Project 4

Alan Neads

17th February, 2017

decaying.glib

```
##OpenGL GLIB

Perspective 70
LookAt 0 0 3  0 0 0  0 1 0

Vertex   decaying.vert
Fragment  decaying.frag
Program   Decaying                                \
            uA <-0.5 0.00 0.5>                    \
            uB <0.0 2.0 5.0>                      \
            uC <0.0 0.0 12.56>                    \
            uD <0. 0. 5.>                         \
            uNoiseAmp  <0.0 0. 5.>                \
            uNoiseFreq <0.1 1. 5.>                \
            uKa <0. 0.1 1.0>                      \
            uKd <0. 0.6 1.0>                      \
            uKs <0. 0.3 1.0>                      \
            uShininess <0.0 10. 50.>              \
            uLightX <-20. 0.0 20.>                \
            uLightY <-20. 0.0 20.>                \
            uLightZ <-20. 20. 20.>                \
            uMaterialColor {1.0 1.0 1.0 1.0}      \
            uLightColor {1. .7 0. 1.}             \
            uSpecularColor {1. 1. 1. 1.}


QuadXY  -0.2  1.  200 200
```

decaying.vert

```
uniform float uA;
uniform float uB;
uniform float uC;
uniform float uD;

out vec3 vPosition;
out vec3 vNormal;

out vec3 modelVec;

#define MATH_PI 3.14159265358979323846264433
#define MATH_E        2.71828182845904523536028752

void main( void )
{
	gl_TexCoord[0] = gl_MultiTexCoord0;

	float radius = gl_Vertex.x * gl_Vertex.x + gl_Vertex.y * gl_Vertex.y;
	gl_Vertex.z += uA * cos(2 * MATH_PI * uB * radius + uC) * pow(MATH_E, -(uD * radius));

	gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
	vPosition = gl_Vertex;

	float drdx = 2.0 * gl_Position.x;
	float dzdx = -uA * sin(2.0 * MATH_PI * uB * radius + uC) * 2.0 * MATH_PI * uB * drdx *
exp(-uD * radius) + uA * cos(2.0 * MATH_PI * uB * radius + uC) * exp(-uD * radius) * -uD * drdx;

	float drdy = 2.0 * gl_Position.y;
	float dzdy = -uA * sin(2.0 * MATH_PI * uB * radius + uC) * 2.0 * MATH_PI * uB * drdy *
exp(-uD * radius) + uA * cos(2.0 * MATH_PI * uB * radius + uC) * exp(-uD * radius) * -uD * drdy;

	vec3 Tx = vec3( 1.0, 0.0, dzdx );
	vec3 Ty = vec3( 0.0, 1.0, dzdy );

	gl_Normal = normalize( cross(Tx, Ty) );
	vNormal = gl_Normal;

	modelVec = vec3(gl_ModelViewMatrix * gl_Vertex);
}
```

