OpenGL - 3D Bar Chart

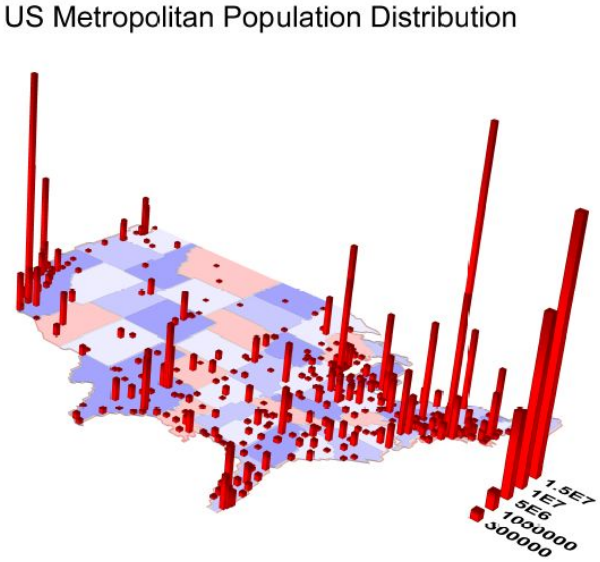
Team: Trilogy Graphs

Juan Luis Flores	A01280767
Alejandro de la Rosa	A01381412
Elí E. Linares	A00815654

Introduction

Introduction

We create a 3D bar charts project to show information about Mexico's population, poverty rate & unemployment because we think that is easier to visualize data with charts and it help us to get conclusions on every state.

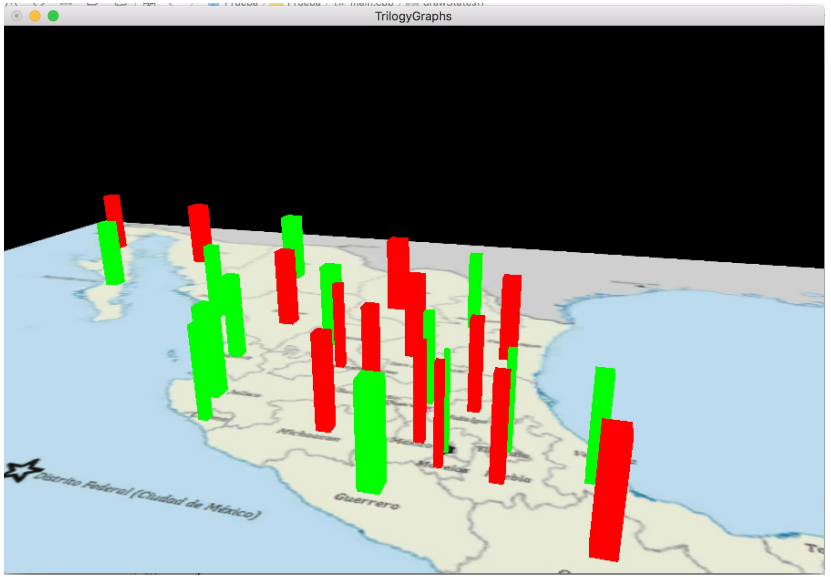


Modelling

General Concept

We made a plane in the X & Z dimensions with a Mexico's map texture.

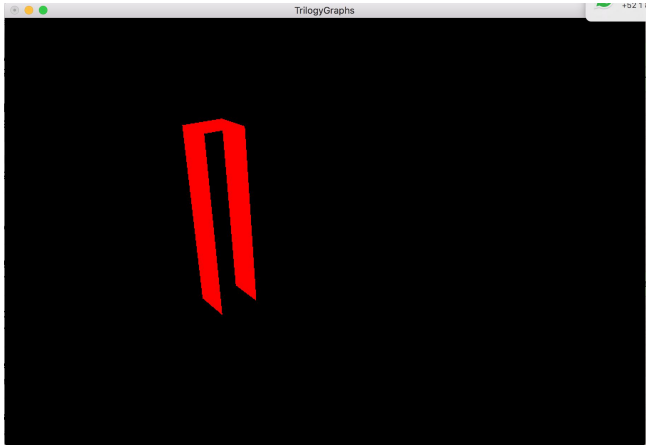
Then, we put some bars (rectangular prisms) on top of each state with a height

