Technical Sketch: Rendering “Totó Sapore”

Eoin Murphy*, Oscar Tornincasa**
Chinatown

Fig. 1: Art Directors rendition of the rooftops of Naples

1. Introduction

Toto’ Sapore is a feature length animated cartoon directed by Maurizio Forestieri featuring a mix of traditional animation and 3D. It tells the story of Toto’ a poor boy who dreams of becoming a great chef, and in the course of the film saves the city by inventing the Pizza. It is set in 18th Century Bourbon Naples.

It will reach Cinema screens in Italy at Christmas 2003, and in other countries a little afterwards. Preliminary work started on Toto’ in 2001, and 3D preproduction began in Autumn of 2002. All 3D elements for Toto’ Sapore were modelled, animated and rendered in Softimage XSI.

Toto’ Sapore was modelled and animated in Softimage XSI Version 3.0 and Rendered entirerly in Mental Ray V2.?

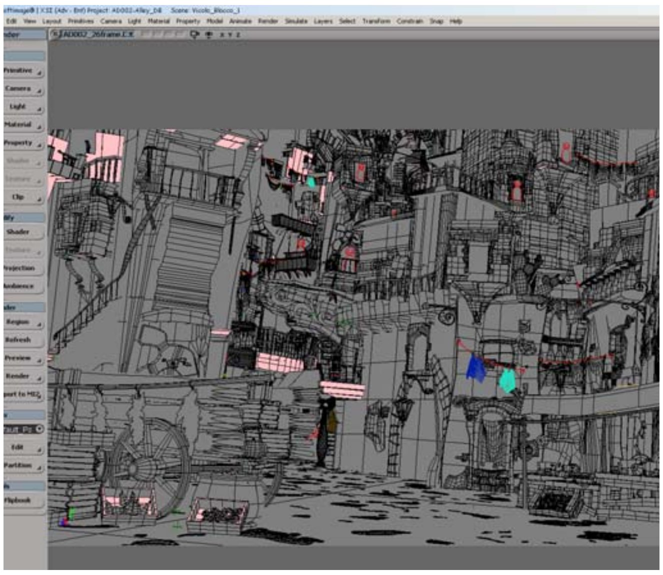
Fig. 2: Naples mid shot being worked on inside XSI

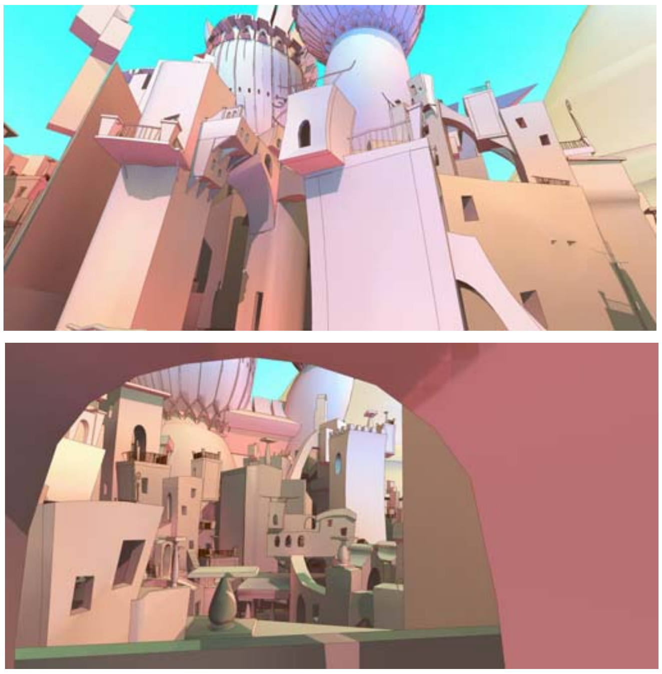
Fig. 3 & 4: Toon Shading tests for the castle & city flythrough

We faced two particular dilemmas in planning Toto’ Sapore:

1. Rendering one extremely complex & detailed sequence
2. Rendering large quantities of simpler scenes

The films opening sequence, known as AD002, was an ambitious wide shot of Naples seen from the bay, with a 40 second camera flythrough finishing as a closeup of a market. The city was entirely created in 3D, with 2d characters placed on grids (never moved at too acute an angle with respect to the the camera). We will be dealing primarily with this sequence in this document, as it was the films main rendering challenge

