# Lecture 19

## Combining 2D and 3D

- De-bugging of the last sample code
- Billboarding
- Program Point Size
- Point Sprite
- Particle methods

## De-bugging of the last sample code

- Error: float diffuse=dot(normalOut,lit);
- Correct: float diffuse=max(0.0,dot(normalOut,lit));

- Error:
  float specular=specularIntensity*pow(dot(midDir,normalOut),specularExponent);
- Correct:
  float
  specular=specularIntensity*pow(max(0.0,dot(midDir,normalOut)),specularExponent);

- Also, DATA_FILE_LOCATION in CMakeLists.txt needs to be a directory. Cannot be an independent file.
- If DATA_FILE_LOCATION is ${CMAKE_CURRENT_SOURCE_DIR}/data, files will be copied as:

```
${CMAKE_CURRENT_SOURCE_DIR}/data          (Program Directory)
    vertex_shader.glsl                        vertex_shader.glsl
    fragment_shader.glsl                      fragment_shader.glsl
    something.png                             something.png
    datafile.dat                              datafile.dat
    data_sub_dir                              data_sub_dir
        more_data_file.txt                        more_data_file.txt
```

- Program Directory is:
  - The directory where .exe file exists in Windows.
  - Contents/Resources directory in the bundle in MacOSX.

## Billboarding

- A billboard is a pimitive that appears the same no matter what direction it is looked from.
- Example:
  - Trees in a walk-through application.
  - Annotation.
  - An icon floating in the 3D space.
- This is another feature frustratingly missing in the fixed-function pipeline.

## Billboading

- In other words, the center (or a reference point) of the billboard may move depending on the view point and orientation.
- But, the billboard will always face perpendicular to the view direction regardless of the view point and orientation.
- I.E., the shape is always the same in the camera’s local coordinate system.

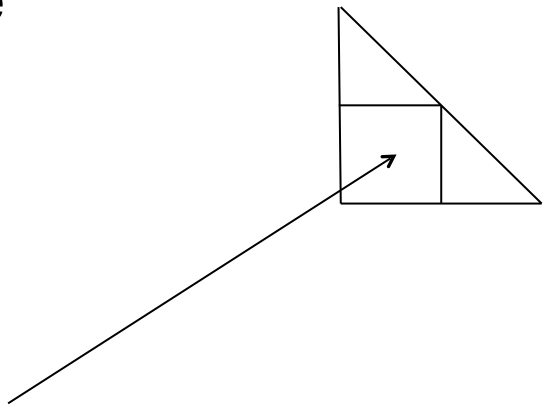
## Billboarding

- It can be achieved by adding offsets to the vertex position in the camera-coordinate system, or in the screen coordinate system.
- Offset in the camera coordinate system:
  – Offset applied after model-view matrix.
  – Will change the size of the primitive depending on the distance from the view point.
- Offset in the screen coordinate system:
  – Offset applied after model-view and projection matrices.
  – Appears as the same size regardless of the distance from the view point.

## Billboarding – Vertex Attributes

- Vertex attributes:
  - Position (x,y,z)
  - Offset (dx,dy)
  - Color
  - Texture Coordinate
- When you draw a triangle, you repeat same (x,y,z) three times, but use different offset vectors.

| Position | Offset |
|---|---|
| (x0,y0,z0) | (-1,-1) |
| (x0,y0,z0) | ( 3,-1) |
| (x0,y0,z0) | (-1, 3) |



# Billboading – Vertex Shader

```
attribute vec3 vertex;
attribute vec2 offset;
attribute vec4 color;
attribute vec2 texCoord;
uniform mat4 projection,modelView;
uniform float offsetInView;
uniform float offsetInPixel;
uniform float viewportWidth;
uniform float viewportHeight;
uniform sampler2D texture2d;
varying vec4 colorOut;
varying vec2 texCoordOut;

void main()
{
   vec4 pos=modelView*vec4(vertex,1.0);
   pos.xy=pos.xy+offsetInView*offset;
   pos=projection*pos;

   vec2 offsetInScreen;
   offsetInScreen.x=offsetInPixel*offset.x/(viewportWidth/2.0)/pos.w;
   offsetInScreen.y=offsetInPixel*offset.y/(viewportHeight/2.0)/pos.w;

   pos.xy=pos.xy+offsetInScreen;

   colorOut=color;
   texCoordOut=texCoord;

   gl_Position=pos;
}
```

Why division by w?

