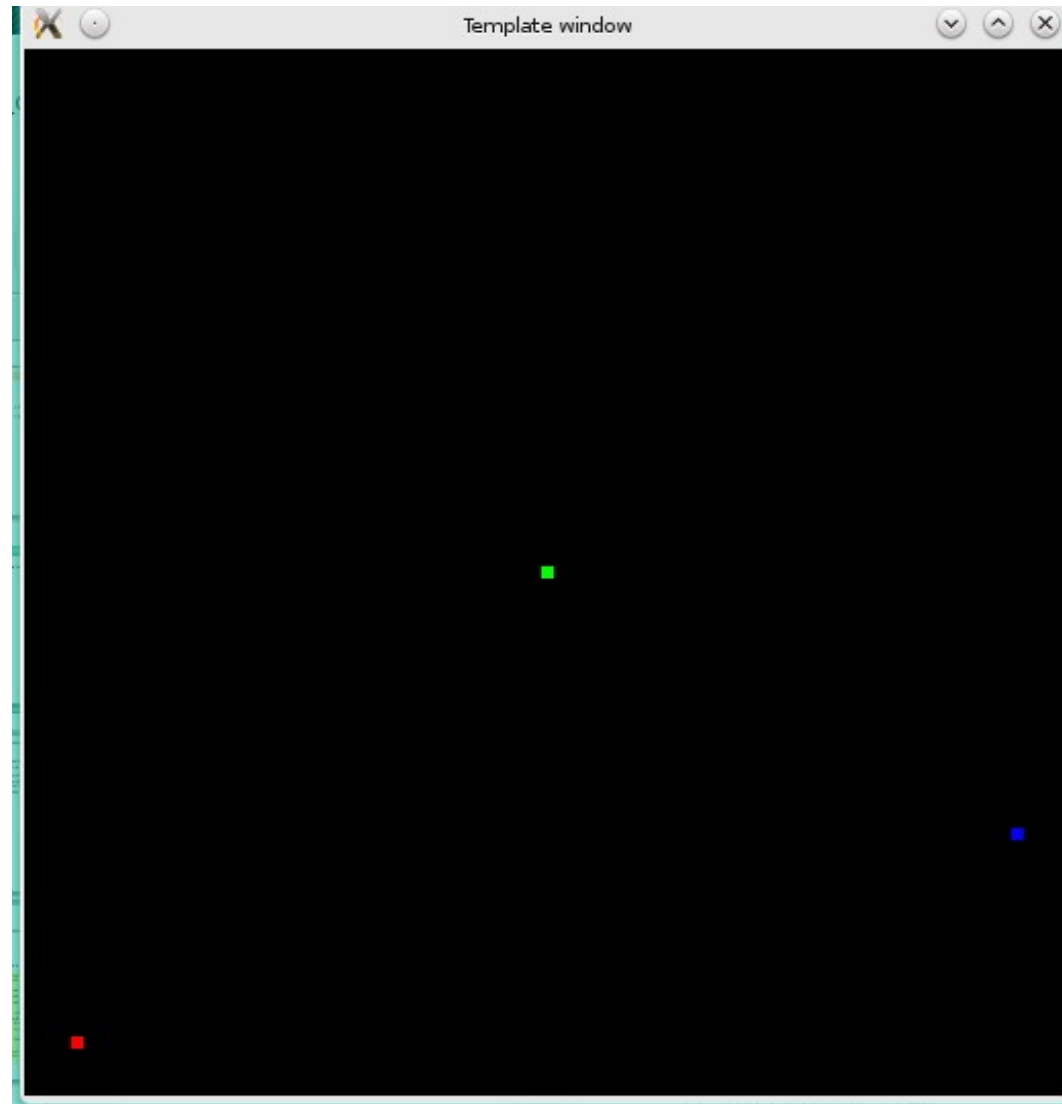- CUDA+OpenGL.
- Вычислительные шейдеры (введение).

