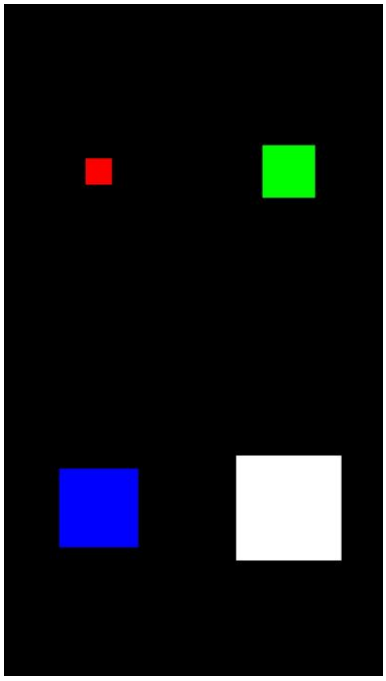
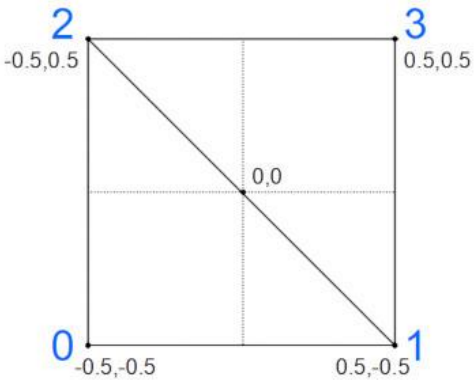
# Android - OpenGL ES 2 - Tutorial 5

# Disegnare Quadrati

Allo stesso modo dei triangoli, disegniamo dei quadrati come nel seguente screenshot:



Consideriamo il quadrato dell'immagine seguente:




Creiamo una classe "ObjSquare" con il seguente codice:

```
package opengles2.tutorial5;

import android.opengl.GLES20;

import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.nio.FloatBuffer;
import java.nio.ShortBuffer;

import javax.microedition.khronos.opengles.GL10;
```

```
public class ObjSquare {
    private FloatBuffer vertexBuffer;
    private ShortBuffer indexBuffer;

    private final int COORDS_PER_VERTEX = 2;
    private final int vertexStride = COORDS_PER_VERTEX * 4;
    private float[] vertices = {
            -0.5f, -0.5f,
            0.5f, -0.5f,
            -0.5f, 0.5f,
            0.5f, 0.5f
    };

    private short[] indices = {
            0, 1, 2,
            2, 1, 3
    };

    private float[] color_red = { 1.0f, 0.0f, 0.0f, 1.0f };
    private float[] color_green = { 0.0f, 1.0f, 0.0f, 1.0f };
    private float[] color_blue = { 0.0f, 0.0f, 1.0f, 1.0f };
    private float[] color_white = { 1.0f, 1.0f, 1.0f, 1.0f };

    private final String vertexShaderCode =
            "uniform mat4 uMVPMatrix;" +
            "attribute vec4 vPosition;" +
            "void main() {" +
            "  gl_Position = uMVPMatrix * vPosition;" +
            "}";

    private final String fragmentShaderCode =
            "precision mediump float;" +
            "uniform vec4 vColor;" +
            "void main() {" +
            "  gl_FragColor = vColor;" +
            "}";

    private int mProgram;

    public ObjSquare() {
        ByteBuffer vbb = ByteBuffer.allocateDirect(this.vertices.length * 4);
        vbb.order(ByteOrder.nativeOrder());
        this.vertexBuffer = vbb.asFloatBuffer();
        this.vertexBuffer.put(this.vertices);
        this.vertexBuffer.position(0);

        ByteBuffer ibb = ByteBuffer.allocateDirect(this.indices.length * 2);
        ibb.order(ByteOrder.nativeOrder());
        this.indexBuffer = ibb.asShortBuffer();
        this.indexBuffer.put(this.indices);
        this.indexBuffer.position(0);
    }
    public void SetProgramShader() {
        int vertexShader = MainRenderer.LoadShader(GLES20.GL_VERTEX_SHADER,
this.vertexShaderCode);
        int fragmentShader = MainRenderer.LoadShader(GLES20.GL_FRAGMENT_SHADER,
this.fragmentShaderCode);

        this.mProgram = GLES20.glCreateProgram();
        GLES20.glAttachShader(this.mProgram, vertexShader);
        GLES20.glAttachShader(this.mProgram, fragmentShader);
        GLES20.glLinkProgram(this.mProgram);
    }

    private int positionHandle;
    private int colorHandle;
    private int modelHandle;
```

```
    public void DrawSquare(float[] modelMatrix, int color) {
        GLES20.glUseProgram(this.mProgram);

        this.positionHandle = GLES20.glGetAttribLocation(this.mProgram, "vPosition");
        GLES20.glEnableVertexAttribArray(this.positionHandle);
        GLES20.glVertexAttribPointer(this.positionHandle, COORDS_PER_VERTEX,
GLES20.GL_FLOAT, false, this.vertexStride, this.vertexBuffer);

        this.colorHandle = GLES20.glGetUniformLocation(this.mProgram, "vColor");
        if (color == 1) // Red.
            GLES20.glUniform4fv(this.colorHandle, 1, this.color_red, 0);
        else if (color == 2) // Green.
            GLES20.glUniform4fv(this.colorHandle, 1, this.color_green, 0);
        else if (color == 3) // Blue.
            GLES20.glUniform4fv(this.colorHandle, 1, this.color_blue, 0);
        else // White.
            GLES20.glUniform4fv(this.colorHandle, 1, this.color_white, 0);

        this.modelHandle = GLES20.glGetUniformLocation(this.mProgram, "uMVPMatrix");
        GLES20.glUniformMatrix4fv(this.modelHandle, 1, false, modelMatrix, 0);

        GLES20.glDrawElements(GL10.GL_TRIANGLES, this.indices.length,
GL10.GL_UNSIGNED_SHORT, this.indexBuffer);

        GLES20.glDisableVertexAttribArray(this.positionHandle);
    }

}
```

