Universidad Rey Juan Carlos

Gráficas y Visualización 3D

Práctica 4: *Buffers*

Katia Leal Algara

Agustín Santos Méndez

1. **Buffers y atributos**: “Example 2-1.html”, este ejemplo dibuja un triángulo sencillo con colores diferentes para cada vértice. Para poner colores a los vértices tenemos que jugar con los shaders y buffers. En este ejemplo definimos en el shader de vértices una nueva variable para recibir el color. También modificamos la construcción del buffer. ¿Puedes localizar esos cambios? Observa cómo los cambios afectan a los elementos del código. Se modifican las definiciones de ambos shaders, la construcción de los buffers y la sentencias que los conectan con los shaders.

   El objetivo del fragment shader es determinar el color de cada pixel. Por lo tanto, ahora lo que hacemos es hacerle llegar un color mediante una variable llamada vColor. Esa variable está declarada como “**varying**” (es decir, es una variable generada en el vertex shader y traspasada al fragment shader). Si ahora miramos el vertex shader, veremos que esa variable toma el valor que procede de “aVertexColor”. Y esa variable a su vez, está declarada como “attribute”. Recordad que una variable “attribute” es una variable que toma valores desde un buffer.

   Lo que significa esto, es que el color de cada vértice vendrá determinado por un valor que tendremos que almacenar en un buffer. El color de los pixels intermedios serán calculados mediante interpolación de los vértices.

   a) Cambia los colores de los vértices para adaptarlos a tu gusto.
   b) ¿Sabrías dibujar un cuadrado con colores? Simplemente añade un nuevo vértice, pero no te olvides de darle un color.
   c) Dibuja un círculo de tamaño unidad centrado en el origen. Haz que cada vértice tenga un color y que el centro sea blanco.

2. **Combinando los buffers**: “Example 2-2.html”, volvemos a dibujar el mismo triángulo, pero esta vez vamos a cambiar la forma de crear y conectar los buffers. En lugar de tener dos buffers con los datos de los vértices y de los colores, se va a crear un único buffer. En el mezclamos los datos de los vértices y los datos de los colores. Observad que el código de los shaders no cambia. Únicamente cambiamos el proceso de definición y uso de los buffers. Es interesante el cambio que se ha producido en las llamadas a glVertexAttribPointer. ¿Podéis determinar las razones? **Nota**. Observad que hemos cambiado la paleta de colores para distinguirlo del ejemplo anterior.

   Cambiamos los parámetros de las llamadas a “ gl.vertexAttribPointer” para indicar cómo debe encontrar los datos dentro del buffer. El valor “6*4” se corresponde al tamaño en bytes que hay entre un vértice y otro. Esos números proceden del hecho de que un float ocupa 32 bits (4 bytes) y que hay 6 datos entre vértices (3 son de coordenadas y otros 3 son de colores). El otro valor “3*4” le indica la posición del color dentro del buffer (hemos consumido 3 floats para indicar la posición).

   a) Cambia los colores de los vértices para adaptarlos a tu gusto.
   b) ¿Sabrías dibujar un cuadrado con colores? Simplemente añade un nuevo vértice, pero no te olvides de darle un color.

3. **Profundizando en los atributos**: “Example 2-3.html”, volvemos a dibujar el mismo triángulo, pero hemos vuelto a cambiar la forma de crear y conectar los buffers. Seguimos usando un único buffer con los datos de los vértices y los datos de los colores. Pero ahora vamos a cambiar el tamaño y naturaleza de los datos. Vamos a transmitir solo dos coordenadas para la posición. **Nota**. Observad que hemos vuelto a cambiar la paleta de colores para distinguirlo del ejemplo anterior.

   Hemos cambiado los vertex shaders para indicar que solo hay dos coordenadas por vértice y ponemos la coordenada “Z” igual a cero por defecto.

   a) Cambia los colores de los vértices para adaptarlos a tu gusto.
   b) ¿Sabrías dibujar un cuadrado con colores? Simplemente añade un nuevo vértice, pero no te olvides de darle un color.

4. **Usando índices**: “Example 2-4.html”, este ejemplo dibuja un cuadrado, pero hemos cambiado la forma de procesar el buffer. Hasta ahora teníamos un buffer con el array de puntos que queríamos dibujar. Si el punto se dibujaba dos veces, lo almacenábamos dos veces. Ahora lo que hacemos es almacenar por un lado el array de puntos (pero cada punto una única vez) y por otro, guardamos los índices de los vértices que vamos a dibujar. De esta forma, solo tenemos cuatro vértices pero seis índices (tres por cada triángulo que forma el cuadrado).

   La construcción de un **buffer de índices** se parece mucho a lo que habíamos visto hasta el momento pero hay dos diferencias principales. Por un lado, el tipo del buffer es ahora “gl.**ELEMENT_ARRAY_BUFFER**”. Eso le indica a WebGL que se tratan de índices. Y por otro lado, al ser números enteros sin decimales, el buffer se crea como enteros de 16 bits (unsigned short).

   Recordad que a la hora de dibujar debemos utilizar “**drawElements**” en lugar de “drawArrays”.

   a) Cambia el modelo para que parezca un hexágono.
   b) ¿Sabrías generar un círculo con esta técnica?