Cindy Marie Méndez Avilés

COMP4046-086: Computer Graphics
Prof. Alcibiades Bustillo Zárate
May 16, 2021

Final Report: Flappy Box

Description of the Project

Flappy Box is quite similar to the popular mobile game Flappy Bird. The game will be a side-scroller where the player controls a box, attempting to jump between rows of pipes without coming into contact with them. If the player touches the pipes, it ends the game. Basic shapes, translation, rotation and scaling have been used in the program to emulate the same experience as to when the user plays Flappy Bird.

Design of the System

The game has been designed and developed in Javascript and WebGL using translation, rotation and scaling. The box is of the boxes done in class, with a little bit of scaling to make it smaller. The program checks for collisions in the Y axis, between pipes and the box. Every time that a pipe gets passed correctly one point is awarded, and if a collision is detected then the program ends. After the game is over, the system will check if the score is a new highscore and if it is, the new high score will be saved. Lastly, the user has the ability to restart the game by pressing the ‘R’ letter.

Implementation Tools

- WebGL
- JavaScript
- HTML

Special Features

- Collision when the box touches any of the pipes which means that the game is over.
- Every time the bunny passes a pipe one point will be added to the score.
- To make the box go up the spacebar must be pressed.
- A sound will be made every time the player gets a point, when the box goes up and when a collision occurs, indicating a game over.
- The position between pipes where the box needs to go through is generated randomly.
- The system will track highscores in every session.
- To restart the game, the ‘R’ letter must be pressed.

Implementation Details

- A model matrix is used for all the scaling and transposing.
- Several control variables have been created to manage the game:

```
/* ------------------------- Variables de Control -------------------------- */
//Maneja la posicion en Y del box. Se va decrementando cada vez que se corre el
//render y se incrementa cuando el usuario entra la tecla de space
var yTranslation = 0.000001;
//Variable que controla si el juego debe empezar o no
var start = 0;
//Variable que nos dice si fue la primera vez que se corrio el render
//para esa forma hacer el render una vez, y despues parar hasta que
//el usuario empieze el jeugo
var firstTime = 1;
//Variable para llevar el score del juego actual
var score = 0;
//Variable que lleva la puntuacion mas alta
var highScore = 5;
//Variable que usamos para las translaciones en x de los pipes, se va
//incrementando cada vez que se hace el render.
var xMovement = 0;
//Variable que usamos para crear espacios entre los pipes.
var xTranslation = 0;
//Arreglo donde se guarda donde esta localizado en Y el espacio por donde
//debe pasar el box entre pipes
var randomNess = [ 0.0, 0.0, 0.0, 0.0 ];
//Variable de control donde sabemos si hubo un gameover para no refrescar
//la pantalla y no aceptar mas users inputs hasta que el juego se re-inicie.
var isGameOver = 0;
```

- Those variables get updated throughout the program to simulate a game just like flappy bird. The variables either get updated through the render function or through user's input

- User's input is captured to play the game, we capture the events in JS to start the game (Enter), to make the box hop (Space) and restart the game (R).

```
//funcion de js para capturar eventos, de teclas presionadas por el usuario
document.onkeydown = function (ev) {
  switch (ev.keyCode) {
      //Capturamos si debemos permitir que el box suba dependiendo si no hubo
      //un gameover y si el usuario comenzo el juego (SPACEBAR)
      case 32: if(isGameOver == 0 && start == 1){
                  yTranslation -= 0.15;
                  var audio = new Audio('Sounds/sfx_wing.wav');
                  audio.play();
                } break;
      //Si no estamos en la pantalla de gameover, permitimos que el usuario
      //comienze el juego (ENTER)
      case 13: if(isGameOver == 0) {
                  start = 1;
                  document.getElementById("score").innerHTML = "Score: " + score;
                } break;
      //Si hubo un gameover, permitimos al usuario resetear el juego. Por
      //ende reseteamos todas las variables de control
      case 82: if(isGameOver == 1){
                  yTranslation = 0.000001;
                  start = 0;
                  isGameOver = 0;
                  firstTime = 1;
                  score = 0;
                  xMovement = 0;
                  //Aqui se encuentra el primer pipe, en la posicion 3. El
                  //primero siempre es zero, porque queremos que el primer
                  //pipe siempre aparezca en la primera posicison
                  randomNess[3] = 0;
                  for (i = 0; i < 3; i++) {
                    randomNess[i] = (1 + Math.floor(Math.random() * 10)) * 0.1;
                    var zeroOrOne = Math.floor(Math.random() * 2);
                    if (zeroOrOne == 1)
                      randomNess[i] = -randomNess[i];
                  }
                  document.getElementById("score").innerHTML = "Press Enter"
                     + "to Start!<br> <br> Current High Score: " + highScore;
                } break;
  }
```

