Implementation of Computer Graphics Algorithm Platform Based on Autodesk Maya

Qi Zhang[1,a] Zhanpeng Huang[2,b]

[1]College of Medical Information Engineering, Guangdong Pharmaceutical University, Guangzhou 510006, China

[2]College of Medical Information Engineering, Guangdong Pharmaceutical University, Guangzhou 510006, China

[a]27427024@qq.com, [b]8126644@qq.com

Guangdong Province Science and Technology Project(2014A020212684)

Keywords: Maya, low-pass filter, node

Abstract. Building a visualization algorithm platform is the first step and very important part of computer graphics research. In this paper, we describe a new method to implement computer graphics algorithm based on Autodesk Maya. By creating the node is to be integrated directly into the Dependency Graph, we can add our own project to the underlying Maya engine. In order to verify this approach, we exploit the C++ API to create a Maya plug-in node for the non-shrinkage low-pass filtering algorithm. And the output mesh and experimental results are both good. Therefore, we can engage computer graphics research under Maya platform.

Introduction

With the development of computer graphic, 3D technology is becoming more and more mature and its application is very extensive. Nowadays, one of the most popular 3D animation software is Autodesk Maya. It delivers an end-to-end creative work flow with comprehensive tools for animation, modeling, simulation, visual effects, rendering, match moving, and compositing on a highly extensible production platform [1]. As everyone knows, Maya is already used in the film, television, and gaming industry. And in fact, Maya can be used as a tool for science research. For example, Maya has good compatibility that it can combine with other software for doing virtual experiments. For example, 3D reconstruction of medical image based on Maya and Virtools, and Maya also can connect with Kinect to capture motion animation data. Especially, Maya can be a visualization platform for computer graphics research by taking advantage of Maya's powerful visual expression and data processing. However, most researchers are not familiar with this complex software. Hence they usually developed their own experiment platform for algorithm research rather than using Maya. In our case, we attempt to write C++ program to extend the basic Maya classes for obtaining all the information of 3D mesh, store them in a node to process the data calculation, and finally display the result visually in Maya.

Related work

Maya Architecture. The entire Maya system can be broken down into three major components shown in Figure 1. The Dependency Graph is like central database that save all the important data and information of entire 3D scene. And the Dependency Graph is a set of interconnected nodes. Each node has one or more attributes and a compute function which is assigned the task of taking one or more input attributes and producing one or more output attributes as shown figure 2. Therefore, when we add some new functionality to Maya, it just needs to create a new node [2].

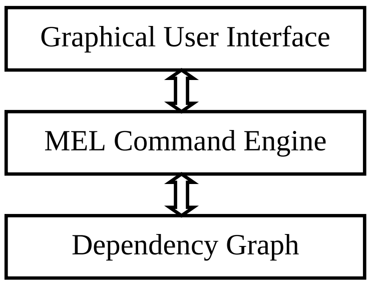

Fig. 1 Maya System

NODE
Input
Output
Compute()

Fig. 2 A general node

Maya Plug-in for Research. In these years, many researchers have some attempts for virtual research experiments. According to [3], they developed a Maya plug-in for automatic 3D scene generation by virtue of C++ API in Maya. They combine the advantages of both automatic 3D scene generation and excellent graphics tools in Maya, to achieve fast and efficient generation of 3D scene and good visual effects as well as convenience for the subsequent operations. Junfa Liu introduced [4] they developed a Maya plug-in to extract output the geometry information of 3D caricatures from Maya in terms of vertex coordinates. In [5], they implemented the deformation transfer algorithm in the form of a Maya plug-in as a convenient and practical solution. Solving the linear system involves factorizing a huge sparse matrix, Maya recomputed the vertex normals after deformation transfer.

A Computer Graphic Classic Problem---Mesh Fairing

The mainstream approach in mesh fairing is a signal processing proposed by Taubin [6], which extended Fourier analysis to signal defined on polyhedral surfaces of arbitrary topology. Because Gaussian filtering produces shrinkage, so they define a low-pass filter to anti-shrinkage. Therefore, this is a very classic mesh filtering algorithm that is why we choose it to implement. Before implementation, we need to extract some important arithmetic operators simply.

