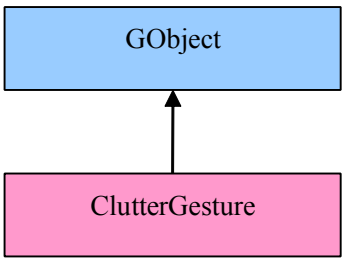
Clutter-Gesture External API

Ver0.5

Long Bu Long.bu@intel.com
Forrest Zhao forrest.zhao@intel.com

Overview

The touch gesture is abstracted with a ClutterGesture object which derives from GObject as below.



An application interested in touch gestures needs to create an instance of the ClutterGuesture object first and then configures some properties of the object. If an interesting gesture is recognized, a corresponding signal is emitted on the ClutterGesture object.

APIs

● clutter_gesture_new

SYNOPSIS

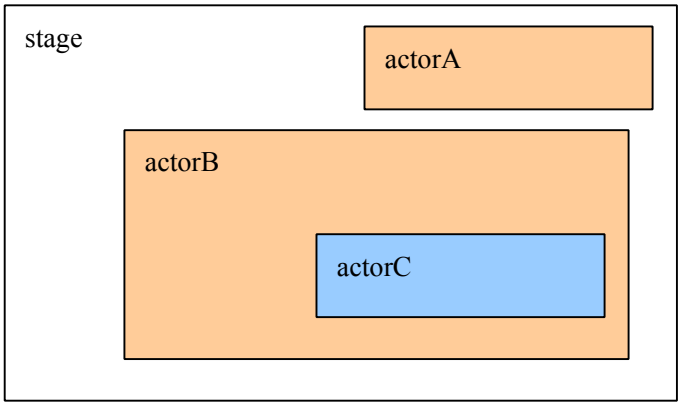
*ClutterGesture *clutter_gesture_new (ClutterActor *actor);*

DESCRIPTION

This function creates an instance of the ClutterGesture object.

The parameter **actor** is the parent of all actors that needs gesture events. Of course an application can always pass the stage. But for the sake of performance, an application should use the closest parent of all actors that needs gestures.

For example:



In the above example, actorC is a child of actorB. actorB and all 3 actors are children of the stage. If an application wants to do gesture recognition on actorB and actorC but not on actorA. So the best way is to pass actorB as the **actor** parameter.

RETURN VALUE

This function returns a pointer to the instance or NULL on error.

● clutter_gesture_set_gesture_mask

SYNOPSIS

*void clutter_gesture_set_gesture_mask (ClutterGesture *gesture, ClutterActor *actor, guint mask);*

DESCRIPTION

This function configures a ClutterGesture instance **gesture** in terms which actor is interested in which gesture events.

The parameter **gesture** is the ClutterGesture instance being configured.

The **actor** parameter specifies the actor on which gesture recognitions should be performed. It must be a descendant of the **actor** passed in *clutter_gesture_new.*

The parameter **mask** is a bit-ORed gesture events. ClutterGesture only does gesture recognitions for the events specified in **mask.** Now valid events are

GESTURE_MASK_SLIDE = 1,
GESTURE_MASK_HOLD = (1<<1)
GESTURE_MASK_PINCH = (1<<2),
GESTURE_MASK_ ROTATE = (1 << 3),
GESTURE_MASK_NAVIGATE = (1 << 4)

For example, if an application just wants to receive SLIDE gesture on an actor *foo.* Then the application could call:

clutter_gesture_set_gesture_mask (gesture, foo, GESTURE_MASK_SLIDE);

Gobject Singals

● slide signal

NAME

"gesture-slide-event"

CALLBACK PROTOTYPE

*gboolean gesture_slide_cb(ClutterGesture *gesture, ClutterGestureSlideEvent *event, gpointer data)*

```
typedef struct {
    ClutterGestureEventType type;     //gesture event type, GESTURE_SLIDE
    guint32 time;                     //event time, i.e. gesture end time
    ClutterEventFlags flags;          //event flags
    ClutterStage *stage;              //event source stage
    ClutterActor *source;             //event source actor
    guint32 start_time;               //gesture start time
    gint x_start;                     //X coordinate of start point
    gint y_start;                     //Y coordinate of start point
    gint x_end;                       //X coordinate of end point
    gint y_end;                       //Y coordinate of end point
    gint direction;                   //1: up, 2: down, 3: left, 4: right
    gint device_id;                   //the device id on which the slide is active
} ClutterGestureSlideEvent;
```

DESCRIPTION

This signal will be emitted when a slide gesture is recognized. An application may use *g_signal_connect (gesture, "gesture-slide-event", G_CALLBACK(your_slide_cb), your_data)*
to connect a callback to the signal

● hold signal

NAME

"gesture-hold-event"

CALLBACK PROTOTYPE

*gboolean gesture_hold_cb(ClutterGesture *gesture, ClutterGestureHoldEvent *event,*

gpointer data)

```
typedef struct {
   ClutterGestureEventType type;          //gesture event type, GESTURE_HOLD
   guint32 time;                          //event time
   ClutterEventFlags flags;               //event flags
   ClutterStage *stage;                   //event source stage
   ClutterActor *source;                  //event source actor
   gfloat x;                              //X coordinate of hold point
   gfloat y;                              //Y coordinate of hold point
}ClutterGestureHoldEvent;
```

