1. Introduction

I've made a couple of improvements to the project to provide a unified way to compile the project on Windows and Linux. Moreover, I've added a square perimeter and circumference random distribution to random utilities to draw squared and round point clouds.

I've included precompiled binaries for Windows and Linux.

2. Compilation on Windows

- Install Visual Studio from `https://visualstudio.microsoft.com/downloads/`
- Install VCPKG from `https://vcpkg.io/en/getting-started.html`
- Add VCPKG folder to PATH:
- Option 1: Command Line tools

```
    REM Assuming that VCPKG cloned and bootstrapped in c:\src\vcpkg
    REM setx for the global environment, set for the local
    setx PATH c:\src\vcpkg;%PATH%
    set  PATH c:\src\vcpkg;%PATH%
```

- Option 2: PowerShell

```
    # Assuming that VCPKG cloned and bootstrapped in c:\src\vcpkg
    [Environment]::SetEnvironmentVariable("PATH", "c:\src\vcpkg;${PATH}", "Machine")
        Set-Item -Path Env:PATH -Value "c:\src\vcpkg;${PATH}"
```

- Option 3: Manually in **System Properties –> Environment Variables**
- Navigate to the folder with the project
- Run *build.bat*
- Open *build*
  EDGSG.sln in Visual Studio to work with the source code

3. Compilation on Linux

- Install VCPKG from `https://vcpkg.io/en/getting-started.html`
- Add VCPKG folder to System PATH
- Option 1: Temporary local environment

```
        # Assuming that VCPKG cloned and bootstrapped in ~/vcpkg
        export PATH="~/vcpkg;${PATH}"
```

- Option 2: Local environment and Bash profile

```
        # Assuming that VCPKG cloned and bootstrapped in c:\src\vcpkg
        export PATH="~/vcpkg;${PATH}"
        echo 'export PATH="~/vcpkg;${PATH}"' >> ~/.bashrc
```