decaying.frag

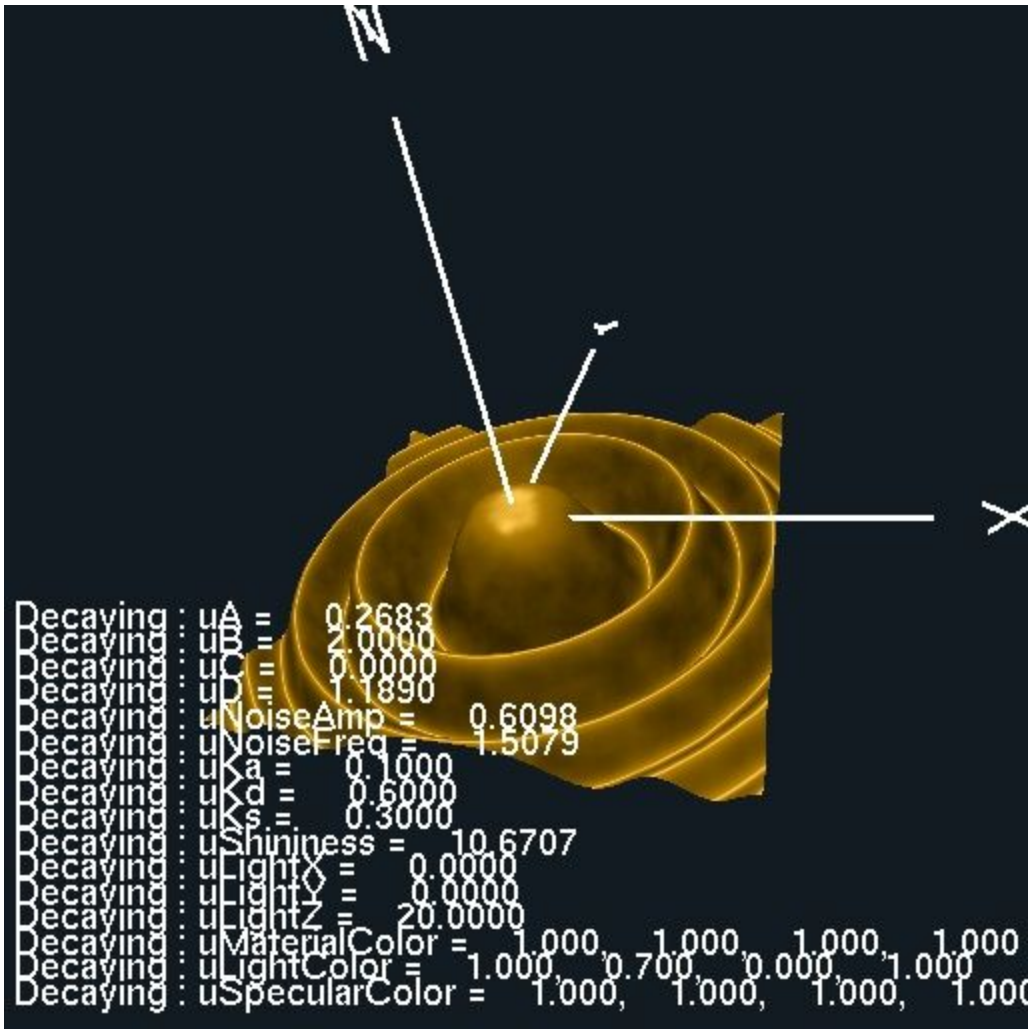
```
uniform float uNoiseAmp;
uniform float uNoiseFreq;
uniform sampler3D Noise3;

uniform float uKa;
uniform float uKd;
uniform float uKs;
uniform float uShininess;
uniform float uLightX;
uniform float uLightY;
uniform float uLightZ;
uniform vec4 uMaterialColor;
uniform vec4 uLightColor;
uniform vec4 uSpecularColor;

in vec3 vPosition;
in vec3 vNormal;

in vec3 modelVec;

vec3 RotateNormal( float angx, float angy, vec3 n )
{
	float cx = cos( angx );
	float sx = sin( angx );
	float cy = cos( angy );
	float sy = sin( angy );

	// rotate about x:
	float yp =  n.y*cx - n.z*sx;    // y'
	n.z      =  n.y*sx + n.z*cx;    // z'
	n.y      =  yp;
	// n.x      =  n.x;

	// rotate about y:
	float xp =  n.x*cy + n.z*sy;    // x'
	n.z      = -n.x*sy + n.z*cy;    // z'
	n.x      =  xp;
	// n.y      =  n.y;

	return normalize( n );
}
```

