Master Thesis

Space-optimized texture atlas

Jonàs Martínez Bayona

Advisor: Carlos Andujar

Master in Computing

Universitat Politècnica de Catalunya
Motivation

- Many applications rely on heavily-textured models with a large number of periodic textures
- 3D model of Barcelona
  - 93,111 buildings
  - 23,939 distinct textures, most of them periodic
Motivation

Real-time rendering of detailed textured models is still a challenging problem:

- Per-corner attributes
- Poor hardware support for per-corner binding. Difficult to use VBOs
- Texture switching
  - Texture atlases
    - Barcelona with wrapping 18 GB
    - Barcelona without wrapping 12.5 TB
  - Texture Arrays
Contributions

- An algorithm for creating a space-optimized texture atlas from heavily-textured models. Novel elements include:
  - A pipeline to generate space-optimized atlases
  - An efficient algorithm for resizing each chart in accordance with its perceptual importance
  - A packing algorithm achieving 100% coverage
  - Within-chart tiling support for periodic textures
- Application to visualization of a huge city
Overview

Texture atlas creation

Input textures \rightarrow Image downsampling \rightarrow View-based downsampling \rightarrow Perception-based downsampling \rightarrow Texture packing \rightarrow Texture atlas

Object-space coverage \rightarrow Visual tolerance

Real time visualization

LOD hierarchy \rightarrow Texture wrapping
Overview

Texture atlas creation

Input textures → Image down sampling → Viewing-based down sampling → Perception-based down sampling → Texture packing → Texture atlas

Object-space coverage → Visual tolerance

Real time visualization

LOD hierarchy → Texture wrapping
Image downsampling to match viewing conditions

- Downsample the textures according to the texture coverage and a user-defined texel size $\lambda$ (cm/texel)
- $(W_i, H_i)$ Size of input image
- $(W_s, H_s)$ Area each tile is mapped onto
- If $(\lambda W_i > W_s)$ or $(\lambda H_i > H_s)$
  - Downsample to $(W_s/\lambda, H_s/\lambda)$
- We need $\lambda$ parameter to implement texture LOD
Overview

Texture atlas creation

Input textures → Image downsampling → Perception-based downsampling → Texture packing → Texture atlas

Object-space coverage → Visual tolerance

Real time visualization

LOD hierarchy
Texture wrapping
Image downsampling to match image saliency

- $M = \text{Image error metric}$
- $I_i = \text{Input image}$
- $I_o = \text{Compressed image}$
- $\alpha = \text{Maximum user-defined error}$

$$M \left( I_i, \tilde{I}_o \right) \leq \alpha, \alpha \in [0, 1]$$
Mean squared error metric

- Measures the average of the square of the error:

\[ MSE = \sum_{x=1}^{W} \sum_{y=1}^{H} (a_{x,y} - b_{x,y})^2 \]

- Error tends to increase slowly as we increase the downsampling factor.

- Signal fidelity is independent of temporal or spatial relationships between the samples of the original signal.
Mean squared error metric

![Graph showing width and height subsampling](image)
Human visual system metric

- Produces a probability map for difference detection as output based on (VDP, Daly ’93):
  1. Sensitivity to contrast changes decreases with increasing light levels.
  2. Sensitivity decreases with increasing spatial frequency.
  3. Variations in sensitivity due to the signal content of the background (masking).
Human visual system metric
Searching compressed size

- Perform a search of the optimum \((w_0, h_0)\) values using a dicotomic search
- Split the 2D rectangular interval of the possible dimensions alternating horizontal and vertical subdivisions
- Similar results varying the search order
Searching compressed size

- Upper row: search on width followed by search on height
- Lower row: alternating search on width and height
Overview

Texture atlas creation

Input textures → Image downsampling → Texture atlas

- Viewing-based downsampling
- Perception-based downsampling
- Object-space coverage
- Visual tolerance
- Texture packing

Real time visualization

- LOD hierarchy
- Texture wrapping
Packing rectangles into an atlas

- Uses a texture atlas binary tree structure
- A non-empty node defines a chart of the texture atlas
Packing overview

