libsmart Reference

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Document serial number: 000004892117

Document serial number: 000004892117

Table of Contents

Common APIs

Structures

SceSmartMemoryAllocator

Memory allocator

Definition

```
#include <libsmart.h>
typedef struct SceSmartMemoryAllocator {
        void *dsc;
        SceSmartAllocate allocate;
        SceSmartDeallocate deallocate;
} SceSmartMemoryAllocator;
```

Members

dsc	Descriptor
allocate	Callback function for allocating memory area
deallocate	Callback function for deallocating memory area

Description

This is a structure defining an application-specific memory allocator. It is used for specifying a memory allocator when calling `sceSmartInit()`.

An arbitrary value can be set for *dsc*. The value will be passed as-is to the callback functions set for *allocate* and *deallocate*.

SceSmartQuaternion

Quaternion

Definition

```
#include <libsmart.h>
typedef struct SceSmartQuaternion {
        float x;
        float y;
        float z;
        float w;
} SceSmartQuaternion;
```

Members

x X
y Y
z Z
w W

Description

This is the structure of a quaternion defined by 4 `float` types. It is used for indicating the camera orientation for a recognized object.

SceSmartStreamIn

Input stream

Definition

```
#include <libsmart.h>
typedef struct SceSmartStreamIn {
        void *dsc;
        SceSmartRead read;
} SceSmartStreamIn;
```

Members

dsc	Descriptor
read	Read callback function

Description

This is a structure defining the input stream.

It is used when loading a dictionary with `sceSmartCreateLearnedImageTarget()`.

The value set for *dsc* will be passed as-is as an argument when the callback function set for *read* is called. This value can be used to pass the information that represents load targets such as file pointers to callback functions, for example.

SceSmartStreamOut

Output stream

Definition

```
#include <libsmart.h>
typedef struct SceSmartStreamOut {
        void *dsc;
        SceSmartWrite write;
} SceSmartStreamOut;
```

Members

dsc	Descriptor
write	Write callback function

Description

This is a structure defining the output stream.

SceSmartTargetInfo

Information on the recognition target

Definition

```
#include <libsmart.h>
typedef struct SceSmartTargetInfo {
        float physicalWidth;
        float physicalHeight;
} SceSmartTargetInfo;
```

Members

physicalWidth	Width of the recognition target [m]
physicalHeight	Height of the recognition target [m]

Description

This is a structure storing physical size information on a recognition target. It stores physical size information on learned images, AR Play Cards, etc.

It is used for receiving the `sceSmartGetTargetInfo()` results.

SceSmartVector

3D vector

Definition

```
#include <libsmart.h>
typedef struct SceSmartVector {
        float x;
        float y;
        float z;
} SceSmartVector;
```

Members

x x
y y
z z

Description

This is the structure of 3D vectors defined by 3 `float` types. It is used for indicating the position of a camera for a recognized object, landmark positions in a coordinate system, etc.

Functions

sceSmartCreateLearnedImageTarget

Create a recognition target from a dictionary file created from an image learned in advance

Definition

```
#include <libsmart.h>
SceInt32 sceSmartCreateLearnedImageTarget(
        SceInt32 *targetIdReturn,
        SceSmartStreamIn *stream
);
```

Arguments

[out] *targetIdReturn*	Destination to store recognition target ID assigned by libsmart
[in] *stream*	Stream loading the dictionary

Return Values

Stores the recognition target ID in **targetIdReturn* and returns SCE_OK (=0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_OUT_OF_MEMORY	0x808c0002	Cannot allocate memory for loading the dictionary
SCE_SMART_ERROR_INVALID_POINTER	0x808c0006	*targetIdReturn* or *stream* is NULL
SCE_SMART_ERROR_VERSION_MISSMATCH	0x808c000c	Dictionary version is inappropriate

Description

This function loads a dictionary from a specified stream, creates a recognition target, and returns the recognition target ID. If loading the dictionary fails, SCE_SMART_INVALID_TARGET_ID will be set to **targetIdReturn*.

sceSmartCreateInstantImageTarget

Create a recognition target from the specified area of an input image

Definition

```
#include <libsmart.h>
SceInt32 sceSmartCreateInstantImageTarget(
        SceInt32* targetIdReturn,
        const void* img,
        int width,
        int height,
        int x,
        int y,
        int cropWidth,
        int cropHeight,
        float physicalWidth
);
```

Arguments

[out] *targetIdReturn*	Destination to store recognition target ID assigned by libsmart
[in] *img*	Input image
[in] *width*	Width of the input image
[in] *height*	Height of the input image
[in] *x*	x coordinate of the upper left corner of the specified area
[in] *y*	y coordinate of the upper left corner of the specified area
[in] *cropWidth*	Width of the specified area(160 to 1280)
[in] *cropHeight*	Height of the specified area (160 to 1280)
[in] *physicalWidth*	Physical width of the specified area (meters)

Return Values

Stores the recognition target ID in **targetIdReturn* and returns SCE_OK (=0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_OUT_OF_MEMORY	0x808c0002	Cannot allocate memory for creating the recognition target
SCE_SMART_ERROR_INVALID_VALUE	0x808c0005	Position, range, size, or physical width of the specified area is invalid
SCE_SMART_ERROR_INVALID_POINTER	0x808c0006	*targetIdReturn* or *img* is NULL
SCE_SMART_ERROR_BAD_IMAGE	0x808c0011	Sufficient performance as a recognition target cannot be expected from the specified area of the input image

Description

This function creates a recognition target from the specified area of an input image and returns the recognition target ID. The width and height of the specified area must be a value within 160 to 1280 pixels.

The recognition target created with this function cannot directly be used to initialize the SceneMapping library. Custom listener is required for initializing the SceneMapping library. Carry out detection and tracking of the recognition target using the TargetTracking library and after confirming the tracking, pass the estimated pose and coordinates on the recognition target with a callback.

If the specified area is not appropriate for recognition and creation of the recognition target fails, `SCE_SMART_INVALID_TARGET_ID` will be set to `*targetIdReturn`.

sceSmartDestroyTarget

Delete a recognition target

Definition

```
#include <libsmart.h>
SceInt32 sceSmartDestroyTarget(
        const SceInt32 targetID
);
```

Arguments

[in] *targetID* Recognition target ID to delete

Return Values

Returns SCE_OK (=0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_INVALID_VALUE	0x808c0005	*targetID* is invalid
SCE_SMART_ERROR_ALREADY_REGISTERED	0x808c0007	The specified recognition target has already been registered
SCE_SMART_ERROR_BUSY	0x808c000e	libsmart is locked by a call from another thread

Description

This function deletes the recognition target with the specified ID.

sceSmartGetTargetInfo

Get information on the recognition target

Definition

```
#include <libsmart.h>
SceInt32 sceSmartGetTargetInfo(
        const SceInt32 targetID,
        SceSmartTargetInfo *info
);
```

Arguments

[in] *targetID* Recognition target ID
[out] *info* Destination to store recognition target information

Return Values

Stores the recognition target information in `*info` and returns `SCE_OK` (=0) for normal termination.
Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_INVALID_VALUE	0x808c0005	*targetID* is invalid
SCE_SMART_ERROR_INVALID_POINTER	0x808c0006	*info* is NULL

Description

This function obtains information on the recognition target. Information on the recognition target with the specified target ID will be obtained. Refer to the "SceSmartTargetInfo" section for details.

sceSmartInit

Initialize libsmart

Definition

```
#include <libsmart.h>
SceInt32 sceSmartInit(
        SceSmartMemoryAllocator *mallocator
);
```

Arguments

[in] *mallocator* Memory allocator, or NULL

Return Values

Returns SCE_OK (=0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_ALREADY_INITIALIZED	0x808c0001	libsmart is already initialized
SCE_SMART_ERROR_OUT_OF_MEMORY	0x808c0002	Cannot allocate memory for initializing libsmart
SCE_SMART_ERROR_INVALID_POINTER	0x808c0006	A member of *mallocator* is NULL
SCE_SMART_ERROR_BUSY	0x808c000e	libsmart is locked by a call from another thread

Description

This function initializes libsmart. Sets the variables to be used in libsmart and allocates memory. The memory allocator and deallocator used in libsmart can be specified. malloc() and free() will be used when NULL is specified for *mallocator*.

sceSmartRelease

Stop the use of libsmart

Definition

```
#include <libsmart.h>
SceInt32 sceSmartRelease(void);
```

Arguments

None

Return Values

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_NOT_STOPPED	0x808c0003	libsmart is not stopped
SCE_SMART_ERROR_NOT_EMPTY	0x808c0004	A registered recognition target has not been unregistered yet .
SCE_SMART_ERROR_BUSY	0x808c000e	libsmart is locked by a call from another thread

Description

This function stops the use of libsmart, and releases resources used by libsmart, such as memory and devices.

Callbacks

SceSmartAllocate

Callback function for allocating memory area

Definition

```
#include <libsmart.h>
typedef void * (*SceSmartAllocate)(
        void *dsc,
        size_t size
);
```

Arguments

dsc	Descriptor
size	Size to be allocated

Return Values

Return the pointer to the allocated memory area.

Return NULL if a memory area cannot be allocated.

Description

This is a prototype of an application-specific memory allocator.

The descriptor and the size of the memory to be allocated are passed as arguments. Allocate memory in the size passed and return the pointer to the allocated memory area. If allocation was not possible, return NULL.

SceSmartDeallocate

Callback function for deallocating memory area

Definition

```
#include <libsmart.h>
typedef void (*SceSmartDeallocate)(
        void *dsc,
        void *ptr
);
```

Arguments

dsc	Descriptor
ptr	Pointer to the memory area to be deallocated

Return Values

None

Description

This is a prototype of an application-specific memory deallocator.

The descriptor and a pointer to the memory area to be deallocated are passed as arguments.

SceSmartRead

Read callback function

Definition

```
#include <libsmart.h>
typedef size_t (*SceSmartRead)(
        void *dsc,
        void *buf,
        size_t size
);
```

Arguments

dsc	Descriptor
buf	Area where loaded data is saved
size	Loaded size

Return Values

Return the successfully loaded size.

Returns an error code (negative value) for errors.

Description

This is a prototype of an application-specific data load function.

The descriptor, buffer and loaded size are passed as arguments. Read data up to the specified size from the descriptor, store the data in *buf*[], and return the actually loaded size.

SceSmartWrite

Write callback function

Definition

```
#include <libsmart.h>
typedef size_t (*SceSmartWrite)(
        void *dsc,
        const void *buf,
        size_t size
);
```

Arguments

dsc	Descriptor
buf	Area storing the data to be written
size	Size to be written

Return Values

Return the successfully written size.