## Billboading –Fragment Shader

```
uniform sampler2D texture2d;

varying vec4 colorOut;
varying vec2 texCoordOut;

void main()
{
    if(0.0<texCoordOut.x && texCoordOut.x<1.0 &&
       0.0<texCoordOut.y && texCoordOut.y<1.0)
    {
        gl_FragColor=colorOut*texture(texture2d,texCoordOut);
    }
    else
    {
        discard;
    }
}
```

## Billboading – In renderer.h

```
class Billboard3dRenderer : public RendererBase
{
public:
    GLuint attribVertexPos;
    GLuint attribOffsetPos;
    GLuint attribColorPos;
    GLuint attribTexCoordPos;

    GLuint uniformProjectionPos;
    GLuint uniformModelViewPos;
    GLuint uniformOffsetInViewPos;
    GLuint uniformOffsetInPixelPos;
    GLuint uniformViewportWidthPos;
    GLuint uniformViewportHeightPos;
    GLuint uniformTexture2dPos;

    virtual void CacheAttributeAndUniformIdent(void);
};
```

## Billboading – In renderer.cpp

```
void Billboard3dRenderer::CacheAttributeAndUniformIdent(void)
{
    attribVertexPos=glGetAttribLocation(programIdent,"vertex");
    printf("attribVertexPos=%d\n",attribVertexPos);
    attribOffsetPos=glGetAttribLocation(programIdent,"offset");
    printf("attribOffsetPos=%d\n",attribOffsetPos);
    attribColorPos=glGetAttribLocation(programIdent,"color");
    printf("attribColorPos=%d\n",attribColorPos);
    attribTexCoordPos=glGetAttribLocation(programIdent,"texCoord");
    printf("attribTexCoordPos=%d\n",attribTexCoordPos);

    uniformProjectionPos=glGetUniformLocation(programIdent,"projection");
    printf("uniformProjectionPos=%d\n",uniformProjectionPos);
    uniformModelViewPos=glGetUniformLocation(programIdent,"modelView");
    printf("uniformModelViewPos=%d\n",uniformModelViewPos);
    uniformOffsetInViewPos=glGetUniformLocation(programIdent,"offsetInView");
    printf("uniformOffsetInViewPos=%d\n",uniformOffsetInViewPos);
    uniformOffsetInPixelPos=glGetUniformLocation(programIdent,"offsetInPixel");
    printf("uniformOffsetInPixelPos=%d\n",uniformOffsetInPixelPos);
    uniformViewportWidthPos=glGetUniformLocation(programIdent,"viewportWidth");
    printf("uniformViewportWidthPos=%d\n",uniformViewportWidthPos);
    uniformViewportHeightPos=glGetUniformLocation(programIdent,"viewportHeight");
    printf("uniformViewportHeightPos=%d\n",uniformViewportHeightPos);
    uniformTexture2dPos=glGetUniformLocation(programIdent,"texture2d");
    printf("uniformTexture2dPos=%d\n",uniformTexture2dPos);
}
```

## Billboading – In main.cpp, Initialize

```
billboard3d.CompileFile(
    "billboard3d_vertex_shader.glsl",
    "billboard3d_fragment_shader.glsl");
```

