Exploring Second Life for collaborative visualisation.

Paul Bourke
WASP, UWA

Contents:

- Introduction
- Requirements
- Examples
- Summary

http://local.wasp.uwa.edu.au/~pbourke/papers/cgat08/
http://local.wasp.uwa.edu.au/~pbourke/papers/cg2008/

Introduction

- Visualisation: Use of computer graphics to assist researchers learn more about the relationships/structure of their data, and to do so more efficiently.
- Distinction between illustrative visualisation and data visualisation in the sciences.
  - Illustrative visualisation is generally used to convey science principles to a general audience.
  - Data visualisation is used directly by researchers with data from observation or simulation.
- Often performed collaboratively in specialised environments: stereoscopic displays, high resolution tiled displays, virtual reality and other immersive environments.
- Very rarely supports remote collaborative exploration/engagement despite the increasingly distributed nature of many scientific projects.
- Traditional remote collaborative tools (chat, video conferencing, AccessGrid) don't generally support more than 2D graphics (images, white board, movies). Sharing data experiences generally relegated to 2D representations on web pages or via email.
- To what extend can Second Life be use as a platform for collaborative scientific visualisation? Emphasis on 3D datasets represented and engaged with within a 3D virtual environment.

Requirements

- Must be possible to import data from which visual geometric elements are created. [Building by hand is generally only suited to illustrative visualisation exercises]
- Need a rich set of geometric primitives with which to represent the data.
- Large volumes of data are often required.
- Ability to create appealing graphics, to map colour, transparency, texture onto the geometry ... these are used in scientific visualisation to represent variables above the 3 available geometric dimensions.
- The 3D interface needs to be intuitive, provide a natural engagement with the virtual environment.
- It must be supported on a wide range of hardware platforms (MSWindows, Linux, Mac OS-X). Due to the UNIX basis of almost all high performance computing, increasingly the platform of choice in the sciences is Mac OS-X.
- Remote collaborative visualisation requires traditional modes of communication: text and voice.

Example: Data import

Representation of JAYFEV molecule (left) and Aspirin (right).

Example: Data import

- Data imported via “list {}” data structure.
- Requires a manual copy/paste. Alternatively use http get from web server to import data.
- Implement Second Life list export as a standard export format from data processing software.

Atom rez script

```
default {
  on_rez(integer n) {
    if (n == 32) {
      llSetColor(<0.8,.2,.2>,ALL_SIDES);
    } else if (n == 32768) {
      llSetColor(<.2,.8,.2>,ALL_SIDES);
    } else if (n == 33792) {
      llSetColor(<.2,.2,.8>,ALL_SIDES);
    } else  {
      llSetColor(<1,1,1>,ALL_SIDES);
    }
    llSetScale( <.5,.5,.5> );
  }
}
```

Bond rez script

```
default
{
  on_rez(integer n) {
    float fn = (float)n / 100.0;
    llSetScale( <0.4,0.4,fn> );
  }
}
```

```
list spherelist = [
 33792,
 <0.154,-0.015,0.7085>,
 32768,
 <1.184,-0.156,1.0125>,
  ...
];

list cylinderlist = [
 33824,
 <0.1437,-0.0136,0.70546>,
 <0.679506,-0.086428,0.863601>,
 32768,
 <1.1943,-0.15742,1.01554>,
 <0.658494,-0.0835516,0.857399>,
  ...
];

default {
  touch_start(integer total_number) {
    integer i;
    float scale = 1;
    vector offset = <0,1,3>;
    vector currentp = llGetPos() + offset;

    // Draw the balls
    integer ns = llGetListLength(spherelist);
    for (i=0;i<ns;i+=2) {
      integer id = llList2Integer(spherelist,i);
      vector p = llList2Vector(spherelist,i+1);
      llRezObject("Atom2",
          currentp + p, ZERO_VECTOR, ZERO_ROTATION, id);
    }

    // Draw the cylinders
    integer nc = llGetListLength(cylinderlist);
    for (i=0;i<nc;i+=3) {
      integer id = llList2Integer(spherelist,i);
      vector p1 = llList2Vector(cylinderlist,i+1);
      vector p2 = llList2Vector(cylinderlist,i+2);
      rotation rot = llRotBetween(<0,0,1>,p2-p1);
      integer len = (integer)(100*llVecMag(p2-p1));
      llRezObject("Bond2",
          currentp + (p2+p1)/2, ZERO_VECTOR, rot, len);
    }
  }
}
```

