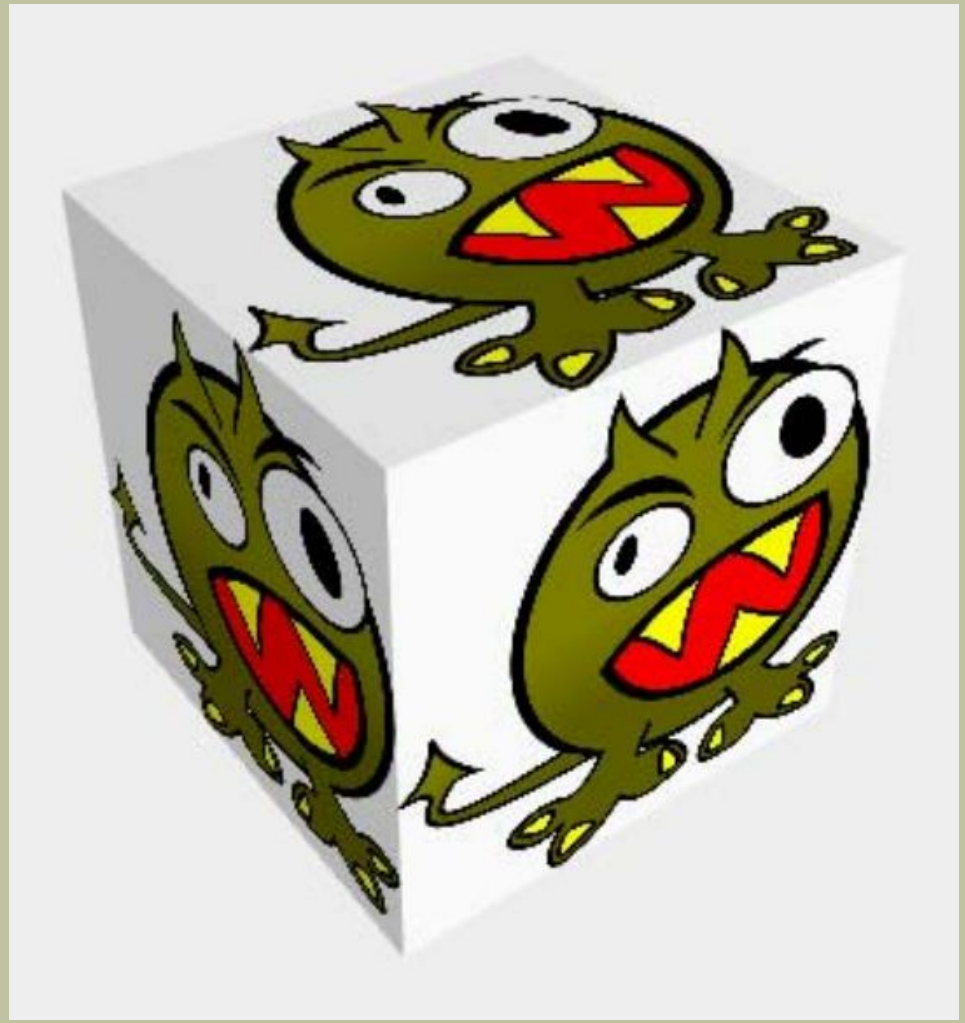
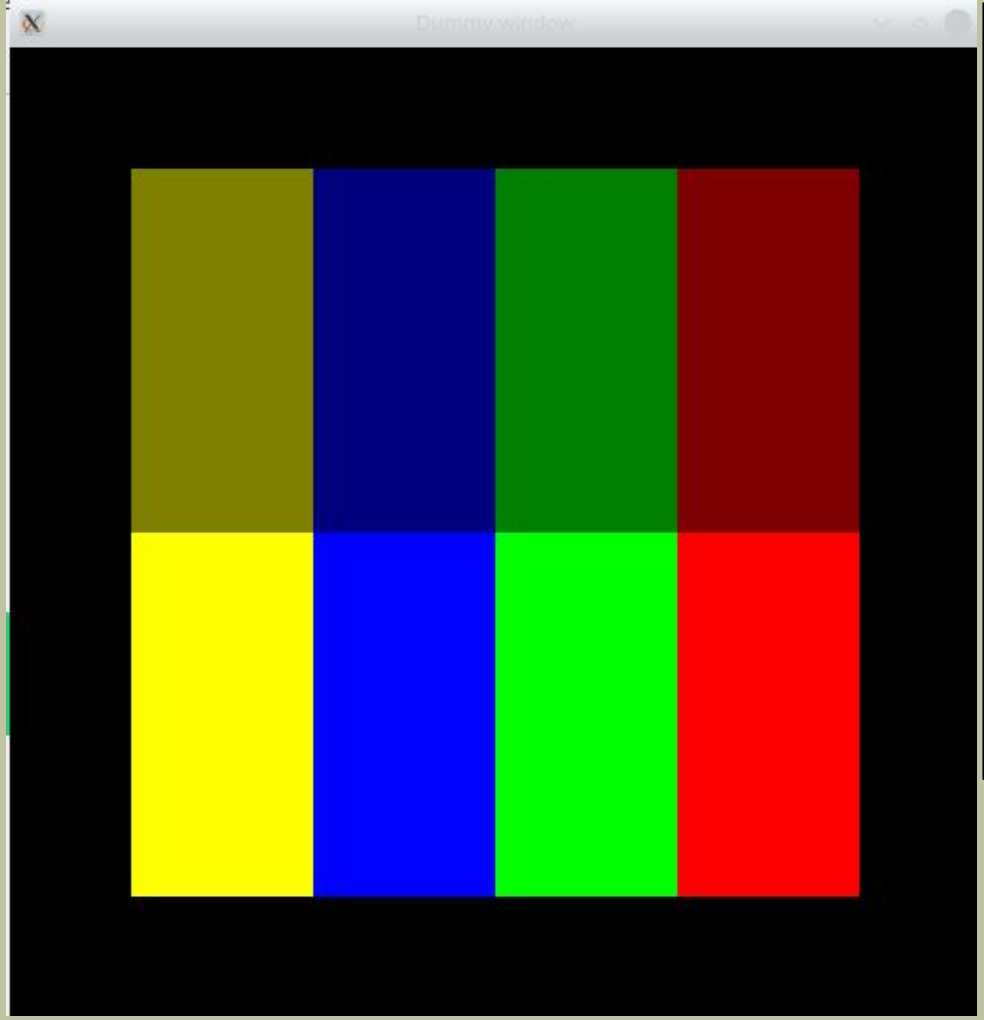
Лекция 14

