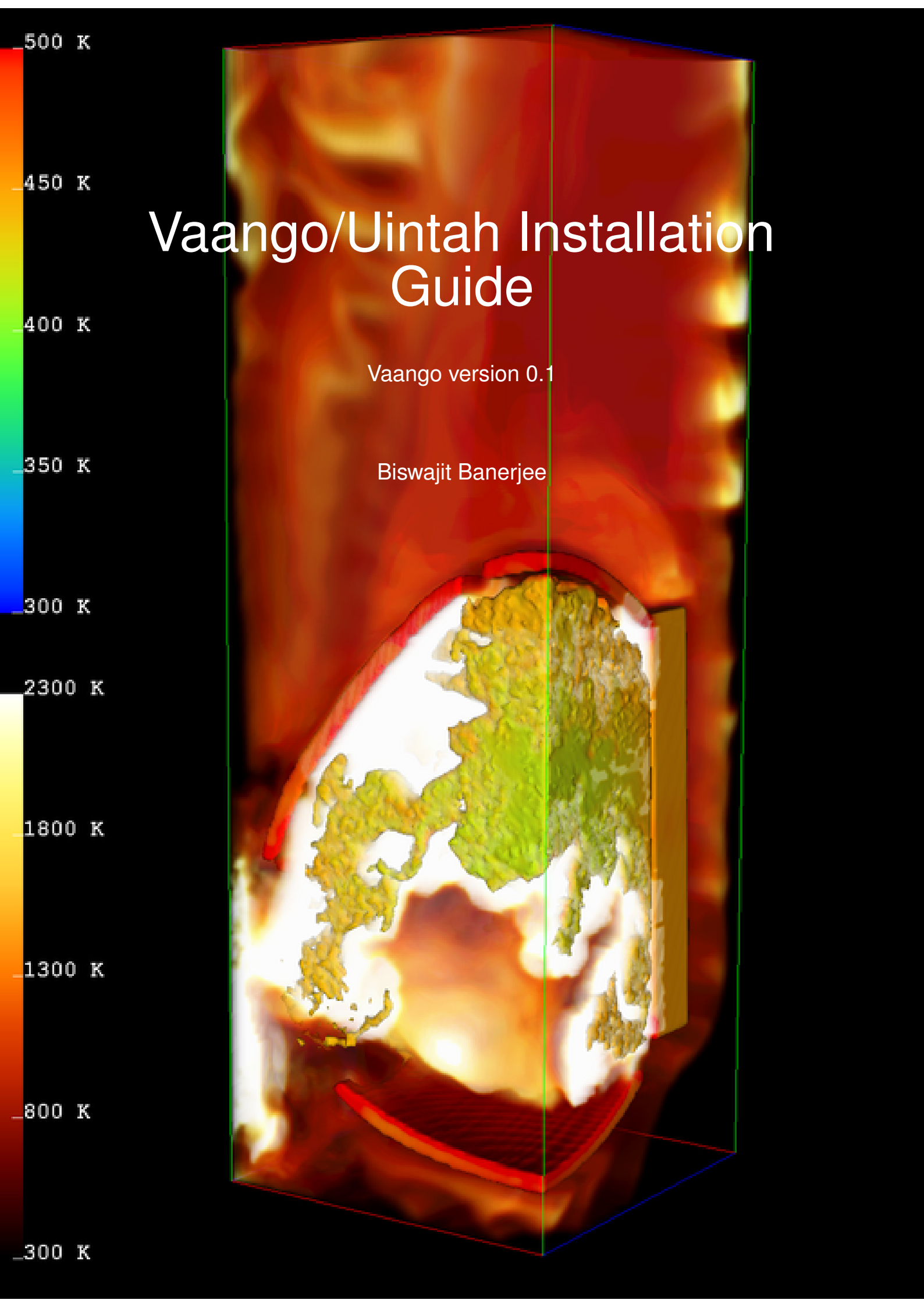
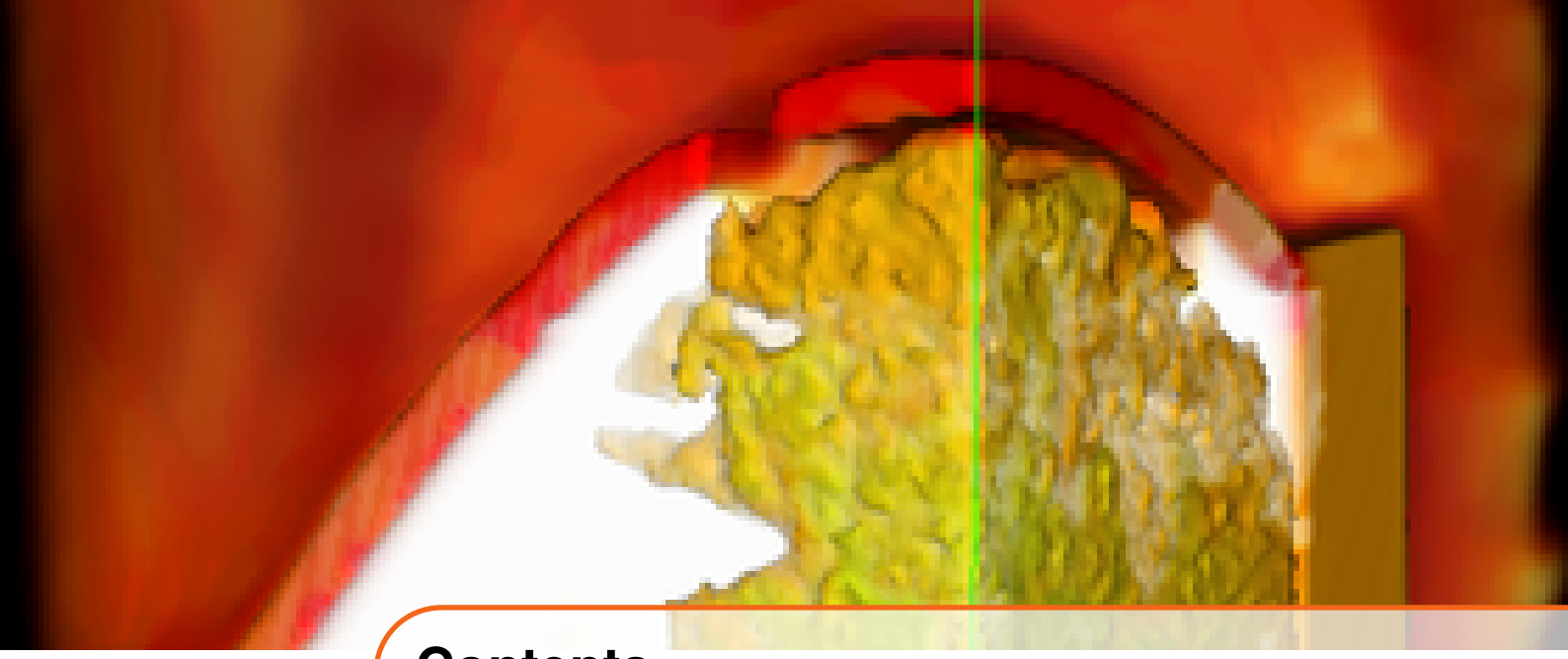


Vaango/Uintah Installation Guide

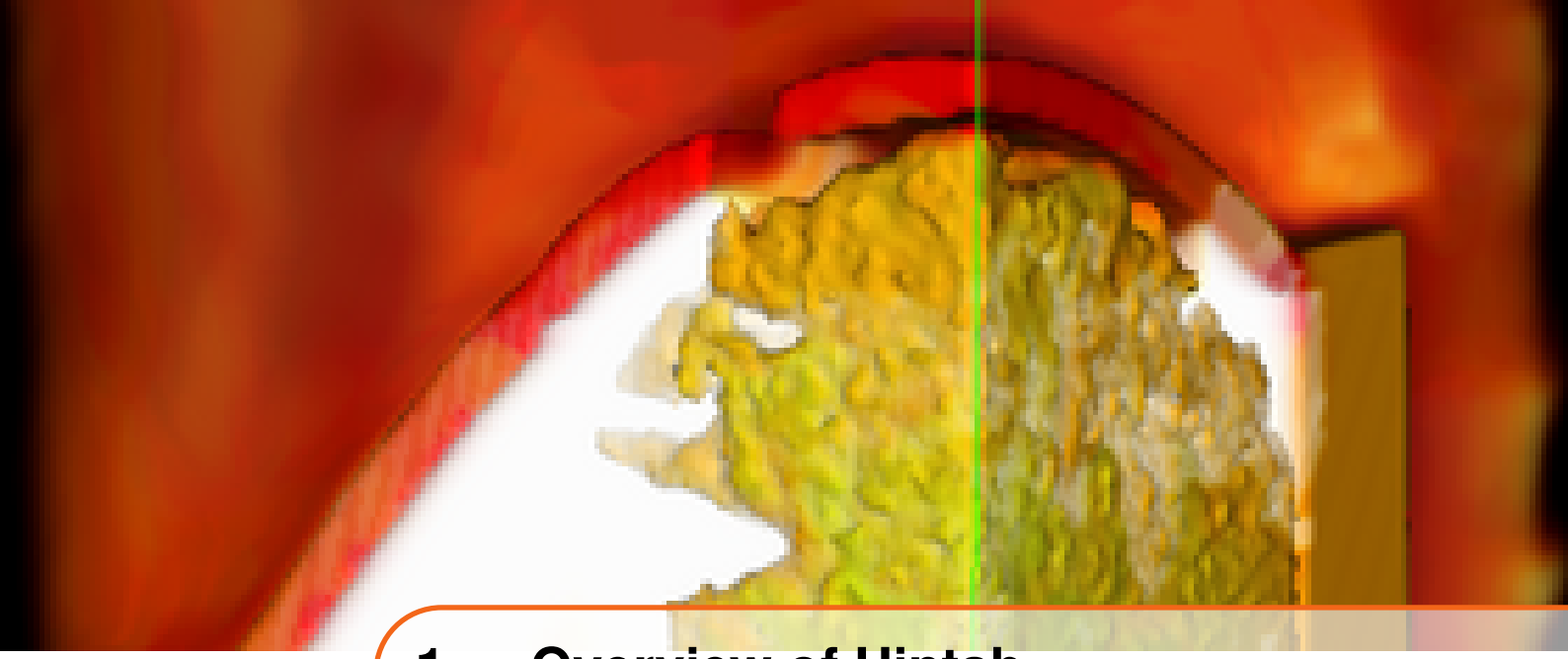
Vaango version 0.1

Biswajit Banerjee

The contents of this manual can and will change significantly over time. Please make sure that all the information is up to date.



