|  |
| --- |
| Master Electronic DesiGN |
| Aspect, Set Perspective |
| Homework6 |
|  |
| **Miguel Tlapa Juárez** |
| **5/05/2014** |



|  |
| --- |
| This document describes the system architecture and design about the body controller module, it’s have block diagram and flowchart to describe software and hardware architecture. |

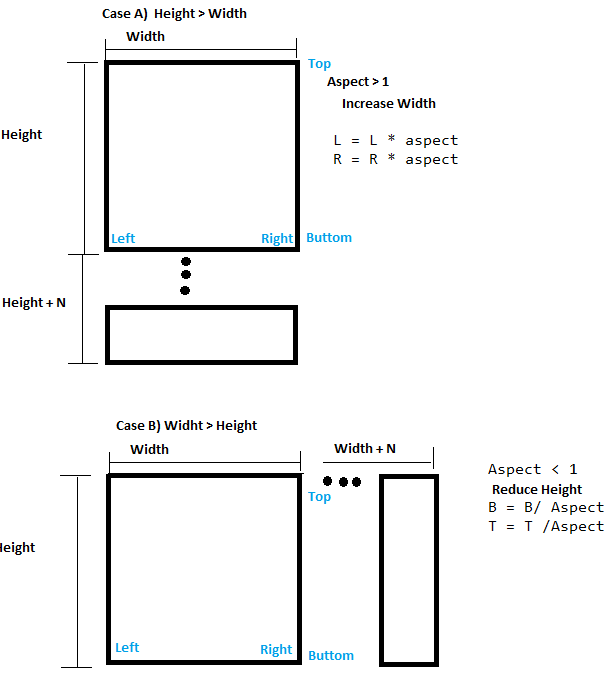
|  |  |  |  |
| --- | --- | --- | --- |
| *Revision History* | | | |
| Date | Revision Number | Author/Editor | Modifications |
| January 2014 | 0.1 | Miguel Tlapa | Created file |
|  |  |  |  |

|  |
| --- |
| *Disclaimers* |

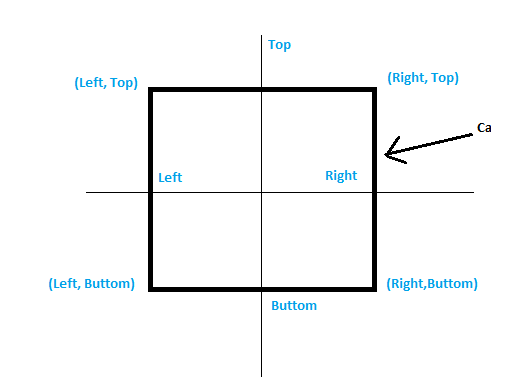
# Explanation/\*



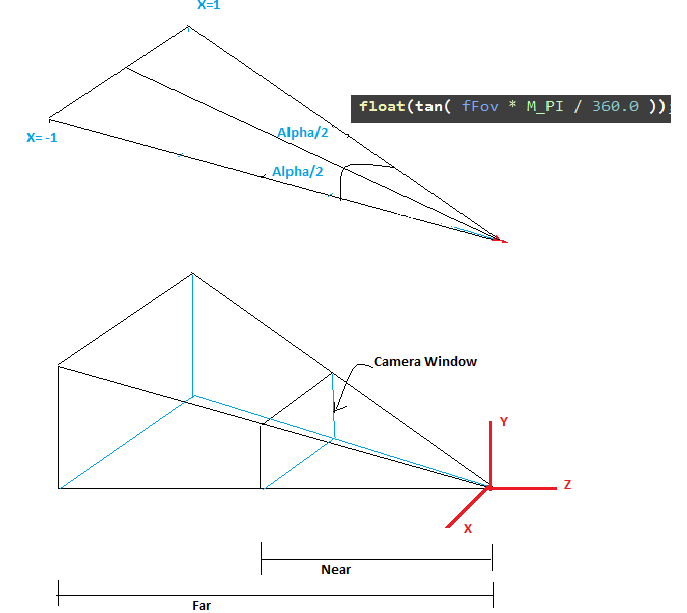
**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ASPECT\*\*\*\*\*\*\*\*\*\*\*\*\***