Discrete surface signal. In [6], they define the discrete Laplacian of a discrete signal by weighted averages over the neighborhoods

$$\Delta x_i = \sum_{j \in i^*} w_{ij}(x_j - x_i) \tag{1}$$

where function $x = (x_1, x_2, ..., x_n)^T$ defined on the vertices of a polyhedral surface, the weights w_{ij} are positive numbers that add up to one. Here we choose the simplest method to set w_{ij}, that is $w_{ij} = 1/|i^*|$, where $|i^*|$ is the number of neighbors.

Transfer function. Define W=(w_{ij}) is the matrix of weights, and the matrix K is equal to I-W. Then,

$$f(k) = (1 - \lambda k)(1 - \mu k) \tag{2}$$

where λ is scale factor, and μ is a new negative scale factor such that $\mu < -\lambda$.

Finally, have the following formula

$$x' = f(K)x = \sum_{i=1}^{n} \xi_i f(k_i) u_i \tag{3}$$

In order to define a low-pass filter that $f(k_i)^N \approx 1$ for low frequencies.

Experimental Approach

Build Experiment Environment. To set up the experiment environment, we must install Maya and Visual Studio first. Here we choose Maya 2013 and Visual Studio 2010 under windows 7. Then follow the Maya Plug-in Wizard to start, see the detailed steps in [7].

Write C++ program. To get the mesh's vertices information and process the equation calculation, we can direct call the Maya API function help to finish.

First of all, write the function computeVertexMatrix(MObject& meshObj,MatrixXd &matrix) to get all the vertices position, and save the coordinates in a matrix. Therefore, we need to include the Eigen library to define a matrix as follows.

```
 matrix.resize(nVertices, 3);
MItMeshVertex it(meshObj, &s);
for (int ithVert=0; !it.isDone(&s); it.next(),ithVert++)
{
    MPoint p = it.position(space, &s)*transMat;
    matrix(ithVert,0) = p.x;
    matrix(ithVert,1)= p.y;
    matrix(ithVert,2) = p.z;
}
```

Then, we had to get every vertex's adjacent vertices information based on low-pass algorithm. To compute the adjacent matrix, we need to define a sparse matrix with including Eigen library. Accordingly, we write the function computeAdjacencyMatrix(MObject& meshObj, SparseMatrix<double> &sm1).

```
 sm1.resize(nVertices,nVertices);
std::vector< Eigen::Triplet<double> > tripletList;
MItMeshVertex it(meshObj, &s);
for (int ithVertex=0; !it.isDone(&s); it.next(),ithVertex++){
    MIntArray adjID;
    it.getConnectedVertices(adjID);
    for (unsigned ithAdjVtx = 0; ithAdjVtx < adjID.length(); ++ithAdjVtx)
    {
      tripletList.push_back(Triplet<double>(ithVertex,adjID[ithAdjVtx],-1.0/adjID.length()));
    }
    tripletList.push_back(Triplet<double>(ithVertex,ithVertex,1));
}
sm1.setFromTriplets(tripletList.begin(), tripletList.end());
```

Besides, the value of λ and μ will be defined by user. And finally, we need to register this plug-in node in the file pluginMain.cpp. The main code as follows.

```
status = plugin.registerNode( "lowpass", smoothAlgorithm::id,
                                    smoothAlgorithm::creator,smoothAlgorithm::initialize );
```

Experimental Results

Without compile error, it will generate a *.mll file which will be loaded into Maya 2013. After import a mesh, it needs to join the node connection first as figure 3 shown. Following we take a test to verify the low-pass filter algorithm. When λ=1.1 and μ=1.4, figure 4 shows the filtering result. Maya has direct displayed the mesh result in the 3D scene. Thus it can be seen that Maya can be used as visualization platform for computer graphics research.

