

```

void Okudagram::LoadSkymap ()
{
    nSkyVertices    = (360/dtheta) * (180/dphi) * 4;

    SkyVertices     = new VERTEX[nSkyVertices];

    AUX_RGBImageRec *pTexture = auxDIBImageLoad ("Skymap.bmp");

    if (!pTexture)    return;

    glGenTextures    (1, &Skymap_TextureID);           // Generate space for Texture Objects

    glBindTexture    (GL_TEXTURE_2D, Skymap_TextureID);

    glTexEnvf        (GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);

    glPixelStorei    (GL_UNPACK_ALIGNMENT,            1);
    glPixelStorei    (GL_UNPACK_ROW_LENGTH,           0);
    glPixelStorei    (GL_UNPACK_SKIP_ROWS,           0);
    glPixelStorei    (GL_UNPACK_SKIP_PIXELS,         0);

    glTexImage2D     (GL_TEXTURE_2D, 0, 3, pTexture->sizeX, pTexture->sizeY, 0, GL_RGB, GL_UNSIGNED_BYTE, pTexture->data);
    glTexImage2D     (GL_TEXTURE_2D, 1, 3, pTexture->sizeX, pTexture->sizeY, 0, GL_RGB, GL_UNSIGNED_BYTE, pTexture->data);
    glTexImage2D     (GL_TEXTURE_2D, 2, 3, pTexture->sizeX, pTexture->sizeY, 0, GL_RGB, GL_UNSIGNED_BYTE, pTexture->data);
    glTexImage2D     (GL_TEXTURE_2D, 3, 3, pTexture->sizeX, pTexture->sizeY, 0, GL_RGB, GL_UNSIGNED_BYTE, pTexture->data);
    glTexImage2D     (GL_TEXTURE_2D, 4, 3, pTexture->sizeX, pTexture->sizeY, 0, GL_RGB, GL_UNSIGNED_BYTE, pTexture->data);
    glTexImage2D     (GL_TEXTURE_2D, 5, 3, pTexture->sizeX, pTexture->sizeY, 0, GL_RGB, GL_UNSIGNED_BYTE, pTexture->data);
    glTexImage2D     (GL_TEXTURE_2D, 6, 3, pTexture->sizeX, pTexture->sizeY, 0, GL_RGB, GL_UNSIGNED_BYTE, pTexture->data);
    glTexImage2D     (GL_TEXTURE_2D, 7, 3, pTexture->sizeX, pTexture->sizeY, 0, GL_RGB, GL_UNSIGNED_BYTE, pTexture->data);
    glTexImage2D     (GL_TEXTURE_2D, 8, 3, pTexture->sizeX, pTexture->sizeY, 0, GL_RGB, GL_UNSIGNED_BYTE, pTexture->data);

    glTexParameterf (GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,    GL_REPEAT);
    glTexParameterf (GL_TEXTURE_2D, GL_TEXTURE_WRAP_T,    GL_REPEAT);
    glTexParameterf (GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameterf (GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);

    free    (pTexture->data);
    free    (pTexture);

    int     vindex = 0;

```

```

float    vx, vy, vz, magnitude;

for      (int phi = 0; phi <= 180 - dphi; phi += dphi)           // phi controls pole-to-pole position (z-axis)
{
    for   (int theta = 0; theta <= 360 - dtheta; theta += dtheta) // theta controls 360 degree position (x & y axes)
    {
        vx    = SkyVertices[vindex].x = a2 * sin(phi*DTOR) * cos(theta*DTOR); // Define texture polygon vertices
        vy    = SkyVertices[vindex].y = b2 * sin(phi*DTOR) * sin(theta*DTOR);
        vz    = SkyVertices[vindex].z = c2 * cos(phi*DTOR);

        magnitude = sqrt (pow(vx, 2) + pow(vy, 2) + pow(vz, 2));

        vx    /= magnitude;
        vy    /= magnitude;
        vz    /= magnitude;

        SkyVertices[vindex].u = hTile * (asin(vx) / PI) + 0.5f;
        SkyVertices[vindex].v = vTile * (asin(vy) / PI) + 0.5f;

        vindex++;

        vx    = SkyVertices[vindex].x = a2 * sin((phi+dphi)*DTOR) * cos(theta*DTOR);
        vy    = SkyVertices[vindex].y = b2 * sin((phi+dphi)*DTOR) * sin(theta*DTOR);
        vz    = SkyVertices[vindex].z = c2 * cos((phi+dphi)*DTOR);

        magnitude = sqrt (pow(vx, 2) + pow(vy, 2) + pow(vz, 2));

        vx    /= magnitude;
        vy    /= magnitude;
        vz    /= magnitude;

        SkyVertices[vindex].u = hTile * (asin(vx) / PI) + 0.5f;
        SkyVertices[vindex].v = vTile * (asin(vy) / PI) + 0.5f;

        vindex++;

        vx    = SkyVertices[vindex].x = a2 * sin(phi*DTOR) * cos((theta+dtheta)*DTOR);
        vy    = SkyVertices[vindex].y = b2 * sin(phi*DTOR) * sin((theta+dtheta)*DTOR);
        vz    = SkyVertices[vindex].z = c2 * cos(phi*DTOR);

        magnitude = sqrt (pow(vx, 2) + pow(vy, 2) + pow(vz, 2));
    }
}

```

```

vx      /= magnitude;
vy      /= magnitude;
vz      /= magnitude;

SkyVertices[vindex].u = hTile * (asin(vx) / PI) + 0.5f;
SkyVertices[vindex].v = vTile * (asin(vy) / PI) + 0.5f;

vindex++;

vx      = SkyVertices[vindex].x = a2 * sin((phi+dphi)*DTOR) * cos((theta+dtheta)*DTOR);
vy      = SkyVertices[vindex].y = b2 * sin((phi+dphi)*DTOR) * sin((theta+dtheta)*DTOR);
vz      = SkyVertices[vindex].z = c2 * cos((phi+dphi)*DTOR);

magnitude = sqrt (pow(vx, 2) + pow(vy, 2) + pow(vz, 2));

vx      /= magnitude;
vy      /= magnitude;
vz      /= magnitude;

SkyVertices[vindex].u = hTile * (asin(vx) / PI) + 0.5f;
SkyVertices[vindex].v = vTile * (asin(vy) / PI) + 0.5f;

vindex++;
}

glNewList      (Skymap_DL, GL_COMPILE);

glBegin        (GL_TRIANGLE_STRIP);

for            (int i = 0; i < nSkyVertices; i++)
{
    glTexCoord2f (SkyVertices[i].u, SkyVertices[i].v);
    glVertex3f   (SkyVertices[i].x, SkyVertices[i].y, SkyVertices[i].z);
}

glEnd          ();

glEndList      ();
}

```