Stripes, Rings, and Dots!

Oregon State University

Mike Bailey

mjb@cs.oregonstate.edu

Oregon State University
Computer Graphics

Cartesian (X) Stripes

stripes.glib

```
##OpenGL GLIB

Perspective 90
LookAt  0 0 2   0 0 0   0 1 0

Vertex      stripes.vert
Fragment stripes.frag
Program  Stripes                                    \
                        uA <0 1. 10>                \
                        uP <0. .25 1.>              \
                        uTol <0. 0. .5>             \
                        uAmp <-5. 0. 5.>            \
                        uFreq <0. 10. 20.>
Color 1 0.5 0
Sphere 1 200 200
```

Oregon State University
Computer Graphics

Cartesian (X) Stripes

stripes.vert

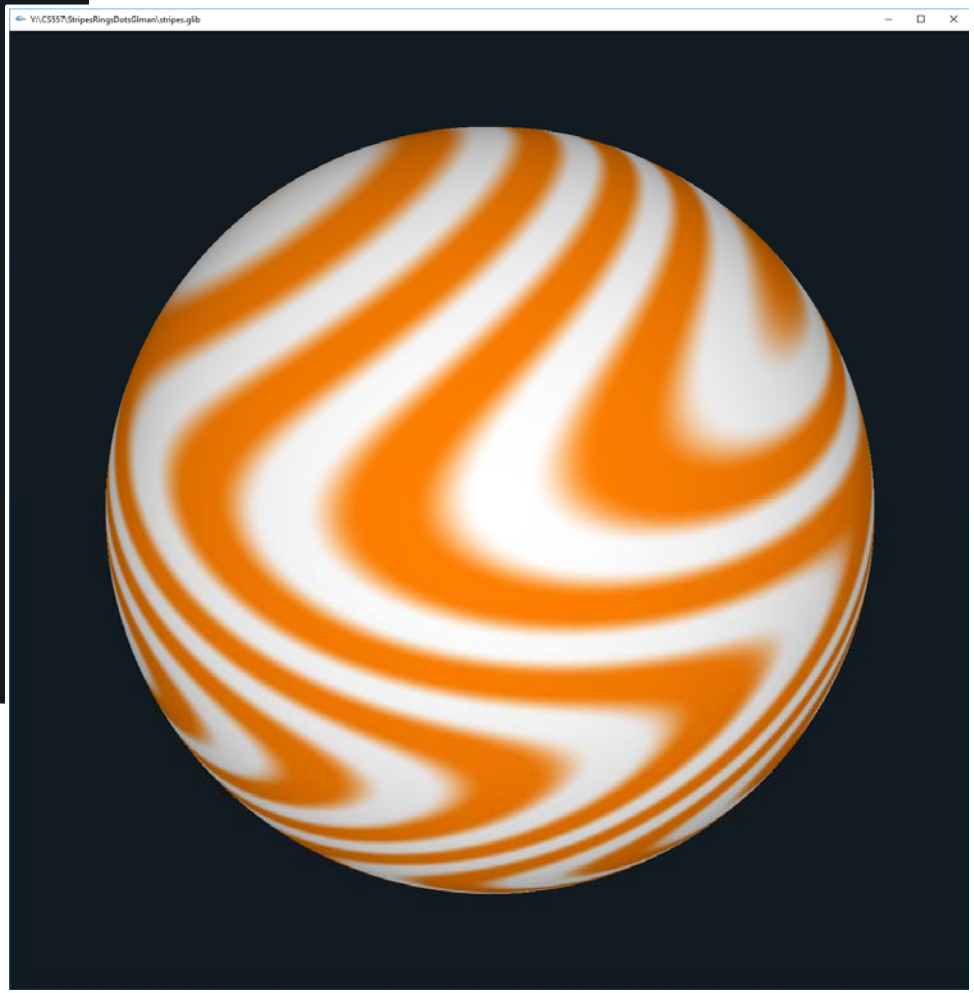
```
#version 330 compatibility

uniform float  uAmp;
uniform float  uFreq;

out vec4  vColor;
out float vX, vY;
out float vLightIntensity;

const vec3 LIGHTPOS = vec3( 0., 0., 10. );

void
main( )
{
        vec3 tnorm = normalize( gl_NormalMatrix * gl_Normal );
        vec3 ECposition = ( gl_ModelViewMatrix * gl_Vertex ).xyz;
        vLightIntensity  = abs( dot( normalize(LIGHTPOS - ECposition), tnorm ) );

        vColor = gl_Color;
        vec3 MCposition = gl_Vertex.xyz;
        vX = MCposition.x;
        vY = MCposition.y;

        vX = vX + uAmp * sin( uFreq * vY );

        gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
}
```

