DisplayCluster Manual

DisplayCluster is a software environment for interactively driving large-scale tiled displays. The software allows users to interactively view media such as high-resolution imagery and video, as well as stream content from remote sources such as laptops / desktops or high-performance remote visualization machines. Many users can simultaneously interact with DisplayCluster with devices such as joysticks or touch-enabled devices such as the iPhone / iPad / iTouch or Android devices. Additionally, a Python scripting interface is provided to automate interaction with DisplayCluster.

This manual provides information on the installation and usage of DisplayCluster.

Table of Contents

License

Installation

Dependencies

The following required dependencies should be installed prior to building DisplayCluster. Many of these can be installed with a package manager. Any packages built manually should be installed outside of the DisplayCluster source / build directory, for example in /usr/local/.

- CMake 2.4 or higher (2.8+ recommended)
- MPI (tested with OpenMPI 1.4.1)
- Qt 4.8.0 or higher
- Boost 1.44.0 or higher
- FFMPEG 0.8.x or higher (tested with 0.8.2 and 0.9.1)
- libjpeg-turbo (tested with 1.1.90)

Building

After unpacking the DisplayCluster source, change to the source directory, create a build directory, and change to that build directory:

```
cd DisplayCluster
mkdir build
cd build
```

Run CMake to configure the build:

```
ccmake ../
```

Press ‘c’ to configure.

Initially you will be presented with two build options, *BUILD_DESKTOPSTREAMER* and *BUILD_DISPLAYCLUSTER*. These options enable the DesktopStreamer client application for streaming remote content to a DisplayCluster instance, and the DisplayCluster application for driving the tiled display, respectively. For a default installation, set these both to *ON*:

- Build option: *BUILD_DESKTOPSTREAMER: ON*
- Build option: *BUILD_DISPLAYCLUSTER: ON*

Press ‘c’ to configure the build again. CMake will attempt to automatically locate all of the required dependencies. It will produce error messages if it cannot find a particular dependency. In these cases, the CMake settings must be edited manually with the proper locations. Pressing ‘t’ will show all build settings. The installation prefix *CMAKE_INSTALL_PREFIX* should be changed. The recommended setting is */opt/displaycluster*. It may be necessary to run the configuration multiple times (by pressing ‘c’). After the build has been successfully configured, the Makefile can be generated by pressing ‘g’.

After CMake has generated the Makefile, simply run:

```
make
```

make install

Configuration

A single configuration file *configuration.xml* must be created to describe the tiled display cluster and screen layout. An example is provided in the distribution and can be modified to a particular tiled display.

Change to the directory where DisplayCluster was installed and copy the example configuration file into place:

cd /opt/displaycluster
cp examples/configuration.xml ./

An example *configuration.xml* is:

```
<configuration>
   <dimensions numTilesWidth="2" numTilesHeight="2" screenWidth="400"
screenHeight="400" mullionWidth="50" mullionHeight="50" fullscreen="0"/>

   <process host="localhost" display=":0">
      <screen x="0" y="0" i="0" j="0"/>
      <screen x="400" y="0" i="1" j="0"/>
   </process>
   <process host="localhost" display=":0">
      <screen x="0" y="400" i="0" j="1"/>
      <screen x="400" y="400" i="1" j="1"/>
   </process>
</configuration>
```

numTilesWidth and *numTilesHeight* represent the monitor layout of the tiled display. *screenWidth* and *screenHeight* represent the resolution of each monitor. *mullionWidth* and *mullionHeight* represent pixels hidden behind monitor mullions / bezels. Finally, *fullscreen* (0 or 1) represents if the window on each monitor should be opened in OpenGL fullscreen mode or not.

Each render node must have at least one process. If the node has only display (in Linux this refers to the X display, for example :0 or :1), only one process should be used for best performance. The *host* attribute is required and represents the host name for the given process. The *display* attribute is required on Linux and represents the X display.

All of the screens for a process must be specified. The *x* and *y* attributes represent the pixel displacement to the upper-left corner of the screen in the given display. The origin is located at the upper-left corner of all the screens in the display. The *i* and *j* attributes represent the tile coordinates of the screen in the tiled display. The origin (0,0) represents the upper-left screen.

Optional Features

There are several optional features which can be enabled at build time. These are described below. After enabling any of these features, it will be necessary to reconfigure in CMake, generate a new Makefile, and run ‘make’ and ‘make install’ again as described in the Building section above.

