CS248: Interactive Computer Graphics

OpenGL Particle Rendering
The best (and easiest) way to render particles in OpenGL is point sprites.

You give OpenGL some points (using `GL_POINTS` rather than `GL_TRIANGLES`).

OpenGL automatically generates small bilboards.

Enabling point sprites:

```c
glEnable (GL_POINT_SPRITE)
```
Point Sprite Size

- Can be set from the vertex shader

- `glEnable(GL_VERTEX_PROGRAM_POINT_SIZE)`

- Then, set the special built-in “gl_PointSize” in the vertex shader

- `gl_PointSize = size of particle in screen units` (i.e., in pixels)

- So, you need to scale by the **viewport width**. You also need to scale by **distance from camera** (length of view vector).

- `gl_PointSize = baseSize * viewportWidth / length(eyePosition.xyz)`
Transparency

- `glEnable(GL_BLEND)`

- Disable depth writes, so that particles don’t block each other: `glDepthMask(GL_FALSE)`. Render particles last!

- Set the blend function: `glBlendFunc(src, dest)`. When blending, `color = src * srcColor + dest * destColor`.

- Smoke and dust (one option)
  - `src = GL_SRC_SLPHA`
  - `dest = GL_ONE_MINUS_SRC_ALPHA`

- Fire, plasma, or energetic particles
  - `src = GL_SRC_ALPHA`
  - `dest = GL_ONE`
Good news: OpenGL generates point sprite texture coordinates for you!

```c
glActiveTexture(GL_TEXTURE0);
sf::Image::Bind(); or glBindTexture(id);
glTexEnvi(GL_POINT_SPRITE, GL_COORD_REPLACE, 1);
```

Then, just use `gl_TexCoord[0]` as the texture coordinate in your fragment shader. Simple!

Still have to pass the texture to your fragment shader.

Use “splat” textures, i.e., alpha fades out from the center.
Point Sprite Shader Attributes

- Each particle basically looks like this C struct:

```c
struct Particle {
    Vector3 position;
    float life;
    ...maybe other parameters for cool effects
};
```

- You can pass this data to your shaders using `glVertexAttribPointer` just like you did in Assignment 4 for the tangents, normals, and positions.

- Don’t forget these guys: `glGetAttribLocation`, `glEnable/DisableVertexAttribArray`, `glUniform`, `glGetUniform`, `glDrawArrays`
Particle Rendering Hints

- For any `glEnable(x)` call you make before calling `glDrawArrays()`, be sure you call `glDisable(x)` when you’re done rendering the particles.

- Otherwise, weird stuff will happen to the rest of your rendering.

- Try to make your renderer general, so that there is one block of rendering code for all your particle system types.