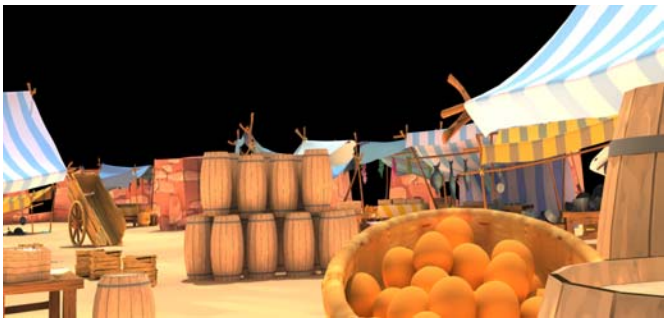
Fig. 5: tests for the look and feel of the Market Scene

2. Lighting & Materials Setup

With such a large exterior shot, the decision on how to light was critical - Using traditional lighting methods would require hundreds of lights,one in every nook & cranny, some casting shadows, others not. On the other hand, using Global Illumination techniques, which if correctly applied would make the scene “light itself” would quadruple rendering times, and present all the other problems of non-production tested techniques.. In the end, we opted to buy a 16 processor Renderfarm, and throw caution to the wind.

*e-mail: eoin@eoinfx.net
**e-mail: oscar@china3d.it

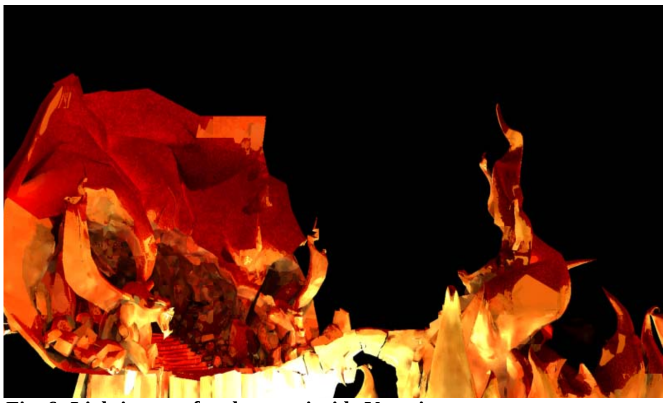
Fig. 6: Lighting test for the cave inside Vesuvia

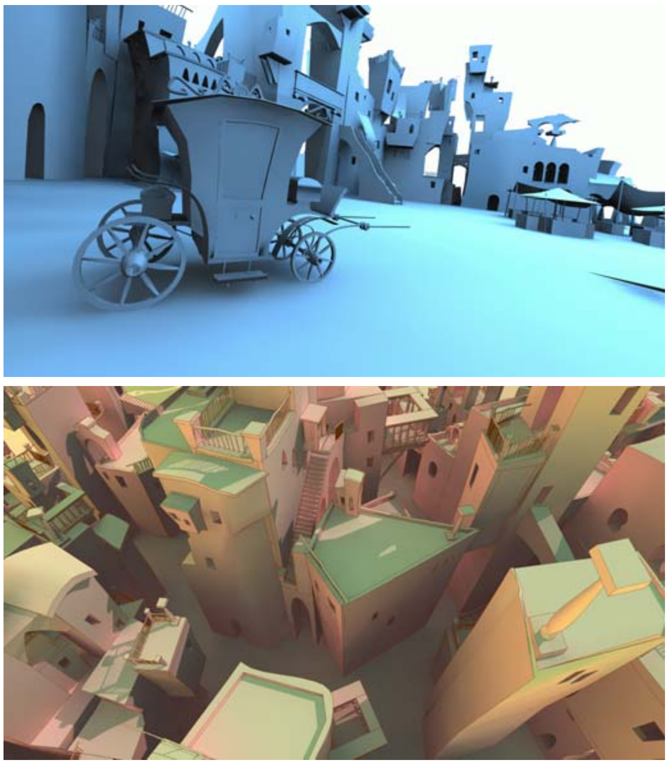
Fig. 7 & 8: some early lighting tests using Final Gathering

3. Final Gathering

Having decided to use Final Gathering, we needed to start rendering and see what we came up with. The first and major problem was the final gathering file itself - we figured out quickly that if final gather was calculated on more than one machine, more than one machine would eventually attempt to write to the FG file simultaneously, and the render would crash badly. The solution was to generate the FG file for the whole scene on one machine only, and disable the writing of FG files when subsuquent renders were launched.
Another trick we discovered was to avoid letting the fg file get over a certain size, say 100Mb. Once over this limit, the file took too long to read. The solution was to delete the file once it neared this size and start creating a new FG file from the appropriate frame. At a high sampling rate, there was no noticeable ‘jump’ between one fg result and another.

