GPU Texture Level of Abstraction in 3D Scenes

Jordane Suarez[*], Farès Belhadj[*] and Vincent Boyer[*]

[*]L.I.A.S.D. Université Paris 8, France
{suarez, amsi, boyer}@ai.univ-paris8.fr

Abstract
We present a method to dynamically control the texture level of abstraction in 3D scene. Level of abstraction consists in visualizing the necessary and sufficient information in an image. Texture generation is generally realized by a designer in a high resolutions with a low level of abstraction. Our model provides automatically texture level of abstraction through offline and online segmentation and lets the designer define the number of colors in the object texture.

1. Introduction

In 3D level of abstraction techniques [KST08, GTDS10] the texture generation is neglected and has never been considered as a part of the abstraction solutions. For example view-dependent toon shading [BTM06] provides level of abstraction as mentioned by the authors, but unfortunately only focuses on the description of a particular illumination model and never considers the 3D model textures. It is tricky to suppose that the texture generation belongs to the designer's job because it supposes that the designer creates all the necessary textures for all available levels of abstraction. In contrast with 3D level of detail techniques where both view dependent geometry and texture LOD rendering are simultaneously considered, the 3D level of abstraction techniques focus only on an illumination model or geometric data of the 3D models and never consider the texture abstraction. We addresse the following question : how must be textured, at different level of abstraction, a 3D polygonal mesh defined with only its textures given at a high level of abstraction? Basic solutions that create abstracted textures through 2D segmentation process are not suitable: how many textures should be precomputed? how are considered negative coordinates and repetitions?

2. Our Model

Starting with a 3D textured model, we create a texture-object model space that takes into account how the texture is mapped on the geometry (see section 2.1). Then the segmentation can be achieved with 2D image segmentation methods (see section 2.2). As we are able to generate textures with different levels of abstraction, different strategies can be used to map texture level of abstraction on the 3D model (see section 2.3).

2.1. Histogram generation

We propose to compute a histogram that takes into account the usage proportion of each texture in the object space. This computation is necessary once (results can be stored and referenced by the model) and should not be recomputed during the rendering process. Thus, for a given mesh and for each triangle T_i that composes the mesh, we compute in a first step: S_i^o, the surface of T_i in the object space; S_i^t, the surface of T_i projected in the texture space according to its texture coordinates $Coord_i$. During this step, we extract the extremal texture coordinates in order to manage texture repetitions and negative coordinates. In a second step, we compute P_i^o (resp. P_i^t) by dividing each S_i^o (resp. S_i^t) by the sum of all object space surfaces S^o (resp. by the sum of all texture space surfaces S^t). Therefore $P_i^o \times P_i^t$ corresponds to the triangle area in the object space weighted by its area in the texture space. Finally we use these proportions to compute color occurrences in a new texture. This is done in the GPU and the results are stored in the fragment's alpha component. Additive blending is enabled and, during this process, for each texel and for each triangle that covers it, the alpha channel accumulates the quantity $P_i^o \times P_i^t$. In order to perform color quantities, we scanline (in CPU) the generated texture pixels to fill a table where colors are unified and quantities are merged. As our histogram generation is computed in the texture-object model, texture repetition and texture deformation are treated by our GPU process.

DOI: 10.2312/conf/EG2012/posters/029-030

2.2. Dynamic Segmentation

We propose to use the hierachical clustering algortihm [HTF09] to produce dynamic segmentations (we have also implemented a *k*-means segmentation, but results show that hierarchical clustering gives better solution while avoiding creation of new colors): we search among the set of *n* colors, each pair of spatially closest ones. Each pair is merged into a new color. All configurations are saved and the process is reiterated until convergence (only when one single color remains). The entire process is realized offline and we store the colors computed at each iteration in a line of a 2D texture. This texture describes a colors tree where we can search for the closest color at any level or find it using an incremental algorithm. Finally, we render the model: a GPU shader is used to colorize each fragment using the texture previously generated according to the texture coordinates and the initial texture. An uniform value *l*, expressing the desired level of abstraction, is given as an input of our shader. This value is used as the *t* coordinate of the generated texture. Therefore texturing each fragment of the mesh consists in finding, for this fragment and in the generated texture, the closest color to the one used in the initial texture: for a given line (*t*) in the generated texture (containing the set of segmented textures colors), the closest color (*s*) to the initial one.

