Building blender on Linux for blenderVR

Some of blender releases hold bugs that may interfere with blenderVR. The last blender version known to work for blenderVR is the 2.69.

During this install, we assume you use the brew package manager to install packages like cmake or wget.

To compile faster, you can replace `make` by `make -j n` to involve n core in the compilation. Check *n* value by typing “sysctl -n hw.ncpu” in the terminal.

Paths

At first, we must define where to install blender for blenderVR. Due to its dependencies, we will define a whole tree with all necessary libraries.

Set an environment variable with the root folder of blender installation:

```
export BLENDER_ROOT_PATH=/usr/local/blender/2.69
export BLENDER_ROOT_DEPENDENCIES=$BLENDER_ROOT_PATH/dependencies
```

By default, we suggest `/usr/local/blender/2.69`, but you can use whatever path you prefer. You can even choose a path in your home directory although you won't be able to share it with other computer users and you should not do `sudo` commands. Anyway, don't forget that you won't be able to move it after compilation as python built in blender will rely on this given path to find some modules.

Ensure that you have read and write access to this path and create its folder:

```
sudo mkdir -p $BLENDER_ROOT_PATH
sudo chown $(whoami) $BLENDER_ROOT_PATH
cd $BLENDER_ROOT_PATH
mkdir $BLENDER_ROOT_DEPENDENCIES
mkdir compile
```

script for environment variables

First, we will create a little script to set correct environment variable:

```
cat > $BLENDER_ROOT_PATH/compile/.vars << EOF
export BLENDER_ROOT_PATH="$BLENDER_ROOT_PATH"
export BLENDER_ROOT_DEPENDENCIES="$BLENDER_ROOT_PATH/dependencies"
```

```
if [[ \$PATH != *$BLENDER_ROOT_PATH/dependencies/bin* ]] ; then
   export PATH="$BLENDER_ROOT_PATH/dependencies/bin:\$PATH"
fi
if [[ \$LD_LIBRARY_PATH != *$BLENDER_ROOT_PATH/dependencies/lib* ]] ; then
   export
LD_LIBRARY_PATH="$BLENDER_ROOT_PATH/dependencies/lib:\$LD_LIBRARY_PATH"
fi
EOF
```

Thus, whenever you want to compile part of blender, you will only have to do

```
. /usr/local/blender/2.69/compile/.vars
```

to get all the relevant environment variables defined (beware of the dot and the space before the path). You can replace `/usr/local/blender/2.69` by your blender root path.

In the following steps, we assume that environment variables defined above are set.

You can make sure by typing:

```
echo $PATH | cut -d ':' -f1
```

If the path is wrong (you should recognize your BLENDER_ROOT_DEPENDENCIES variable before `/bin`), then you can retype the upper command `. /usr/local/blender/2.69/compile/.vars`.

For OSX users, we strongly advise you to use brew package manager for hereafter installs of cmake and wget. You'll then probably have different versions of cmake linked from /usr/local/Cellar to /usr/local/bin than the one listed bellow. Once correct versions installed, you can temporarily use command line “brew switch cmake 2.8.12.2” to be sure you use the correct versions during the compilation.

cmake

Current usable cmake to use is version 2.8.12.2. If your system already have this version, you can bypass this step. Beware: 3.x versions are not currently supported by blender.

```
export CMAKE_VERSION=2.8.12.2
echo "export CMAKE_VERSION=$CMAKE_VERSION" >>
$BLENDER_ROOT_PATH/compile/.vars
brew install cmake
brew switch cmake 2.8.12.2
```

Due to unknown issue, when you try to configure a compilation with cmake after that, you can have wrong cmake behavior (e.g. saying that you have invalid rights). If so,

you should start a new shell, load the environment variables (. /usr/local/blender/2.69/compile/.vars) and continue the process where it failed.

Python and numpy

python

blender 2.69 requires python 3.3.x and associated numpy package. python 3.3 is available in darwin-9.x.universal package provided by the blender community.

pip

numpy

VRPN

VRPN install is not compulsory, yet blenderVR will require the VRPN protocol for every VR related device interactions (e.g. tracking system)

```
export VRPN_VERSION=07_31
echo "export VRPN_VERSION=$VRPN_VERSION" >> $BLENDER_ROOT_PATH/compile/.vars
mkdir -p $BLENDER_ROOT_PATH/compile/vrpn-$VRPN_VERSION/build
cd $BLENDER_ROOT_PATH/compile/vrpn-$VRPN_VERSION/
wget www.cs.unc.edu/Research/vrpn/downloads/vrpn_$VRPN_VERSION.zip
unzip vrpn_$VRPN_VERSION.zip
cd build
cmake -DCMAKE_INSTALL_PREFIX=$BLENDER_ROOT_DEPENDENCIES ../vrpn
make
make install
```

blender

Let start the “hard work” ... Here we'll download blender source, patch them and compile blender into its (slightly) modified version that'll serve as blenderVR core application.

