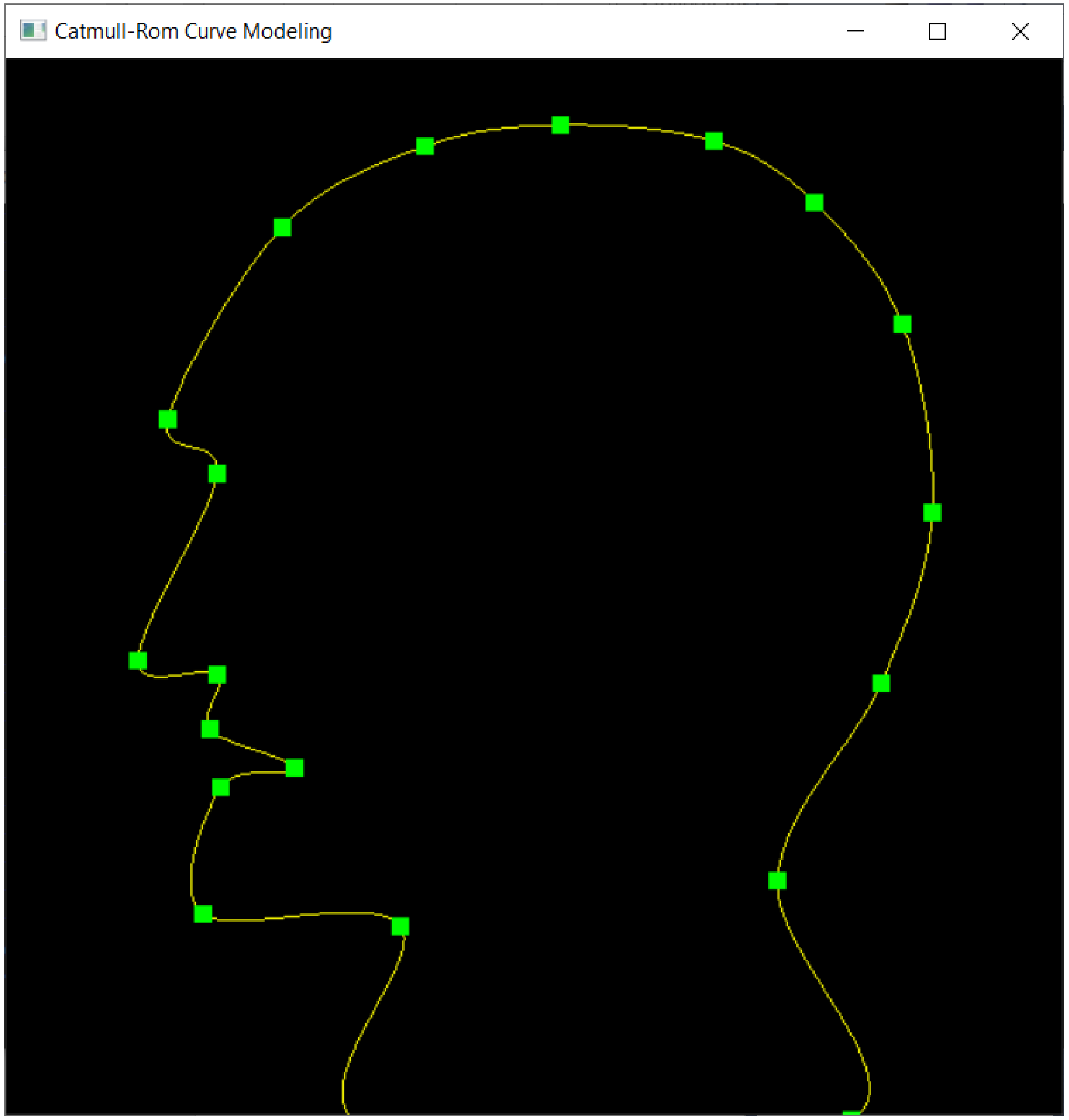
# Computer Graphics Laboratory Exercise 2

