The Sphinx Thesis Resource (sphinxtr)

Jeff Terrace

A Dissertation

Presented to the Faculty

of Princeton University

in Candidacy for the Degree

of Doctor of Philosophy

Recommended for Acceptance

by the Department of

Computer Science

Adviser: Professor John P. Smith

November 2012

Abstract

This project is a collection of extensions and monkey patches to Sphinx to better format a PhD thesis.

Acknowledgements

Thanks to Sphinx for a kickass build system and docutils for the groundwork of multiple format output.

To my parents.

Contents

List of Figures

Chapter 1

Introduction

Creating a PhD thesis is typically done using LaTeX. This works really well for producing a PDF, but a giant PDF file is not a great way to put documents on the web. There are solutions that exist to turn latex source files into HTML, but in my experience, they tend to produce poor HTML output.

The Sphinx project is a wonderful tool for creating portable documents, allowing for output to many different formats. Unfortunately, it has many shortcomings when trying to typeset something so advanced as a PhD thesis. The aim of this project is to modify Sphinx to support all of the needs of a thesis writer. Many of the patches are not appropriate for contributing directly to the upstream Sphinx repository, so this is instead a separate project.

This sphinxtr output is available in several formats at: http://jterrace.github.com/sphinxtr.

The source code for sphinxtr can be found at: https://github.com/jterrace/sphinxtr.

1.1 Installation

Install the required Python packages:

```
pip install -r requirements.txt
```

1.2 Building

You need `make`. The following targets are supported:

html

Builds HTML format, separated into sections

singlehtml

Builds HTML format on a single page

text

Builds text files, separated into sections

singletext

Builds a single text file

latexpdf

Builds into latex source files and then compiles into a PDF. Requires latex.

1.3 Changes

The following changes and additions have been made from vanilla Sphinx:

- A cross-format bibtex bibliography based on sphinx-natbib

- Tables that can go inside figures

- Changed table formatting to look pretty, like booktabs

- Improved alignment in table environment

- Added support for short captions that show up in the "list of figures" section

- Changed equation reference formatting from "(1)" to "1"

- Full customization of latex preamble and style file

- Numbered figures

- Numbered section references

- A singletext output that builds into a single text file, similar to singlehtml

- A subfigure environment

1.4   Documents Using sphinxtr

- Jeff Terrace's PhD Thesis

Chapter 2

Typography

This chapter has typography stuff. This also shows how to have a toctree inside a chapter.

2.1 Headings

The title of this chapter, "Typography", is the first heading level. This section, "Headings", is the second level.

2.1.1 Third-level

The third level heading. You probably shouldn't go beyond this because it just looks ridiculous, but you can anyway.

Fourth-level

The fourth level heading.

Fifth-level   The fifth level heading.

Rubric Heading   A rubric heading is just a paragraph heading without document structure.

2.2   Text

You can make *emphasized text.*

You can make **bold text**.

You can make `fixed-width font`.

You can make block quotes:

> this is a block quote

You can make code blocks:

```
this is a code block
```

Some convenience substitutions are defined in the epilog:

- e.g.,

- i.e.,

- et al.

- dash—

- non-breaking space -> <-

You can have inline ^{superscript} or $_{subscript}$ text.

Big quotes, also known as an epigraph:

Your avatar can look any way you want it to, up to the limitations of your equipment. If you're ugly, you can make your avatar beautiful. If you've just gotten out of bed, your avatar can be wearing beautiful clothes and professionally applied makeup. You can look like a gorilla or a dragon, or a giant talking penis in the Metaverse. Spend five minutes walking down the street, and you will see all of these.

—Neal Stephenson, Snow Crash

Chapter 3

Lists

This is a few examples of different list types.

3.1 Unordered Lists

- item 1
- item 2
  - item 3
  - item 4
- item 5

3.2 Ordered Lists

1. item 1
2. item 2

(a) item 3

(b) item 4

3. item 5

3.3   Description Lists

term1

  definition 1

term2

  definition 2

term3

  definition 3

3.4   Mixed

term

  definition

- list1

- list2

term2

  something

1. num1

2. num2

- list1

- list2

3. num3

Chapter 4

Math

Math uses latex math syntax:

$$A_c'' = \sqrt[3]{(\frac{L_c^2}{\sum L^2})(\frac{A_c}{\sum A})(\frac{A_c'}{\sum A'})} \cdot T$$

Equations can have labels which you can reference 4.1.

$$\frac{log(1 + E_{current})}{log(1 + E_{max})} \tag{4.1}$$

Chapter 5

References and Citations

You can reference a section by its label. This chapter is Chapter 5.

