MASTER ELECTRONIC DESIGN

PACMAN

Homework3

Miguel Tlapa Juárez

19/02/2014

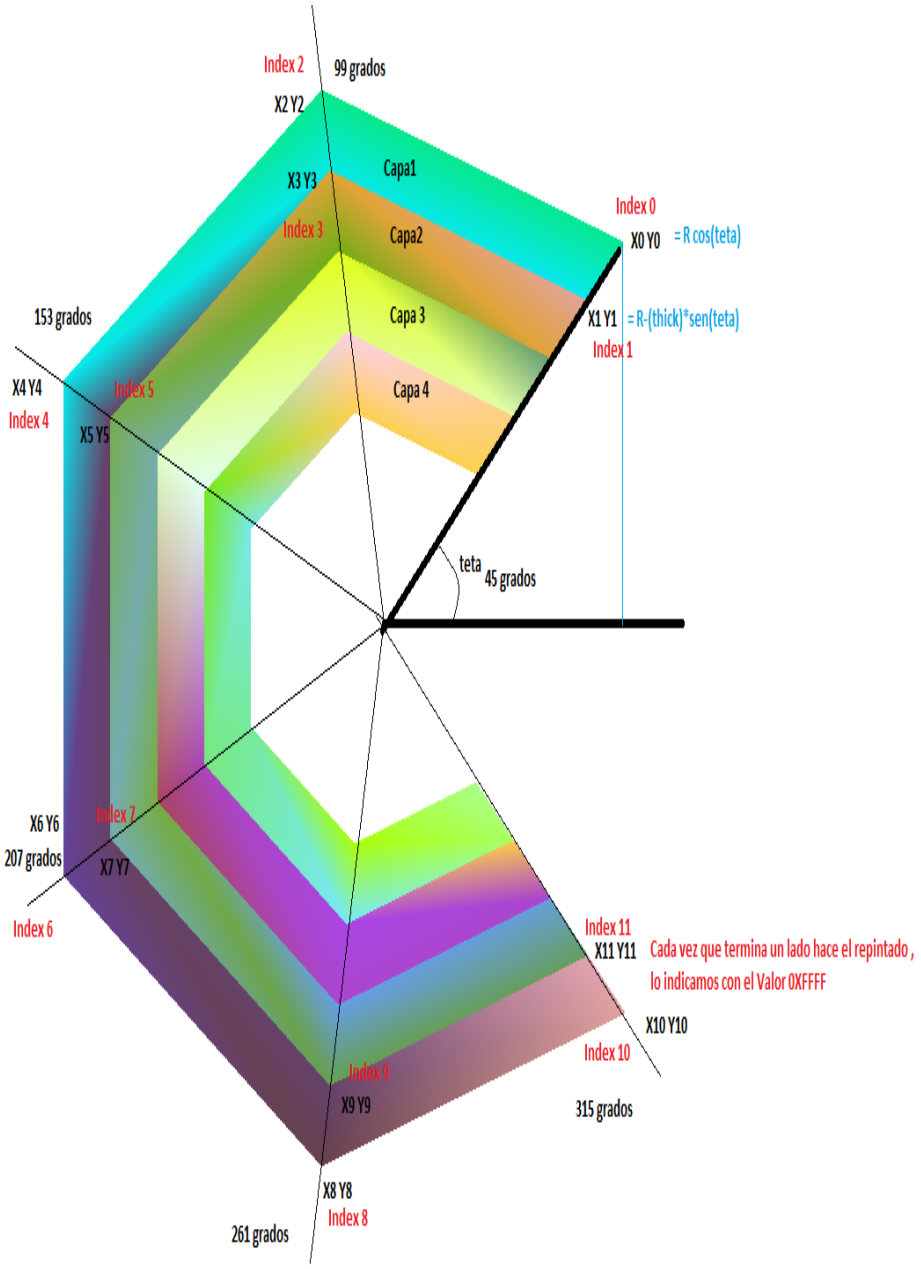
This document describes the system architecture and design about the body controller module, it's have block diagram and flowchart to describe software and hardware architecture.

