University of Utah
School of Computing

CS 4960 Project #2 Spring 2016

Due Feb 16, 2016 at the start of class

Please contact the instructor Bei (beiwang@sci.utah.edu) or TA Vikram at (vikram.raj@utah.edu) for questions regarding the project. Please contact Vikram for questions regarding Blender specifically.

The project submission should include all your source code, the output files of CGAL (according to Blender format requirement described later), the input files to Blender (using the parser.c attached).

For bonus points, your project submission could include the visualization produced by Blender.

For all subsequent projects, Blender visualization of the output is required.

For this project, you get 15 points total, with additional 5 bonus points.

1 Computing and Visualizing 2D Convex Hull

Use CGAL 2D Convex Hull Library: `http://doc.cgal.org/latest/Convex_hull_2/index.html` to complete the following tasks, you may use any convex hull algorithms.

(a) (5 points): Generate a set of 20 points sampled randomly from a 2D circle, compute its convex hull and output the boundary of the convex hull (e.g. vertices and edges) into a file that could be converted to be suitable input for Blender to visualize.
Hint: You may use CGAL Random_points_on_circle_2.

(b) (5 points): repeat the above process, but this time introduce some small perturbation to the random points generated above.

(c) (Bonus 2 points): include correct visualization of the convex hulls from (a) and (b) using Blender.

2 Computing and Visualizing 3D Convex Hull

Use CGAL 3D Convex Hull Library: `http://doc.cgal.org/latest/Convex_hull_3/index.html` to complete the following task:

(a) (5 points): Generate a set of 300 random points chosen from a sphere of radius 100, compute its convex hull and output the boundary of the convex hull (e.g. vertices, edges) into a file that could be converted to be suitable input for Blender to visualize.
Hint: Check out the example via CGAL manual.

(b) (Bonus 1 point) if (a) includes correct visualization of the convex hulls using Blender.

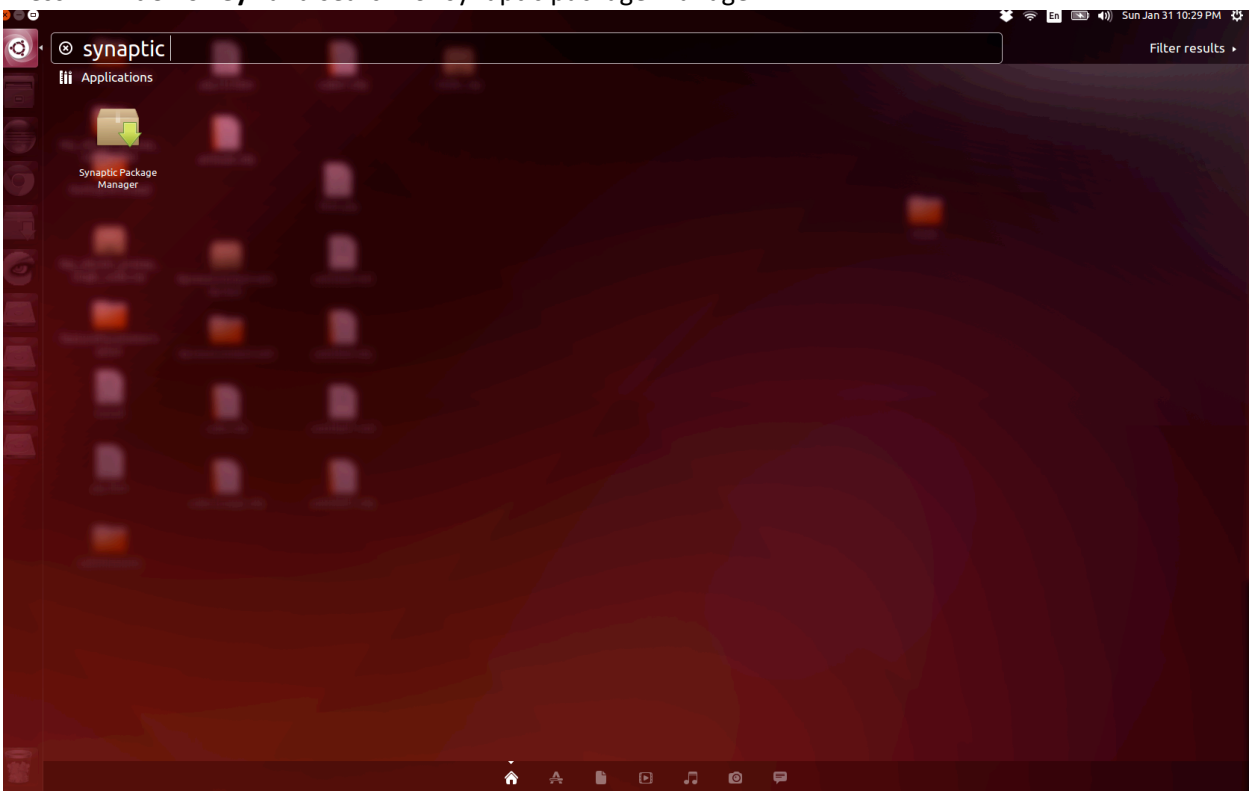
(b) (Bonus 2 points) Repeat same process as (a), but this time using a complex 3D point set, for example, points from `http://pointclouds.org/`. And include its Blender visualization result.

