Test Plan
S2PLOT extension of Omegalib

G.T. Software

April 2014

Signature

The following signature indicates approval of the enclosed Software Test Plan.

Change history

Version	**Date**	**Author**	**Changes**
0.1	10/8/2014	Bhavya Malik	Initial revision

Contents

List of Figures

List of Tables

1 Introduction

The pupose of this document is to layout the approach that we have used to regression test our code base and enlist all the test items and features that would be tested in the S2Plot module in Omegalib, it also talks about the tools we used. This Plan also aims at providing a concise summary of the content and scope of the testing that is involved in the port.
Due to the high complexity and difficulty in testing in a project like this we are using a custom regression testing technique. As the output of two different libraries cannot be tested via a pixel by pixel comparison, hence we are generating our own reference data to test and compare with. Regression tests will be conducted by using a third party image comparison software that will layout the differences in the old and new screenshots.

2 References

2.1 Software Requirements Specification (SRS)

Version	1
Date	18 April 2014
Author	B. Malik, A. Shifaz, A. Hubble, A. Limberopoulos
Access information	https://github.com/limbera/s2plot-omegalib/tree/master/documentation/srs

2.2 Software Quality Assurance Plan (SQAP)

Version	1
Date	18 April 2014
Author	B. Malik, A. Shifaz, A. Hubble, A. Limberopoulos
Access information	Will be provided on the github repository at completion.

2.3 IEEE Standard 1058-1998

Version	1
Date	8 December 1998
Author	The Institute of Electrical and Electronics Engineers, Inc.
Access information	http://shop.ieee.org/store/product.asp?prodno=SS94690

2.4 Meeting Minutes template

Version	1
Date	18 April 2014
Author	B. Malik, A. Shifaz, A. Hubble, A. Limberopoulos
Access information	https://github.com/limbera/s2plot-omegalib/tree/master/documentation/meetings/tex

2.5 Meeting Agenda template

Version	1
Date	18 April 2014
Author	B. Malik, A. Shifaz, A. Hubble, A. Limberopoulos
Access information	https://github.com/limbera/s2plot-omegalib/tree/master/documentation/meetings/tex

2.6 Code Review Checklist

Version	1
Date	9 August 2014
Author	B. Malik, A. Shifaz, A. Hubble, A. Limberopoulos
Access information	https://github.com/limbera/s2plot-omegalib/tree/master/documentation/standard_docs

2.7 Design Review Checklist

Version	1
Date	9 August 2014
Author	B. Malik, A. Shifaz, A. Hubble, A. Limberopoulos
Access information	https://github.com/limbera/s2plot-omegalib/tree/master/documentation/standard_docs

2.8 Test Checklist

Version	1
Date	9 August 2014
Author	B. Malik, A. Shifaz, A. Hubble, A. Limberopoulos
Access information	https://github.com/limbera/s2plot-omegalib/tree/master/documentation/standard_docs

2.9 Test Defect Log

Version	1
Date	9 August 2014
Author	B. Malik, A. Shifaz, A. Hubble, A. Limberopoulos
Access information	https://github.com/limbera/s2plot-omegalib/tree/master/documentation

2.10 Product Backlog

Version	1
Date	9 August 2014
Author	B. Malik, A. Shifaz, A. Hubble, A. Limberopoulos
Access information	https://github.com/limbera/s2plot-omegalib/tree/master/documentation/sprints

2.11 Sprint backlogs, codereviews and risks

Version	1
Date	9 August 2014
Author	B. Malik, A. Shifaz, A. Hubble, A. Limberopoulos
Access information	https://github.com/limbera/s2plot-omegalib/tree/master/documentation/sprints

2.12 An Advanced, Three-Dimensional Plotting Library for Astronomy

Version	1
Date	22 June 2006
Author	David G. Barnes, Christopher J. Fluke, Paul D. Bourke and Owen T. Parry
Access information	`http://www.publish.csiro.au/?act=view_file&file_id=AS06009.pdf`

2.13 CAVE2: A Hybrid Reality Environment for Immersive Simulation and Information Analysis

Version	1
Date	23 March 2014
Author	Alessandro Febrettia, Arthur Nishimotoa, Terrance Thigpena et. al.
Access information	`http://www.evl.uic.edu/files/pdf/SPIE13Paper-final.pdf`

3 Software Risk Issues

This section will discuss the risks and issues involved with the software being used.

Table 1: Table of risks

Risk Item	**I**	**P**	**Contingency**
Inexperience with C++ and OpenGL programming	H	H	Reading online docs for C++ and OpenGL programming, read the FIT3080 Graphics lecture slides
Inexperience with S2PLOT and Omegalib	H	H	Explore examples and understand how the two libraries are coded.
Both S2Plot and Omegalib are poorly documented.	H	H	Try to obtain proper documents from owners or understand and roughly document to proceed further with development.
We are not able to configure an IDE to work in a Linux environment.	H	H	We are using a basic IDE called Geany and CMake.
Inconsistencies between coordinate system of S2PLOT and Omegalib	H	M	We have discussed with David Barnes and there are two feasible approaches. We can either do a global translate at the beginning of the application, or we can generate our own config file.