download

```
export BLENDER_VERSION=2.69
echo "export BLENDER_VERSION=$BLENDER_VERSION" >>
$BLENDER_ROOT_PATH/compile/.vars
mkdir -p $BLENDER_ROOT_PATH/compile/blender-$BLENDER_VERSION/build
cd $BLENDER_ROOT_PATH/compile/blender-$BLENDER_VERSION/
wget download.blender.org/source/blender-$BLENDER_VERSION.tar.gz
tar -xf blender-$BLENDER_VERSION.tar.gz
```

download external libraries

```
cd $BLENDER_ROOT_PATH/compile/blender-$BLENDER_VERSION/
mkdir lib
cd lib
svn checkout https://svn.blender.org/svnroot/bf-
blender/trunk/lib/darwin-9.x.universal -r r61340
```

(copy numpy in python site-packages)

patching for blenderVR

All That... for This?!

```
cd $BLENDER_ROOT_PATH/compile/blender-$BLENDER_VERSION/
wget blendervr.limsi.fr/downloads/patchs/blender-$BLENDER_VERSION.patch
patch -p0 < blender-$BLENDER_VERSION.patch
(need a link to llvm-3.4.patch)
cd blender-$BLENDER_VERSION
patch -p0 < ../llvm-3.4.patch
```

configure and compile

```
cd $BLENDER_ROOT_PATH/compile/blender-$BLENDER_VERSION/build
cmake -DCMAKE_INSTALL_PREFIX=$BLENDER_ROOT_DEPENDENCIES -DWITH_CYCLES=OFF -
DWITH_PLAYER=ON ../blender-$BLENDER_VERSION
make
make install
```

Feel free to use ccmake to adapt to your requirements. For instance, you can add FFMPEG or any other option flag to include associated library to blender compilation.

automatically add VRPN ?

If you use VRPN, you'll want to create a symbolic link towards its shared object to avoid an additional

VRPN folder inside the blender tree.

```
cd $BLENDER_ROOT_PATH/compile/blender-
$BLENDER_VERSION/build/bin/$BLENDER_VERSION/python/lib/python$PYTHON_MAJOR_V
ERSION/lib-dynload
find $BLENDER_ROOT_PATH/dependencies -name vrpn.so -exec ln -s {} . \;
```

Creation of scripts to run blender and blenderplayer

Due to shared library compilation of python, we will require specific scripts to load blender.

```
mkdir -p $BLENDER_ROOT_PATH/bin
cat > $BLENDER_ROOT_PATH/bin/blender << EOF
#!/bin/bash

export LD_LIBRARY_PATH="$BLENDER_ROOT_DEPENDENCIES/lib:\$LD_LIBRARY_PATH"

exec $BLENDER_ROOT_PATH/compile/blender-
$BLENDER_VERSION/build/bin/\$(basename \$0) "\$@"
EOF
chmod a+x $BLENDER_ROOT_PATH/bin/blender
ln -s $BLENDER_ROOT_PATH/bin/blender $BLENDER_ROOT_PATH/bin/blenderplayer
```

Finally, to run blender or blenderplayer, you only need to run:

```
/usr/local/blender/2.69/bin/blender
```

or

```
/usr/local/blender/2.69/bin/blenderplayer
```

You can copy or link (ln -s) both executable to any folder already in your path for easier access. Their is no more need to load variables with the `/usr/local/blender/2.69/compile/.vars` script. As usual, replace `/usr/local/blender/2.69` by the blender root path you have choosen.

Re-owning for all users

You can re-own all the tree to allow any user of the computer to use this blender compilation:

```
sudo chown -R root:root $BLENDER_ROOT_PATH
```

Moreover, it reduces the chances to accidentally modify files inside this (now sacred) tree  .

Adding blender in the main system binary path

Don't do this if you already have a system wide installation of blender (for instance through system package installation process (`yum`, `rpm`, `apt-get`, `dpkg` ...)

You can also add links inside main PATH to use blender:

```
sudo ln -s $BLENDER_ROOT_PATH/bin/* /usr/bin
```