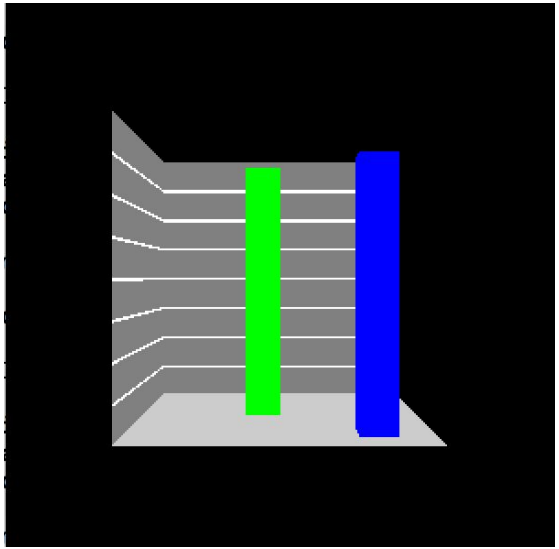


Concept

We based our design from a cube from (-1,-1,-1) to (1,1,1)

Prisms were created the same way cubes were, declaring their vertexes.





OpenGL - Primitives

QUADS.

For the bars we used 4 glVertex to create each side of the bar and glColor4F.

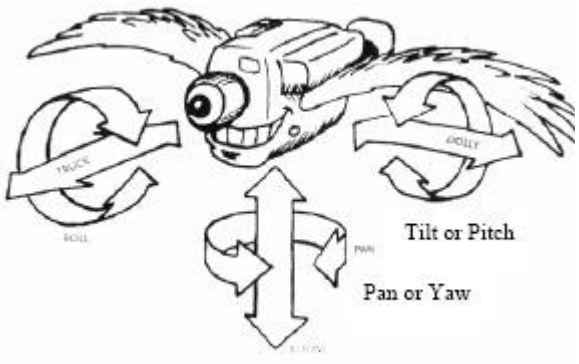
For the plane we loaded an image and bind it as a texture to the Plane QUAD

GUI

We manipulate our analyzed variable with the keyboard. (**This changes each bar's height**)

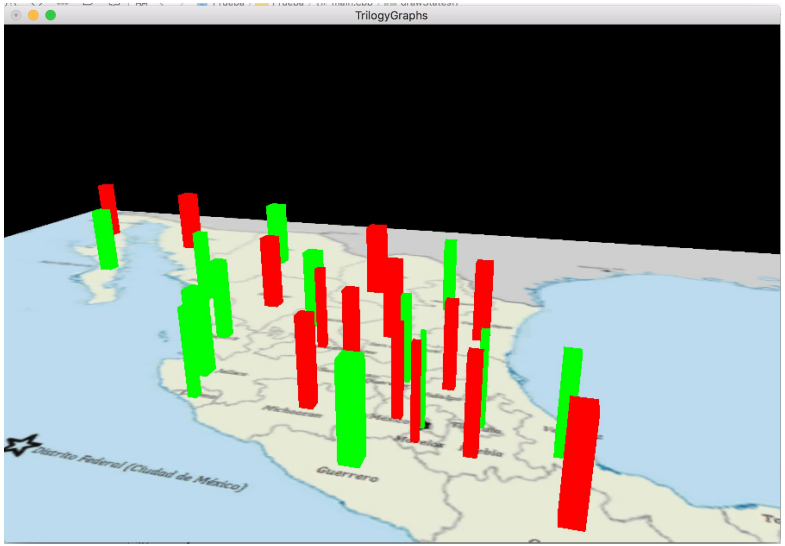
- DOLLY, TILT, TRUCK, BOOM, PAN

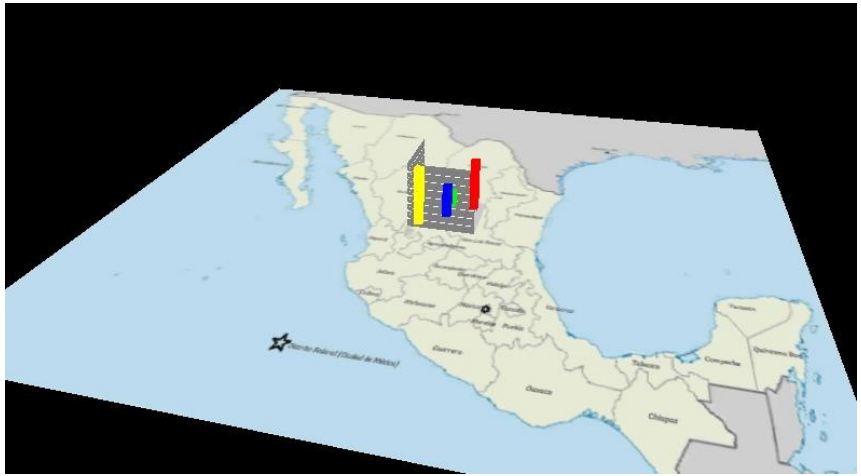
Five camera transformations as well as various scaling options in order to visualize the information at will or concentrate on certain areas of the complete Mexico's map.