Revision History

Date	Revision Number	Author/Editor	Modifications
January 2014	0.1	Miguel Tlapa	Created file

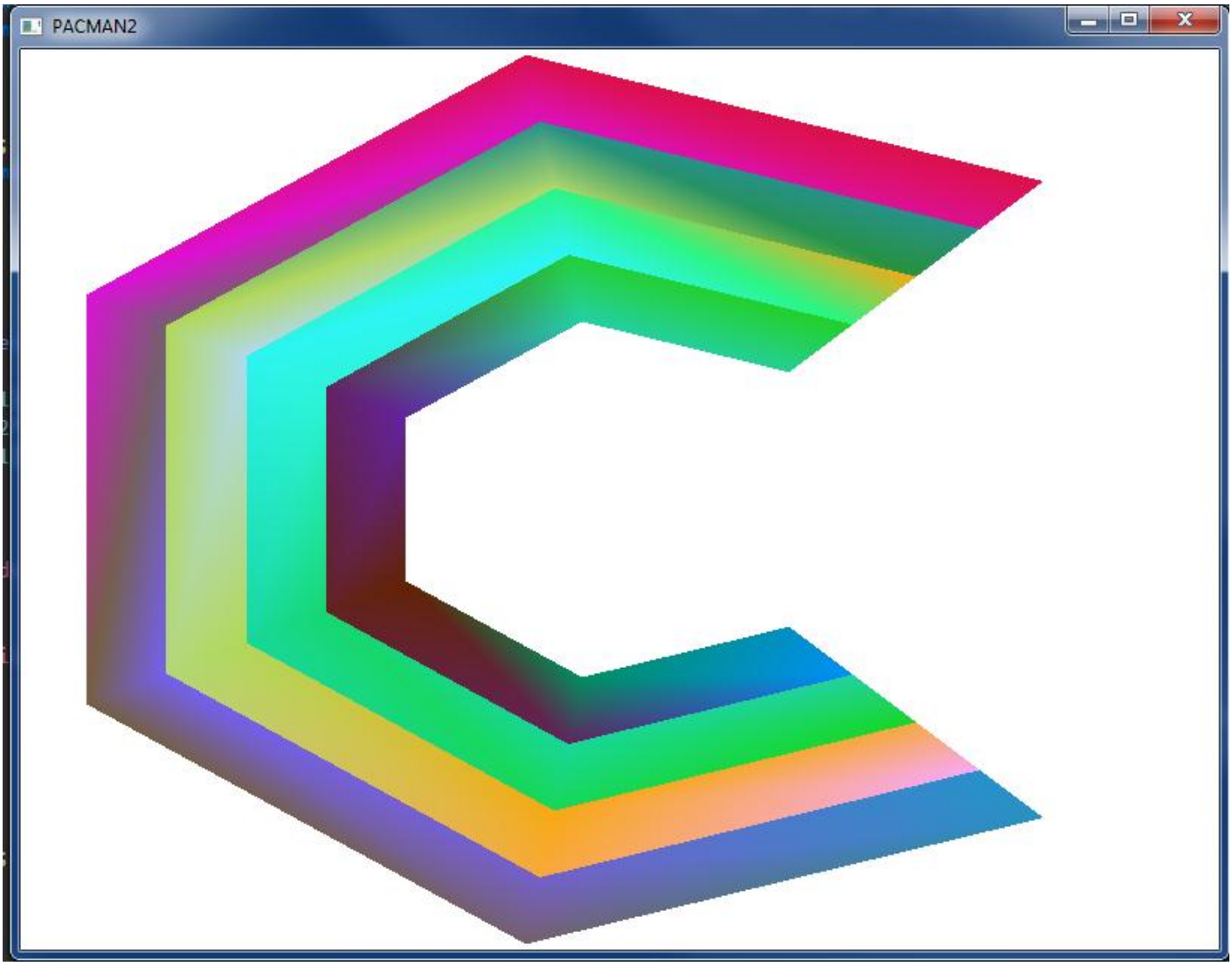
Disclaimers

1. Explanation



Estes valors son Simetricos, esta informacion sirvio para debugear el programa y saber si se estaba realizacion correctamente los calculos, es el primero con el penultimo