Returns an error code (negative value) for errors.

Description

The descriptor, buffer and size to be written are passed as arguments. Read the data from the memory area specified with *buf*, write the data to the descriptor, and return the actually written size.

Constants

List of Constants

List of constants commonly used by libsmart APIs

Definition

Value	(Number)	Description
SCE_SMART_IMAGE_FOV	0.84160f	The estimated value of the vertical viewing angle of the rear camera of PlayStation®Vita. Derived from atan2(480/2, (536.0102+536.5372)/2)*2. (536.0102 and 536.5372 are the camera's fixed value.)
SCE_SMART_IMAGE_HEIGHT	480	Fixed value representing the height of the image input in libsmart
SCE_SMART_IMAGE_WIDTH	640	Fixed value representing the width of the image input in libsmart
SCE_SMART_INVALID_TARGET_ID	-1	Fixed value indicating that the target ID is invalid, for example, when failing to create a recognition target
SCE_SMART_MARKER_01	0x70000001	AR Play Card 01's marker ID
SCE_SMART_MARKER_02	0x70000002	AR Play Card 02's marker ID
SCE_SMART_MARKER_03	0x70000003	AR Play Card 03's marker ID
SCE_SMART_MARKER_04	0x70000004	AR Play Card 04's marker ID
SCE_SMART_MARKER_05	0x70000005	AR Play Card 05's marker ID
SCE_SMART_MARKER_06	0x70000006	AR Play Card 06's marker ID

Return Codes

List of return codes returned by libsmart APIs

Definitions

Value	(Number)	Description
SCE_ERROR_FACILITY_SMART	0x08c	libsmart error code
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_ALREADY_INITIALIZED	0x808c0001	libsmart has already been initialized.
SCE_SMART_ERROR_OUT_OF_MEMORY	0x808c0002	libsmart failed to allocate memory
SCE_SMART_ERROR_NOT_STOPPED	0x808c0003	libsmart is not stopped
SCE_SMART_ERROR_NOT_EMPTY	0x808c0004	Some resources have not been released yet
SCE_SMART_ERROR_INVALID_VALUE	0x808c0005	The value given is invalid
SCE_SMART_ERROR_INVALID_POINTER	0x808c0006	The pointer given is invalid
SCE_SMART_ERROR_ALREADY_REGISTERED	0x808c0007	Input has already been registered
SCE_SMART_ERROR_NOT_REGISTERED	0x808c0008	Input has not been registered
SCE_SMART_ERROR_ALREADY_STARTED	0x808c0009	libsmart has already been started up
SCE_SMART_ERROR_NOT_STARTED	0x808c000a	libsmart has not been started up yet
SCE_SMART_ERROR_NOT_REQUIRED	0x808c000b	libsmart is not yet ready to process the request
SCE_SMART_ERROR_VERSION_MISSMATCH	0x808c000c	Provided value or the file version is inappropriate
SCE_SMART_ERROR_NO_DICTIONARY	0x808c000d	The recognition target has not been registered
SCE_SMART_ERROR_BUSY	0x808c000e	libsmart is locked by a call from another thread
SCE_SMART_ERROR_BAD_IMAGE	0x808C0011	Input is not appropriate as a recognition target

TargetTracking APIs

Structures

SceSmartTargetTrackingInput

Input data for recognition processing

Definition

```
#include <target_tracking.h>
typedef struct SceSmartTargetTrackingInput {
        struct {
              SceUInt32 count;
              const SceMotionSensorState *states;
        } motion;
        struct {
              SceUInt64 timestamp;
              const void *data;
              SceUInt32 padding;
        } image;
} SceSmartTargetTrackingInput;
```

Members

motion.count	Number of elements for *motion.states*[]
motion.states	Array of the motion sensors' measurement value.
image.timestamp	Timestamp of the camera image
image.data	Camera image data.
image.padding	Padding

Description

This structure is used for specifying the input data for recognition processing with `sceSmartTargetTrackingRun2()` and `sceSmartTargetTrackingDispatchAndQuery()`. It stores the images to be processed, the motion sensors' measurement values and their timestamps.

For *states*, prepare an array of measurements obtained from libmotion that have been ordered from the value with the newest observed time without any gaps or overlaps.

The image data to be specified in *data* must have a width of `SCE_SMART_IMAGE_WIDTH` pixels and a height of `SCE_SMART_IMAGE_HEIGHT` pixels. In addition, each pixel value must be an unsigned char (8 bits) that indicates the pixel intensity in grayscale.

SceSmartTargetTrackingResult

Results of recognition concerning recognition targets

Definition

```
#include <target_tracking.h>
typedef struct SceSmartTargetTrackingResult {
        SceSmartTargetTrackingState state;
        SceSmartVector pos;
        SceSmartQuaternion rot;
        SceInt32 detectedTargetID;
} SceSmartTargetTrackingResult;
```

Members

state	Recognition state of recognition targets
pos	Camera's relative position in a coordinate system fixed to the recognition target
rot	Camera's relative orientation in a coordinate system fixed to the recognition target
detectedTargetID	ID of the recognized target; it will be -1 if no target is recognized

Description

This structure stores the results of recognition concerning a recognition target. It is used for receiving the `sceSmartTargetTrackingGetResults()` and `sceSmartTargetTrackingQuery()` results.

See Also

`sceSmartTargetTrackingRun()`

SceSmartTargetTrackingSearchPolicy

Policy for searching recognition targets

Definition

```
#include <target_tracking.h>
typedef enum SceSmartTargetTrackingSearchPolicy {
        SCE_SMART_TARGET_TRACKING_SEARCH_POLICY_FAST,
        SCE_SMART_TARGET_TRACKING_SEARCH_POLICY_PRECISIVE
} SceSmartTargetTrackingSearchPolicy;
```

Enumeration Values

Value	Description
SCE_SMART_TARGET_TRACKING_SEARCH_POLICY_FAST	Prioritizes search speed This search policy focuses on the rapid execution of a single search. The recognition target within the input image needs to be of a certain size.
SCE_SMART_TARGET_TRACKING_SEARCH_POLICY_PRECISIVE	Prioritizes search sensitivity (default) This policy prioritizes search sensitivity, in other words, it allows for smaller recognition targets in the input image to be recognized

Description

This is an enumeration value representing the recognition target's search policy.

According to the set search policy, the TargetTracking library searches for a recognition target in its searching state, and then transitions to tracking state after it finds a target. The calculation time required for a search and the size of searchable target objects within an input image will vary depending on the policy used.

- Calculation time: If speed is prioritized, approximately 1/4 of the time required when sensitivity is prioritized
- Recognition target size: If sensitivity is prioritized, up to approximately 1/4 of searchable size when speed is prioritized

Since prioritizing speed easily obtains smooth user experience, there is value in adjusting the game design and considering user guidance that allows recognition targets to appear with certain sizes.

This setting is only valid when the recognition target is a natural image.

SceSmartTargetTrackingState

Recognition state of recognition targets

Definition

```
#include <target_tracking.h>
typedef enum SceSmartTargetTrackingState {
        SCE_SMART_TARGET_TRACKING_STATE_IDLE,
        SCE_SMART_TARGET_TRACKING_STATE_TARGET_SEARCH,
        SCE_SMART_TARGET_TRACKING_STATE_TARGET_TRACKING
} SceSmartTargetTrackingState;
```

Enumeration Values

Value	Description
SCE_SMART_TARGET_TRACKING_STATE_IDLE	In this state, the TargetTracking library has not been started yet (idle state)
SCE_SMART_TARGET_TRACKING_STATE_TARGET_SEARCH	In this state, no recognition targets have been found in the image (search state)
SCE_SMART_TARGET_TRACKING_STATE_TARGET_TRACKING	In this state, a recognition target has been recognized and is successfully being tracked (tracking state)

Description

These enum constants represent the recognition state of the recognition target. A value of this type will be stored in the *state* member of SceSmartTargetTrackingResult.

SCE_SMART_TARGET_TRACKING_STATE_IDLE indicates that the TargetTracking library has not been started up. After the TargetTracking library is started up, the state for each registered recognition target will transition between SCE_SMART_TARGET_TRACKING_STATE_TARGET_SEARCH and SCE_SMART_TARGET_TRACKING_STATE_TARGET_TRACKING depending on whether the recognition target is found and tracked successfully.

Regarding state transitions of recognition targets, refer to "Using the TargetTracking Library" chapter of the "libsmart Overview" document.

Functions

sceSmartTargetTrackingDispatchAndQuery

Get the latest recognition results

Definition

```
#include <target_tracking.h>
SceInt32 sceSmartTargetTrackingDispatchAndQuery(
        const SceSmartTargetTrackingInput *args
);
```

Arguments

[in] *args* Input data for recognition processing

Return Values

Returns the number of recognized recognition targets (0 or more) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_INVALID_POINTER	0x808c0006	*args* is NULL
SCE_SMART_ERROR_NOT_STARTED	0x808c000a	The TargetTracking library is not started

Description

This function registers an input data to an internal queue and requests the recognition results for each recognition target that are complete at this time. The number of currently recognized recognition targets will be returned.

By using functions such as `sceSmartTargetTrackingGetResults()` or `sceSmartTargetTrackingQuery()`, the recognition results at this time can be obtained. Until this function is newly called, the recognition results will not change. The recognition timestamp for each recognition target may be different depending on the call timing.

