



EyeMoRocknRoll Manual

EyeMoRocknRoll Version 1.0.0
Document Version 1.0.0

© 2006 GestureTek Inc.
317 Adelaide Street West, Toronto, Ontario, M5V 1P9 Canada
Web: <http://www.gesturetek.com> Email: support@gesturetek.com
Phone: 416.340.9290 Fax: 416.348.9809

Table of Contents

1	Introduction	1
1.1	Copyright and Legal Matters.....	1
1.2	Prerequisites.....	1
1.3	Device requirements.....	1
1.4	Overview.....	2
1.4.1	Roll Motion	2
1.4.2	Rock Motion	4
2	Methods	6
2.1	IEYEMOROCKNROLL_GetParam.....	6
2.2	IEYEMOROCKNROLL_GetResult.....	7
2.3	IEYEMOROCKNROLL_GetResultRange.....	7
2.4	IEYEMOROCKNROLL_Initialize.....	8
2.5	IEYEMOROCKNROLL_ProcessFrame.....	9
2.6	IEYEMOROCKNROLL_Reset.....	10
2.7	IEYEMOROCKNROLL_SetParam.....	11
3	Data Reference	12
3.1	EYEMOROCKNROLL_DATA_TRACKER_STATUS.....	12
3.2	EYEMOROCKNROLL_DATA_GESTURE.....	12
3.3	EYEMOROCKNROLL_DATA_ACCROLLX,EYEMOROCKNROLL_DATA_ACCROLLY.....	13
3.4	EYEMOROCKNROLL_DATA_IMMROLLX,EYEMOROCKNROLL_DATA_IMMROLLY.....	13
4	Parameter Reference	14
4.1	Flip	14
4.2	Low Detail Threshold.....	14
4.3	Search Width and Search Height.....	15
4.4	Start Delay and Start Brightness Threshold.....	15
4.5	Gesture Maximum Pause Ticks.....	16
4.6	Gesture Maximum Total Ticks.....	16
4.7	Gesture Minimum Accept Ticks.....	16
4.8	Gesture Motion Threshold.....	17
4.9	Gesture Track Direction.....	17

1 Introduction

EyeMoRocknRoll transforms a camera-enabled mobile device into a controller. Games and applications can be controlled by simply moving the device. There is no need to use the handset keypad.

The EyeMoRocknRoll extension provides an easy-to-use BREW interface that makes camera-based motion tracking available to BREW applications. It supports two types of tracking: *Rock*, where the handset is flicked rapidly in one direction and immediately returned to its original orientation; and *Roll*, where the tracker attempts to determine changes in handset orientation. Rock and Roll are described in more detail in the Overview [\[2 \]](#) section. When non-directional motion tracking (i.e. shaking the device) is desired, the EyeMoShake BREW extension is also available.

EyeMoRocknRoll can be used in applications where analog input is required, such as rolling a ball through a maze, positioning a map, or smoothly scrolling a menu.

While EyeMoRocknRoll provides a look and feel similar to a joystick, it is essential to understand that EyeMoRocknRoll does not work like a joystick, which uses absolute position, but more like a mouse, using relative motion.

It is recommended that EyeMoRocknRoll be used in conjunction with the EyeMoCamera extension. EyeMoCamera simplifies camera access by handling device-specific initialization and implementation complexities that may occur, such as devices that contain more than one camera, various camera orientations, devices supporting only a subset of common resolutions, and those devices which require unusual initialization steps. For more information regarding the EyeMoCamera extension, see the EyeMoCamera documentation.

1.1 Copyright and Legal Matters

GestureTek and EyeMobile are registered trademarks of GestureTek Inc.

BREW is a registered trademark of QUALCOMM Incorporated.

GestureTek technology and software is protected under United States patent number 5,534,917 (Video image based control system).

The contents of this documentation are the property of GestureTek Inc.

1.2 Prerequisites

Familiarity with BREW development in the C programming language.

1.3 Device requirements

- Device with BREW version 2.1.3 or above.
- Device camera should be available to BREW (device specifications are available from the BREW Developer extranet).
- Color or grey-scale displays are supported.
- Frame rate must be at least 15 frames per second.
- Minimum supported resolution is 80x60 pixels.

