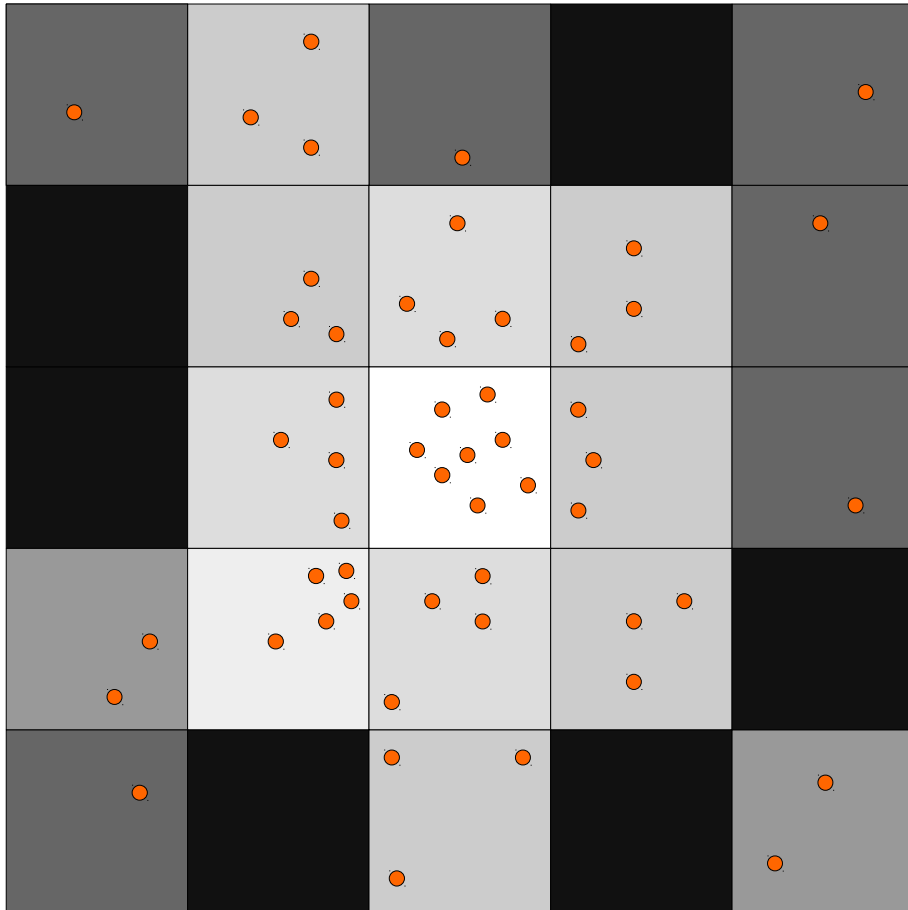


Лекция 14¹⁾

Текстуры.

- Использование текстур в компьютерном моделировании.
- uniform переменные.
- Атомарные функции.

1) В лекции используется материал магистерской диссертации А.В. Нужнова



```
GLuint genInitProg();  
GLuint genMoveProg();
```

```
.....  
i=int((Pos[index*6]-xmin)/hx);\n j=int((Pos[index*6+1]-xmin)/hy);\n atomicAdd(Grid[j+i*M],1);\n.....
```

```
GLint GenGrid2TexProg();
```

```
.....  
color=log2(1.0+float(Grid[index]))/3.0;\n Tex[index*4]=color;\n Tex[index*4+1]=color;\n Tex[index*4+2]=color;\n Tex[index*4+3]=1.0;\n.....
```





```
int main(){  
    initGL();  
    initBuffers();
```

```
.....  
  
do{
```

```
.....  
    display();  
    hMove();  
    glfwSwapBuffers(window);  
.....
```

```
void initBuffers(){  
    genBuffers();  
    genTexture();  
    initMapBuffer();  
    initTexBuffer();  
}
```

```
enum bufferNames{POSITIONS, VELOCITIES, GRID,TEXTURE, MAP,  
                  NUM_OF_BUFFERS};
```

```
const int L=128,M=128;
```

```
const int num_of_verticies=L*M;
```

```
void genBuffers(){  
    glGenBuffers(NUM_OF_BUFFERS, bufferID);  
    glBindBuffer( GL_ARRAY_BUFFER, bufferID[POSITIONS]);  
    glBufferData( GL_ARRAY_BUFFER, 6*num_of_verticies*sizeof(float),  
                  0, GL_DYNAMIC_DRAW );  
    glBindBuffer( GL_ARRAY_BUFFER, bufferID[VELOCITIES]);  
    glBufferData( GL_ARRAY_BUFFER, 3*num_of_verticies*sizeof(float),  
                  0, GL_DYNAMIC_DRAW );  
    glBindBuffer(GL_PIXEL_UNPACK_BUFFER, bufferID[TEXTURE]);  
    glBufferData(GL_PIXEL_UNPACK_BUFFER , 4*L*M*sizeof(float),  
                  0, GL_DYNAMIC_DRAW);  
    glBindBuffer(GL_ARRAY_BUFFER, bufferID[GRID]);  
    glBufferData(GL_ARRAY_BUFFER, L*M*sizeof(uint),  
                  0, GL_DYNAMIC_DRAW);  
}
```