Cartesian (X) Stripes

stripes.frag

```
#version 330 compatibility

uniform float uA;
uniform float uP;
uniform float uTol;

in float   vX, vY;
in vec4  vColor;
in float   vLightIntensity;

const vec4 WHITE = vec4( 1., 1., 1., 1. );

void
main( )
{
	float f = fract( uA*vX );

	float t = smoothstep( 0.5-uP-uTol, 0.5-uP+uTol, f )  -  smoothstep( 0.5+uP-uTol, 0.5+uP+uTol, f );
	gl_FragColor = mix( WHITE, vColor, t );
	gl_FragColor.rgb *= vLightIntensity;
}
```

Cartesian (X) Stripes



Rings

rings.glib

```
##OpenGL GLIB

Perspective 90
LookAt  0 0 2   0 0 0   0 1 0

Vertex   rings.vert
Fragment rings.frag
Program  Rings                                          \
                        uA <0 5. 10>                    \
                        uP <0. .25 1.>                  \
                        uTol <0. 0. .5>
Color 1 0.5 0
Sphere 1 200 200
```

Oregon State University
Computer Graphics

Rings

rings.vert

```
#version 330 compatibility

uniform float  uAmp;
uniform float  uFreq;

out vec4  vColor;
out float vX, vY;
out float vLightIntensity;

const vec3 LIGHTPOS = vec3( 0., 0., 10. );

void
main( )
{
	vec3 tnorm = normalize( gl_NormalMatrix * gl_Normal );
	vec3 ECposition = ( gl_ModelViewMatrix * gl_Vertex ).xyz;
	vLightIntensity  = abs( dot( normalize(LIGHTPOS - ECposition), tnorm ) );

	vColor = gl_Color;
	vec3 MCposition = gl_Vertex.xyz;
	vX = MCposition.x;
	vY = MCposition.y;

	gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
}
```