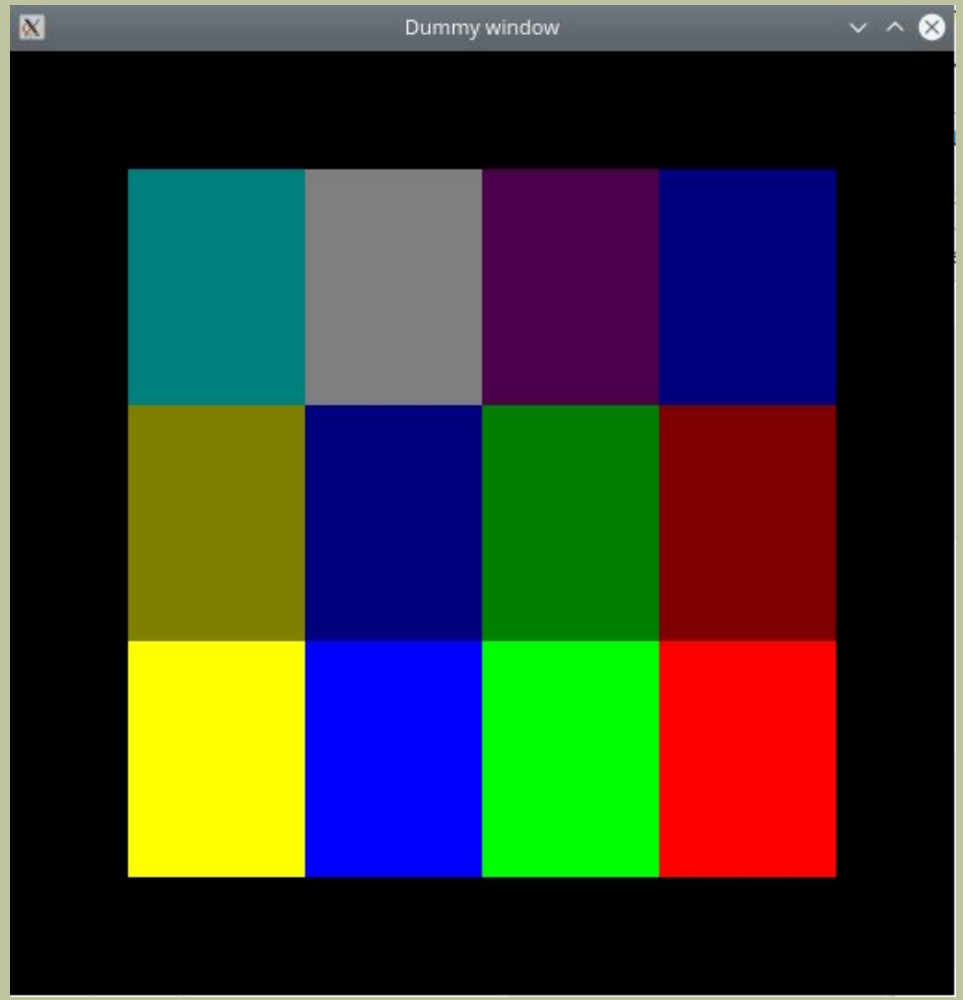
ТЕКСТУРЫ
(конвейер OpenGL)





