

# Heat Haze

## Recommended Reading

[Introduction to Particles](#)

[Transparency and Refraction](#)

## Video

[http://www.leadwerks.com/files/Tutorials/CPP/Heat\\_Haze.wmv](http://www.leadwerks.com/files/Tutorials/CPP/Heat_Haze.wmv)

## Files

[http://developer.leadwerks.com/Tutorials/CPP/Heat\\_Haze\\_Files.zip](http://developer.leadwerks.com/Tutorials/CPP/Heat_Haze_Files.zip)

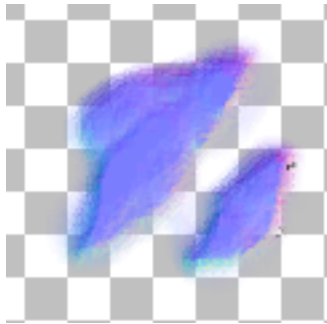


Heat haze refracts light near a heat source (exaggerated for visibility).

## Introduction

Heat haze is an effect that simulates the refraction of light as it passes through a temperature gradient. It's a subtle but impressive effect that can make games look much more interesting.

We're going to simulate a fire by using one particle emitter for flames, and a second emitter for the heat haze effect. The heat haze material uses the regular particle vertex shader along with a special heat haze pixel shader. The pixel shader will read the background color and depth values, and display a skewed image. A normal map with alpha channel will be used for the heat refraction angle and intensity.



Heat haze normal map showing alpha channel.

The source code for the particle\_heathaze.frag shader is show below:

```
uniform sampler2D texture0;// normal + alpha map
uniform sampler2D texture1;// background
uniform sampler2D texture2;// depth
uniform vec2 buffersize;
uniform vec2 camerarange;
uniform float refractionstrength=0.01;

varying vec2 texcoord0;

Include "abstract::DepthToZPosition.frag"

void main(void) {
    vec4 color = gl_Color * texture2D( texture0, texcoord0 );

    if (color.a<0.003) {
        discard;
    }

    vec3 normal = color.xyz * 2.0 - 1.0;

    vec4 refractionvector=vec4( gl_FragCoord.x/buffersize.x, gl_FragCoord.y/buffersize.y, gl_FragCoord.z, 1.0 );
    refractionvector += refractionstrength * vec4(normal,0.0);

    if (gl_FragCoord.z>DepthToZPosition(texture2DProj(texture2,refractionvector).x)) {
        discard;
    }

    vec4 haze = texture2DProj(texture1,refractionvector);

    gl_FragColor = vec4( haze.xyz, color.w );
}
```

This program will create two emitters and display the heat haze effect:

```
#include "engine.h"

int main(int argc, char** argv)
{
    Initialize();

    //Create a graphics context
    Graphics(800,600);

    //Create a world
    TWorld world=CreateWorld();
    if (!world) {
        MessageBoxA(0,"Error","Failed to create world.",0);
        goto exitapp;
    }

    //Create a render buffer
    TBuffer gbuffer=CreateBuffer(800,600,BUFFER_COLOR|BUFFER_DEPTH|BUFFER_COLOR1);
    TBuffer lightbuffer=CreateBuffer(800,600,BUFFER_COLOR);

    //Create a camera
    TEntity campiv=CreatePivot();
    PositionEntity(campiv,Vec3(0,1,0));
    TCamera cam=CreateCamera(campiv);
    PositionEntity(cam,Vec3(0,0,-4));

    //Load the scene
    LoadMesh("scene.gmf");

    TWorld foreground=CreateWorld();
    TCamera fgcam=CreateCamera();
    CameraClearMode(fgcam,0);

    //Create the fire emitter
    TEmitter fire = CreateEmitter(100,1000,Vec3(0,2,0));
    EntityColor(fire,Vec4(1,0.5,0,0.1));
    PaintEntity(fire,LoadMaterial("fire.mat"));
    SetEmitterRadius(fire,0.5,0.25);
    SetEmitterWaver(fire,0.5);
    SetEmitterRotationSpeed(fire,0.5);
    SetEmitterArea(fire,Vec3(1,0.5,1));

    //Create the heat haze emitter
    TEmitter heat = CreateEmitter(20,2000,Vec3(0,1.5,0));
    TMaterial material=LoadMaterial("heathaze.mat");
    SetMaterialTexture(material,GetColorBuffer(lightbuffer),1);
    PaintEntity(heat,material);
    SetEmitterRadius(heat,0.75,0.75);
    SetEmitterWaver(heat,0.5);
    SetEmitterRotationSpeed(heat,0.1);
    SetEmitterArea(heat,Vec3(1,0.5,1));

    SetWorld(world);

    //Create a light
    TLight light=CreatePointLight();
    PositionEntity(light,Vec3(0,0.25,0));
    EntityColor(light,Vec4(1,0.5,0,0.1));
}
```

```

float mx=0.0,my=0.0;
TVec3 camrotation;
HideMouse();
MoveMouse(GraphicsWidth()/2,GraphicsHeight()/2);

//Main loop
while(!KeyHit(KEY_ESCAPE)) {

    mx=Curve(MouseX()-GraphicsWidth()/2,mx,3);
    my=Curve(MouseY()-GraphicsHeight()/2,my,3);
    MoveMouse(GraphicsWidth()/2,GraphicsHeight()/2);
    camrotation=EntityRotation(campiv);
    camrotation.X+=my;
    camrotation.Y-=mx;
    RotateEntity(campiv,camrotation);

    UpdateWorld();

    //Render the main world
    SetBuffer(gbuffer);
    SetWorld(world);
    RenderWorld();

    //Render lighting
    SetBuffer(lightbuffer);
    RenderLights(gbuffer);

    CopyBuffer(lightbuffer,BackBuffer(),BUFFER_COLOR);
    SetBuffer(BackBuffer());

    //Update and render the foreground world
    SetWorld(foreground);

    SetEntityMatrix(fgcam,GetEntityMatrix(cam));
    UpdateWorld(AppSpeed());
    RenderWorld();
    SetWorld(world);

    Flip();
}

exitapp:
return Terminate();
}

```

Copyright Leadwerks Software 2008

All Rights Reserved.