1. Sort textures from biggest to smallest
2. Calculate the minimum size of texture atlas
3. Insert all the textures recursively. Split the node in the direction with more space if node have different size
4. Optimize inserted textures
Optimizing inserted textures

- We are restricted to have power-of-two texture atlas. Optimize process consists in:
  1. Binary search taking the lower bound as the initial size of each inserted textures and the upper bound the maximum size
  2. For each pair of leafs of the binary tree, if we have one filled and the other one is empty expand the filled one to occupy all the empty space
Texture packing example

<table>
<thead>
<tr>
<th>Not optimized</th>
<th>Optimized</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not stretched</td>
<td>Stretched</td>
</tr>
<tr>
<td></td>
<td>Not stretched</td>
</tr>
</tbody>
</table>

![Not optimized](image1)
![Stretched](image2)
![Not stretched](image3)
![Stretched](image4)
Overview

Texture atlas creation

Input textures → Image downsampling → Texture atlas

- Viewing-based downsampling
- Perception-based downsampling
- Texture packing
- Object-space coverage
- Visual tolerance

Real time visualization

- LOD hierarchy
- Texture wrapping
Texture wrapping

- For each chart:
  1. Origin \((O_x, O_y)\)
  2. Size \((S_x, S_y)\)

- Mapping function:

\[
\begin{bmatrix}
  s' \\
  t'
\end{bmatrix} = \begin{bmatrix}
  S_x & 0 & O_x \\
  0 & S_y & O_y
\end{bmatrix} \begin{bmatrix}
  \text{fract}(s) \\
  \text{fract}(t) \\
  1
\end{bmatrix}
\]
Texture wrapping encoding options

- Encode \((O, S)\) in a \texttt{vec4} uniform variable sent for each textured primitive.
- Send \((O, S)\) data as attributes. Must be specified on a per-corner basis.
- Encode \((O, S)\) data as part of texture coordinates.

<table>
<thead>
<tr>
<th>Encoding of ((O, S))</th>
<th>Memory Space</th>
<th>App(\rightarrow)GPU</th>
<th>Vertex Program</th>
<th>VBO compatibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option 1 \texttt{uniform vec4}</td>
<td>4 floats/T</td>
<td>4 floats/T</td>
<td>none</td>
<td>No</td>
</tr>
<tr>
<td>Option 2 \texttt{attribute vec4}</td>
<td>4 floats/T</td>
<td>12 floats/T</td>
<td>copy \texttt{vec4} to varying</td>
<td>Yes</td>
</tr>
<tr>
<td>Option 3 As part of (s,t) coords</td>
<td>none</td>
<td>none</td>
<td>decode (s,t,\text{origin, size})</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Compressed texture coordinates

- Maximum texture atlas size: 4096x4096
- Maximum chart size: 512x512
- Chart’s size must be multiple of four
- Each chart repeated up to 64 times
Overview

Texture atlas creation

Input textures → Image downsampling → Texture packing

Viewing-based downsampling
Perception-based downsampling

Object-space coverage
Visual tolerance

Texture atlas

Real time visualization

LOD hierarchy
Texture wrapping
Real-time visualization

- A Texture Atlas Tree defines a quadtree subdivision of the scene in the x-z-plane and a corresponding hierarchy of texture atlases.
- Hierarchy can be seen as a coarse-level collection of mipmapped texture atlases.
Real-time visualization

- We store a value that defines the texel size (cm/texel) for each node

\[ \lambda_l = \text{Texel size of the highest resolution level} \]

\[ \alpha = \text{Downsampling scale factor} \]

\[ \lambda(L) = \text{Texel size of the level L in atlas hierarchy} \]

\[ \lambda(L) = \alpha^{L-1} \lambda_l \]
Real-time visualization

- Minimize the amount of memory using an adaptive scheme capable to use texture atlas with different mip-levels
  \[ Mip(L) = \begin{cases} 
  0 & L = N \\
  \log_2(\alpha) - 1 & L < N 
  \end{cases} \]