Table 1: Table of risks

Risk Item	I	P	Contingency
CAVE2 failure	H	L	Development and testing using CAVE2 development workstations in New Horizons building, until CAVE2 is returned to operational status.
Github failure	H	L	Secure backups of latest working version offline, taken after each sprint, kept on a physical drive
Omegalib gets updated frequently	H	L	Backup one version of Omegalib and use that only if need be to build again.

4 Testing Approach

4.1 Unit Testing

For unit testing we have used a Sandwich Testing Technique. To test our drivers we used a Module Top Down approach and to test the S2Plot Applications we used a Bottom Up Approach.

4.2 Integration Testing

4.2.1 Boundary Value Testing

The testing approach being used to make sure our geometry rendering functions follow those from the S2plot library is Boundary Value Testing. We developed test cases which involved calculating the Max, Min and Nominal values of every parameter in a function. These test cases would then help us in testing our functions (in Omegalib) with BVT, and compare these (Max,Min,Nom) values with that of S2Plot to ensure functionality was implemented correctly.

4.2.2 Visual Regression Testing

The testing approach involves taking screenshots at the end of every week to compare it with the screenshots taken in the earlier week to make sure that old functionality does not change as the new one is added. Additionally, we will be taking screenshots of the new functionality that is implemented every week to save that as reference data. Hence, every week we will take new screenshots to create reference data and compare them with older screenshots to make sure the implementation works as it should.
The approach we are using to visually regression test this is by taking screenshots of the geometry we implement from 14 different camera angles to make sure we cover each and every aspect. At the end of the next sprint we take screenshots at the same angles and compare them with the reference data by running them through a third party software which compares images pixel by pixel and produces another image that outlines the differences.
The comparison of the images and the details of the tests will be present on the Test Analysis document that will grow every week as new tests are added to it. The Test analysis document will have side by side comparisons of new and old images from all different angles. If there is a failure in any of the tests, there will be a detailed analysis provided as to why the test failed.
On the event of failures occurring, these will be recorded in the test analysis document and a detailed explanation as to how and why the failure occurred will be provided. The failure might or might not be fixed in the next sprint depending on the time constraints.

4.3 Acceptance Testing

Acceptance Testing involves running and displaying the S2Plot functionality implemented in Omegalib on the CAVE2 environment.
Another Non-functional requirement that we have involved in our acceptance testing criteria is to make sure that the S2Plot implementation in Omegalib is 50% faster than the original S2Plot library.

4.4 Testing Tools

4.4.1 ImageMagick

Imagemagick has a command line tool known as compare which takes in three arguments with the first two involving the images that need to be compared and the third one takes the name of the image which would produce the differences between the two images.
The differences in the newly produced image are highlighted by red pixels which are present on the area that is not the same in both the input images and the area covered in white pixels signifies the area that is the same. Hence it makes it easier to distinguish between images by showing the differences in red pixels and all the similarities in white. An image comparison with the image showing the differences is shown below.