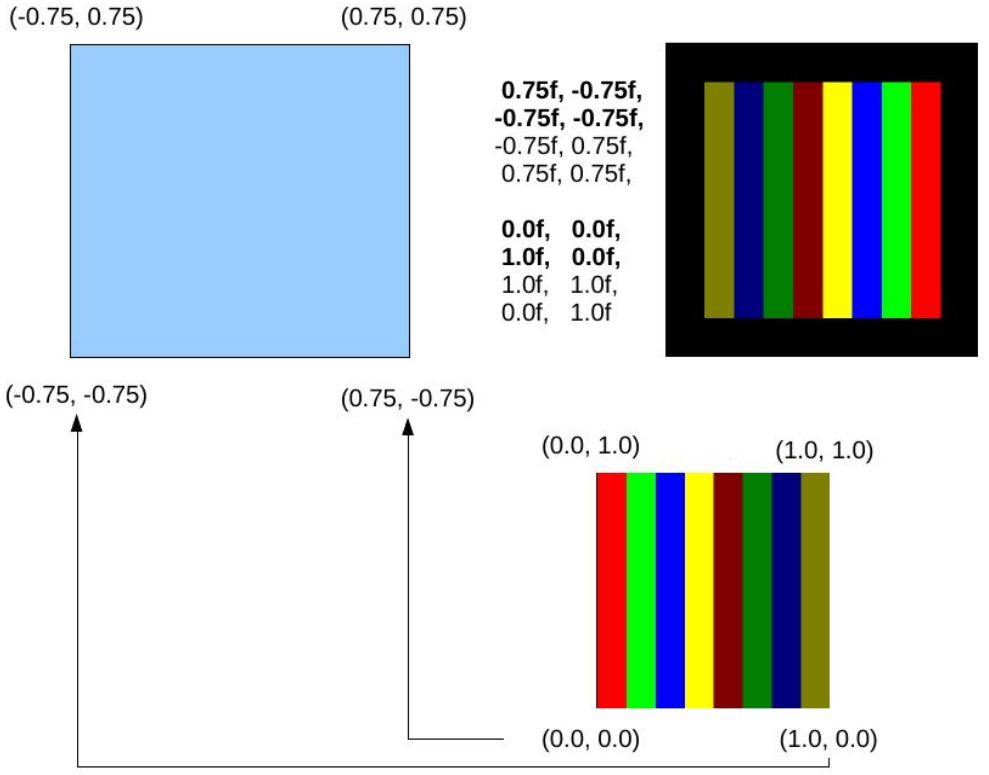

```
 static const GLfloat tex_color_data[]
={
    1.0f, 0.0f, 0.0f, 1.0f,
    0.0f, 1.0f, 0.0f, 1.0f,
    0.0f, 0.0f, 1.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,

    0.5f, 0.0f, 0.0f, 1.0f,
    0.0f, 0.5f, 0.0f, 1.0f,
    0.0f, 0.0f, 0.5f, 1.0f,
    0.5f, 0.5f, 0.0f, 1.0f,
 };
```

int L=4, M=2;