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	
0.707107,	0.707107,	0.601041,	0.601041,	-0.156434,	0.987688,	-0.132969,	0.839535,	-0.891007,	0.45399,	-0.757356,	0.385892,	-0.891007,	-0.453991,	-0.757356,	-0.385892,	-0.156435,	-0.987688,	-0.132969,	-0.839535,	0.707107,	-0.707107,	0.601041,	-0.601041,	(24) Capa 1
0.601041,	0.601041,	0.494975,	0.494975,	-0.132969,	0.839535,	-0.109504,	0.691382,	-0.757356,	0.385892,	-0.623705,	0.317793,	-0.757356,	-0.385892,	-0.623705,	-0.317793,	-0.132969,	-0.839535,	-0.109504,	-0.691382,	0.601041,	-0.601041,	0.494975,	-0.494975,	(48) Capa 2
0.494975,	0.494975,	0.388909,	0.388909,	-0.109504,	0.691382,	-0.086039,	0.543229,	-0.623705,	0.317793,	-0.490054,	0.249695,	-0.623705,	-0.317793,	-0.490054,	-0.249695,	-0.109504,	-0.691382,	-0.086039,	-0.543229,	0.494975,	-0.494975,	0.388909,	-0.388909,	(72) Capa 3
0.388909,	0.388909,	0.282843,	0.282843,	-0.086039,	0.543229,	-0.0625738,	0.395075,	-0.490054,	0.249695,	-0.356403,	0.181596,	-0.490054,	-0.249695,	-0.356403,	-0.181596,	-0.086039,	-0.543229,	-0.0625738,	-0.395075,	0.388909,	-0.388909,	0.282843,	-0.282843,	(96) Capa 4




```
/*
 * PACMAN1
 *
 *  Created on: 19/02/2014
 *      Author: Miguel Tlapa Juarez
 */


#include <GL/glew.h>
#include <GL/freeglut.h>
#include <iostream>
#include "cmath"
#include "Utils.h"
#include <vector>
#include <ctime>

#define RADIANS(x)  (x*M_PI)/180  /* Definir Macro para convertir Radianes */
using namespace std;
```

```
/**************Datos de Entrada ****************/
int   sides         = 5;
int   layers        = 4;
float thick         = 0.15;
/*************************************************/

/************* Calculo de Tamano de Arreglos de Color, Posicion y Bufers *********/
//int size_vertex    = (sides+1)*2*layers;          /* 48 vertices     */
//int size_colors    = size_vertex*3;               /* 144 elementos que necesito para el color RGB */
//int size_index     = (size_vertex) + (layers -1); /* Aqui considero los indices de repintado son 51 elmentos */
//int size_positions = size_vertex *2;              /* 96 elementos */


float vertexPos7[96]  = {};

GLushort index7[51]   = {};

float vertexCol7[144] = {};

/*****************************************************************************/

/******Funcion Random utilizada para el Color *********************************/

float random(){
        return (float) rand()/ RAND_MAX;
}


GLuint va7[1];              /** Definimos el numero de arreglo que necesitamos para construir el Pacman***/
GLuint bufferId7[3];       /**  Definimos el numero de buffers en este caso son 3: Posicion, Color, Inddices **/
GLuint programId7;         /**  Definimos el program ID como int  **/
GLuint vertexPosLoc7, vertexColLoc7;  /** Definimos el Vertex Posicion y del Color que seran enviados al frame shader
**/

void initShaders7() {
        GLuint vShader = Utils::compileShader("Shaders/position_color.vsh", GL_VERTEX_SHADER);
        if(!Utils::shaderCompiled(vShader)) return;

        GLuint fShader = Utils::compileShader("Shaders/color.fsh", GL_FRAGMENT_SHADER);
        if(!Utils::shaderCompiled(fShader)) return;

        programId7 = glCreateProgram();  /** Definimos que programa vamos a utilizar **/
        glAttachShader(programId7, vShader);
        glAttachShader(programId7, fShader);
        glLinkProgram(programId7); /** Linkeadmos el VShader y Fshader **/
