Accelerated movement

How to make your animations fly

By: Alexander Vestin and Johan Fallström

Layout

- Problem
- Our solution
- Features / Limitations
- Demo

Problem

- Animations are done on the client
- Can become tedious
- Not scalable

Our solution

- Animation is done on the GPU
- Enables parallelization of the workload
- Decreases redundancy
- Made with customization in mind

Our solution – Structure

- AnimationShader
  - Loads our custom vertex and fragment shaders
  - Lets you customize the lighting and shading type
  - Enables code insertion into the shaders
- Geometry
  - Holds model data and animation buffers
- Animation buffers
  - Array of attributes sent to vertex shader
  - Holds start/end position, size, color, start time and quantity

Our solution – Code example

```
AnimationShader* as = new AnimationShader(“pos.x += t / 1000.;");
Geometry* g = new Geometry{ "teapot.obj", as->getProgram() };

// Buffer attributes
const int numInstances = 20;
std::vector<GLfloat> startPositions(numInstances * 3);
std::vector<GLfloat> endPositions(numInstances * 3);
std::vector<GLfloat> startTimes(numInstances);
std::vector<GLfloat> sizes(numInstances);
std::vector<GLfloat> colors(numInstances * 3);

// How to upload the buffer
g->setUpInstanceBuffers(buffers);

// How to draw the geometry
g->draw(t, trans.m, Camera::getMatrix().m, &Camera::pos.x, GL_TRIANGLES);
```

Features

- Can easily switch between Flat, Lambert and Phong shading
- Choose between Point, Line and Triangle rendering
- Customizable with code insertion
- Animation on the GPU makes it fast and scalable
- Simple, declarative API

Limitations

- Less control over animations
- Harder to update values from client side
- Not a fully equipped API

Time for a demo

Accelerated movement

How to make your animations fly

By: Alexander Vestin and Johan Fallström