Every time this function is called, multiple `sceSmartTargetTrackingRunWorker()` calls are required. For details, refer to the explanation for `sceSmartTargetTrackingRunWorker()`.

sceSmartTargetTrackingGetResults

Get recognition results

Definition

```
#include <target_tracking.h>
SceInt32 sceSmartTargetTrackingGetResults(
        SceSmartTargetTrackingResult *outResults,
        SceInt32 maxNumResults
);
```

Arguments

[out] *outResults*	Array to store recognition results
[in] *maxNumResults*	Number of elements in *outResults*[]

Return Values

Stores the recognition results up to the *maxNumResults* number in **outResults* and returns the number of stored recognition results for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_INVALID_POINTER	0x808c0006	*outResults* is NULL
SCE_SMART_ERROR_NOT_STARTED	0x808c000a	The TargetTracking library is not started

Description

This function obtains recognition results for all recognition targets that have been recognized in the recognition processing that was executed immediately before. It will not output results for targets that have not been recognized.

sceSmartTargetTrackingQuery

Get recognition results for a specific recognition target

Definition

```
#include <target_tracking.h>
SceInt32 sceSmartTargetTrackingQuery(
        SceSmartTargetTrackingResult *outResult,
        SceInt32 targetID
);
```

Arguments

[out] *outResult* Destination to store the recognition results
[in] *targetID* ID of the recognition target

Return Values

Stores the recognition results in **outResult* and returns SCE_OK (=0) for normal termination.
Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_INVALID_VALUE	0x808c0005	*targetID* is invalid
SCE_SMART_ERROR_INVALID_POINTER	0x808c0006	*outResult* is NULL
SCE_SMART_ERROR_NOT_REGISTERED	0x808c0008	Specified recognition target has not been registered yet
SCE_SMART_ERROR_NOT_STARTED	0x808c000a	The TargetTracking library is not started

Description

This function obtains the results of the recognition processing performed by sceSmartTargetTrackingRun(). This function returns the recognition results for the target with the specified ID. This function will normally terminate and store the recognition results in **outResult* even if the specified recognition target was not recognized. In such cases, it will be possible to determine that recognition was not successful if *outResult->state* is SCE_SMART_TARGET_TRACKING_STATE_TARGET_SEARCH or SCE_SMART_TARGET_TRACKING_STATE_IDLE.

sceSmartTargetTrackingQuery2

Get recognition results and timestamp for a specific recognition target

Definition

```
#include <target_tracking.h>
SceInt32 sceSmartTargetTrackingQuery2(
        SceSmartTargetTrackingResult *outResult,
        SceUInt64 *outTimestamp,
        SceInt32 targetID
);
```

Arguments

[out] *outResult*	Destination to store the recognition results
[out] *outTimestamp*	Destination to store the timestamp for the input image for which recognition processing was performed
[in] *targetID*	ID of the recognition target

Return Values

Stores the recognition results in **outResult*, stores the timestamp for the input image for which recognition processing was performed in **outTimestamp*, and returns SCE_OK (=0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_INVALID_VALUE	0x808c0005	Invalid value is specified for *targetID*
SCE_SMART_ERROR_INVALID_POINTER	0x808c0006	NULL is specified for both *outResult* and *outTimestamp*
SCE_SMART_ERROR_NOT_REGISTERED	0x808c0008	Specified recognition target has not been registered yet
SCE_SMART_ERROR_NOT_STARTED	0x808c000a	The TargetTracking library is not started

Description

This function obtains the results of the recognition processing performed by sceSmartTargetTrackingRun2(). The recognition results and input image timestamp will be returned for the recognition target with the specified ID. When one or the other is not needed, specify NULL for the corresponding argument (*outResult* or *outTimestamp*).

This function will normally terminate and store the recognition results in **outResult* and store the timestamp in **outTimestamp* even if the specified recognition target was not recognized. In such cases, it will be possible to determine that recognition was not successful if *outResult->state* is SCE_SMART_TARGET_TRACKING_STATE_TARGET_SEARCH or SCE_SMART_TARGET_TRACKING_STATE_IDLE.

sceSmartTargetTrackingRegisterTarget

Register recognition targets

Definition

```
#include <target_tracking.h>
SceInt32 sceSmartTargetTrackingRegisterTarget(
        const SceInt32 targetID
);
```

Arguments

[in] *targetID* ID of recognition target to register

Return Values

Returns SCE_OK (=0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_INVALID_VALUE	0x808c0005	*targetID* is invalid
SCE_SMART_ERROR_ALREADY_REGISTERED	0x808c0007	Specified recognition target is already registered
SCE_SMART_ERROR_BUSY	0x808c000e	libsmart is locked by a call from another thread

Description

This function adds the recognition target with the specified ID to the TargetTracking library's recognition candidate list. For *targetID*, specify the ID assigned by libsmart when creating the recognition target with sceSmartCreateLearnedImageTarget().

sceSmartTargetTrackingReset

Reset the TargetTracking library

Definition

```
#include <target_tracking.h>
SceInt32 sceSmartTargetTrackingReset(void);
```

Arguments

None

Return Values

Returns SCE_OK (=0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_NOT_STARTED	0x808c000a	The TargetTracking library is not started
SCE_SMART_ERROR_BUSY	0x808c000e	libsmart is locked by a call from another thread

Description

This function resets the TargetTracking library. Already created recognition targets will be retained, but the recognition state of these targets will return to the SCE_SMART_TARGET_TRACKING_STATE_IDLE state and the search policy will return to the default (prioritize search sensitivity).

sceSmartTargetTrackingRun

Execute recognition processing

Definition

```
#include <target_tracking.h>
SceInt32 sceSmartTargetTrackingRun(
        const void *image
);
```

Arguments

[in] *image* Input image for recognition

Return Values

Returns the number of recognized objects for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_INVALID_POINTER	0x808c0006	*image* is NULL
SCE_SMART_ERROR_NOT_STARTED	0x808c000a	The TargetTracking library is not started
SCE_SMART_ERROR_BUSY	0x808c000e	libsmart is locked by a call from another thread

Description

This function executes recognition processing. This function performs recognition processing on a specified input image, and returns the number of recognized recognition targets. The TargetTracking library's internal state is updated based on recognition results.

The input image's width must be SCE_SMART_IMAGE_WIDTH pixels, while its height must be SCE_SMART_IMAGE_HEIGHT pixels. Furthermore, each pixel value must be an unsigned char (8 bits) that indicates the pixel intensity in grayscale. Prepare image data using a method such as extracting only the Y component from an SCE_CAMERA_FORMAT_YUV422_PACKED image or a method that calculates only the Y component from an SCE_CAMERA_FORMAT_RAW8 image.

Note that YUV422 images are packed at the pixel level, therefore processing that decomposes into a Y-plane is required.

sceSmartTargetTrackingRun2

Execute recognition processing

Definition

```
#include <target_tracking.h>
SceInt32 sceSmartTargetTrackingRun2(
        const SceSmartTargetTrackingInput *args
);
```

Arguments

[in] *args* Input data for recognition processing

Return Values

Returns the number of recognized recognition targets for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_INVALID_POINTER	0x808c0006	*args* is NULL
SCE_SMART_ERROR_NOT_STARTED	0x808c000a	The TargetTracking library is not started
SCE_SMART_ERROR_BUSY	0x808c000e	libsmart is locked by a call from another thread

Description

This function executes recognition processing. This function performs recognition processing on the specified input data, and returns the number of recognized recognition targets.

sceSmartTargetTrackingRunWorker

Extract processes from the internal queue and execute it

Definition

```
#include <target_tracking.h>
SceInt32 sceSmartTargetTrackingRunWorker(void);
```

Arguments

None

Return Values

Returns SCE_OK (=0) or 1 when the queue's last process is over (details below).

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_NOT_STARTED	0x808c000a	The TargetTracking library is not started
SCE_SMART_ERROR_NOT_REQUIRED	0x808c000b	libsmart is not yet ready to process the request

Description

This function executes recognition processing for an input data registered in an internal queue with sceSmartTargetTrackingDispatchAndQuery(). The processing may be internally split and executed into multiple jobs.

When one job completes, This function will return. This function will return 0 if the processing is complete for all jobs, and this will return 1 if there are jobs remaining. Repeatedly call this function as long as the return value is 1.

SCE_SMART_ERROR_NOT_REQUIRED is returned if called when the queue is empty.

It is possible to register the next input data to be processed in the queue in advance by calling sceSmartTargetTrackingDispatchAndQuery(). After all jobs complete for an input data which is being processed, the recognition processing will begin for the newest input data registered in the queue. Therefore, management is not required for the call timing between sceSmartTargetTrackingRunWorker() and sceSmartTargetTrackingDispatchAndQuery(). Note that when multiple threads call sceSmartTargetTrackingRunWorker(), the recognition processing for the next input data may begin even if sceSmartTargetTrackingRunWorker() has not terminated in one thread.

sceSmartTargetTrackingSetSearchPolicy

Set a specified search policy

Definition

```
#include <target_tracking.h>
SceInt32 sceSmartTargetTrackingSetSearchPolicy(
        SceSmartTargetTrackingSearchPolicy policy
);
```

Arguments

[in] *policy* Search policy

Return Values

Returns SCE_OK (=0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_INVALID_VALUE	0x808c0005	*policy* is invalid
SCE_SMART_ERROR_BUSY	0x808c000e	libsmart is locked by a call from another thread

Description

This function sets a specified search policy in the TargetTracking library. This is only valid when the recognition target is a natural image.

sceSmartTargetTrackingStart

Start the TargetTracking library

Definition

```
#include <target_tracking.h>
SceInt32 sceSmartTargetTrackingStart(
        SceInt32 maxNumTargets
);
```

Arguments

[in] *maxNumTargets* Maximum number of recognition targets to track at the same time (1 to SCE_SMART_TARGET_TRACKING_MAX_NUM_TARGETS)

Return Values

Returns SCE_OK (=0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_OUT_OF_MEMORY	0x808c0002	Cannot allocate memory for starting the TargetTracking library
SCE_SMART_ERROR_INVALID_VALUE	0x808c0005	*maxNumTargets* is invalid
SCE_SMART_ERROR_ALREADY_STARTED	0x808c0009	The TargetTracking library is already started
SCE_SMART_ERROR_NO_DICTIONARY	0x808c000d	The recognition target has not been registered yet
SCE_SMART_ERROR_BUSY	0x808c000e	libsmart is locked by a call from another thread

Description

This function starts up the TargetTracking library.

Note that it is not possible to limit the maximum number of recognition targets for AR Play Cards with this function.

sceSmartTargetTrackingStop

Stop the TargetTracking library

Definition

```
#include <target_tracking.h>
SceInt32 sceSmartTargetTrackingStop(void);
```

Arguments

None

Return Values

Returns SCE_OK (=0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_NOT_STARTED	0x808c000a	The TargetTracking library is not started
SCE_SMART_ERROR_BUSY	0x808c000e	libsmart is locked by a call from another thread

Description

This function stops the TargetTracking library.

sceSmartTargetTrackingUnregisterTarget

Unregister recognition target

Definition

```
#include <target_tracking.h>
SceInt32 sceSmartTargetTrackingUnregisterTarget(
        const SceInt32 targetID
);
```

Arguments

[in] *targetID* ID of recognition target to unregister

Return Values

Returns SCE_OK (=0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_INVALID_VALUE	0x808c0005	*targetID* is invalid
SCE_SMART_ERROR_NOT_REGISTERED	0x808c0008	Specified recognition target has not been registered

Description

This function removes the recognition target with the specified ID from the TargetTracking library's recognition candidate list. For *targetID*, specify the ID assigned by libsmart when creating the recognition target with sceSmartCreateLearnedImageTarget().