Our fg settings hovered at around 300 samples, min 3, max 30 for this scene.

4. Render Optimisation

It proved impossible to render AD002 in one go, as it contained hundreds of high-res textures, 18 million polygons and final gathering, all to be rendered at full 2k resolution.
We attempted to surmount memory problems initially by rendering

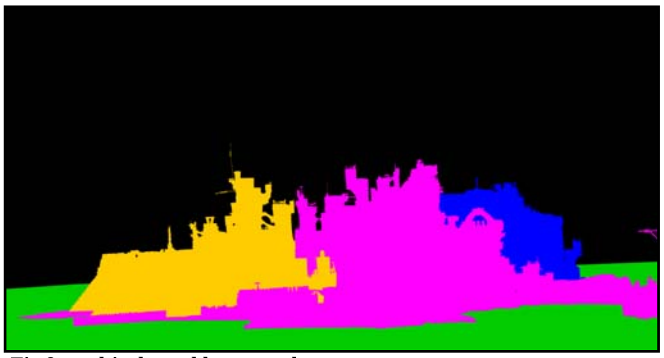
Fig.9: multicoloured layer mask.

on Linux machines instead of Windows, and this did in fact produce more stable rendering. However, it still wasnt enough, and we were forced to develop a strategy to cope with render meltdown.

We split the scene into discreet sections in order to cope. It was divided into 6 main sections, each one progressively further from the camera

1. the port (boats etc.)
2. near buildings: subdivided into 3 subsections
3. Mid buildings: subdivided into 3 subsections
4. Far Buildings: subdivided into 3 subsections
5. Castle
6. Alley: subdivided into 3 subsections

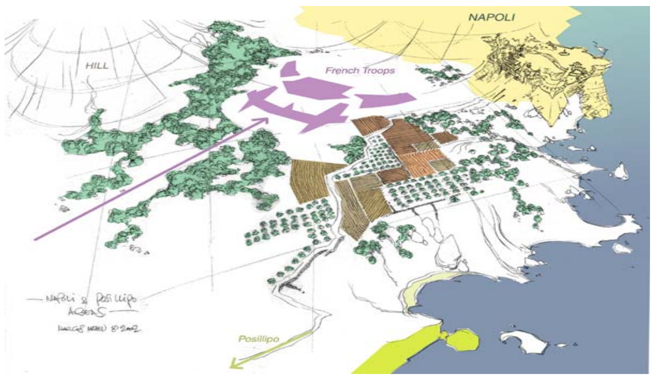

Fig. 10: Geographical Layout of the city of Naples

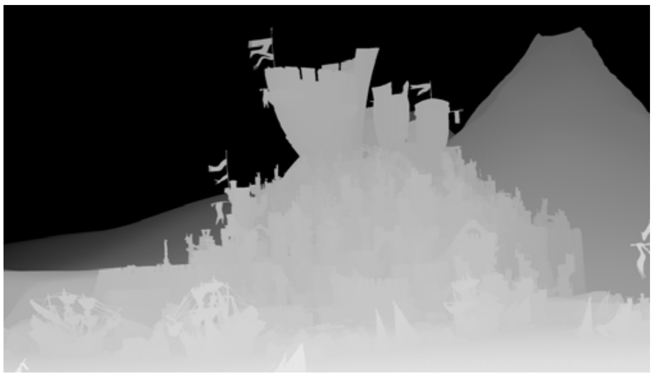
Fig. 11: Composited Z image

In order to render and composite these scenes, we used a multi-coloured mask with a different constant colour for each layer (see fig. X) which allowed us to place each layer in the correct place, providing the necessary occlusion.

To cope with Z-Compositing, a Z conversion utility was developed a proprietry z-processing utility which took softimage floating point zpics, and divided them into discreet “z-blocks” for each layer which were then composited together to provide a single z pass.

Rendering was also optimised with a utility which converted all textures to Mental Ray’s native .map format, which avoided having to load the entire texture into memory at render time (Map is a memo-

