GPU Compositing

About GPU Compositing

GPU compositing is an OpenGL ES2.0 based composition solution that provides the logic to display video and graphics frames from the memory on a display supported by the device. It uses texture streaming extension, which is a proprietary OpenGL ES extension from Imagination Technologies for rapidly changing textures.

The GPU compositing includes the following main features:

- **Build time user configurable number of planes (Graphics & Video separately)**
- **Graphics Pipeline**
  - supported pixel formats: RGB565, ARGB8888
  - Blending - global or pixel level alpha
  - Blending order - Graphics plane 0 being the first.
- **Video Pipelines**
  - supported pixel formats: YUV422, NV12,I420 & YUV420
  - Overlay Order: Video plane 0 at the bottom.
- **Maximum plane resolution of 2048x2048**
- **Plane Resizing - configurable output window position and resolution in terms of normalized device co-ordinates**
- **Plane Rotation (Z-axis only) - in units of decimal degrees**
- **Automatic color conversion to the output frame buffer format (ARGB8888 & RGB565)**
- **Flicker Free Composition - Synchronized with Display Rate**

The solution comprises of the following components

- composition - OpenGL ES2.0 based composition
- gpuvsink - GStreamer GPU based video sink works in conjunction with the composition application
- linuxfbofs - Qt display driver for rendering to an offscreen surface
- transBgSmpQtUI - Sample Qt Application demonstrating simple UI with transparent background

GPU Compositing Source Download [1]

The following representation is used to differentiate between the host and target commands throughout this document

- $ - command to be executed on host
- # - command to be executed on target

Dependencies

- AM SDK [2] - Graphics SDK needs to be upgraded to 04_06_00_03 along with one change [3]

  Steps to upgrade the filesystem with Graphics SDK 04_06_00_03 - for AM335x only

  - change - Modify the texture streaming device count to 10 in the file "GFX_Linux_KM/services4/3rdparty/bufferclass_ti/bc_cat.c'

    ```
    #define DEVICE_COUNT 10
    ```

  - Build the Graphics SDK as per the instructions in the Graphics SDK User Guide for the release build except the last "make install" step

- Instead of the "make install", copy the directory gfx_rel_es8.x (for AM335x) manually on to the target file system from host as below

  Graphics_SDK_4_06_00_03$ cp -r gfx_rel_es8.x /home/a0756700/nfs/am335x_0505_fs/opt/

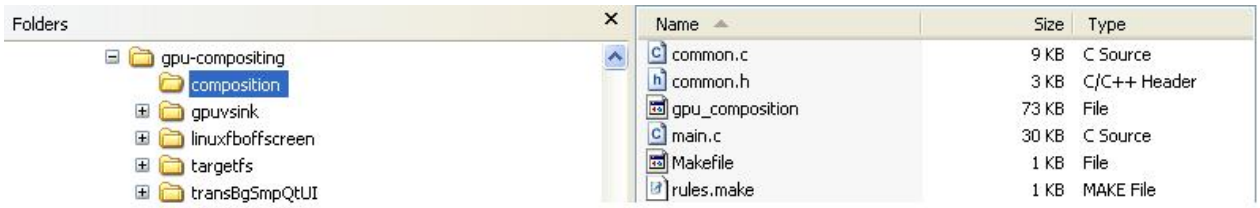
- Run the update script on target

  #./upgradeGfxSdk.sh

- Reboot EVM

- CMEM - Linux Utility for allocating contiguous memory [4]