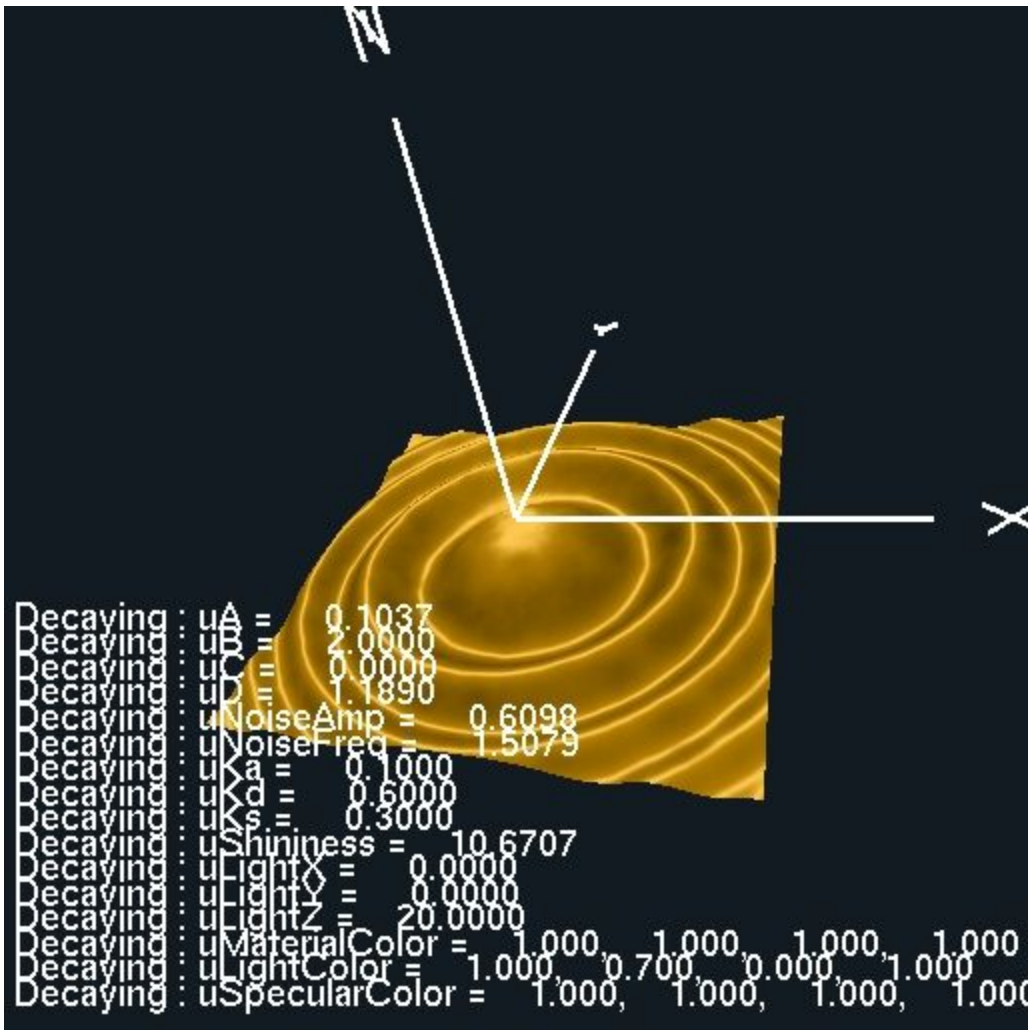
```
void main( void )
{
	vec4 nvx = texture3D(Noise3, vPosition * uNoiseFreq);
	vPosition.z += 0.5;
	vec4 nvy = texture3D(Noise3, vPosition * uNoiseFreq);
	float angx = (nvx[0] + nvx[1] + nvx[2] + nvx[3] - 2.0) * uNoiseAmp;
	float angy = (nvy[0] + nvy[1] + nvy[2] + nvy[3] - 2.0) * uNoiseAmp;

	vNormal = RotateNormal( angx, angy, vNormal );

	vec3 lightPos = vec3(uLightX, uLightY, uLightZ);
	vec4 Color = uLightColor * uMaterialColor;

	vec3 L = normalize( lightPos - modelVec );
	vec3 E = normalize(-modelVec);
	vec3 R = normalize( reflect( -lightPos, vNormal ) );

	vec3 ambient = Color.rgb;
	vec3 diffuse = max( dot(L, vNormal), 0.0 ) * Color.rgb;
	vec3 spec = uSpecularColor * pow( max(dot(R,E), 0.0), uShininess );

	gl_FragColor.rgb = uKa * ambient + uKd * diffuse + uKs * spec;
	gl_FragColor.a = 1.0;
}
```