Joystick Interaction

An arbitrary number of joysticks can be used to interact with DisplayCluster. Logitech gamepad controllers are known to work well. The following additional dependency and CMake build option is required:

- SDL 1.2
- Build option: *ENABLE_JOYSTICK_SUPPORT: ON*

Python Scripting Support

DisplayCluster can be controlled via a Python interface. This allows users to interact via a Python console or run Python scripts. The following additional dependencies and CMake build option are required:

- PythonQt 2.0.1.DC (provided in dependencies/ directory)
- SIP 4.x (tested with 4.13.1)
- PyQt 4.x (tested with 4.9)
- Build option: *ENABLE_PYTHON_SUPPORT: ON*

Note that a modified version of PythonQt is required. This is provided with the source distribution.

Touch Device Interaction

Touch interaction is possible using iOS devices (iPhone, iPad, iTouch) and Android devices. In fact, any device that uses the TUIO protocol can operate with DisplayCluster. The following additional dependency and CMake build option is required:

- TUIO C++ Client Reference Implementation 1.4 (provided in dependencies/ directory)
- Build option: *ENABLE_TUIO_TOUCH_LISTENER: ON*

Usage

The following describes how to start and use DisplayCluster after installation.

Prerequisites

Before running DisplayCluster, you should make sure that your MPI environment is working

correctly. You should be able to run an MPI ‘hello world’ application across the cluster. In general this requires MPI to be installed on all nodes of the cluster, and for SSH public key authentication (passwordless SSH) to be setup.

You should make sure that the tiled display monitors are all on and not in a power-saving mode. In Linux this can be accomplished by executing the following commands on all nodes of the cluster (the display :0 can be adjusted as necessary):

xset -display :0 dpms force on
xset -display :0 s reset

In Linux environments, you need to make sure that remotely executed applications can display to the X server. A simple test is to make sure an *xterm* can be launched on a render node remotely (the display :0 can be adjusted as necessary):

ssh <rendernode> DISPLAY=:0 xterm

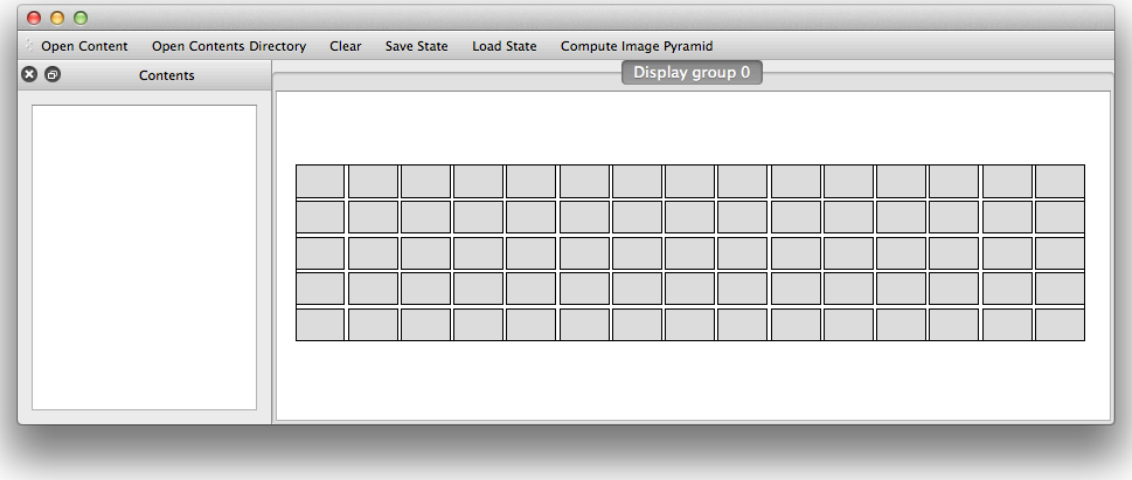
If successful, you will see an xterm appear on a screen of the tiled display. If unsuccessful, it will be necessary to adjust permissions on the X server, possibly by editing the X configuration file or using the *xhost* command.

Starting DisplayCluster

DisplayCluster can be started by executing in a terminal on the head node:

/opt/displaycluster/bin/startdisplaycluster

For convenenience, you may wish to add */opt/displaycluster/bin* to the *$PATH* environment variable. Afterward, you can launch by simpling running *startdisplaycluster*. The user interface (shown below) should then appear on the head node’s display, and the tiled display should turn black.



The user interface shows a representation of the tiled display as specified in the *configuration.xml* file. A test pattern can be displayed across the tiled display to verify everything

has been configured correctly by selecting *Show Test Pattern* in the *View* menu. The test pattern shows a colored diagonal grid and information about each screen. If the test pattern appears incorrectly on the tiled display, you may need to edit the *configuration.xml* file and re-launch DisplayCluster.