Example: Data import

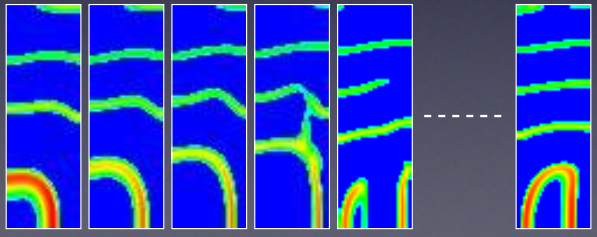
Apollony fractal

Example: Volumetric data

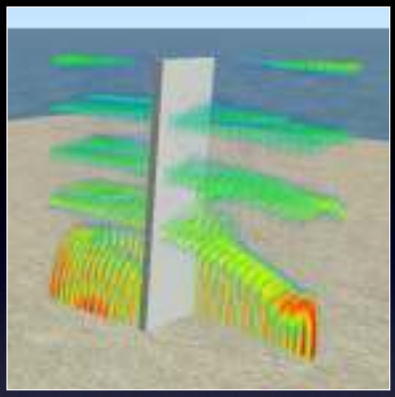
Volume visualisation of helix wave formation in fluid flow

Example: Volumetric data

- Volume rendering by stacking up large numbers of textured boxes.
- Textures are automatically applied to rez'ed planes.
- Voxel values are mapped to colour and transparency to reveal internal data structure.
- Works best from a limited range of viewing angles.



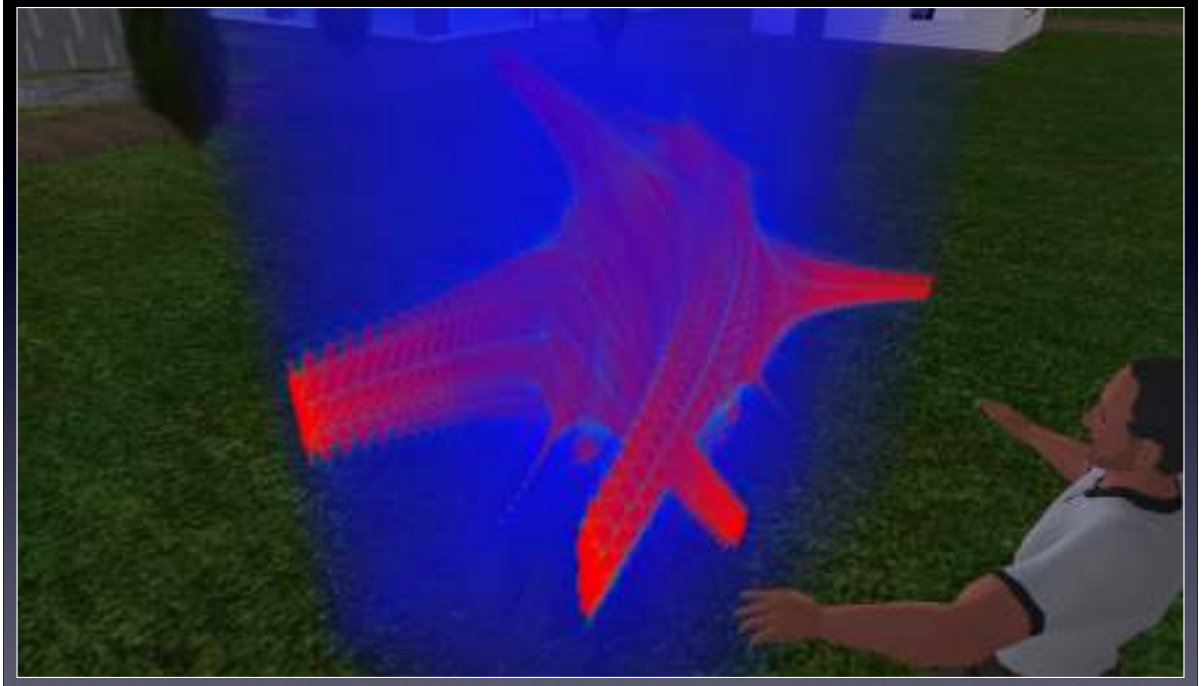
Blue = transparent



```
string object;
string thetexture;
integer textureid = 0;

default {
  touch_start(integer total_number) {
    integer i=0;
    vector p;
    vector offset = <0,2,2.5>;
    float depth = 0.05;
    for (i=0;i<=31;i++) {
      p = llGetPos() + offset + <depth*i,0,0>;
      object=llGetInventoryName(INVENTORY_OBJECT,0);
      llRezObject(object, p, ZERO_VECTOR, ZERO_ROTATION, i);
    }
  }
  object_rez(key id) {
     integer i;
     llOwnerSay("Creator received rez");
     if (textureid < 10)
        lGiveInventory(id,"layer0" + (string)textureid);
     else
        llGiveInventory(id,"layer" + (string)textureid);
     textureid++;
  }
}
```