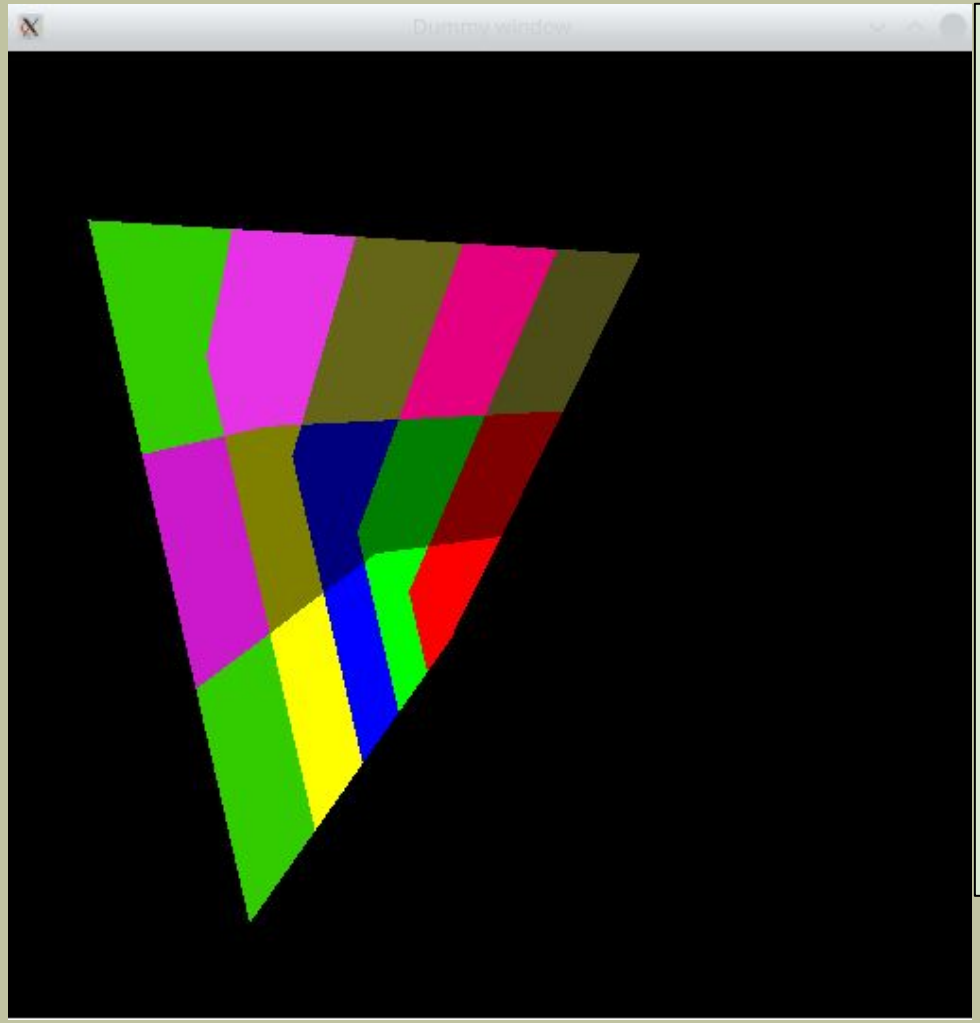



```
 static const GLfloat tex_color_data[]
={
    1.0f, 0.0f, 0.0f, 1.0f,
    0.0f, 1.0f, 0.0f, 1.0f,
    0.0f, 0.0f, 1.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,

    0.5f, 0.0f, 0.0f, 1.0f,
    0.0f, 0.5f, 0.0f, 1.0f,
    0.0f, 0.0f, 0.5f, 1.0f,
    0.5f, 0.5f, 0.0f, 1.0f,

    0.0f, 0.0f, 0.5f, 1.0f,
    0.3f, 0.0f, 0.3f, 1.0f,
    0.0f, 0.5f, 0.5f, 1.0f,
    0.0f, 0.5f, 0.5f, 1.0f
 };
```

int L=4, M=3;

