**ATRAPANDO HUEVOS**

Jesús A. Quintana Cerón,

George S. Rosales Tintaya,

Ruth Valcárcel Sierra,

Josue Choque Gomez,

Luis Beltrán Cansaya Huamán

***RESUMEN:***

*El presente trabajo tocó distintos aspectos relacionados al baloncesto de su variante el juego del 21, desde los principios y reglas de la jugabilidad hasta la implementación del juego, usando la librería OpenGL con una metodología iterativa incremental, las cuales nos permiten lograr el objetivo de nuestro trabajo. En el cuerpo del trabajo también se detallan las reglas necesarias para participar en el juego, como también describimos los materiales ya sean hardware o software necesarios para la realización del trabajo*

*Al final damos una retroalimentación respecto al trabajo realizado.*

***ABSTRACT:***

*The present work touched different aspects related to basketball of its variant of the game of 21, from the principles and rules of the gameplay to the implementation of the game, using the OpenGL library with an incremental iterative methodology, which allow us to achieve the objective of our worked. The body of the work also details the rules necessary to participate in the game, as well as describing the materials, whether hardware or software, necessary to carry out the work.*

*At the end we give feedback on the work done.*

***PALABRA CLAVE****: Huevos, juego, cesta, OpenGL, GLUT, metodología iterativa incremental.*

**INTRODUCCIÓN**

La historia del basquetbol empezó a escribirse en Estados Unidos, el país líder de este deporte. El baloncesto es el único deporte de origen norteamericano, aunque su paternidad se debe a James Naismith, un nativo de Canadá. Los inicios del básquetbol se datan en el año 1891, en el que Naismith ideó sus fundamentos como una salida ingeniosa para escabullir los rigores del invierno en Estados Unidos.

El 21 es un tipo de juego de baloncesto que se usa normalmente cuando no somos los suficientes para jugar un partido de baloncesto normal.

Es un juego muy común en las escuelas de baloncesto ya que es divertido a la vez que puedes practicar el tiro desde distintas distancias.

Consiste en alternar los jugadores y lanzar a canasta por turnos, hasta llegar a los 21 puntos.

El siguiente trabajo desarrollo de un juego en 2D consiste en diseñar el juego haciendo uso la librería GLUT, con los cuales se construyen las figuras y con ellos hacer dibujos mediante el ajuste de coordenadas haciendo que aparente a algún animal en este caso de un pato formado por figuras geométricas, así como el hexágono, cuadrado y rectángulo; así mismo las nubes, el sol y el huevo formados por círculos de distintos tamaños, en cuanto a la cesta con trapecio invertido. Una vez diseñado se hace que las figuras tengan movimiento, el huevo que cae y la cesta o canasta que se mueve de derecha a izquierda y viceversa. El objetivo es atrapar en la cesta o canasta los huevos que caen aleatoriamente de tres patos. Así mismo en cada nivel la velocidad del huevo aumenta.

El objetivo principal de este trabajo es que los participantes a partir de los 8 años puedan desarrollar habilidades de puntería. En cuanto a este juego 2D en atrapar huevos en la cesta permite desarrollar la concentración sobre todo en los niños.

**PROBLEMA**

En la actualidad, múltiples investigaciones han encontrado las ventajas de su uso. Ciertamente, son un apoyo tanto en el ámbito escolar, como en la rehabilitación o estimulación del funcionamiento cognitivo.

Sin embargo, en el presente trabajo se detectaron diversos problemas a la hora de querer jugar algún videojuego, por el mismo hecho de que la gran mayoría de videojuegos están en línea, esto quiere decir que para jugarlo es necesario tener acceso a internet. Así mismo el costo de los juegos es una limitación para muchos jugadores en especial los niños ya que en su mayoría no lo pueden comprar.

A partir de los problemas detectados en la presente investigación se propone la elaboración preliminar de un videojuego en 2D titulado “Atrapando huevos huevos”, cuyo objetivo es que el juego sea accesible para cualquier jugador (gratuito) y también que no requiera acceso a internet (modo offline).

