Errata :

It's possible to speed-up the process by using the build-in user's clipping. In that case, you have to enable clipping for four planes:

```c
glEnable(GL_CLIP_PLANE0);
glEnable(GL_CLIP_PLANE1);
glEnable(GL_CLIP_PLANE2);
glEnable(GL_CLIP_PLANE3);
```

The shaders are transformed as follow:

**Vertex Shader**

```c
#version 120
#pragma optimize(on)

varying out vec4 vertex;
varying out vec3 normal;

uniform vec2 divscreensplit;

vec2 decal_ = -1.0+divscreensplit;
mat4x4 initialposition = mat4x4(divscreensplit.x,0.0,0.0,0.0,
                                  0.0,divscreensplit.y,0.0,0.0,
                                  0.0,0.0,1.0,0.0,
                                  decal_.x,decal_.y,
                                  0.0,1.0)*gl_ProjectionMatrix*gl_ModelViewMatrix;

void main(void){
    normal = gl_Normal;
    gl_FrontColor = gl_Color;
    gl_TexCoord[0] = gl_MultiTexCoord0;
    vertex = gl_Vertex;
    gl_Position = initialposition*gl_Vertex;
}
```

**Geometry Shader**

```c
#version 150 compatibility
#pragma optimize(on)

in vec3 normal[3];

out vec3 normalo;
out float eyeveco;
out vec3 lightveco;

uniform vec2 divscreensplit;
uniform vec2 screensplit;
uniform int numberofviews;
uniform float dio;

vec2 decal_ = -1.0+divscreensplit;
mat4x4 initialposition = mat4x4(divscreensplit.x,0.0,0.0,0.0,
void main(void){
    vec4 pos[3];
    vec2 viewid;
    float start = float(numberofviews*0.5)*dio;
    if(mod(numberofviews,2)==0) start -= dio*0.5;
    for(int current=0;current<numberofviews;++current){
        viewid.y = int(floor(current*divscreensplit.x));
        viewid.x = current%int(screensplit.x);

        vec4 tmp = initialposition*vec4(start,0.0,0.0,0.0);
        for(int i =0; i < 3; ++i){
            pos[i] = tmp + gl_PositionIn[i];
            vec2 coeff = 2.0*divscreensplit*pos[i].w;
            gl_ClipDistance[0] = pos[i].x + pos[i].w;
            gl_ClipDistance[1] = coeff.x - (pos[i].x + pos[i].w);
            gl_ClipDistance[2] = pos[i].y + pos[i].w;
            gl_ClipDistance[3] = coeff.y - (pos[i].y + pos[i].w);
            normalo = normal[i];
            gl_Position = pos[i];
            gl_Position.xy += viewid*coeff;
            gl_FrontColor = gl_FrontColorIn[i];
            gl_TexCoord[0] = gl_TexCoordIn[i][0];
            EmitVertex();
        }
        EndPrimitive();
        start -= dio;
    }
}

Fragment Shader

#version 120
#extension GL_EXT_gpu_shader4 : enable
#pragma optimize(on)

uniform sampler2D texture;

void main(void){
    gl_FragColor = texture2D(texture, gl_TexCoord[0].st);
}

Moreover the fragment shader is now unchanged.