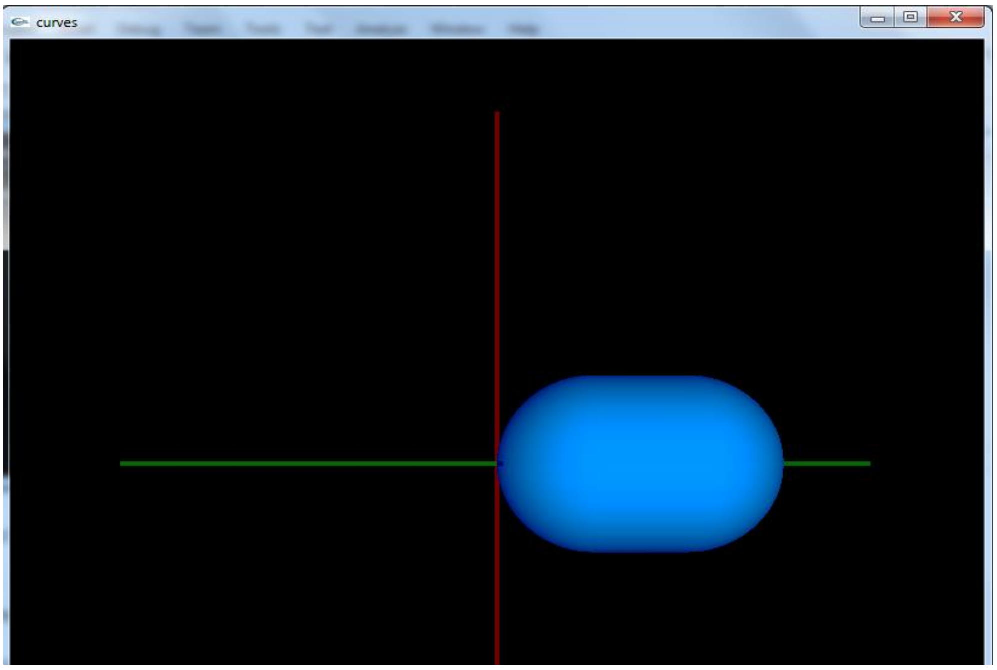
Assignment 1

In this assignment you are required to implement a simple Piecewise Cubic Bezier Curve drawing program in OpenGL using GLM library.



Read carefully and understand the theory behind cubic Bezier curve. You will use cubic Bezier curves to design the contour of your shape.

1. Download the code from Github and try to compile it on your PC. You may need to download GLFW suitable to your Visual Studio version (compile and replace the glfw3.lib in res/libs). See a link to a Youtube guide at useful links on the course site.
2. General instructions:
   a. **DisplayGLFW** responsibles for opening a window and get the input/output functions. This project based on GLFW library. Lib project
   b. **EngineVideoGames** is the graphic engine. It is using Glew OpenGL library. Lib project
   c. You must add exe project which contain main function like main in **Game** project, Class that extend scene and has an **update** function, and InputManager file which includes all the input/output functions (just copy Game's Input/Output file and add more keys and functionality if needed.
   d. Important functions in **Scene class:**
      AddShape: Adds shape which construct in the program (like your Bezier curve or surface)
      AddShapeFromFile: Adds shape from OBJ file.
      AddShapeCopy: duplicates existing shape.
      AddShader: Uploads a shader program from file to the GPU
      Update: Update uniform data on the shader (you must implement it)
      Draw: render scene using a specific shader and camera (point of view)
      ShapeTransformation: translate, rotate or scale a shape using enum. Second parameter is the amount (float). The function will act on the shape in **pickedShape** location in shapes vector.

e. **Picking**: I implemented color picking for picking of a 3D object with the mouse.
   Left mouse button Picking rotates the picked shape.
   Right mouse button Picking translates the shape parallel to the screen plane.
   Pick and **scroll** translates the shape in perpendicular direction to the screen plane.

3. Your goal is to implement the functions appears in Bezier1D.h:

```
IndexedModel GetLine(int resT);
//generates model for rendering using MeshConstructor::initLin
LineVertex GetVertex(int segment,int t);
//returns point on curve in the requested segment for value of t
LineVertex GetControlPoint(int segment,int indx);
//returns a control point in the requested segment. indx will be
0,1,2,3, for p0,p1,p2,p3

glm::vec3 GetVelosity(int segment,int t);
//returns the derivative of the curve in the requested segment for value
of t
void MoveControlPoint(int segment, int indx,bool preserveC1);
//change the positon of one control point. when preserveC1 is true it
may affect other control points
```

   And Bezier2D.h:

```
IndexedModel GetSurface(int resT,int resS);
//generates model for rendering using MeshConstructor::initMeshs
Vertex GetVertex(int segmentT,int segmentS,int t,int s);
//returns point on surface in the requested segments for value of t & s
glm::vec3 GetNormal(int segmentT,int segmentS,int t,int s);
//returns point on surface in the requested segments for value of t & s
```

   The Mesh.h for data structures decleration.

4. **After** implementation build the following framework:
   a. Go to MeshConstructor constructor and uncomment `BezierLine and BezierSurface.`
   b. Render 3 segments (or more) Bezier curve. Put small cube (or other 3D shape) on each control point.
   c. The user can pick each control point with **left mouse button** and move it in XY plane. The curve will change appropriately. First and last control point of the curve, the whole curve will preserve C1 (derivative continuity).
   d. When pressing **'space'** render the 2D Bezier surface based on the curve you edit. Choose the symmetric axis of the constructed shape to lay on the vector between the first point p_0 of the first segment and the last point p_3 of the last segment.

5. Upload Bezier1D.hpp, Bezier1D.cpp, Bezier2D.hpp and Bezier2D.cpp to the submission system. The assignment will be checked during the class.

6. **Extension Ideas (not mandatory):**
   a. If the user presses on control points p_0 with the **right mouse button** the segment define by $p_0, p_1 \dots p_3$ will delete. Minimum segments number is 2.
   b. When the user presses inside the convex hull of one of the Bezier path with the **middle mouse button** he can move the entire segment on XY plan. The near Bezier path will be affected (p2 of the previous path and p1 of the next path will move properly). The entire curve will always be continuous and preserve C1.
   c. When the user presses inside the convex hull of one of the Bezier path with the **right mouse button** the path will split in the middle (t = 0.5) to 2 different Bezier paths with the same degree, according to the rule you have seen in P.S. Maximum sub curve (parts) number is 6.
   d. When the user presses inside the convex hull of one of the Bezier path with the **left mouse button** it will become a straight line when all control points will locate on the line in equal distance spaces between them. P1 and p2 position will not be changed.