Contents



1 — Overview of Uintah

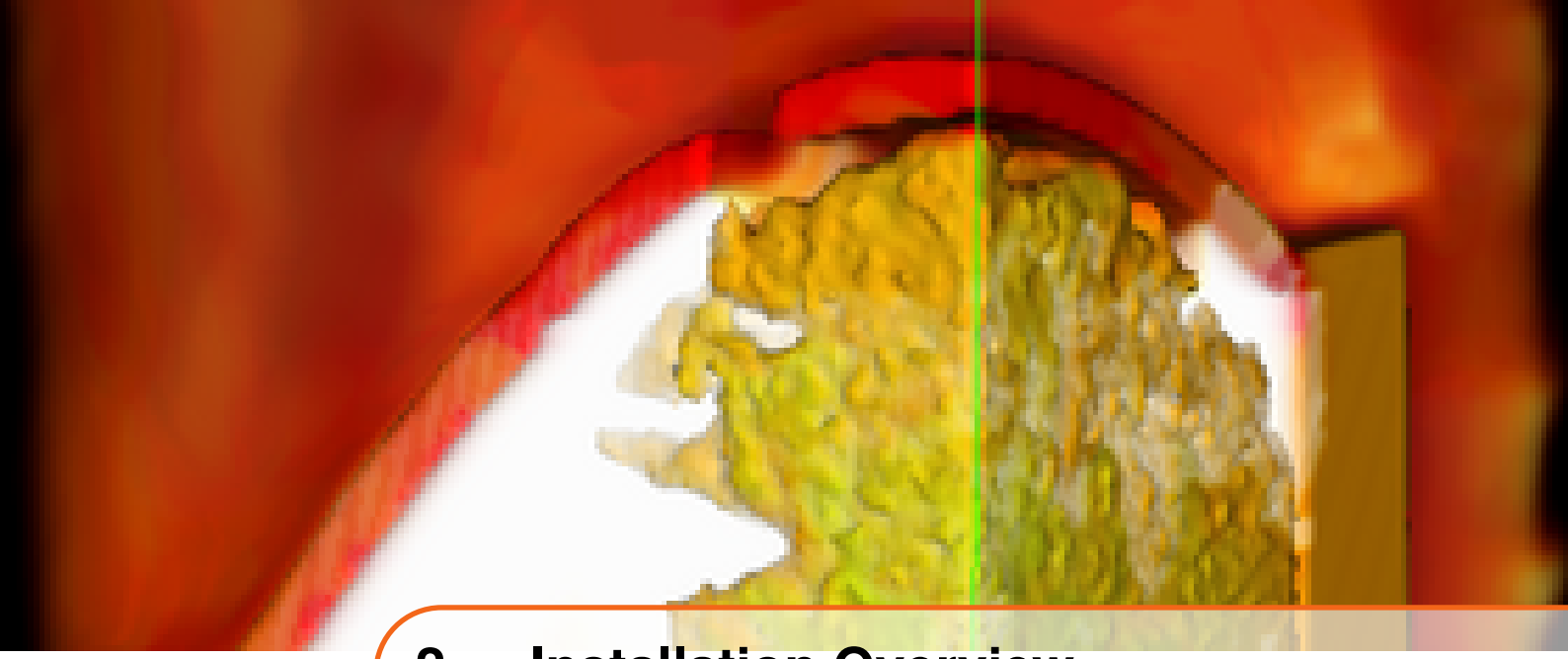
The **Uintah** library is composed of several executables and a generic library framework for solving PDEs on structured AMR grids on large parallel supercomputers. ***VisIt*** is used to visualize data generated from **Uintah**.

1.1 Obtaining the Code

Uintah can be obtained either as a compressed tarball from http://www.uintah.utah.edu/trac/chrome/site/uintah.tar.gz or by using Subversion (SVN) to download the latest source code:

```
svn co https://gforge.sci.utah.edu/svn/uintah/trunk uintah
```

The above command checks out the **Uintah** source tree and installs it into a directory called uintah in the users home directory.



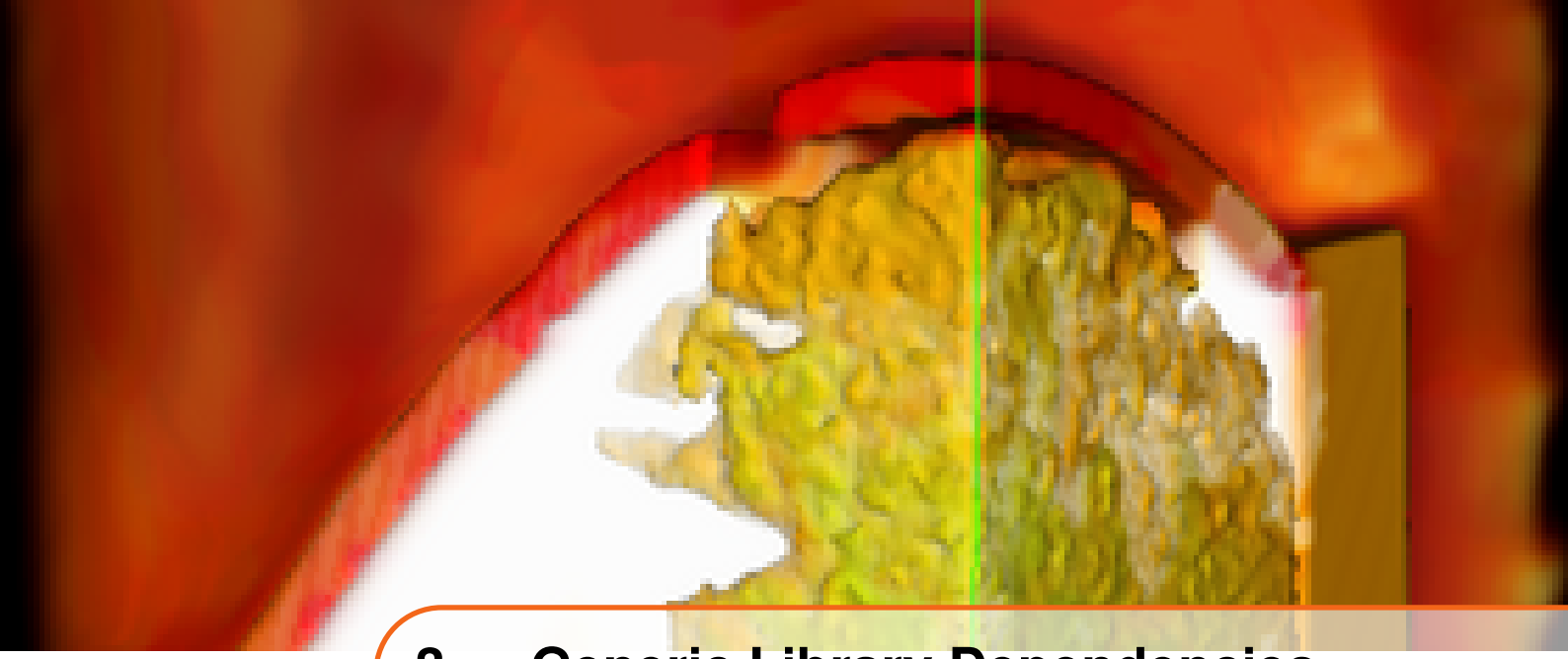
2 — Installation Overview

Uintah can be installed individually and in conjunction with the visualization tool, ***VisIt***. It is recommended for first time users to install **Uintah** first and then install ***VisIt*** second. Once you have a working build of **Uintah** then go ahead and install ***VisIt***. The configure script used by **Uintah** will have to be slightly modified to include the location of the ***VisIt*** installation directory.

However, if **Uintah** is installed on a supercomputer, ***VisIt*** does not need to be installed prior to installation of **Uintah**.

The installation procedure follows the basic outline:

1. Install basic dependencies: MPI, Blas, Lapack, Hypre, PETSc, *etc*.
2. Configure Uintah
3. Compile Uintah
4. Install visualization tools (optional if installing on a supercomputer or first time users) and reconfigure Uintah.



3 — Generic Library Dependencies

Uintah depends on several different libraries that are commonly available or easily installable on various Linux or Unix like OS distributions.

Required libraries:

- mpi (OpenMpi, Mpich, or LAM or a vendor supplied mpi library)
- blas
- lapack
- make
- libxml2-devel
- zlib-devel
- c++
- fortran
- subversion
- cmake
- git

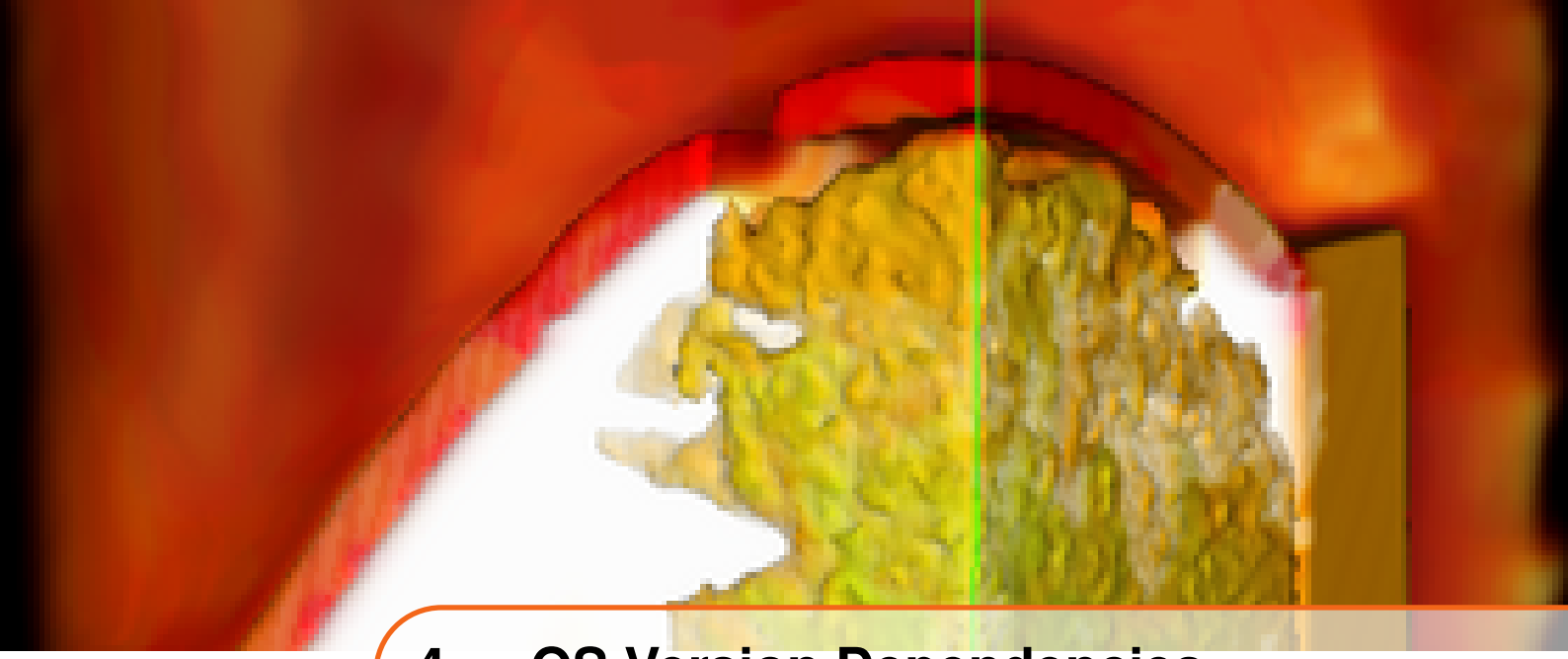
Useful libraries:

- Hypre v2.8.0b (https://computation.llnl.gov/casc/hypre/software.html)
- PETSc v3.3 (http://www.mcs.anl.gov/petsc/petsc-as/download/index.html)

Note, we suggest using Hypre v2.8.0b and PETSc v3.3-p3.