- For best results the camera image should represent the entire field-of-view of the camera, rather than a sub-window of the original view.
- A small amount of memory is required for EyeMoCamera to allocate its internal buffers.
- BREW Simulator (included with BREW SDK version 3 and above) is required to run camera-enabled BREW applications directly from Microsoft Visual Studio.
- Running EyeMoRocknRoll-enabled applications on the BREW Simulator requires a webcam.

1.4 Overview

The normal procedure when using EyeMoRocknRoll is as follows:

- Create a new instance of the extension using ISHELL_CreateInstance and pass it the classID AEECLSID_EYEMOROCKNROLL.
- Initialize the extension with the frame size and the image format of the incoming images using EYEMOROCKNROLL_Initialize^[8].
- Pass frames into the extension using EYEMOROCKNROLL_ProcessFrame^[9].
- Query the data that the extension has computed using EYEMOROCKNROLL_GetResult^[7] for any of the data channels that the tracker supports.
- Modify the behavior of the extension by changing one of the exposed parameters using EYEMOROCKNROLL_SetParam^[11].
- Reset the tracker as needed using EYEMOROCKNROLL_Reset^[10].
- And finally, release the extension when it is no longer being used, with EYEMOROCKNROLL_Release.

A number of the above steps are automated when using EyeMoCamera. The application no longer needs to call EYEMOROCKNROLL_Initialize, or EYEMOROCKNROLL_ProcessFrame after the EyeMoRocknRoll has been passed to IEYEMOCAMERA_AddFrameListener. For more information regarding the EyeMoCamera extension, see the EyeMoCamera documentation.

1.4.1 Roll Motion

The EyeMoRocknRoll extension can track the Roll motion of the device. This is designed for gentle, precise, analog control, and could be used to for such things as the rolling of a virtual ball, thrusters on a spaceship, etc.

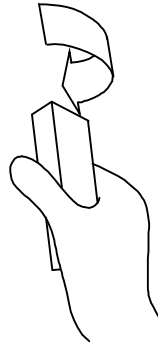
Supported movement

Roll motion supports movement independently in the X and Y axes as illustrated here:

X Axis

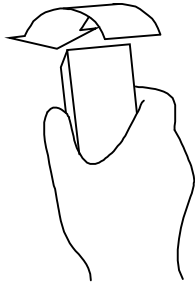


Turning the handset to the left.

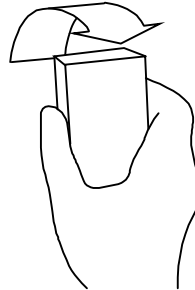


Turning the handset to the right.

Y Axis



Tilting the top of the device away from you.



Tilting the top of the device toward you.

How Roll works

As the device is tilted the tracker estimates the distance that the device has moved between consecutive video frames. This is referred to as immediate motion. Motion is measured in pixel units with respect to the camera image. It is returned as a fixed-point value with an 8-bit decimal, since many BREW phones do not have native hardware support for floating-point. The tracker is designed to provide smooth motion via sub-pixel estimation, which helps to reduce jitter between discrete pixel values.

Accumulating immediate motion can be used to determine the total motion over a longer span of time. However it is essential to note that accumulated motion is not the same as absolute position. The tracker does not use fixed starting points like a joystick. Instead it calculates motion on the basis of frame-by-frame comparison. Therefore using accumulated motion in the attempt to calculate current position is not recommended.

The speed in which the user may tilt or turn the device is limited by the frame-rate and field-of-view of the camera, since consecutive video frames must contain at least some overlapping portion of the image. A warning will be returned if the user moves the device too fast.

For best results, the video frames should contain a static scene containing distinctive visual elements. Output values are not reliable when the camera is being used in low-light situations, high brightness situations (camera pointed to a bright-beam of light), and lack of feature-rich environment (camera pointed to a blank surface). The tracker can track scenes with little detail

or texture, however the accuracy of the results may be reduced. A warning will be returned when the detail is below a specified threshold.

1.4.2 Rock Motion

A Rock gesture is similar to a roll gesture, except that the device is returned to its initial position.