Ogni forma che vogliamo disegnare deve essere ricondotta in triangoli, quindi OpenGL disegnerà il quadrato tramite 2 triangoli.

La variabile "indices" specifica i vertici da prendere da "vertices" per ogni triangolo (in senso antiorario). Per il primo triangolo, coppia coordinate x,y: 0, 1 e 2, per il secondo triangolo, coppia coordinate x,y: 2, 1 e 3.

Nella classe "MainRenderer" dichiariamo:

```
ObjSquare objsquare;
```

e inizializziamo nel costruttore:

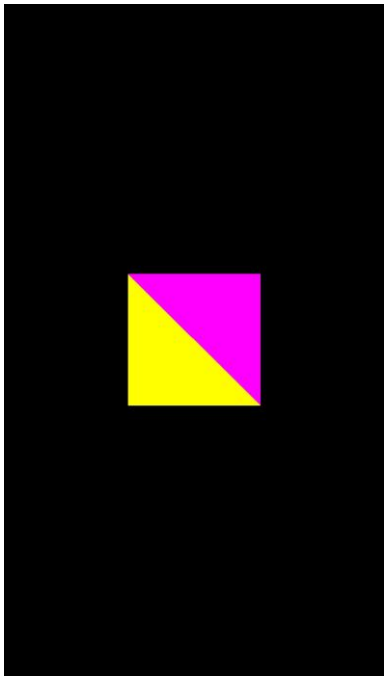
```
public MainRenderer() {
    this.objsquare = new ObjSquare();
}
```