GLuint genTexture(){

GLuint texHandle;

```
glGenTextures(1, &texHandle);
```

```
glBindTexture(GL_TEXTURE_2D, texHandle);
```

```
glTexStorage2D(GL_TEXTURE_2D, 1, GL_RGBA8, L, M);
```

```
glBindBuffer(GL_PIXEL_UNPACK_BUFFER, bufferID[TEXTURE]);
```

glTexSubImage2D(GL_TEXTURE_2D,

0,

0, 0,

L, M,

GL_RGBA, GL_FLOAT,

(Glvaid*)NULL

);

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,  
                GL_LINEAR); //GL_NEAREST);  
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,  
                GL_LINEAR); //GL_NEAREST);  
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,  
                GL_CLAMP_TO_EDGE);  
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T,  
                GL_CLAMP_TO_EDGE);  
  
glGenerateMipmap(GL_TEXTURE_2D);  
  
checkErrors("Gen texture");  
  
return texHandle;  
};
```



```
void initMapBuffer(){  
    static const GLfloat tex_map[] = {  
        0.75f, -0.75f,  
        -0.75f, -0.75f,  
        -0.75f, 0.75f,  
        0.75f, 0.75f,  
  
        0.0f, 0.0f,  
        1.0f, 0.0f,  
        1.0f, 1.0f,  
        0.0f, 1.0f  
    };  
    glBindBuffer(GL_ARRAY_BUFFER, bufferID[MAP]);  
    glBufferData(GL_ARRAY_BUFFER,  
        sizeof(tex_map),  
        tex_map, GL_STATIC_DRAW);  
}
```

```
void initTexBuffer(){
    csDataInit(bufferID, L*M); //запускают вычислительные шейдеры для
    csGrid2Tex(bufferID, L*M); //инициализации буфера текстуры
}
```

```
void csDataInit(GLuint* inBuf,int N){
    glBindBufferBase(GL_SHADER_STORAGE_BUFFER, 0,
                    inBuf[POSITIONS]);
    glBindBufferBase(GL_SHADER_STORAGE_BUFFER, 1,
                    inBuf[VELOCITIES]);
    glBindBufferBase(GL_SHADER_STORAGE_BUFFER, 2, inBuf[GRID]);

    GLuint computeShaderID=genInitProg();
    glUseProgram(computeShaderID); // Связывание вычислительного
                                   // шейдера
    GLuint loc = glGetUniformLocation(computeShaderID,"L");
    glUniform1i(loc, L);
    loc = glGetUniformLocation(computeShaderID,"M");
    glUniform1i(loc, M);
    glDispatchCompute(N/128, 1, 1); // Выполнение вычислительного
                                   //шейдера со 128 рабочими группами (workgroups)
    glMemoryBarrier( GL_SHADER_STORAGE_BARRIER_BIT);
}
```

```
"#version 430\n",  
"layout (local_size_x = 128, local_size_y = 1, local_size_z = 1) in; \  
layout(std430, binding = 0) buffer PositionBuffer{float Pos[];};\  
layout(std430, binding = 1) buffer VelocityBuffer{float Vel[];};\  
layout(std430, binding = 2) buffer GridBuffer{int Grid[];};\  
uniform int L,M;  
float lmap(in uint i){  
    uint count;  
    float x=0.78;  
    for(count=0;count<i;count++)\  
        x=3.99*x*(1-x);\  
    return x;  
}
```