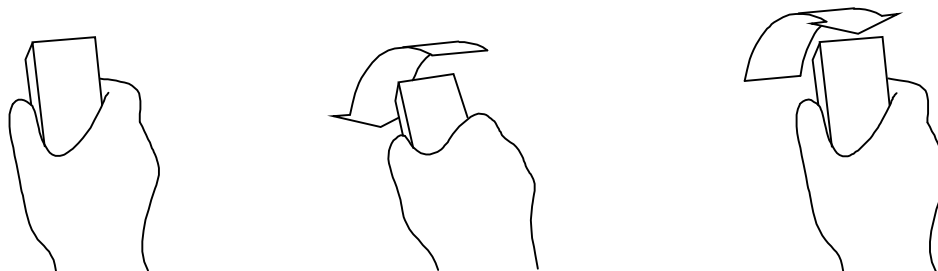
The inclusion of a return motion helps assure that only purposeful gestures are detected, while random and arbitrary motions of the device are ignored.

The gesture can be performed as one fluid motion, or with a slight pause between the two motions. The system will report the gesture as soon as the return motion is detected, and usually before the motion has concluded. This allows the application to respond while the device is still in motion, for a sense of immediacy.

Below is an example of how to perform each gesture:

EYEMOROCKNROLL_GESTURE_UP

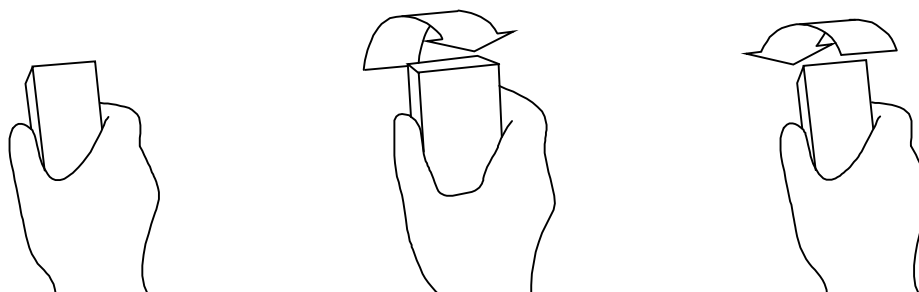
Note: The following illustration is correct. At first glance it may seem to be the reverse of what you might expect.



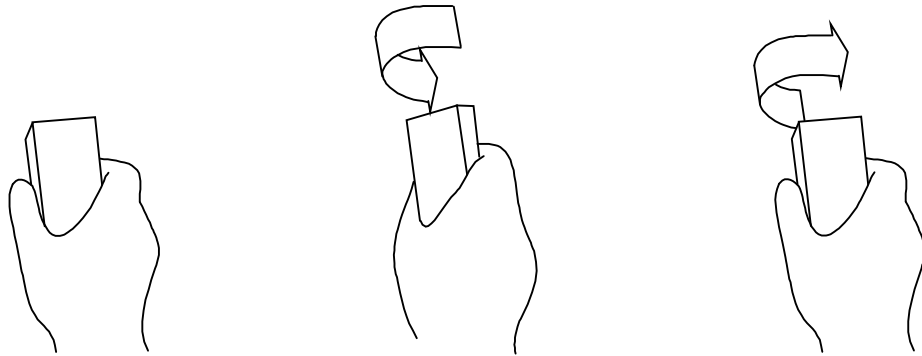
1. Start at the neutral position.
2. Tilt the device away from yourself by bending your wrist.
3. Tilt the device back to the neutral position.

EYEMOROCKNROLL_GESTURE_DOWN

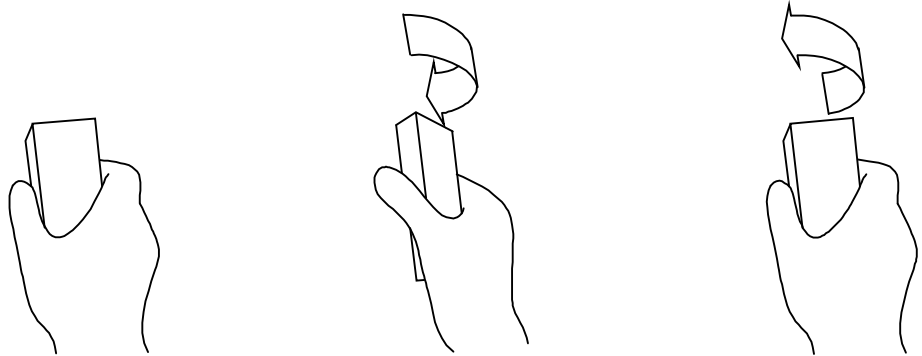
Note: The following illustration is correct. At first glance it may seem to be the reverse of what you might expect.