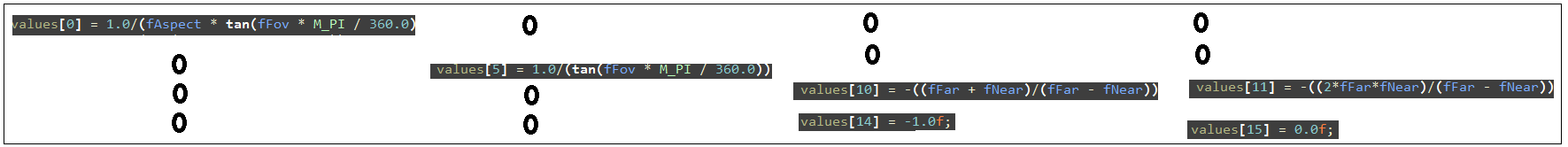
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SET Perspective \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



Camera Window



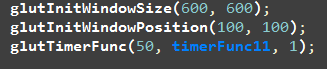
Perspective Matrix



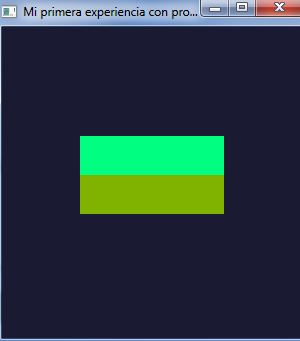
RUNNING PROGRAM

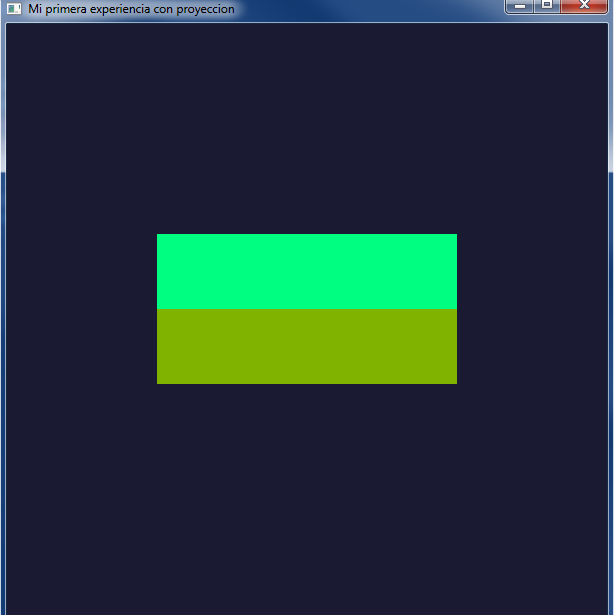
Testing ASPECT

INPUT

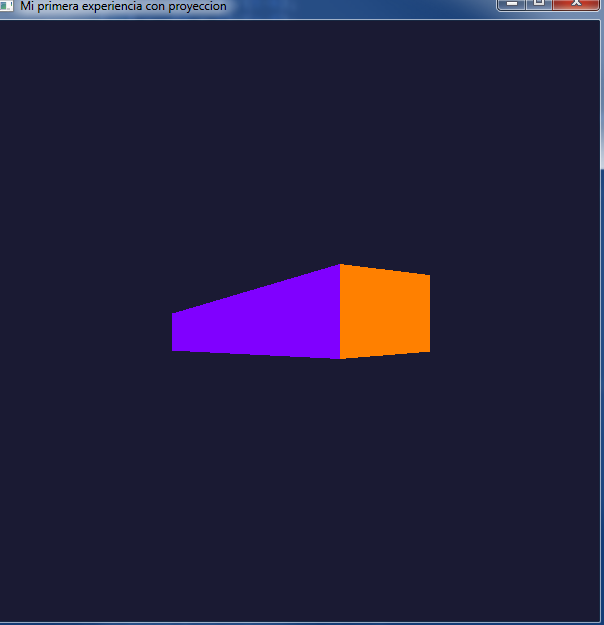




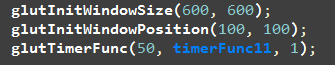




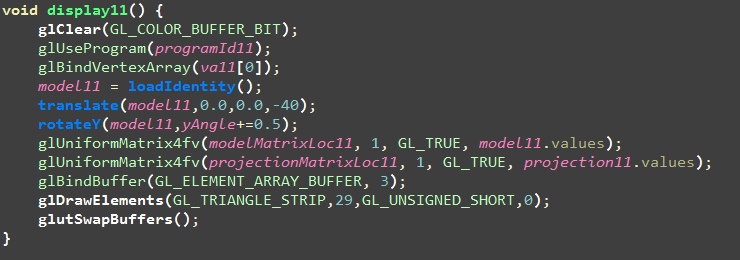
Testing Perspective



INPUT







**#include** <GL/glew.h>

**#include** <GL/freeglut.h>

**#include** "Utils.h"

**#include** "Transforms.h"

**using** **namespace** mat4;

**GLuint** *programId11*, *va11***[**1**]**, *vertexPosLoc11*, *vertexColorLoc11*, *modelColorLoc11*, *modelMatrixLoc11*, *projectionMatrixLoc11*;

