Install Suse 9.1 Professional. Choose a single disk partition if possible and use the 32 bit system. Do a typical install plus the following:

- c/c++ tools
- experienced user
- KDE env, dev
- Multimedia
- office apps
- Help and support
- Tcl/Tkl dev
- Graphical Base

1. All the bits are in a tar file (uno.files.src.tgz) located on the systems in /tmp/uno. This file is still being created as the installation and use of the bit is determined. Unpack this in /tmp/uno and follow the instructions below.

2. Install cmake in /usr/local. This is simply untaring the file cmake-2.0.3-x86-linux-files.tar from /usr/local.

3. extract ve_suite in /tmp/uno/ve_suite from VE_Suite.tar.

4. Insure that the kernel source is installed using Yast.

5. Insure that the Mesa development environment, the XFree86 development headers are installed using Yast. Note: the order of this and step 6 and even step 7 may need to be re-ordered.

6. Insure that the Nvidia accelerated drivers are installed and that glx is operating. Do the command xdpyinfo and look for the GLX and NV-GLX in the extensions section. To install the nvidia drivers use Yast ans YOU. See the instructions at:
ftp://ftp.suse.com/pub/suse/i386/supplementary/X/XFree86/nvidia-installer-HOWTO

7. install/build vtk (follow the readme in that directory)

Must do the command:

```
env CXX=/usr/bin/c++ CC=/usr/bin/cc cmake -i
```

(Go into advanced mode and select
VTK_HYBRID ON
VTK_PARALLEL ON
MPI OFF
use shared libraries accept other defaults.)

```
make
make install
```

8. Install the Java2 SDK by running the .bin file.

```
ln -s /tmp/uno/java_2_sdk/j2sdk1.4.2_05/jre/plugin/i386/ns610-gcc32/libjavaplugin_oji.so /root/.mozilla/plugins/libjavaplugin_oji.so
```

9. install (using YaST):

- python-devel
- python-opengl
- python-orbit
- python-tk

10. Build/install OMNIOrb per the README.unix

```
make
make install
```

11. install OpenGL performer
untar all the performer*tgz files in / .

12. build/install:

- Scons
  build/install scons using the supplied python script

- CppDOM (www.sf.net/projects/xml-cppdom)
  untar and use scons to install.

- Boost-jam (www.boost.org)
  This has a build.sh script. Then copy bjam to /usr/local/bin

- Boost (www.boost.org)
  This uses bjam to build. bjam "sTOOLS=gcc" install

- GMTL (gtt.sf.net)
  This uses scons to install. scons install.

13. Build vr_juggler and tweek (in ve_suite directory)

untar the file vrjuggler-2.0-alpha4.src.tar in the ve_suite path
create a build directory in ...ve_suite/vrjuggler-2.0-alpha4.src
cd to the vrjuggler build directory

The following is one command line:
../configure.pl
--with-java-orb=JDK --with-cppdom=/tmp/uno/cppdom/cppdom-0.32
--with-boost=/tmp/uno/boost/boost_1_31_0
--with-boost-includes=/tmp/uno/boost/boost_1_31_0
--with-cxx-orb-root=/usr/local
--with-cxx-orb=omniORB4

The prefix option below determines where the code is installed. If it is left blank it goes in /usr/local or you can use the following

--prefix=/tmp/uno/ve_suite/vrjuggler-2.0-alpha4.src

gmake build

gmake install

14. cd to ve_suite/VE_Suite/VE_Installer. There is a script - setup.vista.sh which sets up the environment for building VE_Suite. Source this and then build

```
. ./setup.vista.sh

gmake
```

Now it gets a bit unclear.

need to set a bunch of the environment variables:

```
export CONDUCTOR_BASE_DIR=/tmp/uno/ve_suite/VE_Suite/VE_Conductor
export OMNI_HOME= /tmp/uno/omniorb/omniORB-4.0.4
export VJ_BASE_DIR=
   /tmp/uno/ve_suite/vrjuggler-2.0-alpha4.src/build.linux.suse/instlinks
export VE_SUITE_HOME=/tmp/uno/ve_suite/VE_Suite
```

You need to change the omniORB4.cfg file in the VE_Installer dir.
Change this line:

```
 InitRef = NameService=corbaname::your_computername_or_ip_address:2809
```

This tells our app where to find the omni nameserver.

Also, you will need to remove the following lines from sim.base.jconf:

```
   <corba_remote_reconfig name="CORBA Remote Run-Time Reconfiguration"
version="1">
         <naming_service_host>localhost</naming_service_host>
         <naming_service_port>2809</naming_service_port>
         <iiop_version>1.0</iiop_version>
    </corba_remote_reconfig>
```

Before you can run you have to run omniNames to start an omniORB nameserver. Do this by:

```
omniNames -start 2809
```

This only has to be done on the master omniName system. That system is specified in the
omniORB4.cfg file on all systems.

The second time you just use omniNames only on the server. The clients get it from the omniORB4.cfg file.

After this when you run our app it should get to a point where it prompts you for a parameter file. At this point you could consider the app to work. If you would like to test further there are real simple examples in the VE_TestSuite dir. If you run the app in that dir and use the vrxpr.param file you will see the app come up in sim mode.

To get the vrml file into perfly use this notation:

```
Perfly filename_here
```

Note that this doesn't work with all vrml files. The cargo ship file had to be changed to a fly file (pfb).

To get a cluster to work you need to edit the makefile in VE_Xplorer and uncomment the line

```
CLUSTER_APP = TRUE
```

Then you need to edit the cfdApp.cxx file in the same directory and change the hardcoded path to the cluster master. There are two instances at lines 476 and 3038. Also at line 3046 hostname is set to a "/". Changing it to a null "" will allow you to use relative paths for the command line files.

gmake in that directory.

This needs to be done on all systems.

The data needs to be available on all systems. This argues strongly for a shared file system.

To run this start project_cluster <jconf files> on the master and give it a param file if needed. When it has processed all that it can then start project_cluster the same way on the slaves. The windows should startup and away you go.