        vertexPosLoc7 = glGetAttribLocation(programId7, "vertex_position"); /** Definimos Vertex Posicion **/
        vertexColLoc7 = glGetAttribLocation(programId7, "vertex_color");    /** Definimos Vertex Color   **/
}

void pacman_positions(float *positions, int layers, float thick, int sides)
{
        float increment   = (float)RADIANS(270)/(float)sides;          /** Definimos Increment en Radianes
**/
        float start_angle = RADIANS(45);                               /** Definimos el Angulo de Inicio en este caso
es 45 grados         **/
        float radius      = 1;                                         /** Definimos radio unitario
**/
        int   index       = 0;                                         /** Definimos el Indice de inicio del arreglo
Posicion              **/
        float angle = start_angle;                                     /** Definimos el Angulo de Inicio que es 45
grados                  **/

        for (int z= 0; z < layers ; z++){                              /** El For va estar iterando en cada capa
**/

                for (int x = 0 ; x <= sides; x++){                         /** El For va estar iterando hasta el numero
de lados               **/
```

```
                        positions[index]  = radius *cos(angle);              /** Calculo la posicion X0    en el
primer lado                     **/
                        index ++;
                        positions[index]  = radius *sin(angle);              /** Calculo la posicion Y0    en el
primer lado                     **/
                        index ++;
                        positions[index]  = (radius-thick) *cos(angle);      /** Calculo la posicion X1    en el
primer lado                     **/
                        index ++;
                        positions[index]  = (radius-thick) *sin(angle);      /** Calculo la posicion  Y1   en el
primer lado                     **/
                        index ++;

                        angle = angle + increment;                           /** Calculo el siguiente angulo = 45 +
Increment                     **/

                }
                angle = start_angle;                                    /** Reinicio el Angulo = 45 grados
**/
                radius = radius - thick;                                /** Reduzco el tamano del radio para la
siguiene iteracion           **/
        }

        for (int x = 0 ; x <= 1836; x++){
                //cout << "       " <<x;
                cout << ", " <<positions[x];
                }



}

void pacman_colors(float *colors, int layers, int sides){
        int size_vertex     = (sides+1)*2*layers;                   /** size vertex = (50 + 1 ) *2 * 9 = 918 vertices
**/
        int size_colors     = size_vertex*3;                        /** size colors = size_vertex*3 = 2754 elementos
para almacenar RGB            **/
        int control         = sides ;                               /** control, esta variable me sirve para saber si
ya termino de pintar un lado **/
        float red           = random();
        float green         = random();

        for (int x = 0 ; x < size_colors; x +=3) {                  /**  Este For inicio en 0 y termina se incrementa
cada 3 para que pinte de RGB cada vertice **/
                        if (control == sides)
                        {
                                red     = random();                             /** Vario el color rojo y verde cada