glDrawArrays(GL_POINTS /*GL_TRIANGLES*/, 0, num_of_verticies);

```
int initBuffer(){
  glGenBuffers( 1,&bufferID);
  glBindBuffer( GL_ARRAY_BUFFER, bufferID);

  GLfloat*  vertex_buffer_data=(GLfloat*)calloc(num_of_verticies*6, sizeof(GLfloat));

  for(int i=0; i<num_of_verticies;i++){
      vertex_buffer_data[i*6] =  (GLfloat)(0.01f*(-99+2*(rand()%100)));
      vertex_buffer_data[i*6+1]= (GLfloat)(0.01f*(-99+2*(rand()%100)));
      vertex_buffer_data[i*6+2]= (GLfloat)0.0f;
      vertex_buffer_data[i*6+3]= (GLfloat)1.0f;
      vertex_buffer_data[i*6+4]= (GLfloat)1.0f;
      vertex_buffer_data[i*6+5]= (GLfloat)1.0f;
  }
/*  static const GLfloat vertex_buffer_data[] = {
                -0.9f, -0.9f, -0.0f, 1.0f, 0.0f, 0.0f,
                 0.0f,   0.0f,  0.0f, 0.0f, 1.0f, 0.0f,
                 0.9f,  -0.5f,  0.0f, 0.0f, 0.0f, 1.0f,
                };
  */
  glBufferData( GL_ARRAY_BUFFER, 6*num_of_verticies*sizeof(float),
                                vertex_buffer_data, GL_DYNAMIC_DRAW );
  free(vertex_buffer_data);
  return 0;
}
```
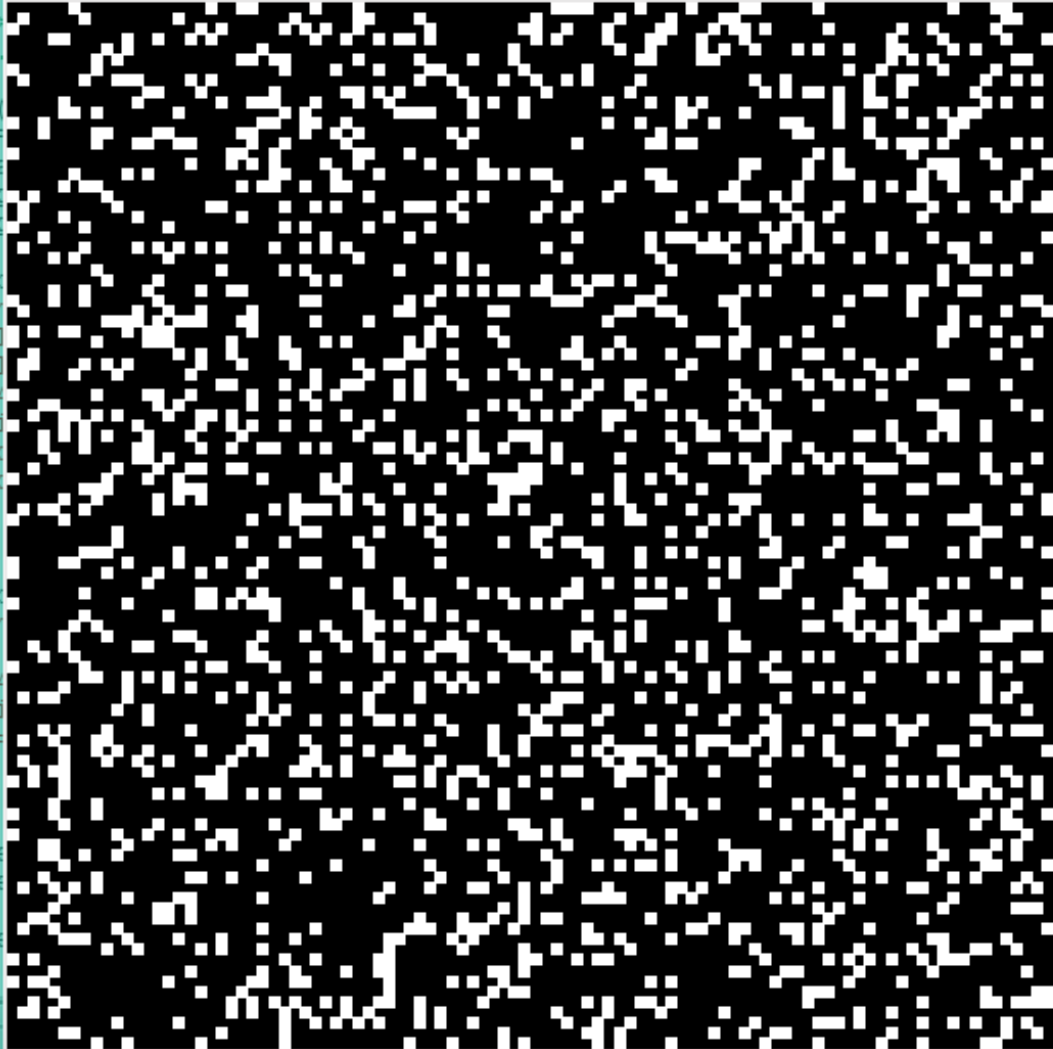
Tools   Settings   Help

ocument   💾 Save   📝 Save As   ❌ Close   ↩ Undo   ↪ Redo

Template window

```cpp
#include <GL/glew.h>
#include <stdio.h>
#include <stdlib.h>
#include <string>

void checkErrors(std::string desc) {
        GLenum e = glGetError();
        if (e != GL_NO_ERROR) {
                fprintf(stderr, "Oper