Example: Volumetric data



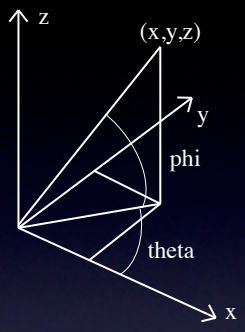
Volume visualisation of 4D fractal, $z_n = (z_n^2 + c_1)^2 + c_2$

Example: Surfaces

Representation of solutions to the Laplace equations in spherical coordinates.

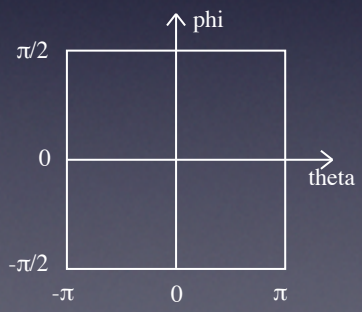
Example: Surfaces

- No support for a general surface mesh!
- Closest that can be used in some circumstances are "sculpt maps".
- Colour of each pixel in a small texture is interpreted as the radial length in polar coordinates.
  (r,g,b) = (x,y,z)
- The sculpt map defines the geometry, additional colour maps or textures can still be applied.

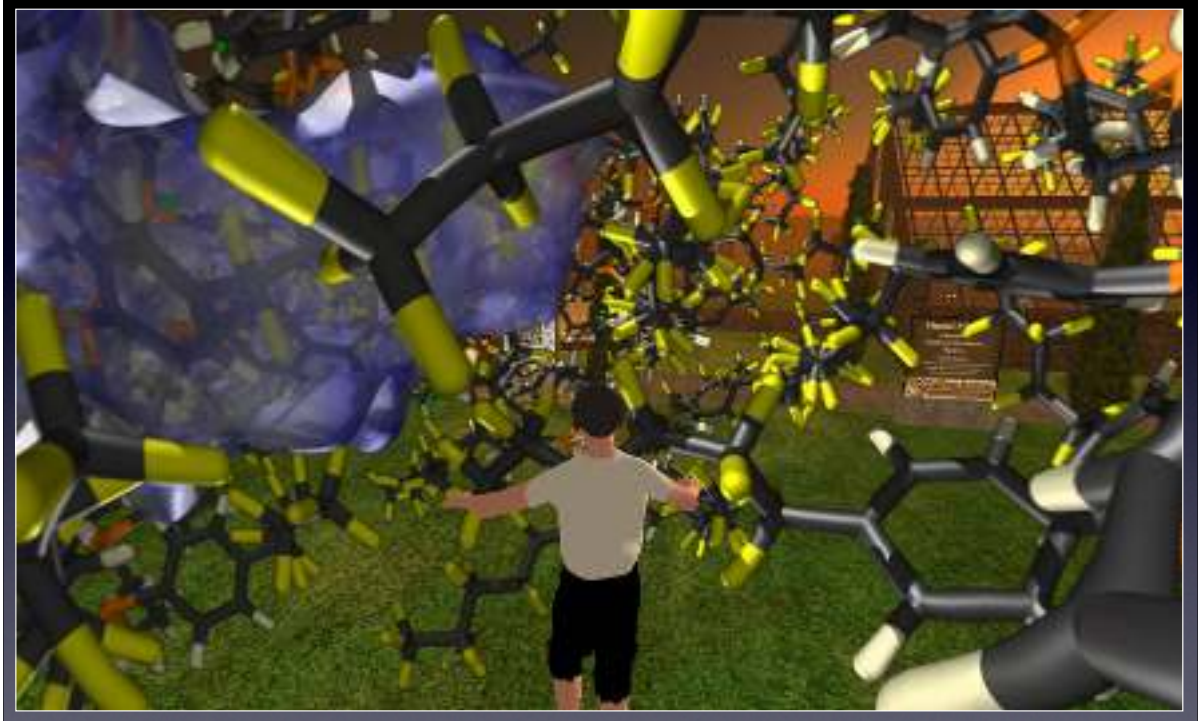


Corresponding surface