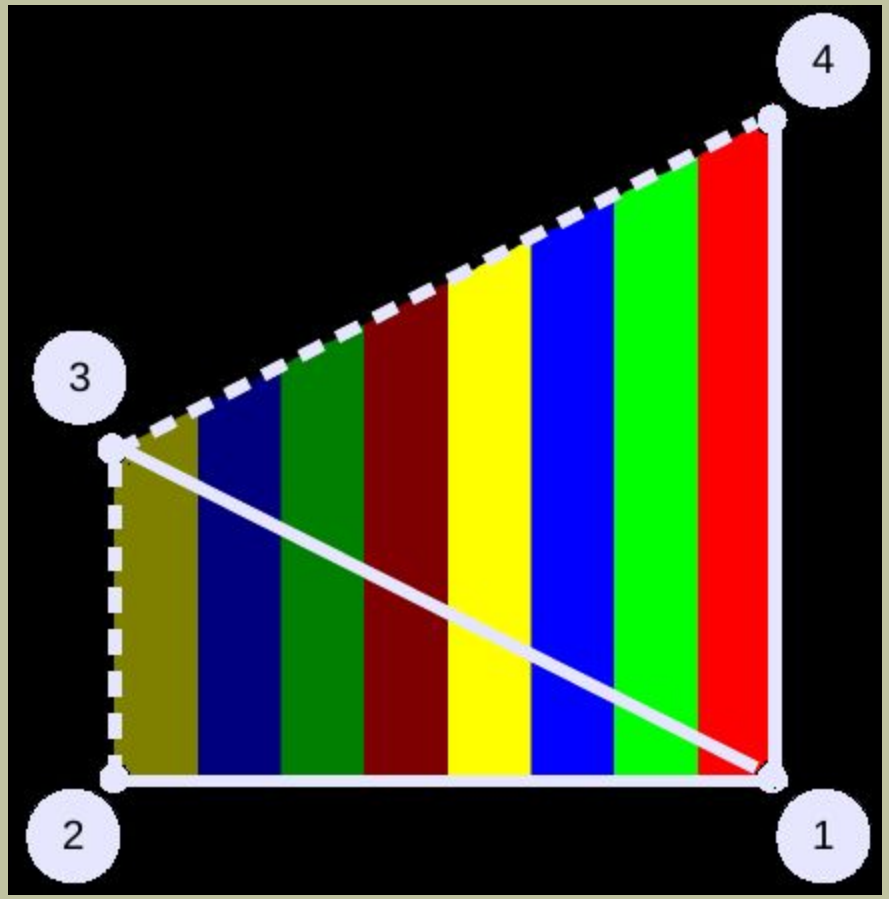


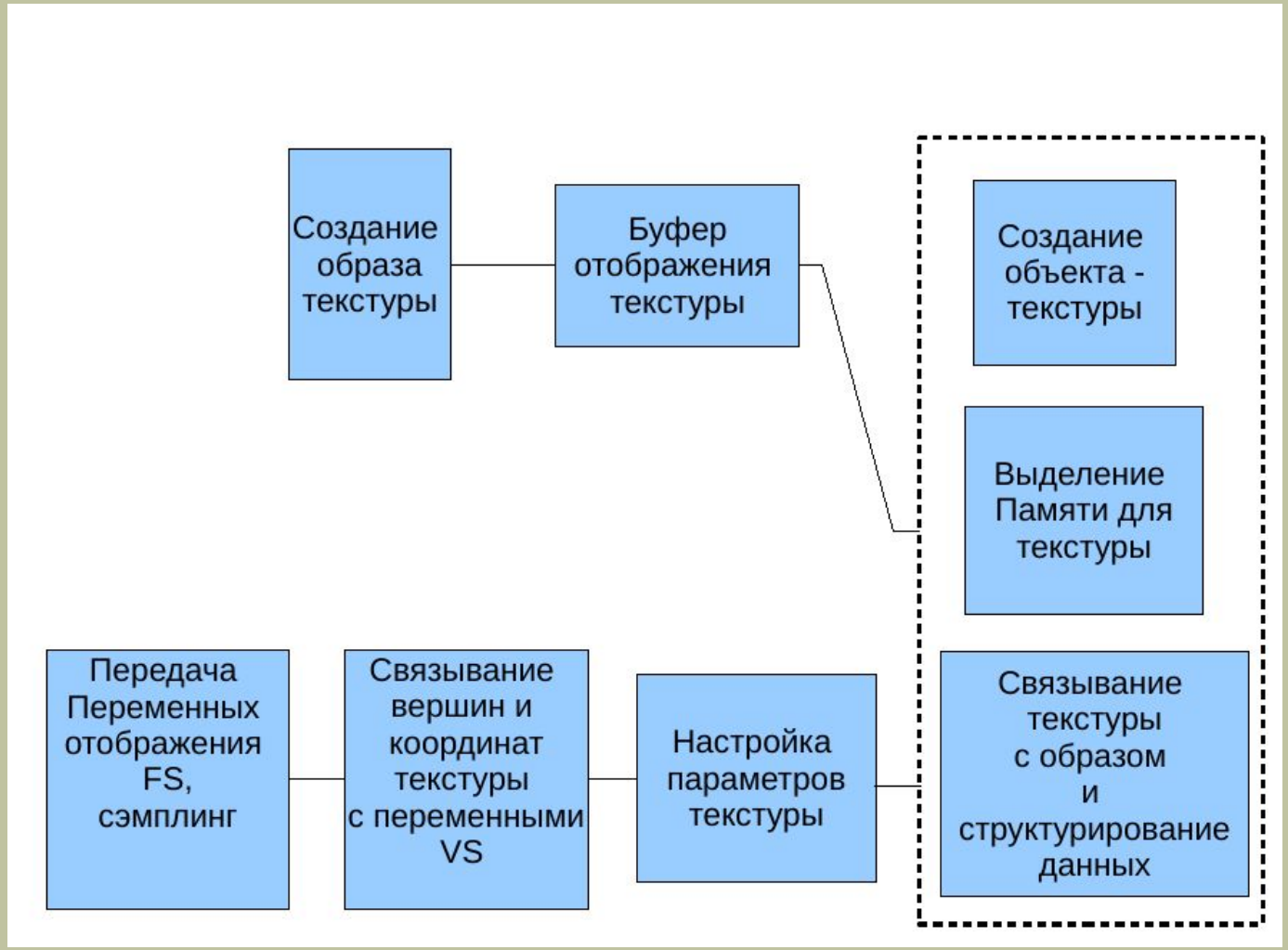

```
static const GLfloat map_data[]
= {
    0.75f, -0.75f,
   -0.75f, -0.75f,
   -0.75f,  0.75f,
    0.75f,  0.75f,

    0.0f, 0.0f,
    1.0f, 0.0f,
    1.0f, 1.0f,
    0.0f, 1.0f
  };
```




```
 static const GLfloat map_data[]
= {
     0.75f, -0.75f, -1.0,
    -0.75f, -0.75f,  0.0,
    -0.75f,  0.75f, -0.5,
     0.75f,  0.75f, -0.1,

     0.0f, 0.0f,
     1.0f, 0.0f,
     1.0f, 1.0f,
     0.0f, 1.0f
   };
```

glDrawArrays(**GL_TRIANGLE_FAN**, 0, 4);




Создание образа текстуры
Буфер отображения текстуры
Создание объекта - текстуры
Выделение Памяти для текстуры
Передача Переменных отображения FS, сэмплинг
Связывание вершин и координат текстуры с переменными VS
Настройка параметров текстуры
Связывание текстуры с образом и структурирование данных

```
#include <GL/glew.h>
#include <string>

void checkErrors(std::string desc);
int L=4, M=2;

GLuint genTexBuffer(){
  GLuint tex_buf;
  glGenBuffers(1, &tex_buf);

  static const GLfloat tex_color_data[] ={
          1.0f, 0.0f, 0.0f, 1.0f,
          0.0f, 1.0f, 0.0f, 1.0f,
          0.0f, 0.0f, 1.0f, 1.0f,
          1.0f, 1.0f, 0.0f, 1.0f,
```

