TEXTURES IN SIMULATION

Miguel A. de Riera Pasenau [1]

[1] *International Centre of Numerical Methods in Engineering*
Mod. C1 Campus Norte UPC, c/ Gran Capitán s/n, 08034 Barcelona, Spain
http://gid.cimne.upc.es gid@cimne.upc.es

Summary: Textures, a well known and widespread used technique in games, can also be useful in simulation of manufacturing processes, not only as a tool to improve the appearance of the results, but also to validate the simulation code and reduce manufacturing costs. Textures, basically, are images glued to polygons. These textured images can be used to make the simulation's results understandable at a glance, to improve quality and manufacturing efficiency, for instance in the can industry, and to contrast the simulation with experimental results.

Keywords: textures, manufacturing, textured images, experimental results comparison, image mapping, image projection, postprocess, realism.

Introduction

Texturing is a technique for efficiently modelling the surface's properties. Textured images, or simply *textures*, provide the most realism in a model and can be used efficiently to hide the model's polygonal aspects. This mechanism plays a key role in games for years and is one of the first features implemented in real-time rendering systems. From the several texturing methods: colour image texturing, which glues simple colour images onto surfaces, alpha blending in texturing, to specify holes and transparencies over the surface, reflections via environment mapping, to generate approximations of reflections in curved surfaces, and rough surface simulation using bump mapping, which makes a surface appear uneven in some manner (bumpy, wrinkle, wavy, etc.), among others, this paper will be focused in the simple colour image texturing.

Textures 1: a way to simplify the model and provide more realism

So far, in post-process, every polygon has been drawn as either a solid colour, smoothly shaded between the colours at its vertices or using colour scale when doing contour fill. In a human behaviour simulation[1], where each person is simulated as a simple point, when viewing the 2D simulation, only moving points are seen. This results representation makes it difficult for a non-initiate user to distinguish if the simulation handles with persons or sand grains through a funnel, a bee swarm or oranges storage.

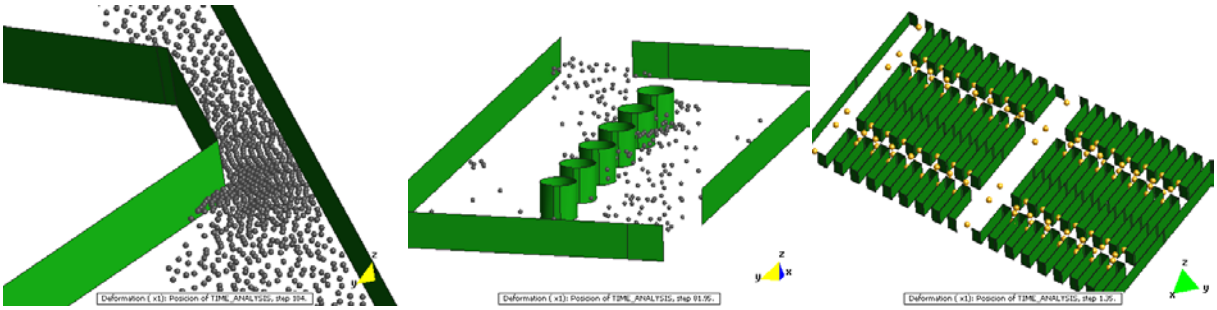

Fig. 1: 2D point simulation: from left to right, persons flow through a narrow pass, persons path while visiting a exposition and evacuation simulation of a airplane (courtesy of Rainald Löhner).

One way to improve the representation will be to drawn a person object for each point of the simulation with lots of polygons: a coarse representation with a sphere as head, and five cylinders as body, legs and arms, needs 20 quadrilaterals, 4 for each cylinder, and 18 triangles, 2 for each cylinder and 8 for the sphere, that is, a total of 58 triangles. In addition to this, if the analysis simulates thousands of points/persons, this will mean to handle hundred thousands of triangles, and even then all the persons will look the same and frozen when the analysis, animation, extends along time.
Another way will be to draw a quadrilateral for each points and glue an image of a person (perhaps a real one). But not only an image, but a sequence of images so that the appearance of each person changes when the animation of the simulation advances.