                exit(20);
        }
}

const unsigned int window_width  = 51
const unsigned int window_height = 51

GLuint bufferID;
GLuint progHandle;
GLuint genRenderProg(); |

const int num_of_verticies=2048;

int initBuffer(){

    glGenBuffers( 1,&bufferID);

    glBindBuffer( GL_ARRAY_BUFFER, buf

    /*
    static const GLfloat vertex_buffer_
            »        »        »        -0.91
            »        »        »        0.01
            »        »        »        0.91
            »        »        »        };
    */
    GLfloat*  vertex_buffer_data=(GLfl
                    num of verticio
```

Line: 19 of 79 Col: 25   LINE   INS

```
void initData();

int initBuffer(){

  glGenBuffers( 1,&bufferID);
  glBindBuffer( GL_ARRAY_BUFFER, bufferID);


  glBufferData( GL_ARRAY_BUFFER, 6*num_of_verticies*sizeof(float),
                                     0, GL_DYNAMIC_DRAW );
  initData();

  return 0;
}
```

```c
void initData(){
  GLfloat*  vertex_buffer_data=(GLfloat*)calloc(
            num_of_verticies*6, sizeof(GLfloat));

  for(int i=0; i<num_of_verticies;i++){
    vertex_buffer_data[i*6] =  (GLfloat)(0.01f*(-99+2*(rand()%100)));
    vertex_buffer_data[i*6+1]= (GLfloat)(0.01f*(-99+2*(rand()%100)));
    vertex_buffer_data[i*6+2]= (GLfloat)0.0f;
    vertex_buffer_data[i*6+3]= (GLfloat)1.0f;
    vertex_buffer_data[i*6+4]= (GLfloat)0.0f;
    vertex_buffer_data[i*6+5]= (GLfloat)1.0f;
  }