5.1   Subsection

This subsection is Section 5.1.

5.2   Citations

COLLADA [1] is a cool 3D file format. I wrote a paper about 3D stuff [3]. The website we built is running [2]. The bibliography is in bibtex format.

5.3   Footnotes

Reference a footnote [1].

---

[1] This is a footnote at the end of the page or document.

5.4   External Links

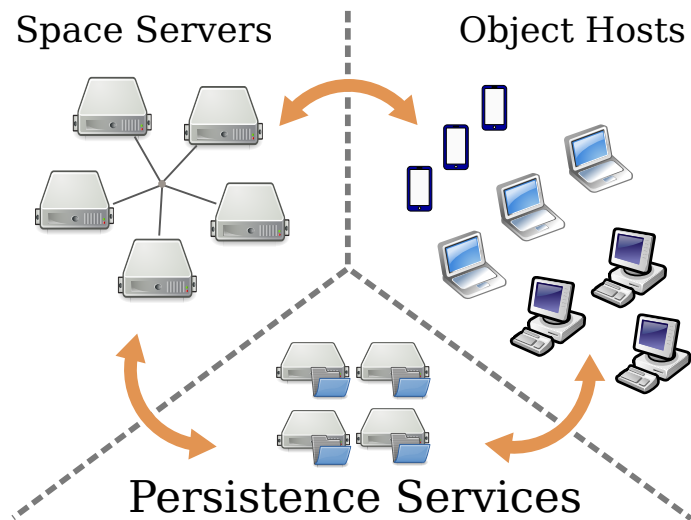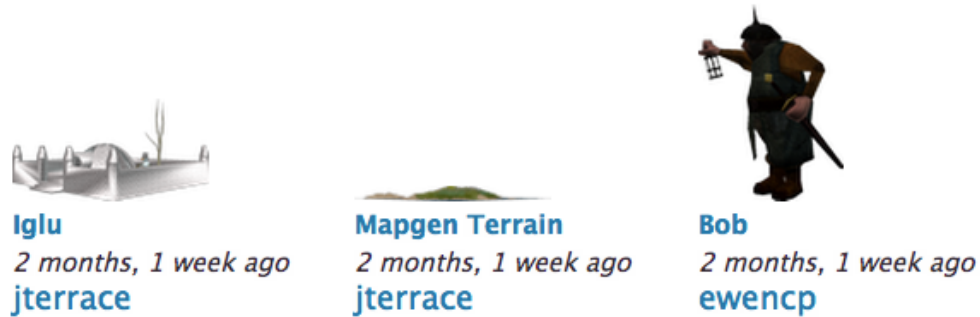You can link to a website.

Chapter 6

Figures and Tables

6.1   Vector SVG Figures

Vector figures are nicely supported. You should have a PDF file and an SVG file.
The PDF will be used for the latex output and the SVG for the HTML output. The
HTML output has a nice zoom feature using Colorbox.

Figure 6.1: The Sirikata metaverse platform architecture.

Figure 6.2: The Open3DHub website allows browsing of 3D meshes.