Balázs Csébfalvi

http://cg.iit.bme.hu/portal/en/cgbme

# Catmull-Rom curve modeling



## Use the original shader programs

GLSL shader programs

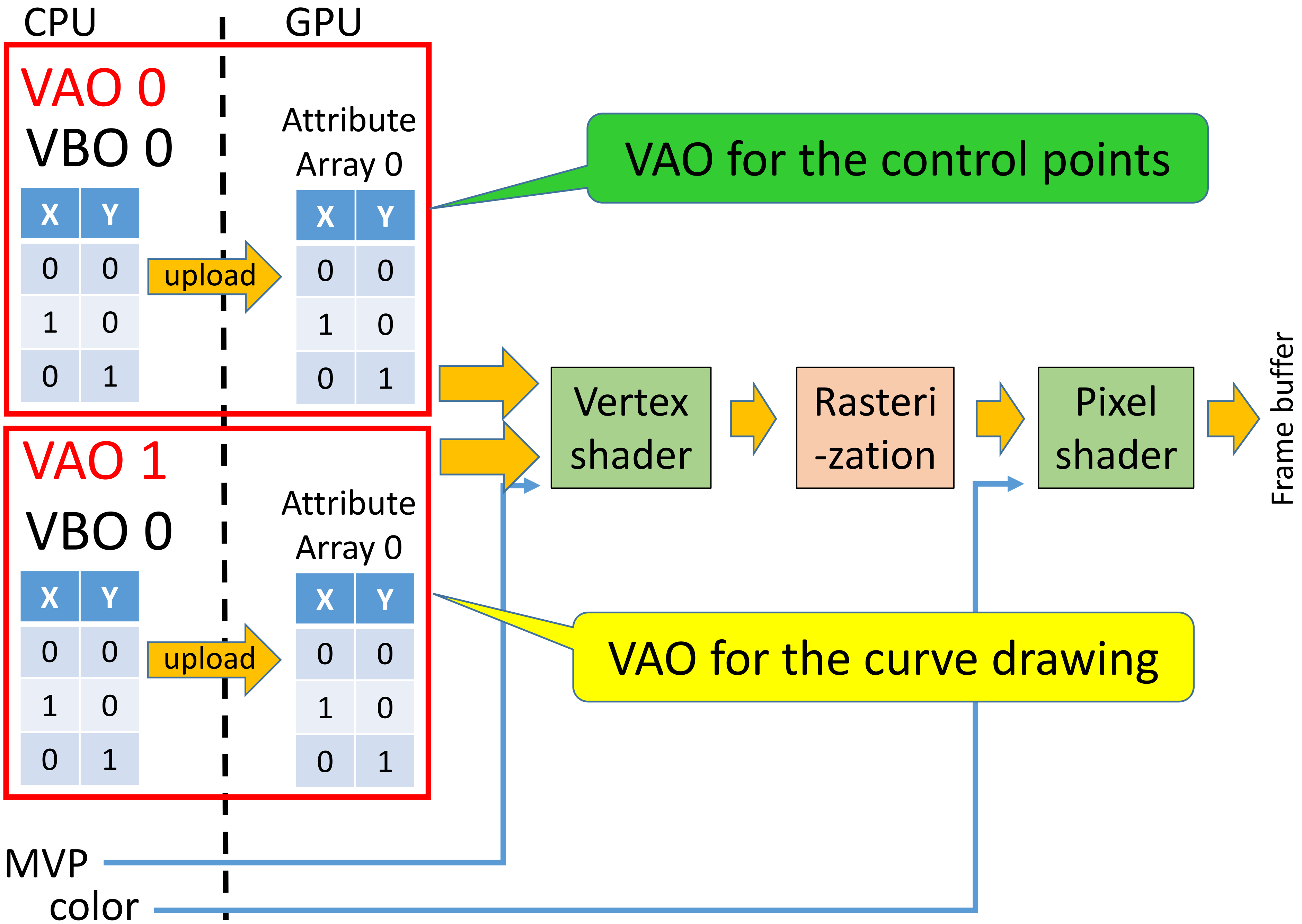
Vertex shader

```
#version 330
precision highp float;
uniform mat4 MVP;
layout(location = 0) in vec2 vp;
void main() {
  gl_Position = vec4(vp.x, vp.y, 0, 1) * MVP;
}
```

Pixel shader

```
#version 330
precision highp float;
uniform vec3 color;
out vec4 outColor;
void main() {
  outColor = vec4(color, 1);
}
```