tex_gen.cpp

```
        0.5f, 0.0f, 0.0f, 1.0f,
        0.0f, 0.5f, 0.0f, 1.0f,
        0.0f, 0.0f, 0.5f, 1.0f,
        0.5f, 0.5f, 0.0f, 1.0f,
};

glBufferData(GL_PIXEL_UNPACK_BUFFER , sizeof(tex_color_data),
                                                tex_color_data, GL_STATIC_DRAW);


  return tex_buf;
}
```

```
GLuint genMapBuffer(){
  GLuint map_buf;
  glGenBuffers(1, &map_buf);

  static const GLfloat map_data[] = {
           0.75f, -0.75f,
          -0.75f, -0.75f,
          -0.75f,  0.75f,
           0.75f,  0.75f,

           0.0f, 1.0f,
           1.0f, 0.0f,
           0.0f, 0.0f,
           1.0f, 1.0f
};
```

```
  glBindBuffer(GL_ARRAY_BUFFER, map_buf);
  glBufferData(GL_ARRAY_BUFFER, sizeof(map_data), map_data,
                                                  GL_STATIC_DRAW);

  return map_buf;
}
```

```
GLuint genTexture(){
  GLuint texHandle;
  glGenTextures(1, &texHandle);
  glBindTexture(GL_TEXTURE_2D, texHandle);
  glTexStorage2D(GL_TEXTURE_2D, 1, GL_RGBA8, L, M);

  glTexSubImage2D(GL_TEXTURE_2D, //тип текстуры
      0,       //уровень детализации (mipmap)
      0, 0,   //смещения текселя в x и y направлениях в массиве текстуры
      L, M,  //ширина и высота текстурного subimage
      GL_RGBA, //формат пикселя, GL_RGB, GL_RGBA и т.д.
      GL_FLOAT,  //тип данных пикселя
      NULL); //указатель на image в памяти
```

```
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
                                                  GL_NEAREST);//GL_LINEAR);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
                                                  GL_NEAREST);//GL_LINEAR);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,
                                                      GL_CLAMP_TO_EDGE);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T,
                                                      GL_CLAMP_TO_EDGE);
  checkErrors("Gen texture");
  return texHandle;
}
void initTexture(){
  genMapBuffer();
  genTexBuffer();
  genTexture();
}
```

tex_sh.cpp

```
#include <GL/glew.h>
#include <stdio.h>
#include <string>
#include <stdlib.h>
void checkErrors(std::string desc);

GLuint genRenderProg() {
  GLuint progHandle = glCreateProgram();
  GLuint vp = glCreateShader(GL_VERTEX_SHADER);
  GLuint fp = glCreateShader(GL_FRAGMENT_SHADER);
```

```
const char *vpSrc[] = {
  "#version 430\n",
  "layout (location = 0) in vec2 in_position;\
   layout (location = 1) in vec2 in_tex_coord;\
   out vec2 tex_coord;\
   void main(void){\
      gl_Position = vec4(in_position, 0.5, 1.0);\
      tex_coord = in_tex_coord;\
   }"
};
```

```
const char *fpSrc[] = {
  "#version 430\n",
  "in vec2 tex_coord;\
   layout (location = 0) out vec4 color;\
   uniform sampler2D tex;\
   void main(void){\
      color = texture(tex, tex_coord);\
   }"
};
```

```
glShaderSource(vp, 2, vpSrc, NULL);
glShaderSource(fp, 2, fpSrc, NULL);

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

```
  glCompileShader(fp);
  glGetShaderiv(fp, GL_COMPILE_STATUS, &rvalue);
  if (!rvalue) {
    fprintf(stderr, "Error in compiling fp\n");
    exit(31);
  }
  glAttachShader(progHandle, fp);
  glLinkProgram(progHandle);
  glGetProgramiv(progHandle, GL_LINK_STATUS, &rvalue);
  if (!rvalue) {
     fprintf(stderr, "Error in linking sp\n");
     exit(32);
  }
  checkErrors("Render shaders");
  return progHandle;
}
```

main.cpp

```
void initGL();
void initTexture();
GLuint genRenderProg();
void display();

int main(){
initGL();
initTexture();
do{
……
glfwTerminate();
return 0;
}
```

```
void display(){
  GLuint progHandle;

  progHandle=genRenderProg();
  glUseProgram(progHandle);

  glVertexAttribPointer(0, 2, GL_FLOAT, GL_FALSE, 0, (GLvoid*)0);
  glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE, 0, (GLvoid*)(8 *
                                                        sizeof(float)));
……..
glDrawArrays(GL_TRIANGLE_FAN, 0, 4);
……..
}
```

Спасибо за внимание!