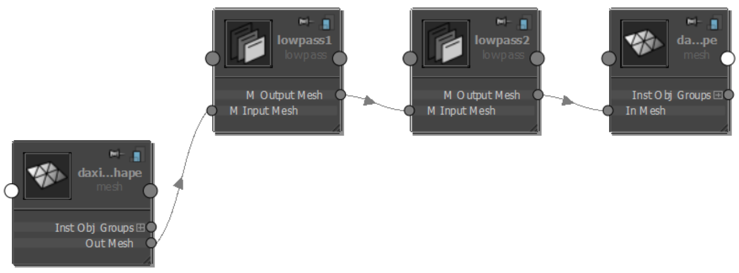

Fig. 3 Node connection in Maya 2013

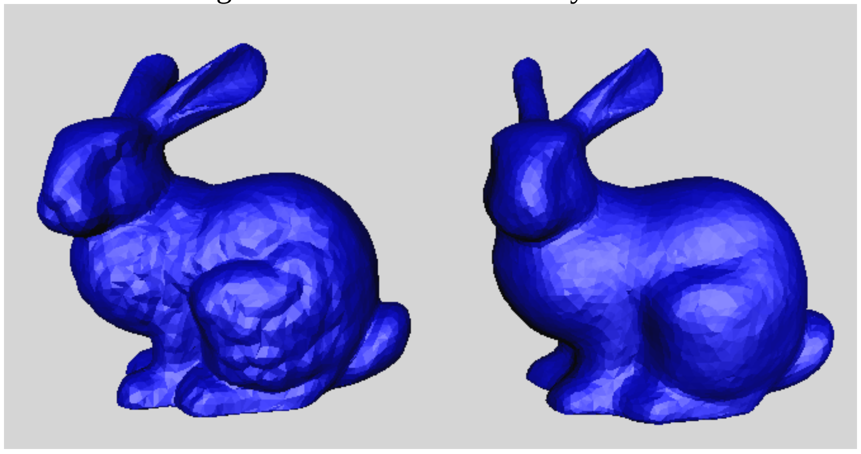

Fig. 4 λ=1.1 and μ=1.4 with one step

Conclusion

Maya is 3D animation software with extended functionality, especially the powerful Maya API. Expect for producing special effect, we can also process computer graphics algorithms data calculation with the help of Maya API. It is very convenience for researchers to implement algorithms visualization under the Maya platform. In this paper, we make an attempt for implementing computer graphic classic algorithm, both the output mesh and calculation speed are good. However, Maya will hard to complete calculation for hundreds of thousands of faces of a mesh because of huge amount of computation. Therefore, in the future work, we will consider that combine Matlab and Maya together, Matlab is responsible for computation and Maya is for visualization. Building a more efficient computer graphics algorithm platform, it is our further research goal.

References

[1] Virtual Reality and Environments, Cecilia Sik Lanyi, Chapter 2 A survey of Some virtual reality tools and resources.

[2] David A.D.Gould. “Complete Maya Programming: An Extensive Guide to MEL and C++ API”, Morgan Kaufmann, 2003.

[3]Li C, Yin C, Lu J, et al. Automatic 3D scene generation based on Maya[C]//Computer-Aided Industrial Design & Conceptual Design, 2009. CAID & CD 2009. IEEE 10th International Conference on. IEEE, 2009, pp.981-985.

[4] Junfa Liu, Wenjing He, Tao Chen, Yiqiang Chen, Manifold Constrained Transfer of Facial Geometric Knowledge for 3D Caricature Reconstruction. Journal of Computer Science & Technology(JCST), 28(3), pp. 479-489, 2013

[5] Pawaskar C, Ma W C, Carnegie K, et al. Expression transfer: A system to build 3D blend shapes for facial animation[C]//Image and Vision Computing New Zealand (IVCNZ), 2013 28th

International Conference of. IEEE, 2013, pp.154-159.

[6] Gabriel Taubin, A Signal Processing Approach to Fair Surface Design, In SIGGRAPH 95 Conference Proceedings, August, 1995, pp. 351-358.

[7]Information on http://docs.autodesk.com/MAYAUL/2013/ENU/Maya-API-Documentation/index.html