```
void main() {\n  uint index = gl_GlobalInvocationID.x;\n  float hx=1.5/L;\n  float hy=1.5/M;\n  int i,j;\n  Pos[index*6]=-0.5+1.0*Imap(index);\n  Pos[index*6+1]=-0.5+1.0*Imap(index*10);\n  Pos[index*6+2]=0.0;\n  Pos[index*6+3]=0.0;\n  Pos[index*6+4]=0.0;\n  Pos[index*6+5]=0.0;\n  Vel[3*index]=-0.5+1.0*Imap(index);\n  Vel[3*index+1]=-0.5+1.0*Imap(index*10);\n  Vel[3*index+2]=0.0;\n  Grid[index]=0;\n  barrier();\n  i=int((Pos[index*6]+0.75)/hx);\n  j=int((Pos[index*6+1]+0.75)/hy);\n  atomicAdd(Grid[j+i*M],1);\n}"
```

```
void csGrid2Tex(GLuint* inBuf,int N){  
  
    glBindBufferBase(GL_SHADER_STORAGE_BUFFER, 0, inBuf[GRID]);  
    glBindBufferBase(GL_SHADER_STORAGE_BUFFER, 1,  
                                                              inBuf[TEXTURE]);  
  
    GLuint computeShaderID=genGrid2TexProg();  
    glUseProgram(computeShaderID);  
  
    glDispatchCompute(N/128, 1, 1);  
  
    glMemoryBarrier( GL_SHADER_STORAGE_BARRIER_BIT);  
}
```

```
"#version 430\n",  
"layout (local_size_x = 128, local_size_y = 1, local_size_z = 1) in;\nlayout(std430, binding = 0) buffer GridBuffer{uint Grid[]};\nlayout(std430, binding = 1) buffer TexBuffer{float Tex[]};\nvoid main() {\n    float color;\n    uint index = gl_GlobalInvocationID.x;\n    color=log2(1.0+float(Grid[index]))/5.0;\n    Tex[index*4]=color;\n    Tex[index*4+1]=color;\n    Tex[index*4+2]=color;\n    Tex[index*4+3]=1.0;\n}"
```

```
void hMove(){  
    csMove(bufferID, num_of_verticies);  
    csGrid2Tex(bufferID, num_of_verticies);  
    genTexture();  
}
```

```
void csMove(GLuint* inBuf,int N){  
    glBindBufferBase(GL_SHADER_STORAGE_BUFFER, 0,  
                                                              inBuf[POSITIONS]);  
    glBindBufferBase(GL_SHADER_STORAGE_BUFFER, 1,  
                                                              inBuf[VELOCITIES]);  
    glBindBufferBase(GL_SHADER_STORAGE_BUFFER, 2, inBuf[GRID]);  
  
    GLuint computeShaderID=genMoveProg();  
    glUseProgram(computeShaderID);  
  
    GLuint loc = glGetUniformLocation(computeShaderID,"L");  
    glUniform1i(loc, L);  
    loc = glGetUniformLocation(computeShaderID,"M");  
    glUniform1i(loc, M);  
  
    glDispatchCompute(N/128, 1, 1);  
    glMemoryBarrier( GL_SHADER_STORAGE_BARRIER_BIT);  
}
```

```
"#version 430\n",  
"layout (local_size_x = 128, local_size_y = 1, local_size_z = 1) in; \  
layout(std430, binding = 0) buffer PositionBuffer{float Pos[];};\  
layout(std430, binding = 1) buffer VelocityBuffer{float Vel[];};\  
layout(std430, binding = 2) buffer GridBuffer{int Grid[];};\  
uniform int L,M;  
void main() {\  
    float x,y,vx,vy;\  
    float tau=0.01;\  
    float eps=0.5;\  
    float hx=1.5/L;\  
    float hy=1.5/M;\  
    int i,j;  
    uint index = gl_GlobalInvocationID.x;
```



```
x=Pos[index*6];\  
y=Pos[index*6+1];\  
vx=Vel[3*index];\  
vy=Vel[3*index+1];\  
\  
vx=vx+tau*10.0*(-x-eps*(2*x*y));\  
vy=vy+tau*10.0*(-y-eps*(x*x-y*y));\  
\  
x=x+tau*vx;\  
y=y+tau*vy;\  
\  
Pos[index*6]=x;\  
Pos[index*6+1]=y;\  
Vel[index*3]=vx;\  
Vel[index*3+1]=vy;\  
Grid[index]=0;\  
barrier(); \  
i=int((Pos[index*6]+0.75)/hx);\  
j=int((Pos[index*6+1]+0.75)/hy);\  
atomicAdd(Grid[j+i*M],1); \  

```

```
}"
```

Спасибо за внимание!