Constants

List of Constants

List of constants used by the TargetTracking library APIs

Definition

Value	(Number)	Description
SCE_SMART_TARGET_TRACKING_MAX_NUM_TARGETS	9	Maximum value for the number of recognition targets in a natural image

SceneMapping APIs

Structures

SceSmartFlags

Type for flags

Definition

```
#include <scene_mapping.h>
typedef SceUInt32 SceSmartFlags;
```

Description

This is a type for flags for checking whether the data to be input in `sceSmartSceneMappingRunCore()` is ready.

SceSmartSceneMappingCustomListener

Custom listener callback definition

Definition

```
#include <scene_mapping.h>
typedef struct SceSmartSceneMappingCustomListener {
        void* dsc;
        SceSmartSceneMappingCustomListenerOnLocalizationRequested
onLocalizeRequested;
        SceSmartSceneMappingCustomListenerOnLandmarkDetected
onLandmarkDetected;
} SceSmartSceneMappingCustomListener;
```

Members

dsc	Descriptor
onLocalizeRequested	Callback function for custom initialization
onLandmarkDetected	Callback function for landmark 3D coordinate estimate

Description

This structure defines the callback function for custom initialization and unique landmark detection.

SceSmartSceneMappingCustomListenerLandmark

Landmark information for custom listener

Definition

```
#include <scene_mapping.h>
typedef enum SceSmartSceneMappingCustomListenerLandmark {
        const SceSmartSceneMappingInitializationPointInfo* initialPoint;
        SceSmartSceneMappingCustomListenerLandmarkResponse res;
        SceSmartVector pos;
} SceSmartSceneMappingCustomListenerLandmark;
```

Members

initialPoint	Information of the feature point serving as a landmark candidate
res	State of the landmark in the callback
pos	Position of the landmark in scene-fixed coordinates

Description

This structure is for exchanging landmark-related information with the SceneMapping library to carry out unique detection processing with a custom listener.

SceSmartSceneMappingCustomListenerLandmarkResponse

State of a feature point to be a landmark candidate for custom listener

Definition

```
#include <scene_mapping.h>
typedef enum SceSmartSceneMappingCustomListenerLandmarkResponse {
    SCE_SMART_SCENE_MAPPING_CUSTOM_LISTENER_LANDMARK_RESPONSE_UPTO_START,
    SCE_SMART_SCENE_MAPPING_CUSTOM_LISTENER_LANDMARK_RESPONSE_DEFERRED,
    SCE_SMART_SCENE_MAPPING_CUSTOM_LISTENER_LANDMARK_RESPONSE_PLACED,
    SCE_SMART_SCENE_MAPPING_CUSTOM_LISTENER_LANDMARK_RESPONSE_FIXED,
    SCE_SMART_SCENE_MAPPING_CUSTOM_LISTENER_LANDMARK_RESPONSE_DISCARD
} SceSmartSceneMappingCustomListenerLandmarkResponse;
```

Members

SCE_SMART_SCENE_MAPPING_CUSTOM_LISTENER_LANDMARK_RESPONSE_UPTO_START	3D coordinates of a feature point was not estimated. Whether it will be used as a landmark will be determined by the library. The feature point will subsequently be excluded as a callback target.
SCE_SMART_SCENE_MAPPING_CUSTOM_LISTENER_LANDMARK_RESPONSE_DEFERRED	3D coordinates of a feature point was not estimated. The feature point will be retained and included in the next callback.
SCE_SMART_SCENE_MAPPING_CUSTOM_LISTENER_LANDMARK_RESPONSE_PLACED	Succeeded in estimating 3D coordinates of the feature point. It has been registered as a landmark in the SceneMapping library and the library will update the 3D position.
SCE_SMART_SCENE_MAPPING_CUSTOM_LISTENER_LANDMARK_RESPONSE_FIXED	Succeeded in estimating 3D coordinates of the feature point. It has been registered as a landmark in the SceneMapping library and the library will not update the 3D position.
SCE_SMART_SCENE_MAPPING_CUSTOM_LISTENER_LANDMARK_RESPONSE_DISCARD	Delete the feature point. The processing in the library has also been deleted and this feature point will not become a landmark.

Description

This structure specifies the feature point to use as a landmark.

SceSmartSceneMappingCustomListenerLocalization

Pose estimate information for custom listener

Definition

```
#include <scene_mapping.h>
typedef enum SceSmartSceneMappingCustomListenerLocalization{
        SceSmartSceneMappingCustomListenerLocalizationResponse res;
        SceSmartVector pos;
        SceSmartQuaternion rot;
} SceSmartSceneMappingCustomListenerLocalization;
```

Members

res	State of post estimate for custom listener
pos	Initial position of the device (camera) in the scene-fixed coordinate system
rot	Initial orientation of the device (camera) in the scene-fixed coordinate system

Description

This structure stores the pose estimate result when using a custom listener.

SceSmartSceneMappingCustomListenerLocalizationResponse

Pose estimate state for custom listener

Definition

```
#include <scene_mapping.h>
typedef enum SceSmartSceneMappingCustomListenerLocalizationResponse {
   SCE_SMART_SCENE_MAPPING_CUSTOM_LISTENER_LOCALIZATION_RESPONSE_CONTINUING,
   SCE_SMART_SCENE_MAPPING_CUSTOM_LISTENER_LOCALIZATION_RESPONSE_RECOGNIZED
} SceSmartSceneMappingCustomListenerLocalizationResponse;
```

Members

SCE_SMART_SCENE_MAPPING_CUSTOM_LISTENER_LOCALIZATION_RESPONSE_CONTINUING	Do not change pose of the SceneMapping library
SCE_SMART_SCENE_MAPPING_CUSTOM_LISTENER_LOCALIZATION_RESPONSE_RECOGNIZED	Reflect the pose estimated using a custom listener to the SceneMapping library

Description

This structure indicates the state of pose estimation for custom initialization.

When the pose estimate state is SCE_SMART_SCENE_MAPPING_CUSTOM_LISTENER_LOCALIZATION_RESPONSE_CONTINUING, it will be determined as failing in custom pose estimation or that there is currently no need to overwrite the pose.

When the pose estimate state is SCE_SMART_SCENE_MAPPING_CUSTOM_LISTENER_LOCALIZATION_RESPONSE_RECOGNIZED, it will be determined that custom pose estimation succeeded and the SceneMapping library will initialize the pose or overwrite the pose based on the externally provided information via SceSmartSceneMappingCustomListenerLocalization.

SceSmartSceneMappingDispatchMode

Dispatch mode

Definition

```
#include <scene_mapping.h>
typedef enum SceSmartSceneMappingDispatchMode {
        SCE_SMART_SCENE_MAPPING_DISPATCH_MODE_SYNC,
        SCE_SMART_SCENE_MAPPING_DISPATCH_MODE_ASYNC,
} SceSmartSceneMappingDispatchMode;
```

Members

SCE_SMART_SCENE_MAPPING_DISPATCH_MODE_SYNC	Synchronous mode
SCE_SMART_SCENE_MAPPING_DISPATCH_MODE_ASYNC	Asynchronous mode

Description

This enumerator type represents the dispatch mode for specifying the timing in `sceSmartSceneMappingSetDispatchMode()` at which to exchange input between `sceSmartSceneMappingDispatchAndQuery()` and `sceSmartSceneMappingRunCore()`.

The default dispatch mode is the synchronous mode (`SCE_SMART_SCENE_MAPPING_DISPATCH_MODE_SYNC`).

When the dispatch mode is `SCE_SMART_SCENE_MAPPING_DISPATCH_MODE_SYNC`, the next input will not be prepared even if `sceSmartSceneMappingDispatchAndQuery()` is called from a different thread while recognition processing is being executed with `sceSmartSceneMappingRunCore()`. When `sceSmartSceneMappingRunCore()` ends and a new image is required, a flag indicating this will return upon calling `sceSmartSceneMappingDispatchAndQuery()` and the next processing will be prepared. Deep copying of the input image and motion sensor measurements to the SceneMapping library will also be carried out in the preparation. The recognition results are updated using the latest results.

When the dispatch mode is `SCE_SMART_SCENE_MAPPING_DISPATCH_MODE_ASYNC`, `sceSmartSceneMappingDispatchAndQuery()` will save input data information (for example, an image) in an internal buffer every time it is called and `sceSmartSceneMappingRunCore()` will always reference the latest input data in the buffer from the timestamp.

Because the wait time to synchronize the `sceSmartSceneMappingDispatchAndQuery()` thread and `sceSmartSceneMappingRunCore()` thread will be gone, processing will be more efficient.

SceSmartSceneMappingInitializationPointInfo

Information on initialization points

Definition

```
#include <scene_mapping.h>
typedef struct SceSmartSceneMappingInitializationPointInfo {
        SceUInt32 id;
        SceSmartVector posInitial;
        SceSmartVector posCurrent;
} SceSmartSceneMappingInitializationPointInfo;
```

Members

id	Initialization point ID
posInitial	2D position within the image when the initialization point begins to be tracked.
posCurrent	Initialization point's current 2D position in the image

Description

This structure is for obtaining information on the initialization point (feature point used in the initialization of SLAM estimation) with `sceSmartSceneMappingGetInitializationPointInfo()`.

posInitial and *posCurrent* are both values expressed as a 3D vector; only the x coordinate and y coordinate are valid, and the z coordinate is not used.

SceSmartSceneMappingInitMode

SLAM's initialization mode

Definition

```
#include <scene_mapping.h>
typedef enum SceSmartSceneMappingInitMode {
        SCE_SMART_SCENE_MAPPING_INIT_LEARNED_IMAGE,
        SCE_SMART_SCENE_MAPPING_INIT_WAAR,
        SCE_SMART_SCENE_MAPPING_INIT_HFG,
        SCE_SMART_SCENE_MAPPING_INIT_VFG,
        SCE_SMART_SCENE_MAPPING_INIT_SFM,
        SCE_SMART_SCENE_MAPPING_INIT_DRYRUN,
        SCE_SMART_SCENE_MAPPING_INIT_NULL,
        SCE_SMART_SCENE_MAPPING_INIT_CUSTOM
} SceSmartSceneMappingInitMode;
```

Enumeration Values

Value	Description
SCE_SMART_SCENE_MAPPING_INIT_LEARNED_IMAGE	Performs initialization with a learned natural image.
SCE_SMART_SCENE_MAPPING_INIT_WAAR	Performs initialization with an AR Play Card.
SCE_SMART_SCENE_MAPPING_INIT_HFG	Horizontal from gravity (HFG) initialization that estimates the horizontal plane using the gravity direction
SCE_SMART_SCENE_MAPPING_INIT_VFG	Vertical from gravity (VFG) initialization that estimates the vertical plane using the gravity direction
SCE_SMART_SCENE_MAPPING_INIT_SFM	Structure from Motion (SFM) initialization that makes an estimate from the motion parallax
SCE_SMART_SCENE_MAPPING_INIT_DRYRUN	Feature point location mode for HFG/VFG/SFM
SCE_SMART_SCENE_MAPPING_INIT_NULL	Scene map loading mode
SCE_SMART_SCENE_MAPPING_INIT_CUSTOM	Custom initialization

Description

This enumeration value represents the initialization mode used by the SceneMapping library to initialize SLAM estimation. For coordinate systems in each initialization mode, refer to the "libsmart Overview" document.