# Billboarding –Draw function

Drawing a wireframe-cube as a reference.

```
GLfloat cubeEdgeVtx[]=
{
   -10,-10,-10,  10,-10,-10,
    10,-10,-10,  10,-10, 10,
    10,-10, 10, -10,-10, 10,
   -10,-10, 10, -10,-10,-10,

   -10, 10,-10,  10, 10,-10,
    10, 10,-10,  10, 10, 10,
    10, 10, 10, -10, 10, 10,
   -10, 10, 10, -10, 10,-10,

   -10, 10,-10, -10,-10,-10,
    10, 10,-10,  10,-10,-10,
    10, 10, 10,  10,-10, 10,
   -10, 10, 10, -10,-10, 10,
};
GLfloat cubeEdgeColor[]=
{
   0,0,0,1, 0,0,0,1,
   0,0,0,1, 0,0,0,1,
   0,0,0,1, 0,0,0,1,
   0,0,0,1, 0,0,0,1,

   0,0,0,1, 0,0,0,1,
   0,0,0,1, 0,0,0,1,
   0,0,0,1, 0,0,0,1,
   0,0,0,1, 0,0,0,1,

   0,0,0,1, 0,0,0,1,
   0,0,0,1, 0,0,0,1,
   0,0,0,1, 0,0,0,1,
   0,0,0,1, 0,0,0,1,
};
```

```
glUseProgram(plain3d.programIdent);
glUniformMatrix4fv(plain3d.uniformProjectionPos,1,GL_FALSE,projMat);
glUniformMatrix4fv(plain3d.uniformModelViewPos,1,GL_FALSE,viewMat);

glEnableVertexAttribArray(plain3d.attribVertexPos);
glEnableVertexAttribArray(plain3d.attribColorPos);

glVertexAttribPointer(plain3d.attribVertexPos,3,GL_FLOAT,GL_FALSE,0,cubeEdgeVtx);
glVertexAttribPointer(plain3d.attribColorPos,4,GL_FLOAT,GL_FALSE,0,cubeEdgeColor);

glDrawArrays(GL_LINES,0,24);

glDisableVertexAttribArray(plain3d.attribVertexPos);
glDisableVertexAttribArray(plain3d.attribColorPos);
```

# Billboarding –Draw function

And then drawing a billboard.

```
GLfloat billboardVtx[]=
{
    -10,-10,-10,
    -10,-10,-10,
    -10,-10,-10,
};
GLfloat billboardOffset[]=
{
    -1.0,-1.0,
     3.0,-1.0,
    -1.0, 3.0,
};
GLfloat billboardTexCoord[]=
{
    -1.0,-1.0,
     3.0,-1.0,
    -1.0, 3.0,
};
GLfloat billboardColor[]=
{
    0,0,0,1,
    0,0,0,1,
    0,0,0,1,
};
```

```
glUseProgram(billboard3d.programIdent);
glUniformMatrix4fv(billboard3d.uniformProjectionPos,1,GL_FALSE,projMat);
glUniformMatrix4fv(billboard3d.uniformModelViewPos,1,GL_FALSE,viewMat);

glUniform1f(billboard3d.uniformOffsetInViewPos,1.0f);
glUniform1f(billboard3d.uniformOffsetInPixelPos,0.0f);
glUniform1f(billboard3d.uniformViewportWidthPos,(GLfloat)wid);
glUniform1f(billboard3d.uniformViewportHeightPos,(GLfloat)hei);

glEnableVertexAttribArray(billboard3d.attribVertexPos);
glEnableVertexAttribArray(billboard3d.attribColorPos);
glEnableVertexAttribArray(billboard3d.attribTexCoordPos);
glEnableVertexAttribArray(billboard3d.attribOffsetPos);

glVertexAttribPointer(
    billboard3d.attribVertexPos,3,GL_FLOAT,GL_FALSE,0,billboardVtx);
glVertexAttribPointer(
    billboard3d.attribColorPos,4,GL_FLOAT,GL_FALSE,0,billboardColor);
glVertexAttribPointer(
    billboard3d.attribTexCoordPos,2,GL_FLOAT,GL_FALSE,0,billboardTexCoord);
glVertexAttribPointer(
    billboard3d.attribOffsetPos,2,GL_FLOAT,GL_FALSE,0,billboardTexCoord);

glDrawArrays(GL_TRIANGLES,0,3);

glDisableVertexAttribArray(billboard3d.attribVertexPos);
glDisableVertexAttribArray(billboard3d.attribColorPos);
glDisableVertexAttribArray(billboard3d.attribTexCoordPos);
glDisableVertexAttribArray(billboard3d.attribOffsetPos);
```

## Adding a texture.

- Copy BlueImpulse.png in the data directory.
- Add a member variable:

```
GLuint textureIdent;
```