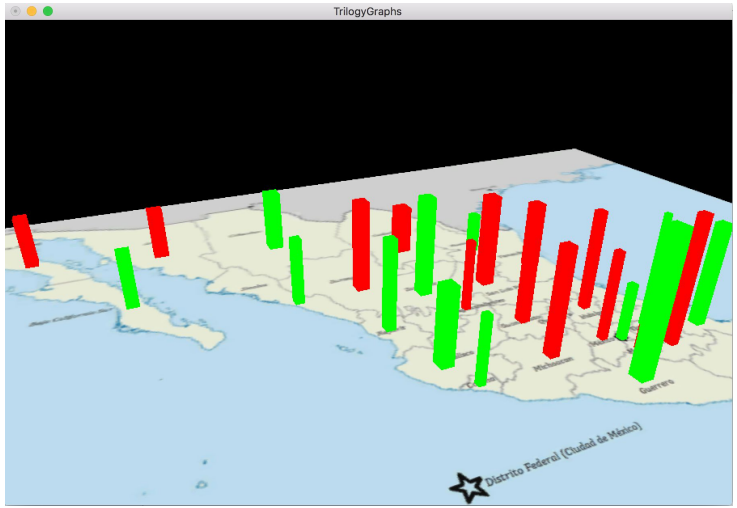


Rendering

Visualization

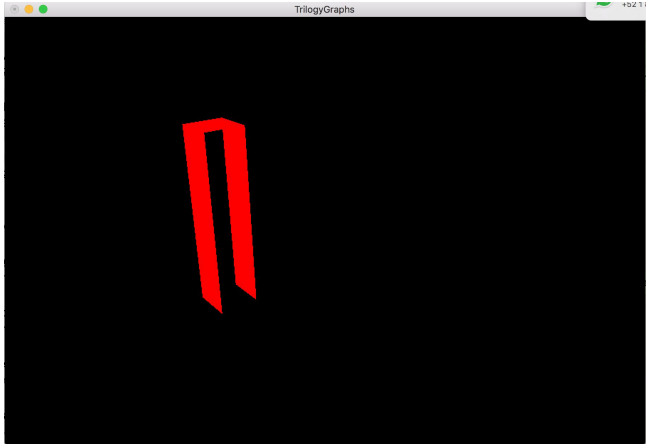


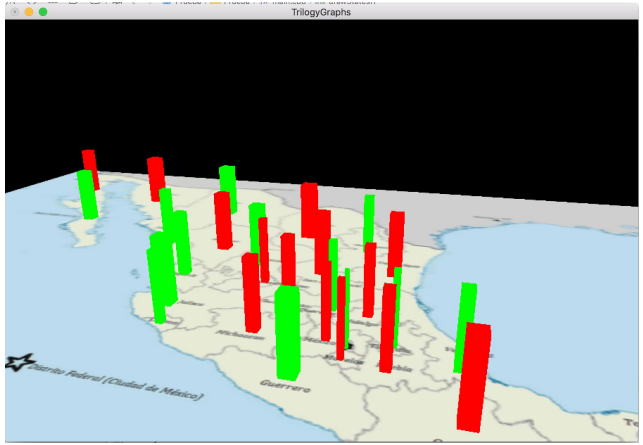




Visualization

- All of the components are made with QUADS.
- This means, vertexes are created manually, as well as faces of each bar, and the plane.