The Wasatch component requires the following:

- git
- boost-1.44 or higher
- cmake 2.8 or above



4 — OS Version Dependencies

4.1 Debian & Ubuntu Dependencies

The **Gnu/Debian** OS http://www.debian.org and **Ubuntu** OS http://www.ubuntu.org offer the vast majority of libraries necessary for installing **Uintah** with the exception of VisIt and Teem. Either one of these distributions are the recommended OS to install.

Installing the following libraries will ensure that all dependencies required for **Uintah** are satisfied:

```
sudo aptitude install subversion libhypre-dev petsc-dev \
libxml2-dev zlib1g-dev liblapack-dev cmake libglew1.5-dev \
libglui-dev libxmu-dev g++ gfortran libboost-all-dev git \
libxrender-dev libxi-dev
```

4.2 Fedora Core Dependencies

Fedora Core http://fedoraproject.org/ offers all the dependencies except for PETSc and HYPRE. Install the following libraries:

```
sudo yum install make openmpi-devel openmpi lapack-devel \
gcc-gfortran blas-devel gcc-c++ libxml2-devel subversion \
cmake tar diffutils libX11-devel libXext-devel patch wget \
mesa-libGL-devel mesa-libGLU-devel libXt-devel git boost-devel \
boost-static libXrender-devel libXi-devel
```

4.3 CentOS Dependencies

CentOS http://www.centos.org/ is similar to **Fedora Core**

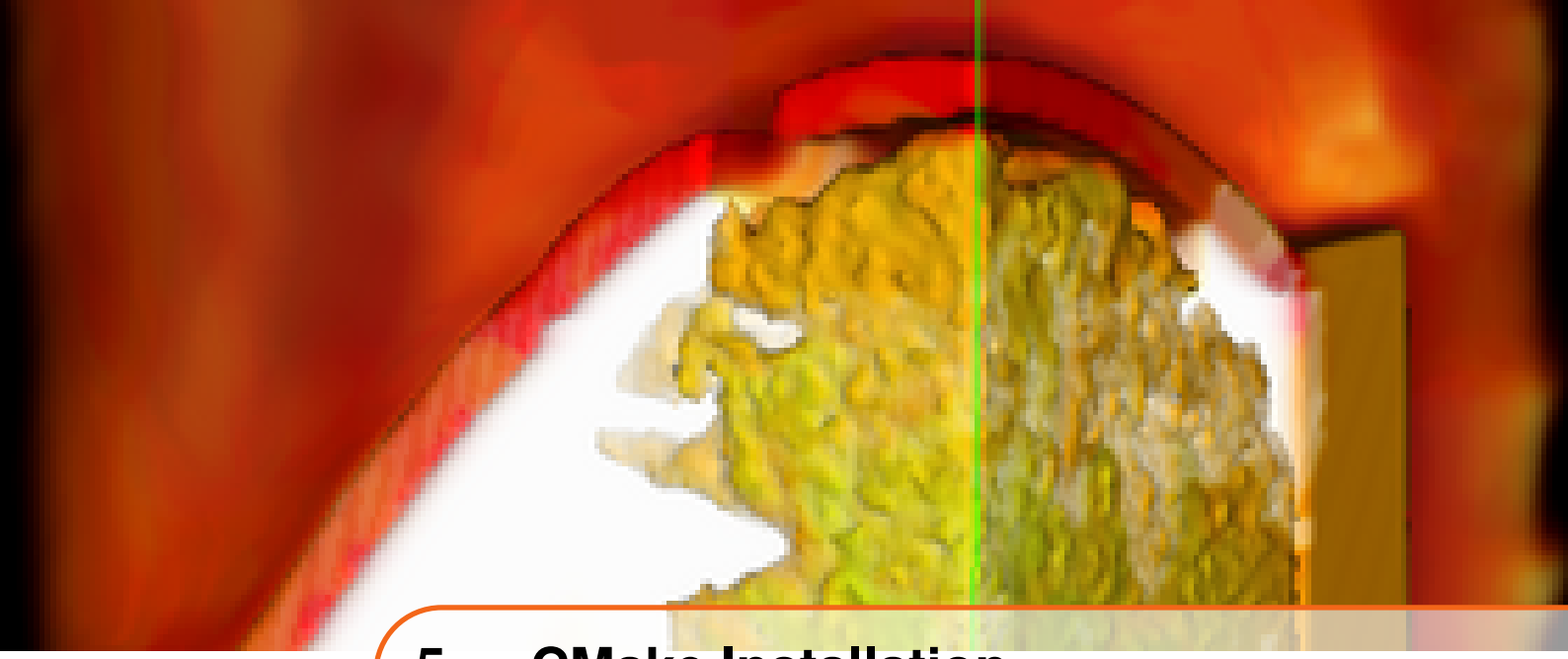
```
sudo yum install make openmpi-devel openmpi lapack-devel \
gcc-gfortran blas-devel gcc-c++ libxml2-devel subversion \
tar diffutils libX11-devel libXext-devel patch wget \
mesa-libGL-devel mesa-libGLU-devel libXt-devel git \
libXrender-devel libXi-devel
```

Note: The packages for boost and cmake are too old and the user must download and build more recent versions, see below for instructions.

4.4 OpenSuse Dependencies

OpenSuse http://www.opensuse.org/ is similar to **Fedora Core**

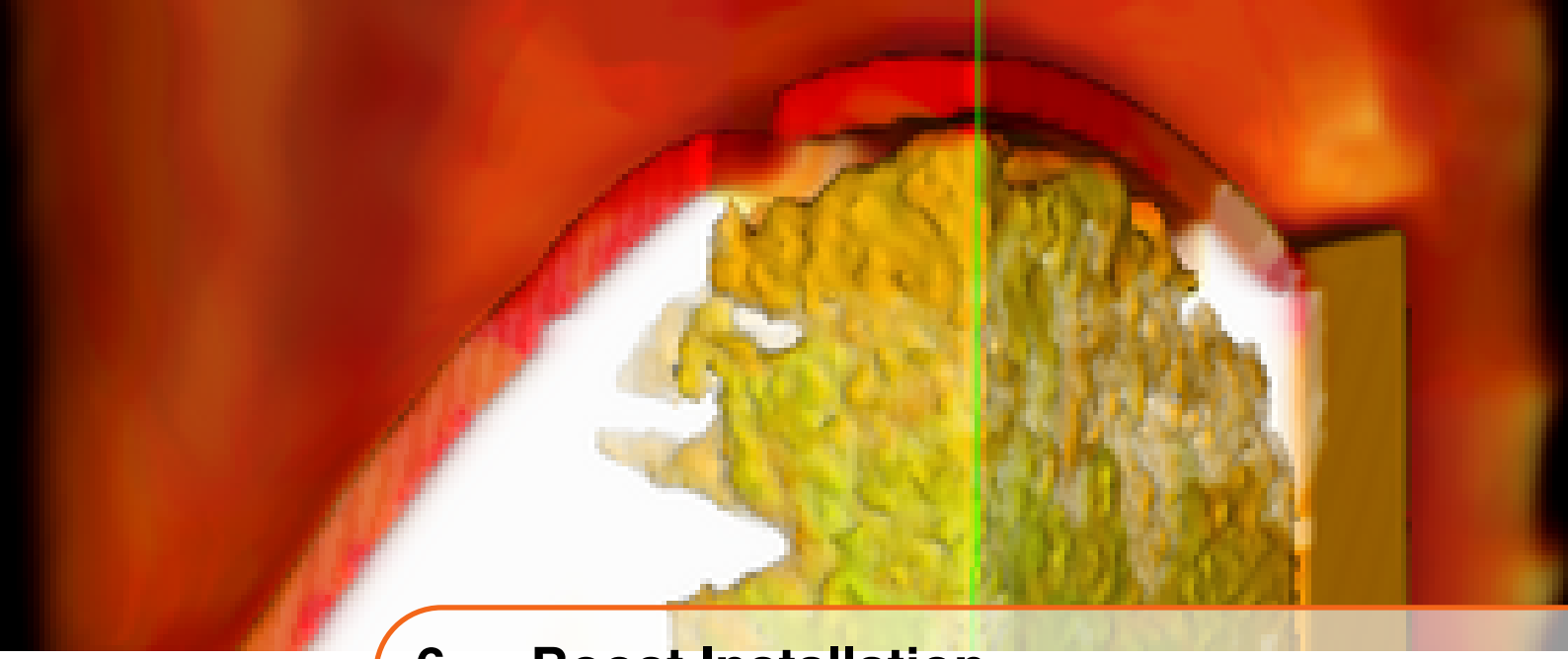
```
sudo zypper install make openmpi-devel openmpi lapack \
gcc-fortran blas gcc-c++ libxml2-devel subversion \
cmake tar diffutils libX11-devel libXext-devel patch wget \
mesa-libGL-devel mesa-libGLU-devel libXt-devel git boost-devel \
boost-static libXrender-devel libXi-devel
```



5 — CMake Installation

A condensed set of instructions for building **CMake** http://www.cmake.org/ is provided. The cmake binary and resulting libraries are installed in the `/usr/local` hierarchy.