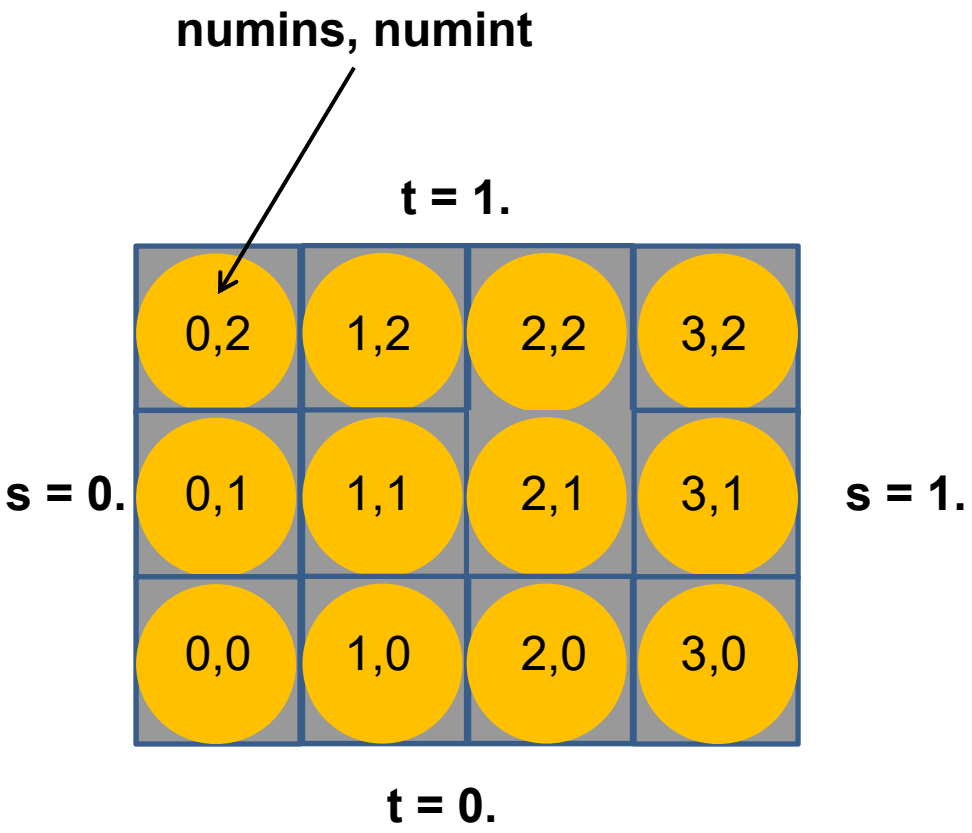
Rings

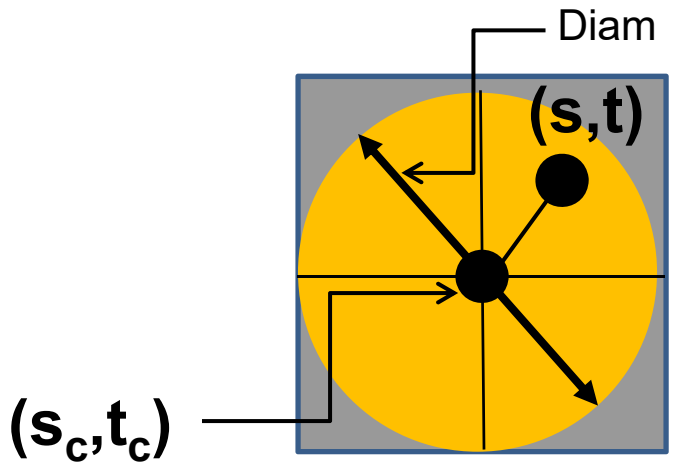
rings.frag

```
#version 330 compatibility

uniform float uA;
uniform float uP;
uniform float uTol;

in float   vX, vY;
in vec4  vColor;
in float  vLightIntensity;

const vec4 WHITE = vec4( 1., 1., 1., 1. );

void
main( )
{
        float r = sqrt( vX*vX + vY*vY );
        float rfrac = fract( uA*r );

        float t = smoothstep( 0.5-uP-uTol, 0.5-uP+uTol, rfrac )  -  smoothstep( 0.5+uP-uTol, 0.5+uP+uTol, rfrac );
        gl_FragColor = mix( WHITE, vColor, t );
        gl_FragColor.rgb *= vLightIntensity;
}
```

Rings (= Polar Stripes)

Circular Dots are a "Local Pattern"