vez que termina un lado                                 **/
                                green   = random();
                                control = 0;
                        }
                        colors[x]   = red;
                        colors[x+1] = green;
                        colors[x+2] = random();                              /** Siempre pinto el color azul en cada
iteracion                                       **/
                        control ++;
                }
}


void pacman_index(GLushort *index, int layers, int sides)
{
        int last_index  =  ((sides+1)*2*layers) + (layers -1); /* 51 elements
**/
        int break_for = ((sides+1)*2*layers)-1;                /* 47
**/
```

```
        int each_side   =  ((sides+1)*2);                    /*  12
**/
        int change_side = 1;                                 /* Init change_side elements
**/
        int t = 0;

        for (int x = 0; x < last_index; x++){                /** For inicia en 0 y termina en 926 elements
**/
                index[x] = t;                                    /** Voy generando el valor de los indices
**/
                t++;
                if ((change_side%each_side) == 0 )               /** Si el valor es igual a 12
**/
                {
                        if (x >= break_for)                          /**  Si el valor de la posicon es mayor o iguak
que al numero de vertices - capas  **/
                                {

                                break;
                                }
                        index[++x] = 0xFFFF;                         /** Guadar el valor de  0xFFFF para reiniciar el
pintado                                  **/
                }
                change_side++;

        }


}

void createStrip7() {
        glGenVertexArrays(1, va7);
    glBindVertexArray(va7[0]);

        glGenBuffers(3, bufferId7);
    glBindBuffer(GL_ARRAY_BUFFER, bufferId7[0]);
    glBufferData(GL_ARRAY_BUFFER, sizeof(vertexPos7), vertexPos7, GL_STATIC_DRAW);
    glVertexAttribPointer(vertexPosLoc7, 2, GL_FLOAT, 0, 0, 0);       /** El 2 es porque son posiciones en x & y **/
    glEnableVertexAttribArray(vertexPosLoc7);

    glBindBuffer(GL_ARRAY_BUFFER, bufferId7[1]);
    glBufferData(GL_ARRAY_BUFFER, sizeof(vertexCol7), vertexCol7, GL_STATIC_DRAW);
    glVertexAttribPointer(vertexColLoc7, 3, GL_FLOAT, 0, 0, 0);      /** El 3 es porque se utiliza RGB x y z    **/
    glEnableVertexAttribArray(vertexColLoc7);

    glBindBuffer(GL_ELEMENT_ARRAY_BUFFER,bufferId7[2]);
    glBufferData(GL_ELEMENT_ARRAY_BUFFER,sizeof(index7),index7, GL_STATIC_DRAW);
    glEnable(GL_PRIMITIVE_RESTART);
    glPrimitiveRestartIndex(0xFFFF);                                 /** El valor del Indice de reinicio siempre se pone
el numero mayor definido por el valor de la variable **/
}

void drawStrip7() {
        glClear(GL_COLOR_BUFFER_BIT);
        glUseProgram(programId7);
        glBindVertexArray(va7[0]);
        glBindBuffer(GL_ELEMENT_ARRAY_BUFFER,bufferId7[2]);
        glDrawElements(GL_TRIANGLE_STRIP,51,GL_UNSIGNED_SHORT,0);     /** Se Define el numero de Triangle STRIP =
Numero de Elementos del Index **/
        glutSwapBuffers();
}

void exitFunc7(unsigned char key, int x, int y) {
    if (key == 27) {
        glDeleteVertexArrays(1, va7);
        exit(0);
    }
}
```