Once launched, imagery and movie content can be opened using the *Open Content* dialog. Entire directories of content can be opened and shown in a grid layout using the *Open Contents Directory* dialog. State files representing all of the currently shown content can be saved and loading via the *Save State* and *Load State* dialogs.

Extremely high-resolution imagery is supported by DisplayCluster. These images can be opened and processed in real time by simply opening them via *Open Content*. For best performance, an image pyramid can be computed and stored to disk for later use. This is accomplished using the *Compute Image Pyramid* dialog. A new directory will be created with the processed pyramid files. A *.pyr* file representing the image pyramid (for example image.jpg.pyr) will be created in the same directory as the original image. This pyramid file can be opened via the *Open Content* dialog. Note that the *.pyr* file can be renamed and relocated for convenience. If the pyramid directory moves however, the *.pyr* file will need to be edited to reference the new location.

Joystick Interaction

Multiple joysticks or gamepad controllers can be used simultaneously to interact with DisplayCluster. Logitech gamepad controllers are known to work well. Controllers should be connected to the head node before launching DisplayCluster. Note that controllers should be used in analog mode if they have that option. The controllers can then be used to interact with windows on the tiled display according to the following diagram:

Python Scripting

Documentation for the Python scripting API is coming soon.

Touch Device Interaction

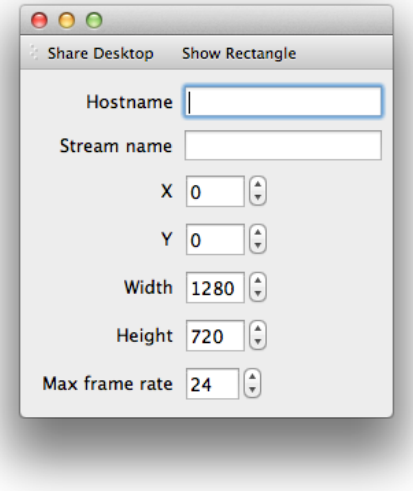
Documentation for touch device interaction is coming soon.

Desktop Streaming

Desktop contents can be streamed directly to DisplayCluster. The *DesktopStreamer* application can be launched with the command:

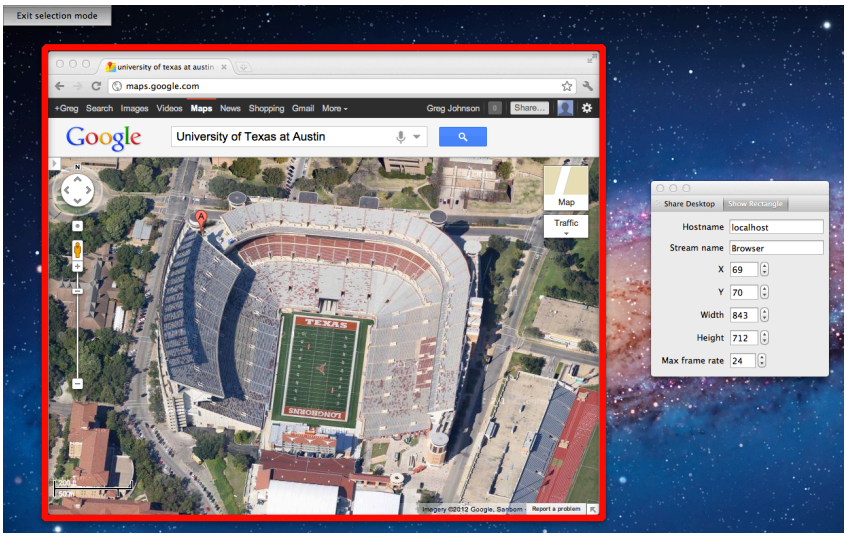
/opt/displaycluster/bin/desktopstreamer

After launch the application will appear as



If launching on the head node where DisplayCluster was started, the hostname should be specified as *localhost*. The stream name is a user-selected unique identifier to name the displayed window. *X*, *Y*, *Width*, and *Height* specify the rectangle on the desktop to stream. The origin (*X*, *Y*) is located at the upper-left corner of the desktop. The maximum frame rate of the stream can be adjusted as well. This can be used, for example, to limit the outgoing network bandwidth from the *DesktopStreamer* application.

For convenience, the *Show Rectangle* button allows users to select the rectangle graphically. After rectangle selection, click the *Exit selection mode* button at the upper-left to hide the rectangle.



The *DesktopStreamer* application can be built on other machines and allows for streaming of remote content from laptops / desktops or high-performance remote visualization machines. In these cases, simply change the hostname to the DisplayCluster head node address. Port 1701 will need to be publicly accessible.