**Matrix4** *model11*, *projection11*;

**float** *yAngle* = 0;

**void** **initShaders11()** **{**

**GLuint** vShader = **Utils**::*compileShader***(**"Shaders/proj\_model\_col\_pos.vsh", GL\_VERTEX\_SHADER**)**;

**if** **(**!**Utils**::*shaderCompiled***(**vShader**))** **return**;

*cout* << "compilo el vertex shader" << **endl**;

**GLuint** fShader = **Utils**::*compileShader***(**"Shaders/color.fsh", GL\_FRAGMENT\_SHADER**)**;

**if** **(**!**Utils**::*shaderCompiled***(**fShader**))** **return**;

*programId11* = glCreateProgram**()**;

glAttachShader**(***programId11*, vShader**)**;

glAttachShader**(***programId11*, fShader**)**;

glLinkProgram**(***programId11***)**;

*vertexPosLoc11* = glGetAttribLocation**(***programId11*, "vertexPosition"**)**;

*vertexColorLoc11* = glGetAttribLocation**(***programId11*, "vertexColor"**)**;

*modelMatrixLoc11* = glGetUniformLocation**(***programId11*, "modelMatrix"**)**;

*projectionMatrixLoc11* = glGetUniformLocation**(***programId11*, "projectionMatrix"**)**;

**}**

**void** **myReshapeFunc(int** width , **int** hight **)**

**{**

*cout* << "valor width "<< width << **endl**;

*cout* << "valor hight "<< hight << **endl**;

/\* void glViewport( GLint x,GLint y,GLsizei width,GLsizei height)

x, y Specify the lower left corner of the viewport rectangle, in pixels. The initial value is (0,0).

width, height Specify the width and height of the viewport.

When a GL context is first attached to a window, width and height are set to the dimensions of that window.

\*/

**glViewport(** 0, 0, width, hight **)**;

**float** c = 1.0 \* width; // Change width and hight in float

**float** d = 1.0 \* hight;

**float** aspect; // Define Aspect Variable

**if(**width == 0**){** // Code that avoids to divide by zero

width = 1;

**}**

//aspect =hight/width;

aspect =d/c;

*cout* << "aspect: "<< aspect << **endl**;

//projection11.setOrtho(aspect,-10, 10,-10, 10, 10, -10); //Send aspect and

*projection11*.setPerspective**(**53,aspect,10,100**)**;

**}**

**void** **createModel11()** **{**

**float** modelPos**[]** = **{** // Cara Frontal (verde obscuro)

-5, -2.0, 7, 5, -2.0, 7, // 0 //1

-5, 0.5, 7, 5, 0.5, 7, //1 //2

// Cara Superior (verde)

-5, 0.5, 7, 5, 0.5, 7,

-5, 3.0, -7, 5, 3.0, -7,

// Cara posterior( naranja)

-5, 3.0, -7, 5, 3.0, -7,

-5, -2.0, -7, 5, -2.0, -7,

// Cara INFErior( AZUL)

-5, -2.0, -7, 5, -2.0, -7,

-5, -2.0, 7, 5, -2.0, 7,

// cARA dERECHA (MORADO)

5,-2.0, 7, 5, -2.0, -7,

5, 0.5, 7, 5, 3.0, -7,

// cARA iZQUIERDA (rOSA)

-5,-2.0, -7, -5, -2.0, 7,

-5, 3, -7, -5, 0.5, 7

**}**;

**float** modelColor**[]** = **{** 0.5, 0.7, 0, 0.5, 0.7, 0,

0.5, 0.7, 0, 0.5, 0.7, 0,

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

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

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

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

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

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

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

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

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

1.0, 0.0, 0.5, 1.0, 0.0, 0.5

**}**;

**GLushort** modelIndex**[]** = **{** 0, 1, 2,3, 0xFFFF,

4,5,6,7,0xFFFF,

8,9,10,11,0xFFFF,

12,13,14,15,0xFFFF,

16,17,18,19,0xFFFF,

20,21,22,23**}**;

glGenVertexArrays**(**1, *va11***)**;

glBindVertexArray**(***va11***[**0**])**;

glBindBuffer**(**GL\_ARRAY\_BUFFER, 1**)**;

