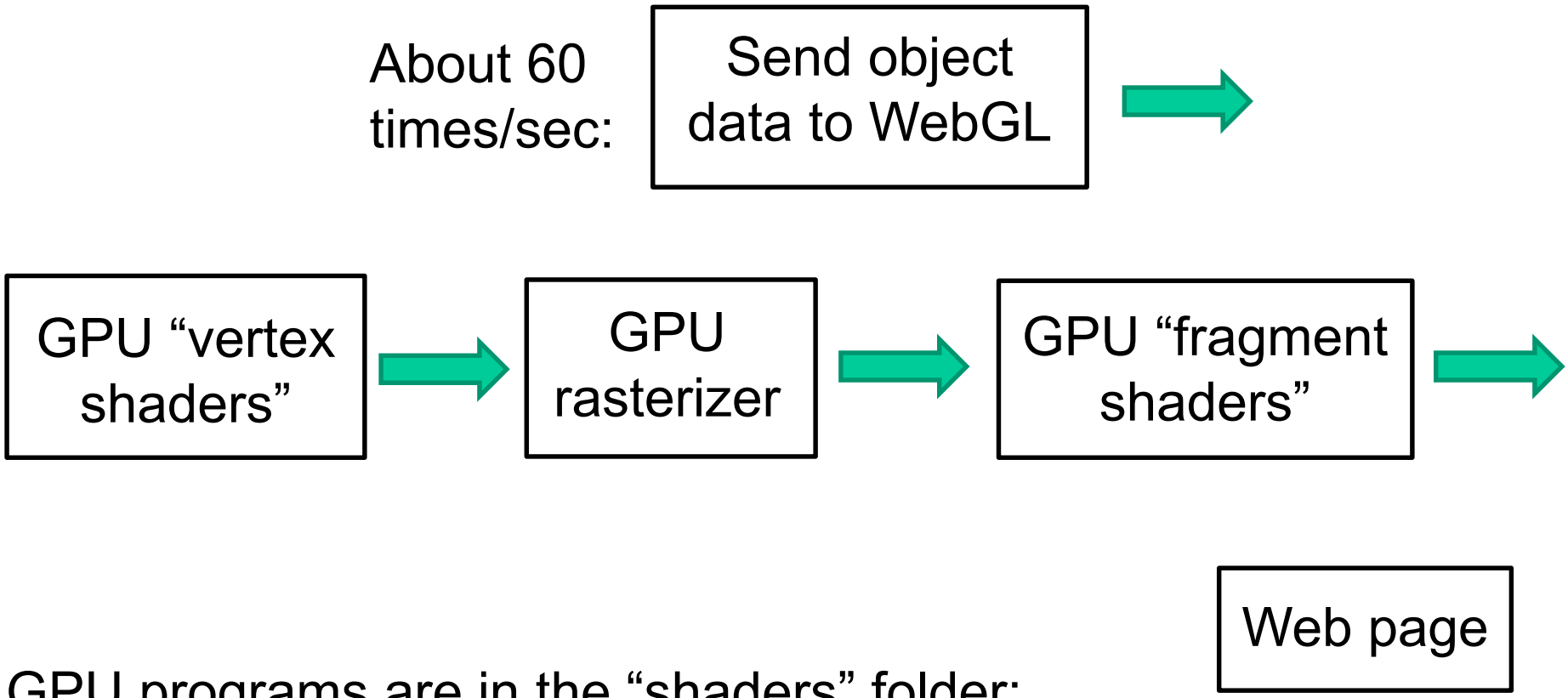
This pattern is widespread throughout GlowScript.

References within the GlowScript Library are often to obj.__opacity when it is known that it is not necessary to trigger the full change/update machinery.

Animations

- rate(200): no more than 200 loop iterations/s
- rate also calls a WebGL renderer about 60 time/s
- Sleep between loop iterations and renders
- rate() is in the file lib/glow/WebGLRenderer.js

GlowScript Rendering of 3D images



GPU programs are in the "shaders" folder; written in GLSL language

Object Models in GPU Memory

- A “model” 1x1x1 box object is stored in GPU memory; model definitions are in mesh.js
- Represented by 12 triangles, each described by 3 vertex objects specifying position, normal, color, and texture coordinates
- Data for a particular box (e.g. pos, color, etc. for an instance of the box class) plus model information is sufficient for the GPU to display that box appropriately in 3D
- GPU memory has models of a box, sphere, cylinder, cone, and pyramid; compounds have the status of primitives
- curve() is a sequence of cylinders capped by hemispheres
- Arbitrary objects built from triangles; extrusions, 3D text

Additional Technical Details

- Portions of objects hidden behind other objects are not seen thanks to “z-depth” blocking by GPU hardware
- Transparency handled by “depth peeling” algorithm
- Mouse “picking” uses false colors
- For transparency and picking details, see glowscript.org/docs/GlowScriptDocs/technical.html

Canvas

- lib/glow/canvas.js contains the 3D canvas-related attributes and methods, including mouse and keyboard handling; also sets up default lighting
- The 3D canvas is created by and used by WebGL
- A transparent 2D canvas for label() objects is in front of the 3D canvas
- lib/glow/autoscale.js positions the camera to include entire scene
- Low-level mouse handling is in orbitalcamera.js
- A graph (lib/glow/graph.js) is a 2D canvas