Nel metodo "onDrawFrame()" disegniamo i nostri quadrati:

```
@Override
public void onDrawFrame(GL10 unused) {
    GLES20.glClear(GLES20.GL_COLOR_BUFFER_BIT);

    this.ResetModelMatrix();
    Matrix.translateM(this.modelMatrix, 0, this.width / 4.0f, this.height / 4.0f * 3.0f,
0.0f);
    Matrix.scaleM(this.modelMatrix, 0, 50.0f, 50.0f, 0.0f);
    this.objsquare.DrawSquare(this.modelMatrix, 1);

    this.ResetModelMatrix();
    Matrix.translateM(this.modelMatrix, 0, this.width / 4.0f * 3.0f, this.height / 4.0f *
3.0f, 0.0f);
    Matrix.scaleM(this.modelMatrix, 0, 100.0f, 100.0f, 0.0f);
    this.objsquare.DrawSquare(this.modelMatrix, 2);
```

```
    this.ResetModelMatrix();
    Matrix.translateM(this.modelMatrix, 0, this.width / 4.0f, this.height / 4.0f, 0.0f);
    Matrix.scaleM(this.modelMatrix, 0, 150.0f, 150.0f, 0.0f);
    this.objsquare.DrawSquare(this.modelMatrix, 3);

    this.ResetModelMatrix();
    Matrix.translateM(this.modelMatrix, 0, this.width / 4.0f * 3.0f, this.height / 4.0f,
0.0f);
    Matrix.scaleM(this.modelMatrix, 0, 200.0f, 200.0f, 0.0f);
    this.objsquare.DrawSquare(this.modelMatrix, 0);
}
```

E se volessimo disegnare un quadrato come questo?