1. Start at the neutral position.
2. Tilt the device towards yourself by bending your wrist.
3. Tilt the device back to the neutral position.

EYEMOROCKNROLL_GESTURE_LEFT

1. Start at the neutral position.
2. Tilt the device to the left by rotating your wrist.
3. Tilt the device back to the neutral position.

EYEMOROCKNROLL_GESTURE_RIGHT

1. Start at the neutral position.
2. Tilt the device to the right by rotating your wrist.
3. Tilt the device back to the neutral position.

2 Methods

2.1 IEYEMOROCKNROLL_GetParam

Returns the value of a parameter that is associated with the given ID. This function can be used to observe the low-level settings of the tracker.

More details about these parameters can be found under the Parameter Reference [\[14\]](#) section.

Definition

```
int  
IEYEMOROCKNROLL_GetParam(  
    IEyeMoRocknRoll * po,  
    int32 nParamID,  
    void * pCurrent,  
    void * pDefault,  
    void * pMin,  
    void * pMax  
);
```

Parameters

po

Pointer to an instance of the extension interface object.

nParamID

The ID of the parameter that is to be queried. The IDs of the possible parameters are given by an enumerated type. Please refer to section, Parameter Reference [\[14\]](#), for more details.

pCurrent (output)

A pointer to the memory location where the current value of the parameter will be stored. It is the application's responsibility to allocate the appropriate amount of space to store the return value. If this field is NULL, the function will not return the current value for the specified parameter.

pDefault (output)

A pointer to the memory location where the default value of the parameter will be stored. It is the application's responsibility to allocate the appropriate amount of space to store the return value. If this field is NULL, the function will not return the default value for the specified parameter.

pMin (output)

A pointer to the memory location where the minimum value of the data stream will be stored. It is the application's responsibility to allocate the appropriate amount of space to store the return value. If this field is NULL, the function will not return the minimum value for the specified parameter.

pMax (output)

A pointer to the memory location where the maximum value of the data stream will be stored. It is the application's responsibility to allocate the appropriate amount of space to store the return value. If this field is NULL, the function will not return the maximum value for the specified parameter.

Returns

EYEMO_OKAY

The tracker has returned the parameter without warnings.

EYEMO_NOT_SUPPORTED

The ID of the data stream given by *nParamID* is invalid.

EYEMO_ERROR

The tracker has generated an error.

EYEMO_ERROR_NOT_INITIALIZED

The extension has not yet been initialized.

Notes

None

2.2 IEYEMOROCKNROLL_GetResult

Returns the current data value of the data stream associated with the given ID. The set of data values that can be returned are discussed in detail in the Data Reference^[12] section. This function can be called multiple times per frame, and the values returned by the function are only updated after each call to IEYEMOROCKNROLL_ProcessFrame^[9].

Definition

```
int
IEYEMOROCKNROLL_GetResult(
    IEyeMoRocknRoll * po,
    int32 nDataID,
    void * pValue
);
```

Parameters

po

Pointer to an instance of the extension interface object.

nDataID

The ID of the data stream that is to be queried. The IDs of the possible data streams are given by an enumerated type and are extension-specific.

pValue (*output*)

A pointer to the variable that receives the returned data value. It is the application's responsibility to allocate a structure of the appropriate size for this function to store the return value.

Returns

EYEMO_OKAY

The tracker has returned the data without warnings.

EYEMO_NOT_SUPPORTED

The ID of the data stream given by nDataID is invalid.

EYEMO_ERROR_NOT_INITIALIZED

The extension has not yet been initialized.

EYEMO_ERROR

The tracker has generated an error.

EYEMO_WARNING_NO_DATA

The tracker has not yet generated any motion data. This is returned on the first few frames, until the tracker has analyzed sufficient information to generate motion data.

Notes

None

2.3 IEYEMOROCKNROLL_GetResultRange

Returns the usable range of the given data stream. If the range of the given data stream is dependent on the current parameters or initialization settings of the extension, the values returned by this function will always contain the correct range for the current settings. If the settings changes later, another call to IEYEMOROCKNROLL_GetResultRange^[7] will be necessary.

