Ejercicio 3 – Dibujar triángulos de color haciendo click en canvas

Este ejercicio tiene como objetivo implementar una aplicación WebGL poniendo en práctica todos los conceptos estudiados en el tema 3 de la asignatura “Conceptos básicos en WebGL”.

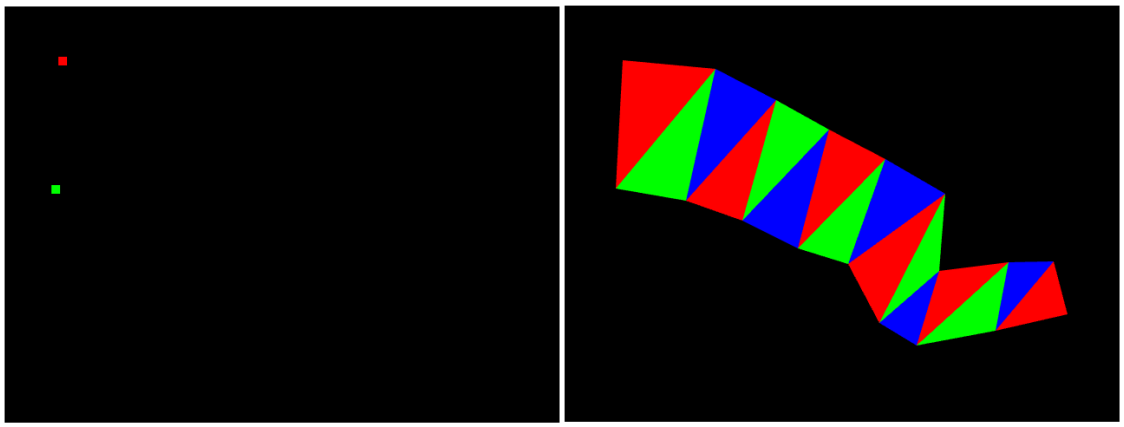
Como resultado de tu práctica deberás generar un **único fichero HTML** que deberás subir al Aula Virtual.

Puntos totales posibles del ejercicio: 10

Instrucciones

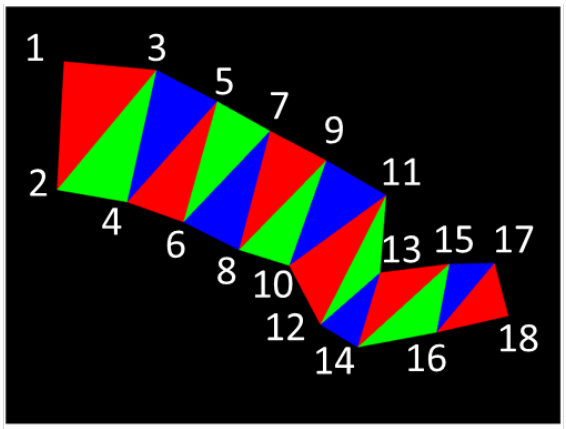
Partiendo de un canvas HTML, se pide realizar una aplicación WebGL que cumpla los siguientes requisitos:

- El color de fondo del canvas se pintará con WebGL en color **negro** (y permanecerá en ese color).
- Al iniciarse la aplicación el canvas no contendrá ningún gráfico.
- Al hacer click en el canvas, mediante JavaScript se localizará la posición de las coordenadas de ese click. Cuando haya **menos de 3 puntos**, se dibujará **un punto** en el canvas mediante WebGL en las posiciones donde se haya hecho click (el color es indiferente). Cuando haya **3 o más puntos**, se utilizará la coordenada del nuevo click así como las dos anteriores, para **dibujar un triángulo** con esos 3 puntos.
- El color del triángulo depende del siguiente orden:
    - El **primer** triangulo será de color **rojo**.
    - El **siguiente** de color **verde**.
    - El **siguiente** de color **azul**.
    - **Repetir** el patrón anterior, **rojo, verde, azul**.



Ayuda

El orden de los clicks del ejemplo ha sido el siguiente:



Se proporciona el siguiente fragmento JavaScript, destinado a capturar el evento de click del ratón encima del canvas y transformar las coordenadas (x,y) de dicho click a coordenadas WebGL. Dado un canvas HTML5 identificado como "myCanvas":

```
// Get canvas object from the DOM
var canvas = document.getElementById("myCanvas");

// Init WebGL context
var gl = canvas.getContext("webgl");

// Register event handler
canvas.onmousedown = function(ev) {
    click(ev, gl, canvas);
};


function click(ev, gl, canvas) {
    // Coordinates of canvas origin
    var rect = ev.target.getBoundingClientRect();

    // relative x coordinate of click in canvas
    var clickX = ev.clientX - rect.left;

    // relative y coordinate of click in canvas
    var clickY = ev.clientY - rect.top;

    // WebGL coordinates (3D)
    var halfCanvasWidth = canvas.width / 2;
    var halfCanvasHeight = canvas.height / 2;
    var x = (clickX - halfCanvasWidth) / halfCanvasWidth;
    var y = (halfCanvasHeight - clickY) / halfCanvasHeight;
    var xyz = [x, y, 0];

    // ...
}
```