SCE_SMART_SCENE_MAPPING_INIT_LEARNED_IMAGE

Initialization is carried out with a learned natural image. This mode assumes that a learned natural image is at the origin of the scene-fixed coordinate system, and performs initialization by detecting the natural image.

SCE_SMART_SCENE_MAPPING_INIT_WAAR

Initialization is carried out with an AR Play Card. This mode assumes that an AR Play Card is at the origin of the scene-fixed coordinate system, and performs initialization by detecting the AR Play Card.

SCE_SMART_SCENE_MAPPING_INIT_HFG

Initialization is carried out with horizontal from gravity (HFG) initialization that estimates the horizontal plane from the gravity direction. This mode uses the motion sensors to estimate the horizontal plane and initialize SLAM. Point the camera in the direction of horizontal planes with rich textures (pictures placed on a table, etc.).

This mode requires input from the motion sensors to find the gravity direction. Use `sceSmartSceneMappingDispatchAndQuery()` and `sceSmartSceneMappingRunCore()` to input motion sensor data.

SCE_SMART_SCENE_MAPPING_INIT_VFG

Initialization is carried out with vertical from gravity (VFG) initialization that estimates the vertical plane from the gravity direction. This mode uses the motion sensors to estimate the vertical plane and initialize SLAM. Point the camera in the direction of vertical plane with a rich texture (the surface of a building's wall, etc.). The camera and the vertical plane need to be directly confronting each other.

This mode requires input from the motion sensors to find the gravity direction. Use `sceSmartSceneMappingDispatchAndQuery()` and `sceSmartSceneMappingRunCore()` to input motion sensor data.

SCE_SMART_SCENE_MAPPING_INIT_SFM

Initialization is carried out with structure from motion (SFM) initialization that makes an estimate from the motion parallax. This mode tracks feature points within a 2D image, estimates a 3D structure from parallax when tracking begins and ends, and initializes SLAM. Instruct users to move the camera slowly and in a parallel fashion toward a texture-rich scene.

If motion sensor data is not passed in this initialization mode, the photographed object will be processed by assuming that it is flat.

SCE_SMART_SCENE_MAPPING_INIT_DRYRUN

This is a feature point location mode for HFG/VFG/SFM. Use it in a practice run for SFM, HFG, or VFG. This mode tracks the feature points used by SFM, HFG and VFG, and checks how many feature points are present within a 2D image. The number and position of feature points can be found with `sceSmartSceneMappingGetNumInitializationPoints()` and `sceSmartSceneMappingGetInitializationPointInfo()`. Actual initialization cannot be performed in this mode. Use the SFM, HFG or VFG initialization mode to perform initialization.

SCE_SMART_SCENE_MAPPING_INIT_NULL

This is the scene map loading mode. This mode loads and processes a previously created and saved scene map. This mode requires loading a previously created scene map with `sceSmartSceneMappingLoadMap()`, and performing re-localization with `sceSmartSceneMappingForceLocalize()`. Only when using this initialization mode, the SceneMapping library's starting state will be `SCE_SMART_SCENE_MAPPING_STATE_LOCALIZE`. Also, the scene map will become fixed, and searches for new landmarks will stop.

SCE_SMART_SCENE_MAPPING_INIT_CUSTOM

This is the custom initialization mode. By setting a unique callback to `SceSmartSceneMappingCustomListenerOnLocalizationRequested` of the `SceSmartSceneMappingCustomListener` structure, it will be possible to pass the device (camera) pose and landmark information held by the application to the SceneMapping library.

SceSmartSceneMappingInput

Input data for recognition/estimate processing

Definition

```
#include <scene_mapping.h>
typedef struct SceSmartSceneMappingInput {
        struct {
              SceUInt32 count;
              const SceMotionSensorState *states;
        } motion;
        struct {
              SceUInt64 timestamp;
              const void *data;
        } image;
} SceSmartSceneMappingInput;
```

Members

motion.count	Number of *motion.states*[]
motion.states	Array of the motion sensors' measurement value
image.timestamp	Timestamp of the camera image
image.data	Camera image data

Description

This structure represents the input data when calling sceSmartSceneMappingRunCore(). It stores the images to be processed, the motion sensors' measurement values and their timestamps. Input data can be held in the library with sceSmartSceneMappingDispatchAndQuery()/sceSmartSceneMappingDispatchAndQueryWithMask().

For *states*, prepare an array of measurements obtained from libmotion ordered from the measurement with the newest observed time, without any gaps or overlaps.

The image data to specify in *data* must have a width of SCE_SMART_IMAGE_WIDTH pixels and a height of SCE_SMART_IMAGE_HEIGHT pixels. Moreover, note that the value held by each pixel must be an unsigned char (8-bit) value representing pixel intensity in that pixel's grayscale.

SceSmartSceneMappingLandmarkInfo

Landmark information

Definition

```
#include <scene_mapping.h>
typedef struct SceSmartSceneMappingLandmarkInfo {
        SceUInt32 id;
        SceSmartSceneMappingLandmarkState state;
        SceSmartVector pos;
} SceSmartSceneMappingLandmarkInfo;
```

Members

id	Landmark ID
state	Landmark's tracking state
pos	Landmark position in the scene-fixed coordinate system

Description

This structure is for obtaining landmark information with `sceSmartSceneMappingGetLandmarkInfo()`.

SceSmartSceneMappingLandmarkState

Landmark's tracking state

Definition

```
#include <scene_mapping.h>
typedef enum SceSmartSceneMappingLandmarkState {
        SCE_SMART_SCENE_MAPPING_LANDMARK_TRACKED,
        SCE_SMART_SCENE_MAPPING_LANDMARK_LOST,
        SCE_SMART_SCENE_MAPPING_LANDMARK_SUSPENDED,
        SCE_SMART_SCENE_MAPPING_LANDMARK_MASKED
} SceSmartSceneMappingLandmarkState;
```

Enumeration Values

Value	Description
SCE_SMART_SCENE_MAPPING_LANDMARK_TRACKED	Tracking successful
SCE_SMART_SCENE_MAPPING_LANDMARK_LOST	Tracking failed
SCE_SMART_SCENE_MAPPING_LANDMARK_SUSPENDED	Other states
SCE_SMART_SCENE_MAPPING_LANDMARK_MASKED	Masking

Description

The landmark tracking state 3D scene map is composed of landmarks. Landmarks are points in a scene whose 3D position has been estimated. The landmarks appearing in the previous input image and in the current input image are tracked, and their 3D positions are then estimated based on the change in their 2D positions.

SceSmartSceneMappingMask

Structure defining mask area

Definition

```
#include <scene_mapping.h>
typedef struct SceSmartSceneMappingMask {
        SceUInt32 num;
        const SceSmartVector *triangleList;
} SceSmartSceneMappingMask;
```

Members

num	Number of elements in *triangleList*[] (number of stored triangular patches)
triangleList	Array storing the vertex coordinate positions (relative values) of triangular patches

Description

This structure is for setting the masking area with sceSmartSceneMappingRunCore().

The masking area is composed of multiple triangular patches in the image. This structure stores the number of those triangular patches and an array of each patch's vertex position (relative value when the image width and height are each set as 1.0) in the image.

Always specify 0.0f for the z coordinate of each element in *triangleList*[].

SceSmartSceneMappingNodePointInfo

Node information

Definition

```
#include <scene_mapping.h>
struct SceSmartSceneMappingNodePointInfo {
        SceUInt32 id;
        SceSmartVector pos;
};
```

Members

id Node ID within scene map
pos 3D position of the node within scene map

Description

This structure represents node information in the DenseMap mode. This structure is not used because the DenseMap mode is currently not provided as a feature of the SceneMapping library.

SceSmartSceneMappingDenseMapMode

DenseMap mode state

Definition

```
#include <scene_mapping.h>
enum SceSmartSceneMappingDenseMapMode {
        SCE_SMART_SCENE_MAPPING_DENSE_MAP_DISABLE,
        SCE_SMART_SCENE_MAPPING_DENSE_MAP_SEMIDENSE
};
```

Enumeration Values

Value	Description
SCE_SMART_SCENE_MAPPING_DENSE_MAP_DISABLE	Disable DenseMap mode
SCE_SMART_SCENE_MAPPING_DENSE_MAP_SEMIDENSE	Enable DenseMap mode

Description

This enumeration value represents the DenseMap mode state.

This enumerator type is not used because the DenseMap mode is currently not provided as a feature of the SceneMapping library.

SceSmartSceneMappingResult

Results of scene mapping process

Definition

```
#include <scene_mapping.h>
typedef struct SceSmartSceneMappingResult {
        SceUInt64 timestamp;
        SceSmartSceneMappingState state;
        SceSmartFlags flags;
        SceSmartVector pos;
        SceSmartQuaternion rot;
        SceSmartVector vel;
        SceSmartVector angVel;
        SceInt32 detectedTargetID;
} SceSmartSceneMappingResult;
```

Members

timestamp	Timestamp
state	Current state of the SceneMapping library
flags	Current state of recognition processing by `sceSmartSceneMappingRunCore()` function
pos	Position of the device (camera) in a scene-fixed coordinate system
rot	Orientation of the device (camera) in a scene-fixed coordinate system
vel	Speed of the device (camera) in a fixed-device coordinate system
angVel	Angular velocity of the device (camera) in a fixed-device coordinate system
detectedTargetID	Recognition target ID used for initialization or -1

Description

This structure is for storing the camera pose information estimated by scene mapping.

It is used when obtaining the results of scene mapping executed by `sceSmartSceneMappingRunCore()` with `sceSmartSceneMappingDispatchAndQuery()`/ `sceSmartSceneMappingDispatchAndQueryWithMask()`.