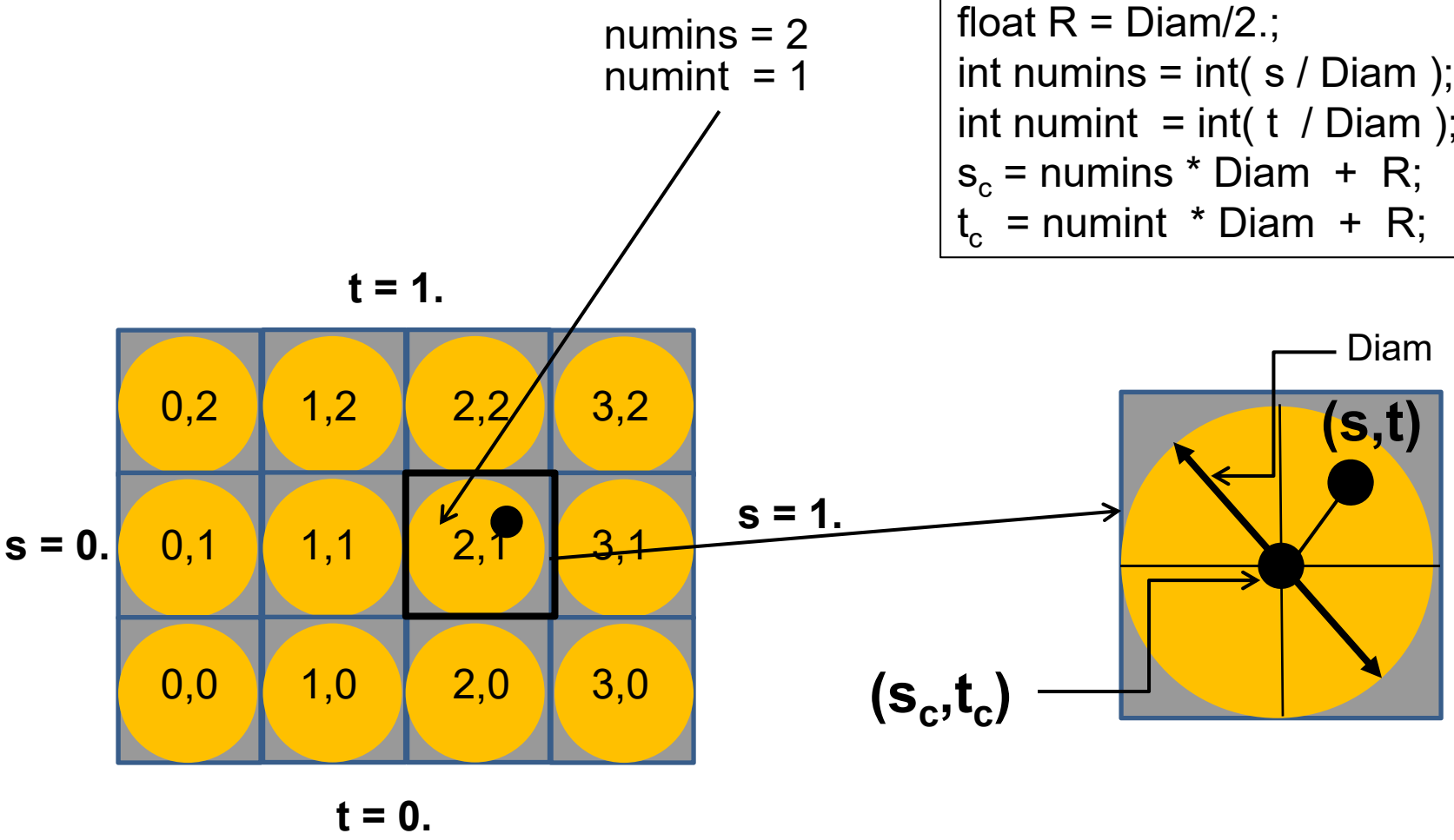




$$(s - s_c)^2 + (t - t_c)^2 \leq \left(\frac{Diam}{2}\right)^2$$

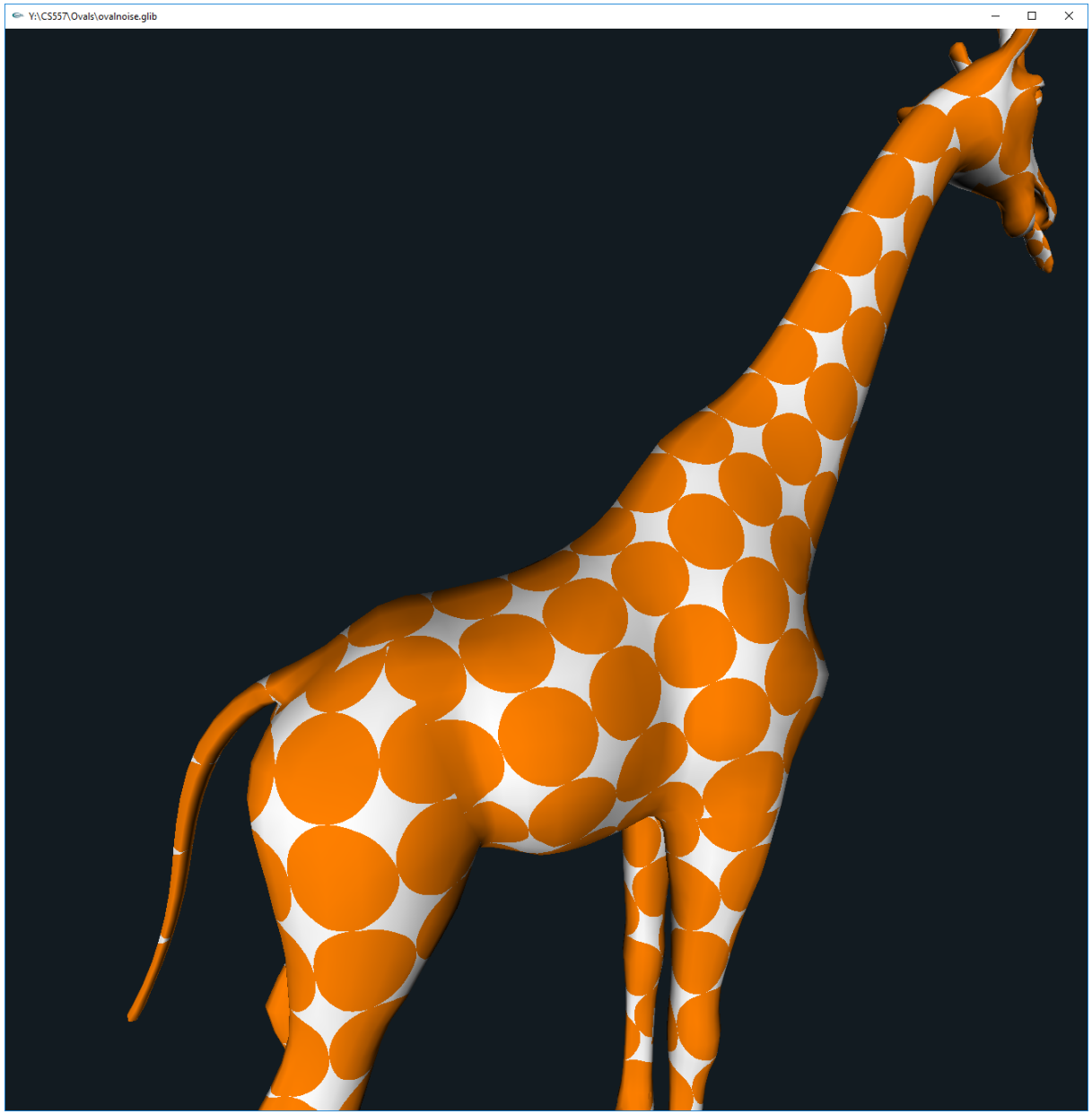
Circular Dots

```
float R = Diam/2.;
int numins = int( s / Diam );
int numint  = int( t  / Diam );
sc = numins * Diam  +  R;
tc  = numint  * Diam  +  R;
```