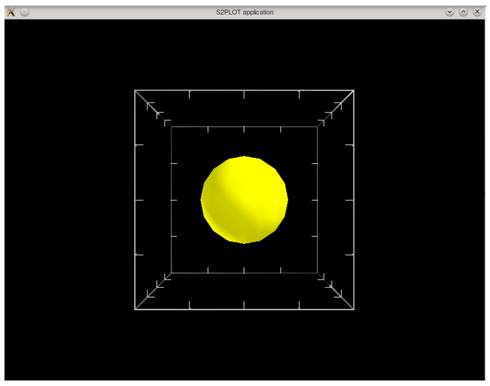

Figure 1: Non-textured sphere

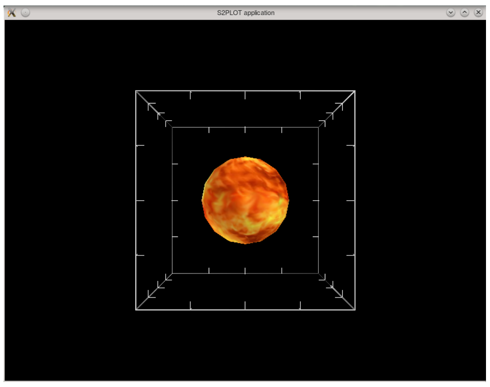

Figure 2: Textured Sphere

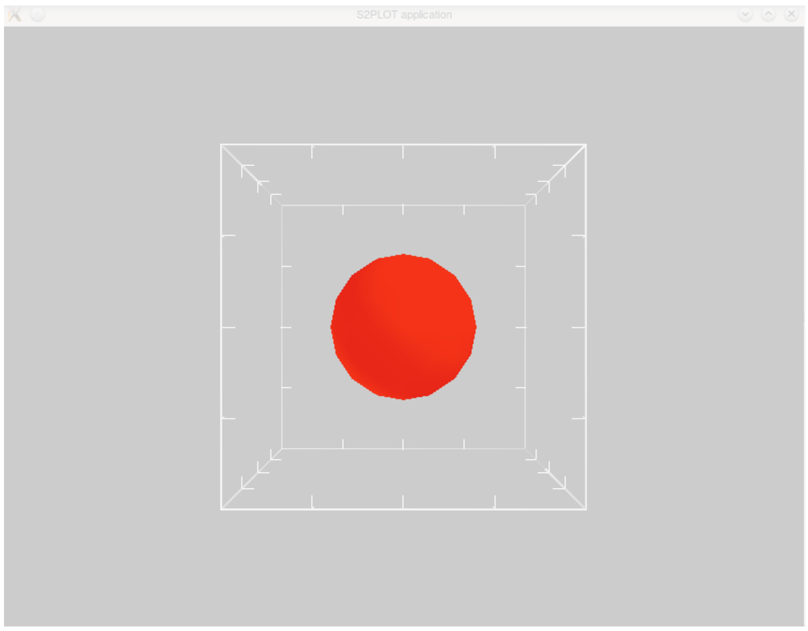

Figure 3: Difference between the spheres

The above images represent spheres with different textures, hence the difference image shows the sphere as red as the textures in that area in particular are different however the rest of the screen is white as the texture there is the same. However the shape and size of both the spheres is the same.
This command can be used with further arguments that would give a more detailed analysis of the images by showing a mathematical measure of the difference with the help of the following command:

```
Example -

compare -verbose -metric mae sphere1.jpg sphere2.jpg difference.png

Channel distortion: MAE
red: 2282.91 (0.034835)
green: 1853.99 (0.0282901)
blue: 2008.67 (0.0306503)
all: 1536.39 (0.0234439)
```

4.4.2 KSnapshot

Ksnapshot is a tool that we will be using to take screenshots of the images for comparison.

4.4.3 Geany

Geany is a text editor using the GTK2 toolkit with basic features of an integrated development environment. It was developed to provide a small and fast IDE, which has only a few dependencies from other packages.
Geany was used to code, edit and run the S2Plot library for testing purposes.

4.4.4 S2Plot library

The S2Plot library was the main tool with the help of which we developed our test cases. As we had to imitate (functions in S2Plot) inside the S2Plot module of Omegalib, thus we could directly perform BVT tests on the S2Plot library and then test our module with the help of these test cases to check how closely the implementation resembled the original API.

4.5 Measures and Metrics

The following information will be collected by the Development team during the testing process.

1. Defects by module and severity.
2. Defect Origin (Requirement, Design, Code)
3. Time spent on defect resolution by defect, for Critical & Major only. All Minor defects can be totaled together.

4.6 Meetings

The test team will have frequent meetings every week to evaluate progress to date and to identify error trends and problems as early as possible, Meetings would also involve fixing design issues and merging code from different developers and dividing tasks. Additional meetings can be called as required for emergency.
Regular meetings will be held with the Client as well to discuss feasible implementation stratergies and to display work done in previous sprints.
Regular meetings with the supervisor will be held as well to discuss current and previous issues and solutions.

5 Item Pass/Fail Criteria

The pass fail criteria for all the tests ran every week will be present in the Test Analysis document as the comparison of the images would decide whether the test passed or failed. A table inside the Test Analysis document will maintain what items passed and failed in the regression testing process ran at the end of every week. The table will act as an overview to see what items succeeded and which ones failed so as to know where and how the fixes need to be done.

6 Test Deliverables

This section will enlist all the Test related documents to be delivered.

- Test Plan
- Test Checklist
- Defect Log
- Test Analysis document

7 Staffing and Training Needs

The Test Lead assigned to the project will be responsible for all the testing related activities. The test lead will come up with test cases and criterion to test the project upon. No training will be provided for testing by G.T. Software and each team member is responsible for understanding how testing will be performed.

8 Responsibilities

The test lead will be responsible for testing all the screenshots and making sure if all the tests are working as they need to. He will need to write down a detailed analysis if the tests fail explaining as to why and where the tests were failing. This information will then be passed on to the implementation team to fix errors and bugs so that testing can resume.
The test lead will also be responsible for maintaining and updating all test related documents, namely the Test Plan and the Test Analysis document and also checking against the test checklist to make sure all test items mentioned there are checked off.

9 Approval

Approval will be obtained from the test lead at the end of every week when the screenshot comparison of the newly obtained screenshots matches with the saved reference screenshots. However if any of the comparison fails then the implementation team would have to reimplement and fix the errors so as to proceed with further testing.

10 Items to be Tested

10.1 Drawing a Circle

void s2circxy(float px, float py, float pz, float r, int nseg, float asp);

Draw a circle in the xy plane at a given z coordinate. Use nseg to control how many line segments are used to draw the circle. Low values of nseg (¿= 3) can be used to draw n-sided polygons. Argument asp controls the aspect ratio and whether a circle or ellipse is drawn. asp¡0 will produce a circle by calculating the radius, asp=1.0 will produce a circle.