Windows :

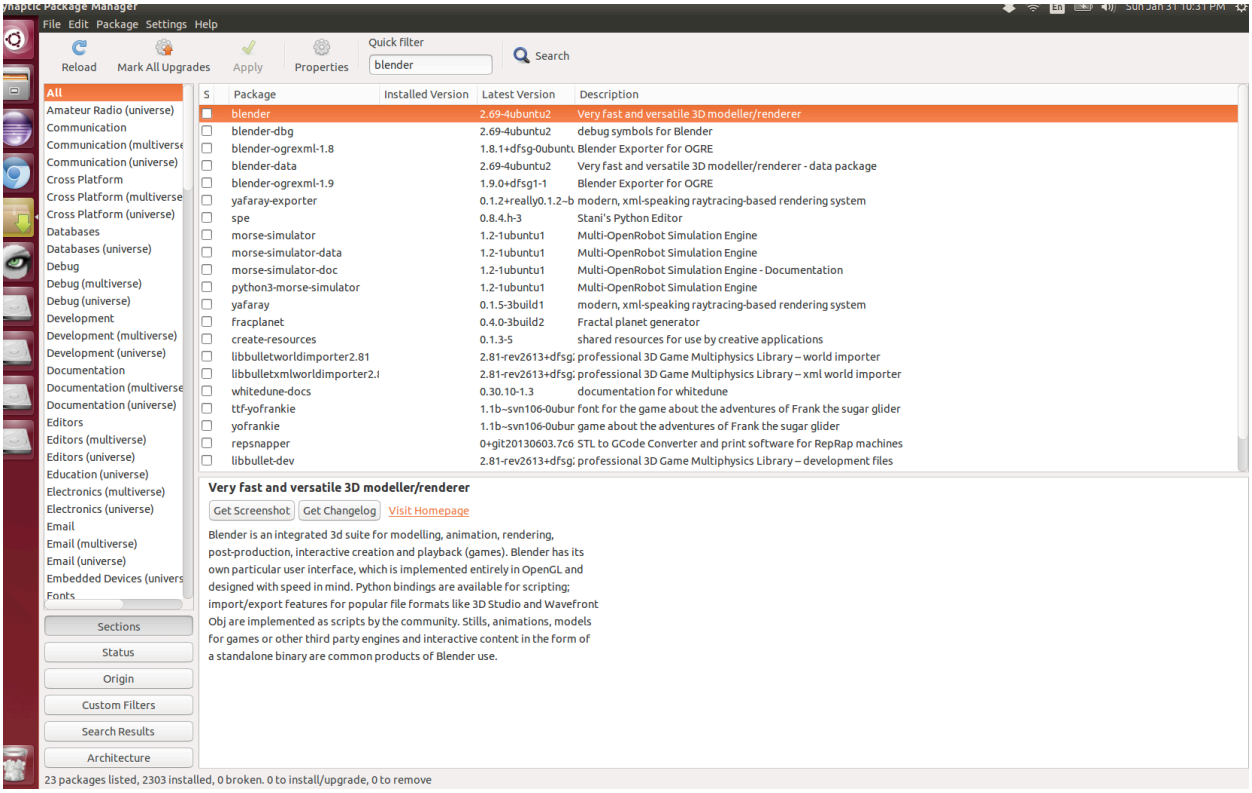
1. Goto https://www.blender.org/download/
2. Download the msi for windows and install it.

Linux:

1. Press “**Windows key**” and search for synaptic package manager.



2. Search for “**Blender**” in the search bar.



3. Click on the checkbox and select ”**mark for installation**”.

4. Click on “**Apply**” button for the installation to start.

Blender Usage

How to open a mesh file:

File -> Import -> select <fileformat> -> “browse the file”

Delete a mesh:

1. Right click on a mesh to select it.
2. Press delete button on keyboard to delete.

Input for parser

For each line in the convex hull, write the corresponding two points to output file name **log.txt**.

Each point should be in a separate line.

In each line x y and z of the point should be separated by spaces.

Example :

If the convex hull had three lines, then write the output in the following format

L1Ax L1Ay L1Az

L1Bx L1By L1Bz

L2Ax L2Ay L2Az

L2Bx L2By L2Bz

L3Ax L3Ay L3Az

L3Bx L3By L3Bz

Where L1Ax mean line 1, first point x – coordinate.

L1Bz mean line1 second point z – coordinate.

I have written a simple c++ program to convert the above **log.txt** file to **output.obj** file. Find the attached parser.cpp file.

Use blender to visualize the obj file.

Middle mouse -> rotate the view

Middle mouse + hold SHIFT -> pan the view

Middle mouse + hold CTRL -> zoom the view.

Load your actual mesh data and then the convex hull (output.obj) to visualize the convex hull on top of the mesh file.

```cpp
#include <iostream>
#include <fstream>

int main()
{
    std::ifstream infile("log.txt");
    std::ofstream outfile;
    outfile.open ("output.obj");

    std::string line;
    unsigned int noLines = 0;

    while (std::getline(infile, line))
    {
        outfile << "v " << line << std::endl;
        noLines++;
    }

    for ( unsigned int i=1; i<=noLines; i=i+2 )
    {
        outfile << "l " << i  << " " << i+1 << std::endl;
    }

    outfile.close();
    infile.close();
    return 0;
}
```