```
// PlaceSpheresProc.cpp

#include <ri.h>
#include <RixInterfaces.h>
#include <RiTypesHelper.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// A RiProcedural must implement these functions. This is a fixed requirement.
extern "C"
{
        PRMANEXPORT RtPointer ConvertParameters ( RtString paramStr             );
        PRMANEXPORT RtVoid    Subdivide         ( RtPointer data, RtFloat detail );
        PRMANEXPORT RtVoid    Free              ( RtPointer data                 );
}

// A custom data structure for defining an arbitrary number of spheres
// positioned at locations defined by the "coords" array.
typedef struct {
        RtFloat radius;
        RtFloat *coords;                    // an array of x,y,z coordinates
        RtInt    num_coords;
        } SpheresData;

RtFloat randBetween(RtFloat min, RtFloat max);
// ------------------------------------------------
// A RiProcedural required function
// ------------------------------------------------
//
RtPointer ConvertParameters(RtString paramStr) {
        // The strtok() function cannot be used on the paramStr directly because
        // it modifies the string.
        long len = strlen(paramStr);

        // We could directly create a copy of the input paramStr as an array and
        // use the strcpy(), string copy, function.
        //char copyStr[len];
        //strcpy(copyStr, paramStr);

        // However, because the paramStr can be very large we allocate memory
        // from the main memory pool (the "heap") and then perform a block
        // copy of the contents of paramStr.
```

```
        char *copyStr = (char*)calloc(len + 1, sizeof(char));
        memcpy(copyStr, paramStr, len + 1);

        // Allocate a block of memory to store one instance of SpheresData.
        SpheresData *dataPtr = (SpheresData*)calloc(1, sizeof(SpheresData));

        // Irrespective of how many values are specified by the paramStr we
        // know the first two values will specify the radius of the spheres
        // and the number of coordinates that define their 3D locations.
        sscanf(copyStr, "%f %d", &dataPtr->radius, &dataPtr->num_coords);

        // Allocate memory to store an array of coordinates
        RtInt num_floats = dataPtr->num_coords;
        dataPtr->coords = (RtFloat*)calloc(num_floats, sizeof(RtFloat));

        char *strPtr = strtok(copyStr, " ");
        strPtr = strtok(NULL, " "); // eat the radius value
        strPtr = strtok(NULL, " "); // eat the num coordinates value
        long count = 0;
        while(strPtr) {
                // Convert each string to a double precision floating point number
                dataPtr->coords[count] = strtod(strPtr, NULL);
                count++;
                strPtr = strtok(NULL, " "); // grab the next part of copyStr.
                }
        // Don't forget to free the memory that was allocated for the copied text.
        free(copyStr);
        return (RtPointer)dataPtr;
}

// ------------------------------------------------
// A RiProcedural required function
// ------------------------------------------------
RtVoid Subdivide(RtPointer data, RtFloat detail) {
   RtFloat radius = ((SpheresData*)data)->radius;
        RtInt    num_coords = ((SpheresData*)data)->num_coords;
        RtFloat *coords =  ((SpheresData*)data)->coords;

        for(int n = 0; n < num_coords; n = n + 3) {
                RtFloat x = coords[n];
                RtFloat y = coords[n + 1];
                RtFloat z = coords[n + 2];
                RiTransformBegin();
```

```
                        RiTranslate(x,y,z);
                        // To assign a color to each sphere un-comment the next two lines
                        // and comment line 92
                        //RtColor cs[1] = {
{randBetween(0,1),randBetween(0,1),randBetween(0,1) } } ;
                //RiSphere(radius, -radius, radius, 360, "constant color Cs", (RtPointer)cs,
RI_NULL);
                RiSphere(radius, -radius, radius, 360, RI_NULL);
                RiTransformEnd();
                }
        }
// ----------------------------------------------------
// A RiProcedural required function
// ----------------------------------------------------
RtVoid Free(RtPointer data) {
        free(((SpheresData*)data)->coords);
   free(data);
        }

// ----------------------------------------------------
// Our utility functions begin here
// ----------------------------------------------------
RtFloat randBetween(RtFloat min, RtFloat max) {
   return ((RtFloat)rand()/RAND_MAX) * (max - min) + min;
   }
```