Range of Values for Number of Line Segments

Max: 50 (Anything above 32 produces the same result)
Min: 20 (Anything below 20 would have edges)
Nom: 32

Range of Values for the radius

Max: No Max Value Set.
Min: 0.1
Nom: 5

10.2 Drawing a poly line

void s2line(int n, float *xpts, float *ypts, float *zpts);

Draw a poly line, n vertices at (xpts, ypts, zpts). The line is drawn in the current colour and with the current line width.

Range of values for n (Number of lines)

Max: 70 (Anything above 70 throws a render error)
Min: 2
Nom: 20

10.3 Drawing a wireframe Cube

void s2wcube(float xmin, float xmax, float ymin, float ymax, float zmin, float zmax);

Draw a wireframe cube with edges parallel to the main coordinate axes. Thickness and colour controlled by standard S2PLOT functions.

Set of values for the Cube co-ordinates

Set of max values: Does not have max values set
Set of Min values:
xmin = 0.7, xmax = +0.8; ymin = 0.7, ymax = +0.8; zmin = 0.7, zmax = +0.8;
Set of nom values:
xmin = -0.7, xmax = +0.8; ymin = -0.7, ymax = +0.8; zmin = -0.7, zmax = +0.8;

10.4 Drawing a set of points

void s2pt(int np, float *xpts, float *ypts, float *zpts, int symbol);

Draw a set of npts points. Symbol value 1 produces a single pixel, independent of distance to the point. Symbols are drawn in the current colour. Markers taking finite size are scaled by the current character height. The current linewidth does not affect markers. Argument symbol should be one of: 0 = wireframe box 1 = point 2 = wireframe 3D cross 4 = shaded sphere 6 = shaded box

Range of values for np (Number of Points)

Max: 100
Min: 1
Nom: 50

10.5 Drawing a coloured Sphere

void ns2sphere(float x, float y, float z, float r, float red, float green, float blue);

Draw a sphere, with a given centre (x,y,z), radius (r) and RGB colour (red, green, blue).

Set of values for r (Radius)

Max: Does not have a max value.
Min: 1
Nom: 10

Set of values for (red,green,blue)

Max: (1.0,1.0,1.0)
Min: (0.1,0.1,0.1)
Nom: Any value in the middle of min to max

10.6 Drawing a solid Cube

10.7 Testing for transparency with two cubes

void ns2scube(float x1, float y1, float z1, float x2, float y2, float z2,float red, float green, float blue,float alpha);

Testing for transparency by making two cubes one inside the other while making the outer cube transparent and the inner cube solid.

Test values for outer cube:

float x1 = -0.8, x2 = +0.8; float y1 = -0.8, y2 = +0.8; float z1 = -0.8, z2 = +0.8;

float r = 1.0, g = 0.3, b = 0.2;

float alpha = 0.4;

Test values for the inner cube:

float x11 = -0.7, x22 = +0.7; float y11 = -0.7, y22 = +0.7; float z11 = -0.7, z22 = +0.7;

float r1 = 0.8, g1 = 0.3, b1 = 0.2; float alpha1 = 1;

Setting alpha to 1 makes the object opaque. The test shows the inner cube clearly inside the transparent outer cube.

10.8 Drawing a Disk

void s2diskxy(float px, float py, float pz, float r1, float r2);

Draw a disk in the xy plane, at a given z coordinate. The disk has an inner and outer radius and the annulus is filled. The radius is given in world coordinates, and the actual radius used is the quadratic mean of the radius converted to x and y normalised device units.

Range of values on r1 (inner radius) and r2 (outer radius)

Max: Does not have a set max value.
Min: r1=0.1, r2=0.2
Nom: r1=0.5, r2=0.8

10.9 Drawing a rectangle

void s2rectxy(float xmin, float xmax, float ymin, float ymax, float z);

Draw a rectangle in the xy plane, at a given z coordinate. The rectangle is a filled quadrangle. The edge of the rectangle is NOT drawn. Use s2line if you need a border to your rectangles.

Set of max values:
Max values are not set
Set of min values:
xmin = 0.4, xmax = +0.6; ymin = 0.1, ymax = +0.2; z = -0.3;
Set of nom values:
xmin = -0.6, xmax = +0.6; ymin = -0.2, ymax = +0.2; z = +0.3;

10.10 Drawing an annulus

void ns2disk(float x, float y, float z, float nx, float ny, float nz, float r1, float r2, float red, float green, float blue);

Draw an annulus with given centre (x,y,z), inner and outer radii (r1, r2) and RGB colour (red, green, blue). The annulus is oriented with normal vector (nx,ny,nz).