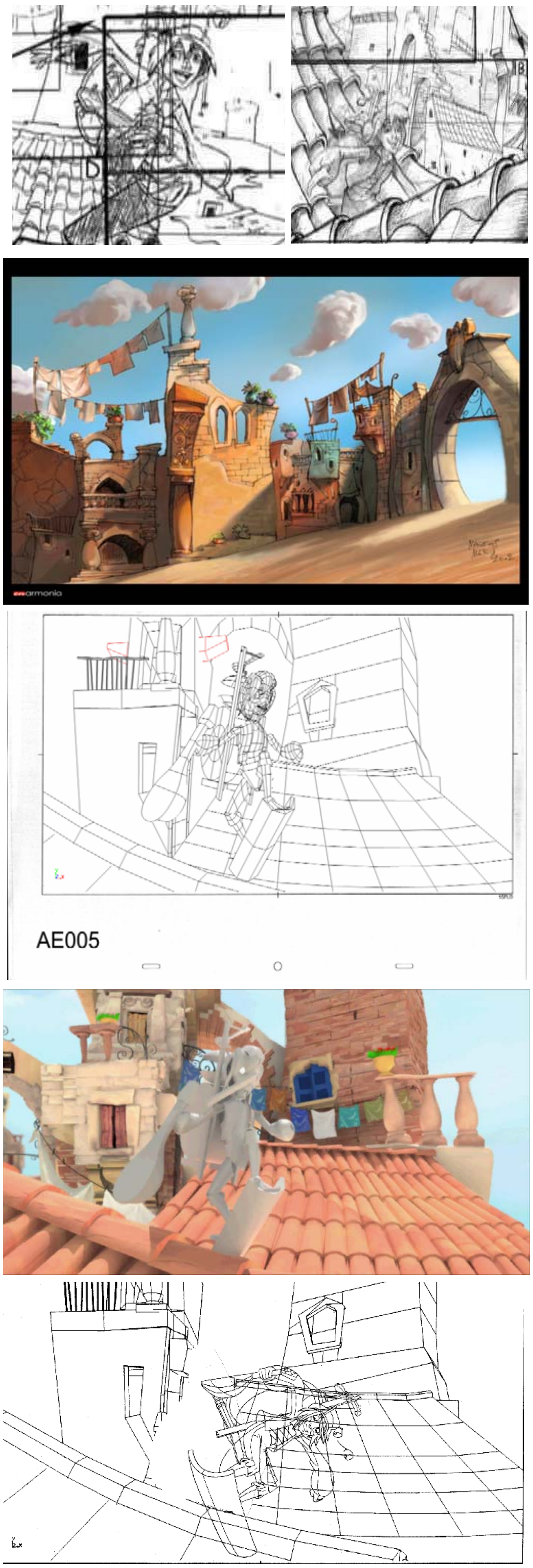

Fig. 12-17: Various stages in the 2D-3D Integration Process

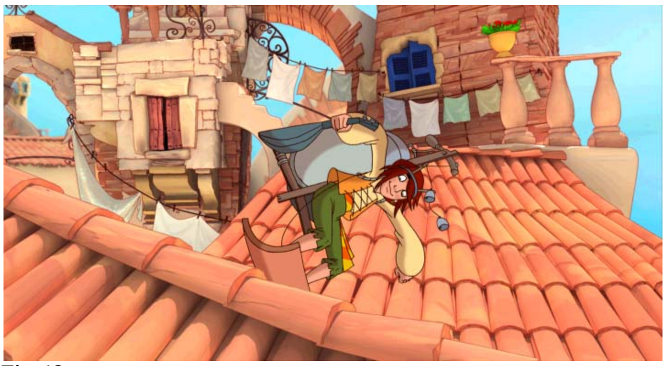

Fig. 18

ry mapped format, which means only the bit actually being rendered is referenced at any given time).

In addition to this, there was the usual tweaking of BSP parameters, and other fiddling with Mental Ray Parameters.

Unfortunately, XSI 3.5 with Mental Ray 3.2 (which included improved BSP and FG handling and diagnostics, and the ability to write incrementalMI files rather than a single large MI file per frame) arrived too late for Toto' Sapore, and massive MI file sizes meant it made more sense to render from the command line using xsibatch rather than directly rendering MI files..

6. 2D Animation Integration Pipeline

There were a number of different kinds of integration with traditional hand-drawn 2D animation

1.Integration of 2D hand-drawn characters with 3D Backgrounds

An example of this is shown on this page

Fig. 12 & 13: The storyboard with rough camera moves

Fig.14: Art Directors visualisation of Naples

Fig. 15: Our Layout for the Animators - this was a wireframe rendering of our background and a 3D Stand-in Character

Fig.16 Final Rendering of Background with 3D standin Character

FIg 17. Animators Pencil test in Traditional Animation (the character is rotoscoped by hand, the backgrounds are our wireframe 3D environment).