  glBufferData( GL_ARRAY_BUFFER, 6*num_of_verticies*sizeof(float),
                              vertex_buffer_data, GL_DYNAMIC_DRAW );

  free(vertex_buffer_data);
}
```

```c
void initData();

int initBuffer(){

  glGenBuffers( 1,&bufferID);
  glBindBuffer( GL_ARRAY_BUFFER, bufferID);


  glBufferData( GL_ARRAY_BUFFER, 6*num_of_verticies*sizeof(float),
                                        0, GL_DYNAMIC_DRAW );
#ifdef HOST_CALC
  initData();
#endif
#ifdef CUDA_CALC
  hInitData(bufferID, num_of_verticies);
#endif
#ifdef  CSH_CALC
  csDataInit(bufferID, num_of_verticies);
#endif

  return 0;
}
```

```
#include <GL/glew.h>
#include <cuda_runtime.h>
#include <cuda_gl_interop.h>

__global__ void gInitData(float* devA);

void hInitData(GLuint bufID, int N){
  cudaGLRegisterBufferObject(bufID);

  float *devA;
  cudaGLMapBufferObject((void**)&devA, bufID);

  dim3 threads_per_block(128);
  dim3 blocks(N/128);

  gInitData<<<blocks, threads_per_block>>>(devA);
  cudaDeviceSynchronize();

  cudaGLUnmapBufferObject(bufID);

  cudaFree(devA);
}
```
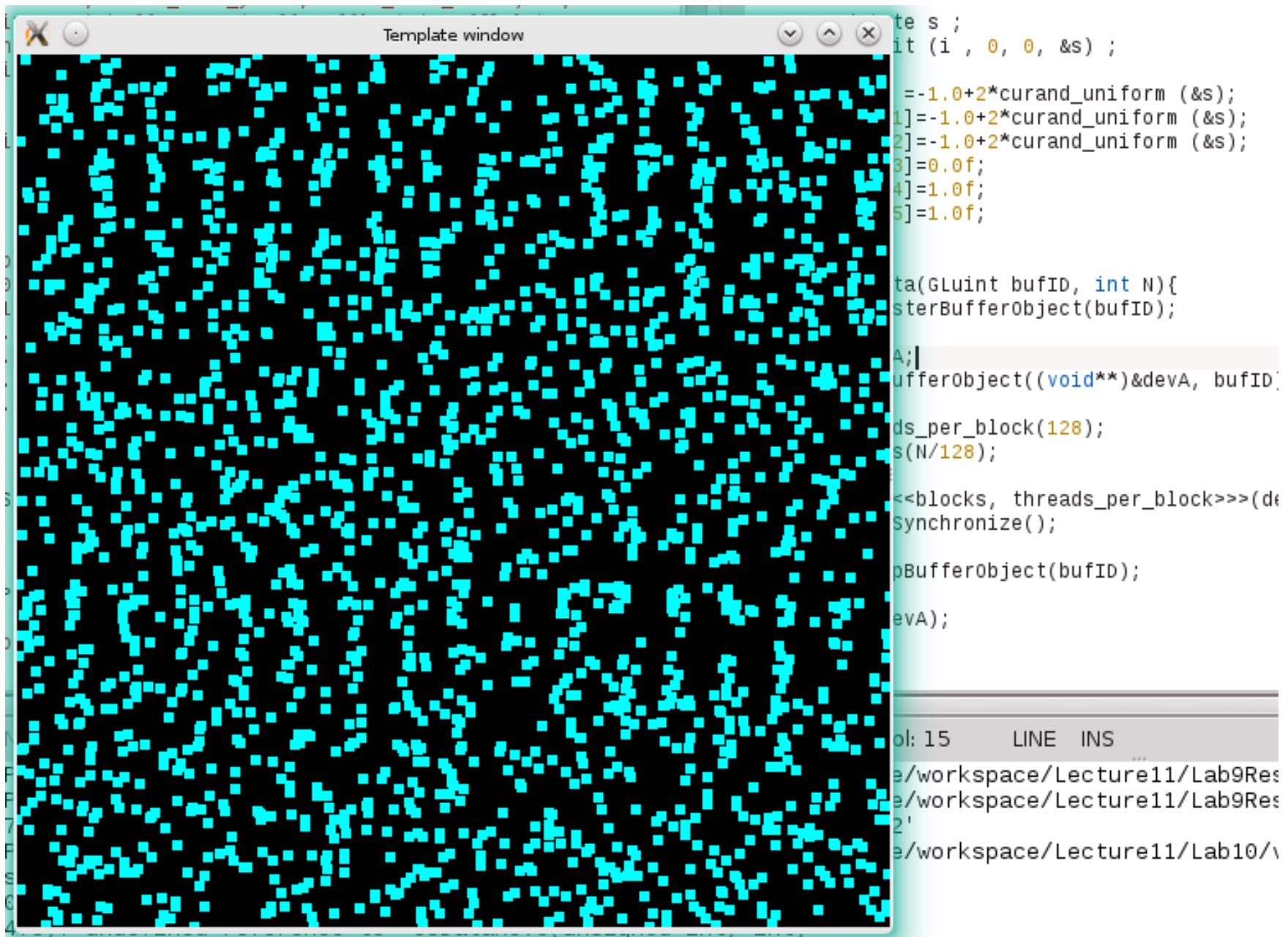
cuda_template.cu

```cpp
#include <curand_kernel.h>

__global__ void gInitData(float* devA){
  int i=threadIdx.x+blockIdx.x*blockDim.x;

  curandState s ;
  curand_init (i , 0, 0, &s) ;

  devA[i*6] =-1.0+2*curand_uniform (&s);
  devA[i*6+1]=-1.0+2*curand_uniform (&s);
  devA[i*6+2]=-1.0+2*curand_uniform (&s);
  devA[i*6+3]=0.0f;