Testing with varying values of r1 (inner) and r2 (outer) radii

Max Values: inner = 0.1; outer = 1;
Min Values: inner = 0.1; outer = 0.2;
Nom Values: inner = 0.1; outer = 0.5;

10.11 Drawing an arc

void ns2arc(float px, float py, float pz, float nx, float ny, float nz, float sx, float sy, float sz, float deg, int nseg);

Draw an arc centered around (px,py,pz) with surface normal (nx,ny,nz) using the current pen thickness and colour. Vector (sx,sy,sz) gives the position (both direction and radius) relative to (px,py,pz), from which the first segment of the arc is drawn. 'deg' specifies the arc angle in degrees, which is broken into 'nseg' segments.

Range of values for deg (arc angle in degrees)

Max: deg = 355.0
Min: deg = 30.0
Nom: deg = 180.0

Range of nseg (number of line segments) values on an arc of 180 degrees

Max: No max number of line segments set as the more the number of lines used the better the clarity of the arc.
Min: nseg = 10
Nom: nseg = 20

10.12 Drawing an arc with an axis ratio

void ns2erc(float px, float py, float pz,float nx, float ny, float nz, float sx, float sy, float sz, float deg, int nseg, float axratio);

Draw an arc with specified major/minor axis ratio. The major axis is given by the starting vector.

Range of values for deg of the arc

Max: deg = 355.0
Min: deg = 30.0
Nom: deg = 180.0

Range of values for nseg on an arc of 180 degrees

Max: No max number of line segments set as the more the number of lines used the better the clarity of the arc
Min: nseg = 10
Nom: nseg = 20

Range of values for axratio (Axis Ratio)

Max: axratio = 10
Min: axratio = 0.1
Nom: axration = 0.5

10.13 Drawing a point

void ns2point(float x, float y, float z, float red, float green, float blue);

Draw a point at the position (x,y,z) with RBG colour (red,green,blue).

Range of values for (red,green,blue)

Max: (1.0,1.0,1.0)
Min: (0.1,0.1,0.1)
Nom: Any value in the range of min to max

10.14 Drawing a given number of points

void ns2vnpoint(XYZ *P, COLOUR col, int n);

Draw n points at positions P with RBG colour (col), using vector data structures.

Range of values for n (total number of points)

Max: 100
Min: 2
Nom: 50

Range of values for colour

Max: col.r = 1.0; col.g = 1.0; col.b = 1.0;
Min: col.r = 0.1; col.g = 0.1; col.b = 0.1;
Nom: Any value between 0.1 and 1

10.15 Drawing a thick point

void ns2thpoint(float x, float y, float z, float red, float green, float blue, float size);

Draw a thick point at the position (x,y,z) with RBG colour (red,green,blue) and thickness size pixels (not world coordinates).

Range of values for size (size of the point)

Max: No set value for size the point can be as big as possible
Min: 1
Nom: 5

Range of values for (red,green,blue)

Max: (1.0,1.0,1.0)
Min: (0.1,0.1,0.1)
Nom: Any value in the middle of min to max

10.16 Drawing a thick transparent point

void ns2vpa(XYZ P, COLOUR icol, float isize, char itrans, float ialpha);

Draw a thick, transparent point with specified size (isize), transparency is one of 's' (standard blend), 't' (transparent) or 'o' (opaque), and alpha channel (ialpha). Takes vector inputs for position P and colour icol.

Testing when setting itrans = s or itrans = t

Range of values for ialpha (alpha channel)

Max: 1.0
Min: 0.1
Nom: 0.5

When setting itrans = o
Max, Min or Nom value does not matter as the object is opaque.

Range of values for size

Max: No set value for size the point can be as big as possible
Min: 1
Nom: 5

10.17 Placing an OpenGL light

void ns2i(float x, float y, float z, float red, float green, float blue);

Place an OpenGL light at location (x,y,z) with RGB colour (red,green,blue).

Range of values for (red,green,blue)

Max: (1.0,1.0,1.0)
Min: (0.1,0.1,0.1)
Nom: Any value in the range of min to max

10.18 Drawing a Line

void ns2line(float x1, float y1, float z1, float x2, float y2, float z2, float red, float green, float blue);

Draw a line from (x1,y1,z1) to (x2,y2,z2) using RGB colour (red,green,blue).

Range of Values for the co-ordinates

Values for Max:
No max value set. Line segment of any length can be drawn.
Values for Min:
x1 = 1.0; y1 = 1.0; z1 = 1.0; x2 = 1.1; y2 = 1.1; z2 = 1.1;
Values for Nom:
x1 = 1.0; y1 = 1.0; z1 = 1.0; x2 = 2.0; y2 = 2.0; z2 = 2.0;

Range of values for (red,green,blue)

Max: (1.0,1.0,1.0)
Min (0.1,0.1,0.1)
Nom: Any value in the range of min to max

10.19 Drawing a thick line

void ns2thline(float x1, float y1, float z1, float x2, float y2, float z2, float red, float green, float blue, float width);

Draw a thick line (width) from (x1,y1,z1) to (x2,y2,z2) using RGB colour (red,green,blue).