- Add YsBitmap in the LIB_DEPENDENCY in CMakeLists.txt
- #include <ysbitmap.h>

## Adding a texture

- In Initialize() function,

```
glGenTextures(1,&textureIdent);
YsBitmap bmp;
if(YSOK==bmp.LoadPng("BlueImpulse.png"))
{
    printf("Texture Loaded.\n");
    glActiveTexture(GL_TEXTURE0);
    glBindTexture(GL_TEXTURE_2D,textureIdent);
    glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_WRAP_S,GL_CLAMP);
    glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_WRAP_T,GL_CLAMP);
    glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_NEAREST);
    glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_NEAREST);
    glTexImage2D(GL_TEXTURE_2D,0,GL_RGBA,
        bmp.GetWidth(),bmp.GetHeight(),0,
        GL_RGBA,GL_UNSIGNED_BYTE,bmp.GetRGBABitmapPointer());
}
```

## Adding a texture

- In Draw() function,

```
glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_2D,textureIdent);
glUniform1i(billboard3d.uniformTexture2dPos,0);
```

- Also change the billboard color from 0,0,0,1 to 1,1,1,1.

## More efficient way of drawing a billboard

- Using GL_POINTS.
- Need to control point-size, and
- Paste texture on points.

## Program Point Size

- Another limitation of older version OpenGL: Cannot specify point-size per vertex without closing a primitive with glEnd().
- You can control point sizes from the vertex shader.

(*) For XCode, the following macro definition is needed.

```
#if !defined(GL_PROGRAM_POINT_SIZE) && defined(GL_PROGRAM_POINT_SIZE_EXT)
#define GL_PROGRAM_POINT_SIZE GL_PROGRAM_POINT_SIZE_EXT
#endif
```

## Program Point Size

- You need to enable GL_PROGRAM_POINT_SIZE to use this functionality.  (In OpenGL ES, this functionality is always on.  Instead, there is no function called glPointSize.)
- Also, if you want to control a point size per vertex, you need an additional attribute.

## Program Point Size

- Vertex Shader

```
attribute vec3 vertex;
attribute vec4 color;
attribute float pointSize;

uniform mat4 projection,modelView;

varying vec4 colorOut;

void main()
{
   colorOut=color;
   gl_Position=projection*modelView*vec4(vertex,1.0);
   gl_PointSize=pointSize;
}
```

- Fragment Shader

```
varying vec4 colorOut;
void main()
{
   gl_FragColor=colorOut;
}
```

## Program Point Size

- In renderer.h

```
class ProgramPointSize3dRenderer : public RendererBase
{
public:
    GLuint attribVertexPos;
    GLuint attribColorPos;
    GLuint attribPointSizePos;
    GLuint uniformProjectionPos;
    GLuint uniformModelViewPos;
    virtual void CacheAttributeAndUniformIdent(void);
};
```

- In renderer.cpp

```
void ProgramPointSize3dRenderer::CacheAttributeAndUniformIdent(void)
{
    attribVertexPos=glGetAttribLocation(programIdent,"vertex");
    printf("attribVertexPos=%d\n",attribVertexPos);
    attribColorPos=glGetAttribLocation(programIdent,"color");
    printf("attribColorPos=%d\n",attribColorPos);
    attribPointSizePos=glGetAttribLocation(programIdent,"pointSize");
    printf("attribPointSizePos=%d\n",attribPointSizePos);
    uniformProjectionPos=glGetUniformLocation(programIdent,"projection");
    printf("uniformProjectionPos=%d\n",uniformProjectionPos);
    uniformModelViewPos=glGetUniformLocation(programIdent,"modelView");
    printf("uniformModelViewPos=%d\n",uniformModelViewPos);
}
```

## Program Point Size

- Additional member variable

```
ProgramPointSize3dRenderer programPointSize;
```

- In Initialize()

```
programPointSize.CompileFile(
    "program_point_size_vertex_shader.glsl",
    "program_point_size_fragment_shader.glsl");
```

