blenderVR XML configuration

The blenderVR XML configuration file is loaded by the console (see Console vs virtual environment) to get the architecture related information to run blenderVR and send it to each rendering node (“virtual environment” part).

This file must contain at least four sections, plus the `plugins` section. It also includes a `blenderVR` section which only option is the network port used for the synchronization between the rendering nodes.

Use of space in `screen` name should work. Beware still Windows users..

redundant sections

Some elements can be specific to one node, other shared. For instance, the `blenderplayer` executable can be the same for all rendering nodes or different on some nodes. In such case, there will be a section called `system` that can be inherited by each `computer` sub-section:

```
<computers>
  <system>
    <blenderplayer executable='/usr/local/blender/2.69/bin/blenderplayer' />
  </system>
  <computer name='front computer' hostname='front.fqdn'>
    <system>
      <blenderplayer executable='/usr/bin/blenderplayer' />
    </system>
  </computer>
  <computer name='left computer' hostname='left.fqdn' />
  <computer name='right computer' hostname='right.fqdn' />
</computers>
```

In this example, `left computer` and `right computer` nodes will use `/usr/local/blender/2.69/bin/blenderplayer` whereas `front computer` node will use `/usr/bin/blenderplayer`.

The `system` section is called *rendundant* as many entries will use the same information

code execution

In the XML file, you can use back-quote to execute code. First, the XML parser will try to execute this code as python code in blenderVR environment system (with all variables and import present in the blenderVR XML parser). If it fails, then, it tries as bash code and take the stdout result. If none is valid it raises an error.

For instance,

```
<environment>HOME=`os.environ['HOME']`</environment>
```

will define an environment variable (passed to the daemon or blenderplayer) called HOME that contains the current value of HOME operating system environment variable (with `os.environ` python code).

You can even use inherited values from redundant section:

```
<login remote_command="ssh `self._attributs_inheritance['hostname']`">
```

used inside the system redundant section will specify that `remote_command` will include the hostname as given in the `computer` entry.

If uncertain, we suggest you to simply print the `self._attributs_inheritance` python dictionary:

```
<login remote_command="`print(self._attributs_inheritance)`">
```

that will raise an exception (which is the point, since your purpose here is to create your configuration file, not to run blenderVR)

starter section

This section only concerns the console. It contains all screen sets definitions.

```
<starter>
  <config name='console'>console</config>
  <config name='virtual environment'>console, front screen, left screen,
right screen</config>
  ...
</starter>
```

You can also add a `hostname` attribute in case of `socket.gethostname()` python function returns wrong hostname. This hostname is used by all “virtual environment” nodes to contact the console for network connection control.

Each `config` sub-section must list all screens, separated by commas, used by this screen set.

De facto, the first screen listed here is the master node.

Anchor

On some devices, the paths are not homogenous: the root path (repository) of `.blend` files on the console is not the same than on the master and/or on the slaves.

To fix that, blenderVR uses the notion of **Anchor**: it is a node specific absolute path on all nodes that prefixes each relative path for blender and processor files.

It is a kind of least common multiple path. For instance, with two computers:
console blender files repository: `/home/me/blender_files`
master node blender files repository: `/remote_home/me/blender_files`
This least common path is `/home` on the console and `/remote_home` on the master node (`me/blender_files` are common on both systems).

In such case, the starter section (console specific section) will start by:

```
<starter anchor='/home'>
```

Whereas system section for the master node will start by:

```
<system anchor='/remote_home'>
```

users section

Each `user` must be listed here. Several users will e.g. enable you to attach a head tracker to adapt stereoscopic rendering to different points of view inside the virtual environment.

The `behavior` redundant section can define the `default_position` (`0.0, 0.0, 0.0` by default) or the `eye_separation` (6 centimeters by default) of the user.

```
<!-- users section with default values -->
<users>
  <behavior eye_separation='0.06'>
    <default_position>0.0, 0.0, 0.0</default_position>
  </behavior>
  <user name="user A" />
</users>
```

computers section

We must describe how each rendering node (computer) works: each computer can have a specific configuration to run blenderplayer (paths, environment variables ...). However, most of the time, all computers are equivalent. Redundant section is useful!