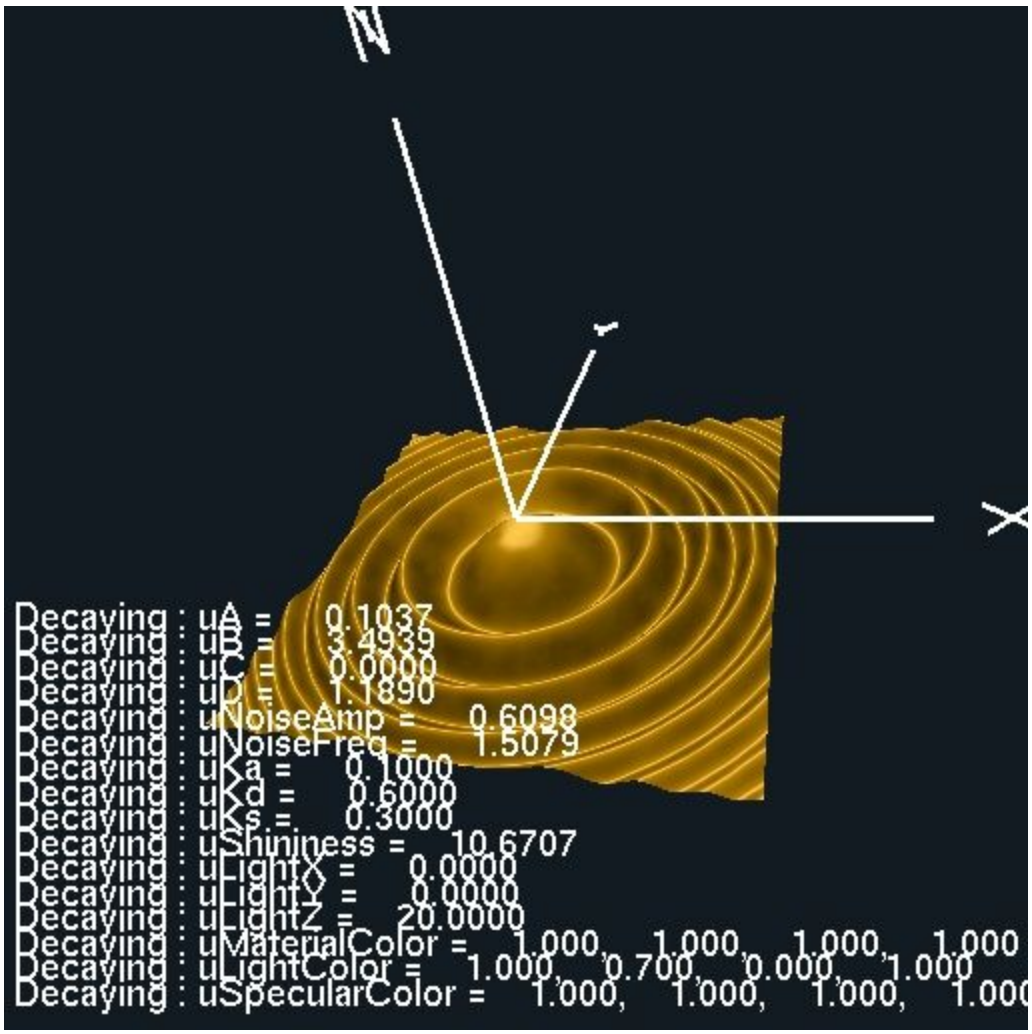
Deconstruction

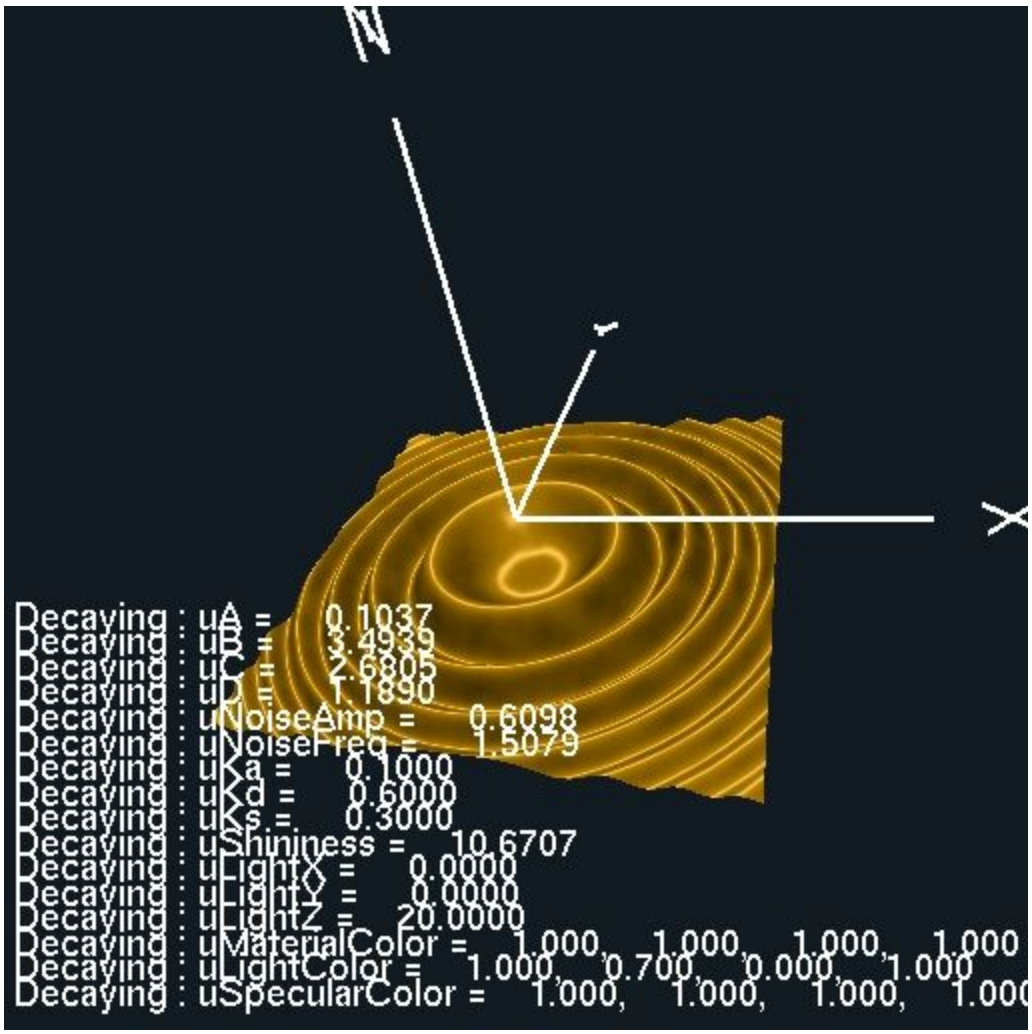
I started with the modification of the vertices position along the z-axis by adjusting it by the amplitude times the cosine of our frequency plus offset, multiplying by the decaying effect. This produces a plane with waves, centering in the middle of the plane. Modifying uA adjusts the maximum amplitude of the waves, uB adjusts the frequency of the waves, uC is an offset value for the frequency, and lastly uD is the rate of amplitude decay. Implementing this is as simple as modifying the z value of the x-y plane by adding the output of the above function. Computing the new surface normal is as simple as taking the cross product of the two vectors representing the angles of each axi. This corrects the normal of the surface, allowing us to utilize it in our noise and lighting computation. The lighting function I implemented for this was smoothed lighting, implementing in the fragment shader. I provided functionality to control the color of the surface material, the light color, and the specular reflection color. Lastly I implemented the bump mapping by rotating the normal by the output of two noise functions, thus providing coherent deformations to the surface of the plane. This can be observed easily by finding an angle in which steep slopes of the displacement have specular reflection visible, and then modifying the noise such that it changes the reflective surface to no longer reflect light to the viewer's eye.