## Program Point Size

- In Draw

```
GLfloat quadVtx[12]=
   {-10,-10,-10,   10,-10,-10,   10, 10,-10,  -10, 10,-10,};
GLfloat quadCol[16]=
   {1,0,0,1,  0,1,0,1,  0,0,1,1,  1,0,1,1};
GLfloat quadPointSize[4]=
   {8,16,24,32};
glUseProgram(programPointSize.programIdent);
glUniformMatrix4fv(programPointSize.uniformProjectionPos,1,GL_FALSE,projMat);
glUniformMatrix4fv(programPointSize.uniformModelViewPos,1,GL_FALSE,viewMat);

glEnableVertexAttribArray(programPointSize.attribVertexPos);
glEnableVertexAttribArray(programPointSize.attribColorPos);
glEnableVertexAttribArray(programPointSize.attribPointSizePos);

glVertexAttribPointer(programPointSize.attribVertexPos,3,GL_FLOAT,GL_FALSE,0,quadVtx);
glVertexAttribPointer(programPointSize.attribColorPos,4,GL_FLOAT,GL_FALSE,0,quadCol);
glVertexAttribPointer(programPointSize.attribPointSizePos,1,GL_FLOAT,GL_FALSE,0,quadPointSize);

glEnable(GL_PROGRAM_POINT_SIZE);
glDrawArrays(GL_POINTS,0,4);
glDisable(GL_PROGRAM_POINT_SIZE);

glDisableVertexAttribArray(programPointSize.attribVertexPos);
glDisableVertexAttribArray(programPointSize.attribColorPos);
```

## Program Point Size – Point-Size in 3D space

- Problem: gl_PointSize is in pixels. What if I want to say the point size must be 1.0x1.0 not in pixels, but in the size in the 3D space?
- Need to calculate point size in the vertex shader.
- To be able to deal with different kinds of projections, the easiest way is:
  1. Transform the vertex position by the model-view matrix, lets' call it *s*.
  2. Add pointSize/2 to s.y.
  3. Transform *s* with the projection matrix.
  4. Calculate difference between gl_Position.y/gl_Position.w and s.y/s.w, let's call it *h*.
  5. Multiply *h* by view-port height. Then, *h* is the point size in pixels.

## Program Point Size – Point-Size in 3D space

- Vertex Shader

```
#version 120
attribute vec3 vertex;
attribute vec4 color;
attribute float pointSize;

uniform mat4 projection,modelView;
uniform float sizeInPixel;
uniform float sizeIn3d;
uniform float viewportHeight;

varying vec4 colorOut;

void main()
{
    colorOut=color;
    gl_Position=projection*modelView*vec4(vertex,1.0);

    vec4 shift;
    shift=modelView*vec4(vertex,1.0);
    shift.y=shift.y+pointSize/2;
    shift=projection*shift;

    float heightInScreen,heightInPixel;
    heightInScreen=abs(shift.y/shift.w-gl_Position.y/gl_Position.w);
    heightInPixel=sizeIn3d*viewportHeight*heightInScreen;

    gl_PointSize=pointSize*sizeInPixel+heightInPixel;
}
```

Need #version 120, or your Xcode may complain division of int and float.

sizeInPiexl=1, sizeIn3d=0 :
   pointSize is in pixels.
sizeInPiexl=0, sizeIn3d=1 :
   pointSize is in 3D space.

Viewport height is needed for calculate number of pixels.

## Program Point Size – Point-Size in 3D space

- In renderer.h

```
class ProgramPointSize3dRenderer : public RendererBase
{
public:
    GLuint attribVertexPos;
    GLuint attribColorPos;
    GLuint attribPointSizePos;
    GLuint uniformProjectionPos;
    GLuint uniformModelViewPos;
    GLuint uniformSizeInPixelPos;
    GLuint uniformSizeIn3dPos;
    GLuint uniformViewportHeightPos;

    virtual void CacheAttributeAndUniformIdent(void);
};
```