**METODOLOGÍA**

El proceso de realización de este trabajo es el modelo iterativo incremental, el cual nos permitirá generar incrementos operacionales, los cuales permitirán una idea del avance y cumplimiento de los objetivos del trabajo.

Así mismo mencionamos que esta metodología sólo se tomará como una referencia, debido a que el proceso de desarrollo del juego del 21 del baloncesto es muy distinto al desarrollo tradicional del juego, en donde el proceso de desarrollo es más flexible y un poco más informal. Sin embargo, se utilizarán aspectos de la metodología, con el objetivo de promover la realización de juegos en entorno 2D.

El desarrollo consta de 8 incrementos, iniciando por el desarrollo de la pantalla, el sol, las nubes, (líneas y terreno) que seccionan la pantalla, los patos en representación de lanzamiento de balón, pero en este caso el huevo, el huevo que simula el balón, la cesta y la implementación de los contadores para saber la puntuación del juego en cada nivel.

<http://repobib.ubiobio.cl/jspui/bitstream/123456789/1605/1/Aravena%20Vergara%2C%20Cristian%20Enrique.pdf>

**MATERIAL Y MÉTODOS.**

Se viene analizando y observando que hoy en día los seres humanos dependen de alguna tecnología ya sea una laptop, un computador o un celular, es por esta situación que se desea implementar y desarrollar el

“Juego de Huevos “en la ciudad de Abancay, Provincia de Abancay, departamento Apurímac-PERU.

**MATERIALES**

Hardware

* Nombre: ASUS-ROG-strix-g15-g512

Procesador: intel Core(TM) i5-10750 H CPU @ 12 gen 3.5 GHZ a 4.5 GHZ

RAM: 8.0 GB (7.94 GB utilizable)

Tipo de Sistema: Sistema operativo de 64 bits, procesador x64

Edición: Windows 11 pro.

Versión: 21H2

* Nombre: ASUS-ROG-Sephirus

Procesador: Core(TM) i7-10750 H CPU @ 3.5 4.7 GHz

RAM: 16.0 GB

Tipo de Sistema: Sistema operativo de 64 bits, procesador basado en x64

Edición: Windows 11 pro

Versión: 21H2

Software

* Visual Studio 2019
* Lenguaje de programación utilizado: C++ con OpenGL - GLUT

**MÉTODOS**

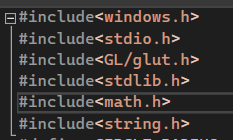
**OpenGL**

<https://www.profesionalreview.com/2019/11/15/opengl/>

Se considera principalmente como una API que nos proporciona un gran conjunto de funciones que podemos usar para manipular gráficos e imágenes 3D y 2D. Sin embargo, por sí solo no es simplemente una API, sino una especificación, desarrollada y mantenida por el Grupo Khronos

Los que desarrollan las bibliotecas OpenGL suelen ser los fabricantes de tarjetas gráficas. Cada GPU admite versiones específicas de esta API, que son las versiones de OpenGL desarrolladas específicamente para esa tarjeta gráfica.

**LIBRERÍAS UTILIZADAS**

****

**Figura 1:** Librerías que se utilizaran en el trabajo (imagen propia).

**CÓDIGO**

**Variables**

// variables globales para el tamaño de la ventana

int a = 600, b = 650;

// conjunto de funciones y procedimientos del juego

void myinit();

void start\_screen(int, int);

void cloud1();

void egg();

void basket(int, int);

void duck(int, int);

void print\_score();

void egg\_start();

void color();

void score();

void display(void);

void basket\_set(int, int);

void myReshape(int, int);

void keys(unsigned char, int, int);

void menu(int);

**Coordenadas**

// a, b es la coordenada superior izquierda de la paleta

void myinit()

{

glViewport(0, 0, a, b);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0, (GLdouble)a, 0, (GLdouble)b);

glMatrixMode(GL\_MODELVIEW);

glutPostRedisplay();