glBufferData**(**GL\_ARRAY\_BUFFER, **sizeof(**modelPos**)**, modelPos, GL\_STATIC\_DRAW**)**;

glEnableVertexAttribArray**(***vertexPosLoc11***)**;

glVertexAttribPointer**(***vertexPosLoc11*, 3, GL\_FLOAT, 0, 0, 0**)**;

glBindBuffer**(**GL\_ARRAY\_BUFFER, 2**)**;

glBufferData**(**GL\_ARRAY\_BUFFER, **sizeof(**modelColor**)**, modelColor, GL\_STATIC\_DRAW**)**;

glEnableVertexAttribArray**(***vertexColorLoc11***)**;

glVertexAttribPointer**(***vertexColorLoc11*, 3, GL\_FLOAT, 0, 0, 0**)**;

glBindBuffer**(**GL\_ELEMENT\_ARRAY\_BUFFER, 3**)**;

glBufferData**(**GL\_ELEMENT\_ARRAY\_BUFFER, **sizeof(**modelIndex**)**, modelIndex, GL\_STATIC\_DRAW**)**;

**glEnable(**GL\_PRIMITIVE\_RESTART**)**;

glPrimitiveRestartIndex**(**0xFFFF**)**;

**glEnable(**GL\_CULL\_FACE**)**;

// glFrontFace(GL\_CCW);

**}**

**void** **display11()** **{**

**glClear(**GL\_COLOR\_BUFFER\_BIT**)**;

glUseProgram**(***programId11***)**;

glBindVertexArray**(***va11***[**0**])**;

*model11* = **loadIdentity()**;

**translate(***model11*,0.0,0.0,-40**)**;

**rotateY(***model11*,*yAngle*+=0.5**)**;

glUniformMatrix4fv**(***modelMatrixLoc11*, 1, GL\_TRUE, *model11*.values**)**;

glUniformMatrix4fv**(***projectionMatrixLoc11*, 1, GL\_TRUE, *projection11*.values**)**;

glBindBuffer**(**GL\_ELEMENT\_ARRAY\_BUFFER, 3**)**;

**glDrawElements(**GL\_TRIANGLE\_STRIP,29,GL\_UNSIGNED\_SHORT,0**)**;

**glutSwapBuffers()**;

**}**

**void** **exitFunc11(unsigned** **char** key, **int** x, **int** y**)** **{**

**if** **(**key == 27**)** **{**

glDeleteVertexArrays**(**1, *va11***)**;

**exit(**0**)**;

**}**

**}**

**void** **timerFunc11(int** id**)** **{**

**glutTimerFunc(**10, **timerFunc11**, id**)**;

**glutPostRedisplay()**;

**}**

**int** **main(int** argc, **char** \*\*argv**)** **{**

glutInit**(**&argc, argv**)**;

**glutInitDisplayMode(**GLUT\_DOUBLE**)**;

**glutInitWindowSize(**600, 600**)**;

**glutInitWindowPosition(**100, 100**)**;

**glutTimerFunc(**50, **timerFunc11**, 1**)**;

glutCreateWindow**(**"Mi primera experiencia con proyeccion"**)**;

**glutDisplayFunc(display11)**;

**glutReshapeFunc(myReshapeFunc)**;// It is a callback that send width and height to myReshapeFunc

**glutKeyboardFunc(exitFunc11)**;

**glewInit()**;

**initShaders11()**;

**createModel11()**;

//projection11.setOrtho(-10, 10,-10, 10, 10, -10);

**glClearColor(**0.1, 0.1, 0.2, 1.0**)**;

**glutMainLoop()**;

**return** 0;

**}**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*MATRIX4.CPP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*

\* Matrix3.cpp

\*

\* Created on: 14/01/2014

\* Author: Iván

\*/

#include "Matrix4.h"

#include <cmath>

using namespace std;

namespace mat4 {

Matrix4::Matrix4() {

setIdentity();

}

Matrix4::~Matrix4() {

}

void Matrix4::setIdentity() {

for(int i = 1; i < 16; i ++) values[i] = 0;

values[0] = 1;

values[5] = 1;

values[10] = 1;

values[15] = 1;

}

void Matrix4::setOrtho(float aspect,float l, float r, float b, float t, float n, float f)

{

if(aspect >= 1)

{

l = l\*aspect;

r = r\*aspect;

/\* Widht

\* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\* | |

\* H | |

\* e | |

\* i | |

\* g | |

\* h | |

\* t | |

\* |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|

\*

\*/

cout<<"modifica l y r" << endl;