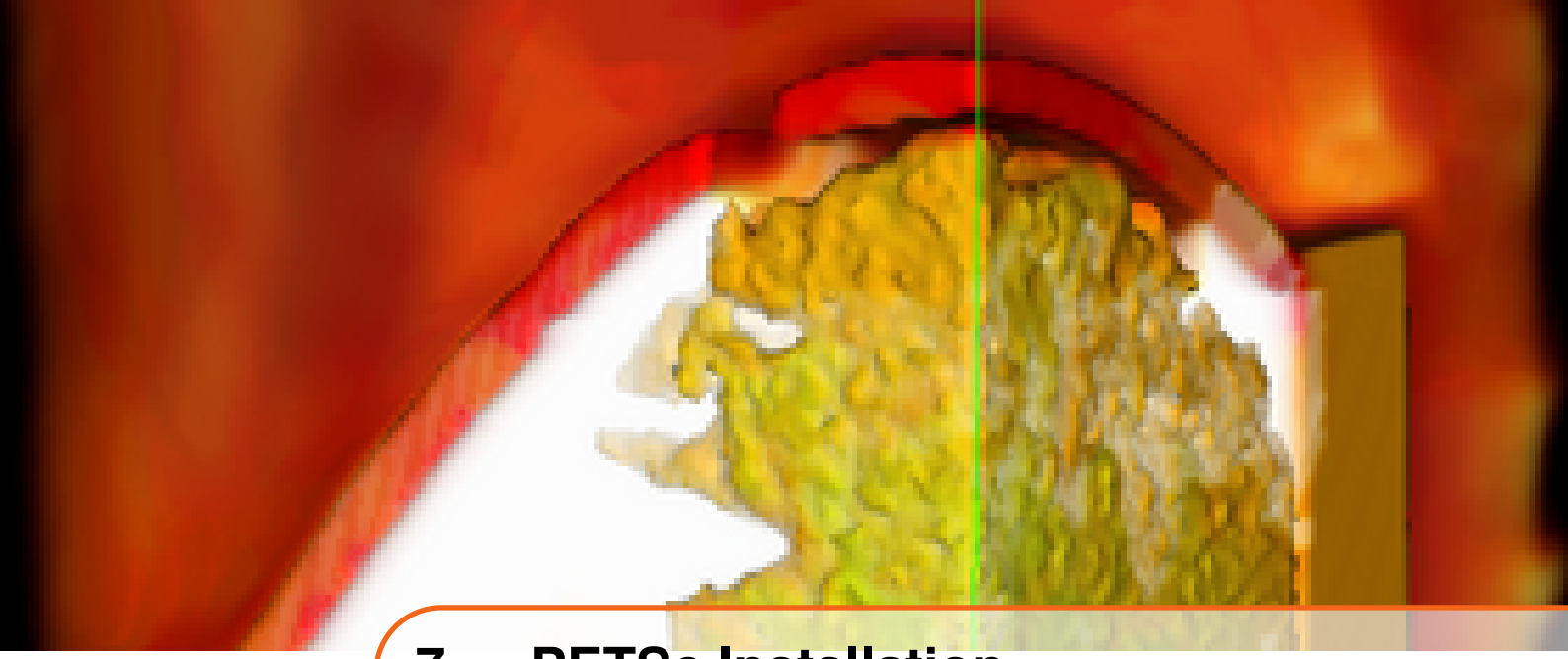
```
 tar xf cmake-2.8.9.tar.gz
 cd cmake-2.8.9
./bootstrap --prefix=/usr/local
make
sudo make install
```



6 — Boost Installation

A condensed set of instructions for building the **Boost Libraries** http://www.boost.org/ is provided. The libraries are installed in the user's home directory.

```
tar xvf boost_1_51_0.tar.bz2
cd boost_1_51_0
./bootstrap.sh --prefix=~/boost-1.51
./b2
./b2 install
```



7 — PETSc Installation

PETSc can be installed by executing the following simplified instructions. Please refer to the PETSc website for comprehensive installation instructions if you encounter any difficulties.

Download petsc-v3.3-p3 from http://www.mcs.anl.gov/petsc/petsc-as/download/index.html

7.1 Fedora & CentOS

The following configure script assumes the location of MPI libaries are in `/usr/lib64/openmpi/lib`. Note that the `LD_LIBRARY_PATH` must be set. It is helpful to include the `/usr/lib/64/openmpi/bin` in your path for the location of mpirun and mpiCC, mpicc, and mpif77.

```
tar zxf petsc-3.3-p3.tar.gz
cd petsc-3.3-p3

LD_LIBRARY_PATH=/usr/lib64/openmpi/lib \
./configure --with-shared-libraries \
--with-debugging=0 \
--with-mpi-dir=/usr/lib64/openmpi  \
--prefix=/home/jas/petsc

make PETSC_DIR=/home/jas/petsc-3.3-p3 PETSC_ARCH=arch-linux2-c-opt all

make PETSC_DIR=/home/jas/petsc-3.3-p3 PETSC_ARCH=arch-linux2-c-opt install
```

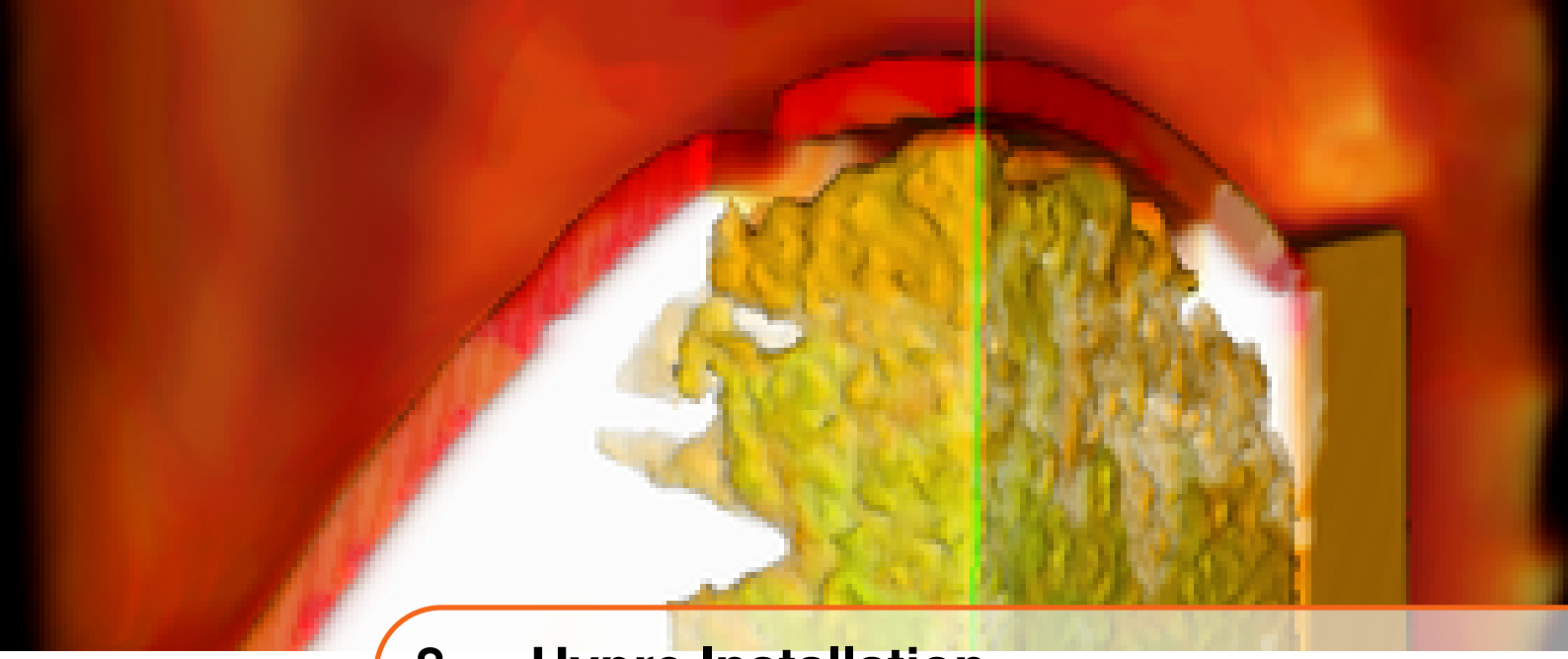
7.2 OpenSuse

```
tar zxf petsc-3.3-p3.tar.gz
cd petsc-3.3-p3

./configure --with-mpi-dir=/usr/lib64/mpi/gcc/openmpi \
--with-shared-libraries --with-debugging=0 --prefix=/home/jas/petsc
```

```
make PETSC_DIR=/home/jas/petsc-3.3-p3 PETSC_ARCH=arch-linux2-c-opt all

make PETSC_DIR=/home/jas/petsc-3.3-p3 PETSC_ARCH=arch-linux2-c-opt install
```



8 — Hypre Installation

Hypre can be installed by executing the following simplified instructions, however, if you encounter problems please refer to the hypre website for troubleshooting.

Download hypre-2.8.0b from https://computation.llnl.gov/casc/hypre/software.html

8.1 Fedora & CentOS

The following configure script assumes the location of MPI libaries are in `/usr/lib64/openmpi/lib`. Note that the `LD_LIBRARY_PATH` must be set. It is helpful to include the `/usr/lib/64/openmpi/bin` in your path for the location of mpirun and mpiCC, mpicc, and mpif77.

```
cd hypre-2.8.0b/src

LD_LIBRARY_PATH=/usr/lib64/openmpi/lib/ \
 ./configure --enable-shared \
 --with-MPI-lib-dirs=/usr/lib64/openmpi/lib/ \
--with-MPI-include=/usr/include/openmpi-x86_64/ \
--with-MPI-libs="mpi mpi_cxx" \
CC=/usr/lib64/openmpi/bin/mpicc \
 CXX=/usr/lib64/openmpi/bin/mpic++ \
 F77=/usr/lib64/openmpi/bin/mpif77

make
```

8.2 OpenSuse

```
cd hypre-2.8.0b/src

./configure --enable-shared \
  --with-MPI-libs="mpi mpi_cxx" \
  CC=/usr/lib64/mpi/gcc/openmpi/bin/mpicc \
  CXX=/usr/lib64/mpi/gcc/openmpi/bin/mpic++ \
```

```
  F77=/usr/lib64/mpi/gcc/openmpi/bin/mpif77

make
```

