Abstract

All the visible objects have shape and texture. The goal of computer graphics is to simulate the real world. To make it look realistic from geometry point of view, we have to make sure that the shape and texture of the object are accurate. Shape can be hand crafted using various 3D modeling tools like Autodesk Maya, 3ds Max, Blender to name a few or they can be acquired from real world using laser scanners, etc. The output of these modeling tools is in the form of mesh models, which can be stored in different formats. Typically these mesh models consists of vertex position in 3D coordinates system and information regarding the relation between these vertices like which vertex is connected to which other vertex. Texture is the other half that must be ensured for the things to look real. We have to paste the texture on the surface, such that it correctly overlaps with the geometry of the mesh. The process of pasting the texture on the surface of the mesh model is called Texture mapping. Texture mapping can be done in two ways. First way is to texture the surface by synthesizing the texture directly on the surface. The second method is to wrap the texture around the surface and cut the texture so that it will fit nicely on the surface. To visualize this problem we have to think of the texture as a cloth. The first method can be thought of weaving around the body to fit exactly, while the second method is like stiching. In this thesis a new method is proposed for the second approach and the goal is to map texture at interactive rates.

In Computer graphics, mesh models of arbitrary shape are parameterized into planar 2D grids for texture mapping. To describe parameterization, it is the process of mapping points of particular dimension into an another and simultaneously maintaining some constraints. When parameterizing these mesh models, we try to keep the geometric correspondence between the mesh vertices intact to reduce the distortion of the texture. Typically, parameterizing a mesh model involves solving a set of linear equations representing the geometric correspondence of the triangles and normally these methods produce good results. However, these energy minimization procedures are normally computationally expensive and cannot be applied for real time application or large mesh models.

To make parameterization a real time procedure for large meshes, we propose a greedy approach that operates using local optimization function rather than the traditional global optimization function. We express the surface in terms of local curvature and store these values in a priority queue. Our algorithm is simple to implement and the algorithm can texture at approx one million polygons per second on typical desktop setting. We prove that our algorithm is robust to dynamic mesh and produces the same texture, when a part of the mesh is removed and brought back again. This is an important feature because in case of optimized rendering of objects, you might require to remove some objects from rendering and render them back again when these objects fall back into the view frustrum. In such situations, if you are generating the texture coordination at run time, then this feature would be very useful. This algorithm is not affected by the sudden change in the density of points on the mesh surface or noisy surface, i.e the algorithm's runtime complexity does not depend upon the complexity of the surface. Whereas in the case of tradional algorithms, the run time complexity depends on the complexity of the mesh surface. The algorithm needs to run only for one iteration and is not affected by the size of the texture.

To go with the texture mapping algorithm, we propose a method on how to make a texture self tileable and store only the texel structure, if the texture is repetitive. We present qualitative and quantitative results in comparison with several other texture mapping algorithms. We believe that our algorithm has robust output quality and finds its application in different fields of science. Because of the interactive speeds at which our algorithm can perform, we show case some situations were we can use them like moving the texture at particular point in required direction, moving the texture seam to a required position and we can also use our method for generating runtime texture for MMORPG games. For example, in some applications you require interactive texture mapping rates like rapid prototyping, in areas where you have dynamic mesh topology and for texture mapping heritage mesh models reconstructed using multiview geometry, as these mesh models are very dense and noisy.