  devA[i*6+4]=1.0f;
  devA[i*6+5]=1.0f;
}
```

```
~> nvcc -DCUDA_CALC -o s_templ  main.cpp util_template.cpp
sh_template.cpp cuda_template.cu    -lGLEW -lGL  -lGLU -lglfw
```

```
te s ;
it (i , 0, 0, &s) ;

=-1.0+2*curand_uniform (&s);
]=-1.0+2*curand_uniform (&s);
]=-1.0+2*curand_uniform (&s);
]=0.0f;
]=1.0f;
]=1.0f;

ta(GLuint bufID, int N){
sterBufferObject(bufID);

A;
ufferObject((void**)&devA, bufID)

ds_per_block(128);
s(N/128);

<<blocks, threads_per_block>>>(de
Synchronize();

pBufferObject(bufID);

evA);
```

Template window

ol: 15   LINE   INS

e/workspace/Lecture11/Lab9Res
e/workspace/Lecture11/Lab9Res
2'
e/workspace/Lecture11/Lab10/

~> nvcc -DCUDA_CALC -o s_templ main.cpp util_template.cpp
sh_template.cpp cuda_template.cu   -lGLEW -lGL -lGLU -lglfw

```cpp
#include <GL/glew.h>
#include <stdio.h>
#include <string>
#include <string.h>
#include <stdlib.h>

void checkErrors(std::string desc);
GLuint genComputeProg();
GLuint computeShaderID;

void csDataInit(GLuint inBuf,int N){
  glBindBufferBase(GL_SHADER_STORAGE_BUFFER, 0, inBuf);

  computeShaderID=genComputeProg();
  glUseProgram(computeShaderID);

  glDispatchCompute(N/128, 1, 1);

  glMemoryBarrier( GL_SHADER_STORAGE_BARRIER_BIT );

  }
```

csh_template.cpp

```
GLuint genComputeProg(){
    GLuint progHandle = glCreateProgram();
    GLuint cs = glCreateShader(GL_COMPUTE_SHADER);

    const char *cpSrc[] = {
      "#version 430\n",
