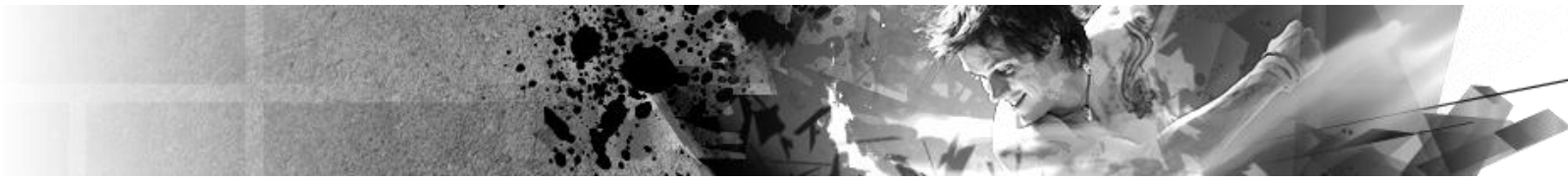
EXTREME REALITY

Extreme Motion SDK
C API – Hands On

Extreme Reality

8/15/2013



EXTREME REALITY

EXTREME MOTION

Table of Contents

1 Overview 2
2 Image Acquisition 2
3 Initializing the Extreme Motion SDK 3
4 Starting Frame Streaming 3
5 Waiting for Any Frame 3
6 Waiting For All Frames – Synchronizing Output 3
7 Getting the Latest Frame from a Given Set of Streams 4
8 Releasing Received Frames 4
9 Stopping Frame Streaming 4
10 Resetting the System – Introducing a New User 4
11 Shutting the System Down 4

1 Overview

The Extreme Motion C API is a set of blocking "C-style" calls designed to offer low latency data access to the Extreme Motion SDK output. While some API calls may perform lengthy operations and offer an optional timeout mechanism, some return immediately. All API calls are context-unaware, and thus, thread-unsafe, be sure to handle any context switching yourself.
The Extreme Motion C API introduces the concept of streams as its data source and frames as its data-token. The API currently exposes the **EM_STREAM_RAW_IMAGE, EM_STREAM_SKELETON, EM_STREAM_WARNING** and **EM_STREAM_GESTURES** streams, which, in-turn, publish the SRawImageFrame, SSkeletonFrame, SWarningsFrame and SGesturesFrame frames, respectively.

- SRawImageFrame holds RGB raw image captured by the camera.
- SSkeletonFrame holds the coordinates of the skeleton joints.
- SWarningsFrame holds warnings regarding run-time conditions that may hinder performance. Be sure to use it and alert the user where possible.
- SGesturesFrame holds a list of gestures that were detected in a specific frame.

A session is the duration between initializing the system and shutting it down (see 2 and 11 respectively). During a session any and all of the system streams may become started or stopped (see 4 and 9 respectively). Whenever a given stream is started and a frame is received (see 5, 6, 7), it is locked until explicitly released (see 8); failing to release frames may degrade performance, ultimately exhausting the system memory.
Follow this document to discover the various operations available via Extreme Motion C API calls. For further information, review the API reference and the documented samples.

2 Image Acquisition

Extreme Motion SDK captures images directly from the camera and processes them. Developers may change this behavior and capture images on their own by modifying the “Source” field in the supplied configuration file, ConfigBase.xml:

```
<param name="Source" value="External" />
```

Once the configuration file is modified, the developer is responsible for supplying images to Extreme Motion SDK by calling emHandleImage method. Calling emHandleImage method without setting the configuration file as shown above will result with an error.
When using an external image source, the developer can control the way injected images are handled using the parameter "SyncExternalSource" in the configuration file ConfigBase.xml:

- Setting the parameter to “false” (which is the default mode):
  `<param name="SyncExternalSource" value="false" />`
  Will configure emHandleImage method to run in a non-blocking mode, meaning the method will return immediately but the frame might be dropped.
- Setting the parameter to “true”:

<param name="SyncExternalSource" value="true" />
Will configure emHandleImage method to run in a blocking mode, meaning the method will return only when Extreme Motion SDK has started processing the injected image.

3 Initializing the Extreme Motion SDK

To initialize the system call emInitialize specifying a SImageDescriptor struct and the StreamTypes you wish to receive frames from. You may use **S_IMAGE_DESCRIPTOR_INITIAL_VALUE** as your SImageDescriptor; it is set to best match the concrete platform you are running on. You may safely call this several times, if the system is already initialized you will receive an error return value. To reconfigure the system, call emShutdown to terminate the session (see 11). Also note that the last parameter may be safely set to **NULL**.

4 Starting Frame Streaming

To start streaming frames from the streams you initialized (see 2), call emStartStreams specifying a bitwise-OR combination of **EM_STREAM** constants you used when calling emInitialize. Starting the **EM_STREAM_RAW_IMAGE** stream acquires the camera and thus may prove to be a lengthy operation. Starting the **EM_STREAM_SKELETON, EM_STREAM_WARNING** or **EM_GESTURES_STREAM** streams returns promptly but asynchronously waits until valid SRawImageFrame are available, and starts processing them. You may safely call this several times, if a stream is already running, its start command will be ignored.

5 Waiting for Any Frame

To receive any frame when it is first available, you may call emWaitForAndLockAnyFrame specifying a bitwise-OR combination of **EM_STREAM** constants you wish to wait on and the duration, in milliseconds, to wait for. Upon **EM_STATUS_OK** return, resultStream will hold the received StreamType, and frameAddress, the reference to the frame data. When you are done using the resulting frame, make sure to call emUnlockFrames specifying the value provided for streamstoWaitOn and the one received for framesAddress (see 8).

6 Waiting For All Frames – Synchronizing Output

If you wish to receive notice only when a SRawImageFrame and its resulting frames are ready, i.e. get synchronized output, you may call emWaitForAndLockAllFrames specifying the duration, in milliseconds, you wish to wait for. Upon **EM_STATUS_OK** return, frameAddresses will hold the references to the frame data. The frame order corresponds to the compacted bit-order in the specified streamstoWaitOn, i.e. if waiting for all frame types, the first entry in frameAddresses will hold a reference to the SRawImageFrame, the second to its resulting SSkeletonFrame, and so on. When you are done using the resulting frames, make sure to call emUnlockFrames

specifying the value provided for streamstoWaitOn and the one received for frameAddresses (see 7).

7 Getting the Latest Frame from a Given Set of Streams

If you wish to receive the latest available frame for a given set of streams, if all are available, and without waiting for more recent data, you may call emLockCurrentFrames specifying a bitwise-OR combination of **EM_STREAM** constants you wish to receive frames for. Please note, that you will receive any available frames, not necessarily the ones originating in the received SRawImageFrame in case the **EM_STREAM_RAW_IMAGE** was specified. Upon **EM_STATUS_OK** return, frameAddresses will hold the references to the frame data. The frame order corresponds to the compacted bit-order in the specified streamstoWaitOn, i.e. if waiting for all frame types, the first entry will hold a reference to the SRawImageFrame, the second to its resulting SSkeletonFrame, and so on. When you are done using the resulting frames, make sure to call emUnlockFrames specifying the value provided for streamFlags and the one received for frameAddresses (see 7).

8 Releasing Received Frames

Once you are done using any frames, make sure to call emUnlockFrames specifying the frame types you used, as well as the addresses from which they were obtained. Failing to do so will promptly result in degraded system performance, and ultimately in process failure due to memory exhaustion.

9 Stopping Frame Streaming

In order to stop receiving data frames from streams, call emStopStreams with a bitwise-OR combination of **EM_STREAM** constants you wish to stop. Please note that this will cause a system-wide stopping of the specified streams, as the CAPI is context-unaware. You may safely call this several times, if a stream is already stopped, its stop command will be ignored.

10 Resetting the System – Introducing a New User

When introducing a new user to the system, or whenever leaving the scene for a considerable duration, it is best to "reset" the system. This can be achieved by stopping the **EM_STREAM_SKELETON** stream, then starting it again (see 8 and 4, respectively) 9.

11 Shutting the System Down

To terminate a session, call emShutdown. You may safely call this several times.