## Program Point Size – Point-Size in 3D space

- In renderer.cpp

```
void ProgramPointSize3dRenderer::CacheAttributeAndUniformIdent(void)
{
    attribVertexPos=glGetAttribLocation(programIdent,"vertex");
    printf("attribVertexPos=%d\n",attribVertexPos);
    attribColorPos=glGetAttribLocation(programIdent,"color");
    printf("attribColorPos=%d\n",attribColorPos);
    attribPointSizePos=glGetAttribLocation(programIdent,"pointSize");
    printf("attribPointSizePos=%d\n",attribPointSizePos);
    uniformProjectionPos=glGetUniformLocation(programIdent,"projection");
    printf("uniformProjectionPos=%d\n",uniformProjectionPos);
    uniformModelViewPos=glGetUniformLocation(programIdent,"modelView");
    printf("uniformModelViewPos=%d\n",uniformModelViewPos);
    uniformSizeInPixelPos=glGetUniformLocation(programIdent,"sizeInPixel");
    printf("uniformSizeInPixelPos=%d\n",uniformSizeInPixelPos);
    uniformSizeIn3dPos=glGetUniformLocation(programIdent,"sizeIn3d");
    printf("uniformSizeIn3dPos=%d\n",uniformSizeIn3dPos);
    uniformViewportHeightPos=glGetUniformLocation(programIdent,"viewportHeight");
    printf("uniformViewportHeightPos=%d\n",uniformViewportHeightPos);
}
```

## Program Point Size – Point-Size in 3D space

- In Draw()

```
GLfloat quadVtx[12]={ -10,-10,-10, 10,-10,-10, 10, 10,-10, -10, 10,-10,};
GLfloat quadCol[16]={ 1,0,0,1, 0,1,0,1, 0,0,1,1, 1,0,1,1};
GLfloat quadPointSize[4]={1,1,2,2};

glUseProgram(programPointSize.programIdent);
glUniformMatrix4fv(programPointSize.uniformProjectionPos,1,GL_FALSE,projMat);
glUniformMatrix4fv(programPointSize.uniformModelViewPos,1,GL_FALSE,viewMat);

glUniform1f(programPointSize.uniformSizeInPixelPos,0.0f);
glUniform1f(programPointSize.uniformSizeIn3dPos,1.0f);
glUniform1f(programPointSize.uniformViewportHeightPos,(float)hei);

glEnableVertexAttribArray(programPointSize.attribVertexPos);
glEnableVertexAttribArray(programPointSize.attribColorPos);
glEnableVertexAttribArray(programPointSize.attribPointSizePos);

glVertexAttribPointer(programPointSize.attribVertexPos,3,GL_FLOAT,GL_FALSE,0,quadVtx);
glVertexAttribPointer(programPointSize.attribColorPos,4,GL_FLOAT,GL_FALSE,0,quadCol);
glVertexAttribPointer(programPointSize.attribPointSizePos,1,GL_FLOAT,GL_FALSE,0,quadPointSize);

glEnable(GL_PROGRAM_POINT_SIZE);
glDrawArrays(GL_POINTS,0,4);
glDisable(GL_PROGRAM_POINT_SIZE);

glDisableVertexAttribArray(programPointSize.attribVertexPos);
glDisableVertexAttribArray(programPointSize.attribColorPos);
```

## Point Sprite

- This is also loooooong wanted feature that was missing in OpenGL 1.1.
- With point sprite, you can paste textures on GL_POINTS.
- Even you can paste a part of a texture.
- By making a texture atlas, you can draw different patterns for different points in one glDrawArrays call.

## Point Sprite

- In the fragment shader, coordinate within a point is given as gl_PointCoord.
- Need GLSL version 120.
- Let's first use coordinate as R and G values to see what we get.

## Point Sprite

- Vertex shader is same as the one from program_point_size_in_3d example.
- Fragment Shader

```
#version 120
varying vec4 colorOut;
void main()
{
    gl_FragColor=vec4(gl_PointCoord.xy,0,1);
}
```

- In renderer.h

```
class PointSprite3dRenderer : public RendererBase
{
public:
    GLuint attribVertexPos;
    GLuint attribColorPos;
    GLuint attribPointSizePos;
    GLuint uniformProjectionPos;
    GLuint uniformModelViewPos;
    GLuint uniformSizeInPixelPos;
    GLuint uniformSizeIn3dPos;
    GLuint uniformViewportHeightPos;
    virtual void CacheAttributeAndUniformIdent(void);
};
```