      "layout (local_size_x = 128, local_size_y = 1, local_size_z = 1) in;\
       layout(std430, binding = 0) buffer InputBufferA{float inBuf[];};\
       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() {\
        uint index = gl_GlobalInvocationID.x;\
        inBuf[index*6]=-1.0+2.0*lmap(index);\
        inBuf[index*6+1]=-1.0+2.0*lmap(index*10);\
        inBuf[index*6+2]=0.0;\
        inBuf[progex*6+3]=1.0;\
        inBuf[index*6+4]=1.0;\
        inBuf[index*6+5]=0.0;\
         }"
    };
```

```c
glShaderSource(cs, 2, cpSrc, NULL);

glCompileShader(cs);
int rvalue;
glGetShaderiv(cs, GL_COMPILE_STATUS, &rvalue);
if (!rvalue) {
    fprintf(stderr, "Error in compiling cs\n");
    exit(30);
}
glAttachShader(progHandle, cs);

glLinkProgram(progHandle);

glGetProgramiv(progHandle, GL_LINK_STATUS, &rvalue);
if (!rvalue) {
    fprintf(stderr, "Error in linking cs\n");
    exit(32);
}

checkErrors("Render shaders");

return progHandle;
}
```

Compute Shaders	CUDA
glUseProgram(computeShaderID); glDispatchCompute(N/128,1,1);	gDataInit<<<dim3(N/128,1,1)>>>(devA);
glMemoryBarrier(...);	cudaDeviceSynchronize();
gl_NumWorkGroups	gridDim
gl_WorkGroupSize	blockDim
gl_WorkGroupID	blockIdx
gl_LocalInvocationID	threadIdx
gl_GlobalInvocationID	threadIdx+ blockIdx* blockDim

```c
int info[3];

glGetIntegeri_v(GL_MAX_COMPUTE_WORK_GROUP_COUNT, 0,
&info[0]);
glGetIntegeri_v(GL_MAX_COMPUTE_WORK_GROUP_COUNT, 1,