}

else {

b = b/aspect;

t = t/aspect;

cout<<"modifica b y t" << endl;

}

/\*

\* |2/(r-l) 0 0 -(l + r)/(r-l) |

\* |0 2/(t-b) 0 -(b + t)/(t-b) |

\* |0 0 2/(n-f) -(f + n)/(n-f) |

\* |0 0 0 1 |

\*

\*/

setIdentity();

values[0] = 2/(r-l);

values[5] = 2/(t-b);

values[10] = 2/(n-f);

values[3] = -(l + r)/(r-l);

values[7] = -(b + t)/(t-b);

values[11] = -(f + n)/(n-f);

}

void Matrix4::setPerspective(float fFov, float fAspect, float fNear, float fFar)

{

cout << "fFov: " << fFov << " Aspect: " << fAspect << " fNear: " << fNear << " fFar: " << fFar << endl;

setIdentity();

// Construct the projection matrix

values[0] = 1.0/(fAspect \* tan(fFov \* M\_PI / 360.0));

values[5] = 1.0/(tan(fFov \* M\_PI / 360.0));

values[10] = -((fFar + fNear)/(fFar - fNear));

values[11] = -((2\*fFar\*fNear)/(fFar - fNear));

values[14] = -1.0f;

values[15] = 0.0f;

// cout << "ymax:" << ymax<< " ymin:"<< ymin << " xmin:" << xmin << " xmax:" << xmax << endl;

// cout << "[0] " << values[0] << endl;

// cout << "[1] " << values[1] << endl;

// cout << "[2] " << values[2] << endl;

// cout << "[3] " << values[3] << endl;

// cout << "[4] " << values[4] << endl;

// cout << "[5] " << values[5] << endl;

// cout << "[6] " << values[6] << endl;

// cout << "[7] " << values[7] << endl;

// cout << "[8] " << values[8] << endl;

// cout << "[9] " << values[9] << endl;

// cout << "[10] " << values[10] << endl;

// cout << "[11] " << values[11] << endl;

// cout << "[12] " << values[12] << endl;

// cout << "[13] " << values[13] << endl;

// cout << "[14] " << values[14] << endl;

// cout << "[15] " << values[15] << endl;

}

void Matrix4::set(int c, int r, float v) {

if(c < 0 || r < 0 || c > 3 || r > 3) return;

values[r \* 4 + c] = v;

}

float Matrix4::get(int c, int r) const {

if(c < 0 || r < 0 || c > 3 || r > 3) return 0;

return values[r \* 4 + c];

}

Matrix4 Matrix4::operator \*(const Matrix4& m) {

Matrix4 res;

for(int c = 0; c < 4; c ++) {

for(int r = 0; r < 4; r ++) {

float sum = 0;

for(int k = 0; k < 4; k ++) {

sum += get(k, r) \* m.get(c, k);

}

res.set(c, r, sum);

}

}

return res;

}

ostream& operator<<(ostream& o, const Matrix4& m) {

for(int r = 0; r < 4; r ++) {

for(int c = 0; c < 4; c ++) {

o << m.get(c, r) << " ";

}

o << endl;

}

return o;

}

} /\* namespace CG \*/

\*\*\*\*\*\*\*\*\*\*MATRIX4.h\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**#ifndef** MATRIX4\_H\_

**#define** **MATRIX4\_H\_**

**#include** <iostream>

**using** **namespace** std;

**namespace** mat4 **{**

**class** **Matrix4** **{**

**public**:

**float** values**[**16**]**;

**Matrix4()**;

**virtual** **~Matrix4()**;

**void** **setIdentity()**;

**void** **setOrtho(float** aspect, **float** l, **float** r, **float** b, **float** t, **float** n, **float** f**)**;

**void** **set(int** c, **int** r, **float** v**)**;

**float** **get(int** c, **int** r**)** **const**;

**void** **setPerspective(float** fov, **float** ratio, **float** nearZ, **float** farZ**)**;

**void** **setView(float** x, **float** y, **float** z**)**;

**void** **setView(float** x, **float** y, **float** z, **float** lookAtX, **float** lookAtY, **float** lookAtZ, **float** upX, **float** upY, **float** upZ**)**;

**void** **setTranslation(float** tx, **float** ty, **float** tz**)**;

**Matrix4** **operator\*(const** **Matrix4** &m1**)**;

**friend** **ostream**& **operator<<(ostream**& o, **const** **Matrix4**& m**)**;

**}**;

**}** /\* namespace CG \*/

**#endif** /\* MATRIX4\_H\_ \*/