Visualization

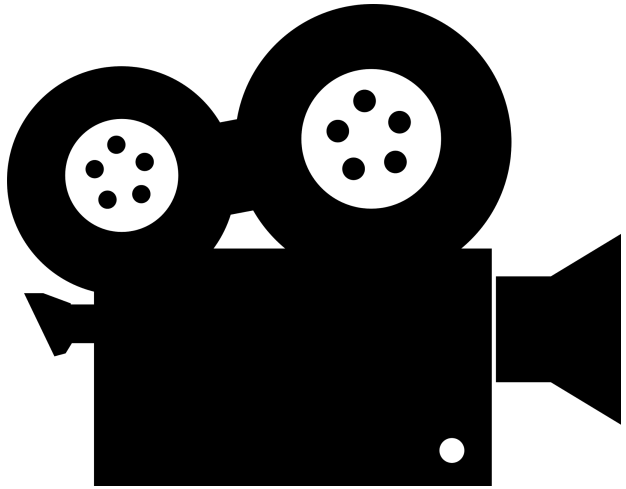
- The components can be scaled in their different axis, which can offer a better or worse **resolution**.
- Size of window can also be **reshaped.**
- All components are part of a whole, so rotations, translations and scales are performed as a whole. However, each bar has **it’s own height** and changes depending on the context

Colouring, Illumination and Cameras

Basic Cameras and Light

We use a perspective camera.

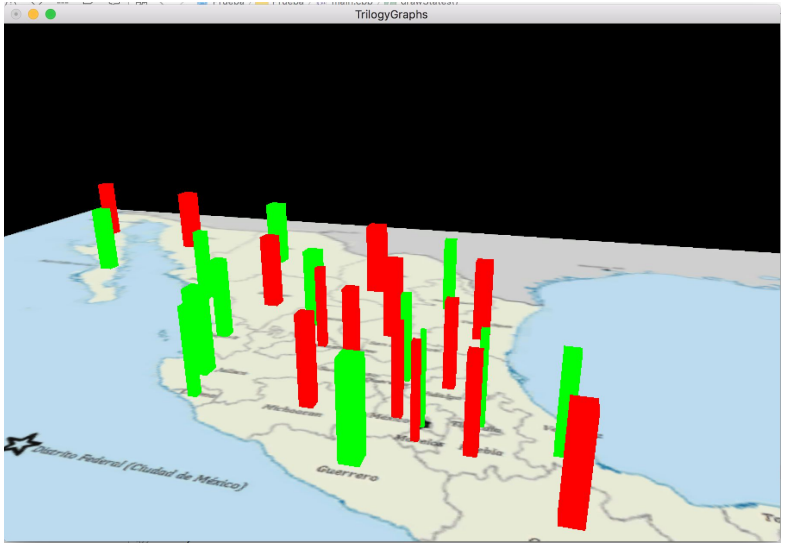
- 45º degrees.
- width * 1 / height ratio.
- 0.1 to the near clipping plane.
- 100 to the far clipping plane.
- Initial position: (0.0f, 1.0f, -5.0f)

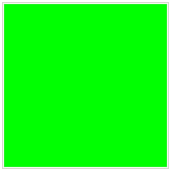


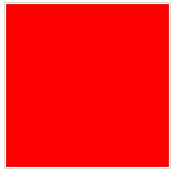
About the Light, we used a single source of light. It is the default light that GLUT's framework offers.

Cameras and Colour

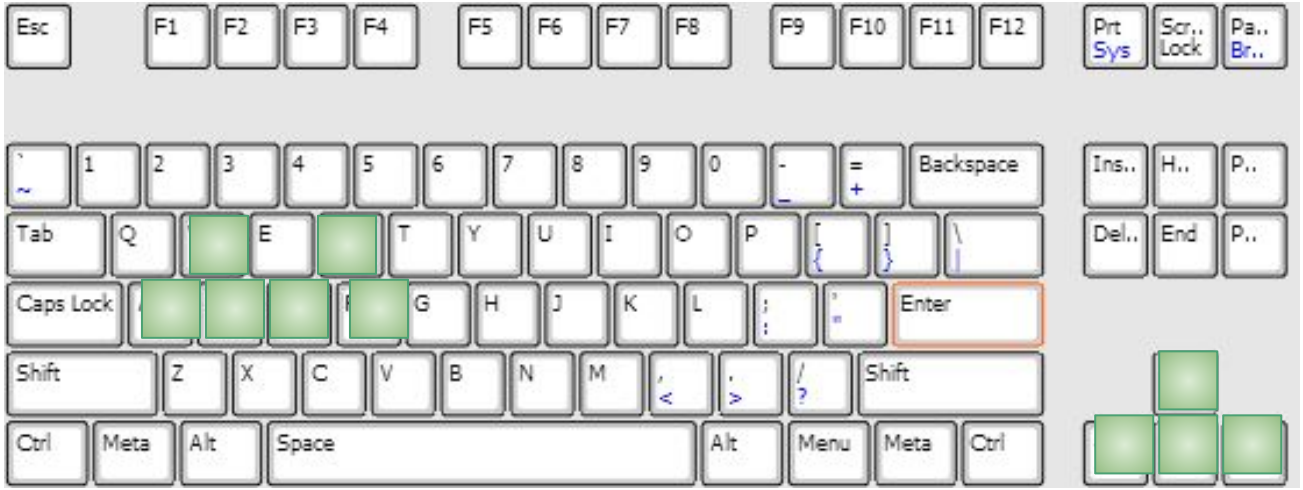
- Our Camera can be moved using the keyboard arrows.
- We use red and green bars to show our data. This colors contrast with our map. It's easier to see the bars.