Definition

```
int
IEYEMOROCKNROLL_GetResultRange(
    IEyeMoRocknRoll * po,
    int32 nDataID,
    void * pMin,
    void * pMax
);
```

Parameters

po

Pointer to an instance of the extension interface object.

nParamID

The ID of the parameter that is to be queried. The IDs of the possible parameters are given by an enumerated type, and are extension-specific.

pMin (output)

A pointer to the memory location where the minimum value of the data stream will be stored. It is the application's responsibility to allocate the appropriate amount of space to store the return value.

pMax (output)

A pointer to the memory location where the maximum value of the data stream will be stored. It is the application's responsibility to allocate the appropriate amount of space to store the return value.

Returns

EYEMO_OKAY

The tracker has returned the parameter without warnings.

EYEMO_NO_RANGE

The current data stream does not have a meaningful range associated with it. This can occur when the type of the data stream does not have a meaningful order defined on it, as is the case with a structure or an enumerated type.

EYEMO_NOT_SUPPORTED

The ID of the data stream given by *nParamID* is invalid.

EYEMO_ERROR_NOT_INITIALIZED

The extension has not yet been initialized.

EYEMO_ERROR

The tracker has generated an error.

Notes

None

2.4 IEYEMOROCKNROLL_Initialize

Initializes the EyeMoRocknRoll extension. This function allocates the necessary buffers and initializes the underlying tracker.

Definition

```
int
IEYEMOROCKNROLL_Initialize(
    IEyeMoRocknRoll * po,
    uint16 nFrameW,
    uint16 nFrameH,
    int32 nColorMode
);
```

Parameters`po`

Pointer to an instance of the extension interface object. This object must be allocated separately using `ISHELL_CreateInstance` before calling this function. Use the constant `EECLSID_EYEMOROCKNROLL` as the `CLSID` value for the instance.

`nFrameW, nFrameH`

The dimensions of the video frames that will be passed to the extension for processing.

`nColorMode`

The color mode of the incoming video frames.

Returns`EYEMO_OKAY`

No problems were encountered during initialization.

`EYEMO_ERROR_NO_MEMORY`

There was insufficient memory available to allocate the necessary resources.

`EYEMO_ERROR`

One of the components could not be initialized.

`EYEMO_WARNING_ALREADY_INITIALIZED`

The extension was already initialized.

Notes

The `nFrameW` and `nFrameH` parameters define the dimensions of the video frames that will be passed to the `IEYEMOROCKNROLL_ProcessFrame` function. Due to inconsistencies between BREW implementations on different devices, obtaining an accurate frame size prior to starting the camera is not very reliable. The most accurate way of finding out the actual frame size is to start the camera and wait until you receive the first frame. Then you can use the dimensions of this frame as values for these parameters.

However, if you are designing for a small subset of devices and can reliably predict the camera frame size, you can call this function prior to starting the camera without waiting for the first frame.

The `nColorMode` will need to be set to match the color format that the device is using. Since initialization is usually deferred until the first frame of video is received, the color mode can be determined by casting the frame to an `IDIB` and retrieving the `nColorScheme` field. EyeMoRocknRoll supports a number of color modes, but the value used will typically be one of the following (which are simply aliases to the appropriate platform enums):

```
EYEMO_COLORMODE_X1R5G5B5
EYEMO_COLORMODE_R5G6B5
EYEMO_COLORMODE_X14R6G6B6
EYEMO_COLORMODE_R8G8B8
```

2.5 IEYEMOROCKNROLL_ProcessFrame

Called whenever a new video frame is captured. The calculated results are stored by the extension and can be retrieved by the application using `IEYEMOROCKNROLL_GetResult`.

Definition

```
int
IEYEMOROCKNROLL_ProcessFrame(
    IEyeMoRocknRoll * po,
    IBitmap * pFrame,
    uint32 nElapsedTicks
);
```

Parameters`po`

Pointer to an instance of the extension interface object.

`pFrame`

A pointer to the captured frame. You can pass the same pointer returned by the camera callback function.

`nElapsedTicks`

The number of milliseconds that have elapsed since the last frame was produced by the camera.

Returns`EYEMO_OKAY`

The tracker has processed the motion without warnings.

`EYEMO_WARNING_NO_DATA`

The tracker has not yet generated any motion data. This is returned on the first few frames, until the tracker has analyzed sufficient information to generate motion data.

`EYEMO_WARNING_STARTUP`

This warning is generated whenever the tracker is still in the process of starting up. When the device camera first starts, the first few frames captured are not very reliable. The results generated by the tracker will, therefore, not be reliable. During this time results will be generated by their accuracy cannot be guaranteed.