Range of values for the width

Max: 10
Min: 2
Nom: 5

Range of values for the co-ordinates

Values for Max:
No max value set. Line segment of any length can be drawn.
Values for Min:
x1 = 1.0; x2 = 1.1; y1 = 1.0; y2 = 1.1; z1 = 1.0; z2 = 1.1;
Values for Nom:
x1 = 1.0; y1 = 1.0; z1 = 1.0; x2 = 2.0; y2 = 2.0; z2 = 2.0;

Range of values for (red,green,blue)

Max: (1.0,1.0,1.0)
Min: (0.1,0.1,0.1)
Nom: Any value in the range of min to max

10.20 Drawing a blended coloured line

void ns2cline(float x1, float y1, float z1, float x2, float y2, float z2, float red1, float green1, float blue1, float red2, float green2, float blue2);

Draw a coloured line from (x1,y1,z1) to (x2,y2,z2). The colour is blended along the line between the two input RGB colours (red1,green1,blue1) and (red2,green2,blue2).

Range of values for the co-ordinates

Values for Max:
No max value set. Line segment of any length can be drawn.
Values for Min:
z1 = 1.0; x2 = 1.1; y2 = 1.1; x1 = 1.0; y1 = 1.0; z2 = 1.1;
Values for Nom:
x1 = 1.0; y1 = 1.0; z1 = 1.0; x2 = 2.0; y2 = 2.0; z2 = 2.0;

Range for the color values

Values for Max:

r1 = 0.9; g1 = 0.9; b1 = 0.9; r2 = 1.0; g2 = 1.0; b2 = 1.0;
Values for Min:
r1 = 0.1; g1 = 0.1; b1 = 0.1; r2 = 0.2; g2 = 0.2; b2 = 0.2;
Values for Nom:
r1 = 0.5; g1 = 0.5; b1 = 0.5; r2 = 0.6; g2 = 0.6; b2 = 0.6;

10.21 Drawing a blended coloured thick line

void ns2thcline(float x1, float y1, float z1, float x2, float y2, float z2, float red1, float green1, float blue1, float red2, float green2, float blue2,float width);

Draw a thick coloured line, with colour blended between the two given colours along the line, and given width.

Range of values for the co-ordinates

Values for Max:
No max value set. Line segment of any length can be drawn.
Values for Min:
z1 = 1.0; x2 = 1.1; y2 = 1.1; x1 = 1.0; y1 = 1.0; z2 = 1.1;
Values for Nom:
x1 = 1.0; y1 = 1.0; z1 = 1.0; x2 = 2.0; y2 = 2.0; z2 = 2.0;

Range of values for colour

Values of Max:
r1 = 0.9; g1 = 0.9; b1 = 0.9; r2 = 1.0; g2 = 1.0; b2 = 1.0;
Values of Min:
r1 = 0.1; g1 = 0.1; b1 = 0.1; r2 = 0.2; g2 = 0.2; b2 = 0.2;
Values of Nom:
r1 = 0.5; g1 = 0.5; b1 = 0.5; r2 = 0.6; g2 = 0.6; b2 = 0.6;

Range of values for width

Max: 10
Min: 2
Nom: 5

10.22 Drawing a marker

void ns2m(int type, float size, float x, float y, float z, float red, float green, float blue);

Draw a marker at (x,y,z) with size and RGB colour (red,green,blue). Argument type should be one of: 0 = tetrahedron (pyramid) 1 = wireframe 3D cross 2 = shaded box 3 = octahedron (diamond)

Range of values for size

Max: No max value set for size
Min: size =1;
Nom: size =5;

Range for Type values

Max: 4

Min: 0
Nom: Any value between 0 to 4

Range of values for (Red,Green,Blue)

Max: (1.0,1.0,1.0)
Min: (0.1,0.1,0.1)
Nom: Any value in the range of min to max

10.23 Drawing a dynamic billboard

void ds2bb(float x, float y, float z, float strx, float stry, float strz, float isize, float r, float g, float b, unsigned int itextid, float alpha, char trans);

Draw a (dynamic) billboard. This is a textured facet (4 vertex) that sits at a given location and always faces the camera. By using small, rotationally symmetric texture maps, the appearance of soft, 3d objects can be produced at low frame-rate cost. This function should only be called from a dynamic callback: billboards cannot be cached static geometry as they typically change with every refresh. The billboard texture is at coordinates (x,y,z), and can be stretched towards the coordinates (strx, stry, strz) (for no stretching, set these coordinates to 0.0). The overall scale of the billboard texture is controlled by isize. The RGB texture colour is specified by parameters (r, g, b), and the texture to use is identified by itextid. Transparency is controlled by the alpha channel, with value in the range [0,1] and trans: trans = 'o' opaque; trans = 't' transparent; and trans = 's' transparency + absorption.