- Navigate to the folder with the project
- Run build.sh

Important Note: the VCPKG requests installation of additional packages

4. Point Clouds

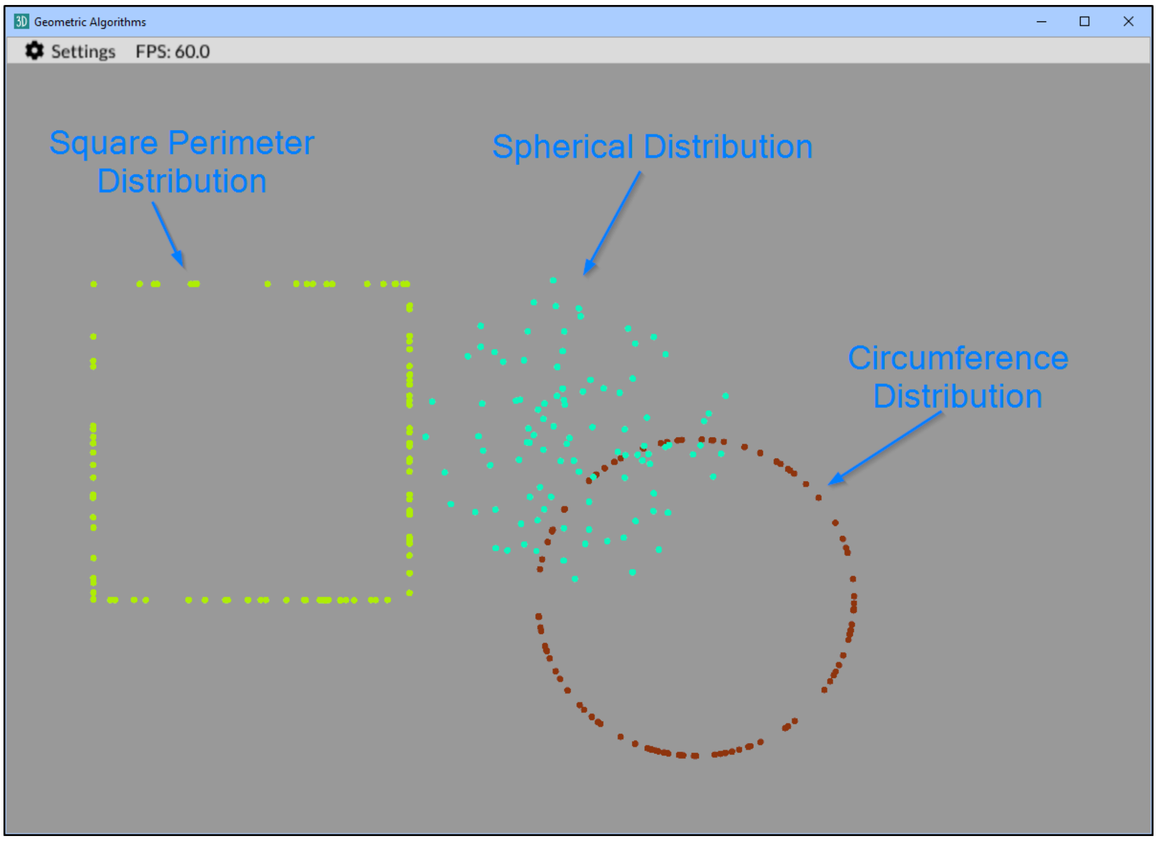

Figure 1: Point Clouds.

I've included multiple ways to generate point clouds: using Hemispherical random distribution, Spherical random distribution, circle random distribution, and square random distribution. The first point cloud is generated in a Spherical manner. Following point clouds are smaller and generated squarely for uneven point cloud IDs and circle manner for even IDs.

Point clouds could be saved to a file and loaded to a file using the following pattern:

```
    auto* pointCloud = new PointCloud("PointCloud" + std::to_string(pcIdx) + ".txt");
    ...
    ...
    pointCloud->save("PointCloud" + std::to_string(pcIdx) + ".txt");
```