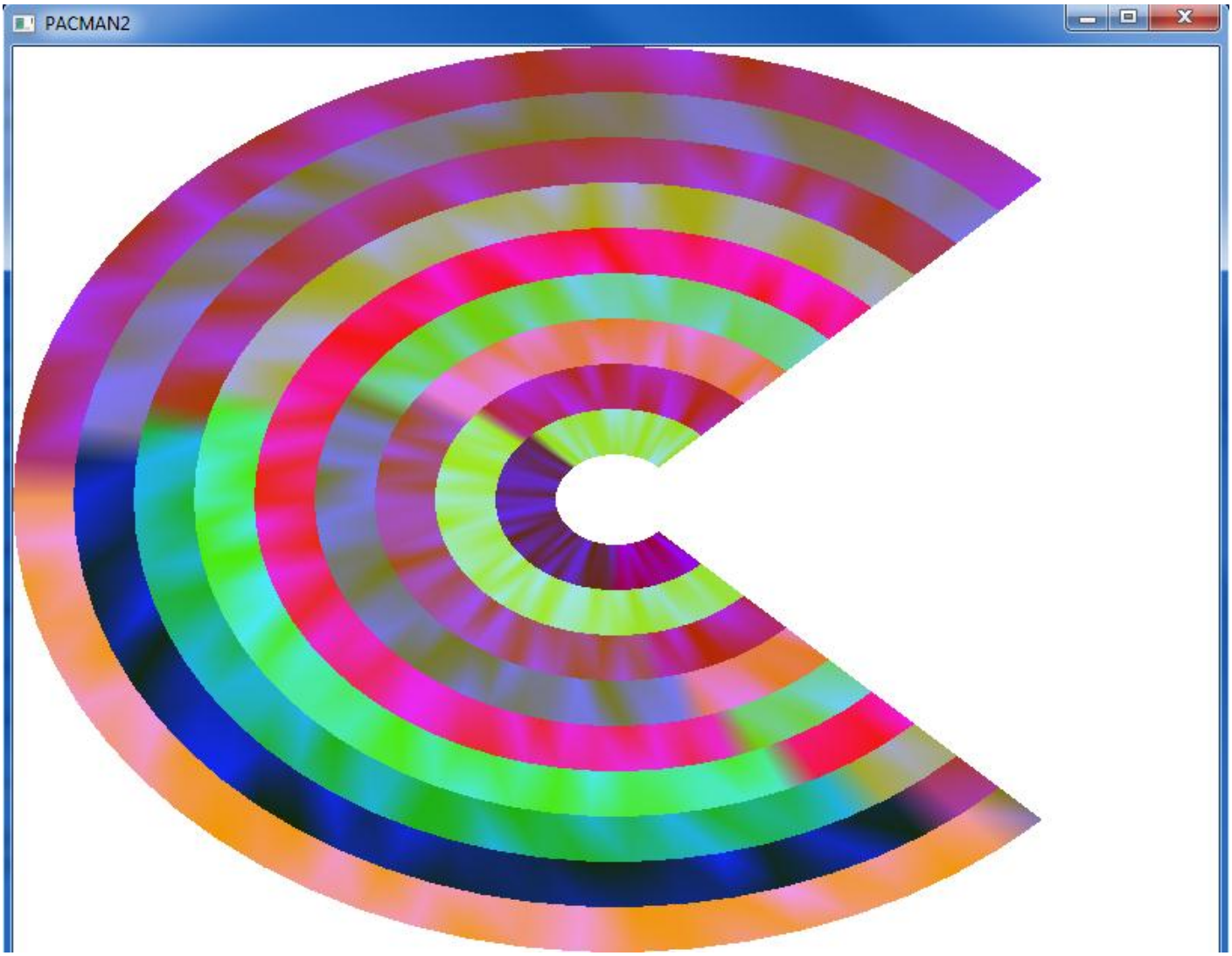
```
int main(int argc, char **argv) {
        srand(time(NULL));
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_DOUBLE);
    glutInitWindowSize(800, 600);
    glutInitWindowPosition(300, 300);


    glutCreateWindow("PACMAN2");
    glutDisplayFunc(drawStrip7);
    glutKeyboardFunc(exitFunc7);
    glewInit();
    initShaders7();
    pacman_positions(vertexPos7, layers, thick,sides);
    pacman_colors(vertexCol7, layers, sides);
    pacman_index(index7,layers, sides);
    createStrip7();
    glClearColor(1, 1, 1, 1.0);
    glutMainLoop();
    return 0;
}
```