`EYEMO_WARNING_LOW_DETAIL`

This warning is generated whenever there is very little detail within the image, such as when the camera is pointing at a nearly blank wall or floor. Motion will still be estimated, though the calculated results may be less reliable. This threshold can be set by the Low Detail parameter.

`EYEMO_WARNING_TOO_FAST`

This warning is generated whenever consecutive frames do not appear to have any overlap. The most likely cause of this is that the device has been moved too fast. Motion cannot be determined if there is no overlap. Motion results will be zero when this result is returned.

`EYEMO_ERROR_PAUSED`

This warning is generated if the function is called while the tracker is paused.

`EYEMO_ERROR_INVALID_IMAGE`

This warning is generated when the frame being input is invalid. This can occur if the pointer is NULL, or if the color mode or dimensions of the frame do not match the parameters used to initialize the tracker.

`EYEMO_ERROR`

The tracker has generated an error.

Notes

None

2.6 IEYEMOROCKNROLL_Reset

Recenter the tracker.

Definition

```
int
IEYEMOROCKNROLL_Reset (
    IEyeMoRocknRoll * po
);
```

Parameters`po`

Pointer to an instance of the extension interface object.

Returns

EYEMO_OKAY

The tracker was successfully reset.

EYEMO_ERROR_NOT_INITIALIZED

The extension has not yet been initialized.

EYEMO_ERROR

An unexpected error has occurred.

Notes

Resetting the tracker sets all tracking values to zero. The application should recenter the tracker at the start of a game or level, as well as allowing the user to press a key to recenter the device, so that the tracker can be reset when the position of the device has been changed, such as when the device is handed to another person.

2.7 IEYEMOROCKNROLL_SetParam

Sets the value of the tracking parameter that is associated with the given ID. This function can be used to change some of the low-level settings of the tracker.

The set of parameters that can be set are discussed in detail under the Parameter Reference [\[14\]](#) section.

Definition

```
int
IEYEMOROCKNROLL_SetParam(
    IEyeMoRocknRoll * po,
    int32 nParamID,
    void * pValue
);
```

Parameters

po

Pointer to an instance of the extension interface object.

nParamID

The ID of the parameter to be set. The IDs of the possible parameters are given by an enumerated type and are extension-specific.

pValue

A pointer to the value to be assigned to the parameter.

Returns

EYEMO_OKAY

The tracker has returned the parameter without warnings.

EYEMO_NOT_SUPPORTED

The ID of the data stream given by nParamID is invalid.

EYEMO_ERROR_NOT_INITIALIZED

The extension has not yet been initialized.

EYEMO_ERROR

The tracker has generated an error.

Notes

Calling IEYEMOROCKNROLL_Initialize [\[8\]](#) sets all the parameters to their default values, so it is only necessary to call this function to change a parameter to something other than its default value. In most applications, only the Flip and Track directions parameters need to be set.

IEYEMOROCKNROLL_Initialize should be called prior to calling this function. Parameters can be changed at any time thereafter, even before the first call to IEYEMOROCKNROLL_ProcessFrame [\[9\]](#).

3 Data Reference

The EyeMoRocknRoll extension computes the number of pixels that the camera was tilted between two consecutive frames. All of the Roll channels are returned as fixed-point values with 8-bit decimals, since many BREW phones do not have native hardware support for floating-point.

Since the units of this value are in pixels, the potential range of the values is dependent on both the resolution of the camera image and the current value of the search width parameter. Once the extension has been initialized, the IEYEMOROCKNROLL_GetResultRange^[7] function will return the correct range for the data stream.

3.1 EYEMOROCKNROLL_DATA_TRACKER_STATUS

Returns the status reported by the tracker for the last frame processed. If the frame was processed successfully, this value will be identical to the return value of IEYEMOROCKNROLL_ProcessFrame^[9]. The tracker status may indicate that a problem has occurred related to tracking that may affect the accuracy of the tracking results. For instance, if the tracker returns EYEMO_WARNING_TOO_FAST, the application may wish to take different action than when the immediate motion is "(0, 0)".

The actual return value from IEYEMOROCKNROLL_ProcessFrame^[9] may not be available to the application if it is using the EyeMoCamera extension to manage all interaction with the camera. This data channel is provided largely for developer convenience.