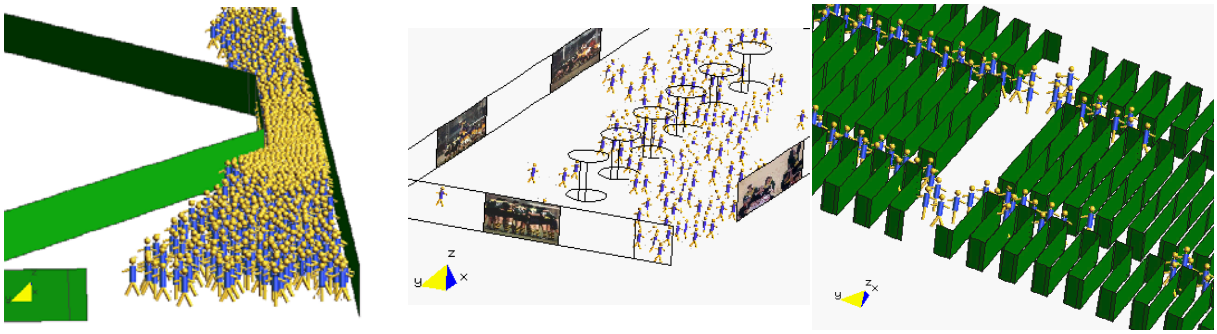

Fig. 2: 2D point simulation with images attached, the image is a snapshot of a person represented as a sphere and five cylinders: from left to right, persons flow through a narrow pass, persons path while visiting a exposition and evacuation simulation of a plane (courtesy of Rainald Löhner).

Understanding texture mapping

Textures are simply rectangular arrays of data, colour data in our case, but it can also be luminance data for light mapping, transparency values for alpha mapping, normal values for bump mapping, among others. Over this image-space, where coordinates $(a, b) \in ([0, image\_width], [0, image\_height])$ in pixels, a parameter-space is defined, called *texture-space*, where coordinates $(s, t) \in [0, 1]^2$. These texture-space values are used to find what the colour of the image is at this location.
To glue an image to a polygon, a *projector* function is needed so that for each point (x, y, z) of the polygon texture coordinates (s, t) are obtained. This process is called *mapping*. To draw the image region over a triangle using the facilities of the rendering system, only the texture coordinate (s, t) over each vertex has to be assigned. The texture coordinates at the vertices are interpolated by the rendering system to obtain texture coordinates at other pixels

in the triangle. Each interpolated coordinate is also used to do a colour lookup in the image. The texture coordinates *(s, t)* together with colour image values is called a *texture element*, or *texel* for short.
What makes texture mapping tricky is that a rectangular texture can be mapped to nonrectangular regions, and this must be done in a reasonable way. [2, 3, 4]

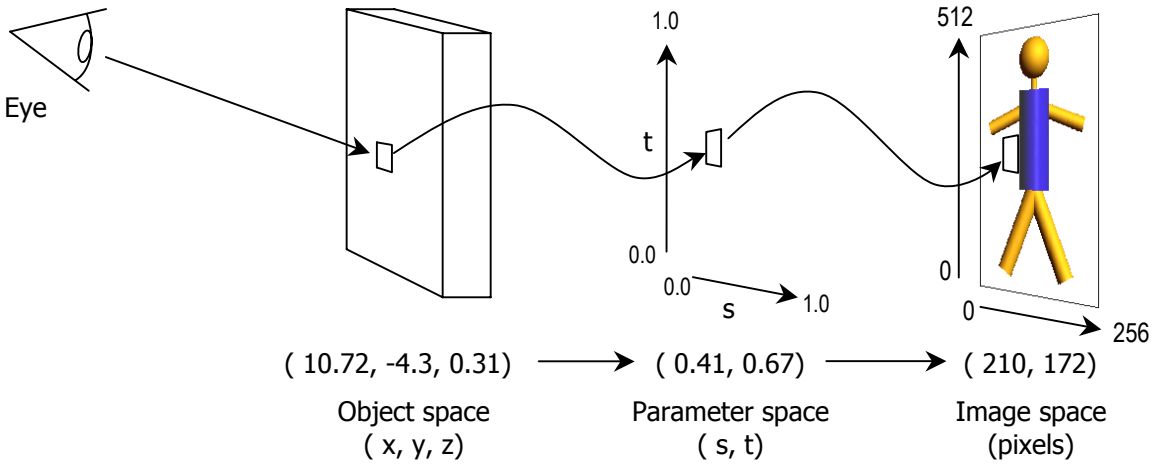

Fig. 3: texture mapping process: a world point is mapped to the parameter space to get the texture coordinates *(s, t)*, and with these coordinates the colour pixel is located on the textured image.