Range of values for isize (size of the texture)

Max: Does not have a max value set. Texture can be of any size.
Min: 0.01
Nom: 0.06

Range of values for the stretching co-ordinates

Max:
Does not have a max value as it can stretch up to any co-ordinate position
Min:
float strx = 0, stry = 0, strz = 0;
Nom:
float strx = 5.0, stry = 5.0, strz = 5.0;

Range of values for (red, green, blue)

Max: (1.0,1.0,1.0)
Min: (0.1,0.1,0.1)
Nom: Any value in the middle of min to max

Range of values for ialpha (Alpha Channel) which configures the transparency

Max: 1.0
Min: 0.2
Nom: 0.5

10.24 Drawing a dynamic billboard with a rotation angle

void ds2vbbr(XYZ iP, XYZ iStretch, float isize, float ipangle, COLOUR iC, unsigned int itexid, float alpha, char trans);

Draw a (dynamic) ”billboard”, with rotation ipangle (in degrees) of the texture about the view direction

Range of values for pangle

Max: 360
Min: 0
Nom; 45

10.25 Drawing an error

void s2arro(float x1, float y1, float z1,float x2, float y2, float z2);

Draw an arrow from the point with world-coordinates (x1,y1,z1) to (x2,y2,z2).

Max:
No max values set as a line segment of any length can be drawn.
Min:
x1 = -0.3, x2 = -0.5; y1 = -0.3, y2 = -0.5; z1 = -0.3, z2 = -0.5;
Nom:
x1 = 0.5, x2 = -0.5; y1 = 0.5, y2 = -0.5; z1 = 0.5, z2 = -0.5;

10.26 Setting arrowhead style

void s2sah(int fs, float angle, float barb);

Set the style to be used for arrowheads drawn with s2arro. Currently, all arrowheads are drawn as cones, with the rendering mode (wireframe or shaded) affecting the look. Line-based arrowheads may be added at a later stage. Argument fs is the fill style: fs = 1: filled fs = 2: outline [not yet implemented] Argument angle [5.0, 135.0] is the angle of the arrow point in degrees, with default value 45.0 (a cone of semi-vertex angle 22.5 degrees). Argument barb [0.05, 1.0] is the fraction of the cone that is cut away from the back, such that 0.05 gives an open cone and 1.0 gives a cone with a solid base, with default value 0.3.

Range of values for angle

Max: 360
Min: 0
Nom: 45

Range of values for barb

Max: 1.0 (gives a solid cone)
Min: 0.05 (gives an open cone)
Nom: 0.5

10.27 Setting the character height

void s2sch(float size);

Set the character height in "arbitrary" units. The default character size is 1.0, corresponding to a character about 15 pixels in size. Changing the character height also changes the scale of tick marks and symbols.

Range of values for size

Max: Does not have a set value. Arrow head can be as big as possible.
Min: 1
Nom: 5

10.28 Setting Pen colour index

void s2sci(int idx);

Set the pen colour by index. If it lies outside the defined range, it will be set to the default colour (white).

Range of values for idx

Max: 15 (Anything above 15 will display white as it is out of range)
Min: 1
Nom: Any value between Max and Min

10.29 Setting line width

void s2slw(float width);

Set the linewidth in pixels. The default is 1. Changing the line width affects lines and vector symbols. Unlike PGPLOT, text is not affected.

Range of values for width

Max: No max value set as line can be as wide as possible
Min: 1
Nom: 5

10.30 Setting Line style

void s2sls(int ls);

Set the line style for drawing with s2line. Codes are identical to PGPLOT: 1 = solid line (default line style) 2 = dashed 3 = dot-dash-dot-dash 4 = dotted 5 = dash-dot-dot-dot

Range of values for ls (Line Style)

Max: 5
Min: 0
Nom: Any value between Max and Min

10.31 Drawing error bars

void s2errb(int dir, int n, float *xpts, float *ypts, float *zpts, float *edelt, int termsize);

Draw error bars at the coordinates (xpts[i], ypts[i], zpts[i]). Error bars are drawn in the direction indicated by argument dir as described below. One-sided error bar lengths are given by edelt, such that for error bars in eg. the x direction, error bars are drawn to xpts[i]+edelt[i], to xpts[i]-edelt[i], or to both. Argument termsize gives the size of the terminals to draw for each error bar; it is given in an integer increment of the current linewidth. Eg. if t=1, then end points are one pixel larger than the line width used to draw the bars.

Range of values for dir

Max: 9
Min: 1
Nom: Any value between Min and Max

Range of values for np (number of points)

Max: No max value set. Can draw any number of error bars
Min: 1
Nom: 100

Range of values for termsize

Max: No max value set. Size of the terminals can be anything.
Min: 1
Nom: 5

10.32 Drawing a curve defined by a parametric equations

void s2funt(float(*fx)(float*), float(*fy)(float*), float(*fz)(float*), int N, float tmin, float tmax);

Draw a curve defined by parametric equations fx(t), fy(t) and fz(t). N points are constructed, uniformly spaced from tmin to tmax.