## Point Sprite

- In renderer.cpp

```
void PointSprite3dRenderer::CacheAttributeAndUniformIdent(void)
{
    attribVertexPos=glGetAttribLocation(programIdent,"vertex");
    printf("attribVertexPos=%d\n",attribVertexPos);
    attribColorPos=glGetAttribLocation(programIdent,"color");
    printf("attribColorPos=%d\n",attribColorPos);
    attribPointSizePos=glGetAttribLocation(programIdent,"pointSize");
    printf("attribPointSizePos=%d\n",attribPointSizePos);
    uniformProjectionPos=glGetUniformLocation(programIdent,"projection");
    printf("uniformProjectionPos=%d\n",uniformProjectionPos);
    uniformModelViewPos=glGetUniformLocation(programIdent,"modelView");
    printf("uniformModelViewPos=%d\n",uniformModelViewPos);
    uniformSizeInPixelPos=glGetUniformLocation(programIdent,"sizeInPixel");
    printf("uniformSizeInPixelPos=%d\n",uniformSizeInPixelPos);
    uniformSizeIn3dPos=glGetUniformLocation(programIdent,"sizeIn3d");
    printf("uniformSizeIn3dPos=%d\n",uniformSizeIn3dPos);
    uniformViewportHeightPos=glGetUniformLocation(programIdent,"viewportHeight");
    printf("uniformViewportHeightPos=%d\n",uniformViewportHeightPos);
}
```

## Point Sprite

- Add a member variable:

```
PointSprite3dRenderer pointSprite;
```

- In Initialize():

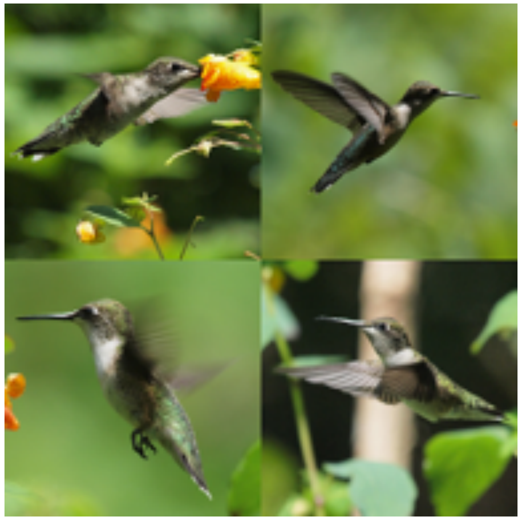
```
pointSprite.CompileFile(
    "pointsprite_vertex_shader.glsl",
    "pointsprite_fragment_shader.glsl");
```

## Point Sprite

- In Draw(), replace programPointSize with pointSprite.

## Point Sprite with Texture Atlas

- Goal: Want to draw multiple patterns within one glDrawArrays call.
- Problem: Typical OpenGL can use up to two or three textures at a time.
- Solution: TextureAtlas put multiple patterns in one texture image.



## Point Sprite with Texture

- Additional uniform:  sampler2D texture;
- Additional vertex attribute: vec4 texCoordRange;
- Additional varying: vec4 texCoordRangeOut;

- Vertex shader pass texCoordRange to texCoordRangeOut.
- Fragment shader interpolates between texCoordRange.xy and texCoordRange.zw.

## Point Sprite with Texture

- Vertex Shader

```
#version 120
attribute vec3 vertex;
attribute vec4 color;
attribute float pointSize;
attribute vec4 texCoordRange;
uniform mat4 projection,modelView;
uniform float sizeInPixel;
uniform float sizeIn3d;
uniform float viewportHeight;
uniform sampler2D texture;
varying vec4 colorOut;
varying vec4 texCoordRangeOut;

void main()
{
    colorOut=color;
    texCoordRangeOut=texCoordRange;
    gl_Position=projection*modelView*vec4(vertex,1.0);

    vec4 shift;
    shift=modelView*vec4(vertex,1.0);
    shift.y=shift.y+pointSize/2;
    shift=projection*shift;
    float heightInScreen,heightInPixel;
    heightInScreen=abs(shift.y/shift.w-gl_Position.y/gl_Position.w);
    heightInPixel=sizeIn3d*viewportHeight*heightInScreen;

    gl_PointSize=pointSize*sizeInPixel+heightInPixel;
}
```