Sculpt map image



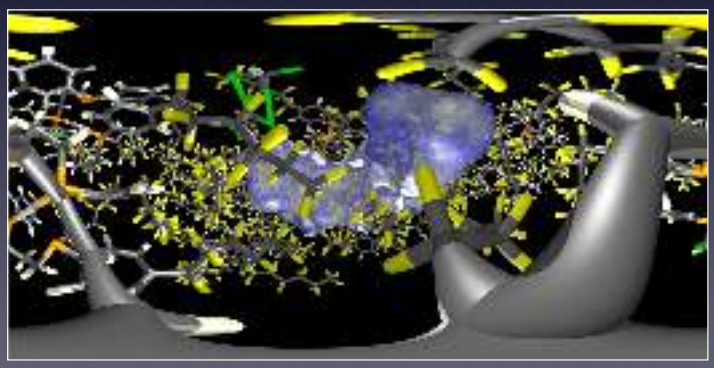
Example: Texturing tricks



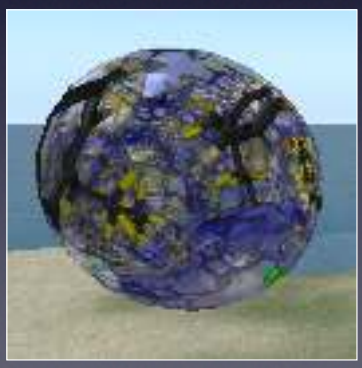
Preprocessed spherical projections from inside a crystal including the Hershfield surface, computed from Crystal Explorer

Example: Texturing tricks

- Many datasets are simply too large to even contemplate explicit representation in Second Life.
- One approach is to prerender to spherical projections and present the result in Second Life as a texture mapped onto a sphere.
- Texture size limitations often mean multiple texture tiles are required.
- When the avatar is located at the center of the sphere a surprisingly convincing sense of 3D can be achieved.