See an example in Figure 6.1. I suggest making figures in something like Inkscape. If you have only a vector PDF, you can use pdf2svg to convert (`brew install pdf2svg` or `apt-get install pdf2svg`).
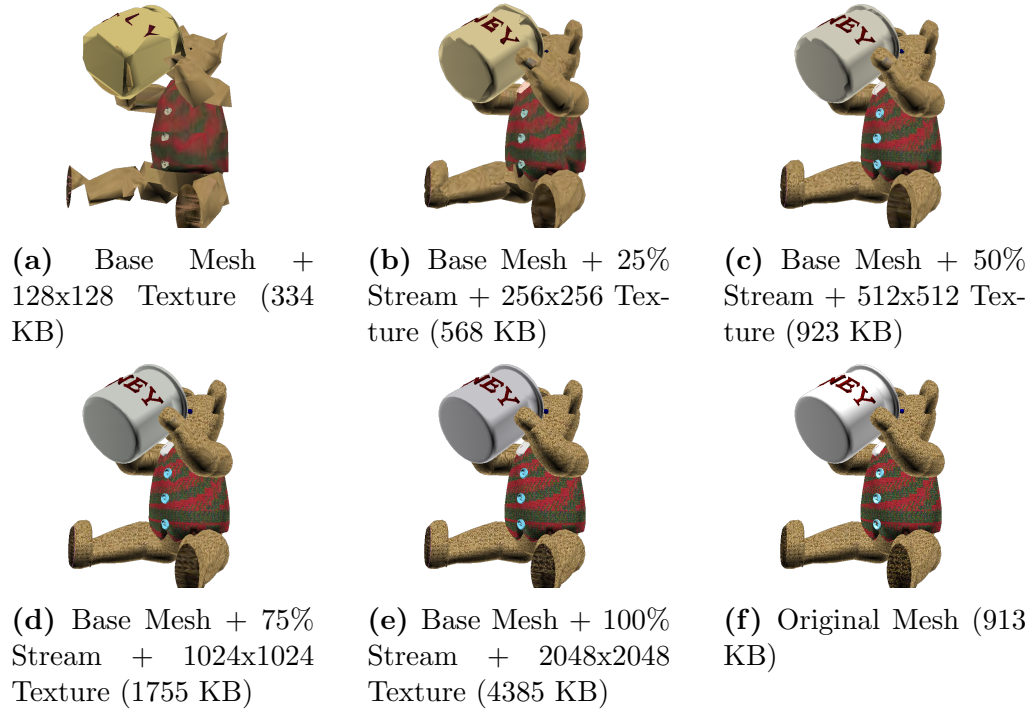
6.2 Image Figures

Regular rasterized images work fine too.

A PNG example is shown in Figure 6.2.

6.3 Subfigures

The subfigure directives allow you to place multiple figures side-by-side in the document. Here's an example:

You can reference the entire Figure 6.3 or one of its subfigures, e.g., Figure 6.3f.

14

(a) Base Mesh + 128x128 Texture (334 KB)

(b) Base Mesh + 25% Stream + 256x256 Texture (568 KB)

(c) Base Mesh + 50% Stream + 512x512 Texture (923 KB)

(d) Base Mesh + 75% Stream + 1024x1024 Texture (1755 KB)

(e) Base Mesh + 100% Stream + 2048x2048 Texture (4385 KB)

(f) Original Mesh (913 KB)

Figure 6.3: Example of a teddy bear model at different resolutions of the progressive format (1 draw call) and its original format (16 draw calls). The size in KB assumes downloading progressively, e.g., 6.3e's size includes lower-resolution textures.

Progressive	128	256	512	1024	2048
0%	0.53	0.63	0.81	1.03	1.35
25%	0.65	0.75	0.97	1.16	1.45
50%	0.74	0.85	1.02	1.26	1.58
75%	0.79	0.95	1.11	1.34	1.70
100%	0.88	0.99	1.20	1.44	1.82

Figure 6.4: Mean size of progressive format as a fraction of the original across all test models, shown as a function of the progressive stream downloaded and texture resolution.

6.4 Table

Tables can be put inside the figtable directive which automatically numbers them, adds a caption, and adds a label.

Table 6.4 has all right-aligned columns.

Left Align	Right Align
Some text is left align	Followed by right-aligned
Some more text here	And more text here
And even more text	Also even more text here

Figure 6.5: This table has mixed alignment

Table 6.5 has one column left-aligned and one column right-aligned.

6.5 Text Wrapping Table

Text wrapping in tables work if you specify the width and either raggedleft or raggedright.

A text wrapping table example is shown in Figure 6.6.

Column Family	Description
Users	Stores a list of users who have authenticated with OpenID.
Names	Stores a list of the 3D models in the database with their associated metadata.
TempFiles	Temporarily stores the binary file data of uploaded files until they have been processed.
Files	Stores the binary file data for uploaded and verified files.
Sessions	Stores HTTP session information used by the Django framework to look up session state associated with a user's browser cookie.
OpenIdAssocs, OpenIdNonces	Stores OpenID authentication information for users.
CeleryResults	Stores the result of application processing tasks (see Section something).
APIConsumers	Stores a list of consumers of the API for use with the OAuth protocol.

Figure 6.6: A list of Open3DHub's Cassandra column families and their descriptions

Bibliography

[1] Khronos Group Inc., The. COLLADA - Digital Asset Schema Release 1.4.1 Specification (2nd Edition). `http://www.khronos.org/files/collada_spec_1_4.pdf`, 2008.

[2] Open3DHub. `http://open3dhub.com/`.

[3] J. Terrace, E. Cheslack-Postava, P. Levis, and M. J. Freedman. Unsupervised Conversion of 3D Models for Interactive Metaverses. In *Proc. ICME '12*, 2012.