}

**Inicio del juego**

// la función para imprimir las opciones del juego.

void start\_screen(int i, int j)

{

int k;

char cat[6] = "HUEVO";

char orr[11] = "Atrapando";

char hatch[7] = "Juego";

char start[29] = "Presiona 'S' para comenzar";

glColor3f(0, 1, 0);

glRasterPos2i(150, 320);

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

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18, cat[k]);

glColor3f(1, 1, 1);

glRasterPos2i(200, 320);

for (k = 0; k < 9; k++)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18, orr[k]);

glColor3f(1, 0, 0);

glRasterPos2i(300, 320);

for (k = 0; k < 5; k++)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18, hatch[k]);

glColor3f(1, 1, 0);

glRasterPos2i(210, 200);

for (k = 0; k < 20; k++)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18, start[k]);

glColor3f(1, .5, .5);

}

**Funciones para el paisaje**

// se construye el sol de color rojo y se posiciona

void sun()

{

float theta;

GLfloat angle;

glLineWidth(1.5);

glColor3f(255, 0, 0);

glBegin(GL\_POLYGON);

for (int i = 0; i < 360; i++)

{

theta = i \* PI \* i / 180;

glVertex2f(500 + 80 \* cos(theta) / 2, 600 + 80 \* sin(theta) / 2);

}

glEnd();

glFlush();

}

//se construye la primera nube de 2 totales

void cloud1()

{

float theta;

GLfloat angle;

glLineWidth(1.5);

glColor3f(1, 1, 1);

glBegin(GL\_POLYGON);

for (int i = 0; i < 360; i++)

{

theta = i \* PI \* i / 180;

glVertex2f(100 + 50 \* cos(theta) / 2, 590 + 50 \* sin(theta) / 2);

}

glEnd();

//GLfloat angle;

glLineWidth(1.5);

glColor3f(1, 1, 1);

glBegin(GL\_POLYGON);

for (int i = 0; i < 360; i++)

{

theta = i \* PI \* i / 180;

glVertex2f(130 + 50 \* cos(theta) / 2, 580 + 50 \* sin(theta) / 2);

}

glEnd();

glLineWidth(1.5);

glColor3f(1, 1, 1);

glBegin(GL\_POLYGON);

for (int i = 0; i < 360; i++)

{

theta = i \* PI \* i / 180;

glVertex2f(140 + 50 \* cos(theta) / 2, 600 + 50 \* sin(theta) / 2);

}

glEnd();

glLineWidth(1.5);

glColor3f(1, 1, 1);

glBegin(GL\_POLYGON);

for (int i = 0; i < 360; i++)

{

theta = i \* PI \* i / 180;

glVertex2f(170 + 50 \* cos(theta) / 2, 590 + 50 \* sin(theta) / 2);

}

glEnd();

glFlush();

}

//se construye la segunda f nube de 2 totales

void cloud2()

{

float theta;

glLineWidth(1.5);

glColor3f(1, 1, 1);

glBegin(GL\_POLYGON);

for (int i = 0; i < 360; i++)

{

theta = i \* PI \* i / 180;

glVertex2f(420 + 45 \* cos(theta) / 2, 540 + 45 \* sin(theta) / 2);

}

glEnd();

glLineWidth(1.5);

glColor3f(1, 1, 1);

glBegin(GL\_POLYGON);

for (int i = 0; i < 360; i++)

{

theta = i \* PI \* i / 180;

glVertex2f(455 + 45 \* cos(theta) / 2, 550 + 45 \* sin(theta) / 2);

}

glEnd();

glLineWidth(1.5);

glColor3f(1, 1, 1);

glBegin(GL\_POLYGON);

for (int i = 0; i < 360; i++)

{

theta = i \* PI \* i / 180;

glVertex2f(455 + 45 \* cos(theta) / 2, 530 + 45 \* sin(theta) / 2);