5. Segment, Ray, Line

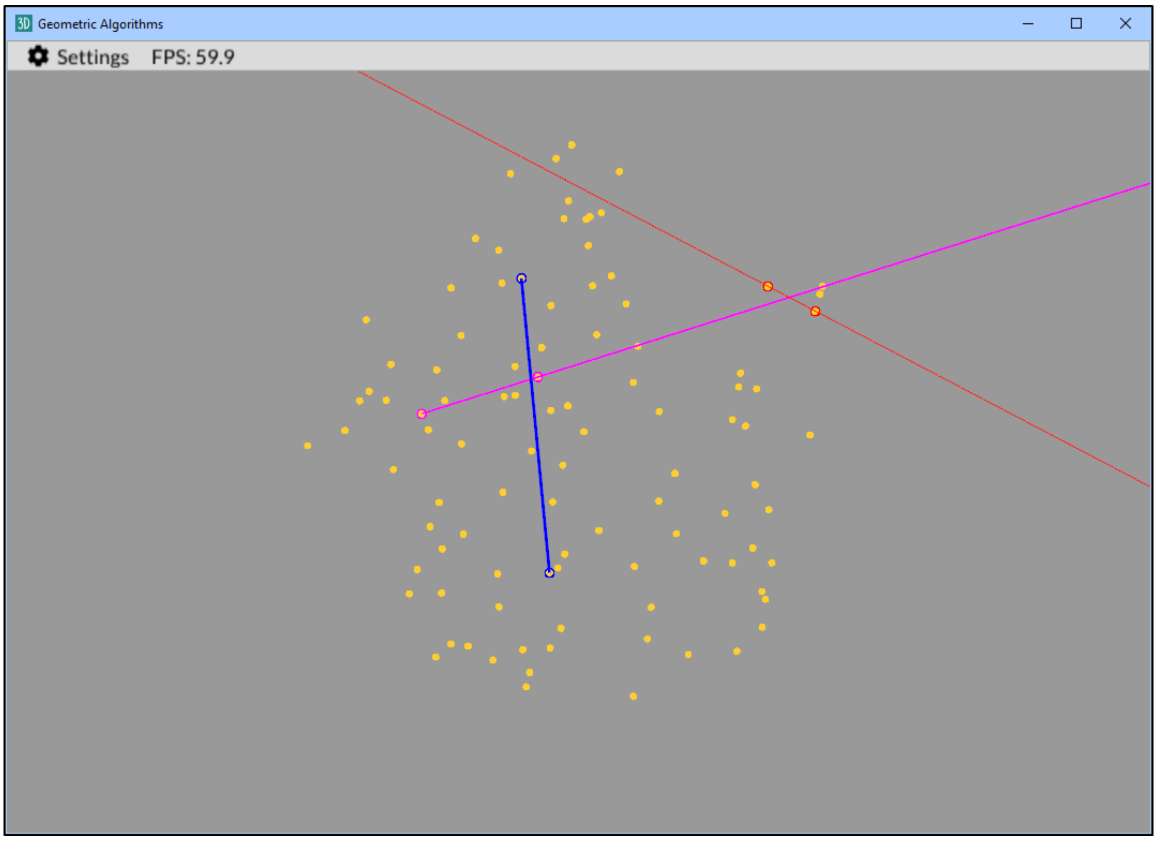

Figure 2: Blue Segment, red Line, and magenta Ray.

To generate Segment, Ray, and Line, I've taken 6 random points from the central point cloud. The point selection is totally random, so one point, potentially, can belong to all three objects. To distinguish them, I draw them in different sizes. Additionally, I draw the origin and destination points in bigger sizes to emphasize them.