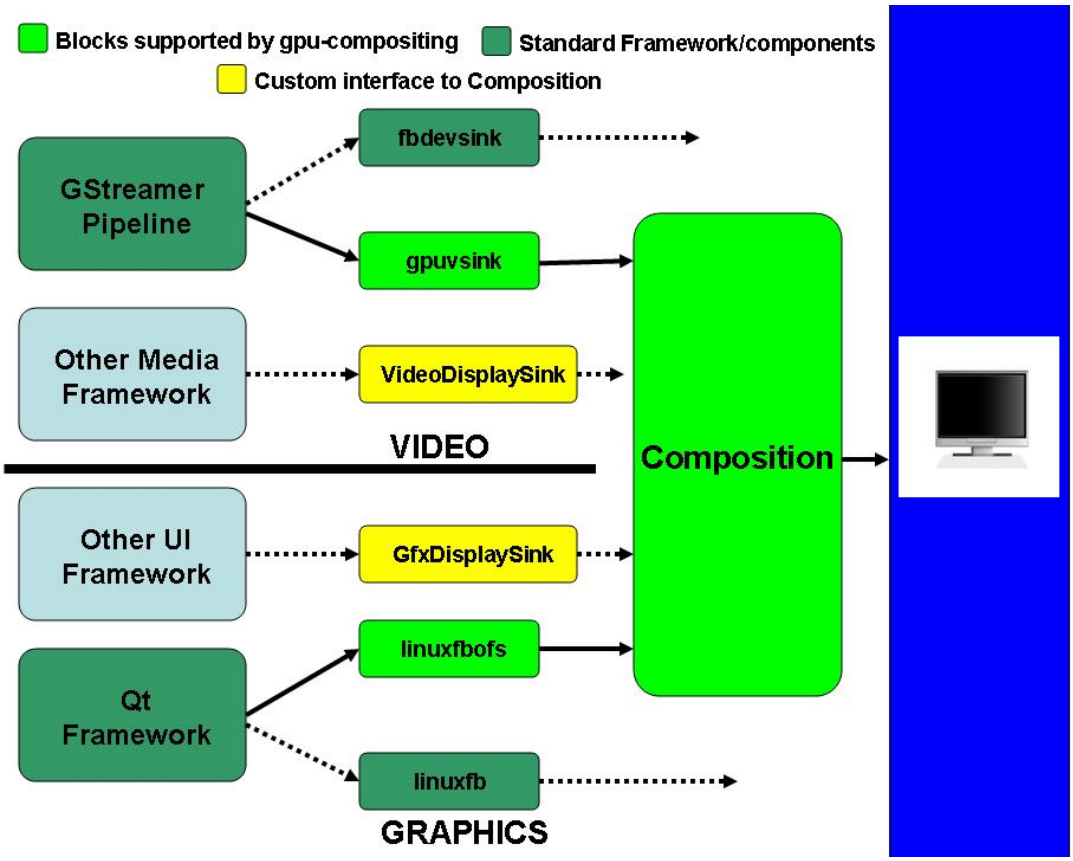
Directory structure



gpu-compositing - Top level folder containing the following sub-directories

- composition - GPU based composition - OpenGL ES2.0 based application
- gpuvsink - GStreamer plugin - GPU based video sink, which works in conjunction with the composition application using named pipe.
- linuxfboffscreen - Qt Display driver for rendering to offscreen surface (depends on CMEM module for allocating contiguous memory for the offscreen surface)
- targetfs - contains target init script, creates required named pipes
- transBgSmpQtUI - Sample Qt Application with transparent background
- gpucomp.h - Public header file for the gpu composition - defines the external interface and config structures
- userguide.pdf - This document

Functional Flow Diagram



composition - OpenGLES2.0 based compositing

Composition is an OpenGL ES 2.0 based application, responsible for compositing video and graphics planes. The graphics planes are alpha blended over the video planes with higher numbered plane being blended on top of the lower numbered planes.

It receives commands for graphics and video configuration and video data buffer pointers via named pipes.

> Named Pipe for video config and data - "/opt/gpu-compositing/named_pipes/video_cfg_and_data_plane_X"
>
> Named pipe for grphics config - "/opt/gpu-compositing/named_pipes/video_data_plane_X"

where X denotes the plane number.

Graphics Configuration

The structure below is used for configuring a graphics plane via named pipe.

```
typedef struct
{
   int enable;                      /* 1 - enable the gfx plane; 0 - disable */
   int input_params_valid;          /* 1 - valid i/p parameters; 0 - invalid */
   struct in_g {
       unsigned long data_ph_addr;  /* physical address of the gfx  buffer   */
       int width;                   /* gfx plane width in pixels             */
       int height;                  /* gfx plane height in pixels            */
       unsigned int pixel_format;   /* fourcc pixel format                   */
       int enable_blending;         /* 1 - blending enabled;  0 - disabled   */
       int enable_global_alpha;     /* 1 - global alpha;  0 - pixel alpha    */
```

```
        float global_alpha;          /* global alpha value  [0.0 to 1.0]      */
        float rotate;                /* rotate angle in decimal degrees [-180.0 to 180.0]*/
    } in_g;
   /* output window position and resolution in normalized device co-ordinates */
    int output_params_valid;         /* 1 - valid o/p parameters; 0 - invalid */
    struct out_g {
        float xpos;   /* x position [-1.0 to 1.0] */
        float ypos;   /* y position [-1.0 to 1.0] */
        float width;  /*  width  - [0.0 to 2.0], 2.0 correspond to fullscreen width */
        float height; /*  height - [0.0 to 2.0], 2.0 correspond to fullscreen height */
    } out_g;
} gfxCfg_s;
```