- The safe spot is generated randomly.

```
randomNess[3] = 0;
//Tenemos el for loop que va por los remaining pipes, y para cada posicion
//se crea un valor random que va desde -1 a 1 en incrementos de .1. Estos
//valores se usan para posicionar el espacio seguro entre pipes donde el
//box tiene que pasar aleatoreamente
for (i = 0; i < 3; i++)
{
  randomNess[i] = (1 + Math.floor(Math.random() * 10)) * 0.1;
  var zeroOrOne = Math.floor(Math.random() * 2);
  //Anadimos valores negativos de vez en cuando aleatoreamente
  if (zeroOrOne == 1)
    randomNess[i] = -randomNess[i];
}
```

- The program gets printed once before everything starts. Once the user hits the start button the renderization gets going until a game over occurs.

```
//Funcion para renderizar nuestro programa
function render()
{
  //Si la variable de control de start es cierto, el programa continua el
  //rendering. La variable de control firstTime la usamos para imprimir la
  //pantalla una vez cuando sube el programa
  if(start == 1 || firstTime == 1)
  {
    initBuffers(vertexColorGreen, vertices, indices);
    gl.clear( gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
```

- Every time the box passes a pipe a point is awarded. Here's how it was achieved:

```
//Si el movimiento en x es .3, aproximado, ese es el momento en el que
//el box pasa el pipe. Se añade un punto y suena el sonido del punto
if (xMovement > .298 && xMovement < .3) {
  score++;
  var audio = new Audio('Sounds/sfx_point.wav');
  audio.play();
  //El score se le da update en el HTML
  document.getElementById("score").innerHTML = "Score: " + score;
}
```

- To have random safe zones the following was done:

```
//Si el movimiento en x es .5 o mas, significa que el pipe en la posicion 3,
//o sea, el pipe mas cercano al box ya va a salir de la pantalla. Por eso
//tenemos que hacer varias cosas para que el programa siga funcionando sin
//ningun problema.
if (xMovement > .5)
{
  //Inicializamos la variable de xMovement a cero, para que los pipes vuelvan
  //a desplegarse desde el principio del canvas
  xMovement = 0;

  //Guardamos los valores de randomNess en una arreglo temporero ya que
  //tenemos que cambiar el orden de randomNess ya que uno de los arreglos,
  //el tercero salio de la pantalla y un nuevo pipe se tiene que dibujar tambien
  var tempArray = [0, 0, 0, 0];
  for (i = 0; i < 4; i++)
    tempArray[i] = randomNess[i];

  //Nuevo random para el pipe en la posicion cero, q es el pipe mas a la
  //derecha de nuestro canvas
  randomNess[0] = (1 + Math.floor(Math.random() * 10)) * 0.1;
  var zeroOrOne = Math.floor(Math.random() * 2);
  if (zeroOrOne == 1)
    randomNess[0] = -randomNess[0];

  //Ahora, lo que hacemos es asignamos los valores que teniamos de randomNess
  //de la posicion 0, 1 y 2 a , 1, 2, 3, respectivamente. EL pipe que teniamos
  //en la posicion 3 se fue, y el que teniamos en la posicion 2, ahora estara
  //en la posicion 3 (con el xMovement reseteado), el que teniamos en la posicion
  //1, ahora estara en la 2, y el de la posicion 0, en la 1. El de la posicion
  //0 es un pipe nuevo.
  randomNess[1] = tempArray[0];
  randomNess[2] = tempArray[1];
  randomNess[3] = tempArray[2];
}
```

- Randomness stores values from -1 to 1, in .1 increments. Meaning the possible values are -1, -.9, -.8, -.7, -.6 until we reach 1.