6. Convex Polygon

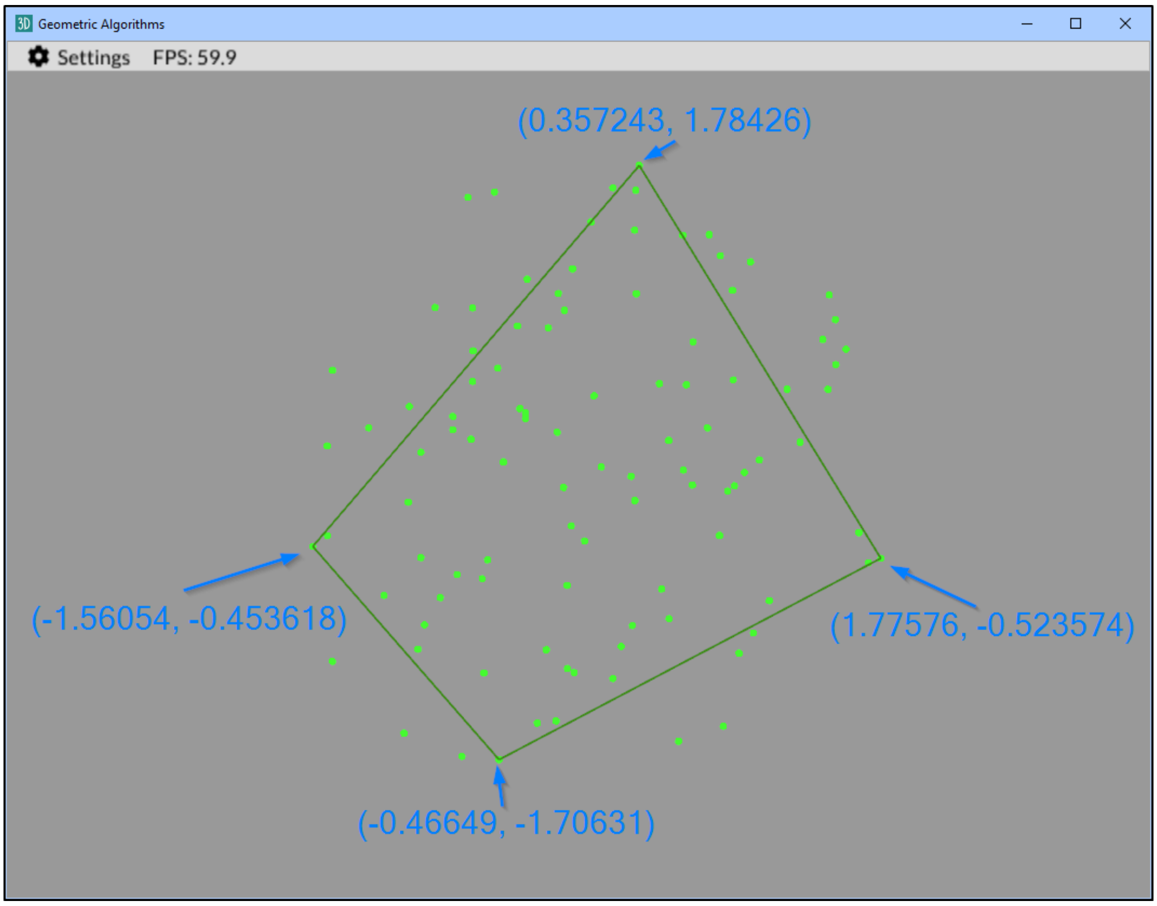

Figure 3: Convex Polygon from 4 points of point cloud.

Four points for convex were selected from the point cloud using maximum and minimum x and y values.

7. Bezier Curve

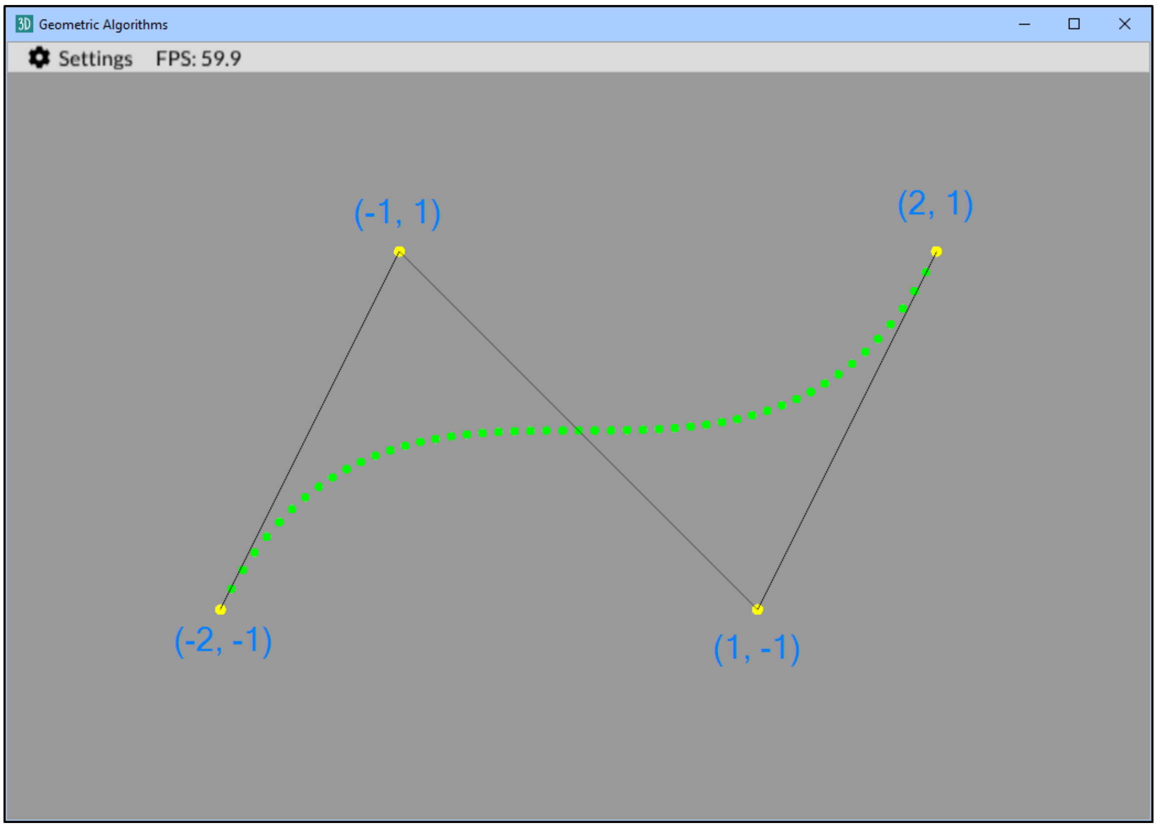

Figure 4: Bezier Curve.