Computer itself must have a `name` and a `hostname`. The name will be used by the screen.

```
<computers>
  <system>
    ... <!-- computers global information -->
  </system>
  <computer name='front computer' hostname='front.fqdn'>
    <system>
```

```
      ... <!-- front computer specific information -->
    </system>
  </computer>
  <computer name='left computer' hostname='left.fqdn' />
</computers>
```

system section

The `system` redundant section defines many things:

```
<system root='C:\\program\\blenderVR' anchor='U:\\blender_files'>
  <login remote_command="ssh `self._attributs_inheritance['hostname']`"/>
  <daemon>
    <environment>SystemRoot=C:\\Windows</environment>
  </daemon>
  <blenderplayer
executable='C:\\blenderCave\\blender\\v2.70a\\blenderplayer.exe'>
    <environment>PYTHONPATH=C:\\Python33\\Lib;</environment>
  </blenderplayer>
</system>
```

The `root` parameter specifies the root path of blenderVR (where resides the `blenderVR` python script, the `modules` folder, etc.). By default, it is set to blenderVR root path on the console computer. However, due to not homogenous paths between nodes, you may have to define it for each system.

See Anchor to know the purpose of anchor parameter.

login sub-section

This section explains how to connect console and hosts computers.

```
<login remote_command="ssh me@host" python="/usr/bin/python3"/>
```

or

```
<login remote_command="psexec -d \\host" python="C:\\python33\\python.exe"/>
```

remote_command specifies the command, from the computer running the console to connect to the remote host.
python contains the path and the name of the python3 executable.

Generally, we use redundant system section with code execution to create this section (see example of the redundant section upper).

daemon sub-section

The daemon sub-section explains how to run the daemon (now that we know how to connect to the

remote computer).

```
<daemon transmit='True'>
  <environment>SystemRoot=C:\\Windows</environment>
</daemon>
```

transmit parameter specifies if the daemon must transmit the environment variables to blenderplayer while it runs it.
environment sub-section adds some specific environment variable to the daemon.

On Windows, you must at least, set the `SystemRoot` variable to points towards the path of your Windows installation (generally: `C:\\Windows`)

blenderplayer sub-section

This section defines how to run blenderplayer.

```
<blenderplayer
executable='C:\\blenderVR\\blender\\v2.69\\blenderplayer.exe'>
<environment>PYTHONPATH=C:\\Python33\\Lib;C:\\Python33\\DLLs;C:\\Python33\\L
ib\\site-packages</environment>
</blenderplayer>
```

The **executable** parameter contains the path and the binary name of patched version of blenderplayer.
The **environment** sub-sections allows you to add specific environment variables for blenderplayer. You can add PYTHONPATH environment to specify paths for optional modules (such as for VRPN).

screens section

The screen is the unit of rendering: there is bijection between screen and instance of `blenderplayer`. Each screen has a `name` and a `computer` (actually the name of the computer section, above).

```
<screens>
  <display>
    ... <!-- screens global informations -->
  </display>
  <screen name='front screen' computer='front computer'>
    <display>
      ... <!-- front screen specific informations -->
    </display>
    <wall>
      ...
    </wall>
  </screen>
```

```
  <screen name='left screen' computer='left computer'>
</screens>
```

The `display` redundant section defines several things:

- **`options`** passed as argument to `blenderplayer` (for instance, `-f -s hwpageflip` to request a stereoscopic full screen `blenderplayer` window).
- **`environment`** to pass specific environment variables to `blenderplayer`.
- **`graphic_buffer`** to associate:
  - `buffer` (`alone` = no stereo, `left` graphic buffer or `right` graphic buffer,
  - `user` (as given inside `users` section),
  - eye of the user (`left`, `middle` or `right`).
- **`viewport`** to reduce the screen (usefull if you have occlusion).

```
<display options='-w 400 400'>
  <viewport>420, 0, 1500, 1080</viewport>
  <environment>DISPLAY=:0.0</environment>
  <graphic_buffer buffer='alone' user='user A' eye='middle'/>
</display>
```

Each screen must have one sub-section `wall` or `hmd`.

Wall or HMD differs in the way they manage the projection. Wall screens are fixed in the real world but HMD screen are attached to head of the user, moving along.