Range of values between tmin and tmax

Maximum range between tmin and tmax: No max value set
Minimum range between tmin and tmax: 1
Nominal range between tmin and tmax: 5

Range of values for N (number of points required to plot the equation)

Max: No max value set. Any number of points can be used to plot the equation.
Min: 1
Nom: 50

10.33 Drawing a surface by a function fxy

void s2funxy(float(*fxy)(float*,float*), int nx, int ny, float xmin, float xmax, float ymin, float ymax, int ctl);

Draw the surface described by the function fxy. The function is evaluated on a nx * ny grid whose world coordinates extend from (xmin,ymin) to (xmax,ymax). The ctl argument is the control function. ctl = 0: curve plotted in current window and viewport. Caller is responsible for setting the viewport and world coordinate system suitably. ctl = 1: s2env is called automatically to fit the plot in the current viewport.

Minimum difference between xmax and xmin

Max difference: No max difference defined.
Min difference: 1
Nom difference: 4

Minimum difference between ymax and ymin

Max difference: No max difference defined.
Min difference: 1
Nom difference: 4

Range of values for ctl

Max: 1
Min: 0
Nom: 0 or 1

10.34 Drawing a three vertex facet

void ns2vf3(XYZ *P, COLOUR col);

Draw a 3-vertex facet with a single colour. The vertices are given by the array P, normals are calculated automatically, and the RGB colour is col.

Range of values for colour

Max: col = 1.0, 1.0, 1.0 ;
Min: col = 0.1,0.1,0.1
Nom: Any values between min and max.

10.35 Adding a new panel

int xs2ap(float x1, float y1, float x2, float y2);

Add a new panel to the S2PLOT window. The panel goes from (x1,y1) to (x2,y2) where these are fractions of the window coordinates. Individual panels can be activated and deactivated by providing the panel id to the toggle function.

Range of values for x1,y1 and x2, y2 in the function in order of (float x1, float y1, float x2, float y2)

Max: No max value set. The panel can be as big as possible.
Min: (0.0, 0.0, 0.1, 0.1)
Nom: (0.0, 0.0, 1.0, 1.0)

10.36 Setting Panel properties

void xs2spp(COLOUR active, COLOUR inactive, float width);

Set panel frame properties.

Range of values for width

Max: No max value defined. Width of the panel frame can be as much as possible.
Min: 1.0
Nom: 5.0

Range of values for colour for both active and inactive

Max: (1.0,1.0,1.0)
Min: (0.1,0.1,0.1)
Nom: Any value between Max and Min

10.37 Set foreground colour

void ss2sfc(float r, float g, float b);

Set the foreground colour. This call should usually be followed by calls to s2scr to set the 1st colour index to be the same as the foregrond, and the 0th colour index to be the opposite.

Range of values for (Red,Green,Blue) values

Max: (1.0,1.0,1.0)
Min: (0.1,0.1,0.1)
Nom: Any value between max and min

10.38 Set background colour

void ss2sbc(float r, float g, float b);

Set the background colour. This call should usually be followed by calls to s2scr to set the 0th colour index to be the same as the background, and the 1st colour index to be the opposite.

Range of values for (Red,Green,Blue) values

Max: (1.0,1.0,1.0)
Min: (0.1,0.1,0.1)
Nom: Any value between max and min

10.39 Drawing a 3-D textured polygon

void ns2texpoly3d(XYZ *iP, XYZ *iTC, float in, unsigned int texid, char itrans, float ialpha);

Draw a 3-d textured polygon. The array iP holds the vertex coordinates, array iTC holds the texture coordinates, parameter in is the number of vertices, itrans is one of 'o' (opaque), 't' (transparent) or 's' (standard) and ialpha is the alpha channel value.

Range of values for ialpha (Alpha Channel)

Max: 1.0
Min: 0.1
Nom: 0.5

Range of values for in(number of vertices in the ploygon)

Max: Any number of vertices can be used in making the polygon
Min: 3
Nom: 10

10.40 Set Sphere resolution

void ss2ssr(int res);

Set sphere resolution. Spheres are drawn with (res*res) flat surfaces. Larger spheres (or spheres that will be viewed closer-up) require higher sphere resolutions. Be warned that rendering time takes a severe hit with resolutions much larger than about 12.

Range of values for res

Max: No max value set. The resolution can be set to any value.
Min: 1
Nom: 10

10.41 Set projection type

void ss2spt(int projtype);

Set the projection type of the device in use. Only certain changes are allowed.

Range of values for projtype

Max: 2
Min: 0
Nom: Any value between Max and Min

10.42 Setting whether clipping is enabled

void s2twc(int enabledisable);

Set whether clipping is being applied to geometry. When enabled, clipping means points not within the world coordinate bounds as defined by the most recent call to s2swin, will not be drawn. The initial implementation applies to points only, future work may apply the clipping to lines and facets as well.

Range of values for enabledisable

Max: 1
Min: 0
Nom: Does not have a nom value as the function acts as a switch.

10.43 Setting cursor cross-hair

void ss2txh(int enabledisable);

Enable, disable or toggle the visibility of the cursor cross-hair depending on the value of enabledisable. The cross-hair can also be toggled by using the key combination Cntrl-C.

Range of values for enabledisable

Max: enabledisable = 1 Enable cross-hair
Min: enabledisable = 0 Disable cross-hair
Nom: Does not have nom values as this function acts as a switch