Project 1


```
29  // ATTN 1A is the general place in the program where you have to change the code base to satisfy a Task of Project 1A.
30  // ATTN 1B for Project 1B. ATTN 1C for Project 1C. Focus on the ones relevant for the assignment you're working on.
31
32  typedef struct Vertex {
33      float Position[4];
34      float Color[4];
35      void SetCoords(float *coords) {
36          Position[0] = coords[0];
37          Position[1] = coords[1];
38          Position[2] = coords[2];
39          Position[3] = coords[3];
40      }
41      void SetColor(float *color) {
42          Color[0] = color[0];
43          Color[1] = color[1];
44          Color[2] = color[2];
45          Color[3] = color[3];
46      }
47  };
```

Project 1


```
48
49  // ATTN: use POINT structs for cleaner code (POINT is a part of a vertex)
50  // allows for (1-t)*P_1+t*P_2  avoiding repeat for each coordinate (x,y,z)
51  typedef struct point {
52      float x, y, z;
53      point(const float x = 0, const float y = 0, const float z = 0) : x(x), y(y), z(z){};
54      point(float *coords) : x(coords[0]), y(coords[1]), z(coords[2]){};
55      point operator -(const point& a) const {
56          return point(x - a.x, y - a.y, z - a.z);
57      }
58      point operator +(const point& a) const {
59          return point(x + a.x, y + a.y, z + a.z);
60      }
61      point operator *(const float& a) const {
62          return point(x * a, y * a, z * a);
63      }
64      point operator /(const float& a) const {
65          return point(x / a, y / a, z / a);
66      }
67      float* toArray() {
68          float array[] = { x, y, z, 1.0f };
69          return array;
70      }
71  };
```

Project 1 global arrays


```
// ATTN: INCREASE THIS NUMBER AS YOU CREATE NEW OBJECTS
const GLuint NumObjects = 1; // Number of objects types in the scene

// Keeps track of IDs associated with each object
GLuint VertexArrayId[NumObjects];
GLuint VertexBufferId[NumObjects];
GLuint IndexBufferId[NumObjects];

size_t VertexBufferSize[NumObjects];
size_t IndexBufferSize[NumObjects];
size_t NumVerts[NumObjects];    // Useful for glDrawArrays command
size_t NumIdcs[NumObjects]; // Useful for glDrawElements command

// Initialize ---  global objects -- not elegant but ok for this project
const size_t IndexCount = 4;
Vertex Vertices[IndexCount];
GLushort Indices[IndexCount];
```

Project 1 init


```
218
219     // ATTN: create VAOs for each of the newly created objects here:
220     // for several objects of the same type use a for-loop
221     int obj = 0;  // initially there is only one type of object
222     VertexBufferSize[obj] = sizeof(Vertices);
223     IndexBufferSize[obj] = sizeof(Indices);
224     NumIdcs[obj] = IndexCount;
225
226     createVAOs(Vertices, Indices, obj);
227 }
```

Project 1 init


```
272  void createObjects(void) {
273      // ATTN: DERIVE YOUR NEW OBJECTS HERE:  each object has
274      // an array of vertices {pos;color} and
275      // an array of indices (no picking needed here) (no need for indices)
276      // ATTN: Project 1A, Task 1 == Add the points in your scene
277      Vertices[0] = { { 1.0f, 1.0f, 0.0f, 1.0f }, { 1.0f, 0.0f, 0.0f, 1.0f } };
278      Vertices[1] = { { -1.0f, 1.0f, 0.0f, 1.0f }, { 0.0f, 1.0f, 0.0f, 1.0f } };
279      Vertices[2] = { { -1.0f, -1.0f, 0.0f, 1.0f }, { 0.0f, 0.0f, 1.0f, 1.0f } };
280      Vertices[3] = { { 1.0f, -1.0f, 0.0f, 1.0f }, { 1.0f, 1.0f, 1.0f, 1.0f } };
281
282      Indices[0] = 0;
283      Indices[1] = 1;
284      Indices[2] = 2;
285      Indices[3] = 3;
286
287      // ATTN: Project 1B, Task 1 == create line segments to connect the control points
288
289      // ATTN: Project 1B, Task 2 == create the vertices associated to the smoother curve generated by subdivision
290
291      // ATTN: Project 1B, Task 4 == create the BB control points and apply De Casteljau's for their corresponding for each piece
292
293      // ATTN: Project 1C, Task 3 == set coordinates of yellow point based on BB curve and perform calculations to find
294      // the tangent, normal, and binormal
295  }
```

Project 1 pickVertex


```
    // Convert the color back to an integer ID
    gPickedIndex = int(data[0]);

    // ATTN: Project 1A, Task 2
    // Find a way to change color of selected vertex and
    // store original color

    // Uncomment these lines if you wan to see the picking shader in effect
    // glfwSwapBuffers(window);
    // continue; // skips the visible rendering
}
```

Project 1 init


```
// ATTN: Project 1A, Task 3 == Retrieve your cursor position, get corresponding world coordinate, and move the point accordingly

// ATTN: Project 1C, Task 1 == Keep track of z coordinate for selected point and adjust its value accordingly based on if certain
// buttons are being pressed
void moveVertex(void) {
    glm::mat4 ModelMatrix = glm::mat4(1.0);
    GLint viewport[4];
    glGetIntegerv(GL_VIEWPORT, viewport);
    glm::vec4 vp = glm::vec4(viewport[0], viewport[1], viewport[2], viewport[3]);

    if (gPickedIndex >= IndexCount) {
        // Any number > vertices-indices is background!
        gMessage = "background";
    }
    else {
        std::ostringstream oss;
        oss << "point " << gPickedIndex;
        gMessage = oss.str();
    }
}
```

Project 1 renderScene


```
glBindVertexArray(VertexArrayId[0]);    // Draw Vertices
glBufferSubData(GL_ARRAY_BUFFER, 0, VertexBufferSize[0], Vertices);     // Update buffer data
glDrawElements(GL_POINTS, NumIdcs[0], GL_UNSIGNED_SHORT, (void*)0);
// // If don't use indices
// glDrawArrays(GL_POINTS, 0, NumVerts[0]);

// ATTN: OTHER BINDING AND DRAWING COMMANDS GO HERE
// one set per object:
// glBindVertexArray(VertexArrayId[<x>]); etc etc

// ATTN: Project 1C, Task 2 == Refer to https://learnopengl.com/Getting-started/Transformations and
// https://learnopengl.com/Getting-started/Coordinate-Systems - draw all the objects associated with the
// curve twice in the displayed fashion using the appropriate transformations

glBindVertexArray(0);
```

Project 1 renderScene


```
// DRAGGING: move current (picked) vertex with cursor
if (glfwGetMouseButton(window, GLFW_MOUSE_BUTTON_LEFT)) {
    moveVertex();
}

// ATTN: Project 1B, Task 2 and 4 == account for key presses to activate subdivision and hiding/showing f
// for respective tasks

// DRAWING the SCENE
createObjects();    // re-evaluate curves in case vertices have been moved
renderScene();
```