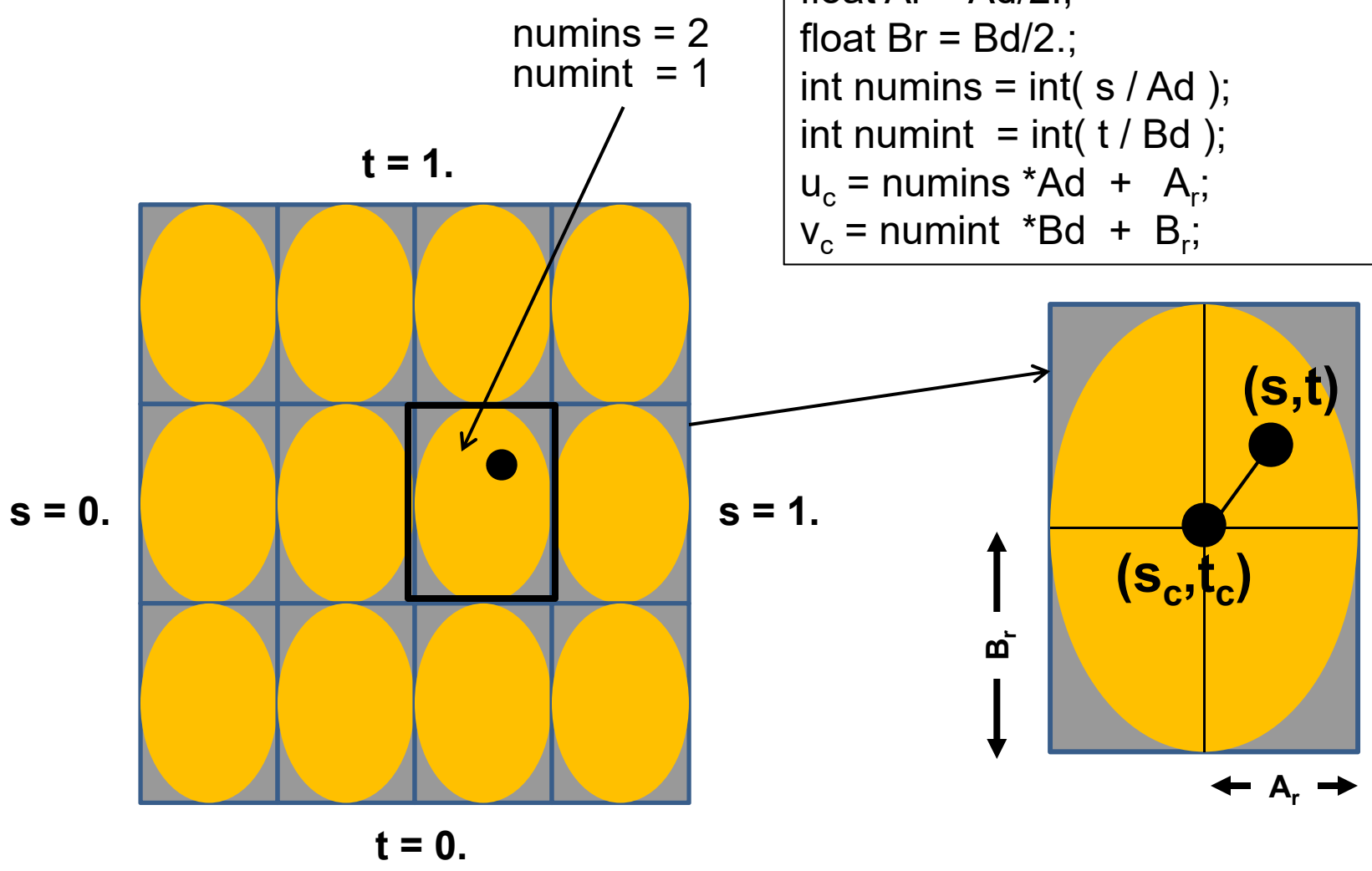
$$(s - s_c)^2 + (t - t_c)^2 \leq (R)^2$$