GUI Navigation



WASD: Basic movement front, back, move to sides

R key move up, F key move down

Arrow Keys: Pan and Tilt (both directions each)

GUI Navigation

Scalling

- 5 & 4 in Y axis
- 7 & 6 in X axis
- 9 & 8 in Z axis

(upscaling and downscaling)

- 1, 2 & 3 changes context

GUI Navigation

- 1 - National population per state
- 2 - Unemployment rate per state
- 3 - Poverty rate per state

Storage

Storage

Session parameters are saved in a session.txt

Each time you open the application all the last session parameters are restored. (Analyzed variable*, camera setup)

Besides, data for each analyzed variable is **stored and retrieved** from different txt files that have the data of **all Mexico's States.**

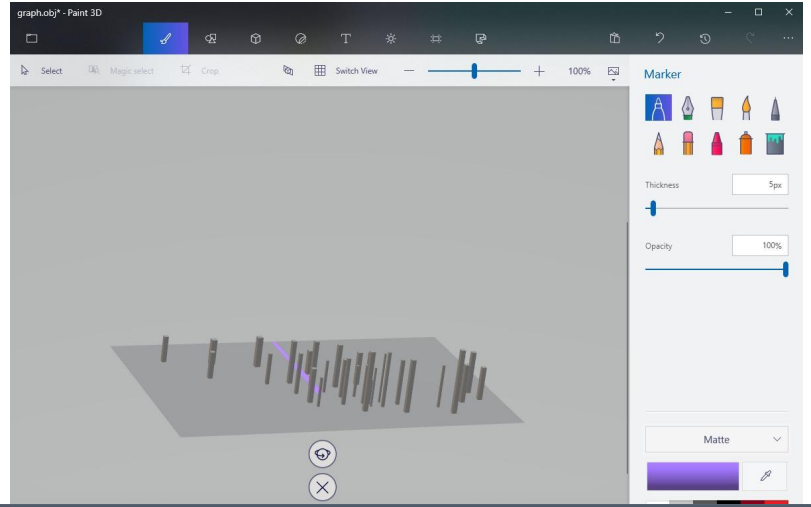
***Population, Poverty Rate or Unemployment**

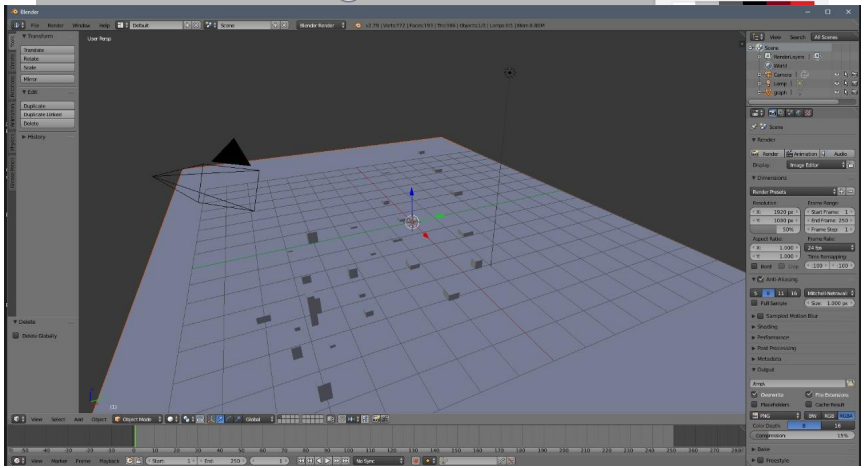
Storage

Using ‘p’ key you can generate an OBJ file that you can import to other software like Blender.