8.3 For OSX (Macs)

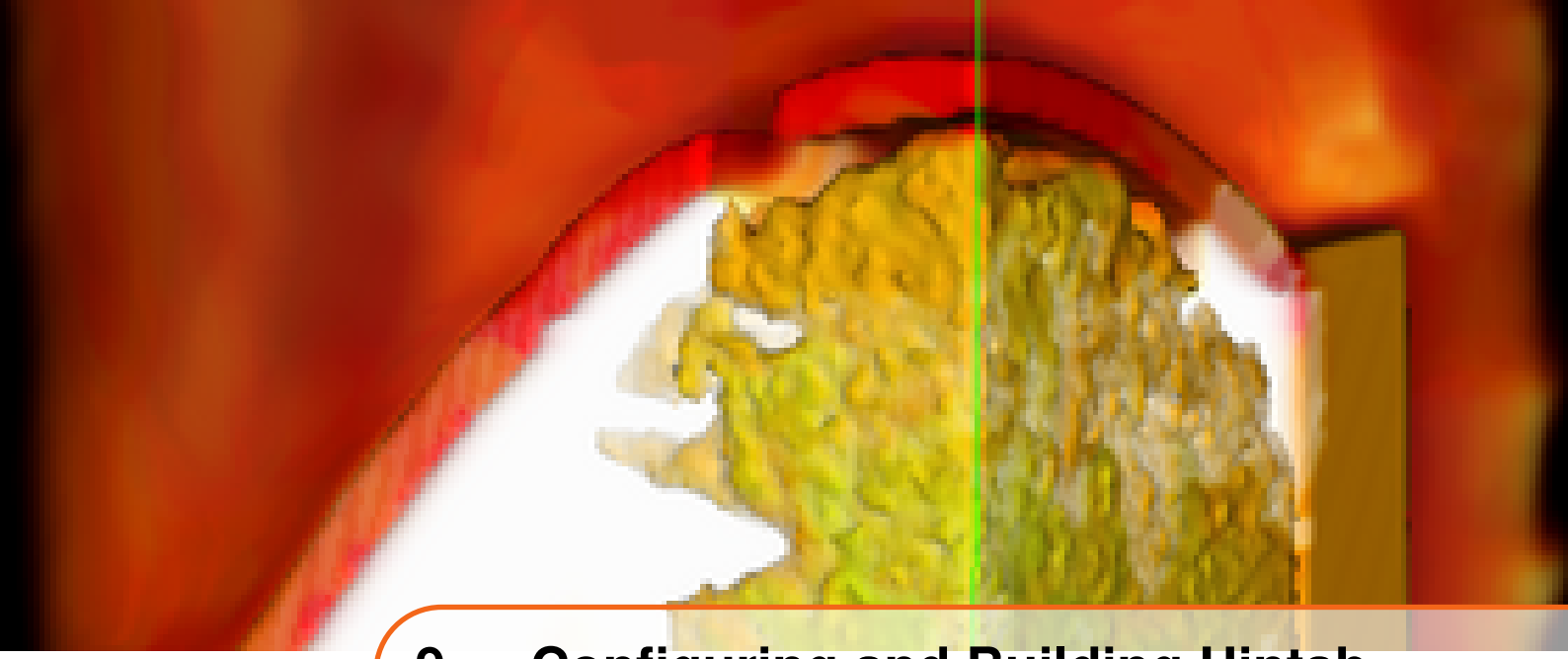
```
cd hypre-2.8.0b/src

./configure \
  --prefix=/usr/local \
  CC=/usr/bin/gcc \
  CXX=/usr/bin/g++ \
  CFLAGS=-DMPIPP_H CXXFLAGS=-DMPIPP_H \
  F77=/usr/local/bin/gfortran \
  FLIBS="-lcrt1.10.6.o"
```

(Note, under OSX, you will not be builing a shared library version of Hypre. Also, the 'FLIBS' line is required for use with some gfortran compilers as they try to use crt1.10.5.o (which causes duplicate symbol errors).)

```
make
make install
```

Note, under Mac OSX, Hypre does not handle the "–enable-shared" configure flag correctly.



9 — Configuring and Building Uintah

9.1 Configure

The following information specifies how to run the "configure" script to prepare for building the Uintah executable. "Configure" is a program that checks the computer's environment and looks for the location of libraries and header files, etc. When it finishes running, it will create the `configVars.mk` file which will include all the information it has determined about the computer. The following lines show **generically** how to use "configure".

Goto the directory `~/uintah` and create the following directories: `dbg` (debug) and `opt` (optimized)

```
cd ~/uintah-1.5.0
mkdir dbg opt
```

Go to the `dbg` directory and enter the following:

```
../src/configure --enable-debug
```

After the configuration process, type `make` to compile Uintah.

If you wish to attach a debugger to Uintah, make sure you use the `-enable-debug` configure option.

If you would like to reduce the amount of output given by the make system when making Uintah, you can set the environment variable `SCI_MAKE_BE_QUIET` to true:

```
setenv SCI_MAKE_BE_QUIET true
```

Once a debug build has finished, change to the `opt` directory and enter the following configure line:

```
../src/configure '--enable-optimize=-O3 -mfpmath=sse'
```

The debug/optimize flags do the following: –enable-debug adds "-g" to the compile line; –enable-optimize, if used without parameters, adds "-O2" to the compile line. Otherwise it adds the parameters specified during configure.

9.1.1 Other configure options

Other useful configure options are seen below. Note, depending on where libraries are installed on your system, you may or may not need to specify some (or all) of these options:

```
--with-mpi=/usr/lib64/openmpi <= Specifies location of MPI
--enable-64bit                <= Specifies a 32 or 64bit build
--enable-32bit
--with-petsc=/path/petsc-2.3.3-p13 <= Specifies location of PETSc
--with-hypre=/path/hypre-2.0.0/hypre_install  <= Specifies location of Hypre

F77=gfortran-4.3               <= Specifies compilers (Fortran, C, C++).
CC=/usr/bin/gcc-4.3
CXX=/usr/bin/g++-4.3
PETSC_ARCH=linux-gnu-c-real-opt <= Specifies PETSc architecture

        --without-fortran     <= Turns off the use of Fortran (and the
            Arches/Radiation component.)

        USE_ARCHES=no         <= Turn off the building of specific Uintah
            components.
        USE_ICE=no               (Note, turning off Arches also turns off
            SpatialOps.)
        USE_MPM=no
        USE_RADIATION=no
```

To compile the Wasatch component add:

```
--with-boost=<path to boost install>
--enable-wasatch_3p
```

to the configure line.

To see a full list of all the configure options type:

```
../src/configure --help
```

9.1.2 Common Configure Lines

Debian & Ubuntu

```
../src/configure --enable-debug --with-boost=/usr --enable-wasatch_3p
```

Fedora-17 & CentOS-6.3

```
../src/configure --enable-debug --with-mpi-lib=/usr/lib64/o
penmpi/lib --with-mpi-include=/usr/include/openmpi-x86_64 --with-boost=/usr
    --en
able-wasatch_3p --with-hypre=/home/jas/hypre2.8.0b/src/hypre --with-petsc=/
    home/
jas/petsc
```

OpenSuse-12.2

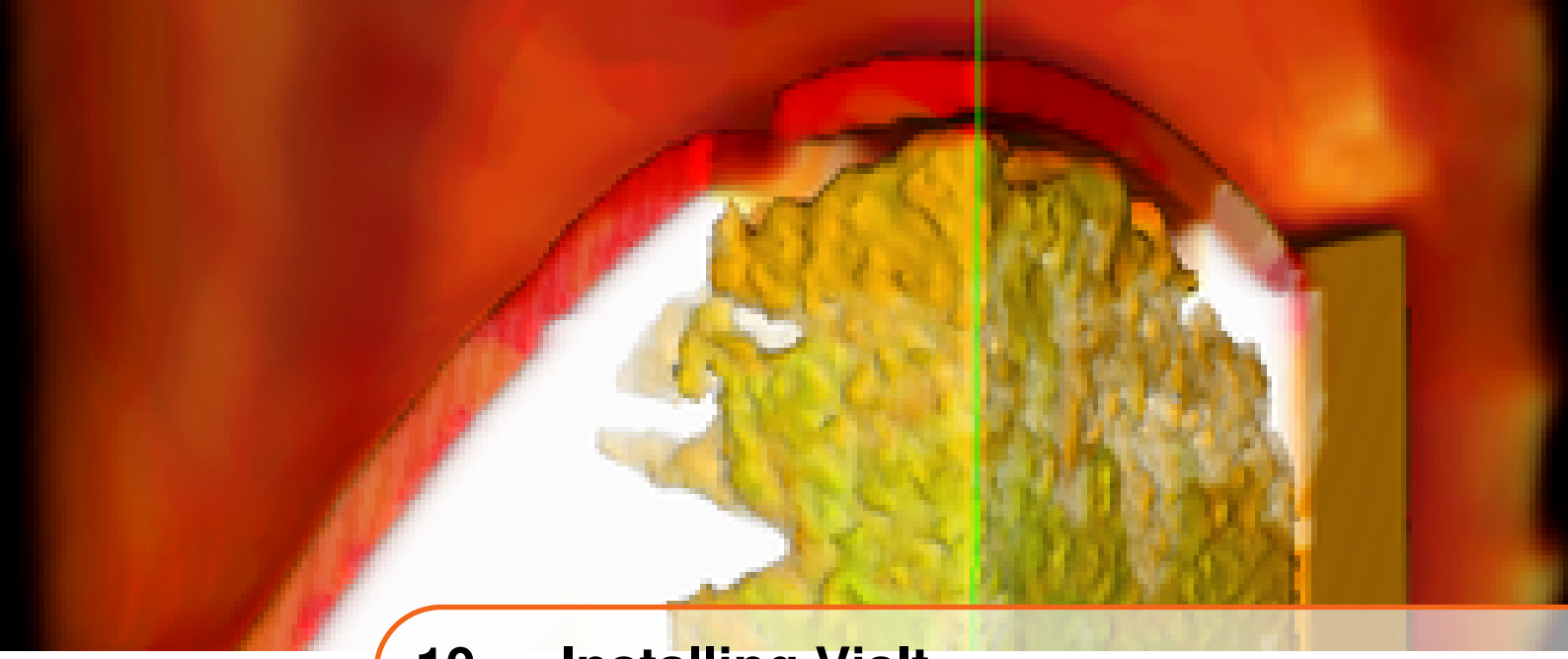
```
../src/configure --enable-debug --with-mpi-lib=/usr/lib64/o
penmpi/lib --with-mpi-include=/usr/include/openmpi-x86_64 --with-boost=/usr
    --en
able-wasatch_3p --with-hypre=/home/jas/hypre2.8.0b/src/hypre --with-petsc=/
    home/
jas/petsc
```

9.2 Build

Then build the software by typing `make` at the command line and create symbolic links to the inputs directory found in `~/uintah/src/StandAlone/inputs`.

```
make
```

Please see the **User Guide** http://www.uintah.utah.edu/trac/chrome/site/uintah.pdf on how to use the various tools and executables within the **Uintah** framework.



10 — Installing VisIt

Visualization of **Uintah** data is currently possible using ***VisIt***. The VisIt package from LLNL is general purpose visualization software that offers all of the usual capabilities for rendering scientific data. It is still developed and maintained by LLNL staff, and its interface to **Uintah** data is supported by the Uintah team.

10.1 VisIt Installation

The latest official release of ***VisIt*** is 2.5.2. However, the instructions apply to earlier versions such as 2.5.X. The UDA reader plugin must be built against an installed version of ***VisIt***. ***VisIt*** can be installed either from one of their binary distributions, or by compiling and installing it from source. You may also need to build ***VisIt*** from source if a binary is not provided for your exact OS and MPI versions and is the recommended installation method outlined below.

10.1.1 Caveats

- Before you install ***VisIt***, a number of dependency files are required. These include X11, GL, GLU, GLX, and libz libraries (it is possible that you will find these libraries in the following rpms: libx11-dev, libxext-dev, libxt-dev, libglu, zlib-dev – notice that most of these are the 'development' rpm). You can either install these directly and hope you get them right, or you can install the files listed here 4.1 (which will hopefully pull in all the libraries that you need).
- MPI must be installed and the MPI compilers (mpicc, etc) must be in your path.
- The ***VisIt*** executable is not relocatable once it is built, so make sure you specify the correct installation location in the beginning.
- Building ***VisIt*** from scratch can take a while. As mentioned below, you can

  ```
  build_visit_log
  ```

  to watch the progress.
- If you are having problems please check the Uintah wiki page on building ***VisIt*** (http://www.uintah.utah.edu/trac/wiki/visit).