Both require a screen definition: three corners (top right, top left and bottom right):

```
<wall or hmd>
  <corner name="topRightCorner">1.0, 1.0, -1.0</corner>
  <corner name="topLeftCorner">-1.0, 1.0, -1.0</corner>
  <corner name="bottomRightCorner">1.0, -1.0, -1.0</corner>
</wall or /hmd>
```

For Wall, the screens are defined in vehicle reference frame. For HMD, the screens are defined in the reference frame of head tracker.

Sample configuration file

This sample configuration file can be used for a cave with three vertical square (2 m x 2 m) screens (left, front and right) plus a console computer with a single windowed screen.

config.xml

```
<?xml version="1.0"?>
<blenderVR>

  <starter anchor='/tmp/console'>
      <config name='console'>console screen</config>
      <config name='virtual environment'>console screen, front screen,
```

```
left screen, right screen</config>
  </starter>

  <users>
    <user name='user A' />
  </users>

  <!-- Here, we define the console parameters -->
  <computers>
    <computer name='console computer' hostname='console.fqdn'/>
  </computers>
  <screens>
    <screen name='console screen' computer='console computer'>
      <display options='-w 600 600'>
    <environment>DISPLAY=:0.0</environment>
    <graphic_buffer user='user A'/>
      </display>
      <wall>
    <corner name='topRightCorner'>1.0, 1.0, -1.0</corner>
    <corner name='topLeftCorner'>-1.0, 1.0, -1.0</corner>
    <corner name='bottomRightCorner'>1.0, -1.0, -1.0</corner>
      </wall>
    </screen>
  </screens>

  <computers>
    <system root='/usr/local/blender/vr/1.0' anchor='/tmp/node'>
      <login remote_command="ssh
`self._attributs_inheritance['hostname']`"
python='/usr/local/blender/2.69/dependencies/bin/python3.3'/>
      <daemon transmit='True'>
    <environment>PATH=/usr/bin:/bin</environment>
      </daemon>
      <blenderplayer
executable='/usr/local/blender/2.69/bin/blenderplayer' />
    </system>
    <computer name='front computer' hostname='front.fqdn' />
    <computer name='right computer' hostname='right.fqdn' />
    <computer name='left computer' hostname='left.fqdn' />
  </computers>
  <screens>
    <display options='-f -s hwpageflip'>
      <environment>DISPLAY=:0.0</environment>
      <graphic_buffer buffer='left' user='user A' eye='left'/>
      <graphic_buffer buffer='right' user='user A' eye='right'/>
    </display>
    <screen name='front screen' computer='front computer'>
      <wall>
    <corner name='topRightCorner'>1.0, 1.0, -1.0</corner>
    <corner name='topLeftCorner'>-1.0, 1.0, -1.0</corner>
    <corner name='bottomRightCorner'>1.0, -1.0, -1.0</corner>
```

```
      </wall>
    </screen>
    <screen name='left screen' computer='left computer'>
      <wall>
    <corner name='topRightCorner'>-1.0, 1.0, -1.0</corner>
    <corner name='topLeftCorner'>-1.0, 1.0, 1.0</corner>
    <corner name='bottomRightCorner'>-1.0, -1.0, -1.0</corner>
      </wall>
    </screen>
    <screen name='right screen' computer='right computer'>
      <wall>
    <corner name='topRightCorner'>1.0, 1.0, 1.0</corner>
    <corner name='topLeftCorner'>1.0, 1.0, -1.0</corner>
    <corner name='bottomRightCorner'>1.0, -1.0, 1.0</corner>
      </wall>
    </screen>
  </screens>

  <plugins>
    <vrpn>
      <floor x='0.0'/>
      <tracker device='GTK' host='localhost'>
    <transformation>
      <post_translation z='-1.6'/>
      <post_rotation x='1.0' y='1.0' z='1.0' angle="`-2*math.pi/3`"/>
      <pre_rotation x='1.0' y='1.0' z='1.0' angle="`2*math.pi/3`"/>
    </transformation>
    <sensor id='0' processor_method='user_position' users='user A'/>
    <sensor id='1' processor_method='tracker_1'/>
    <sensor id='2' processor_method='tracker_2'/>
    <sensor id='3' processor_method='tracker_3'/>
      </tracker>
      <analog device='GTK' host='localhost'
processor_method='movements'/>
      <button device='GTK' host='localhost'
processor_method='buttons'/>
    </vrpn>
  </plugins>
</blenderVR>
```