```
/*
   Linux:
     cc -I../include colorspheres.c -L../lib -lribout -o colorspheres

   Win32:
     cl /I..\include /c colorspheres.c
     link colorspheres.obj ..\lib\libribout.lib /out:colorspheres.exe
*/

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

#define NFRAMES 100
#define NSPHERES  4
#define FRAMEROT 15.0f

void ColorSpheres(int n, float s)
{
    int x, y, z;
    RtColor color;

    if(n <= 0)  return;

    RiAttributeBegin();

    RiTranslate(-0.5f, -0.5f, -0.5f );
    RiScale(1.0f/n, 1.0f/n, 1.0f/n);

    for (x = 0; x < n; x++) {
      for (y = 0; y < n; y++) {
          for (z = 0; z < n; z++) {
            color[0] = ((float) x+1) / ((float) n);
            color[1] = ((float) y+1) / ((float) n);
            color[2] = ((float) z+1) / ((float) n);

            RiColor(color);
            RiTransformBegin();
            RiTranslate(x+.5f, y+.5f, z+.5f);
            RiScale(s, s, s);
            RiSphere(0.5f, -0.5f, 0.5f, 360.0f, RI_NULL);
            RiTransformEnd();
          }
      }
    }

    RiAttributeEnd();
    return ;
}

int main(int argC, char** argV)
{
    RtInt  frame;
    float  scale;
    char   filename[64];
    char*  renderer = RI_NULL;

    if (argC != 2) {
      fprintf(stderr,
            "USAGE: %s ribFile|rgl|rendrib\n\n",
            argV[0]);
      exit (-1);
    }
    renderer = argV[1];

    /* if the variable renderer is "rgl" or "rendrib", RiBegin() will
       attempt to start up that renderer and pipe its output directly to
       that renderer.  Since RiDisplay is set to put its output to the
       framebuffer, that renderer will attempt to open the framebuffer
       and render directly to it.  If the variable renderer is set to
       some other value, RiBegin() will open that as a file and put the
       RIB commands in it.
    */

    RiBegin(renderer);

    for (frame = 0; frame <= NFRAMES; frame++) {
      sprintf(filename, "colorSpheres.%03d.tif", frame );
      RiFrameBegin(frame);

        RiProjection("perspective", RI_NULL);
        RiTranslate(0.0f, 0.0f, 1.5f);
        RiRotate(40.0f, -1.0f, 1.0f, 0.0f);

        RiDisplay(filename, RI_FRAMEBUFFER, RI_RGBA, RI_NULL);
        RiFormat((RtInt)256, (RtInt)192, -1.0f);
        RiShadingRate(1.0);

        RiWorldBegin();

          RiLightSource("distantlight", RI_NULL);

          RiSides((RtInt)1);

          scale = (float)(NFRAMES-(frame-1)) / (float)NFRAMES;
          RiRotate(FRAMEROT*frame, 0.0f, 0.0f, 1.0f);
          RiSurface("plastic", RI_NULL);
          ColorSpheres(NSPHERES, scale);
        RiWorldEnd();
      RiFrameEnd();
    }

    RiEnd();
    return 1;
}




/* ============================================================= */
NOTE: The following is written in RenderMan shader language, not in C.
/* ============================================================= */
/*  plastic.sl - simple plastic surface
 */

surface
plastic (float Ka = 1, Kd = 0.5, Ks = 0.5, roughness = 0.1;
       color specularcolor = 1;)
{
    normal Nf = faceforward (normalize(N),I);
    Ci = Cs * (Ka*ambient() + Kd*diffuse(Nf)) +
       specularcolor * Ks*specular(Nf,-normalize(I),roughness);
    Oi = Os;  Ci *= Oi;
}
```