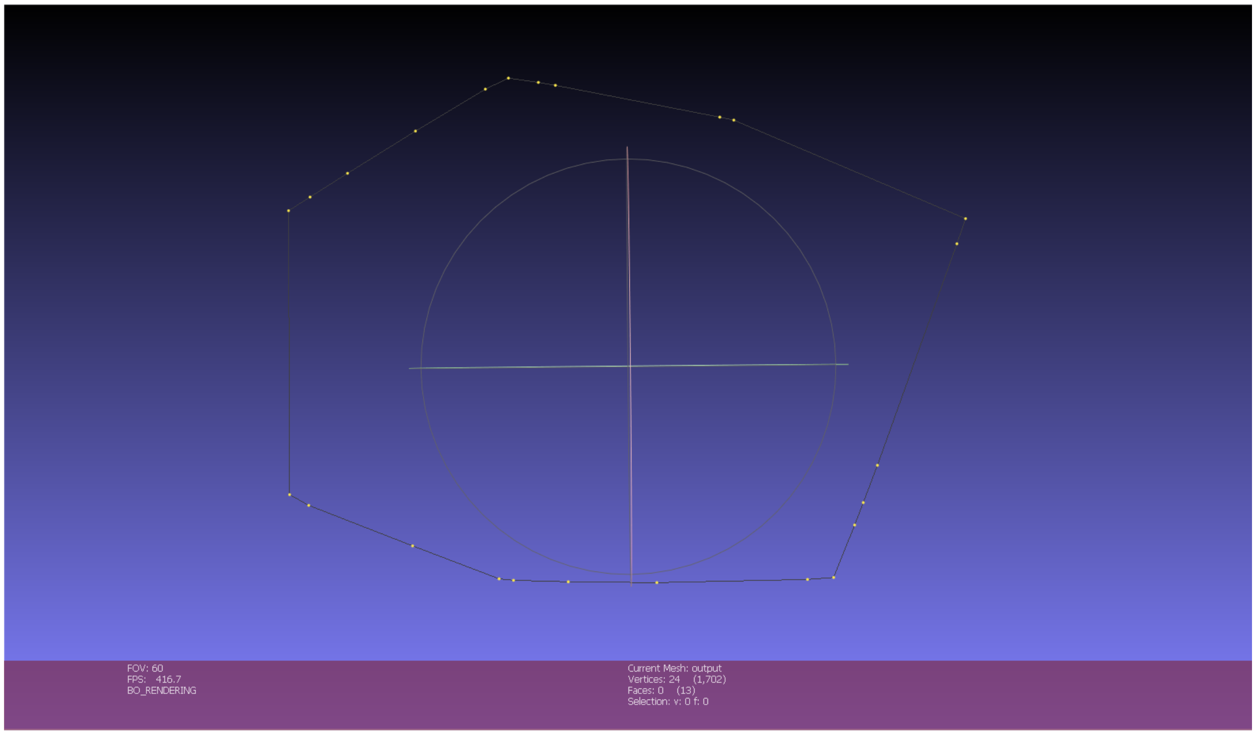
CSC305
Xinlu Chen
Assignment 1

Name and version of operating system and compiler:
Windows 11 for x64-based Systems
cmake version 3.16.3, g++ (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0

compile command:

```
mkdir build
cd build
 cmake ..
make
./convex_hull ../data/points.xyz output.obj
./point_in_polygon ../data/points.xyz ../data/polygon.obj result.xyz
```