When `SCE_SMART_FLAG_FRESH_INPUT` is set to *flag* upon calling `sceSmartSceneMappingDispatchAndQuery()`/ `sceSmartSceneMappingDispatchAndQueryWithMask()`, this means that recognition processing has already completed; the application should immediately execute `sceSmartSceneMappingRunCore()` and process the next input data.

When initializing the SceneMapping library with a natural image or with an AR Play Card, the ID of the recognition target recognized upon initialization will be stored in *detectedTargetID*. In all other cases, -1 will always be stored.

SceSmartSceneMappingState

Internal state of the SceneMapping library

Definition

```
#include <scene_mapping.h>
typedef enum SceSmartSceneMappingState {
        SCE_SMART_SCENE_MAPPING_STATE_IDLE,
        SCE_SMART_SCENE_MAPPING_STATE_SEARCH,
        SCE_SMART_SCENE_MAPPING_STATE_SLAM,
        SCE_SMART_SCENE_MAPPING_STATE_LOCALIZE,
        SCE_SMART_SCENE_MAPPING_STATE_LOCALIZE_IMPOSSIBLE
} SceSmartSceneMappingState;
```

Enumeration Values

Value	Description
SCE_SMART_SCENE_MAPPING_STATE_IDLE	Idle state
SCE_SMART_SCENE_MAPPING_STATE_SEARCH	Initialization/search state
SCE_SMART_SCENE_MAPPING_STATE_SLAM	SLAM state
SCE_SMART_SCENE_MAPPING_STATE_LOCALIZE	Localization state
SCE_SMART_SCENE_MAPPING_STATE_LOCALIZE_IMPOSSIBLE	Localization impossible state

Description

SceSmartSceneMappingState represents the SceneMapping library's internal state.

SCE_SMART_SCENE_MAPPING_STATE_IDLE

This is the idle state, in which the SceneMapping library is ready to be started.

Because the SceneMapping library is not yet started in this state, even if functions such as sceSmartSceneMappingRun() and sceSmartSceneMappingRunCore() are called in this state, the SceneMapping library will remain inactive.

SCE_SMART_SCENE_MAPPING_STATE_SEARCH

This is the initialization/search state.

In this state, the SceneMapping library is attempting to initialize the camera's pose and the scene map in the specified initialization mode. When initialization succeeds, the state will automatically transition to SCE_SMART_SCENE_MAPPING_STATE_SLAM.

SCE_SMART_SCENE_MAPPING_STATE_SLAM

This is the SLAM state.

This state is the SceneMapping library's main state. While in this state, the SceneMapping library can satisfactorily estimate the camera's pose in relation to the scene, as well as the scene map. The scene map will be created gradually as the camera moves through the scene (if the map has not been fixed with sceSmartSceneMappingFixMap()). The application will instruct the user to move the camera back and forth, up and down, left and right as opposed to simply rotating it in the same spot. If the camera's pose cannot be estimated due to tracking fails or the camera moving too rapidly, etc. the state will automatically transition to SCE_SMART_SCENE_MAPPING_STATE_LOCALIZE.

SCE_SMART_SCENE_MAPPING_STATE_LOCALIZE

This is the localization state.

In this state, the SceneMapping library is attempting to re-localize (restore camera pose estimation) using the scene map created in the `SCE_SMART_SCENE_MAPPING_STATE_SLAM` state. When the library is in this state, the application will instruct the user to return the camera to the pose it was in during the `SCE_SMART_SCENE_MAPPING_STATE_SLAM` state.

SCE_SMART_SCENE_MAPPING_STATE_LOCALIZE_IMPOSSIBLE

This is the localization impossible state.

This state indicates that the SceneMapping library is unable to re-localize because, during `SCE_SMART_SCENE_MAPPING_STATE_SLAM`, it failed to obtain sufficient information on the scene map required for attempting to restore the camera's pose in the `SCE_SMART_SCENE_MAPPING_STATE_LOCALIZE` state. The application will instruct the user to initialize it again.

Functions

sceSmartSceneMappingDispatchAndQuery

Prepare input data and get the results of recognition processing

Definition

```
#include <scene_mapping.h>
SceInt32 sceSmartSceneMappingDispatchAndQuery(
        SceSmartSceneMappingResult *outResult,
        const SceSmartSceneMappingInput *args
);
```

Arguments

[out] *outResult*	Destination to store results of scene mapping
[in] *args*	Input data, or NULL

Return Values

Stores scene mapping results in **outResult* and returns SCE_OK (=0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_INVALID_POINTER	0x808c0006	*outResult* is NULL
SCE_SMART_ERROR_NOT_STARTED	0x808c000a	The SceneMapping library is not started
SCE_SMART_ERROR_NO_DICTIONARY	0x808c000d	The recognition target is not registered

Description

This function prepares input data for the next recognition process, and obtains the results of the previous one.

When the dispatch mode is the synchronous mode, it checks the state of sceSmartSceneMappingRunCore(), prepares input data for the next processing if recognition processing is complete, and sets SCE_SMART_FLAG_FRESH_INPUT to the *flags* member of *outResult*. If recognition processing has not been completed yet, 0 will be stored in the *flags* member of *outResult* and SCE_OK will return.

When the dispatch mode is the asynchronous mode, the next input data is immediately prepared regardless of the sceSmartSceneMappingRunCore() state.

For the preparation of input data, input images and motion sensor measurement values are deep copied to the SceneMapping library. Recognition results are updated to the latest ones.

When *args* is NULL, only the recognition results will be updated.

See Also

sceSmartSceneMappingRunCore()

sceSmartSceneMappingDispatchAndQueryWithMask

Prepare input data and get the results of recognition processing (using the masking function)

Definition

```
#include <scene_mapping.h>
SceInt32 sceSmartSceneMappingDispatchAndQueryWithMask(
        SceSmartSceneMappingResult *outResult,
        const SceSmartSceneMappingInput *args,
        const SceSmartSceneMappingMask *mask
);
```

Arguments

[out] *outResult*	Destination to store results of scene mapping
[in] *args*	Input data, or NULL
[in] *mask*	mask area

Return Values

Stores scene mapping results in `*outResult` and returns `SCE_OK`(=0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_INVALID_POINTER	0x808c0006	*outResult*, *args* or *mask* is NULL
SCE_SMART_ERROR_NOT_STARTED	0x808c000a	The SceneMapping library is not started
SCE_SMART_ERROR_NO_DICTIONARY	0x808c000d	The recognition target is not registered

Description

This function prepares input data for the next recognition process, and obtains the results of the previous one. This function is the same as `sceSmartSceneMappingDispatchAndQuery()` excluding the fact that it uses the masking feature.

See Also

`sceSmartSceneMappingDispatchAndQuery()`,`sceSmartSceneMappingRunCore()`

sceSmartSceneMappingEnableMask

Enable the masking feature

Definition

```
#include <scene_mapping.h>
SceInt32 sceSmartSceneMappingEnableMask(
        SceUInt32 max
);
```

Arguments

[in] *max* Maximum number of triangular patches or 0

Return Values

Returns SCE_OK (=0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_ALREADY_STARTED	0x808c0009	libsmart is already started
SCE_SMART_ERROR_BUSY	0x808c000e	libsmart is locked by a call from another thread

Description

This function enables the masking feature.

If *max* is larger than 0, the masking feature will be enabled and the maximum number of triangular patches comprising the masking area will be set to *max*.

This function must be executed before sceSmartSceneMappingStart() is called.

sceSmartSceneMappingFixMap

Fix or unfix the Scene Map

Definition

```
#include <scene_mapping.h>
SceInt32 sceSmartSceneMappingFixMap(
        SceBool flag
);
```

Arguments

[in] *flag* Flag indicating fixing or unfixing

Return Values

Returns SCE_OK (=0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_NOT_STARTED	0x808c000a	The SceneMapping library has not been started yet
SCE_SMART_ERROR_BUSY	0x808c000e	libsmart is locked by a call from another thread

Description

This function fixes and unfixes the SceneMapping library's scene maps. Fixing the scene maps will slightly increase the SceneMapping library's processing speed, as its calculation volume will decrease in proportion.

If a scene map is fixed, the position of the landmarks already stored by the scene map will subsequently stop being updated, and new landmarks will not be searched for.

If the scene map is unfixed, the library will search for new landmarks, and estimate their position if found. However, once a landmark has been fixed with this function it will remain fixed even if its map is unfixed.

sceSmartSceneMappingForceLocalize

Force the library to transition to the localization state

Definition

```
#include <scene_mapping.h>
SceInt32 sceSmartSceneMappingForceLocalize(void);
```

Arguments

None

Return Values

Returns SCE_OK (=0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_NOT_STARTED	0x808c000a	The SceneMapping library has not been started yet
SCE_SMART_ERROR_BUSY	0x808c000e	libsmart is locked by a call from another thread

Description

This function forcibly switches the SceneMapping library's state to localization state.

Transition to the localization state is meaningless unless a scene map of a sufficient size required for localization will be created or a scene map has already been loaded.

sceSmartSceneMappingGetInitializationPointInfo

Get information on initialization points

Definition

```
#include <scene_mapping.h>
SceUInt32 sceSmartSceneMappingGetInitializationPointInfo(
        SceSmartSceneMappingInitializationPointInfo *outInitPoints,
        SceUInt32 num
);
```

Arguments

[out] *outInitPoints*	Destination to store obtained initialization point information
[in] *num*	Maximum value of the quantity of information on initialization points to obtain

Return Values

Stores the obtained initialization point information in **outInitPoint* and returns quantity of the information on initialization points (*num* or less) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_INVALID_POINTER	0x808c0006	*outInitPoints* is NULL

Description

This function obtains information such as the 2D position of the initialization points used to initialize SLAM estimation.

Initialization points are feature points that are tracked by the library in a 2D image when initialization is carried out in one of the following modes.

- SCE_SMART_SCENE_MAPPING_INIT_HFG
- SCE_SMART_SCENE_MAPPING_INIT_VFG
- SCE_SMART_SCENE_MAPPING_INIT_SFM

This function is invalid for initialization modes other than these. It also is invalid when the SceneMapping library is in states other than SCE_SMART_SCENE_MAPPING_STATE_SEARCH.

