A08 – 3D Maze generation

The Vulkan application whose source code is contained in file `A08.cpp`, generates a random maze and allows the player to explore it. In file `mazeGen.hpp`, you have to implement the procedure that create the 3D mesh of the maze, starting from a textual representation. In particular, you have to work inside method `void Assignment08::createMazeMesh(…)`:

```
The procedure gets in input the number of rows <row> of the maze, and the number of
columns <col>. Element <maze[r][c]>, with 0 <= r <= row-1 and 0 <= c <= col-1 contains:
   • Symbol ' ' if there is an empty space
   • Symbol '#' if there is a wall
     //

Example: The following piece of code executes the instruction in the ... section if there
is a wall in position r=5, c=7:

int r, c;
r = 5; c = 7;
if(maze[r][c] == '#') {
      ...
}
```

You have to create the vertices inside array `vPos`, by placing in order the x, y and z component of a vertex, and create the index buffer inside array `vIdx`. The mesh is encoded as an indexed triangle list, so any group of three successive indices defines a new triangle. The sample code, creates a square as a set of two triangles, sharing four vertices:

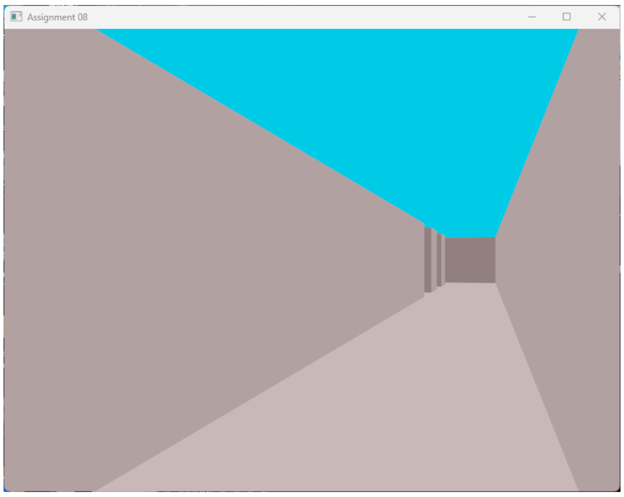
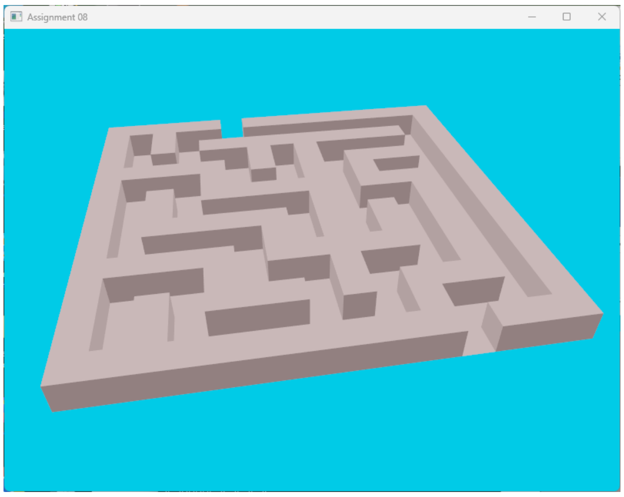
```
      // Fill array vPos with the positions of the vertices of the mesh
      vPos.push_back(0.0); vPos.push_back(0.0); vPos.push_back(0.0);      // vertex 0
      vPos.push_back(1.0); vPos.push_back(0.0); vPos.push_back(0.0);      // vertex 1
      vPos.push_back(0.0); vPos.push_back(1.0); vPos.push_back(0.0);      // vertex 2
      vPos.push_back(1.0); vPos.push_back(1.0); vPos.push_back(0.0);      // vertex 3

      // Fill the array vIdx with the indices of the vertices of the triangles
      vIdx.push_back(0); vIdx.push_back(1); vIdx.push_back(2);     // First triangle
      vIdx.push_back(1); vIdx.push_back(2); vIdx.push_back(3);     // Second triangle
```

- You have to replace this section, with an algorithm that builds the maze, starting from its textual description, and which produces the correct vertex and index buffers.
- Try to include all the walls, but leave the ceiling of the maze “open air”.
- Try to use the least possible number of vertices and triangles, reusing vertices whenever it is possible.

You can move the view using the same keys as in *Assigment0*:

ESC – quit the application						
	W: forward		**R**: up		↑: look up	
A: left	**S**: backward	**D**: right	F: down	←: look left	↓: look down	→: look right

If everything work, you should be able to have screenshots like the following:


Assignment 08

Assignment 08