Images


Decaying : uA = 0.2683
Decaying : uB = 2.0000
Decaying : uC = 0.0000
Decaying : uD = 1.1890
Decaying : uNoiseAmp = 0.6098
Decaying : uNoiseFreq = 1.5079
Decaying : uKa = 0.1000
Decaying : uKd = 0.6000
Decaying : uKs = 0.3000
Decaying : uShininess = 10.6707
Decaying : uLightX = 0.0000
Decaying : uLightY = 0.0000
Decaying : uLightZ = 20.0000
Decaying : uMaterialColor = 1.000, 1.000, 1.000, 1.000
Decaying : uLightColor = 1.000, 0.700, 0.000, 1.000
Decaying : uSpecularColor = 1.000, 1.000, 1.000, 1.000


N
Y
X
Decaying : uA = 0.1037
Decaying : uB = 2.0000
Decaying : uC = 0.0000
Decaying : uD = 1.1890
Decaying : uNoiseAmp = 0.6098
Decaying : uNoiseFreq = 1.5079
Decaying : uKa = 0.1000
Decaying : uKd = 0.6000
Decaying : uKs = 0.3000
Decaying : uShininess = 10.6707
Decaying : uLightX = 0.0000
Decaying : uLightY = 0.0000
Decaying : uLightZ = 20.0000
Decaying : uMaterialColor = 1.000, 1.000, 1.000, 1.000
Decaying : uLightColor = 1.000, 0.700, 0.000, 1.000
Decaying : uSpecularColor = 1.000, 1.000, 1.000, 1.000


Z
X
Decaying : uA = 0.1037
Decaying : uB = 3.4939
Decaying : uC = 0.0000
Decaying : uD = 1.1890
Decaying : uNoiseAmp = 0.6098
Decaying : uNoiseFreq = 1.5079
Decaying : uKa = 0.1000
Decaying : uKd = 0.6000
Decaying : uKs = 0.3000
Decaying : uShininess = 10.6707
Decaying : uLightX = 0.0000
Decaying : uLightY = 0.0000
Decaying : uLightZ = 20.0000
Decaying : uMaterialColor = 1.000, 1.000, 1.000, 1.000
Decaying : uLightColor = 1.000, 0.700, 0.000, 1.000
Decaying : uSpecularColor = 1.000, 1.000, 1.000, 1.000


N
Y
X
Decaying : uA = 0.1037
Decaying : uB = 3.4939
Decaying : uC = 2.6805
Decaying : uD = 1.1890
Decaying : uNoiseAmp = 0.6098
Decaying : uNoiseFreq = 1.5079
Decaying : uKa = 0.1000
Decaying : uKd = 0.6000
Decaying : uKs = 0.3000
Decaying : uShininess = 10.6707
Decaying : uLightX = 0.0000
Decaying : uLightY = 0.0000
Decaying : uLightZ = 20.0000
Decaying : uMaterialColor = 1.000, 1.000, 1.000, 1.000
Decaying : uLightColor = 1.000, 0.700, 0.000, 1.000
Decaying : uSpecularColor = 1.000, 1.000, 1.000, 1.000