One of the advantages of this process is that, once the texture coordinates *(s, t)* are *mapped* to the vertices of a polygon, the same image region is rendered. If this polygon deforms or distorts in some way, the image region defined by its texture coordinates *(s, t)* will be deformed, distorted, in the same way.

Textures 2: improving quality and manufacturing efficiency

Using this advantage, another field of application of texture mapping is opened: the pre-distorted printing. An image applied to a surface involved in a, for instance, stamping analysis process will get the form of these surface at the end of the simulation. And the other way works too. An image applied to an already stamped surface will be deformed when this surface get its original form back.

Fig. 4: Can stamping simulation made by Quanteck ATZ for Metalpack SA: after deforming the sheet mesh to the final can state, the decorative labels are applied to the mesh and then its original flat shape is restored to get the deformed images, ready for the press (courtesy of Quantech ATZ, Metalpack SA).

This application of textures is being used by the Stampgen program, at Quanteck ATZ[5, 6]. The tool-making module employed for can-making has been enhanced by simulating the distortion of flat images printed on sheet into the finished product. So time can be saved by just applying the decorated label on the finished can at the end of the simulation process and then obtain the initial decoration which will be printed to the flat sheet.
Two limitations have to be overcome to match the quality and needs that this printing process needs. These limitations are inherent to the texture mapping process and the rendering system: one is the size constraints on textured images that the rendering system imposes, and the other is the limited texture coordinate interpolation scheme used by the rendering system, which is shown when using a too coarse mesh for the definition of the surface.

Texture limitations

One texture limitation is that the size of images used for texture mapping must be power of 2 and cannot exceed the size limit imposed by the rendering system. On some systems this limitation is as small as 256x256 pixels, but generally a texture image can be as big as 1024x1024 or 2048x2048. This is not enough for the press, where they demand images with a resolution of, at least, 1200dpi. For example, a can with a diameter of 70mm and a height of 30mm the image size of the decorative label has 10390x1417 pixels.
To map these huge images to the finished can and get its original flat sheet printing, these images have to be subdivided into several textures and map each of them over the corresponding surface region. Following the previous example, the 10390x1417 image must be subdivided into 41x6 = 246 textures of 256x256 pixels, or 11x2 = 22 textures of 1024x1024 pixels.

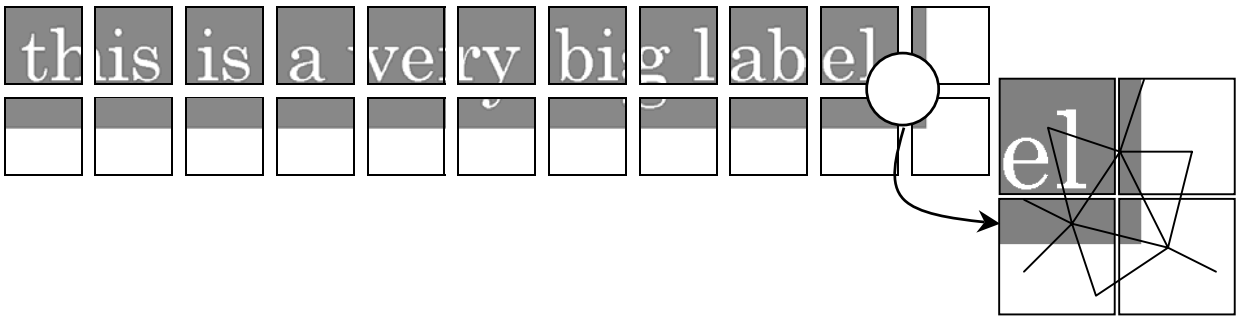

Fig. 5: example of a label subdivision, the grey shaded label is subdivided into 11x2 textures of 1024x1024 pixels. The zoom shows a triangle which shares 4 textures at once.