Note that the number of initialization points to be created is always SCE_SMART_SCENE_MAPPING_MAX_NUM_INITIALIZATION_POINTS at maximum.

sceSmartSceneMappingGetLandmarkInfo

Get landmark information

Definition

```
#include <scene_mapping.h>
SceInt32 sceSmartSceneMappingGetLandmarkInfo(
        SceSmartSceneMappingLandmarkInfo *outLandmarks,
        SceUInt32 num
);
```

Arguments

[out] *outLandmarks*	Destination to store the obtained landmark information
[in] *num*	Maximum value of the quantity of landmark information to obtain

Return Values

Stores the obtained landmark information in **outLandmark* and returns quantity of the landmark information (*num* or less) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_INVALID_POINTER	0x808c0006	*outLandmarks* is NULL

Description

This function obtains information on the landmarks managed by the SceneMapping library. Landmark information will be stored in the memory area specified with *outLandmarks*.

sceSmartSceneMappingGetNodePointInfo

Get node information

Description

```
#include <scene_mapping.h>
SceInt32 sceSmartSceneMappingGetNodePointInfo(
        SceSmartSceneMappingNodePointInfo *outPoints,
        SceUInt32 num
);
```

Arguments

[out] *outPoints*	Destination to store the obtained node information
[in] *num*	Maximum number of node information to obtain

Return Values

Stores the obtained node information in **outPoints* and returns the number of the node information (*num* or less) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_INVALID_POINTER	0x808c0006	*outPoints* is NULL

Description

Because the DenseMap mode is not provided as a feature of the SceneMapping library, this function returns the same information as sceSmartSceneMappingGetLandmarkInfo().

sceSmartSceneMappingGetNumInitializationPoints

Get the number of initialization points

Definition

```
#include <scene_mapping.h>
SceUInt32 sceSmartSceneMappingGetNumInitializationPoints(void);
```

Arguments

None

Return Values

Returns the number of initialization points for normal termination.

Returns the following error code (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized

Description

This function returns the number of initialization points used to initialize SLAM estimation.

Initialization points are feature points that are tracked by the library in a 2D image when initialization is carried out in one of the following modes.

- SCE_SMART_SCENE_MAPPING_INIT_HFG
- SCE_SMART_SCENE_MAPPING_INIT_VFG
- SCE_SMART_SCENE_MAPPING_INIT_SFM

This function is invalid with initialization modes other than these. It also is invalid when the SceneMapping library is in states other than SCE_SMART_SCENE_MAPPING_STATE_SEARCH.

Note that the number of initialization points to be created is always SCE_SMART_SCENE_MAPPING_MAX_NUM_INITIALIZATION_POINTS at maximum.

sceSmartSceneMappingGetNumLandmarks

Get the number of current landmarks

Definition

```
#include <scene_mapping.h>
SceInt32 sceSmartSceneMappingGetNumLandmarks(void);
```

Arguments

None

Return Values

Returns the number of landmarks that are currently being created by the library for normal termination.

Returns the following error code (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized

Description

This function obtains the number of landmarks currently being created by the library.

0 is returned when the library state is a state other than SCE_SMART_SCENE_MAPPING_STATE_SLAM.

sceSmartSceneMappingGetNumNodePoints

Get the number of nodes in scene map

Definition

```
#include <scene_mapping.h>
SceInt32 sceSmartSceneMappingGetNumNodePoints(void);
```

Return Values

Returns the number of nodes in scene map for normal termination.

Returns the following error code (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized

Description

This function returns the number of nodes in a scene map.

0 is returned when the library state is a state other than SCE_SMART_SCENE_MAPPING_STATE_SLAM.

sceSmartSceneMappingLoadMap

Load the scene map

Definition

```
#include <scene_mapping.h>
SceInt32 sceSmartSceneMappingLoadMap(
        SceSmartStreamIn *stream
);
```

Arguments

[in] *stream* Input stream

Return Values

Returns SCE_OK (=0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_INVALID_POINTER	0x808c0006	*stream* is NULL
SCE_SMART_ERROR_NOT_STARTED	0x808c000a	The SceneMapping library has not been started yet
SCE_SMART_ERROR_VERSION_MISSMATCH	0x808c000c	The version of the specified scene map file is invalid
SCE_SMART_ERROR_BUSY	0x808c000e	libsmart is locked by a call from another thread

Description

This function loads a scene map from a specified input stream. Before calling this function, it is necessary to specify SCE_SMART_SCENE_MAPPING_INIT_NULL in sceSmartSceneMappingStart(), and to start up (or reopen) the SceneMapping library with the scene map loading mode.

If the loaded map is sufficiently good for its purpose, library processing speed will improve when the map is fixed by calling sceSmartSceneMappingFixMap().

Note
Scene map file versions do not have backward compatibility. In other words, a library of an earlier SDK version cannot load a scene map of a later version.
Refer to the description of sceSmartSceneMappingSaveMap() concerning scene map file versions.

sceSmartSceneMappingPropagateResult

Propagate recognition results

Definition

```
#include <scene_mapping.h>
SceInt32 sceSmartSceneMappingPropagateResult(
        const SceSmartSceneMappingResult *result,
        SceUInt64 timestamp,
        SceSmartSceneMappingResult *outResult
);
```

Arguments

[in] *result*	Results of the propagation source
[in] *timestamp*	Time of the propagation destination
[out] *outResult*	Destination to store propagated results

Return Values

Stores the propagated results in **outResult* and returns SCE_OK (=0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_INVALID_VALUE	0x808c0005	The propagated time is negative or SCE_SMART_SCENE_MAPPING_MAX_PROPAGATION_DURATION is exceeded
SCE_SMART_ERROR_INVALID_POINTER	0x808c0006	*result* or *outResult* is NULL
SCE_SMART_ERROR_NOT_STARTED	0x808c000a	The SceneMapping library has not been started yet

Description

This function propagates the recognition results of a given time to a specified time.

Propagation will, to a certain extent, compensate for delays in calculation. However, if the difference in the time of the propagation-source results and the time of the propagation-destination results exceed SCE_SMART_SCENE_MAPPING_MAX_PROPAGATION_DURATION, propagation will not be carried out and in such a case, the input will directly be copied to the output. If propagation over a long period of time should be absolutely necessary, distribute it over several calls.

sceSmartSceneMappingRegisterTarget

Register recognition targets

Definition

```
#include <scene_mapping.h>
SceInt32 sceSmartSceneMappingRegisterTarget(
        const SceInt32 targetID
);
```

Arguments

[in] *targetID* Recognition target ID to register

Return Values

Returns SCE_OK (=0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_INVALID_VALUE	0x808c0005	*targetID* is invalid
SCE_SMART_ERROR_ALREADY_REGISTERED	0x808c0007	Specified recognition target is already registered
SCE_SMART_ERROR_BUSY	0x808c000e	libsmart is locked by a call from another thread

Description

This function adds the recognition target with the specified ID to the SceneMapping library's recognition candidate list. The SceneMapping library will attempt to recognize the target in an initialization mode using AR Play Cards or learned images.

For *targetID*, specify the ID assigned by libsmart when creating the recognition target with sceSmartCreateLearnedImageTarget().

sceSmartSceneMappingRemoveLandmark

Delete landmarks

Definition

```
#include <scene_mapping.h>
SceInt32 sceSmartSceneMappingRemoveLandmark(
        const SceInt32 id
);
```

Arguments

[in] *id* Landmark ID to delete

Return Values

Returns SCE_OK (=0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_INVALID_VALUE	0x808c0005	*id* is invalid
SCE_SMART_ERROR_NOT_STARTED	0x808c000a	The SceneMapping library has not been started yet
SCE_SMART_ERROR_BUSY	0x808c000e	libsmart is locked by a call from another thread

Description

This function deletes the landmark with the specified ID.

Landmarks are managed by libsmart; applications can obtain landmark IDs through the function sceSmartSceneMappingGetLandmarkInfo().

Note
libsmart also automatically deletes landmarks that hinder estimations.

sceSmartSceneMappingReset

Reset the SceneMapping library

Definition

```
#include <scene_mapping.h>
SceInt32 sceSmartSceneMappingReset(void);
```

Arguments

None

Return Values

Returns SCE_OK (=0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_NOT_STARTED	0x808c000a	The SceneMapping library has not been started yet
SCE_SMART_ERROR_BUSY	0x808c000e	libsmart is locked by a call from another thread

Description

This function returns the SceneMapping library to the same state it was in when sceSmartSceneMappingStart() was called.

The scene map will not be deleted and will remain only if the initialization mode is the scene map loading mode (SCE_SMART_SCENE_MAPPING_INIT_NULL).

sceSmartSceneMappingRun

Execute recognition processing on input images

Definition

```
#include <scene_mapping.h>
SceInt32 sceSmartSceneMappingRun(
        SceSmartSceneMappingResult *outResult,
        const void *image,
        SceUInt64 timestamp
);
```

Arguments

[out] *outResult*	Destination to store recognition results
[in] *image*	Input image
[in] *timestamp*	Input image timestamp

Return Values

Stores the recognition results in **outReult* and returns the number of recognized targets for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_INVALID_POINTER	0x808c0006	*outResult* or *image* is NULL
SCE_SMART_ERROR_NOT_STARTED	0x808c000a	The SceneMapping library has not been started yet
SCE_SMART_ERROR_NO_DICTIONARY	0x808c000d	The recognition target is not registered
SCE_SMART_ERROR_BUSY	0x808c000e	libsmart is locked by a call from another thread

Description

This function executes recognition processing on input images.

It returns the recognition results once recognition processing is complete. The input image will be deep-copied in the SceneMapping library.

`sceSmartSceneMappingRun()` envisages the use of camera input only with a single thread. This function is equivalent to executing the following functions consecutively on one thread without entering the motion sensor values.

```
sceSmartSceneMappingDispatchAndQuery(outResult, args);
sceSmartSceneMappingRunCore();
sceSmartSceneMappingDispatchAndQuery(outResult, NULL);
```

See Also

`sceSmartSceneMappingDispatchAndQuery()`,`sceSmartSceneMappingRunCore()`

sceSmartSceneMappingRunCore

Execute recognition processing on input data

Definition

```
#include <scene_mapping.h>
SceInt32 sceSmartSceneMappingRunCore(void);
```

Arguments

None

Return Values

Returns SCE_OK (=0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_NOT_STARTED	0x808c000a	The SceneMapping library has not been started yet
SCE_SMART_ERROR_NOT_REQUIRED	0x808c000b	libsmart is not yet ready to process the request
SCE_SMART_ERROR_BUSY	0x808c000e	libsmart is locked by a call from another thread

Description

This function executes recognition processing on input data prepared with sceSmartSceneMappingDispatchAndQuery(). The recognition results can be obtained by calling sceSmartSceneMappingDispatchAndQuery() after recognition processing is complete.