Ovviamente possiamo disegnare singolarmente due triangoli adiacenti tra loro, ma creiamo una classe dedicata "ObjSquare2" con il seguente codice:

```
package opengles2.tutorial5b;

import android.opengl.GLES20;

import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.nio.FloatBuffer;
import java.nio.ShortBuffer;

import javax.microedition.khronos.opengles.GL10;

public class ObjSquare2 {
    private FloatBuffer vertexBuffer;
    private ShortBuffer indexBuffer;
    private FloatBuffer colorBuffer;

    private final int COORDS_PER_VERTEX = 2;
    private final int vertexStride = COORDS_PER_VERTEX * 4;
    private float[] vertices = {
            -0.5f, -0.5f,
            0.5f, -0.5f,
            -0.5f, 0.5f,
            0.5f, -0.5f,
            -0.5f, 0.5f,
```

```
            0.5f, 0.5f
    };

    private short[] indices = {
            0, 1, 2,
            3, 4, 5
    };

    private final int COMPONENT_PER_COLOR = 4;
    private final int colorStride = COMPONENT_PER_COLOR * 4;
    private float[] colors = {
            1.0f, 1.0f, 0.0f, 1.0f, // Vertex 0 triangle 1 (Red+Green).
            1.0f, 1.0f, 0.0f, 1.0f, // Vertex 1 triangle 1 (Red+Green).
            1.0f, 1.0f, 0.0f, 1.0f, // Vertex 2 triangle 1 (Red+Green).
            1.0f, 0.0f, 1.0f, 1.0f, // Vertex 2 triangle 2 (Red+Blue).
            1.0f, 0.0f, 1.0f, 1.0f, // Vertex 1 triangle 2 (Red+Blue).
            1.0f, 0.0f, 1.0f, 1.0f // Vertex 3 triangle 2 (Red+Blue).
    };

    private final String vertexShaderCode =
            "uniform mat4 uMVPMatrix;" +
            "attribute vec4 vPosition;" +
            "attribute vec4 iColor;" +
            "varying vec4 vColor;" +
            "void main() {" +
            "  vColor = iColor;" +
            "  gl_Position = uMVPMatrix * vPosition;" +
            "}";

    private final String fragmentShaderCode =
            "precision mediump float;" +
            "varying vec4 vColor;" +
            "void main() {" +
            "  gl_FragColor = vColor;" +
            "}";

    private int mProgram;

    public ObjSquare2() {
        ByteBuffer vbb = ByteBuffer.allocateDirect(this.vertices.length * 4);
        vbb.order(ByteOrder.nativeOrder());
        this.vertexBuffer = vbb.asFloatBuffer();
        this.vertexBuffer.put(this.vertices);
        this.vertexBuffer.position(0);

        ByteBuffer ibb = ByteBuffer.allocateDirect(this.indices.length * 2);
        ibb.order(ByteOrder.nativeOrder());
        this.indexBuffer = ibb.asShortBuffer();
        this.indexBuffer.put(this.indices);
        this.indexBuffer.position(0);

        ByteBuffer vcc = ByteBuffer.allocateDirect(this.colors.length * 4);
        vcc.order(ByteOrder.nativeOrder());
        this.colorBuffer = vcc.asFloatBuffer();
        this.colorBuffer.put(this.colors);
        this.colorBuffer.position(0);
    }
    public void SetProgramShader() {
        int vertexShader = MainRenderer.LoadShader(GLES20.GL_VERTEX_SHADER,
this.vertexShaderCode);
        int fragmentShader = MainRenderer.LoadShader(GLES20.GL_FRAGMENT_SHADER,
this.fragmentShaderCode);

        this.mProgram = GLES20.glCreateProgram();
        GLES20.glAttachShader(this.mProgram, vertexShader);
        GLES20.glAttachShader(this.mProgram, fragmentShader);
        GLES20.glLinkProgram(this.mProgram);
    }
```

```
    private int positionHandle;
    private int colorHandle;
    private int modelHandle;

    public void DrawSquare(float[] modelMatrix) {
        GLES20.glUseProgram(this.mProgram);

        this.positionHandle = GLES20.glGetAttribLocation(this.mProgram, "vPosition");
        GLES20.glEnableVertexAttribArray(this.positionHandle);
        GLES20.glVertexAttribPointer(this.positionHandle, COORDS_PER_VERTEX,
GLES20.GL_FLOAT, false, this.vertexStride, this.vertexBuffer);

        this.colorHandle = GLES20.glGetAttribLocation(this.mProgram, "iColor");
        GLES20.glEnableVertexAttribArray(this.colorHandle);
        GLES20.glVertexAttribPointer(this.colorHandle, COMPONENT_PER_COLOR,
GLES20.GL_FLOAT, false, this.colorStride, this.colorBuffer);

        this.modelHandle = GLES20.glGetUniformLocation(this.mProgram, "uMVPMatrix");
        GLES20.glUniformMatrix4fv(this.modelHandle, 1, false, modelMatrix, 0);

        GLES20.glDrawElements(GL10.GL_TRIANGLES, this.indices.length,
GL10.GL_UNSIGNED_SHORT, this.indexBuffer);

        GLES20.glDisableVertexAttribArray(this.positionHandle);
    }

}
```

In questo caso, la variabile "indices" è utilizzata per indicare, oltre ai vertici dei triangoli, anche i rispettivi colori da prendere da "colors". I colori dei due triangoli non sono in comune, quindi siamo costretti a specificarli separatamente e di conseguenta a ripetere in "vertices" le coordinate.

Per il primo triangolo, coppia coordinate x,y: 0, 1 e 2 con colori in posizione 0, 1 e 2, per il secondo triangolo, coppia coordinate x,y: 3, 4 e 5 con colori in posizione 3, 4 e 5.

Nella classe "MainRenderer" dichiariamo:

```
ObjSquare2 objsquare2;
```

e inizializziamo nel costruttore:

```
public MainRenderer() {
    this.objsquare2 = new ObjSquare2();
}
```

Quindi nel metodo "onDrawFrame()":

```
@Override
public void onDrawFrame(GL10 unused) {
    GLES20.glClear(GLES20.GL_COLOR_BUFFER_BIT);

    this.ResetModelMatrix();
    Matrix.translateM(this.modelMatrix, 0, this.width / 2.0f, this.height / 2.0f, 0.0f);
    Matrix.scaleM(this.modelMatrix, 0, 250.0f, 250.0f, 0.0f);
    this.objsquare.DrawSquare(this.modelMatrix);
}
```