I've provided commented-out predefined points for the Bézier curve. In the picture, I've carefully selected them to show the beauty of the Bézier curve. The point cloud could be used as a source of control points, but there are too many to make a good curve. Additionally, I draw control points and control segments.

8. Everything on one picture

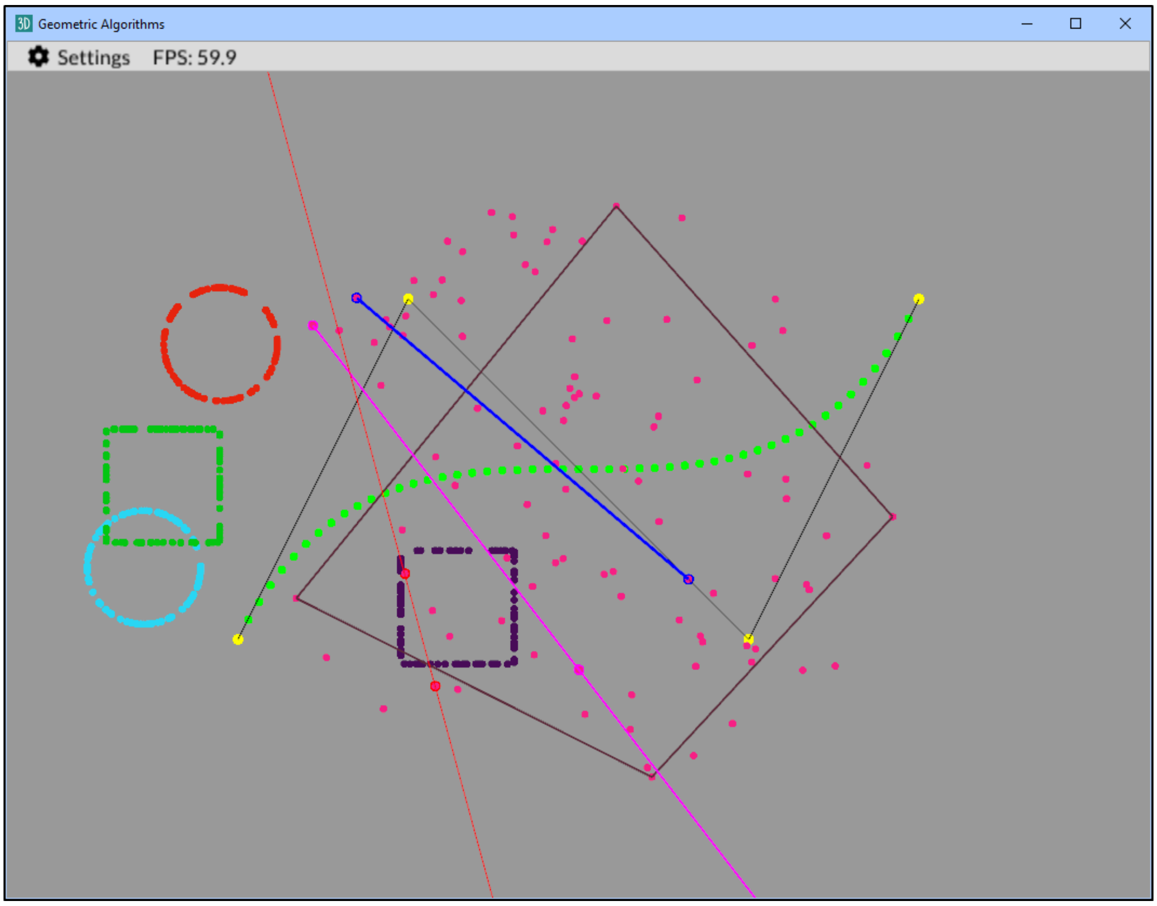

Figure 5: All in one.