Circular Dots



Elliptical Dots

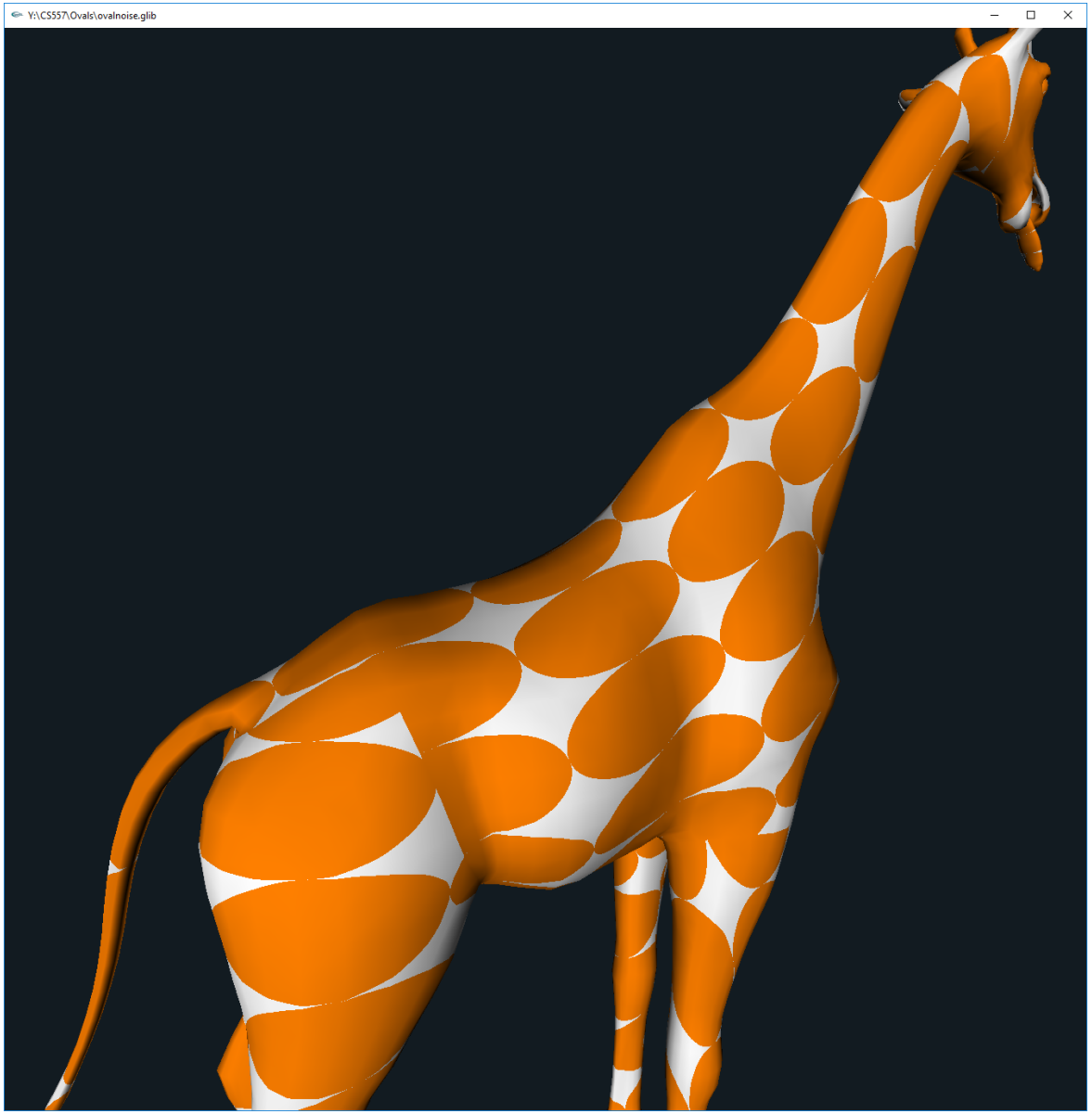
```
float Ar = Ad/2.;
float Br = Bd/2.;
int numins = int( s / Ad );
int numint  = int( t / Bd );
u_c = numins *Ad  +  A_r;
v_c = numint  *Bd  +  B_r;
```



$$(s-s_c)^2+(t-t_c)^2 \le R^2 \Rightarrow \left(\frac{s-s_c}{R}\right)^2+\left(\frac{t-t_c}{R}\right)^2 \le 1 \Rightarrow \left(\frac{s-s_c}{A_r}\right)^2+\left(\frac{t-t_c}{B_r}\right)^2 \le 1$$