With this subdivision another difficulty appears, some triangles of the surface mesh will lie across two, three or even four textures and have to be divided accordingly, so that each piece of each triangle is rendered with its corresponding texture.
It is also worth to note that the context exchange when loading different textures in a rendering system is a big penalty which has to be minimized. When drawing the mesh, instead of traversing the elements and for each one of them load its texture and draw it, it is better to load the texture and then traverse the triangles and render the ones that uses this texture.
Another limitation is that there are cases where the interpolation of the texture coordinates along the triangles does not match the desired result. If a texture is applied to a warped quadrilateral, neither triangulation will do the correct interpolation as if the texture where

applied directly to the quadrilateral. What is needed is either another interpolation scheme, or a previously deformed image or a fine meshing. [2, 3, 4]

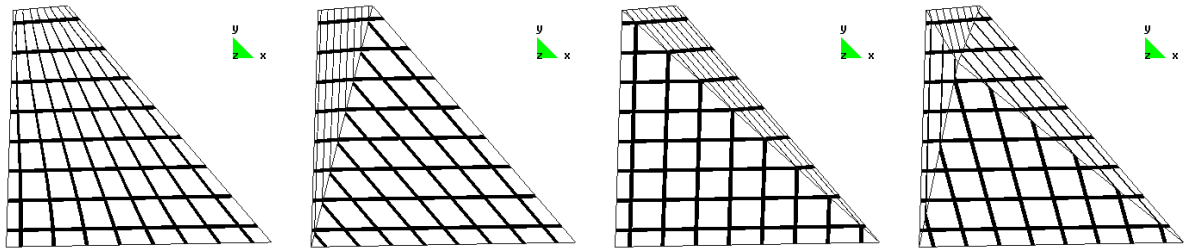

Fig. 6: applying a grid image to a quadrilateral surface: at the left, the desired result, on the right three images showing the interpolation problem with three different triangulations.

Textures 3: comparison between real and virtual results

In addition to the decorative use of textures, and to the industrial applications, using textures in simulation can also be used to contrast experimental results and the ones from the analysis. This can be accomplished by applying a precise pattern to a physical sheet in laboratory and applying the same pattern, as a textured image, in the computer sheet. After running the analysis and carry out the experiment in laboratory, the final deformed sheet can be compared measuring the pattern deviation from the original state and the deformed one, and between real and virtual analysis.

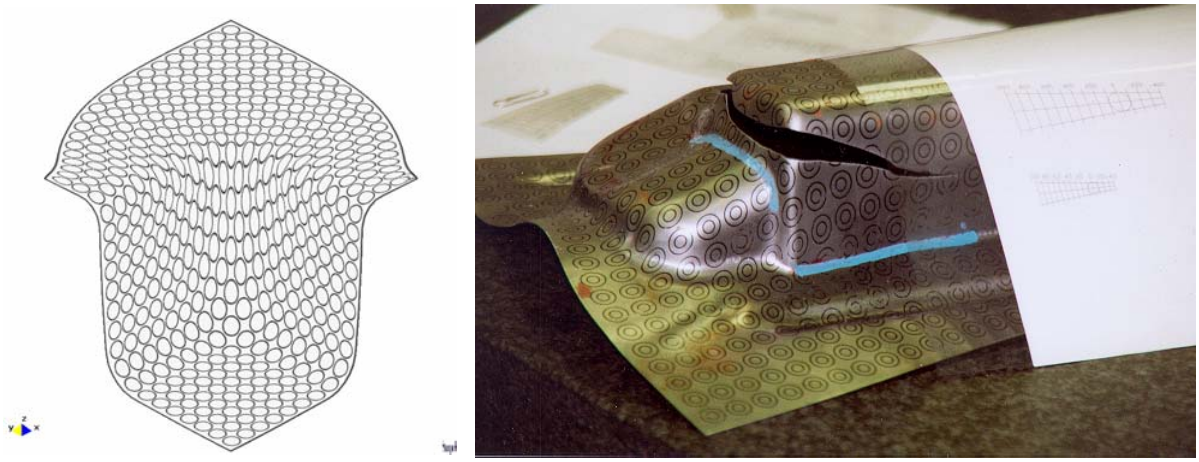

Fig. 5: Circles image applied to a quarter butterbox in simulation (courtesy of Quanteck ATZ) and using circles with the Mylar rule at the laboratory (courtesy of ASCAMM).

References

1. Rainald Löhner, Jul 2000.
2. Mason Woo, Jackie Neider and Tom Davis, *OpenGL Programming Guide, Second Edition*, Addison-Wesley Developers Press, 1997.
3. Tomas Möller and Eric Haines, *Real-Time Rendering*, A.K. Peters, Ltd., 1999.
4. David H. Eberly, *3D Game Engine Design*, Morgan Kaufmann Publishers, 2001.
5. "Distortion Made Clear", *The Canmaker*, May 2001, p. 35.
6. Quanteck ATZ, http://www.quantech.es