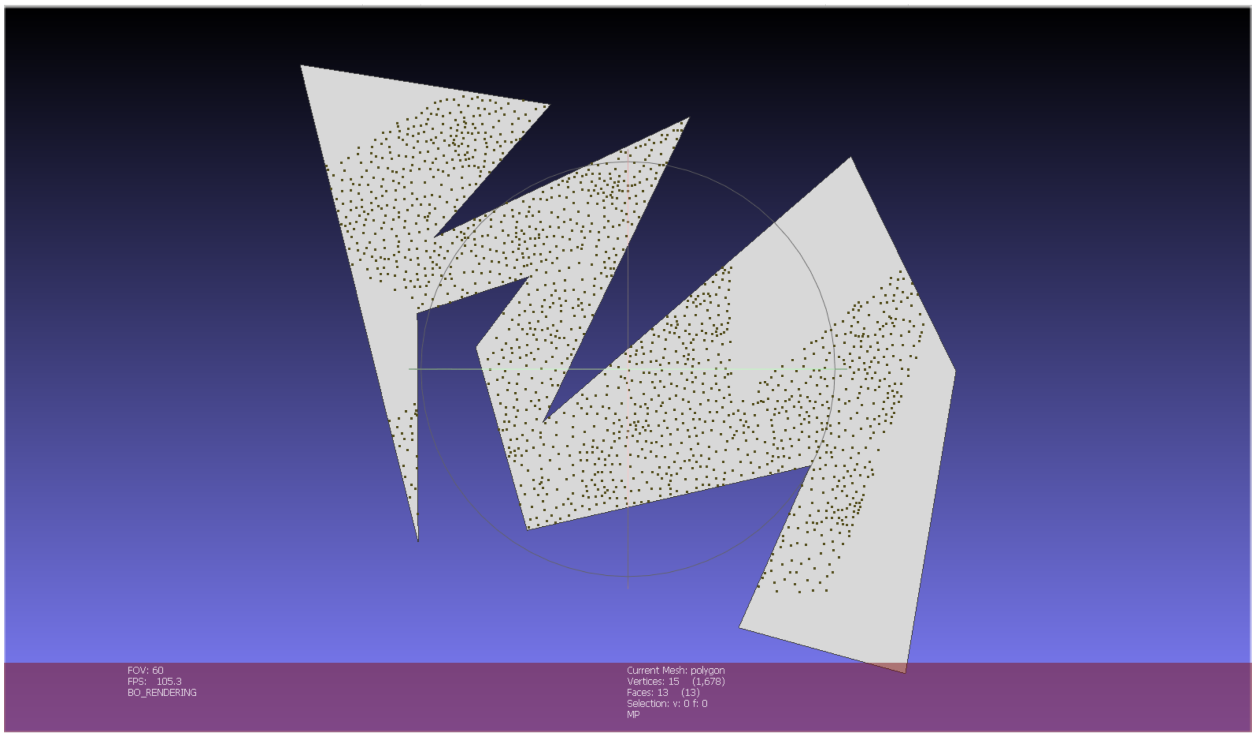
Screenshots of results with description:



```
FOV: 60
FPS:   416.7
BO_RENDERING
```

```
Current Mesh: output
Vertices: 24    (1,702)
Faces: 0    (13)
Selection: v: 0 f: 0
```

Ex1. convex hull, overall, has 24 vertexes



Ex2. Point In Polygon, overall, has 1663 points.

src/hull/main.cpp

Brief summary:

1. Find the lower - left point in the input point cloud P0.
2. Sort all input points counter - clockwise with respect to P0.
3. Greedily compute the convex - hull polygon by iterating over the sorted points.Whenever a "right-turn" is encountered, pop the middle - point from the hull.

Function description:

#
bool check_equal(Point& p1, Point& p2, Point& p0)
//check the phase angle of p1 and p2 base on p0.
//If p1 and p2 have same phase angle, return true, false otherwise.

#
Point nextToTop(std::stack<Point>& s)
//find the second top point value on stack s

#
double inline det(const Point &u, const Point &v)
//not being used, originally include in source file

#
Struct compare{bool operator ()(const Point& p1, const Point& p2)}
//comparison structure for two points, point with smaller phase angle goes first

#
int orientation(Point a, Point b, Point c)
//use formula value = (y2-y1)*(x3-x2)-(x2-x1)*(y3-y2) to check a,b,c 3 points
//if value >0, clockwise, not salientangle, return 2
//if value <0, counterclockwise, salientangle, return 1
//else, collinear, return 3

#
bool inline salientAngle(Point a, Point b, Point c)
//if 3 points are counterclockwise, salientangle, return true, false otherwise

#
Polygon convex_hull(std::vector<Point> &points)
//find lower-left most point, and sort points using oder in compare.
//make a stack, and use salientAngle func to pop and push to get the final hull

#
std::vector<Point> load_xyz(const std::string &filename)
//get points cloud from xyz file. Return points

#
save_obj(const std::string &filename, Polygon &poly)
//save the polygon into the file

#
int main(int argc, char * argv[])
//call functions and get input argument info

src/inside/main.cpp

<u>Brief summary:</u>
Make a point outside the polygon Q, to check if point p is inside of the polygon or not, if the intersection points of line segment PQ with polygon is odd number, it is inside the polygon, outside otherwise. If the intersection points is vertex point, count it as 2.

<u>Function description:</u>

#

```
double inline det(const Point &u, const Point &v)
//return u and v’s determinant value

#
bool intersect_segment(const Point &a, const Point &b, const Point &c, const Point &d, Point
&ans)
//Return true iff [a,b] intersects [c,d], and store the intersection in ans
//use formula   x = (b1c2-b2c1)/(a1b2-a2b1)
                y = (a2c1-a1c2)/(a1b2-a2b1)
//to calculate the intersection points of two lines
//then check the intersection point is on the two line segments or not
//if yes, return true, else, return false

#
bool on_vertex(const Polygon& poly, const Point& p)
//check if point p is on the vertex of poly or not. If yes, return true, false otherwise

#
bool is_inside(const Polygon &poly, const Point &query)
//check if point query is inside of poly or not using algorithm of even-odd rule.
//If yes, return true, false otherwise

#
std::vector<Point> load_xyz(const std::string &filename)
//load points from xyz file

#
Polygon load_obj(const std::string& filename)
//load polygon from obj file

#
void save_xyz(const std::string &filename, const std::vector<Point> &points)
//save the in-polygon points into xyz file

#
int main(int argc, char * argv[])
//call functions and get input arguments
```