}

glEnd();

glLineWidth(1.5);

glColor3f(1, 1, 1);

glBegin(GL\_POLYGON);

for (int i = 0; i < 360; i++)

{

theta = i \* PI \* i / 180;

glVertex2f(490 + 45 \* cos(theta) / 2, 540 + 45 \* sin(theta) / 2);

}

glEnd();

glFlush();

}

// se construye la línea que separa a los patos en el área de drop

void line(int i, int j)

{

glBegin(GL\_QUADS);

glColor3f(1.0, .5, 0.5);

glVertex2f(0.0 + i, 10.0 + j);

glVertex2f(0.0 + i, 15.0 + j);

glVertex2f(600.0 + i, 15.0 + j);

glVertex2f(600.0 + i, 10.0 + j);

glEnd();

glFlush();

}

// se construye la parte del fondo en dos partes separadas

void backk(int i, int j)

{

glColor3f(0, .5, 1);

glBegin(GL\_QUADS);

glVertex2f(0.0 + i, 0.0 + j);

glVertex2f(600.0 + i, 0.0 + j);

glVertex2f(600.0 + i, -500 + j);

glVertex2f(0.0 + i, -500 + j);

glEnd();

glFlush();

}

void ground(int i, int j)

{

glBegin(GL\_QUADS);

glColor3f(0, 1.0, 0);

glVertex2f(0.0 + i, 0.0 + j);

glVertex2f(600.0 + i, 0.0 + j);

glVertex2f(600.0 + i, -j);

glVertex2f(0.0 + i, -j);

glEnd();

}

//se construye el modelo base del huevo.

void egg()

{

float x, y, z;

int t;

glColor3f(1.0, 10, 1.0);

glBegin(GL\_POLYGON);

for (t = 0; t <= 360; t += 1)

{

x = egg\_xc + 8 \* (cos(t));

y = egg\_yc + 16 \* (sin(t));

z = 0;

glVertex3f(x, y, z);

}

glEnd();

}

// se construye el modelo del cesto en que se recepcionan los huevos.

void basket(int i, int j)

{

j = 10;

if (i >= a - 60)i = a - 60;

glColor3f(1.0, 0.0, 0.0);

glBegin(GL\_QUADS);

glVertex2f(0.0 + i, 30.0 + j);

glVertex2f(10.0 + i, 10.0 + j);

glVertex2f(50.0 + i, 10.0 + j);

glVertex2f(60.0 + i, 30.0 + j);

glEnd();

}

// se construye el modelo de los patos (dckys)

void duck(int i, int j)

{

int h;

glColor3f(1.0, 1.0, 0.0);

glBegin(GL\_POLYGON);

glVertex2f(45 + i, 45 + j);

glVertex2f(70 + i, 20 + j);

glVertex2f(95 + i, 20 + j);

glVertex2f(120 + i, 45 + j);

glVertex2f(95 + i, 70 + j);

glVertex2f(70 + i, 70 + j);

glVertex2f(95 + i, 95 + j);

glVertex2f(82.5 + i, 107.5 + j);

glVertex2f(32.5 + i, 57.5 + j);

glEnd();

glFlush();

for (h = 0; h < 13; h += 4)

{

glBegin(GL\_LINES);

glColor3f(0.7, 0.4, 0);

glVertex2f(57.5 + h + i, 52.5 + h + j);

glVertex2f(100 + h + i, 30 + h + j);

glEnd();

glFlush();

}

glColor3f(0.0, 1.0, 0.0);

glBegin(GL\_POLYGON);

glVertex2f(82.5 + i, 107.5 + j);

glVertex2f(65 + i, 107.5 + j);

glVertex2f(50 + i, 95 + j);

glVertex2f(70 + i, 95 + j);

glEnd();

glFlush();

glColor3f(0.0, 0.0, 0.0);

glPointSize(5);

glBegin(GL\_POINTS);

glVertex2f(76 + i, 101 + j);

glEnd();

glColor3f(0.0, 1.0, 0.0);

glBegin(GL\_LINE\_LOOP);

glVertex2f(72.5 + i, 107.5 + j);

glVertex2f(67.5 + i, 112.5 + j);

glVertex2f(72.5 + i, 110 + j);

glVertex2f(77.5 + i, 112.5 + j);

glEnd();

glFlush();