- We are storing the pyramid of mip-levels explicitly in the quadtree hierarchy and implicitly in the mip-levels associated to each texture atlas.
Real-time visualization

- Rendering is performed by a top-down traversal of the texture atlas tree.
- The texel-per-screen pixel ratio can be increased to decrease the texture quality and increase the performance.

\[ S_{proj} = \frac{\lambda w}{2d \tan \left( \frac{f}{2} \right)} \]
Real-time visualization

- If $S_{proj}$ is less or equal than a texture resolution threshold (texel/pixel), the texture atlas of the current level of resolution is used to wrap the textures.

- The quadtree subdivision guarantees that for a given maximum deep, we also have an upper bound of texture switches.
# Test model: Barcelona 3D

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Triangles</strong></td>
<td>3,266,469</td>
</tr>
<tr>
<td><strong>Vertices</strong></td>
<td>6,307,109</td>
</tr>
<tr>
<td><strong>Buildings</strong></td>
<td>93,111</td>
</tr>
<tr>
<td><strong>Building blocks</strong></td>
<td>13.879</td>
</tr>
<tr>
<td><strong>Terrain area</strong></td>
<td>108.43 $km^2$</td>
</tr>
<tr>
<td><strong>Total surface area (facades+ceilings)</strong></td>
<td>234.4 $km^2$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Number of distinct textures</strong></th>
<th>23,939</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Memory space without compression</strong></td>
<td>18 GB (3 bytes/texel)</td>
</tr>
<tr>
<td><strong>Memory space with DXT1 compression</strong></td>
<td>3 GB (0.5 bytes/texel)</td>
</tr>
<tr>
<td><strong>Memory space with JPEG compression</strong></td>
<td>0.5 GB (36:1)</td>
</tr>
<tr>
<td><strong>Average resolution</strong></td>
<td>0.5 $cm^2$/texel</td>
</tr>
<tr>
<td><strong>Needed texture memory space for all the city without wrapping</strong></td>
<td>4.2 Tera texels</td>
</tr>
<tr>
<td><strong>Needed memory space for all the city without compression</strong></td>
<td>12.6 TB (3 bytes/texel)</td>
</tr>
<tr>
<td><strong>Needed memory space for all the city with DXT1 compression</strong></td>
<td>2.1 TB (0.5 bytes/texel)</td>
</tr>
</tbody>
</table>
Downsampling results
Reconstruction of test images with varying RMSE visual tolerance
Reconstruction of test images with varying HVSE visual tolerance
## Image downsampling test model

<table>
<thead>
<tr>
<th>Uncompressed size (MB)</th>
<th>View matching size (MB)</th>
<th>CR View matching</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 0</td>
<td>622130.85</td>
<td>437460.93</td>
</tr>
<tr>
<td>Level 1</td>
<td>622130.85</td>
<td>3860.91</td>
</tr>
<tr>
<td>Level 2</td>
<td>622130.85</td>
<td>16.41</td>
</tr>
<tr>
<td>Total</td>
<td>1866395.80</td>
<td>441396.86</td>
</tr>
</tbody>
</table>

- **RMSE 10%**

<table>
<thead>
<tr>
<th>Saliency matching size (MB)</th>
<th>CR wrt view matching (MB)</th>
<th>Total CR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 0</td>
<td>13079.76</td>
<td>33:1</td>
</tr>
<tr>
<td>Level 1</td>
<td>3337.20</td>
<td>1.16:1</td>
</tr>
<tr>
<td>Level 2</td>
<td>15.87</td>
<td>1.03:1</td>
</tr>
<tr>
<td>Total</td>
<td>16432.</td>
<td>26:1</td>
</tr>
</tbody>
</table>