Spherical projection



External view

Scripted construction within SL

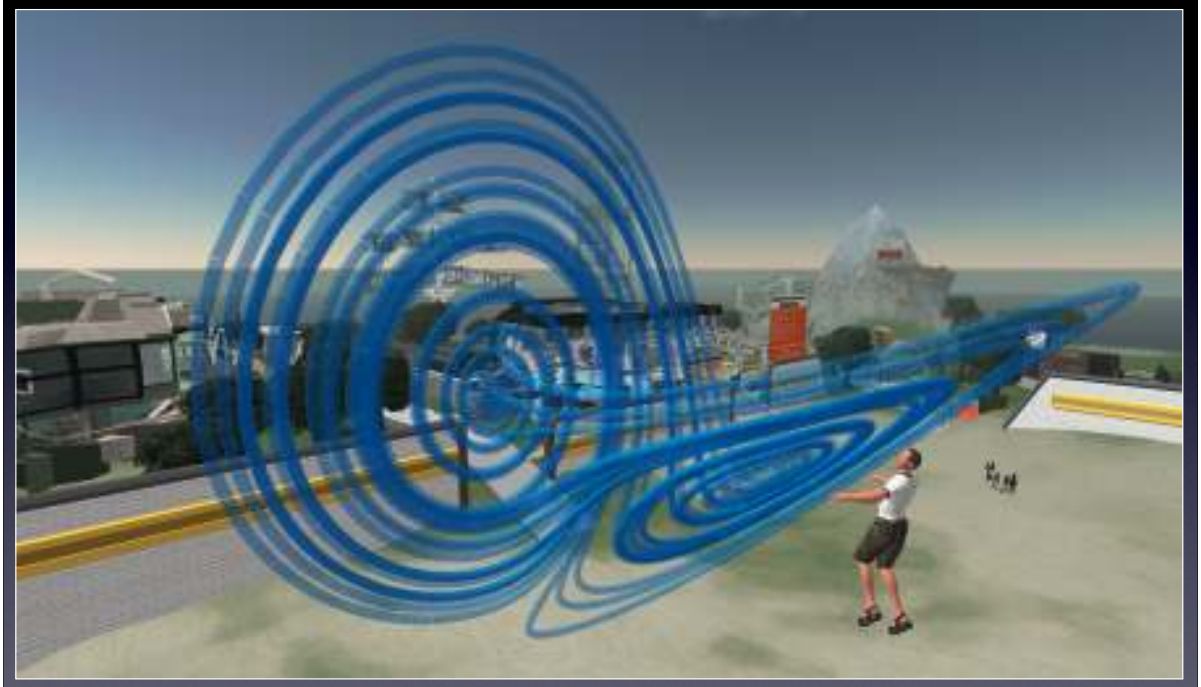
- Linden script supports standard programming constructs/logic ... very similar in style to JavaScript.
- Each primitive created requires communication with the remote server. This latency generally precludes any animation except at relatively slow rates (~1 or 2 fps).
- Geometric recursion supported by primitive groupings.

```
default {
    touch_start(integer total_number) {
        integer i,j,k;
        vector p;
        vector offset = <5,0,5>;
        float size = 1.5;
        for (k=-1;k<=1;k++) {
            for (i=-1;i<=1;i++) {
                for (j=-1;j<=1;j++) {
                    if (k == 0) {
                        if (i != 0 && j != 0) {
                            p = llGetPos() + offset + <-size*i,-size*j,k*size>;
                            llRezObject("Menger1", p, ZERO_VECTOR, ZERO_ROTATION, 1);
                        }
                    } else if (i != 0 || j != 0) {
                        p = llGetPos() + offset + <-size*i,-size*j,k*size>;
                        llRezObject("Menger1", p, ZERO_VECTOR, ZERO_ROTATION, 1);
                    }
}}}}}
```

