Body Scanning

Thor Underbjerg
University of Applied Science Upper Austria
Hagenberg, Austia
S1200629023@students.fh-hagenberg.at

Edwin Salinas Licea
University of Applied Science Upper Austria
Hagenberg, Austria
S1210629017@students.fh-hagenberg.at

ABSTRACT
The Kinect is a powerful camera with many different attributes that can facilitate its implementation in several applications. This paper will focus on an application that allows scanning a person and generating a 3D mesh ready to be opened in most of the standard 3D modeling software.

INTRODUCTION
The main goal of the project is to create a 3D model, using the Kinect camera, of a person in a standard format that is ready to be used in other softwares.

The basic steps to accomplish this will be, first to get the information from the Kinect, then process the data to create all the information needed for a mesh, and finally save this in a standardized format that can be taken alone to any computer and used as a normal mesh. Still each of these steps will require further researching to make this work properly together.

First the Kinect can give different kind of information so it is needed to find the right one for the goal and to get it as precise as possible avoiding the usual noise in this kind of input.

The second step will create the missing data that is needed to create the mesh, but it will depend on which format is chosen as the final result.

The third step basically consist in putting all the data into a file with a specific format known by other softwares so they know how to interpret what it is stored inside.

The way to test the result will be to open the final file in some commercial modelling software.

PROCESS
During the design of this new software the general ideas from the beginning needed to transform into more concrete solutions, so the technologies were decided, the steps were expanded, other files were created to be used within the system and at the end all this worked together to try to achieve and approximation of the original idea.

Finally the whole process was divided in four steps, each one create at least one file that is passed to the next step that will use the data into its calculations.

Passing the data through files makes it easier to work with different libraries and languages, because one program can create one file that will later be opened in other program maybe from a different environment, though this files need to be standardized so each program knows how to save and load the corresponding files.

Scanning
This is step consist on the Kinect scanning the user, the way this is done is thanks to the special features about the Kinect camera. This device has some data streams, firs it's used the Depth View that allows getting an approximation of the depth in millimetres from every pixel in the screen.

This will work along other data stream, skeleton tracking; this will set another data for every pixel, showing if it belong to a user or not so now will be possible to get only the depth from the user's pixels.

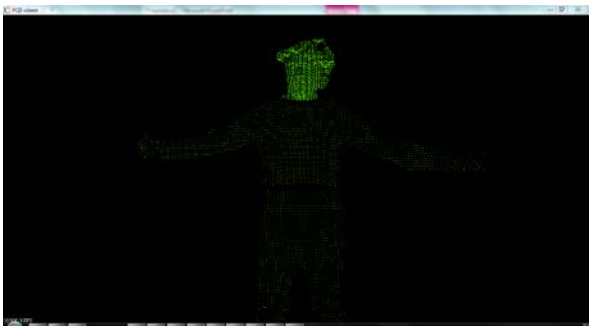
At this point the data will consist in all the points that form the user front body, but the measures are based in millimetres for the depth and in pixels for the X and Y. The ideal is to have everything with the same scale, being the best option millimetres for this case and one way to calculate this is using the tangent

$$\tan A = \frac{\text{opposite}}{\text{adjacent}}$$

The angle *A* will represent either half of the horizontal view angle or half of the vertical view angle, the *adjacent* is the depth given, and the *opposite* will either be the position X or Y.

After this it is possible to store X, Y and Z based on millimetres for each one of the points that shape the body.

Point cloud
With the data created from the scanning it's possible to generate a point cloud, this is a 3D structure made only by points located somewhere in a three-dimensional space.



This file will allow using a specific library to manipulate the point cloud and generate other data that would be too complex to get by hand but it is necessary to create a standard mesh.

Connecting the points

Up to this point it is possible to visualize the body thanks to the point cloud, but the idea is to create a triangle mesh connecting the points previously shown.

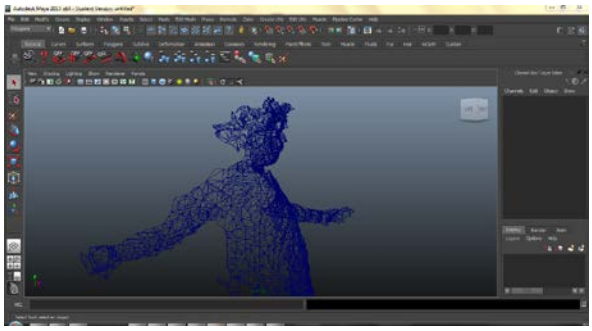
The real way to do this is getting an index that define which points are connected to create each one of the triangles, this based on the id given to each point according to its position.