2.3. Adapting the texture level of abstraction

Our model is able to produce, in real-time, different levels of abstraction for a 3D scene. Every function (or function composition) which is able to produce values in a given range can be used as an input for *l*. These can be classified into two main categories: static and dynamical-based strategies.

- Static-based strategy: to let the designer choose the level of abstraction; depending on the significance of each object in the scene, the user can affect a fixed value to the object.
- Dynamic-based strategy: depending on the depth/orientation of each object, the level of abstraction is automatically adapted as in [BTM06]; perception based: the level of abstraction is automatically adapted at each frame depending on the objects meaning.

Different strategies can be simultaneously combined on the scene, on a part of it (a model), or on multiple textures used for a given model.

3. Results

The figure 1 shows results obtained applying our model on a complete 3D scene and gives an overview of the good behavior of our model. Here different shadings (toon, edge drawing, color enhancement, etc.) follow the texture abstraction; respectively 13, 16 and 12 colors are chosen for the segmentation of the terrain, the water and the sky.

The figure 2 illustrates the perception-based strategy on a crowd scene. We consider that characters with a gun are dangerous. Using this information, we adapt the texture level of abstraction to render other characters. Note that this scene is composed by 57 textures, 36 Vertex Buffer Objects, 200000 polygons and we obtain 200 frames per second on a NVIDIA Quadro FX while with a classical OpenGL rendering, we obtain 300 frames per second on the same architecture.

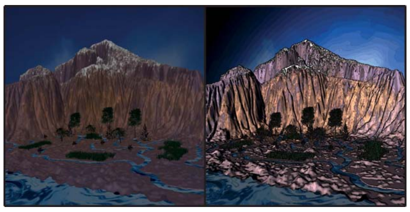
Figure 1: *Texture-object model space segmentation on a complete scene.*

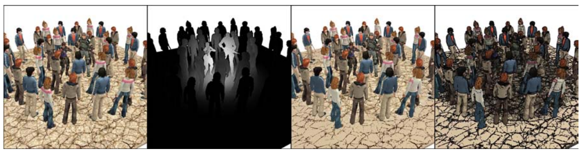
Figure 2: *Texture level of abstraction using dynamic perception based strategy*

4. Conclusion and Future Work

We have presented a method that automatically generates and dynamically uses texture levels of abstraction in a 3D scene. As future work, first we plan to apply this model to the visualization of crowd simulation (agent behavior, emergence of group dynamics, etc.) to reduce or emphasize scene details. Applications on NPR effects (Toon Shading, Abstraction) and illustration applications will also be investigated.

References

[BTM06] Barla P., Thollot J., Markosian L.: X-toon: an extended toon shader. In *Proceedings of the 4th international symposium on Non-photorealistic animation and rendering* (New York, NY, USA, 2006), NPAR '06, ACM, pp. 127–132. 1, 2

[GTDS10] Grabli S., Turquin E., Durand F., Sillion F. X.: Programmable rendering of line drawing from 3d scenes. *ACM Trans. Graph. 29* (April 2010), 18:1–18:20. 1

[HTF09] Hastie T., Tibshirani R., Friedman J.: 14.3.12 hierarchical clustering. In *The Elements of Statistical Learning (2nd ed.)* (2009), Springer N. Y., (Ed.), vol. 1, New York: Springer, pp. 520–528. 2

[KST08] Kolomenkin M., Shimshoni I., Tal A.: Demarcating curves for shape illustration. In *ACM SIGGRAPH Asia 2008 papers* (New York, NY, USA, 2008), SIGGRAPH Asia '08, ACM, pp. 157:1–157:9. 1