Video Configuration

The structure below is used for video plane configuration and passing the video data pointers.

```
#define MAX_VIDEO_BUFFERS_PER_CHANNEL 16
typedef struct
{
    int config_data;   /* 1 - config   0 - data */
    int buf_index;     /* if data, buffer index */
    int enable;        /* 1 - enable the video plane; 0 - disable */
    /* Video plane config structure */
    struct in {
        float rotate;  /* rotate angle in decimal degrees [-180.0 to 180.0]*/
        int count;     /* Number of video buffers */
        int width;     /* video frame width in pixels */
        int height;    /* video frame height in pixels */
        unsigned int fourcc;    /* pixel format */
        unsigned long phyaddr[MAX_VIDEO_BUFFERS_PER_CHANNEL]; /* Physical addresses of video buffers */
    } in;
    /* output video window position and resolution in normalized device co-ordinates */
    struct out {
        float  xpos;   /* x position [-1.0 to 1.0] */
        float  ypos;   /* y position [-1.0 to 1.0] */
        float  width;  /*  width  - [0.0 to 2.0], 2.0 correspond to fullscreen width */
        float  height; /*  height - [0.0 to 2.0], 2.0 correspond to fullscreen height */
    } out;
} videoConfig_s;
```

Build/Install Steps

- $ cd composition/
- edit rules.make for the toolchain, cmem, Graphics SDK and target filesystem paths
- $ make clean
- $ make
- $ make install

Execution steps

- Boot up EVM with the required memory reserved for cmem module.

e.g.

setenv bootargs 'console=ttyO0,115200n8 root=/dev/nfs nfsroot=172.24.132.46:/home/user/targetfs,nolock rw mem=128M ip=dhcp'

Initial 128M is allocated for the kernel, cmem memory can be beyond this.

- Run init script and insert the cmem module (dependent module, to be built from the source available at the link under the dependencies section)

cd /opt/gpu-compositing/

./init.sh /* Needs to be run only once on the target */

#insmod cmemk.ko "phys_start=0x88000000 phys_end=0x8F800000 pools=1x10000000,1x10000000,1x10000000,1x10000000,1x5000000,1x5000000,1x5000000,1x5000000 allowOverlap=1"

Refer to the section "Overall Execution steps on Target" to know calculating the pool sizes.

./composition &

The screen should look black after executing the above commands. The composition module is now ready to accept the video and graphics configurations from other applications via named pipes.

composition performance

720x480 video (covering full output screen of 800x480) - 88 fps

720x480 video + 800x480 Graphics (Both covering full output screen 800x480) - 63 fps

gpuvsink – GStreamer GPU based video sink

gpuvsink is gstreamer video sink module, responsible for synchronized displaying of the received video frames using gpu (sgx). This works in conjuction with composition application for actual displaying of frames. It uses named pipe as the method for communicating video plane configurable parameters and data to the composition application.

The named pipes are /opt/gpu-compositing/named_pipes/video_cfg_and_data_plane_X, where X denotes the plane number.

Build Steps

- $ cd gpuvsink/
- edit src/rules.make for CMEM PATH variable
- edit doconfigure.sh for prefix path (target installation path)
- $ ./autogen.sh
- Set up the AM SDK build environment

  $ source /home/user/ti-sdk-am335x-evm-05.05.00.00/linux-devkit/environment-setup
- $ ./doconfigure.sh
- $ make
- $ make install /* e.g. install to "/home/user/targetfs/usr/lib/gstreamer-0.10/" directory */

Command Line Option

Multiple options are specified separated with space.

- x-pos <float - x position of the output video window in normalized device co-ordinates (-1.0 to +1.0) default=-.5>
- y-pos <float - y position of the output video window in normalized device co-ordinates (-1.0 to +1.0) default=0.5>
- width <float - width of the output video window in normalized device co-ordinates (0.0 to 2.0) default=1.0>
- height <float - height of the output video window in normalized device co-ordinates (0.0 to 2.0) default=1.0>
- channel-no=<int - video plane number(0 to MAX_VID_PLANES) default=0 >
- rotate=<float - rotate angle in degrees (-180.00 to 180.00) default=0.0>

Usage Examples

#gst-launch filesrc location=/opt/gpu-compositing/streams/AAC_HEv1/2012_AAC_HE_720x480_24fps.mp4 ! decodebin2 name=dec ! gpuvsink x-pos=-1.0 y-pos=1.0 width=2.0 height=2.0 channel-no=0 rotate=0.0 &

#gst-launch filesrc location=streams/test2.mp4 ! decodebin2 name=dec ! gpuvsink x-pos=-0.4 y-pos=0.0 width=1.0 height=1.0 channel-no=1 rotate=25.0 &

Display Snapshot:

linuxfbofs - qt display driver for rendering to offscreen surface

linuxfbofs is a Qt display driver, which supports rendering to an offscreen surface. It allocates physically contiguous memory using CMEM module for the offscreen surface with the same properties as linuxfb. It works in conjunction with the composition, which is basically an OpenGL ES 2.0 based compositing application. The physically contiguous memory for the offscreen surface is required as it's shared across processes. It uses named pipe as the method for communicating the offscreen buffer and it's properties to the composition application. The named pipe used for this purpose are /opt/gpu-compositing/named_pipes/gfx_cfg_plane_X, where X indicates the plane number. It uses gfxCfg_s (gpucomp.h) structure for communicating the surface properties.

Build Steps

- Set up the Qt build environment

  e.g. run $source /home/user/ti-sdk-am335x-evm-05.05.00.00/linux-devkit/environment-setup
- Edit the Qt project file (linuxfbofs.pro) for the CMEM inlcude and library paths

  e.g. INCLUDEPATH +=/home/user/cmem/include

  LIBS += /home/user/cmem/lib/cmem.a470MV
- Generate Makefile from the Qt project file

  e.g. $/home/user/ti-sdk-am335x-evm-05.05.00.00/linux-devkit/bin/qmake -o Makefile linuxfbofs.pro
- $make
- $make install /* set up the installation path to copy libqscreenlinuxfbofs.so file on to the target - /usr/lib/*qt plugins directory* (e.g. /usr/lib/qtopia/plugins/gfxdrivers/)

Specifying the Driver

To specify the driver set the display environment variable as below

export QWS_DISPLAY=linuxfbofs[:<driver specific options>]

Alternatively, The QWS_DISPLAY environment variable can be set using the -display option when running an application. e.g.

#./app -qws -display linuxfbofs[:<driver specific options>]

Driver Specific Options

The multiple driver options are specified with each separated with a colon

- gfx_no=<int - graphics plane number(0 to MAX_GFX_PLANES) default=0 >
- xpos=<float - x position of the output graphics window in normalized device co-ordinates (-1.0 to +1.0) default=-1.0>
- ypos=<float - y position of the output graphics window in normalized device co-ordinates (-1.0 to +1.0) default=1.0>
- width=<float- width of the output graphics window in normalized device co-ordinates (0.0 to 2.0) default=2.0>
- height=<float- height of the output graphics window in normalized device co-ordinates (0.0 to 2.0) default=2.0>
- blend_en=<int- 1-enable 0-disable blending default=1>
- glob_alpha_en=<int- 1 - global alpha 0 - pixel level alpha default=1>
- global_alpha=<float- value of the global alpha (0.0 to 1.0), valid if glob_alpha_en=1 default=0.5 >
- rotate=<float- rotate angle in degrees (-180.00 to 180.00) default=0.0>

All the float parameters must be specified in x.x format only even if the value is '0' after the decimal point.

Blend Equation

Cf = Cg * alpha + (1-alpha) Cb

Cf - Final color

Cb - Background color

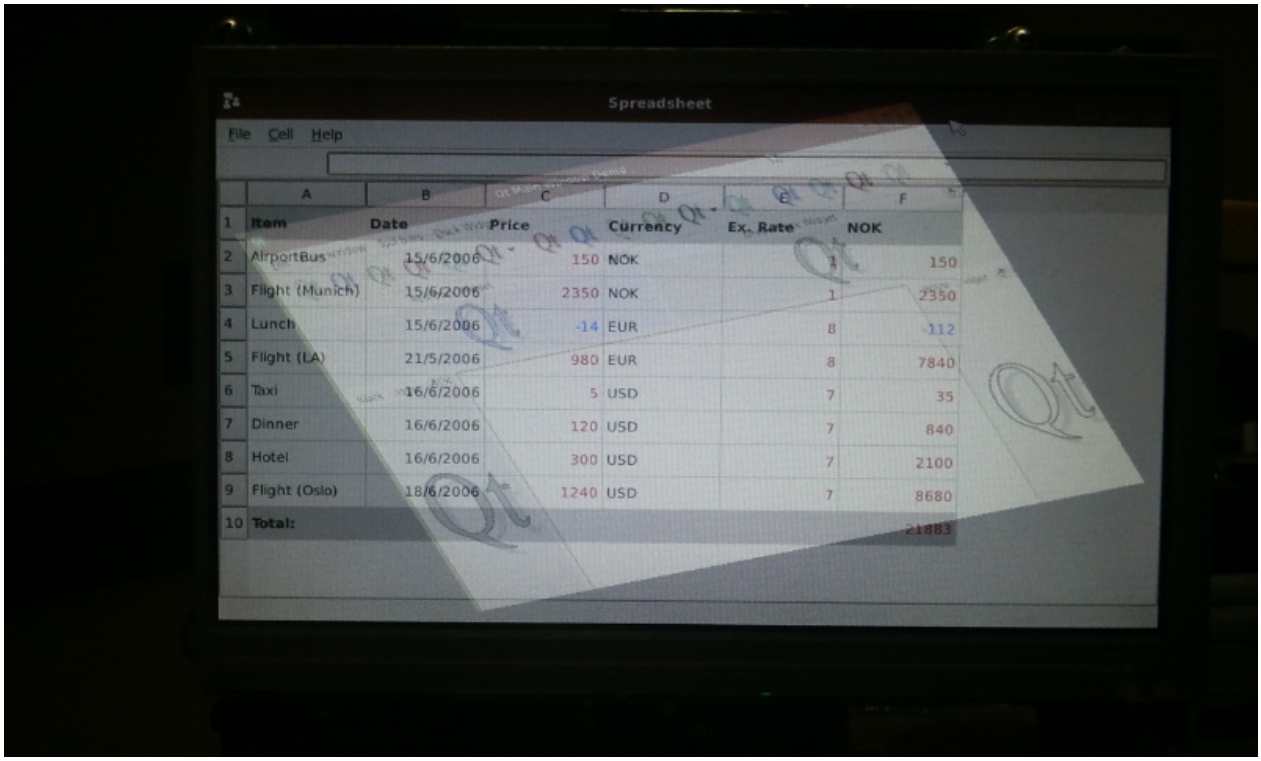
Cg - Graphics color

Usage Examples

#/usr/bin/qtopia/demos/mainwindow/mainwindow -qws -display linuxfbofs:gfx_no=0:xpos=-0.75:ypos=0.75:width=1.5:height=1.5:blend_en=0:rotate=20.0 &

#/usr/bin/qtopia/demos/spreadsheet/spreadsheet -qws -display linuxfbofs:gfx_no=1:xpos=-1.0:ypos=1.0:width=2.0:height=2.0:blend_en=1:glob_alpha_en=1:global_alpha=0.5:rotate=0.0 &

Display Snapshot:



transBgSmpQtUI

A sample Qt application, demonstrating a simple UI with transparent background, alpha channel set to '0' for the background plane.

Refer to the source for more details.

Overall Execution steps on Target

- run the init script, required to be run only once. Creates named pipes needed for communicating across modules.

  #./init.sh

- set the Window System EGL for flip mode, content of the "/etc/powervr.ini" should be as below

  [default]

  #WindowSystem=libpvrPVR2D_FRONTWSEGL.so

  WindowSystem=libpvrPVR2D_FLIPWSEGL.so

- Insert the bufferclass (texture streaming extn module) module if it's not inserted already

  #insmod /opt/gfxlibraries/gfx_rel_es8.x/bufferclass_ti.ko

- insert cmem module, allocating memory for each plane.

  Pool size for Graphics Plane = width * height * Bytes Per Pixel

  Pool size for Video Plane = video frame width * height * 2 (Bytes Per Pixel) * 6 (buffers)

  #insmod cmemk.ko "phys_start=0x88000000 phys_end=0x8F800000 pools=1x10000000,1x10000000,1x10000000,1x10000000,1x5000000,1x5000000,1x5000000,1x5000000 allowOverlap=1"

- Run the composition application

  # ./composition &

- Set up the Graphics planes -

e.g. for global alpha

#/usr/bin/qtopia/demos/spreadsheet/spreadsheet -qws -display linuxfbofs:gfx_no=0:xpos=-1.0:ypos=1.0:width=2.0:height=2.0:blend_en=1:glob_alpha_en=1:global_alpha=0.25:rotate=0.0 &

e.g. for pixel level alpha

#./transBgSmpQtUI -qws -display linuxfbofs:gfx_no=1:xpos=-1.0:ypos=1.0:width=2.0:height=2.0:blend_en=1:glob_alpha_en=0:rotate=0.0 &

- Set up the gstreamer video

  e.g.

  gst-launch filesrc location=/opt/gpu-compositing/streams/AAC_HEv1/2012_AAC_HE_720x480_24fps.mp4 ! decodebin2 name=dec ! gpuvsink x-pos=-1.0 y-pos=1.0 width=2.0 height=2.0 channel-no=0 rotate=0.0 &

Display snapshot:

Limitations/Known Issue

- Color distortion in video planes if formats selected across channels results in different Bytes Per Pixel Size
- A channel (Graphics/Video) can not be re-opened with different resolution
- Running two Qt applications on the same display may not always work fine as it's not a practical usage. The demos showing two qt applications are just for illustrating the composition of multiple graphics planes
- Sanity tested only. Very limited testing for the number of planes more than 2
- GStreamer throws an error "There may be a timestamping problem, or this computer is too slow" along with display artifact when CPU is loaded more than 100%
- Input controls like touch screen and cursor position in Qt framework maps to the full screen and not the resized UI
- Tested on AM335x platform only

References

[1] https://github.com/murthygp/gpu-compositing

[2] http://software-dl.ti.com/dsps/dsps_public_sw/am_bu/sdk/AM335xSDK/latest/index_FDS.html

[3] http://software-dl.ti.com/dsps/dsps_public_sw/sdo_sb/targetcontent/gfxsdk/4_06_00_03/index_FDS.html

[4] http://software-dl.ti.com/dsps/dsps_public_sw/sdo_sb/targetcontent/linuxutils/index.html

Article Sources and Contributors

GPU Compositing *Source*: http://processors.wiki.ti.com/index.php?oldid=119155 *Contributors*: Mahesh

Image Sources, Licenses and Contributors

File:gpu-compositing-directory.jpg *Source*: http://processors.wiki.ti.com/index.php?title=File:Gpu-compositing-directory.jpg *License*: unknown *Contributors*: Mahesh

File:gpu-comp-BlockDiagram-upd.jpg *Source*: http://processors.wiki.ti.com/index.php?title=File:Gpu-comp-BlockDiagram-upd.jpg *License*: unknown *Contributors*: Mahesh

File:gpu-comp-2vidplanes800x480_480x214.JPG *Source*: http://processors.wiki.ti.com/index.php?title=File:Gpu-comp-2vidplanes800x480_480x214.JPG *License*: unknown *Contributors*: Mahesh

File:gpu-comp-2gfxplanes.JPG *Source*: http://processors.wiki.ti.com/index.php?title=File:Gpu-comp-2gfxplanes.JPG *License*: unknown *Contributors*: Mahesh

File:gpu-comp-final.JPG *Source*: http://processors.wiki.ti.com/index.php?title=File:Gpu-comp-final.JPG *License*: unknown *Contributors*: Mahesh

License

EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Adaptations or Collections from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.
b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

a. Each time You Distribute or Publicly Perform the Work or a Collection, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.
b. Each time You Distribute or Publicly Perform an Adaptation, Licensor offers to the recipient a license to the original Work on the same terms and conditions as the license granted to You under this License.
c. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.
d. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.
e. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.
f. The rights granted under, and the subject matter referenced, in this License were drafted utilizing the terminology of the Berne Convention for the Protection of Literary and Artistic Works (as amended on September 28, 1979), the Rome Convention of 1961, the WIPO Copyright Treaty of 1996, the WIPO Performances and Phonograms Treaty of 1996 and the Universal Copyright Convention (as revised on July 24, 1971). These rights and subject matter take effect in the relevant jurisdiction in which the License terms are sought to be enforced according to the corresponding provisions of the implementation of those treaty provisions in the applicable national law. If the standard suite of rights granted under applicable copyright law includes additional rights not granted under this License, such additional rights are deemed to be included in the License; this License is not intended to restrict the license of any rights under applicable law.