}

//se construye el modelo de la vista al score actual del juego

void print\_score()

{

printf("\nNivel alcanzado: %d\n\n", level\_count);

printf("\nNo. de huevos caídos= %d \n", dropped\_eggs);

printf("\nNo. de huevos atrapados = %d\n", eggs\_caught);

printf("\nNo. de huevos perdidos = %d\n", missed\_eggs);

getchar();

exit(0);

}

// se construye el funcionamiento del juego en base a los huevos,

void egg\_start()

{

egg\_yc = 375;

if (missed\_eggs >= 10)

{

printf("\n\n\t\t\t\tJUEGO TERMINADO\n\n");

print\_score();

}

dropped\_eggs++;

switch (rand() % 9)

{

case 0:egg\_xc = 115; break;

case 1:egg\_xc = 255; break;

case 2:egg\_xc = 390; break;

case 5:egg\_xc = 115; break;

case 3:egg\_xc = 255; break;

case 4:egg\_xc = 390; break;

case 7:egg\_xc = 115; break;

case 6:egg\_xc = 255; break;

case 8:egg\_xc = 390; break;

}

}

// se construye la función encargada de contar los huevos recogidos, huevos perdidos, y huevos totales

void score()

{

if (egg\_yc <= 50 && (egg\_xc >= basket\_x && egg\_xc <= basket\_x + 60))

{

printf("\a");

eggs\_caught++;

egg\_yc = -10;

}

missed\_eggs = dropped\_eggs - eggs\_caught;

}

//se construye el modelo

**Función Principal**

// se construye la función principal encargada de construir el juego c o los modelos antes mencionados en la documentación del código, se usa cada una de funciones y procedimientos dándole un valor base a cada uno de ellos

void display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

ground(0, 650);

backk(0, 650);

duck(40, 375);

duck(180, 375);

duck(320, 375);

sun();

cloud1();

cloud2();

line(0, 375);

int i;

char z[12] = "KhNishad";

char level1[12] = "NIVEL 1";

char level2[12] = "NIVEL 2";

char level3[12] = "NIVEL 3";

char level4[12] = "NIVEL 4";

if (s >= 1)

{

glColor3f(0, 0, 1);

glRasterPos2i(10, 10);

for (i = 0; i < 12; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, z[i]);

if (level\_count == 1)

{

glColor3f(1, 1, 1);

glRasterPos2i(500, 300);

for (i = 0; i < 12; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, level1[i]);

}

else if (level\_count == 2)

{

glColor3f(1, 0, 0);

glRasterPos2i(500, 300);

for (i = 0; i < 12; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, level2[i]);

}

else if (level\_count == 3)

{

glColor3f(0, 1, 1);

glRasterPos2i(500, 300);

for (i = 0; i < 12; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, level3[i]);

}

if (level\_count == 4)

{

glColor3f(1, 0, 1);

glRasterPos2i(500, 300);

for (i = 0; i < 12; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, level4[i]);

}

if (egg\_yc <= 10)

egg\_start();

egg();

basket(basket\_x, basket\_y);

if (eggs\_caught >= 10)

{

egg\_yc -= speed\_2;

level\_count = 2;

}

if (eggs\_caught >= 20)

{

egg\_yc -= speed\_3;

level\_count = 3;

}

if (eggs\_caught >= 30)

{

egg\_yc -= speed\_4;

level\_count = 4;

}

else

egg\_yc -= speed\_1;

score();

}

else

start\_screen(40, 300);

glFlush();

glutSwapBuffers();

}

**Funciones tras ganar o perder el juego, de inicio y de funcionamiento de la cesta**

// se construye la función encargada de colocar el cesto en la posición del mouse

void basket\_set(int a, int b)

{

basket\_x = a;

basket\_y = b;

glutPostRedisplay();

}

void myReshape(int w, int h)

{

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, (GLdouble)w, 0.0, (GLdouble)h);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glViewport(0, 0, w, h);

a = w;

b = h;

}

// se construye la función encargada de designar las teclas para el funcionamiento de las teclas

void keys(unsigned char key, int x, int y)

{

if (key == 'q' || key == 'Q')

{

printf("\n\n\n\t\tSALIR DEL JUGADOR\n\n");

print\_score();

}

if (key == 's' || key == 'S') s += 1;

if (key == 'a' || key == 'A')

{

egg\_xc -= 10;

if (egg\_xc <= 0) egg\_xc = 10;

}

if (key == 'd' || key == 'D')

{

egg\_xc += 10;

if (egg\_xc >= 500) egg\_xc = 490;

}

}

// se construye el siguiente modelo si se presiona una tecla o se pierde o gana el juego, para uno de los casos, para ello se usa un case

void menu(int id)

{

switch (id)

{

case 1: s += 1;

break;

case 2:print\_score();

break;

case 3: printf("\n\n\n\t\tSALIR POR JUGADOR\n");

break;

default:exit(0);

}

glutPostRedisplay();

}

// no se documenta lo que va dentro del int main (ya que ahí irá la explicación del funcionamiento del juego que será mostrado en la terminal (no dentro del juego).

**REGLAS DEL JUEGO**

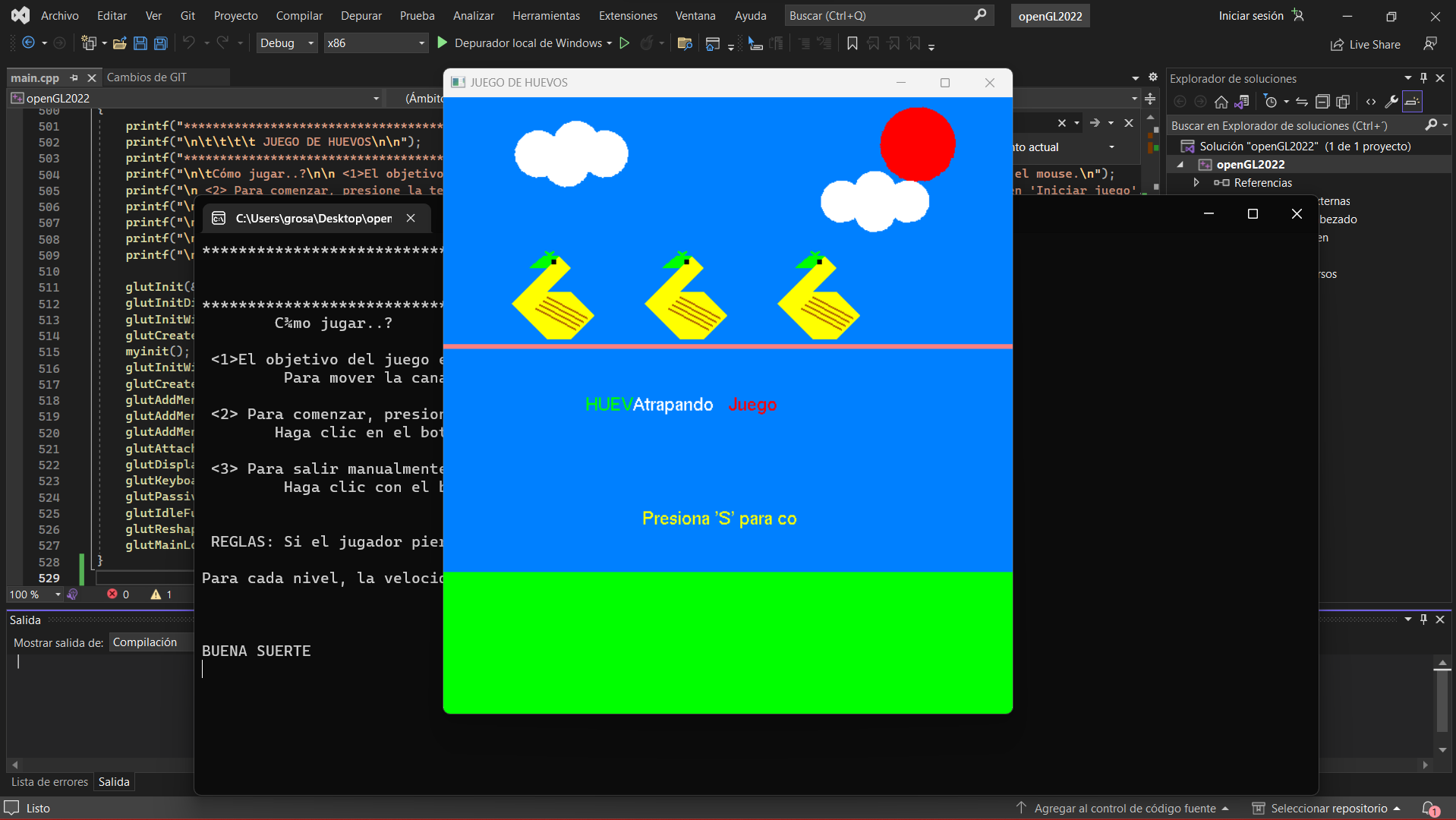
* El equipamiento básico es una canasta que en nuestro caso será un trapecio invertido, tres patos formados por figuras geométricas y la pelota con forma de elipsoide que simula al huevo.
* Para mover la cesta o canasta usa el mouse. También la opción de usar las teclas ‘A’ y ‘D’.
* La pelota tiene que ser elipsoide en representación al huevo.
* Una vez que se presione ‘S’ comenzará el juego haciendo que caiga huevo de forma aleatoria de uno de los patos. El huevo puede caer de manera vertical, pero es dinámico ya que se puede mover con las teclas hasta la dirección de la cesta.
* Si el jugador pierde 10 huevos, entonces 'Finaliza el juego'
* La velocidad del huevo aumentará en cada nivel y para ello es necesario atrapar 9 huevos
* Por el momento el juego consta de 4 niveles siendo finito los niveles (1, 2, 3) y para el nivel 4 en de manera infinita, así mismo la velocidad es constante.