Fig. 19: Naples - storyboard image

Fig 18. (top right previous page) Final Composited shot with intergrated Cel-coloured 2D animation character

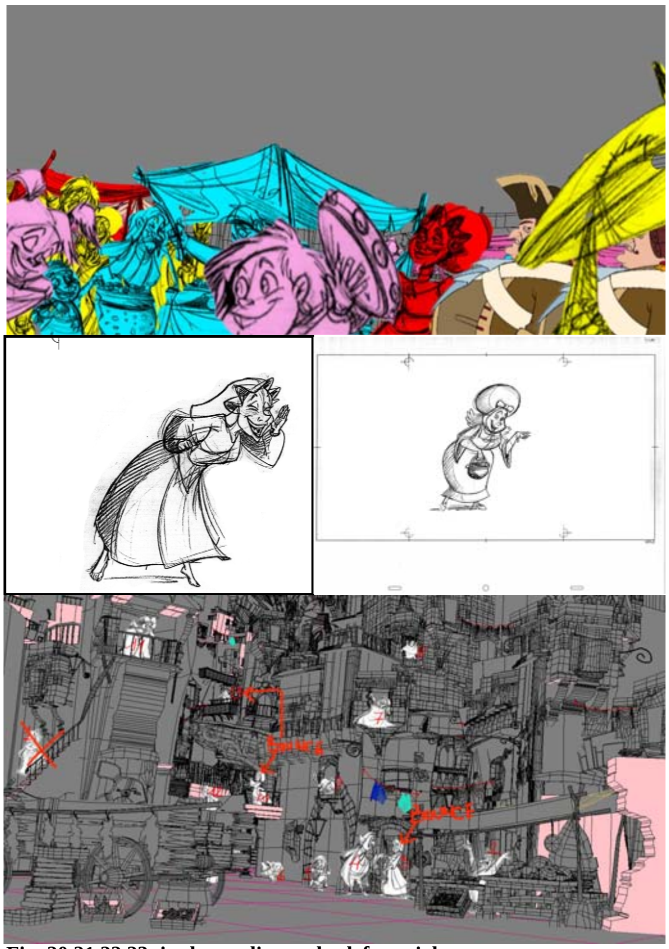
Fig. 20,21,22,23 in descending order left to right

2. Integration of 2D hand-drawn characters into an otherwise completely 3D Scene

This was mainly used for crowd shots with a 3D environment and lots of 2d "extras" moving around the scene.

The workflow was:

Fig 20. Taking images from the animatic, we placed them in the scene in roughly the correct position, mapping them onto grids in Softimage XSI, and working out a rough camera movement

Fig 21 the Director provided us with still images of each character to be inserted into the final scene.

Working with the director, we placed each character into the scene with roughly the correct scaling, position and rotation, and numbered each one (Fig 23)

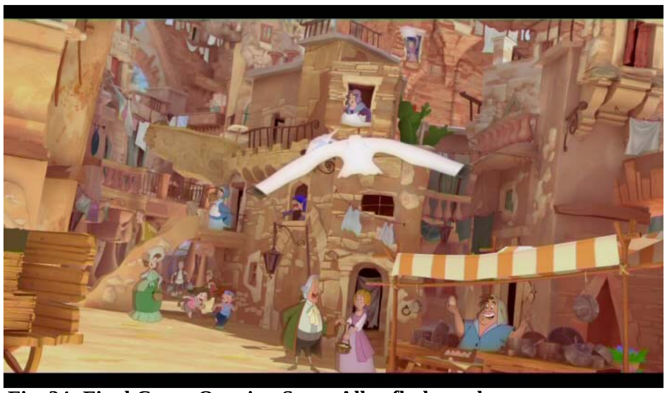
Fig. 24: Final Comp Opening Scene Alley flythrough

(Fig. 22) Characters hand-drawn by animators and passed back to Colouring Department, who colour each character, adding a matte. These layers are then mapped onto the 3D grids which were preously used for the animatic figures, to produce the end result shown above in Fig. 24

5. Render Queue Management

To manage the huge number of scenes to be rendered we needed software to manage the rendering of XSI, various compositing packages on both our renderfarm and Artists workstations (used for rendering at night/weekend/holidays). We tried a number of packages, including Virtual Vertex's Muster, Softimages Batch Render and Royal Render (previously known as "Render Server"). While all of these applications had their good points, we finally settled on Uberware's Smedge (http://www.uberware.net/) both for it's flexibility and it's reasonable price...it proved to be exceptional at managing large numbers of scenes from different programs on large numbers of machines (from dual processor machines with lots of hard disk space and memory to machines normally only used for reading e-mail..)

8. Acknowledgements

Thanks to Maurizio Forestieri, Alessandro Saponi & Carlo Alfano.