The algorithm used to connect the right points uses the library PCL for C++ and is described in the paper "On Fast Surface Reconstruction Methods for Large and Noisy Datasets" by Zoltan Csaba, Radu Bogdan and Michael Beetz.

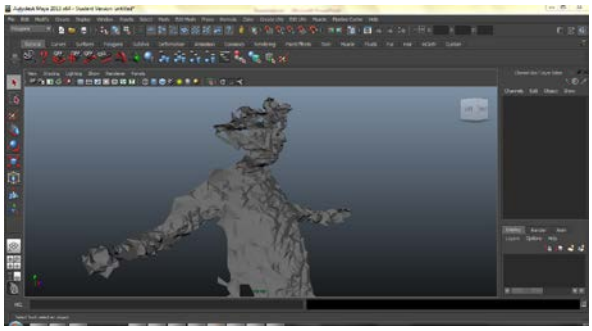
Create Mesh

The last step creates a Collada file; this one is able to be opened in most of the standard modeling software. This file has many sections that define different properties for the model, first is the one that defines how many numbers are needed to store all the vertexes (X, Y and Z per vertex) followed by the list of vertexes' position, then specify the amount of vertexes in the whole mesh, and the last part is the quantity of triangles in the mesh and the index that show how to shape those triangles.

All this work is done easily thanks to the library made by Alexandre Mute that focus on the use of Collada file, facilitating loading a template file with default content so it is only needed to change the sections mentioned before.



This file can be opened to see how the final result was and now it's possible to modify it using the tools from the software used to open it. At least a texture can be added to make it easier to see the quality lever.

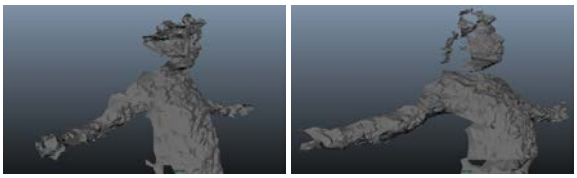


RESULTS

At the end, after the systems does all its work, the main result would be the Collada file, this contain a mesh with the frontal view of the user's body, but this scanning can be all at once or scanning different parts of the body independently; each one has its own advantages and disadvantages.

Scanning the full body (left) will create one single mesh where the different parts of the body will be smoothly united along to each other, but in the other hand, the quality of these sections is not going to be that good, mainly with the face features.

Dividing the scanning process (right) will improve the quality of each sections, but this add another problem, because each one of these meshes need to be put together so this need some calculations to move all the meshes to the right place.



Issues

There are two main problems with the models so far, first will be the quality of it, it was already said that dividing the scanning will improve the quality but anyway the mesh is not as smooth as it should; one partial solution will be to get a newer camera with improved hardware, beside it is possible to research for a smoothing algorithm that could improve the way the model looks.

The other problem is about some holes in the mesh, this is caused because while is connecting the vertexes it doesn't find points close enough to connect in that area. The solution for this could be to modify the parameter used to generate the mesh so it allow to connect farther points, or scan the body from different angles avoiding that some spot can be hidden because of the perspective.

Future

The immediate next step would be to refactor the project so it becomes easier to apply future changes, this will required to create some other classes to have a better control and organization of the whole system.

After the project is ready to seriously continue improving the system, then the next step will be to create a full body mesh (front and back), all this while trying to improve the quality of the mesh.

Once this function works properly then it will be good to continue with other features, like scanning color along to shape or adding bones to the mesh so it can articulate in a more realistic way.

REFERENCES

http://code4k.blogspot.co.at/2010/08/import-and-export-3d-collada-fileswith.html

http://pointclouds.org/

http://pointclouds.org/documentation/tutorials/greedy_projection.php

http://www.microsoft.com/en-us/kinectforwindows/

MILES, Rob. Start Here! Learn the Kinect API

WEBB, Jarret; ASHLEY, James. Beginning Kinect Programming with the Microsoft Kinect SDK