**RESULTADOS**

Por tanto, una vez terminada la codificación del trabajo en el lenguaje de programación c + + utilizando visual studio mostramos los resultados en tres tiempos según la explicación de las reglas mencionadas anteriormente.

**INICIO DEL JUEGO**

Figura 2: inicio del juego



**DURANTE EL JUEGO**

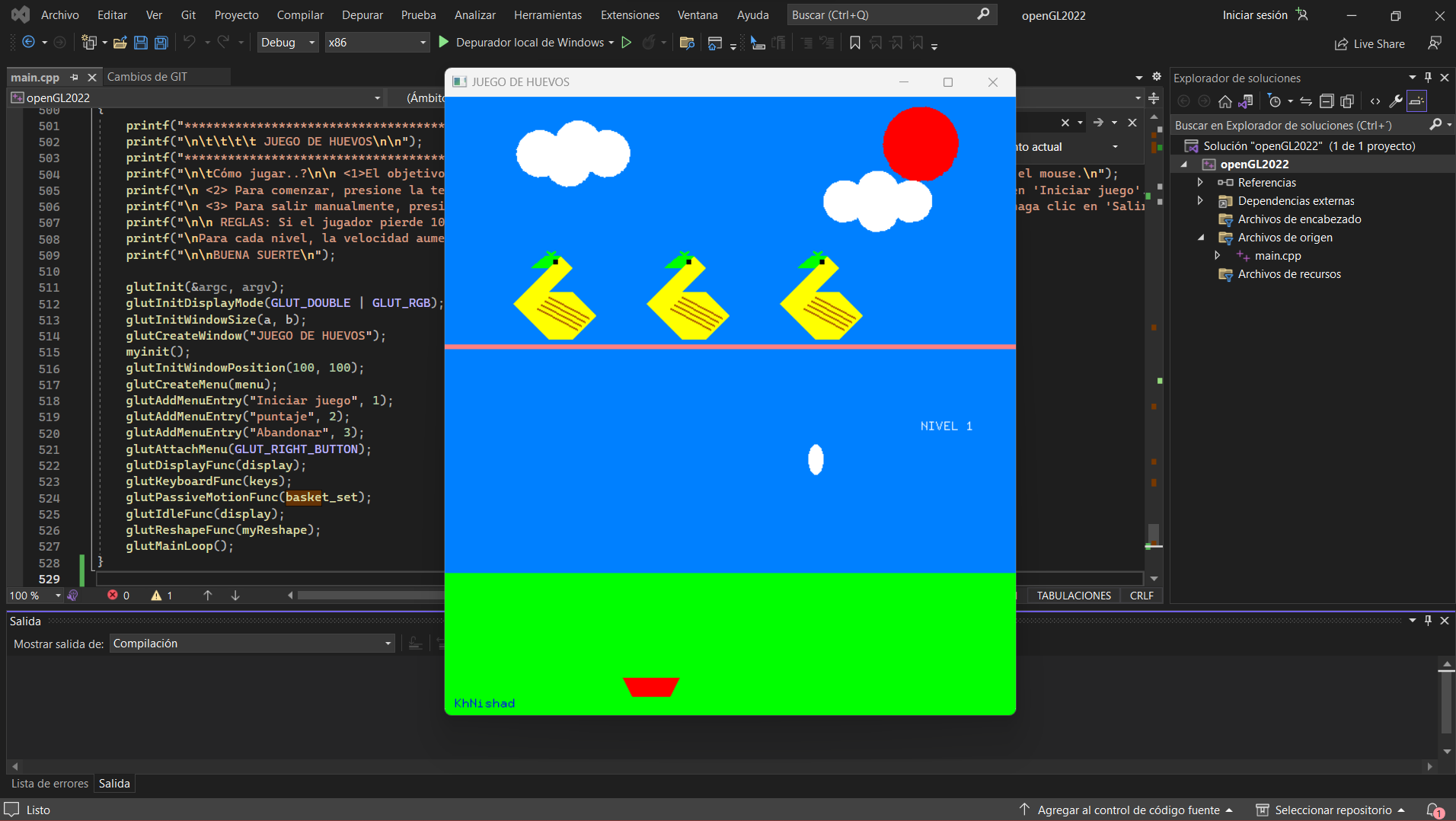


Figura 3: Durante el juego

**FIN DEL JUEGO**

****

Figura 4: Fin del juego

**TABLA**

|  |  |
| --- | --- |
| **FIGURAS** | **DESCRIPCIÓN** |
| Figura 1 | En la figura se puede observar que se dio inicio al juego, por lo tanto, el marcador está en 0 y el juego está pausado. |
| Figura 2 | En la figura se puede observar que el juego está en proceso y el marcador está activo. |
| Figura 3 | En la figura se puede observar que el juego está concluido, y ya que se ha detenido y no logra poder hacer nada, en la terminal se muestra un anuncio declarando el juego terminado y una muestra de las estadísticas |

**CONCLUSIÓN**

Eso es todo por este trabajo. El” Juego de huevos “es un juego muy familiar e interesante, por lo que realmente no había muchas cosas nuevas que añadir. Lo más importante que hicimos aquí fue desarrollar nuestro juego, que es fácil de usar y el juego terminará cuando el participante llegue a perder 10 puntos o lo que significa dejar caer 10 huevos fuera de la cesta, lo cual no es una tarea tan sencilla por que la velocidad se incrementa según los puntos acumulados Así que creo que a la mayoría de los participantes les gustara y definitivamente disfrutarán el juego.

Gracias.