Type

Enumerated (*int32*)

Range

EYEMO_OKAY
EYEMO_WARNING_NO_DATA
EYEMO_WARNING_LOW_DETAIL
EYEMO_WARNING_TOO_FAST
EYEMO_WARNING_STATUP
EYEMO_ERROR

3.2 EYEMOROCKNROLL_DATA_GESTURE

Returns the last gesture that was recognized by the tracker since this function was last queried. Possible gestures are Left, Right, Up and Down.

Type

Enumerated (*int32*)

Range

EYEMOROCKNROLL_GESTURE_NONE
EYEMOROCKNROLL_GESTURE_UP
EYEMOROCKNROLL_GESTURE_DOWN
EYEMOROCKNROLL_GESTURE_LEFT
EYEMOROCKNROLL_GESTURE_RIGHT

3.3 EYEMOROCKNROLL_DATA_ACCROLLX, EYEMOROCKNROLL_DATA_ACCROLLY

Returns the amount of tilt in the X or the Y direction that has occurred since this function was last queried. The frame-to-frame tilt values are added together for each camera frame processed. This ensures that no tilt data is ever lost or double-counted, if the application queries the tracker at a rate that is different from the camera frame rate.

Type

Fixed Point, 8-bit Decimal (*int32*)

Alias

EYEMOROCKNROLL_DATA_ROLLX, EYEMOROCKNROLL_DATA_ROLLY

3.4 EYEMOROCKNROLL_DATA_IMMROLLX, EYEMOROCKNROLL_DATA_IMMROLLY

Returns the amount of tilt in the X or the Y direction that has occurred between the previous two frames. Therefore, this data channel will produce values of similar magnitude, if the device is tilted at a constant rate. This method can miss motion occurring between frames or report it more than once, if the application queries the tracker more or less often than the camera frame rate.

Type

Fixed Point, 8-bit Decimal (*int32*)

4 Parameter Reference

The following parameters are implemented for both Roll and Rock:

- Flip
- Low Detail Threshold
- Search Width and Search Height
- Start Delay and Start Brightness Threshold

The following parameters are implemented for Rock only:

- Gesture Maximum Pause Ticks
- Gesture Maximum Total Ticks
- Gesture Minimum Accept Ticks
- Gesture Motion Threshold
- Gesture Track Direction

4.1 Flip

Flip effects the sign of the output values and direction of Rock gestures.

Definition

EYEMOROCKNROLL_PARAM_FLIP

Type

Bit-mask (*int32*)

Value

A combination of the following bit-flags:

EYEMO_FLIP_HORIZONTAL
EYEMO_FLIP_VERTICAL

Default

IEYEMO_FLIP_DEFAULT_FLIP (defined as neither of the above bit-flags.)

Notes

Set the Flip parameter if the camera image is flipped. This may occur if the camera can rotate away or towards the user, or if the image origin (top-left or bottom-left) is opposite from the platform's default.

4.2 Low Detail Threshold

The low detail threshold is used to generate a warning when there is insufficient detail in the camera image to ensure tracker accuracy. This warning does not otherwise affect tracking.

Definition

EYEMOROCKNROLL_PARAM_LOW_DETAIL_THRESHOLD

Type

Integer (*int32*)

Value

Custom

Default

EYEMOROCKNROLL_DEFAULT_LOW_DETAIL_THRESHOLD

Notes

The threshold depends on the device; use a low value for good quality cameras and a high value for poorer quality cameras. Camera quality refers to signal-to-noise ratio, not resolution.

4.3 Search Width and Search Height

The search range that is used to compare consecutive video frames.

Definition

EYEMOROCKNROLL_PARAM_SEARCH_WIDTH
EYEMOROCKNROLL_PARAM_SEARCH_HEIGHT

Type

Integer (*int32*)

Value

Search area is specified in percent (0–100). The valid range is 5–95, representing 5–95 percent.

Default

EYEMOROCKNROLL_DEFAULT_SEARCH_WIDTH
EYEMOROCKNROLL_DEFAULT_SEARCH_HEIGHT

Notes

The larger the value, the larger the area that will be searched when comparing consecutive video frames. For example, if the search range is set to 75 percent, objects within the image may be displaced by up to 75 percent of the image size. However, a large search area allows for less overlap and therefore less confident results. Increasing the search range allows faster movements to be tracked, but also increases the processing time.