- To print the pipes correctly we had to do the following:

```
//Loop que imprime los cuatro pipes en la pantalla. Para cada pipe hay dos
//figuras rectangulares, arriba y abajo. El randomNess se usa en el scaling
//para hacer la figura mas larga o mas corta. El mismo size que se alarga
//en uno de los dos pipes, se acorta en el otro
for (i = 0; i < 4; i++) {
  //Con la funcion de translate se hace el translate del pipe  incluyendo
  //el movimiento en x, y la translacion en x, que indica el spacing entre
  //los pipes. Primero se dibuja el pipe de arriba aqui.
  modelMatrix = translate((0.9 - xTranslation - xMovement), 1, 0);
  //Con la funcion del scale, hacemos el scaling de nuestro primer pipe de
  //arriba. El randomNess se usa en el scaling para hacer la figura mas
  //larga o mas corta. El mismo size que se alarga en uno de los dos pipes,
  //se acorta en el otro
  modelMatrix = mult(modelMatrix, scale(.15, 1.5 + randomNess[i], 0));

  //Se le da update a la matrix de modelado
  gl.uniformMatrix4fv(u_modelMatrix, false, flatten(modelMatrix));
  //Se dibuja el pipe de arriba
  gl.drawElements( gl.TRIANGLES, indices.length, gl.UNSIGNED_BYTE, 0 );

  //Aqui se hace el translate del pipe de abajo
  modelMatrix = translate((0.9 - xTranslation - xMovement), -1, 0);
  //Aqui se hace el scaling del pipe de abajo. El randomNess que se sumo
  //en el de arriba, se resta aqui. Com el randomNess siempre va en
  //incrementos de .1 el 'safe zone' siempre es del mismo tamano
  modelMatrix = mult(modelMatrix, scale(.15, 1.5 - randomNess[i], 0));

  //Se le da update a la matrix de modelado
  gl.uniformMatrix4fv(u_modelMatrix, false, flatten(modelMatrix));
  //Se dibuja el pipe de abajo
  gl.drawElements( gl.TRIANGLES, indices.length, gl.UNSIGNED_BYTE, 0  );

  //Movemos a .5 y alli es donde imprimiremos el proximo pipe
  xTranslation += .5;
  //Con esto controlamos el moviento en x.
  xMovement += 0.0006;
}
```

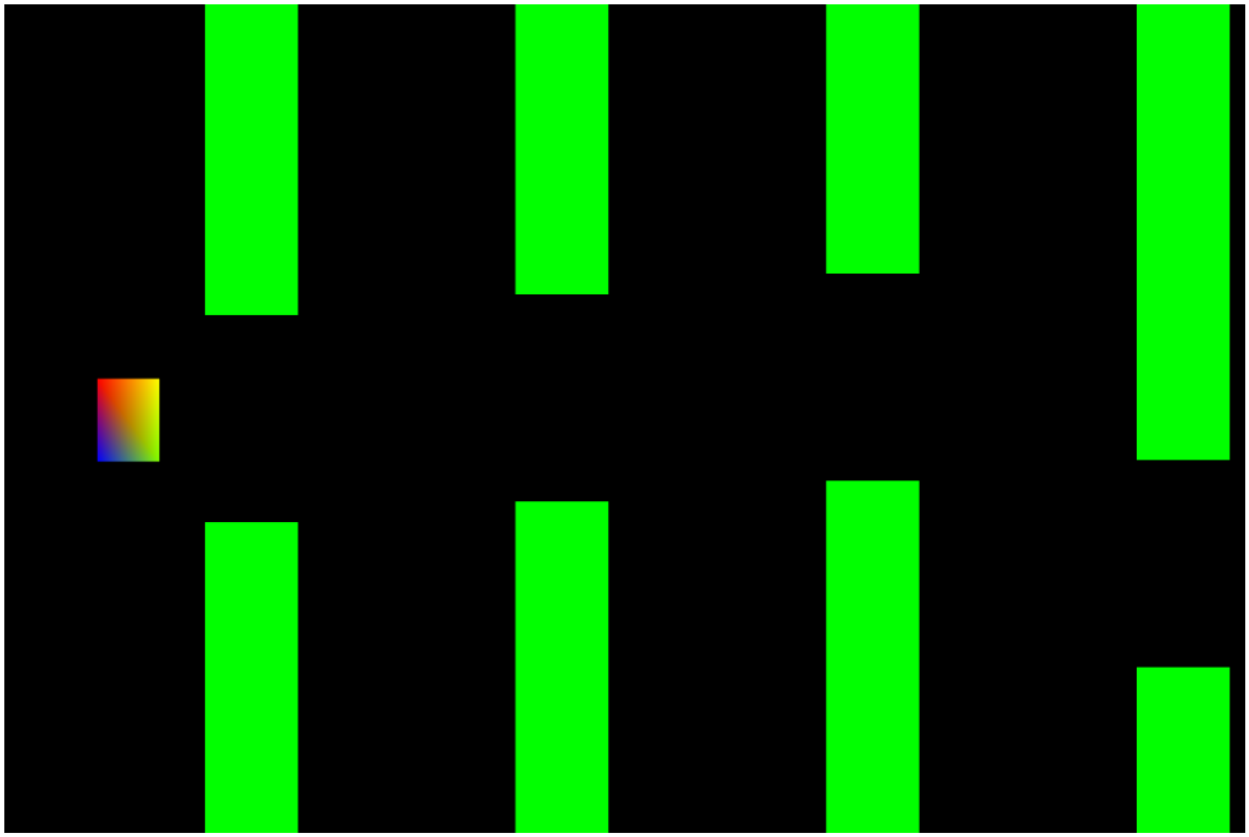
- To check for collisions the following was done:

```
//Si el movimiento en x esta entre .1 y .3, esto implica que el box esta
//pasando por el safe zone y tenemos que verificar si hay colisiones
if (xMovement > .1 && xMovement < .3)
{
  //Esta es una ecuacion(es) que encontramos para determinar si hay colision en
  //base a la posicion en y del box, y el valor de randomNess. Se verifica
  //si hay colision en el pipe de arriba y el de abajo.
  //Para encontrar esta formula, manualmente capturamos puntos de yTranslation
  //donde habia colision arriba y abajo para varios valores de -1 a 1 con
  //incrementos en .1, y asi encontramos la formula
  if( yTranslation < randomNess[3] * .5 - 0.15 || yTranslation > randomNess[3] * .5 + 0.15 )
  {
    //Hubo colision, y por eso el programa deja de renderizar imagenes ya
    //que seteamos la variable de control isGameOver a 1. Tambien enviamos
    //el audio de gameover y verificamos si hubo un nuevo highScore
    var audio = new Audio('Sounds/sfx_hit.wav');
    audio.play();
    if(score > highScore)
      highScore = score;
    document.getElementById("score").innerHTML =
              "GAME OVER! <br> SCORE: " + score + "<br>HIT 'R' TO TRY AGAIN!"
              + "<br> <br> Current High Score: " + highScore;
    isGameOver = 1;
    start = 0;
  }
}
```