## Two VAOs: one for the control points and one for the curve

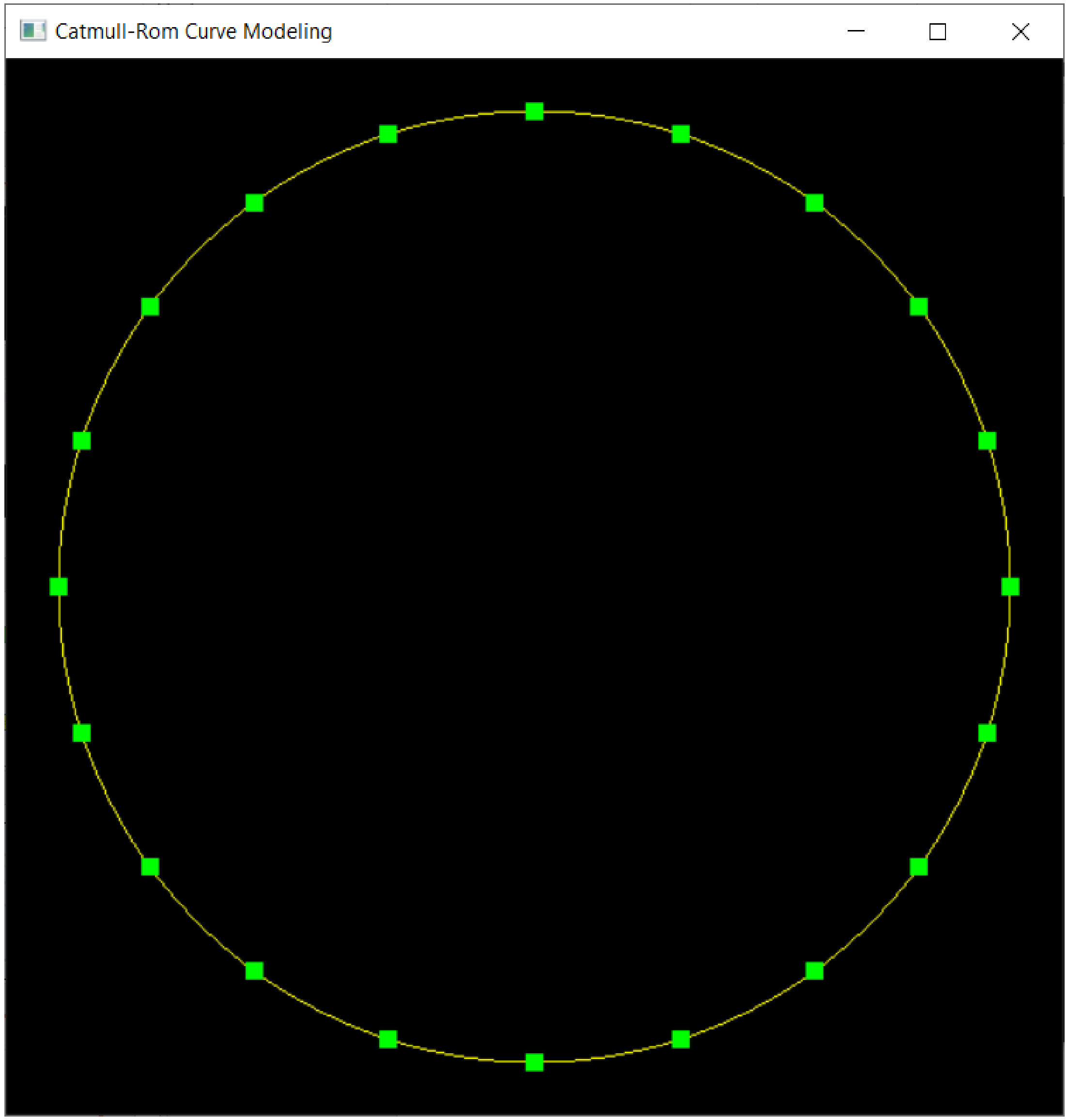


# Define class Curve

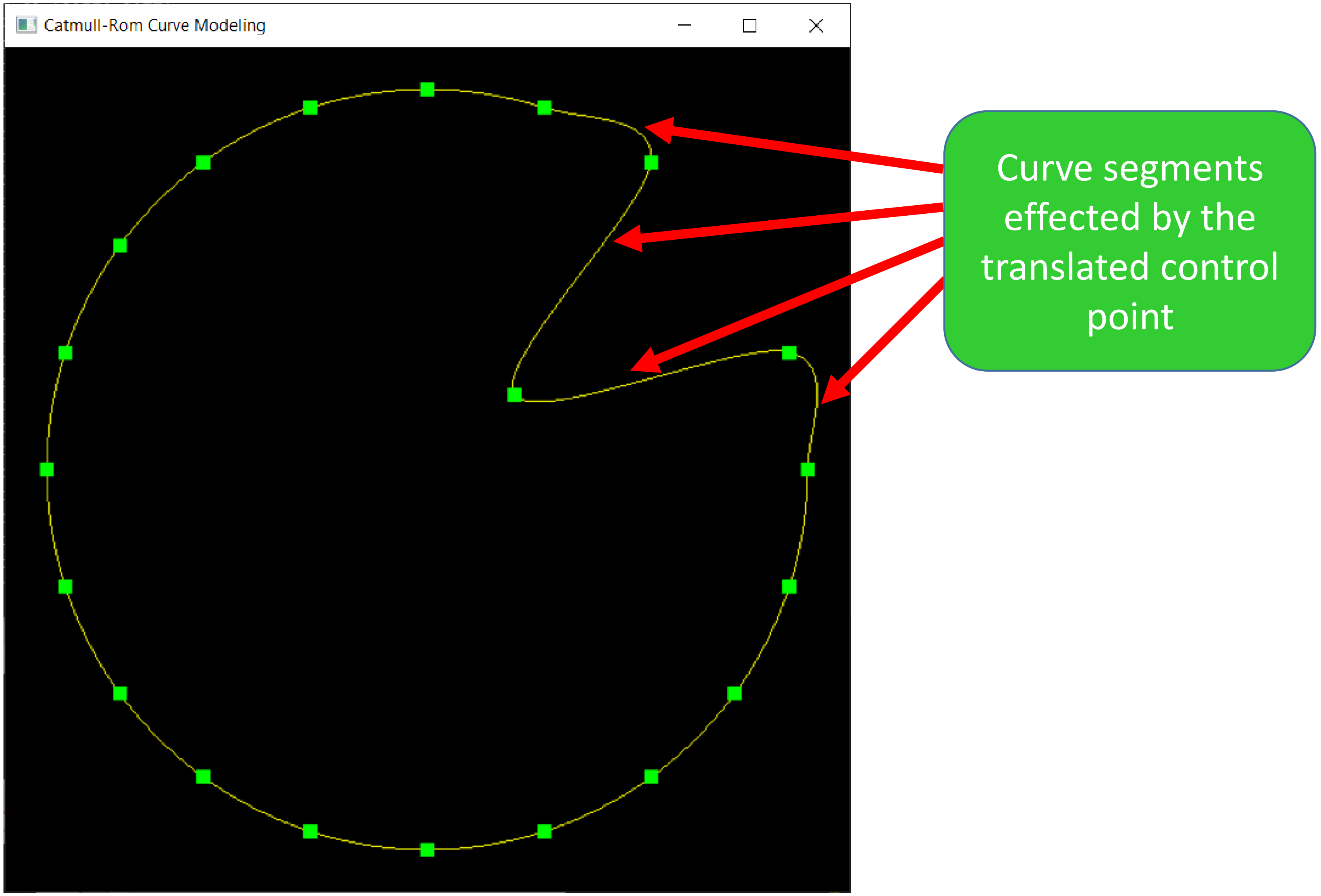
```
class Curve {
    unsigned int vaoPoints, vboPoints;
    unsigned int vaoVertices, vboVertices;
    int npoints; // number of control points
    vec2 *points; // array for the control points
    int nvertices; // number of vertices for the polyline
    vec2 *vertices; // array for the vertices
    vec2* activecp; // address of the moving control point

    vec2 CurvePoint(float t) {
        // evaluate the coordinates of the point corresponding to parameter t
    }
public:
    Curve() {
        points = 0; npoints = 0; vertices = 0; nvertices = 0; activecp = 0;
    }
    ~Curve() {
        if(points == 0) delete points;
        if(vertices == 0) delete vertices;
    }
    void Create(int np = 20, int nv = 1000){
        // create VAOs and VBOs here
    }
    void Render() {
        // activate VAOs and render the curve together with the control points
    }
// . . .
};
```

## Initialize the control points along a circle



## Implement drag and drop for the control points


Catmull-Rom Curve Modeling
Curve segments effected by the translated control point

## Add member functions for moving a control point, and call these functions from the corresponding message handlers

```
class Curve {
    // . . .
public:
    // . . .
    void PickControlPoint(float x, float y) {
        // activate the control point that is close enough to
        // position (x,y)
    }
    void ReleaseControlPoint() {
        activecp = 0; // there is no active control point
    }
    void MoveControlPoint(float x, float y) {
        // move the active control point if activecp is not null
        // upload the modified VBOs to the GPU
    }
};
```

## Implement member function Render()

```
class Curve {
    // . . .
public:
    // . . .
    void Render()
    {
        // activate vaoVertices
        // set the uniform color to yellow
        // draw the curve as GL_LINE_STRIP

        // activate vaoPoints
        // set the uniform color to green
        // set the point size by glPointSize(10);
        // draw the control points as GL_POINTS
    }
    // . . .
};
```