Example: Scripted internal construction

Menger Sponge: 4 geometric iterations, 3 textured iterations

Example: Scripted internal construction



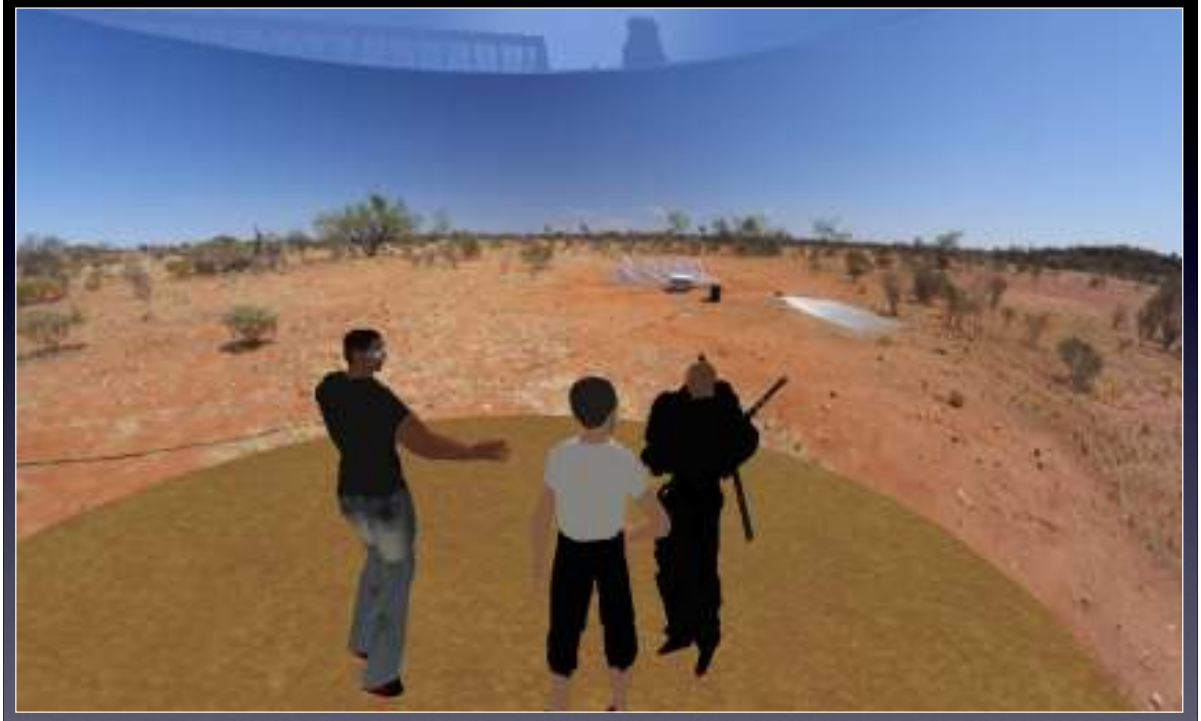
Lorenz Attractor: the slow construction can be informative.

Summary

- Limitations (Roughly in order of decreasing importance)
  - Lack of a surface mesh primitive.
    Precludes the representation of many datasets.
  - Limits on the data in a list{} or amount of data downloaded with a http get.
  - Limits on the number of geometric primitives per area of land.
    An limit for some datasets but reasonable given the realities SL needs to cope with.
    There are ways around these limitations, however inconvenient.
  - Time fidelity is too coarse for most time varying visualisation.
    Fact of life given the client - server data transfer latency.
  - Lack of support for stereoscopic and immersive displays.
    Not unexpected.
  - 1K limit on textures.
    More of an inconvenience as there are ways around this limit.
  - Lack of photographic avatar appearance.

- Strengths
  - Collaborative aspects are perhaps unrivaled (or even supported)
    by any other visualisation software.
  - Intuitive easy to learn/use 3D environment.
  - Cross platform support, "relatively" stable software, and free.
  - Ability to create high quality visuals.
  - Exploits graphics card capabilities.

Questions?



Meetings at the ASKAP site