++

PACMAN2

La explicacion de como hacer esta figura obedece a los mismos principios que la anterior.

++




```
/*
 * PACMAN2
 *
 *  Created on: 19/02/2014
 *      Author: Miguel Tlapa Juarez
 */



#include <GL/glew.h>
#include <GL/freeglut.h>
#include <iostream>
```

```
#include "cmath"

#include "Utils.h"

#include <vector>

#include <ctime>


#define RADIANS(x)  (x*M_PI)/180  /* Definir Macro para convertir Radianes */

using namespace std;


/**************Datos de Entrada ****************/

int   sides         = 50;

int   layers        = 9;

float thick         = 0.10;

/*****************************************/

/************* Calculo de Tamano de Arreglos de Color, Posicion y Bufers *********/

//int size_vertex    = (sides+1)*2*layers;          /* 918 vertices     */

//int size_colors    = size_vertex*3;               /* 2754 elementos que necesito para el color RGB */

//int size_index     = (size_vertex) + (layers -1); /* Aqui considero los indices de repintado son 926 elmentos */

//int size_positions = size_vertex *2;              /* 1836 elementos */

float vertexPos7[1836]  = {};

GLushort index7[926]   = {};

float vertexCol7[2754] = {};

/*********************************************************************************/

/******Funcion Random utilizada para el Color *********************************/

float random(){

        return (float) rand()/ RAND_MAX;

}

GLuint va7[1];              /** Definimos el numero de arreglo que necesitamos para construir el Pacman***/

GLuint bufferId7[3];       /**  Definimos el numero de buffers en este caso son 3: Posicion, Color, Inddices **/

GLuint programId7;         /**  Definimos el program ID como int  **/

GLuint vertexPosLoc7, vertexColLoc7;  /** Definimos el Vertex Posicion y del Color que seran enviados al frame shader
**/
```

```
void initShaders7() {

	GLuint vShader = Utils::compileShader("Shaders/position_color.vsh", GL_VERTEX_SHADER);

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

	GLuint fShader = Utils::compileShader("Shaders/color.fsh", GL_FRAGMENT_SHADER);

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


	programId7 = glCreateProgram();  /** Definimos que programa vamos a utilizar **/

	glAttachShader(programId7, vShader);

	glAttachShader(programId7, fShader);

	glLinkProgram(programId7); /** Linkeadmos el VShader y Fshader **/

	vertexPosLoc7 = glGetAttribLocation(programId7, "vertex_position"); /** Definimos Vertex Posicion **/

	vertexColLoc7 = glGetAttribLocation(programId7, "vertex_color");    /** Definimos Vertex Color   **/

}


void pacman_positions(float *positions, int layers, float thick, int sides)

{

	float increment   = (float)RADIANS(270)/(float)sides;          /** Definimos Increment en Radianes
**/

	float start_angle = RADIANS(45);                               /** Definimos el Angulo de Inicio en este caso
es 45 grados         **/

	float radius      = 1;                                         /** Definimos radio unitario
**/

	int   index       = 0;                                         /** Definimos el Indice de inicio del arreglo
Posicion              **/

	float angle = start_angle;                                     /** Definimos el Angulo de Inicio que es 45
grados                  **/


	for (int z= 0; z < layers ; z++){                              /** El For va estar iterando en cada capa
**/


		for (int x = 0 ; x <= sides; x++){                         /** El For va estar iterando hasta el numero
de lados               **/
```

```
                        positions[index]  = radius *cos(angle);              /** Calculo la posicion X0    en el
primer lado                     **/

                        index ++;

                        positions[index]  = radius *sin(angle);              /** Calculo la posicion Y0    en el
primer lado                     **/

                        index ++;

                        positions[index]  = (radius-thick) *cos(angle);      /** Calculo la posicion X1    en el
primer lado                     **/

                        index ++;

                        positions[index]  = (radius-thick) *sin(angle);      /** Calculo la posicion  Y1   en el
primer lado                     **/

                        index ++;



                        angle = angle + increment;                           /** Calculo el siguiente angulo = 45 +
Increment                     **/



                }

                angle = start_angle;                                    /** Reinicio el Angulo = 45 grados
**/

                radius = radius - thick;                                /** Reduzco el tamano del radio para la
siguiene iteracion           **/

        }


        for (int x = 0 ; x <= 1836; x++){

                //cout << "       " <<x;

                cout << ", " <<positions[x];

                }









}


