Combining Procedural, Polygonal, and Bitmap Representations using XML

Anders Henrysson and Mark Ollila
Norrköping Visualization and Interaction Studio
University of Linköping, Norrköping Campus, Sweden

Hybrid representations - that of using image and procedural methods to synthesize images. Procedural methods allow us to describe media (2D images, 3D objects etc.) with very little information and render it photorealistically. Since the procedure is run on the client (for instance a PC or a mobile phone with limited network), it makes sense to adapt the procedure to the properties of the client.

The traditional usage of procedural methods has been to use low level shading languages or specialized applications only running one procedure and thus limiting the user to one category of media. Even though a lot can be represented with procedural methods there is a need to use traditional media representations (bitmaps, polygons etc.) as well.

We have come up with a concept where we achieve all of the above using hybrid representations. We have adopted an object based media representation where an object can either be represented with a procedure or its traditional representation. To keep the application as small and flexible as possible, each procedure is implemented as a library which is loaded when needed. The media representation is written in XML to make it human readable and easy to edit. The application is thus document driven where the content of the XML document determines which libraries to be loaded. The media objects resulting from the procedures are then composited to the media representation preferred by the renderer together with the non-procedural objects. The parameters in the XML document are relative to parameters determined by the system properties (resolution, performance etc.) and thus adapting the procedures to the client. By mapping objects to individual libraries, the architecture is easy to make multi-threaded and distributed.

The input to the procedures are in text-files pointed to in the XML document and parsed by the procedure library. The procedures in the example presented here include texture synthesis techniques and Perlin noise cloud. To make the procedures adapt to the system properties, the objects are re-rendered when the resolution is changed. The bitmap object is simply scaled, but other objects are recomputed. An array of the appropriate size is created in the main application and a pointer to it is sent to the procedure along with the name of the input file and the size of the region to be rendered. The implementation is limited but shows the strength of the concept and is a work in progress.

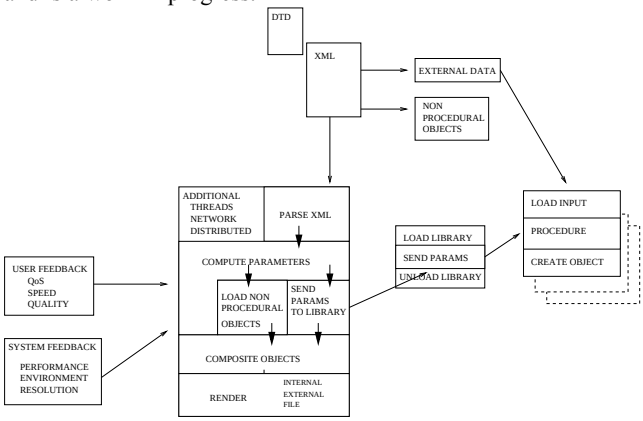

Figure 1: System design of hybrid representations.

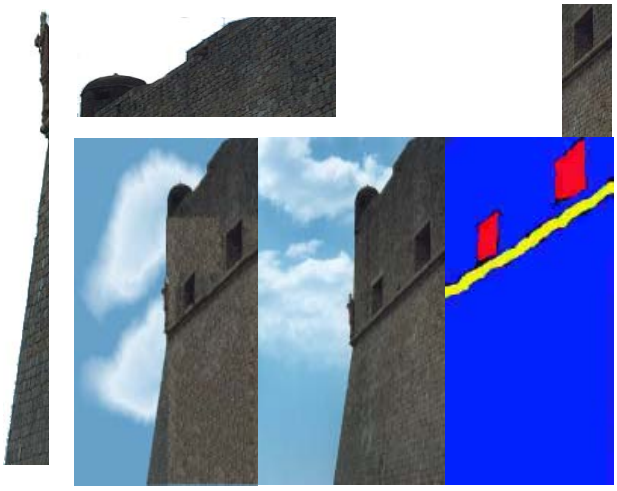

Figure 2: From left moving clockwise across the top, we have two bitmap Objects. We then have the sample texture and target image where synthesis takes place. This is followed by the original image, and the result of the hyrid rendering scheme.

XML description:

```
<?xml version=’1.0' encoding=’utf-8'?>
<!DOCTYPE image SYSTEM “image.dtd”>
<image size_y=’550' size_x=’378'>
<region size_y=’100' size_x=’100' pos_y=’0' pos_x=’0'>
<procedure type=”cloud”>blaise.pcf</procedure>
</region>
<region size_y=’77' size_x=’49' pos_y=’23' pos_x=’51'>
<procedure type=”textureSynthesis”>blaise.tsf</procedure>
</region>
<region size_y=’23' size_x=’51' pos_y=’0' pos_x=’51'>
<procedure type=”bitmap”>top.png</procedure>
</region>
<region size_y=’60' size_x=’12' pos_y=’40' pos_x=’40'>
<procedure type=”bitmap”>left.png</procedure>
</region>
</image>
```

Parameter file cloud “blaise.pcf”:

```
4                       #number of colors to form gradient
255 102 161 201 0   #R G B offset for color one
255 102 161 201 2    #R G B offset for color two
255 224 238 249 40  #R G B offset for color three
255 159 199 225 100 #R G B offset for color four
2                       #number of ellipses
50 30 48 15         #x.center y.center x.radius y.radius for ellipse 1
50 60 48 15         #x.center y.center x.radius y.radius for ellipse 2
0.7                     #whispiness
```

Parameterfile texture synthesis”blaise.tsf”:

```
1                       #has target
blaise_t.jpg        #target image
1                       #number of input images
blaise_in_0_100.jpg#input image 1
0 100                   #y,x coords for image 1 in the target image
12                      #num: num x num = # of sample points region
30                      #regsize: size of region to be quilted
0.45                    #overlap: size of overlap between regions
0.85                    #alpha:
```