Web Issues

- ide/api.py is the Python server code including datastore management; ide/index.html and ide/ide.js produce the web page displays
- The ACE text editor is in lib/ace/ace.js; lib/editor.js is invoked on mobile devices where ACE doesn't work
- The files in the untrusted folder comprise a safe "sandbox" environment for executing the user program

JavaScript Used in GlowScript

- Semicolons are essentially optional in JavaScript, and to reduce clutter they are not used in the GlowScript code
- 2011-vintage JavaScript syntax has not been updated, with one important exception: the use of “await” and “async”
- Some open-source libraries used by GlowScript were modified to remove the use of “require” or for other reasons; this is documented at the start of the file

Speed Issues

- Python is an interpreted language and so execution is significantly slower than compiled languages.
- Computationally intensive GlowScript VPython programs run several times faster than VPython 7 programs, because they are compiled to (fast) JavaScript, but there is no access to Python modules

Python 2 -> Python 3

- As of January 2020, support by the Python community for Python 2 has ended. Because some of its own software uses Python 2, Google intends to maintain Python 2 itself. However, we are strongly advised to update to Python 3, which will require changes to the glowscript.org Python server code's access to the datastore (ide/api.py).

- Useful advice on the 2 -> 3 conversion: https://gaedevs.com/blog/things-to-understand-before-migrating-your-python-2-gae-app-to-python-3

For More Information

- **brucesherwood.net** – "A time line for VPython development"
- **github.com/vpython/glowscript** – GlowScript repository; see the README, docs/GlowScriptOverview.txt, and docs/MakingNewVersion.txt
- **glowscript.org/docs/GlowScriptDocs/local.html** – installing Google App Engine software
- **vpython.org** – obtaining and using VPython
- **glowscript.org** – full VPython documentation, many examples
- **trinket.io** – browser-based notebook-like use of VPython; choose "GlowScript", then choose "Python"
- **matterandinteractions.org** – calculus-based contemporary intro physics curriculum in which VPython plays an important role
- **matterandinteractions.org/student** – includes 70 physics demo programs written in GlowScript VPython

Participants in the workshop had installed the Google App Engine software and could modify the GlowScript code and try out the modifications (specifying GlowScript X.Ydev VPython). Tasks 1 through 3 hint at a possible addition to GlowScript but one should consider a more general approach in which one could attach a list of any kinds of objects, each with its own offset from the position of the moving object.

Task #1: Make a label Move with an Object

- In lib/glow/primitives.js/Primitive() add "label: null,“ and in __update(), after setting pos, add
  if (this.label !== null) this.label.pos = pos

- Get a copy of the binary star demo program code by going to glowscript.org, clicking “Example programs” and viewing the program “BinaryStar-VPython”

- Add labels to the program with yoffset=30; for example sphere(…. label=label(text=‘Giant’, yoffset=30))

Task #2: Check for label Object

- Insert the at the start of primitives.js/init():

```
if (args.label !== undefined && args.label !== null &&
  !(args.label instanceof label))
    throw new Error('obj.label must be a label object')
```

- Test this with "sphere(…. label=5)"

Task #3a: 3D label Offset

- Modify Primitives __update() like this:

- ```
  if (this.label !== null) {
     if (this.label.offset !== undefined) {
          var x = axis.hat
          var y = up.hat
          var z = cross(x,y)
          var o = this.label.offset // a 3D offset
          // The current position of the offset location:
          this.label.pos = pos + o.x*x + o.y*y + o.z*z
     } else this.label.pos = pos
  }
  ```

- Before proceeding, what's wrong with this code?
  ```

The problem is that the internal JavaScript code is not processed by the operator overloading machinery, so one has to write the vector calculation as shown below.

Task #3b: 3D label Offset

- Modify Primitives __update() like this:

- ```
  if (this.label !== null) {
     if (this.label.offset !== undefined) {
          var x = axis.hat
          var y = up.hat
          var z = cross(x,y)
          var o = this.label.offset
          this.label.pos = pos.add(x.multiply(o.x).add(
              y.multiply(o.y).add(z.multiply(o.z))) )
     } else this.label.pos = pos
  }
  ```

- Change the binary star program labels:
  label=label(text='Giant',align='left',offset=vec(5e10, 0, 0))
  label=label(text='Dwarf',align='left',offset=vec(5e10, 0, 0))
  ```

Task #3c: 3D label Offset

- In function label(args), after the init() statement, add this:

```
if (this.offset !== undefined &&
   !(this.offset instanceof vec))
      throw new Error('label "offset" must be a vector’)
```

Task #4: Change cylinder Mesh

- In lib/glow/mesh.js/makeCylinder() change N from 50 to 5 and see how a cylinder display is affected (then reset to 50)

Task #5: An Open-ended Box

- Create open_box (inherit from box) and vp_open_box (inherit from vp_box)
- Add references to end of primitives.js.
- Define makeOpenCube in mesh.js. Make a copy of makeCube, and comment out the first 2 lines of each element, except for index.push (comment out 16 through 23)
- In WebGLRender.js search for "new Model" and add
  var mopenbox = new Model(Mesh.makeOpenCube(), false)
- A few lines later, insert
  object_models.vp_open_box = mopenbox
  and a bit later, insert
  object_models.open_box = mopenbox
- In compiling/GScompiler.js, search for "var vp_primitives" and add open_box to the list

Task #6: Miscellany

- In canvas.js change the default lighting
- Make a visible change in the initial page display at lib/glow/ide/index.html
- If you finish early, invent a task of interest to you!
- Note: The VPython user documentation was prepared using Adobe Dreamweaver