&info[1]);
glGetIntegeri_v(GL_MAX_COMPUTE_WORK_GROUP_COUNT, 2,
&info[2]);

printf("max work group size x:%i y:%i z:%i\n", info[0], info[1], info[2]);

glGetIntegeri_v(GL_MAX_COMPUTE_WORK_GROUP_SIZE, 0, &info[0]);
glGetIntegeri_v(GL_MAX_COMPUTE_WORK_GROUP_SIZE, 1, &info[1]);
glGetIntegeri_v(GL_MAX_COMPUTE_WORK_GROUP_SIZE, 2, &info[2]);

printf("max local work group sizes x:%i y:%i z:%i\n", info[0], info[1], info[2]);
```

```
t genComputeProg(){
Luint progHandle = glCreateProgram();
Luint cs = glCreateShader(GL_COMPUTE_SHADER);

onst char *cpSrc[] = {
    "#version 430\n",
    "layout (local_size_x = 128, local_size_y
    layout(std430, binding = 0) buffer InputB
    /*layout(std430, binding = 1) buffer Outp
    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() {\
    uint index = gl_GlobalInvocationID.x;\
    inBuf[index*6]=-1.0+2.0*lmap(index);\
    inBuf[index*6+1]=-1.0+2.0*lmap(index*10)
    inBuf[index*6+2]=0.0;\
    inBuf[index*6+3]=1.0;\
    inBuf[index*6+4]=1.0;\
    inBuf[index*6+5]=0.0;\
    }"
    };

lShaderSource(cs, 2, cpSrc, NULL);

lCompileShader(cs);
nt rvalue;
lGetShaderiv(cs, GL_COMPILE_STATUS, &rvalue);
f (!rvalue) {
    fprintf(stderr, "Error in compiling cs\n")
    exit(30);
```
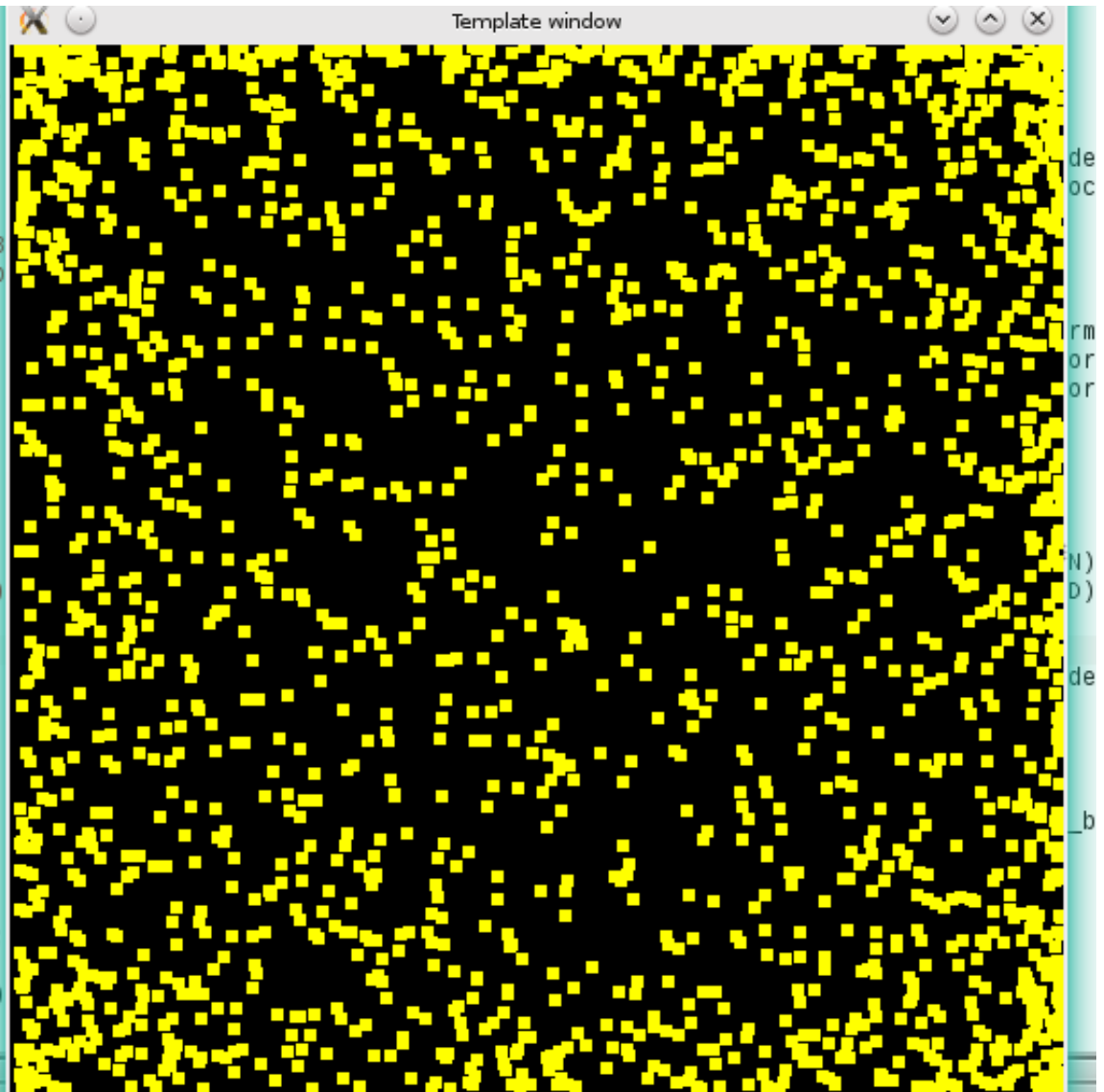
```
~> g++ -DCSH_CALC -o s_templ  main.cpp util_template.cpp
sh_template.cpp  csh_template.cpp   -lGLEW -lGL  -lGLU -lglfw
```

Спасибо за внимание!