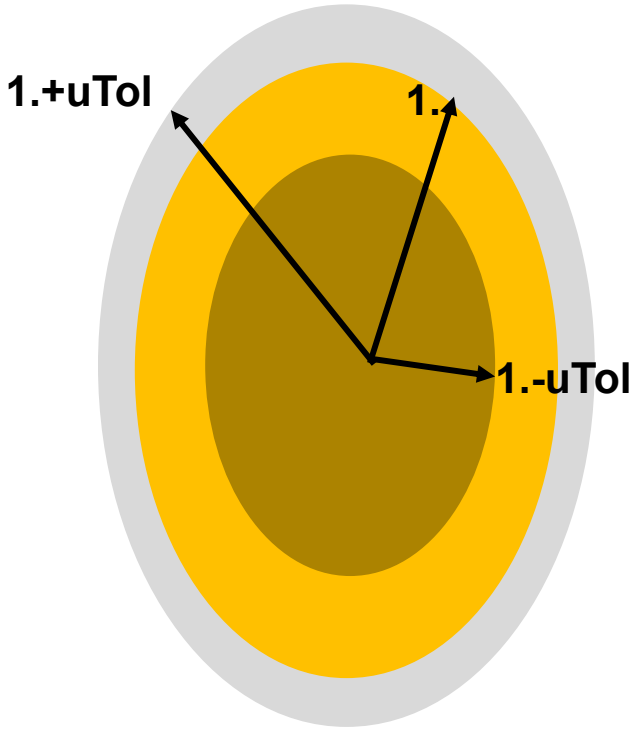
Circle — Circle — Ellipse

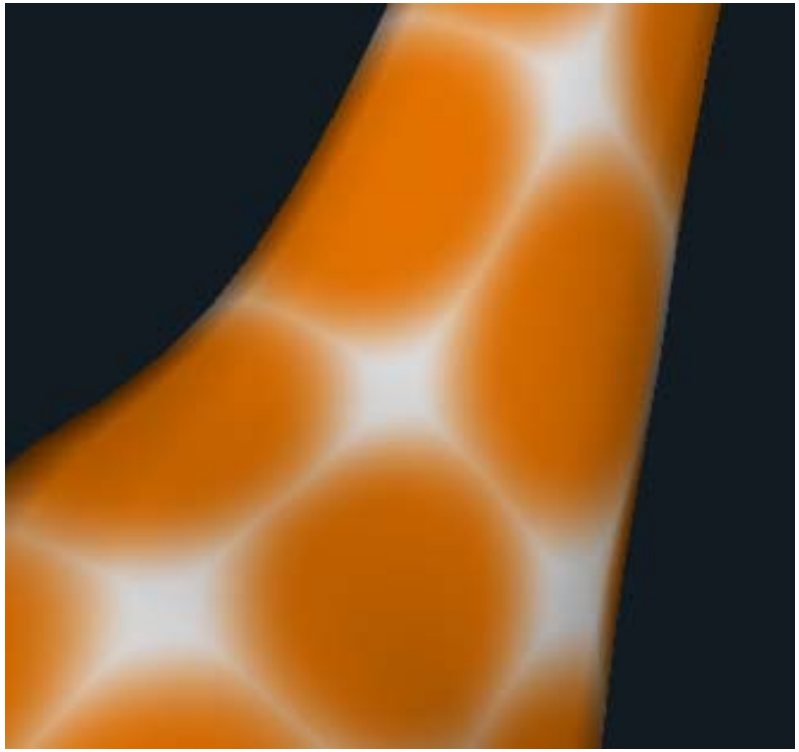
Elliptical Dots



Elliptical Dots with Tolerance

Elliptical Dots with Tolerance





$$1 - uTol \leq \left(\frac{s - s_c}{A_r}\right)^2 + \left(\frac{t - t_c}{B_r}\right)^2 \leq 1 + uTol$$

Coming Soon -- Elliptical Dots with Noise!