- Once a collision occurs, the game is over and an appropriate message is displayed on screen. Users have the ability to restart the game by hitting the R key.
- Lastly, to print and handle the box movement the following was done:

```
//Cada vez que se renderiza la figura, le restamos un poco a la variable de
//control de yTranslation para que el box baje (esta sube cuando se entra
//un input en el spacebar)
yTranslation += 0.004;
initBuffers(vertexFunColors, vertices, indices);
//Le damos update a nuestra matriz de escalado para el box, usando translacion
modelMatrix = translate(-.8, -yTranslation, 0);
//Y aqui scaling
modelMatrix = mult(modelMatrix, scale(.1, .2, 0));
//Subimos nuestra nueva modelMatrix antes de dibujar nuestro box
gl.uniformMatrix4fv(u_modelMatrix, false, flatten(modelMatrix));
//Dibujamos nuestro box
gl.drawElements( gl.TRIANGLES, indices.length, gl.UNSIGNED_BYTE, 0 );
```

And with that, the project was completed successfully. It was a fun project where many of the lessons learned in class were applied in a fun application. The application looks like this:



Press Enter to Start!

Press the Space Bar To Make the Box Go up!

Current High Score: 5

Lessons Learned

- It was a really fun project where I was able to learn how to make a practical application with the concepts shown and taught in class.
- How simple things like translation and scaling can be used to create extremely successful applications.
- I got more understanding of the underlying concepts of graphic engines, which is great.
- Handling collisions and movement is a lot more complicated when you don't have objects or can use powerful design tools.
- How powerful can control variables be to make sure your program works successfully and smoothly.

Possible Future Work

- Better models can be used to make the program look better.
- The ability to make the box go down faster.
- Work on performance, i.e.removing the init buffers inside the render function.
- Add textures
- Create a module for tournaments, or to keep track of highscores over the web.
- Implement this in a 3D environment
- And many other things, the possibilities are endless.

Video Demo

A demo of the program can be found at: https://youtu.be/IGOZ2nNt3jE

Github

https://github.com/Cindy-Mendez/Flappy-Box

Conclusion

In conclusion, I felt like I've learned so many new things with this class and I am extremely grateful for it. It was extremely challenging throughout the semester, and the struggle was real, but definitely worth it as I have expanded my knowledge and my skill set. I am looking forward to my other achievements in computer graphics.