void pacman_colors(float *colors, int layers, int sides){
```

```
        int size_vertex     = (sides+1)*2*layers;                   /** size vertex = (50 + 1 ) *2 * 9 = 918 vertices
**/

        int size_colors     = size_vertex*3;                        /** size colors = size_vertex*3 = 2754 elementos
para almacenar RGB            **/

        int control         = sides ;                               /** control, esta variable me sirve para saber si
ya termino de pintar un lado **/

        float red           = random();

        float green         = random();



        for (int x = 0 ; x < size_colors; x +=3) {                  /**  Este For inicio en 0 y termina se incrementa
cada 3 para que pinte de RGB cada vertice **/

                        if (control == sides)

                        {

                                red     = random();                             /** Vario el color rojo y verde cada
vez que termina un lado                                 **/

                                green   = random();

                                control = 0;

                        }

                        colors[x]   = red;

                        colors[x+1] = green;

                        colors[x+2] = random();                         /** Siempre pinto el color azul en cada
iteracion                                       **/

                        control ++;

                }

}




void pacman_index(GLushort *index, int layers, int sides)

{

        int last_index  =  ((sides+1)*2*layers) + (layers -1); /* 926 elements
**/

        int break_for = ((sides+1)*2*layers)-1;                /* 917
**/

        int each_side   =  ((sides+1)*2);                      /*  102
**/
```

```
        int change_side = 1;                                    /* Init change_side elements
**/

        int t = 0;

        for (int x = 0; x < last_index; x++){                   /** For inicia en 0 y termina en 926 elements
**/

                index[x] = t;                                       /** Voy generando el valor de los indices
**/

                t++;

                if ((change_side%each_side) == 0 )                  /** Si el valor es igual a 102
**/

                {

                        if (x >= break_for)                             /**  Si el valor de la posicon es mayor o iguak
que al numero de vertices - capas  **/

                                {



                                break;

                                }

                        index[++x] = 0xFFFF;                            /** Guadar el valor de  0xFFFF para reiniciar el
pintado                             **/

                }

                change_side++;



        }




}


void createStrip7() {

        glGenVertexArrays(1, va7);

    glBindVertexArray(va7[0]);



        glGenBuffers(3, bufferId7);

    glBindBuffer(GL_ARRAY_BUFFER, bufferId7[0]);

    glBufferData(GL_ARRAY_BUFFER, sizeof(vertexPos7), vertexPos7, GL_STATIC_DRAW);
```

```
    glVertexAttribPointer(vertexPosLoc7, 2, GL_FLOAT, 0, 0, 0);       /** El 2 es porque son posiciones en x & y **/

    glEnableVertexAttribArray(vertexPosLoc7);


    glBindBuffer(GL_ARRAY_BUFFER, bufferId7[1]);

    glBufferData(GL_ARRAY_BUFFER, sizeof(vertexCol7), vertexCol7, GL_STATIC_DRAW);

    glVertexAttribPointer(vertexColLoc7, 3, GL_FLOAT, 0, 0, 0);      /** El 3 es porque se utiliza RGB x y z    **/

    glEnableVertexAttribArray(vertexColLoc7);


    glBindBuffer(GL_ELEMENT_ARRAY_BUFFER,bufferId7[2]);

    glBufferData(GL_ELEMENT_ARRAY_BUFFER,sizeof(index7),index7, GL_STATIC_DRAW);

    glEnable(GL_PRIMITIVE_RESTART);

    glPrimitiveRestartIndex(0xFFFF);                                  /** El valor del Indice de reinicio siempre se pone
el numero mayor definido por el valor de la variable **/

}


void drawStrip7() {

        glClear(GL_COLOR_BUFFER_BIT);

        glUseProgram(programId7);

        glBindVertexArray(va7[0]);

        glBindBuffer(GL_ELEMENT_ARRAY_BUFFER,bufferId7[2]);

        glDrawElements(GL_TRIANGLE_STRIP,926,GL_UNSIGNED_SHORT,0);     /** Se Define el numero de Triangle STRIP =
Numero de Elementos del Index **/

        glutSwapBuffers();

}


void exitFunc7(unsigned char key, int x, int y) {

    if (key == 27) {

        glDeleteVertexArrays(1, va7);

        exit(0);

    }

}
```

```
int main(int argc, char **argv) {
        srand(time(NULL));
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_DOUBLE);
    glutInitWindowSize(800, 600);
    glutInitWindowPosition(300, 300);



    glutCreateWindow("PACMAN2");
    glutDisplayFunc(drawStrip7);
    glutKeyboardFunc(exitFunc7);
    glewInit();
    initShaders7();
    pacman_positions(vertexPos7, layers, thick,sides);
    pacman_colors(vertexCol7, layers, sides);
    pacman_index(index7,layers, sides);
    createStrip7();
    glClearColor(1, 1, 1, 1.0);
    glutMainLoop();
    return 0;
}
```