Schritt 5: Vertex Array Objects

Im letzten Schritt haben wir gehen, wie man die Daten aus dem Buffer Object in den Vertex Shader bekommt. Damit sind wir eigentlich schon in der glücklichen Lage, rendern zu können; zumindest bei einem rückwärts-kompatiblen (→ Compatibility) OpenGL Context, wie wir ihn mit SFML 2.1 bekommen.

Allerdings müssen wir vorm Rendern jedes Mal sämtliche Buffer Objects binden, die Vertex Attribute mit `glVertexAttribPointer` definieren usw., was den Render Code unübersichtlich macht und auch auf die Performance geht: jeder OpenGL Befehl, und wenn es nur der kleinste State Change ist, kostet prinzipiell Performance. Deshalb wurden die Vertex Array Objects eingeführt, es handelt sich dabei um Objekte welche genau den OpenGL State speichern, der für die Vertex Attribute zuständig ist. In einem nicht rückwärts-kompatiblen (→ Core) OpenGL Context stellen diese Vertex Array Objects sogar die einzige Möglichkeit dar, irgendetwas zu zeichnen.

- Im INIT werden sie erzeugt & auch gleich gebunden.
- Danach muss man den State der Vertex Attribute mit `glVertexAttribPointer` und `glEnableVertexAttribArray` so setzen, wie man selbigen fürs Rendern braucht.
- Auch den Index Buffer ans ELEMENT_ARRAY_BUFFER Target binden.
- Das Vertex Array Object unbinden.

Sämtlicher gesetzter State ist daraufhin im Vertex Array Object gespeichert. Vor dem Rendern muss dann nur noch das Vertex Array Object gebunden werden. Wenn Sie das entsprechende Code Sample betrachten, werden Sie sehen, dass hauptsächlich eine Verschiebung einiger relevanter Zeilen aus dem RENDER ins INIT stattgefunden hat.

INIT

```
GLuint handleToVertexArray = 0;
glGenVertexArrays( 1, std::addressof( handleToVertexArray ) );
glBindVertexArray( handleToVertexArray );
glBindBuffer( GL_ELEMENT_ARRAY_BUFFER, handleToIndexBuffer );
glBindBuffer( GL_ARRAY_BUFFER, handleToAttribBuffer );
glVertexAttribPointer( 0, 3, GL_FLOAT, false, sizeof( VertexData ), reinterpret_cast<
GLvoid const * >( offsetof( VertexData, position ) ) );
glVertexAttribPointer( 8, 4, GL_FLOAT, false, sizeof( VertexData ), reinterpret_cast<
GLvoid const * >( offsetof( VertexData, color ) ) );
glEnableVertexAttribArray( 0 );
glEnableVertexAttribArray( 8 );
glBindVertexArray( 0 );
```

RENDER

```
glBindVertexArray( handleToVertexArray );
glDrawElements( GL_TRIANGLES, indexData.size(), GL_UNSIGNED_INT, nullptr );
glBindVertexArray( 0 );
```