This file contains the position of each vertex and which vertices create each face.

We create our own method to create the file, saving each vertex position and the relation of vertices that form faces in arrays and then writing them into our output file.





Storage

Set vertex arrays values and faces values

```
41  float vertexArrayX[32][24];
42  float vertexArrayY[32][24];
43  float vertexArrayZ[32][24];
44  int facesArray[32][6][4];
45  float planeVertices[4][3];
```

Each bar has 24 vertices. We have 32 bars. The first 3 arrays store x, y, z of each vertex.

The fourth array store 4 vertices Index for each side (6) for each bar.

The last one store x, y, z for each vertex of the plane.

```
// FIRST FACE
glVertex3f( x_coordinate, 1.0f * height, z_coordinate);    // Top Right Of The Quad (Top)
vertexArrayX[barNumber][0] = x_coordinate;
vertexArrayY[barNumber][0] = 1.0f * height;
vertexArrayZ[barNumber][0] = z_coordinate;
facesArray[barNumber][0][0] = barNumber * 24 + 1;

glVertex3f(x_coordinate - square_base_size, 1.0f * height, z_coordinate);    // Top Left Of The Quad (Top)
vertexArrayX[barNumber][1] = x_coordinate - square_base_size;
vertexArrayY[barNumber][1] = 1.0f * height;
vertexArrayZ[barNumber][1] = z_coordinate;
facesArray[barNumber][0][1] = barNumber * 24 + 2;

glVertex3f(x_coordinate - square_base_size, 1.0f * height, z_coordinate - square_base_size);    // Bottom Left Of The Quad (Top)
vertexArrayX[barNumber][2] = x_coordinate - square_base_size;
vertexArrayY[barNumber][2] = 1.0f * height;
vertexArrayZ[barNumber][2] = z_coordinate - square_base_size;
facesArray[barNumber][0][2] = barNumber * 24 + 3;

glVertex3f( x_coordinate, 1.0f * height, z_coordinate - square_base_size);    // Bottom Right Of The Quad (Top)
vertexArrayX[barNumber][3] = x_coordinate;
vertexArrayY[barNumber][3] = 1.0f * height;
vertexArrayZ[barNumber][3] = z_coordinate - square_base_size;
facesArray[barNumber][0][3] = barNumber * 24 + 4;
```

```
855 void saveOBJFile() {
856     ofstream objFile("graph.obj");
857     if (objFile.is_open()) {
858         for(int i = 0; i < 32; i++) {
859             for(int j = 0; j < 24; j++) {
860                 objFile << "v " << vertexArrayX[i][j] << " " << vertexArrayY[i][j] << " " << vertexArrayZ[i][j] << endl;
861             }
862         }
863
864         for (int i = 0; i < 4; i++) {
865             objFile << "v " << planeVertices[i][0] << " " << planeVertices[i][1] << " " << planeVertices[i][2] << endl;
866         }
867
868         for(int i = 0; i < 32; i++) {
869             for(int j = 0; j < 6; j++) {
870                 objFile << "f " << facesArray[i][j][0] << " " << facesArray[i][j][1] << " " << facesArray[i][j][2] << " " << facesArray[i][j][3] << endl;
871             }
872         }
873         // Plane face
874         objFile << "f " << 32 * 6 * 4 + 1 << " " << 32 * 6 * 4 + 2  << " " << 32 * 6 * 4 + 3 << " " << 32 * 6 * 4 + 4 << endl;
875     }
876     else{
877         printf("There's a problem while creating new session file.");
878     }
879     objFile.close();
880 }
```

Write the OBJ File using ofstream methods. Called when user press ‘p’.

Conclusions

Conclusions

- We learn something new about using Open GL. We put in practice what we learn through tutorials and homeworks.
- Communication and Team Work: **ESSENTIAL**
- It is a challenge to create a scene without viewing the elements on the screen.
- We visualize important information about our country and realized about Mexico’s reality per states.
- Data visualization is a very important tool that help us to comprehend information in a very simple way.

References

References

- http://www.stps.gob.mx/bp/secciones/conoce/areas_atencion/areas_atencion/web/pdf/perfiles/perfil%20nacional.pdf
- https://en.wikipedia.org/wiki/List_of_Mexican_states_by_population
- http://www.martinreddy.net/gfx/3d/OBJ.spec