- **HVSE 10%**

<table>
<thead>
<tr>
<th>Saliency matching size (MB)</th>
<th>CR wrt view matching (MB)</th>
<th>Total CR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 0</td>
<td>243120.43</td>
<td>1.8:1</td>
</tr>
<tr>
<td>Level 1</td>
<td>3642.16</td>
<td>1.06:1</td>
</tr>
<tr>
<td>Level 2</td>
<td>15.56</td>
<td>1.05:1</td>
</tr>
<tr>
<td>Total</td>
<td>246778.16</td>
<td>1.79:1</td>
</tr>
</tbody>
</table>
Encoding texture chart coordinate

- Option 3 (packed coordinates) has the highest vertex rate and framerate, despite decompression overhead at vertex level

<table>
<thead>
<tr>
<th>Encoding technique</th>
<th>Vertices/s</th>
<th>Fragments/s</th>
<th>Frames/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option 1</td>
<td>44,032,002</td>
<td>1,570,752,000</td>
<td>14.57</td>
</tr>
<tr>
<td>Option 2</td>
<td>436,739,971</td>
<td>1,382,832,000</td>
<td>132.37</td>
</tr>
<tr>
<td>Option 3</td>
<td>527,426,950</td>
<td>1,362,096,000</td>
<td>163.71</td>
</tr>
</tbody>
</table>
Performance
Performance

![Performance Chart]

- Encoding option 3
- Encoding option 2
- Encoding option 1
- VBO / No texture atlas
- Immediate / No texture atlas

Frames/s

Maximum
Minimum
Average
Previous work

- Image comparison metrics
  1. Spatial domain metrics (RMSE, template matching).
  2. Spatial-frequency domain metrics (Fast Fourier Transform, CSF).
  3. Perceptually-based metrics (VDP).
  4. Tone mapping metrics (single and multi scale tone reproduction operators)
Buchholz and Döllner presented a level-of-detail texturing technique that creates a hierarchical data structure for all textures used by scene objects.
Summary of contributions

- **Creating optimized texture atlas**

  1. An algorithm for resizing each chart in accordance with the **object-space size** of the surface the chart is mapped onto and the **perceptual importance** under a given viewing conditions.

  2. An algorithm to **pack rectangular charts** into a single texture that minimizes the unused space.
Summary of contributions

- Rendering optimized texture atlas
  1. Implicit mipmap levels per atlas using compressed texture coordinate format designed to support tiled textures avoiding the unfolding of periodic textures. Several shader techniques providing within-chart tiling support and decompression of texture coordinates.
  2. Full support to DXT1 formats avoiding artifacts due the texture atlas compression. A texture atlas hierarchy supporting explicit user-defined texture LOD.
Overview

Texture atlas creation

Input textures

Image downsampling

Viewing-based downsampling

Perception-based downsampling

Object-space coverage

Visual tolerance

Texture packing

Texture atlas

Real time visualization

LOD hierarchy

Texture wrapping
Texture mipmapping

- Artifacts may appear at the borders as we use texels from foreign textures to filter
- Need to determine the lowest mip-level that we are able to use
Texture mipmapping

- For the highest resolution level is enough to clamp to the edge texture coordinates:

  \[(s, t) \in \left( \frac{1}{2width} \ldots 1 - \frac{1}{2width}, \frac{1}{2height} \ldots 1 - \frac{1}{2height} \right)\]

- Two options to provide bilinear filtering:
  1. Clamp atlas coordinates taking into account which mip-level the texture operation is about to access
  2. Pad charts with border texels
Texture mipmapping

The interpolation in the lower mip-levels takes only the colours of the chart, not the neighbours:

\[
 f_b (M) = \begin{cases} 
 0 & M = 0 \\
 2^{M-1} & M > 0 
\end{cases}
\]

The corner points clamped to the edge never reaches further from the dead-center.
Texture mipmapping

- Incorrect bilinear filtering:

- Correct bilinear filtering:
DXT1 texture atlas compression

- DXT1 breaks a texture map into $4 \times 4$ blocks of compressed texels. Artifacts appear in the borders of mapped textures.
- Solution: have in the lowest mip-level a set of charts with a dimension multiple and not less of four:

$$f_d (M, \alpha) = 4\alpha (M + 1) \quad \alpha \geq 1$$
Selected snapshots
Application demo...
Thank you for your attention