## Point Sprite with Texture

- Fragment Shader

```
#version 120
uniform sampler2D texture;

varying vec4 colorOut;
varying vec4 texCoordRangeOut;

void main()
{
    vec2 texCoord;
    texCoord.x=texCoordRangeOut[0]*(1.0-gl_PointCoord.x)
            +texCoordRangeOut[2]*gl_PointCoord.x;
    texCoord.y=texCoordRangeOut[1]*(1.0-gl_PointCoord.y)
            +texCoordRangeOut[3]*gl_PointCoord.y;

    gl_FragColor=colorOut*texture2D(texture,texCoord);
}
```

## Point Sprite with Texture

- In Initialize() function

```
glGenTextures(1,&textureIdent);
YsBitmap bmp;
if(YSOK==bmp.LoadPng("Hummingbird.png"))
{
    printf("Texture Loaded.\n");
    glActiveTexture(GL_TEXTURE0);
    glBindTexture(GL_TEXTURE_2D,textureIdent);
    glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_WRAP_S,GL_CLAMP);
    glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_WRAP_T,GL_CLAMP);
    glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_NEAREST);
    glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_NEAREST);
    glTexImage2D(GL_TEXTURE_2D,0,GL_RGBA,bmp.GetWidth(),bmp.GetHeight(),0,
        GL_RGBA,GL_UNSIGNED_BYTE,bmp.GetRGBABitmapPointer());
}
```

# Point Sprite with Texture

- In Draw() function:

```
GLfloat quadVtx[12]=   {-10,-10,-10, 10,-10,-10, 10, 10,-10,-10, 10,-10,};
GLfloat quadCol[16]=   {1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1    };
GLfloat quadPointSize[4]={1,1,2,2};
GLfloat cubeTexCoordRange[]={
   0.0f,0.5f,  0.5f,0.0f,
   0.5f,0.5f,  1.0f,0.0f,
   0.0f,1.0f,  0.5f,0.5f,
   0.5f,1.0f,  1.0f,0.5f,
};
glUseProgram(pointSprite.programIdent);
glUniformMatrix4fv(pointSprite.uniformProjectionPos,1,GL_FALSE,projMat);
glUniformMatrix4fv(pointSprite.uniformModelViewPos,1,GL_FALSE,viewMat);

glUniform1f(pointSprite.uniformSizeInPixelPos,0.0f);
glUniform1f(pointSprite.uniformSizeIn3dPos,1.0f);
glUniform1f(pointSprite.uniformViewportHeightPos,(float)hei);
```

## Point Sprite with Texture

- In Draw() function (continued):

```
glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_2D,textureIdent);
glUniform1f(pointSprite.uniformTexture2dPos,0);

glEnableVertexAttribArray(pointSprite.attribVertexPos);
glEnableVertexAttribArray(pointSprite.attribColorPos);
glEnableVertexAttribArray(pointSprite.attribPointSizePos);
glEnableVertexAttribArray(pointSprite.attribTexCoordRangePos);

glVertexAttribPointer(pointSprite.attribVertexPos,3,GL_FLOAT,GL_FALSE,0,quadVtx);
glVertexAttribPointer(pointSprite.attribColorPos,4,GL_FLOAT,GL_FALSE,0,quadCol);
glVertexAttribPointer(pointSprite.attribPointSizePos,1,GL_FLOAT,GL_FALSE,0,quadPointSize);
glVertexAttribPointer(pointSprite.attribTexCoordRangePos,4,GL_FLOAT,GL_FALSE,0,cubeTexCoordRange);

glEnable(GL_POINT_SPRITE);
glEnable(GL_PROGRAM_POINT_SIZE);
glDrawArrays(GL_POINTS,0,4);
glDisable(GL_PROGRAM_POINT_SIZE);
glDisable(GL_POINT_SPRITE);

glDisableVertexAttribArray(pointSprite.attribVertexPos);
glDisableVertexAttribArray(pointSprite.attribColorPos);
glDisableVertexAttribArray(pointSprite.attribTexCoordRangePos);
```