See Also

sceSmartSceneMappingDispatchAndQuery()

sceSmartSceneMappingSaveMap

Save the current scene map

Definition

```
#include <scene_mapping.h>
SceInt32 sceSmartSceneMappingSaveMap(
        SceSmartStreamOut *stream
);
```

Arguments

[in] *stream* Output stream

Return Values

Returns SCE_OK (=0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_OUT_OF_MEMORY	0x808c0002	Cannot allocate memory for starting the SceneMapping library
SCE_SMART_ERROR_INVALID_POINTER	0x808c0006	*stream* is NULL
SCE_SMART_ERROR_NOT_STARTED	0x808c000a	The SceneMapping library has not been started yet
SCE_SMART_ERROR_BUSY	0x808c000e	libsmart is locked by a call from another thread

Description

This function saves the current scene map in a specified output stream. Saved maps can be loaded with sceSmartSceneMappingLoadMap().

Compatibility between PlayStation®Vita SDK versions and scene map file versions is as follows:

SDK Version	Scene Map File Version
1.650	1
1.800 or later	3

sceSmartSceneMappingSetDenseMapMode

Set DenseMap mode

Definition

```
#include <scene_mapping.h>
SceInt32 sceSmartSceneMappingSetDenseMapMode(
        SceSmartSceneMappingDenseMapMode mode
);
```

Arguments

[in] *mode* DenseMap mode to set

Return Values

Returns SCE_OK (=0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_INVALID_VALUE	0x808c0005	*mode* is invalid.
SCE_SMART_ERROR_ALREADY_STARTED	0x808c0009	libsmart is already started
SCE_SMART_ERROR_BUSY	0x808c000e	libsmart is locked by a call from another thread

Description

Because the DenseMap mode is currently not provided as a feature of the SceneMapping library, the SCE_SMART_SCENE_MAPPING_DENSE_MAP_DISABLE mode will be set regardless of the mode set with this function.

sceSmartSceneMappingSetCustomListener

Set custom listener

Definition

```
#include <scene_mapping.h>
SceInt32 sceSmartSceneMappingSetCustomListener (
        SceSmartSceneMappingCustomListener* listener
);
```

Arguments

[in] *listener* Structure defining the custom listener callback

Return Values

Returns SCE_OK (=0) for normal termination.

Returns the following error code (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_NOT_STOPPED	0x808c0003	libsmart is not stopped

Description

This function sets a custom listener. As an argument, pass the SceSmartSceneMappingCustomListener structure for defining the custom listener callback.

sceSmartSceneMappingSetDispatchMode

Set dispatch mode

Definition

```
#include <scene_mapping.h>
SceInt32 sceSmartSceneMappingSetDispatchMode(
        SceSmartSceneMappingDispatchMode mode
);
```

Arguments

[in] *mode* Dispatch mode

Return Values

Returns SCE_OK (=0) for normal termination.

Returns the following error code (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_NOT_STOPPED	0x808c0003	libsmart is not stopped

Description

This function sets the method to transfer input in the sceSmartSceneMappingDispatchAndQuery() and sceSmartSceneMappingRunCore() functions. For details on the dispatch mode, refer to the "SceSmartSceneMappingDispatchMode" section.

sceSmartSceneMappingStart

Start up the SceneMapping library

Definition

```
#include <scene_mapping.h>
SceInt32 sceSmartSceneMappingStart(
        SceSmartSceneMappingInitMode mode
);
```

Arguments

[in] *mode* Initialization mode

Return Values

Returns SCE_OK (=0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_OUT_OF_MEMORY	0x808c0002	Cannot allocate memory for starting the SceneMapping library
SCE_SMART_ERROR_INVALID_VALUE	0x808c0005	*mode* is invalid
SCE_SMART_ERROR_ALREADY_STARTED	0x808c0009	libsmart is already started
SCE_SMART_ERROR_BUSY	0x808c000e	libsmart is locked by a call from another thread

Description

This function starts up the SceneMapping library.

sceSmartSceneMappingStop

Stop the SceneMapping library

Definition

```
#include <scene_mapping.h>
SceInt32 sceSmartSceneMappingStop(void);
```

Arguments

None

Return Values

Returns SCE_OK (=0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_NOT_STARTED	0x808c000a	The SceneMapping library has not been started yet
SCE_SMART_ERROR_BUSY	0x808c000e	libsmart is locked by a call from another thread

Description

This function stops the SceneMapping library. The SceneMapping library's memory for calculation will be released.

sceSmartSceneMappingUnregisterTarget

Unregister recognition targets

Definition

```
#include <scene_mapping.h>
SceInt32 sceSmartSceneMappingUnregisterTarget(
        const SceInt32 targetID
);
```

Arguments

[in] *targetID* Recognition target ID to unregister

Return Values

Returns SCE_OK (=0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_SMART_ERROR_UNINITIALIZED	0x808c0000	libsmart is not initialized
SCE_SMART_ERROR_INVALID_VALUE	0x808c0005	*targetID* is invalid.
SCE_SMART_ERROR_NOT_REGISTERED	0x808c0008	Specified recognition target is not registered
SCE_SMART_ERROR_BUSY	0x808c000e	libsmart is locked by a call from another thread

Description

This function removes the recognition target with the specified ID from the SceneMapping library's recognition candidate list.

For *targetID*, specify the ID assigned by libsmart when creating the recognition target with sceSmartCreateLearnedImageTarget().

Callbacks

SceSmartSceneMappingCustomListenerOnLocalizationRequested

Callback function for custom initialization when using a custom listener

Definition

```
#include <libsmart.h>
typedef void (*SceSmartSceneMappingCustomListenerOnLocalizationRequested)(
        void *dsc,
        const SceSmartSceneMappingInput* input,
        const SceSmartSceneMappingResult* result,
        SceSmartSceneMappingCustomListenerLocalization* localization,
        SceSmartSceneMappingCustomListenerLandmark* landmarks,
        SceUInt32 numLandmarks
);
```

Arguments

dsc	Descriptor
input	Input by sceSmartSceneMappingDispatchAndQuery()
result	SceneMapping library state in the previous frame
localization	Pose estimated in the callback
landmarks	Landmarks estimated in the callback
numLandmarks	Number of landmarks being tracked

Return Values

None

Description

This is a prototype of the callback function for performing custom initialization unique to the application.

The descriptor, input data, and state of the SceneMapping library in the previous frame will be passed as arguments. Estimate the pose within the callback function and store the results in **localization*.

Design the callback function so that *res* of *localization* becomes SCE_SMART_SCENE_MAPPING_CUSTOM_LISTENER_LOCALIZATION_RESPONSE_RECOGNIZED when pose estimation succeeds.

If the state of the SceneMapping library that is passed to *result* is SCE_SMART_SCENE_MAPPING_STATE_SEARCH or SCE_SMART_SCENE_MAPPING_STATE_LOCALIZE, the SceneMapping library will be started with the set pose as the initial value. If the state of the SceneMapping library is SCE_SMART_SCENE_MAPPING_STATE_SLAM and SCE_SMART_SCENE_MAPPING_CUSTOM_LISTENER_LOCALIZATION_RESPONSE_RECOGNIZED is returned, the pose of the SceneMapping library will be overwritten with the custom initialization pose.

For the SCE_SMART_SCENE_MAPPING_STATE_SEARCH state and upon first estimating the orientation, the 3D coordinates of landmarks must be estimated in addition to the pose and passed to the library. If the number of points set with coordinates is too small, the library will transition to SCE_SMART_SCENE_MAPPING_STATE_LOCALIZE_IMPOSSIBLE.

Moreover, landmarks detected with this callback will not be passed to the library in any state other than SCE_SMART_SCENE_MAPPING_STATE_SEARCH. If necessary, use SceSmartSceneMappingCustomListenerOnLandmarkDetected().

SceSmartSceneMappingCustomListenerOnLandmarkDetected

Callback function for landmark detection when using a custom listener

Definition

```
#include <libsmart.h>
typedef void (*SceSmartSceneMappingCustomListenerOnLandmarkDetected)(
        void *dsc,
        const SceSmartSceneMappingInput* input,
        const SceSmartSceneMappingResult* result,
        SceSmartSceneMappingCustomListenerLandmark* landmarks,
        SceUInt32 numLandmarks
);
```

Arguments

dsc	Descriptor
input	Input by `sceSmartSceneMappingDispatchAndQuery()`
result	SceneMapping library state in the previous frame
landmarks	Landmark information
numLandmarks	Number of landmarks being tracked

Return Values

None

Description

This is a prototype of the callback function for performing landmark detection unique to the application.

When the library state is `SCE_SMART_SCENE_MAPPING_STATE_SLAM`, feature point candidates are passed and this callback will be called. As the *`initialPoint`* member of *`landmarks`*, 2D coordinates of feature point candidates will be passed within the callback, determine 3D coordinates in some manner and set them to the *`pos`* member of the same *`landmarks`*. Landmark state can be set with the *`res`* member.

The SceneMapping library continues feature points for tracking. Even if the value passed by the callback is `SCE_SMART_SCENE_MAPPING_CUSTOM_LISTENER_LANDMARK_RESPONSE_DEFERRED` or `SCE_SMART_SCENE_MAPPING_CUSTOM_LISTENER_LANDMARK_RESPONSE_UPTO_START`, it is possible to add as a landmark if coordinate estimation succeeds within the library. Tracking of feature points by the SceneMapping library itself is deleted with `SCE_SMART_SCENE_MAPPING_CUSTOM_LISTENER_LANDMARK_RESPONSE_DISCARD`.

Note that a landmark registered with this function can still be deleted in the library.

Constants

List of Constants

List of constants of SceneMapping APIs

Definition

Value	(Number)	Description
SCE_SMART_FLAG_FRESH_INPUT	1 << 0	Value of the flag indicating that the data to be input in sceSmartSceneMappingRunCore() is ready
SCE_SMART_SCENE_MAPPING_ CUSTOM_LISTENER_MAX_NUM_LANDMARKS	512	Maximum number of custom landmarks
SCE_SMART_SCENE_MAPPING_ MAX_NUM_INITIALIZATION_POINTS	64	Maximum number of usable initialization points
SCE_SMART_SCENE_MAPPING_ MAX_NUM_LANDMARKS	512	Maximum number of usable landmarks
SCE_SMART_SCENE_MAPPING_ MAX_NUM_NODE_POINTS	2048	Maximum number of usable nodes
SCE_SMART_SCENE_MAPPING_ MAX_PROPAGATION_DURATION	3000000	Longest duration allowed for the propagation of estimation results [μ sec]