4.4 Start Delay and Start Brightness Threshold

A camera with auto-exposure may take time to settle. The tracker does not process motion until either:

- The brightness exceeds the minimum specified by this brightness threshold, or
- The number of frames specified by the start delay is met.

If the brightness criteria is satisfied prior to the frame count (which is usually the case), extra processing will be performed to normalize the camera's image until the frame count is met.

Type

Integer (*int32*)

Definition

EYEMOROCKNROLL_PARAM_START_DELAY
EYEMOROCKNROLL_PARAM_START_BRIGHTNESS_THRESHOLD

Value

Start delay is specified in number of frames.

Brightness threshold is specified in the range of 0–255.

Default

EYEMOROCKNROLL_DEFAULT_START_DELAY
EYEMOROCKNROLL_DEFAULT_START_BRIGHTNESS_THRESHOLD

Notes
None

4.5 Gesture Maximum Pause Ticks

A Rock gesture is composed of two motions. This parameter determines the maximum amount of time between each motion.

Definition
EYEMOROCKNROLL_PARAM_GESTURE_MAX_PAUSE_TICKS

Type
Integer (*int32*)

Value
Elapsed time units (in milliseconds).

Default
EYEMOROCKNROLL_DEFAULT_GESTURE_MAX_PAUSE_TICKS

Notes
A Rock gesture is composed of two motions: a movement in the direction of the flick, and a motion to return the camera to its original position. There may be a slight pause between those motions, the maximum duration of which is determined by this parameter. There is no minimum pause duration; a Rock gesture can be performed as a fluid motion forward and back without any pause.

4.6 Gesture Maximum Total Ticks

A gesture will be ignored if it is too slow. This parameter determines the maximum amount of time a user has to perform a gesture.

Definition
EYEMOROCKNROLL_PARAM_GESTURE_MAX_TOTAL_TICKS

Type
Integer (*int32*)

Value
Elapsed time units (in milliseconds).

Default
EYEMOROCKNROLL_DEFAULT_GESTURE_MAX_TOTAL_TICKS

Notes
A gesture must be completed within the time allotted by this parameter.

4.7 Gesture Minimum Accept Ticks

A gesture can be recognized and accepted while in motion before the user has completed the entire motion. This parameter determines the minimum duration to analyze a motion before attempting to classify it as a gesture.

Definition
EYEMOROCKNROLL_PARAM_GESTURE_MIN_ACCEPT_TICKS

Type

Integer (*int32*)

Value

Elapsed time units (in milliseconds), or you may indicate "0" (zero) to wait until the gesture is complete before reporting it.

Default

EYEMOROCKNROLL_DEFAULT_GESTURE_MIN_ACCEPT_TICKS

Notes

The system can report the gesture as soon as the return motion is detected and, in most cases, before the motion has concluded. This parameter determines exactly how much of the return motion portion of the gesture must be observed before reporting the gesture. This should be a sufficient amount in order to be confident of the gesture. If the user completes a motion sooner than this duration, it will still be accepted and classified.

4.8 Gesture Motion Threshold

The immediate (frame-to-frame) motion required to register a motion.

Definition

EYEMOROCKNROLL_PARAM_GESTURE_MOTION_THRESHOLD

Type

Integer (*int32*)

Value

A percentage of the search range (0–100), the useful range of which is 5–50.

Default

EYEMOROCKNROLL_DEFAULT_GESTURE_MOTION_THRESHOLD

Notes

This should be larger than incidental motion.

4.9 Gesture Track Direction

Determines which axis or axes of Rock gesture are detected.

Definition

EYEMOROCKNROLL_PARAM_GESTURE_TRACK_DIR

Type

Bit-mask (*int32*)

Value

A combination of the following bit-flags:

EYEMOROCKNROLL_HORIZONTAL - Left and Right gestures will be tracked.

EYEMOROCKNROLL_VERTICAL - Up and Down gestures will be tracked.

Default

EYEMO_DEFAULT_GESTURE_TRACK_DIR (defined as both of the above bit-flags)

Notes

Normally, both horizontal and vertical Rock gestures are tracked. However, if only one direction (horizontal or vertical) is required by the application, setting the track direction allows the system to be more lenient when interpreting diagonal motions as gestures.

A Rock gesture will be classified by direction according to the following illustration:

