libsystemgesture Overview

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Document serial number: 000004892117

Table of Contents

1 Library Overview

Scope of this Document

This document is for the libsystemgesture library that provides the gestures used by the GUI of the system software. It introduces the procedure for using the five touch gestures currently supported by this library, namely Tap, Drag, Tap and Hold, Pinch Out/In, and Rotation within game applications.

Purpose and Features

libsystemgesture is a library that provides gestures that are currently used by the GUI of the system software or that may be used in the future. Gestures refer to operations performed continuously by the user in relation to the device. Such operations include continuous touch operations in relation to the touch panel, operations to change the orientation of the device in relation to the ground, and operations tilting the device left and right.

The GUI of the system software itself also uses gestures, and by using this library, gestures with the same operational feeling as the GUI of the system software can be used within game applications. Therefore, it is recommended to try out this library first when wishing to use gestures similar to those used by the GUI of the system software within game applications.

Currently, libsystemgesture supports five gestures, namely Tap, Drag, Tap and Hold, Pinch Out/In, and Rotation (all gestures use the touch panel). The gesture lineup may change according to the implementation of the GUI of the system software. Provisional values are used for the thresholds and other parameters used for each gesture, and these values will be finalized later through a process of fine tuning.

Files

The following files are required in order to use libsystemgesture.

File Name	Description
libSceSystemGesture_stub.a	Stub library file
systemgesture.h	Header file

Sample Programs

Samples are provided in the following directories as examples of programs that use libsystemgesture.

sample_code/engines/api_systemgesture/

This is a sample of the basic use of libsystemgesture.

sample_code/engines/demo_systemgesture/

This is a demonstration of how to operate graphic objects using libsystemgesture.

Definitions of Terms

The terms used in this document are defined below.

Term	Definition
Gesture	An operation that is continuously performed by the user in relation to the device
Touch gesture	A gesture that uses the touch panel
Touch event	A parameter change, such as touch position or state, related to the touch panel

2 libsystemgesture Usage Procedure

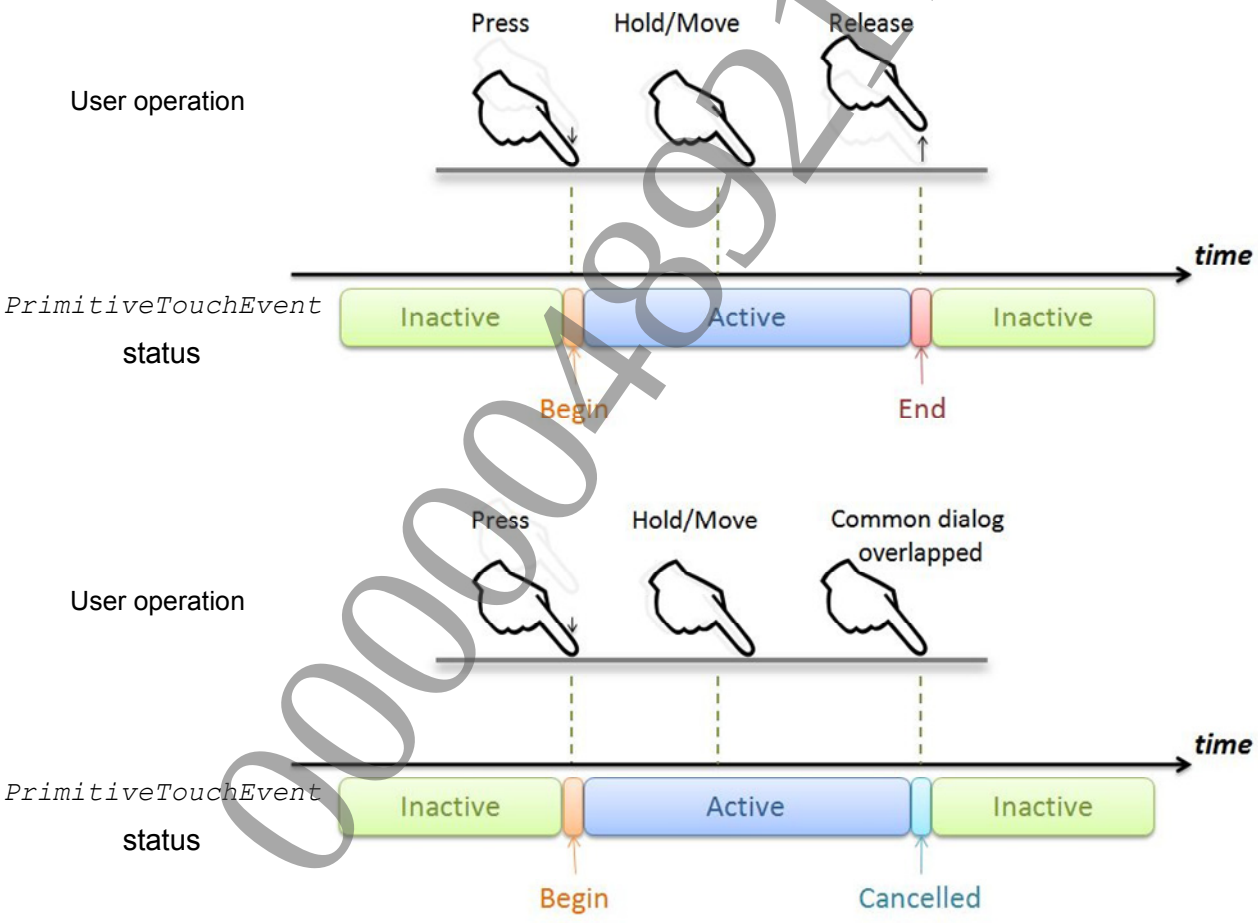
This chapter explains the judgment flow for the touch gestures provided by libsystemgesture and the basic usage procedure. For details on the various touch gestures, refer to the following chapter.

libsystemgesture touch gesture recognition flow

libsystemgesture divides touch gesture recognition into two steps. First, recognition of primitive touch gestures is done, and based on that information, recognition of touch gestures one level above, such as Tap and Drag, is done.

A primitive touch gesture is a touch gesture that has five states, namely touching the touch panel with a finger (Begin), maintaining contact with the touch panel with a finger (Active), releasing the touch panel by lifting one's finger off the touch panel (End), not touching the touch panel with a finger (Inactive), and cancelling the touch gesture (Cancelled). (Figure 1) Primitive touch gestures always cycle through the following sequence of states: Inactive -> Begin -> Active -> End -> Inactive.

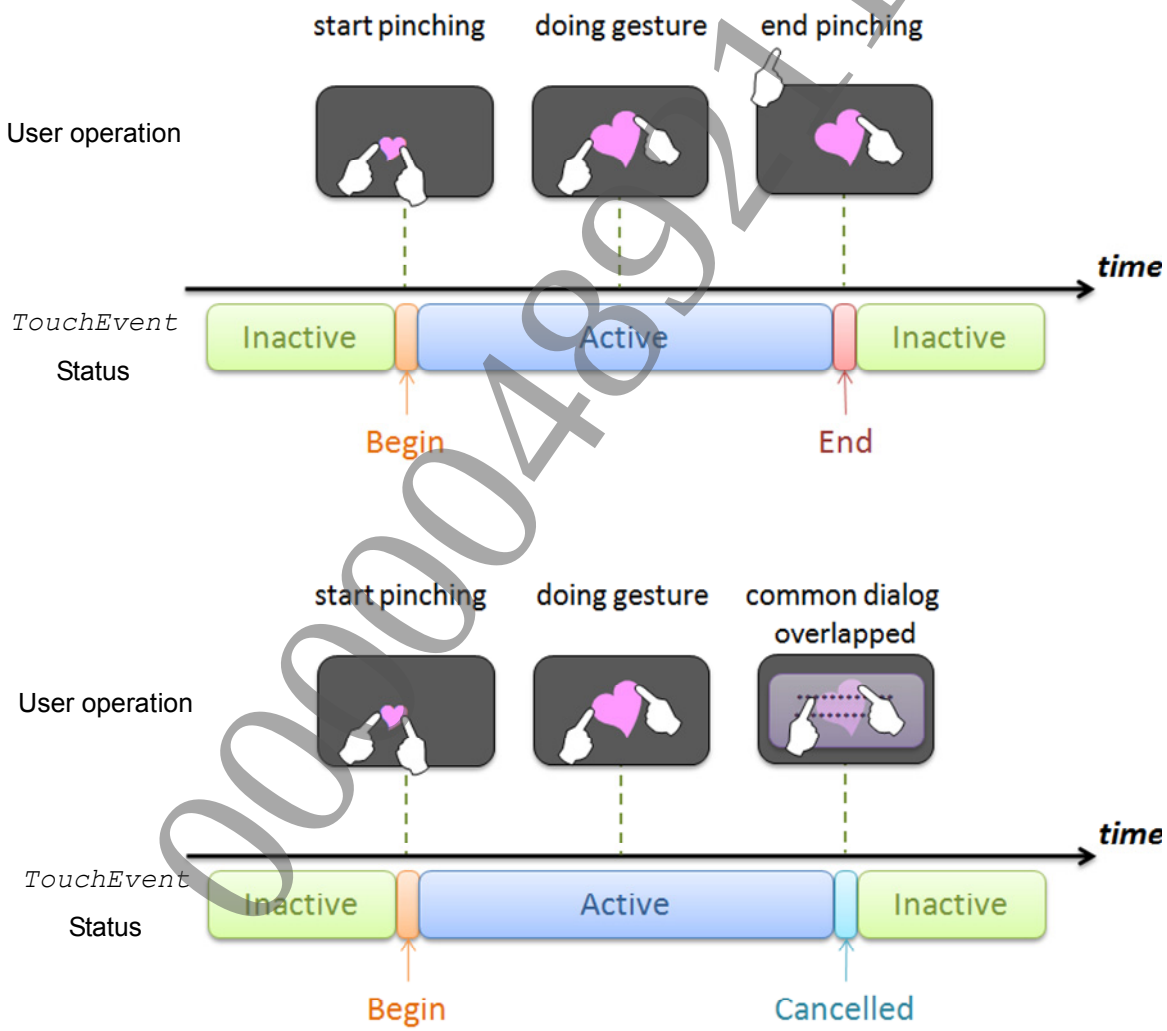
Figure 1 User Operation Flow and Primitive Touch Event



However, note that if a finger's contact point comes into the system management area (i.e. an area managed by a high-priority application or some kind of system service), such as when drawing is performed through Common Dialog, the gesture is considered to be cancelled and will transition to the Cancelled state. As an exception, given that the system messages and "Notifications" displayed when a game application is running in full-screen do not affect touch operation, transition to Cancelled state will not occur. Transition to Cancelled state will also occur if the interval for updating primitive touch gestures with `sceSystemGestureUpdatePrimitiveTouchRecognizer()` (described below) exceeds a given period of time (200 msec).

Recognition of the Tap and Drag touch gestures is done based on "primitive touch events". Because of this, it is necessary to update the applicable gesture recognizer by calling `sceSystemGestureUpdateTouchRecognizer()` every time a primitive touch gesture is updated with `sceSystemGestureUpdatePrimitiveTouchRecognizer()`. Similarly to primitive touch gestures, these touch gestures cycle through five states (Inactive, Begin, Active, End, and Cancelled), and their information can be acquired as "touch event (`SceSystemGestureTouchEvent`)". (Figure 2) However, the states that can be acquired differ depending on the touch gesture type. For details, refer to Chapter 3.

Figure 2 User Operation Flow and Touch Gesture (Pinch Out/In Gesture Example)



Basic Usage Procedure

Since libsystemgesture uses data acquired by Touch Service, following Touch Service initialization or data update, processing related to libsystemgesture is added. The basic usage procedure is as follows.

(1) Preliminary preparations for Touch Service

Using the `sceTouchSetSamplingState()` function, make the settings of the touch panel used for sampling by Touch Service. Also acquire the information of the touch panel as needed with the `sceTouchGetPanelInfo()` function.

(2) libsystemgesture initialization processing

libsystemgesture is offered as a system module and explicit module loading and initialization are required before it is used. Call the `sceSysmoduleLoadModule()` function (kernel function) to load libsystemgesture. The module ID of libsystemgesture being `SCE_SYSMODULE_SYSTEM_GESTURE`, perform loading as follows.

```
ret = sceSysmoduleLoadModule(SCE_SYSMODULE_SYSTEM_GESTURE);
```

(3) Initialization of primitive touch gesture recognizer

Call the `sceSystemGestureInitializePrimitiveTouchRecognizer()` function to initialize the primitive touch gesture recognizer.

```
ret = sceSystemGestureInitializePrimitiveTouchRecognizer(NULL);
```

(4) Creation of touch gesture recognizers

Call the `sceSystemGestureCreateTouchRecognizer()` function to create a recognizer for a touch gesture to be recognized. For how to create the various touch gesture recognizers, refer to Chapter 3.

```
ret = sceSystemGestureCreateTouchRecognizer(&tapReocgnizer,
                                            SCE_SYSTEM_GESTURE_TYPE_TAP,
                                            SCE_TOUCH_PORT_FRONT,
                                            NULL,
                                            NULL);
```

(5) Data acquisition from Touch Service

Call the `sceTouchRead()` function or the `sceTouchPeek()` function to acquire the touch data.

```
SceTouchData tdf, tdb;
int ret;
// Acquire data from front touch panel
ret = sceTouchRead(SCE_TOUCH_PORT_FRONT, &tdf, 1);
// Acquire data from rear touch panel
ret = sceTouchRead(SCE_TOUCH_PORT_BACK, &tdb, 1);
```

(6) Update of primitive touch gesture recognizer

Using the acquired touch data, update the primitive touch gesture recognizer.

```
ret = sceSystemGestureUpdatePrimitiveTouchRecognizer(&tdf, &tdb);
```

(7) Update of touch gesture recognizers

Update the touch gesture recognizers created in step (4). After carrying out step (6), always update the applicable gesture recognizer.

```
ret = sceSystemGestureUpdateTouchRecognizer(&tapRecognizer);
```

(8) Acquisition and referencing of primitive touch events

Call a function such as the `sceSystemGestureGetPrimitiveTouchEvents()` function to acquire primitive touch events. Primitive touch event acquisition can be done in one of the following three ways.

(8-1) Acquisition of all primitive touch events at one time

Call the `sceSystemGestureGetPrimitiveTouchEvents()` function.

```
SceSystemGesturePrimitiveTouchEvent tEvent[24];
SceUInt32 numberOfEvent = 0;
ret = sceSystemGestureGetPrimitiveTouchEvents(&tEvent[0],24,&numberOfEvent);
for( int i=0; i<numberOfEvent; i++ )
{
        printf("[%d] ID = %d\n", tEvent[i].eventState, tEvent[i].primitiveID);
}
```

(8-2) Acquisition of primitive touch events one at a time

After calling the `sceSystemGestureGetPrimitiveTouchEventsCount()` function, call the `sceSystemGestureGetPrimitiveTouchEventByIndex()` function.

```
SceSystemGesturePrimitiveTouchEvent tEvent;
int eventCount = sceSystemGestureGetPrimitiveTouchEventsCount();
for( int i=0; i<eventCount; i++ )
{
        ret = sceSystemGestureGetPrimitiveTouchEventByIndex(i, &tEvent);
        printf("[%d] ID = %d\n", tEvent.eventState, tEvent.primitiveID);
}
```

(8-3) Acquisition of specific primitive touch event

Call the `sceSystemGestureGetPrimitiveTouchEventByPrimitiveID()` function.

```
SceSystemGesturePrimitiveTouchEvent tEvent;
ret = sceSystemGestureGetPrimitiveTouchEventByPrimitiveID(primitiveID,
                                                          &tEvent);
```

(9) Acquisition and referencing of touch events

Call a function such as the `sceSystemGestureGetTouchEvents()` function to acquire the touch events from the touch gesture recognizers. Similarly to primitive touch events, touch event acquisition can be done in one of the following three ways.

(9-1) Acquisition of all touch events at one time

Call the `sceSystemGestureGetTouchEvents()` function.

```
SceSystemGestureTouchEvent tEvent[6];
SceUInt32 numberOfEvent = 0;
ret = sceSystemGestureGetTouchEvents(&tRecognizer,
                                     &tEvent[0], 6, &numberOfEvent);
for( int i=0; i<numberOfEvent; i++ )
{
        printf("[%d] ID = %d\n", tEvent[i].eventState, tEvent[i].eventID);
}
```

(9-2) Acquisition of touch events one at a time

After calling the `sceSystemGestureGetTouchEventsCount()` function, call the `sceSystemGestureGetTouchEventByIndex()` function.

```
SceSystemGestureTouchEvent tEvent;
int eventCount = sceSystemGestureGetTouchEventsCount(&tRecognizer);
for( int i=0; i<eventCount; i++ )
{
        ret = sceSystemGestureGetTouchEventByIndex(&tRecognizer, i, &tEvent);
        printf("[%d] ID = %d\n", tEvent.eventState, tEvent.eventID);
}
```

(9-3) Acquisition of specific touch event

Call the `sceSystemGestureGetTouchEventByEventID()` function.

```
SceSystemGestureTouchEvent tEvent;
ret = sceSystemGestureGetTouchEventByEventID(&tRecognizer,
                                             eventID,
                                             &tEvent);
```

(10) Termination processing of libsystemgesture

When libsystemgesture is no longer needed, call the `sceSystemGestureFinalizePrimitiveTouchRecognizer()` function to perform the termination processing of the primitive touch gesture recognizer. No particular termination processing is required for the touch gesture recognizers created with the `sceSystemGestureCreateTouchRecognizer()` function.

Lastly, stop and unload the system module.

```
ret = sceSysmoduleUnloadModule(SCE_SYSMODULE_SYSTEM_GESTURE);
```

3 Explanation of Available Touch Gestures and their Usage Method

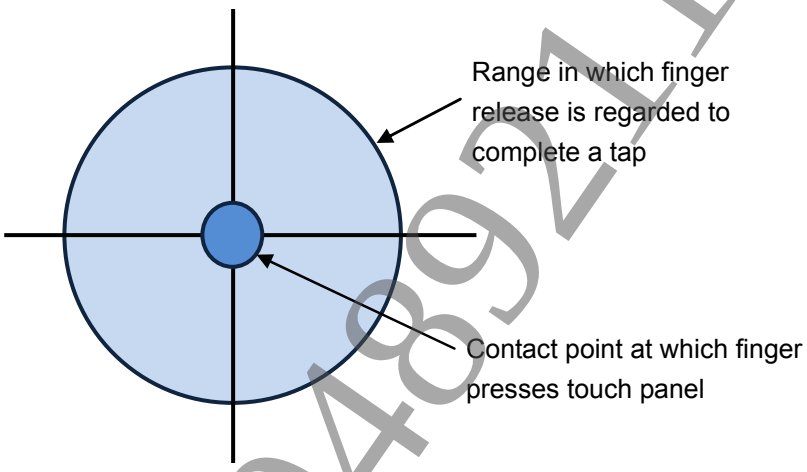
This chapter describes the available touch gestures provided by libsystemgesture and how to use them.

Tap

Definition of Tap gesture

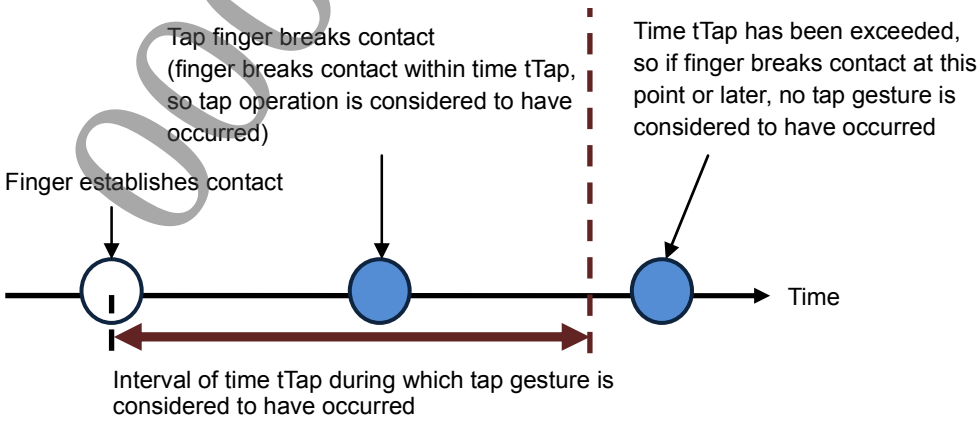
The Tap gesture consists in placing a finger on one point of the screen and then lifting it off the screen. A Tap gesture is considered to have occurred if the finger is removed from the screen within a given distance range from where the finger first established contact with the touch panel. If the finger travels outside the preset range, that Tap gesture is considered to have been canceled by the user.

Figure 3 Tap Gesture Realization Conditions



A time limitation applies to the realization of a Tap gesture. A Tap gesture is considered to have occurred when the finger is removed from the screen within a given time interval from when the finger first established contact with the touch panel. If this time interval is exceeded, no Tap gesture is considered to have occurred.

Figure 4 Time Condition for Tap Gesture



Usage Method

The Usage method is explained along with Double-Tap.

Double-Tap

Definition of Double-Tap gesture

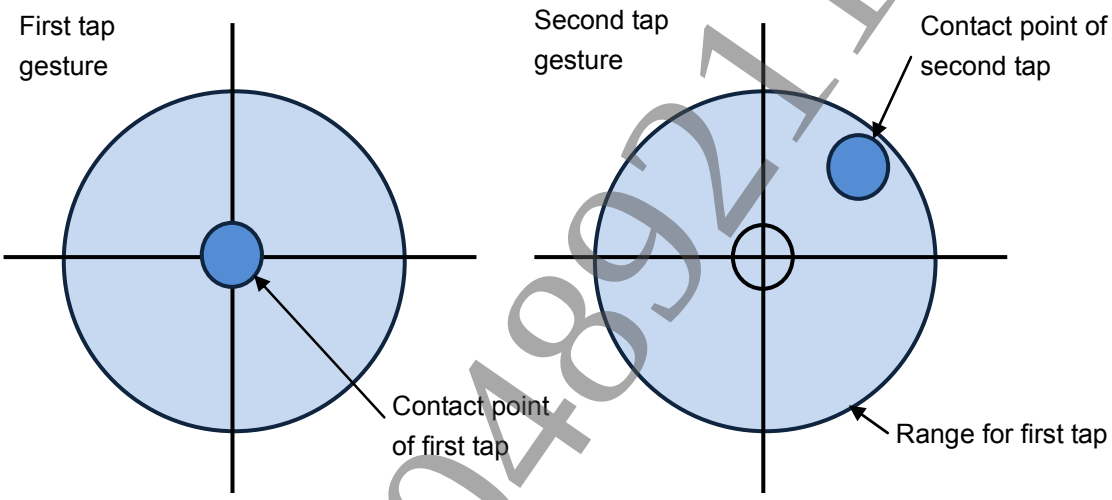
A Double-Tap gesture consists in two consecutive tap gestures on the screen. General Double-Tap gestures are as follows.

(1) Similarly to a normal Tap gesture, the first Tap gesture is considered to occur when the finger is removed from the touch panel within a given distance range from where the finger first established contact with the touch panel (Active state).

(2) The second Tap gesture is considered to have begun when the finger touches the screen within the range considered to be a Tap gesture defined by the first tap gesture. The second Tap gesture too is considered to have occurred when the finger is removed from the touch screen within a given range from the second contact point.

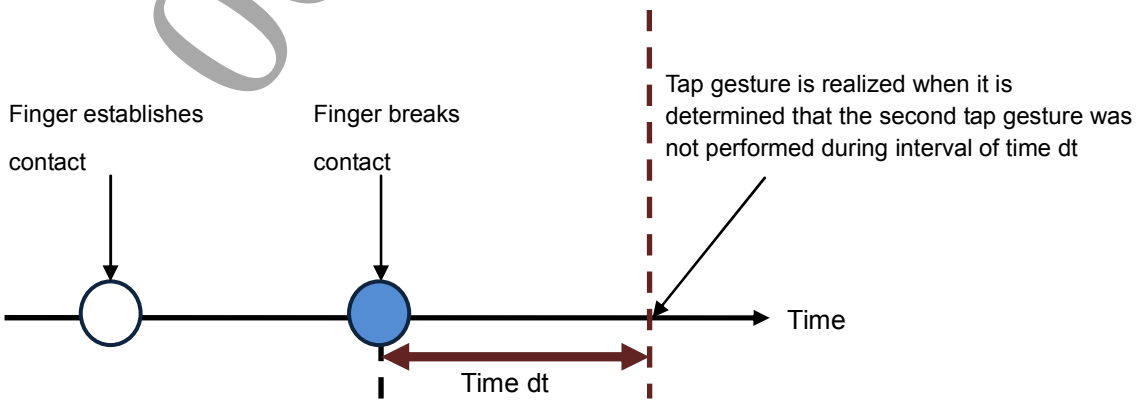
For example, in a case such as Figure 5, the contact point of the second Tap gesture falls within the range in which the first tap was realized, so that contact of the second Tap is considered to have occurred.

Figure 5 Conditions for Contact Point of Second Tap Gesture for Double-Tap Gesture



Usually Double-Tap gestures include a time condition. In this case, if after the first tap gesture is completed and the finger is removed from the screen, the finger does not contact the screen within a specified time interval to perform the second Tap gesture, the Double-Tap gesture is considered to have been canceled. Note that when Tap gestures and Double-Tap gestures are used together, a Tap gesture must be recognized after verifying that the second tap gesture of a Double-Tap gesture was not performed, the timing at which Tap gesture recognition occurs differs compared with when only Single-Tap gestures are used.

Figure 6 Tap Gesture Realization Conditions when both Tap and Double-Tap Are Supported



The point where a Double-Tap gesture is considered to have occurred is not the contact point of the last Tap gesture, but the point of the first Tap gesture. In the case of three continuous taps, the same approach as above can be assumed, and in libsystemgesture, recognition is done by one touch gesture recognizer without differentiating between Single-Taps and multiple continuous Taps.

Usage method

To use Single-Taps and multiple Tap gestures in mix, initialize the `SceSystemGestureTouchRecognizer` structure for touch gesture recognizers by specifying `SCE_SYSTEM_GESTURE_TYPE_TAP` for `SceSystemGestureType` as the gesture type. In the following example, a Tap gesture recognizer is created using the entire front touch panel as the target area. If NULL is specified as the gesture recognizer initialization parameter, multiple consecutive Taps are not recognized. In other words, when three consecutive Tap gestures are performed, three distinct Tap gesture events occur.

```
SceSystemGestureTouchRecognizer tapRecognizer;
ret = sceSystemGestureCreateTouchRecognizer(&tapRecognizer,
                                            SCE_SYSTEM_GESTURE_TYPE_TAP,
                                            SCE_TOUCH_PORT_FRONT,
                                            NULL,
                                            NULL);
```

To use multiple consecutive Taps such as Double-Taps, set the maximum number of Taps that can be recognized as a continuous gesture in the *`maxTapCount`* member of the `SceSystemGestureTapRecognizerParameter` structure.

```
SceSystemGestureTouchRecognizer doubleTapRecognizer;

//union of parameters for initialization of gesture recognizers
SceSystemGestureTouchRecognizerParameter doubleTapParam;

doubleTapParam.tap.maxTapCount = 2;
ret = sceSystemGestureCreateTouchRecognizer(&doubleTapRecognizer,
                                            SCE_SYSTEM_GESTURE_TYPE_TAP,
                                            SCE_TOUCH_PORT_FRONT,
                                            NULL,
                                            &doubleTapParam);
```

In the case of this *`doubleTapRecognizer`*, if two consecutive Tap gestures are performed, a Tap gesture event occurs at the timing of the release of the second Tap. If only one Tap was performed, the Tap gesture event occurs with a brief delay from the release of the first Tap. If three consecutive Tap gestures are performed, a Tap gesture event occurs at the timing of the release of the second Tap and another Tap gesture event occurs with a brief delay from the third Tap.

Call the `sceSystemGestureUpdateTouchRecognizer()` function in each frame for the Tap gesture recognizer created as described above.

```
ret = sceSystemGestureUpdateTouchRecognizer(&tapRecognizer)
ret = sceSystemGestureUpdateTouchRecognizer(&doubleTapRecognizer)
```

Tap gesture events can be acquired as follows from an updated Tap gesture recognizer.

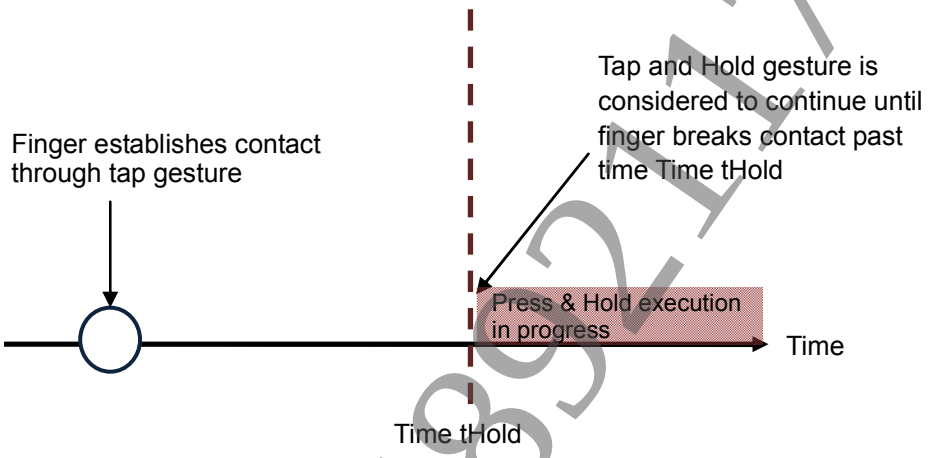
```
SceSystemGestureTouchEvent touchEvent;
int eventCount = sceSystemGestureGetTouchEventsCount(&doubleTapRecognizer);
for( int i=0; i<eventCount; i++ )
{
  ret = sceSystemGestureGetTouchEventByIndex(&tapRecognizer, i, &touchEvent);
  printf("Tap!! tappedCount = %d\n", touchEvent.property.tap.tappedCount);
}
```

Tap and Hold

Definition of Tap and Hold gesture

A Tap and Hold gesture is considered to have occurred when the finger stays in contact with the screen within a given distance range of the initial contact point and for a given time. A Tap and Hold gesture is considered to begin not at the time point when the finger breaks contact but after the lapse of a given time interval (Begin state), and it is considered to continue while the finger remains in contact with the screen (Active state). The gesture is considered to end when the finger breaks contact or moves outside the set range (End state). After the gesture is started, if a finger's contact point comes into the system management area (i.e. an area managed by a high-priority application or some kind of system service), such as when drawing is performed through Common Dialog, the gesture is considered to be cancelled (Cancelled state).

Figure 7 Time Condition for Tap and Hold Gesture



Usage method

To use Tap and Hold gestures, initialize the `SceSystemGestureTouchRecognizer` structure for the touch gesture recognizer by specifying `SCE_SYSTEM_GESTURE_TYPE_TAP_AND_HOLD` for `SceSystemGestureType` as the gesture type. In the following example, a Tap and Hold gesture recognizer is created using the entire front touch panel as the target area. If NULL is specified as the gesture recognizer initialization parameter, a gesture recognizer with the tHold value of 1000 ms in Figure 7 is created.

```
SceSystemGestureTouchRecognizer tapAndHoldRecognizer;
ret = sceSystemGestureCreateTouchRecognizer(&tapAndHoldRecognizer,
                                SCE_SYSTEM_GESTURE_TYPE_TAP_AND_HOLD,
                                            SCE_TOUCH_PORT_FRONT,
                                            NULL,
                                            NULL);
```

To change the value of tHold, set a time to the *`timeToInvokeEvent`* member of the `SceSystemGestureTapAndHoldRecognizerParameter` structure as follows and initialize the `SceSystemGestureTouchRecognizer` structure.

```
SceSystemGestureTouchRecognizer tapAndHoldRecognizer;

//union of parameters for initialization of gesture recognizers
SceSystemGestureTouchRecognizerParameter tapAndHoldParam;

tapAndHoldParam.tapAndHold.timeToInvokeEvent = 2000;
ret = sceSystemGestureCreateTouchRecognizer(&tapAndHoldRecognizer,
                                SCE_SYSTEM_GESTURE_TYPE_TAP_AND_HOLD,
```

```
                                                SCE_TOUCH_PORT_FRONT,
                                                NULL,
                                                &tapAndHoldParam);
```

Call the `sceSystemGestureUpdateTouchRecognizer()` function in each frame for the Tap and Hold gesture recognizer created as described above.

```
ret = sceSystemGestureUpdateTouchRecognizer(&tapAndHoldRecognizer)
```

Tap and Hold gesture events can be acquired as follows from an updated Tap and Hold gesture recognizer.

```
SceSystemGestureTouchEvent touchEvent;
int eventCount = sceSystemGestureGetTouchEventsCount(&tapAndHoldRecognizer);
for( int i=0; i<eventCount; i++ )
{
  ret = sceSystemGestureGetTouchEventByIndex(&tapAndHoldRecognizer,
                                              i, &touchEvent);
  printf("Tap and Hold!! \n");
}
```

Drag

Definition of Drag gesture

A drag gesture consists in moving an object by dragging the finger and tracing the course on the touch panel after establishing contact.

When using Drag gestures, note that the motion of the object is affected by the graphic draw delay. Also, if the Drag gesture target is smaller than the size of the finger, the operation target will be hidden by the finger. Thus from the viewpoint of usability, it is desirable to select objects that are larger than fingers as gesture targets. Some operation examples using the Drag gesture are described below.

- Assignment of Drag gesture to operation target scroll
  (e.g., Assignment of screen scrolling to upward/downward Drag gestures)
- Partial selection
  (e.g., Selection of range by sliding finger through a Drag gesture over a list, character string, etc.)

Similarly to the Tap and Hold gesture, if a finger's contact point comes into the system management area (i.e. an area managed by a high-priority application or some kind of system service) after the gesture is started, such as when drawing is performed through Common Dialog, the gesture is considered to be cancelled (Cancelled state).

Usage method

To use Drag gestures, initialize the `SceSystemGestureTouchRecognizer` structure for the touch gesture recognizer by specifying `SCE_SYSTEM_GESTURE_TYPE_DRAG` for `SceSystemGestureType` as the gesture type. In the following example, a Drag gesture recognizer is created using the entire front touch panel as the target area.

```
SceSystemGestureTouchRecognizer dragRecognizer;
ret = sceSystemGestureCreateTouchRecognizer(&dragRecognizer,
                                            SCE_SYSTEM_GESTURE_TYPE_DRAG,
                                            SCE_TOUCH_PORT_FRONT,
                                            NULL,
                                            NULL);
```

Call the `sceSystemGestureUpdateTouchRecognizer()` function in each frame for the Drag gesture recognizer created as described above.

```
ret = sceSystemGestureUpdateTouchRecognizer(&dragRecognizer)
```

Drag gesture events can be acquired as follows from an updated Drag gesture recognizer.

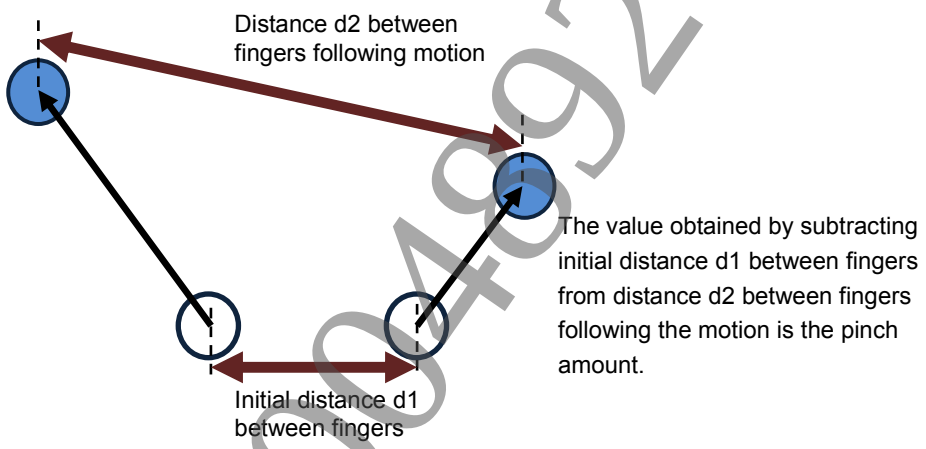
```
SceSystemGestureTouchEvent te;
int eventCount = sceSystemGestureGetTouchEventsCount(&dragRecognizer);
for( int i=0; i<eventCount; i++ )
{
  ret = sceSystemGestureGetTouchEventByIndex(&dragRecognizer,
                                             i, &te);
  printf("Dragged!! deltaVector = %d,%d\n", te.property.drag.deltaVector.x,
                                            te.property.drag.deltaVector.y);
}
```

Pinch Out/In

Definition of Pinch Out/In gesture

Pinch Out/In gestures are performed by using two fingers. In the state in which two fingers are in contact with the touch panel, the Pinch Out/In gesture consists in changing the distance between the two fingers, either increasing or decreasing it. The Pinch Out/In gesture is used mainly to magnify/reduce items such as pictures and objects.

Figure 8 Method to Calculate Pinch Amount of Pinch In/Out Operation

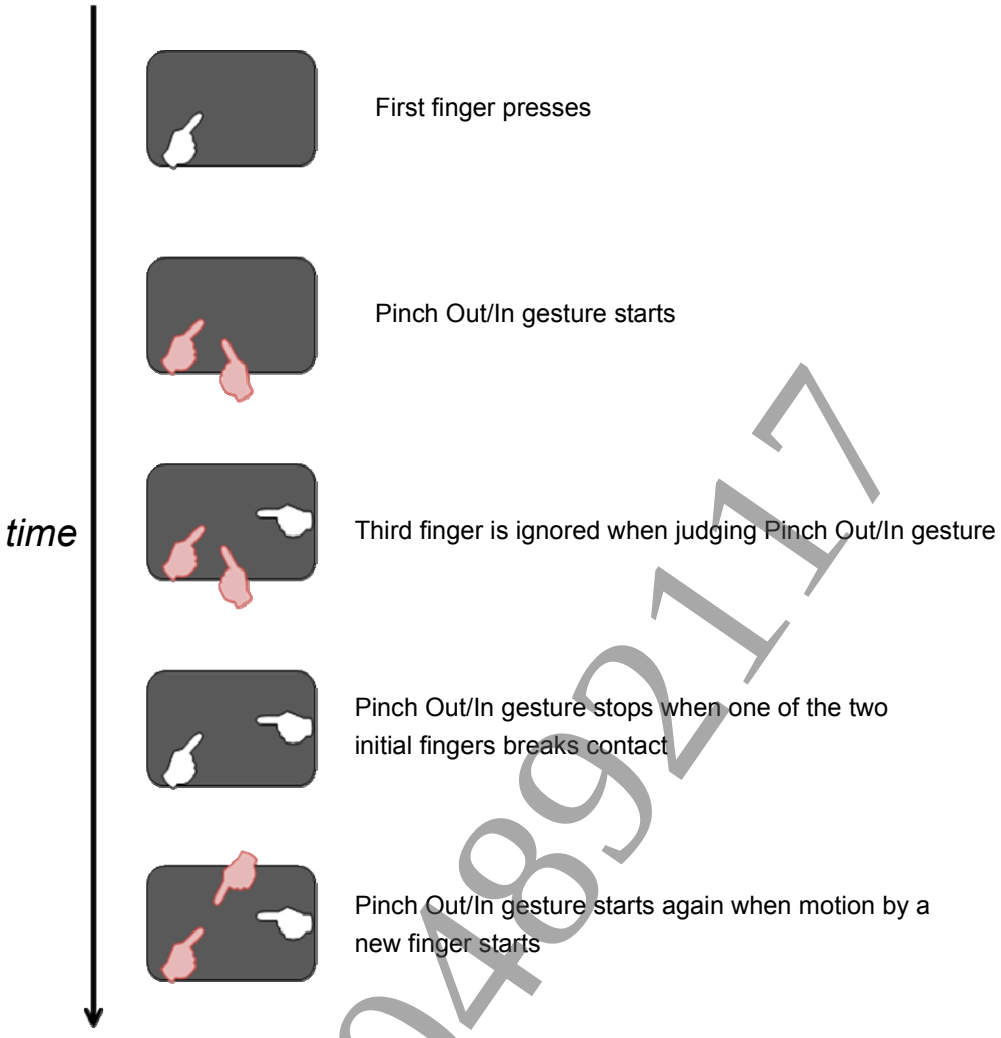


The directions in which the two fingers move is not prescribed for the Pinch Out/In gesture, and all directions, whether up/down, left/right, or diagonal, can be used for touch gestures, and only the distance between the two fingers is used to effect the gesture.

Judgment of Pinch Out/In gestures is done by the two fingers initially pressed within the target area. When the second finger has established contact with the touch panel within the target area, a Pinch Out/In gesture is considered to have begun (Begin state). If one of the two fingers that was initially pressed onto the touch panel breaks contact, or if it travels out of the target area, the Pinch Out/In gesture stops (End state). If later a third finger is pressed onto the touch panel, that finger is ignored. Note that when the first finger breaks contact with the touch panel, the Pinch Out/In gesture operation does not continue through the second and third fingers. If a fourth finger is additionally used in this state, the Pinch Out/In gesture resumes through the second and fourth fingers (Figure 9).

Also, after the gesture is started, if a finger's contact point comes into the system management area (i.e. an area managed by a high-priority application or some kind of system service), such as when drawing is performed through Common Dialog, the gesture is considered to be cancelled (Cancelled state).

Figure 9 Flow of Judgment of 2 Fingers Constituting Pinch Out/In Gesture



Usage method

To use Pinch Out/In gestures, initialize the `SceSystemGestureTouchRecognizer` structure for the touch gesture recognizer by specifying `SCE_SYSTEM_GESTURE_TYPE_PINCH_OUT_IN` for `SceSystemGestureType` as the gesture type. In the following example, a Pinch Out/In gesture recognizer is created with part of the rear touch panel as the target area.

```
SceSystemGestureTouchRecognizer pinchOutInRecognizer;

SceSystemGestureRectangle rect;
rect.x = 200;
rect.y = 0;
rect.width = 1920 - 200 * 2;
rect.height = 1088;

ret = sceSystemGestureCreateTouchRecognizer(&pinchOutInRecognizer,
                                            SCE_SYSTEM_GESTURE_TYPE_PINCH_OUT_IN,
                                                 SCE_TOUCH_PORT_BACK,
                                                 &rect,
                                                 NULL);
```

Call the `sceSystemGestureUpdateTouchRecognizer()` function in each frame for the Pinch Out/In gesture recognizer created as described above.

```
ret = sceSystemGestureUpdateTouchRecognizer(&pinchOutInRecognizer)
```

Pinch Out/In gesture events can be acquired as follows from an updated Pinch Out/In gesture recognizer.

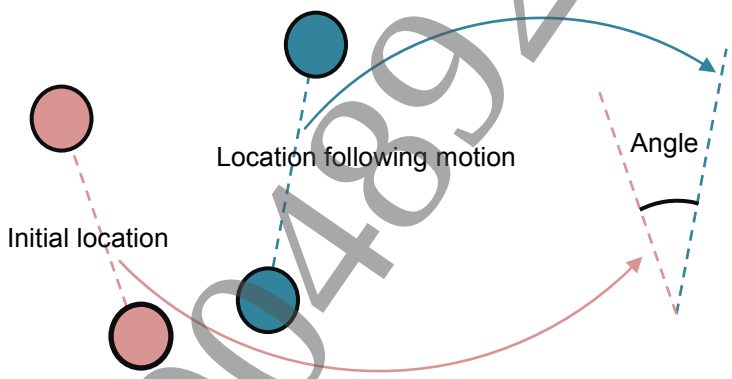
```
SceSystemGestureTouchEvent te;
int eventCount = sceSystemGestureGetTouchEventsCount(&pinchOutInRecognizer);
for( int i=0; i<eventCount; i++ )
{
  ret = sceSystemGestureGetTouchEventByIndex(&pinchOutInRecognizer,
                                             i, &te);
  printf("state = %d, scale = %lf\n", te.eventState,
                                      te.property.pinchOutIn.scale);
}
```

Rotation

Definition of Rotation gesture

Rotation gestures are performed by using two fingers. In the state in which two fingers are in contact with the touch panel, the Rotation gesture consists in rotating the two fingers, either clockwise or counterclockwise (Figure 10).

Figure 10 Rotation Gesture



The judgment flow of two fingers constituting the Rotation gesture and the state transition are exactly the same as Pinch Out/In gesture (Figure 9). It is possible to acquire the angle between the vectors formed by the two fingers at both the initial location and the location following motion.

Usage method

To use Rotation gestures, initialize the `SceSystemGestureTouchRecognizer` structure for the touch gesture recognizer by specifying `SCE_SYSTEM_GESTURE_TYPE_ROTATION` for `SceSystemGestureType` as the gesture type. In the following example, a Rotation gesture recognizer is created with part of the rear touch panel as the target area.

```
SceSystemGestureTouchRecognizer rotationRecognizer;

SceSystemGestureRectangle rect;
rect.x = 200;
rect.y = 0;
rect.width = 1920 - 200 * 2;
rect.height = 1088;

ret = sceSystemGestureCreateTouchRecognizer(&rotationRecognizer,
                                            SCE_SYSTEM_GESTURE_TYPE_ROTATION,
                                                SCE_TOUCH_PORT_BACK,
```

```
                                                    &rect,
                                                    NULL);
```

Call the `sceSystemGestureUpdateTouchRecognizer()` function in each frame for the Rotation gesture recognizer created as described above.

```
ret = sceSystemGestureUpdateTouchRecognizer(&rotationRecognizer)
```

Rotation gesture events can be acquired as follows from an updated Rotation gesture recognizer.

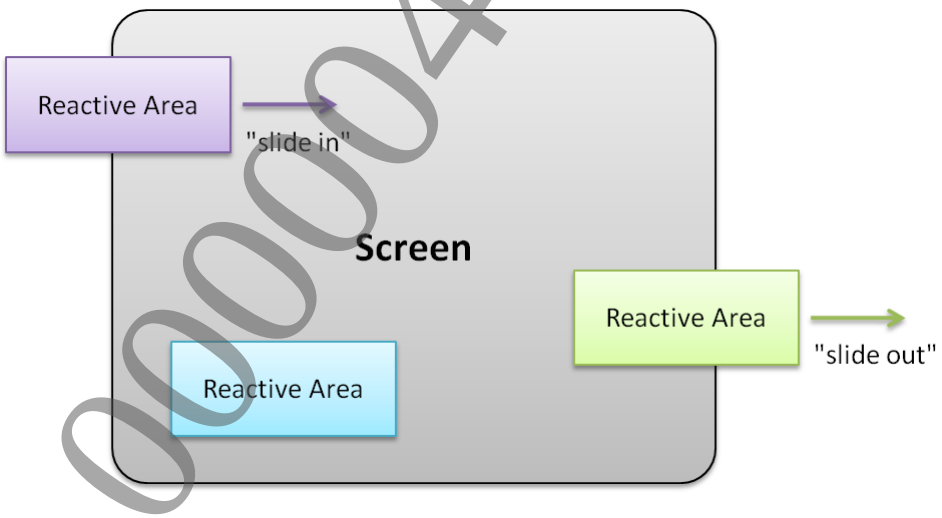
```
SceSystemGestureTouchEvent te;
int eventCount = sceSystemGestureGetTouchEventsCount(&rotationRecognizer);
for( int i=0; i<eventCount; i++ )
{
  ret = sceSystemGestureGetTouchEventByIndex(&rotationRecognizer,
                                                    i, &te);
  printf("state = %d, scale = %lf\n", te.eventState,
                                          te.property.rotation.angle);
}
```

Recognition Target Area of Gesture Recognizer

The recognition target area of each gesture recognizer can be updated at any given point in time by calling the `sceSystemGestureUpdateTouchRecognizerRectangle()` function. It is recommended to update the recognition target area before updating the gesture recognizer through the `sceSystemGestureUpdateTouchRecognizer()` function in each frame.

Moreover, a value outside the range of the touch panel can be specified for the recognition target area because it is conceivable that an area which reacts to the touch operation may "slides in" from outside the screen or "slides out" of the screen (Figure 11).

Figure 11 Location Example of Recognition Target Area of Gesture Recognizer



4 Precautions

Lineup of gestures offered by libsystemgesture

The gesture lineup may change according to the implementation of the GUI of the system software.

Parameters Used in libsystemgesture

In libsystemgesture, parameters shared with the GUI of the system software are set for the following touch gestures.

Tap

- Distance between Press position and Release position
- Time difference between Press and Release

Multiple consecutive Tap

- Distance between the release position during the immediately preceding Tap and the press position of the new Tap
- Difference between the release time during the immediately preceding Tap and the press time of the new Tap

Tap and Hold

- Default continuous time for considering a Tap and Hold gesture to have occurred
- Finger travel amount for considering a Tap and Hold gesture to have been canceled