10.1.2 Using the "build_visit" Script to Build VisIt

1. Download the build_visit script from http://portal.nersc.gov/svn/visit/trunk/releases/2.5.2/build_visit2_5_2.
2. Change the mode of the build_visit script so it can execute, and copy it to a VISIT directory (calling it Visit_2.3.2 in the following description) where you will build the VisIT source code:

```
mkdir ~/Visit_2.5.2
cp build_visit_2.5.2 Visit_2.5.2
cd Visit_2.5.2
chmod +x build_visit_2.5.2
```

3. Build the VisIT source (after it downloads everything) using two processors (use the number of available processors on your machine). This will take a long time since all the prerequisite software will be downloaded and compiled. Select the 'parallel' option when the script starts up.

```
./build_visit2_5_2 --makeflags -j2 --no-icet --console
```

NOTE: If you run into problems building QT, you can try setting the environment variable LD_LIBRARY_PATH to the following:
for csh users:

```
setenv LD_LIBRARY_PATH complete_path_to/qt-everywhere-opensource-src
    -4.6.1/lib
```

or
for bash users:

```
export
LD_LIBRARY_PATH=complete_path_to/qt-everywhere-opensource-src-4.6.1/lib
```

where complete_path_to is the full path to the qt-everywhere-opensource-src-4.6.1/lib found in the Visit directory.
Once this is set, you can start the build_visit script again and it should pick up where it left off.

4. After VisIT finishes building, go into the visit2.3.2/src directory and make a binary distribution for it so it can be installed.

```
cd visit2.5.2/src
 make package
```

Within the visit2.5.2/src directory there should be a gzip'ed tar file that can be installed in different location (VisIT must be installed in order to compile and build the UDA reader plugin to visualize Uintah data), i.e

```
ls visit2.5.2/src
 . . .
 visit2_5_2.linux-x86_64.tar.gz
 . . .
```

5. Copy the gzip'd tar file to another location and install VisIT (using the visit-install script found in Visit_2.5.2/visit2.5.2/src/svn_bin/visit-install) in the directory *visit_bin_dir*.

```
cp visit2_5_2.linux-x86_64.tar.gz ~/
cd ~/
~/Visit_2.5.2/visit2.5.2/src/svn_bin/visit-install 2.5.2 linux-x86_64
visit_bin_dir
```

6. Use the following '–with-visit' configuration line when configuring uintah. Use the full path name for the location of the *visit_bin_dir* directory.

```
--with-visit=/full_path/visit_bin_dir/
```

10.1.3 Parallel OS X build

A full parallel OS X build can be built using the instructions above, but it will fail when it tries to build the visit executable (the last stage of the script). When it fails, you will need to follow the instructions for building in parallel on a Mac.

http://visitusers.org/index.php?title=Building_in_parallel_on_a_Mac

Note particularly the changes that need to be made in the file,

```
<path\_to\_visit\_src>/config-site/<your_machine_name>.cmake
```

Once you make these edits, you can then finish building VisIt by typing make in the `<path_to_visit>/src` directory.

10.1.4 When the build script fails

Sometimes the build script doesn't work depending on your installation. When this happens, you may be forced to build everything by hand. The instructions for doing this are carefully outlined in the VisIt Build Notes file,

```
https://wci.llnl.gov/codes/visit/2.5.2/BUILD_NOTES
```

10.1.5 VisIt executable

After a successful build, the executable will be placed in,

```
``visit_bin_dir''/bin/
```

where "visit_bin_dir" is the directory specified either for the visit_install script if using a prebuilt binary, or for cmake if building VisIt from source.

10.1.6 Building and installing the UDA reader plugin

Add the following arguments to the **Uintah** configure line,

```
--with-visit=/path/to/visit
```

For example,

```
--with-visit=/home/csafe/dav/VisIt/blaze/visit \
```

Note, the `visit` directory you point at should have `data`, and `bin` sub-directories, and a sub-directory for each version of visit installed. This is the installation directory, not the source directory.

10.1.7 How it works

By specifying the above configure arguments (to the **Uintah** configure), **Uintah** will automatically build a VisIt plugin (when you type "make"). The plugin will be placed

in (a sub-directory of) ~/.visit. When VisIt is run, it will find the plugin and allow for reading UDA's.

10.1.8 Remote visualization

In order to read data on a remote machine, you will need to have a version of VisIt and the UDA reader plugin installed on that machine. You will also need to make sure that the VisIt executable is in your path. You may need to edit your shell's .rc file so that the path to VisIt is included in your shell's PATH variable.

NOTE: The version of VisIt installed on your local desktop and the remote machine should be the same.

10.1.9 Host Profile

Next you will want to set up a Host Profile for your remote machine. Select 'Host Profiles' from the 'Options' menu and set up a new profile as shown in figure 10.1.

After filling in the remote machine information, select the 'Advanced options' tab, then 'Networking' and check the 'Tunnel data connections through SSH' option. This is illustrated in figure 10.2a. Click on 'Apply' and then do a 'Save Settings' in the 'Options' menu on the gui.

For the remote visualization option to work, you must have ports 5600 - 5609 open. You can try this by running the following on your local desktop (on Linux distributions),

```
traceroute
 -p 560[0-9] <remote machine>
```

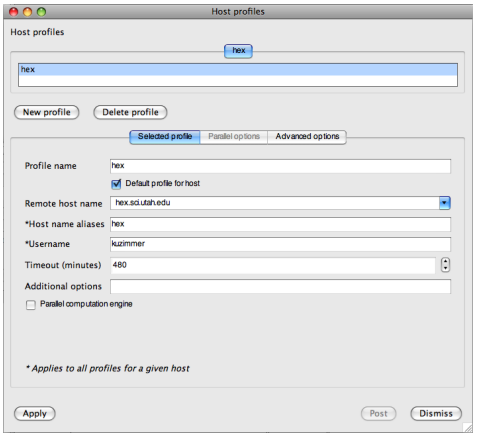
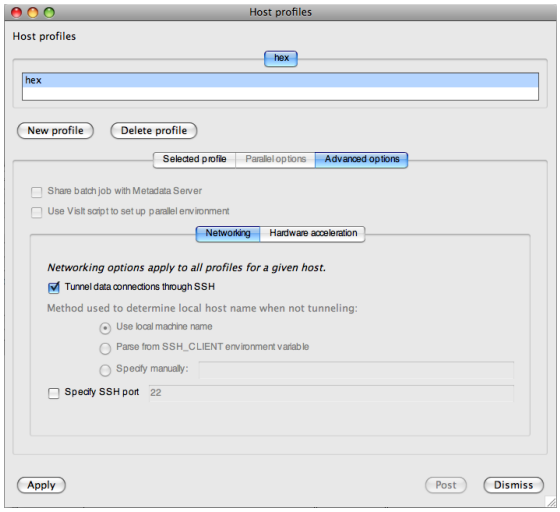


Figure 10.1: Setting up Host Profile

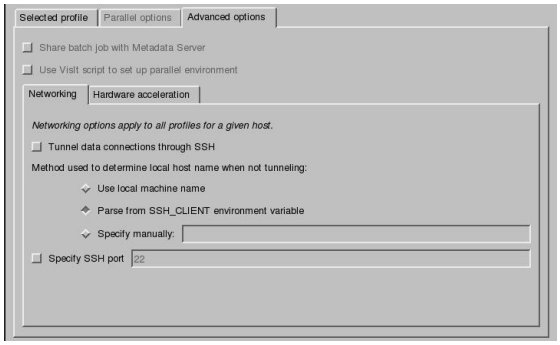
If the tunneling option doesn't works, try the option as shown in figure 10.2b.

Now you should be able select 'Open' from VisIt's 'File' menu. After selecting your host from the host entry list, you will be prompted for a password on the remote machine (unless you have set up passwordless ssh access).

Once the ssh login has completed, you should see the directory listing. You can then change directories to your UDA and load the data.

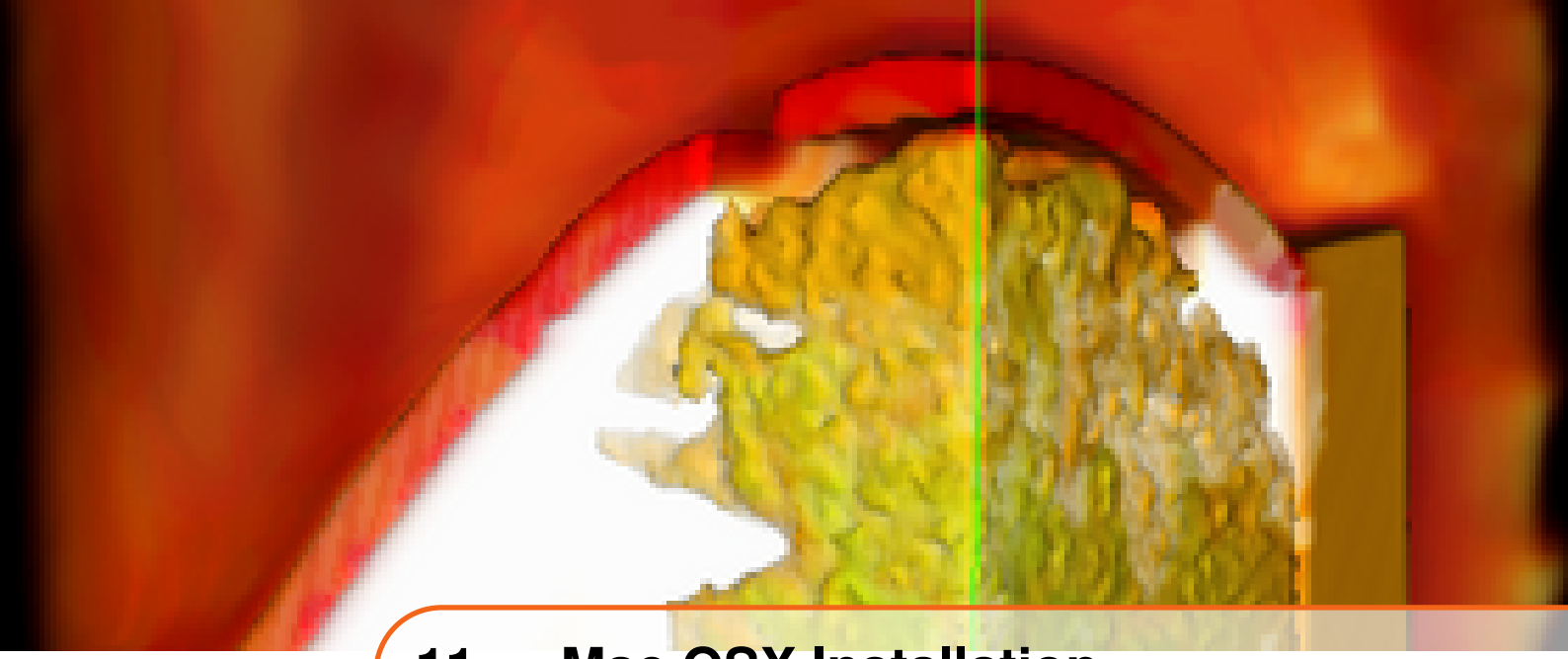


(a) Setting up advance options



(b) Setting up advance options

Figure 10.2



11 — Mac OSX Installation

Following are step-by-step instructions for installing the **Uintah Computational Framework** on **Mac OSX**. It is assumed that all commands are run as administrator (which may require a sudo). These instructions were tested on OSX 10.5.6 and 10.5.7 but should work on previous versions. **If you have Snow Leopard, go to Section 10.**

11.1 Installing Developer Tools

Install the suite of developer tools that are packaged with Mac OS X. These can be found in the "Optional Installs" directory on the Mac OS X Installation CD for newer versions (i.e. Tiger, Leopard) or on the included Developer CD if your version of OSX is older). If you don't have the installation CD you can download older versions from: http://connect.apple.com

Double-click the XCodeTools.mpkg and follow the installer instructions. This will install GCC, as well as many other useful developer tools.

11.2 Installing Fortran Compiler

Download gfortran from http://hpc.sourceforge.net/ making sure the package is designed for your CPU architecture–Intel or PowerPC. Put package in root directory and extract. Extracting from / puts everything in the correct places. Extract using:

```
tar -xvf gfortran-*-bin.tar.gz
```

Note: If the Uintah builds correctly, but you are receiving link errors, see section 10.2.

11.3 Installing Subversion/libpng3/libxml2/other dependencies

As mentioned above, several libraries are required for Uintah, and may need to be updated or installed depending on the version of Mac running. Some of these include

SVN, libpng3, cmake, zlib and libxml2, which are all conveniently part of the fink project.

Download the appropriate version of fink for your version of OSX from http://www.finkproject.org/download/srcdist.php. Where "x" and "y" are version numbers, extract and install with:

```
tar -xvf fink-0.x.y-full.tar
cd fink-0.x.y-full
./bootstrap
pathsetup.sh
```

For a GUI version of fink, download Fink Commander and install from disk image from http://finkcommander.sourceforge.net/ following instructions in the installer. The Fink Commander website stresses that you install Commander after fink.

Finally, update or install any necessary packages using Fink Commander's search bar.

11.4 Install/Update X11

X11 can be found in the "Optional Installs" directory accompanying your Mac OS Installation CD. This copy should be updated. Alternately, X11 can be downloaded from the web and installed. Current versions can be found at http://xquartz.macosforge.org/trac/wiki/Releases. Find a package that matches your version of OSX (or update OSX to newest version), and install it. Installation is as simple as downloading the disk image from this site, mounting it, double-clicking the installer application, and following instructions.

11.5 Installing an LAM MPI

We recommend that you avoid using fink to install LAM MPI. The files and libraries are installed in non-traditional locations and Uintah cannot find them. (11/10/2010)

Download LAM MPI from http://www.lam-mpi.org version 7.1.4 or newer. Make a new directory in ~/ for this and all of the following source packages. This documentation assumes this directory is ~/Uintah. Inside this directory make another for installation directories. This documentation assumes this directory is ~/Uintah/installs

```
cd ~/Uintah/installs
tar -xvf lam-7.1.4.tar.gz
cd lam-7.1.4
./configure --prefix=/Users/<username>/Uintah/installs/lam-7.1.4 \
            --enable-shared=yes --with-fc=gfortran
make
make install
```

These last two commands will build LAM in the directory to which –prefix= points.

11.6 Installing PETSc

PETSc and Hypre (the next two packages to install) are used by Uintah for their PDE and linear algebra and capabilities respectively. Download PETSc from http://www-unix.

mcs.anl.gov/petsc/petsc-as/download/index.html. Unarchive it in ~/Uintah/installs directory using a similar tar command as is found above. Configure using:

```
./config/configure.py \
   --prefix=/Users/<username>/Uintah/installs/petsc-2.3.3-p13 \
   --with-matlab=false \
   --with-x=false \
   --with-shared=0 \
   --with-debugging=0 \
   --with-mpi-dir=/Users/<username>/Uintah/installs/lam-7.1.4
```

Set the environmental variables the configuring process asks you to set, and install using:

```
make
make install
```

11.7 Installing Hypre

Download Hypre version 2.x from https://computation.llnl.gov/casc/hypre/software.html to ~/Uintah/installs. Unarchive and install by running

```
cd hypre-2.x.x/src
./configure \
   --with-MPI-include=/Users/<username>/Uintah/installs/lam-7.1.4/include \
   --with-MPI-lib-dirs=/Users/<username>/Uintah/installs/lam-7.1.4/lib \
   --with-MPI-libs="mpi lam pmpi util" \
   --prefix=/Users/<username>/Uintah/installs/hypre-2.0.0/hypre_install \
   CFLAGS="-DMPIPP_H" \
   CXXFLAGS="-DMPIPP_H"
make
make install
```

Take down of your –prefix line for future reference, as this is where you will point Uintah's configure.

11.8 Installing Uintah

Download the Uintah source from our website using:

```
cd ~/
svn co https://gforge.sci.utah.edu/svn/uintah/trunk ~/Uintah
```

Uintah will not install correctly if configured from within the source directory. As such, it is advised to create a directory on the same level as /Uintah/src named mac[Numofbits]opt in which to configure and build.

```
cd Uintah
mkdir mac32opt
cd mac32opt
./configure \
   --enable-optimize \
   --with-mpi=/Users/<username>/Uintah/installs/lam-7.1.4 \
   --with-petsc=/Users/<username>/Uintah/installs/petsc-2.3.3-p13 \
   --with-hypre=/Users/<username>/Uintah/installs/hypre-2.0.0/hypre_install
       \
   PETSC_ARCH=darwin9.3.0-c-opt \
   F77=gfortran
```

The previous configure command is a template, and may need editing. Point each of the components to the proper places and it should configure just fine. If you want to build Uintah with support for VisIt, add –with-teem=/dir/to/teem/ and –with-visit=/dir/to/visit/ Finish the process off with:

```
make all
```

Some useful programs can be found in the installation: sus will be built and installed in ~/Uintah/mac32opt/StandAlone Extraction utilities that extract specific variables from UDA (Uintah Data Archive) can be found in ~/Uintah/mac32opt/StandAlone/tools

11.9 Installing VisIt and udaReader on Mac

A Mac binary distribution is available from LLNL at https://wci.llnl.gov/codes/visit/executables.html. The main difficulty in installing Uintah with udaReader plugin has to do with actually finding the VisIt installation executable. Furthermore, naming conventions are slightly different, and thus several links must be set up.

Extract the mac install and create links by (substitute i386 and VisIt version where necessary):

```
tar xvf visit1_11_2.darwin-i386.tar.gz
cd visit/1.11.2
export PATHTOVISIT=\$(pwd)
mkdir src
ln -s \$PATHTOVISIT/darwin-i386/bin src/bin
ln -s \$PATHTOVISIT/../bin/frontendlaucher src/bin/visit
```

Now that the links have been set up, configure Uintah with –with-visit=$PATHTOVISIT and –with-teem=DIR/TO/TEEM. See Section 8.1 for Teem installation.

An easier solution is to download the VisIt directory from the Uintah source, and run xml2makefile from $PATHTOVISIT/darwin-i386/bin on udareaderMTMD.xml within VisIt/udaReaderMTMD/udaREaderMTMD.xml such as:

```
xml2makefile -clobber -private VisIt/udaReaderMTMD/udaReaderMTMD.xml
```

To install VisIt from source, see build notes for Mac inside source tarball.

11.9.1 Installing VisIt and the udaReader on a Mac running OSX 10.6.1

The following steps should allow one to build VisIt and the Uintah VisIt uda reader plug-in on a Macintosh running OsX 10.6.1 (Snow Leopard):

1. Obtain the VisIt source from:

   ```
   svn co http://portal.nersc.gov/svn/visit/trunk/src visit/src

   \item \begin{lstlisting} mkdir third-party/
   ```

   (here we will compile all the third-party software needed by visit)
2. Manually download Python.2.6.4

   ```
   http://www.python.org/
   ```

   to third-party/
3. Manually download cmake 2.6.4

   ```
   http://www.cmake.org/
   ```

to third-party/

5. Manually download qt-mac-opensource-src-4.5.3

```
http://get.qt.nokia.com/qt/source/qt-mac-opensource-src-4.5.3.tar.gz
```

to third-party/

6. Manually download VTK 5.0.0d

```
http://portal.nersc.gov/svn/visit/trunk/third_party/
```

to third-party/

7. Inside of third-party,

```
cp ../src/svn-build/build_visit .
```

8. Execute

```
./build_visit --no-visit
```

9. Edit the host.conf generated by build_visit by adding:

```
-D__USE_ISOC99
```

to the CXXFLAGS variable.

10. Copy the host.conf in the third-party/ directory to ../src/config-site/

```
cp <some name>.conf ../src/config-site/

\item \begin{lstlisting}./configure
```

11.
```
\item When the make fails with an error of not finding -lGLU, open
    databases/Shapefile/Makefile and replace -lGLU with -framework
    OpenGL
\item Type make again. It should finish successfully and the VisIt
    executable will be in visit/src/bin
\item Add
\begin{lstlisting}
#define VERSION "2.1.0"
```

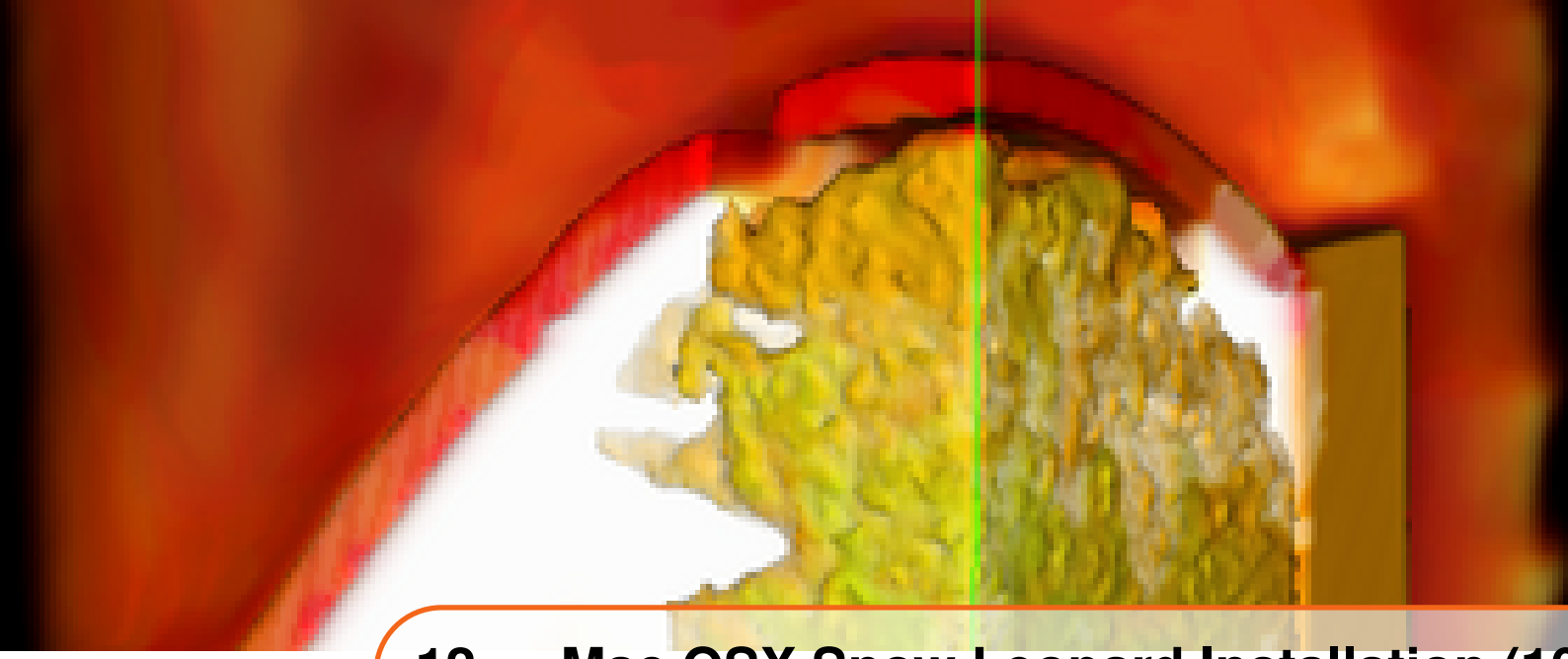
to src/include/visit-config.h

13. Configure Uintah with (for example):

```
--with-visit=/Users/<path to visit>  \
--with-teem=/Users/<path to SCI thirdparty>/install
```

14. Build Uintah

```
make -j#
```



12 — Mac OSX Snow Leopard Installation (10

What follows is a working configuration for installing Uintah on Mac OSX Snow Leopard (10.6.3) under 64 bit. In this example, it is assumed that Uintah is installed under

```
/Users/username/uintah
```

Also, all packages will be downloaded and installed under

```
/Users/username/uintah/packages
```

12.1 Fink

At the time of writing of this documentation, there is no binary version of the latest Fink distribution for Mac OSX. One must compile from the source by following the instructions on the website http://www.finkproject.org/download/srcdist.php. Make sure you select 64 bit mode when prompted. Keep all other options to default.

12.2 gfortran

Install the binary package for gfortran from: http://r.research.att.com/tools/. Then, set the path properly. If you are using bash, this is done as:

```
export PATH=/usr/local/bin:$PATH
```

Note: If the Uintah builds correctly, but you receive a dynamic link error from fortran when you try to load a uda, the links in fortran library may be pointing to the wrong place. This can easily be fixed by navigating to the fortran library and using "otool -L" to recognize the problem, then using install_name_tool to fix the problem. Both are included in the mac developer tools.

12.3 openmpi

Download the latest version of openmpi from: http://www.open-mpi.org/. Then configure using:

```
cd ~/uintah/packages
tar -xvf openmpi-1.4.x
cd openmpi-1.4.x
./configure --prefix=/Users/username/uintah/packages/openmpi-1.4.x-
    install
\F77=gfortran CFLAGS=-m64 CXXFLAGS=-m64 FFLAGS=-m64
```

Then

```
make
make install
```

12.4 hypre

Download the latest version of Hypre from here: https://computation.llnl.gov/casc/hypre/software.html. Then, configure using:

```
cd ~/uintah/packages
tar -xvf hypre-2.6.0b
cd hypre-2.6.0b/src
./configure --prefix=/Users/username/uintah/packages/hypre-2.6.0b-
    install
                    \--with-MPI-include=/Users/username/uintah/packages
                        /openmpi-1.4.x-install/include
                    \--with-MPI-lib-dirs=/Users/username/uintah/
                        packages/openmpi-1.4.x-install/lib
                    \CC='gcc -arch x86_64' \CXX='g++ -arch x86_64' \F77
                        ='gfortran -arch x86_64'
```

Then

```
make
make install
```

12.5 PETSc

For the current release of Uintah, Petsc2.3.3 or Petsc2.2.0b has been found to work well. Download the source from here: http://www.mcs.anl.gov/petsc/petsc-as/download/index.html. Then browse to the packages directory and configure as follows:

```
cd ~/uintah/packages
tar -xvf petsc-2.x.y-p15.tar.gz
cd petsc-2.x.y-p15
./config/configure.py \--with-shared \--with-debugging=0
\--with-mpi-include=/Users/username/uintah/packages/openmpi-1.4.x-
    install/include
\--with-mpi-lib=/Users/username/uintah/packages/openmpi-1.4.x-install/
    lib/libmpi.dylib
\--without-fortran \--prefix=/Users/username/uintah/packages/petsc-2.x
    .y-p15-install
```

Finally,

```
PETSC_DIR=/Users/username/uintah/packages/petsc-2.x.y-p15-install;
    export PETSC_DIR
make all test
make install
```

12.6 Uintah

Finally, create a directory in uintah

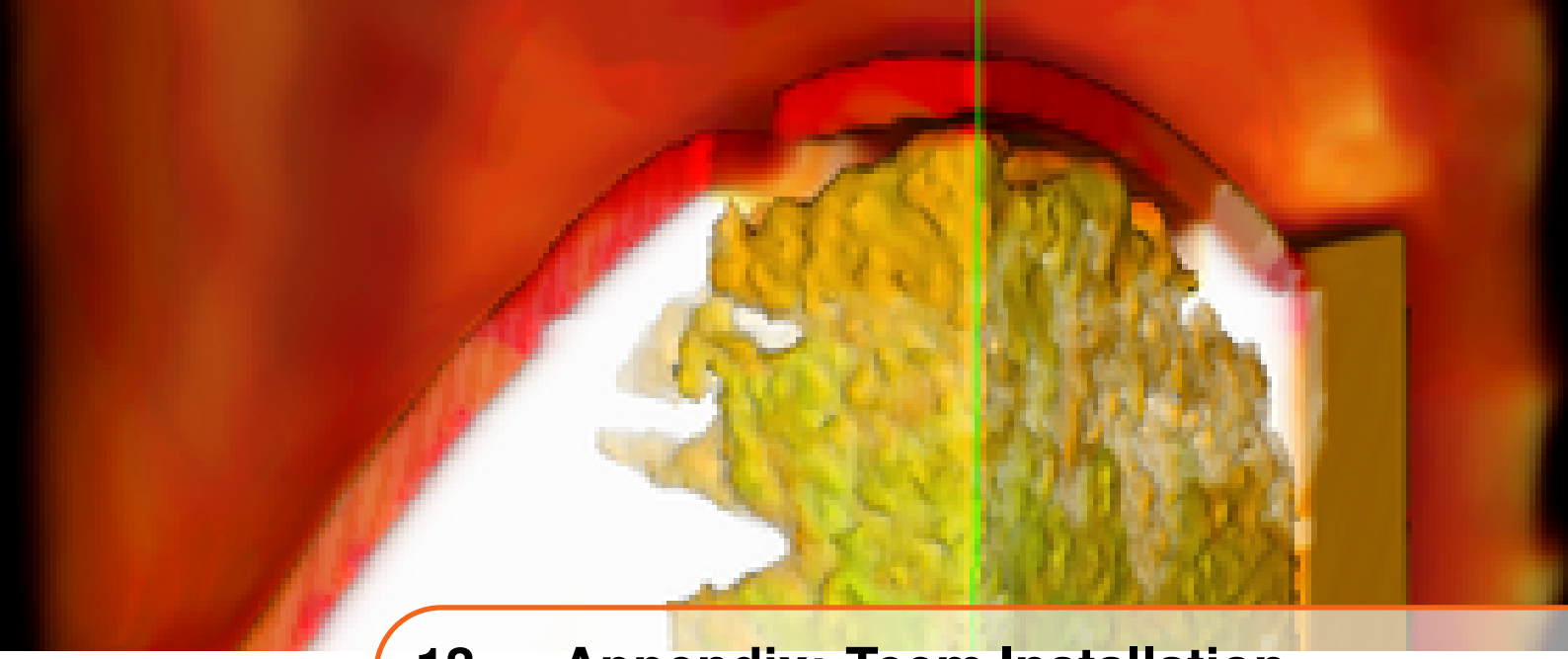
```
mkdir ~/uintah/mac64opt
```

Now browse to the mac64opt directory and configure Uintah using the following: *Note, under Mac OSX 10.6, the build has to be 64-bit.*

```
cd ~/uintah/mac64opt
../src/configure --with-mpi=/Users/username/uintah/packages/openmpi
    -1.4.1-install
\--with-hypre=/Users/username/uintah/packages/hypre-2.6.0b-install
\--with-petsc=/Users/username/uintah/packages/petsc-2.3.3-p15-install
\--enable-optimize='-O2' \--enable-64bit
CC=gcc CXX=g++ F77=gfortran CFLAGS=-m64 CXXFLAGS=-m64 FFLAGS=-m64
PETSC_ARCH=darwin10.3.1-c-opt USE_ICE=yes USE_MPM=yes
```

Finally

```
make all
```



13 — Appendix: Teem Installation

Note:*Teem is only needed for UdaToNrrd (which is used to translate UDA into NRRD files for visualization with the Manta or RTRT ray tracers. Users of* **VisIt** *do not need to install Teem.*

Download Teem from http://teem.sourceforge.net/download/index.html. Teem uses CMake to configure the build system. Installation instructions for Teem are found at http://teem.sourceforge.net/build.html. Simplified instructions are as follows:

```
tar zxf teem-1.10.0-src.tar.gz
cd teem-1.10.0-src/
mkdir teem-build
cd teem-build
ccmake ../
```

Use the arrow keys to scroll down to the fourth line BUILD_SHARED_LIBS and press the return key to toggle the ON flag.

```
Press the 'c' key

Press the 'g' key

make
make install
```

The Teem libraries and include files will be install in /usr/local hierarchy. If you have a /usr/local/lib64 directory please make a symbolic link to the libteem.so found in /usr/local/lib, i.e.

```
cd /usr/local/lib64
ln -s ../lib/libteem.so libteem.so
```

After Teem has been installed then add the location of your Teem install to the **Uintah** as in:

```
--with-teem=/usr/local
```