DESCRIPTION

This signal will be emitted when a touch_and_hold gesture is recognized. An application may use

g_signal_connect (gesture, "gesture-hold-event", G_CALLBACK(your_hold_cb), your_data)

to connect a callback to the signal

● pinch signal

NAME

"gesture-pinch-event"

CALLBACK PROTOTYPE

*gboolean gesture_pinch_cb(ClutterGesture *gesture, ClutterGesturePinchEvent *event, gpointer data)*

```
typedef struct {
    ClutterGestureEventType type;   //gesture event type, GESTURE_PINCH
    guint32 time;                   //event time
    ClutterEventFlags flags;        //event flags
    ClutterStage *stage;            //event source stage
    ClutterActor *source;           //event source actor
    guint32 start_time;             //gesture start time
    gint x_start_1;                 //X coordinate of start point of first finger
    gint y_start_1;                 //Y coordinate of start point of first finger
    gint x_end_1;                   //X coordinate of end point of first finger
    gint y_end_1;                   //Y coordinate of end point of first finger
    gint x_start_2;                 //X coordinate of start point of second finger
    gint y_start_2;                 //Y coordinate of start point of second finger
    gint x_end_2;                   //X coordinate of end point of second finger
    gint y_end_2;                   //Y coordinate of end point of second finger
} ClutterGesturePinchEvent;
```

DESCRIPTION

This signal will be emitted when a pinch gesture is recognized. An application may use *g_signal_connect (gesture, "gesture-pinch-event", G_CALLBACK(your_pinch_cb), your_data)*
to connect a callback to the signal

● rotate signal

NAME

"gesture-rotate-event"

CALLBACK PROTOTYPE

*gboolean gesture_rotate_cb(ClutterGesture *gesture, ClutterGestureRotateEvent *event, gpointer data)*

```
typedef struct {
    ClutterGestureEventType type;    //gesture event type, GESTURE_ROTATE
    guint32 time;                     //event time
    ClutterEventFlags flags;          //event flags
    ClutterStage *stage;              //event source stage
    ClutterActor *source;             //event source actor
    guint32 start_time;               //gesture start time
    gint x_start_1;                   //X coordinate of start point of first finger
    gint y_start_1;                   //Y coordinate of start point of first finger
    gint x_end_1;                     //X coordinate of end point of first finger
    gint y_end_1;                     //Y coordinate of end point of first finger
    gint x_start_2;                   //X coordinate of start point of second finger
    gint y_start_2;                   //Y coordinate of start point of second finger
    gint x_end_2;                     //X coordinate of end point of second finger
    gint y_end_2;                     //Y coordinate of end point of second finger
} ClutterGestureRotateEvent;
```

DESCRIPTION

This signal will be emitted when a rotate gesture is recognized. An application may use *g_signal_connect (gesture, "gesture-rotate-event", G_CALLBACK(your_rotate_cb), your_data)*
to connect a callback to the signal

● navigate signal

NAME

"gesture-navigate-event"

CALLBACK PROTOTYPE

*gboolean gesture_navigate_cb(ClutterGesture *gesture, ClutterGestureNavigateEvent *event, gpointer data)*

```
typedef struct {
  ClutterGestureEventType type;      //gesture event type, GESTURE_NAVIGATE
  guint32 time;                      //event time
  ClutterEventFlags flags;           //event flags
  ClutterStage *stage;               //event source stage
  ClutterActor *source;              //event source actor
  guint32 start_time;                //gesture start time
  gint x_start_1;                    //X coordinate of start point of first finger
  gint y_start_1;                    //Y coordinate of start point of first finger
  gint x_end_1;                      //X coordinate of end point of first finger
  gint y_end_1;                      //Y coordinate of end point of first finger
  gint x_start_2;                    //X coordinate of start point of second finger
  gint y_start_2;                    //Y coordinate of start point of second finger
  gint x_end_2;                      //X coordinate of end point of second finger
  gint y_end_2;                      //Y coordinate of end point of second finger
} ClutterGestureNavigateEvent;
```

DESCRIPTION

This signal will be emitted when a navigate gesture is recognized. An application may use

g_signal_connect (gesture, "gesture-window-event", G_CALLBACK(your_window_cb), your_data)

to connect a callback to the signal

● Catch-all signal

NAME

"gesture-any-event"

CALLBACK PROTOTYPE

*gboolean gesture_any_cb(ClutterGesture *gesture, ClutterGestureEvent *event, gpointer data)*

```
typedef struct {
    ClutterGestureEventType type;   //gesture event type,
```

```
    guint32 time;                  //event time
    ClutterEventFlags flags;       //event flags
    ClutterStage *stage;           //event source stage
    ClutterActor *source;          //event source actor
} ClutterGestureEvent;
```

DESCRIPTION

This signal will be emitted when any gesture is recognized. An application may use *g_signal_connect (gesture, "gesture-any-event", G_CALLBACK(your_event_cb), your_data)*
to connect a callback to the signal

In the callback, the actual event could be retrieved like:

```
switch (event->type) {
    case GESTURE_PINCH:
        ClutterGesturePinchEvent *pinch_event = (ClutterGesturePinchEvent
        *)event;
        …..
        break;
    case GESTURE_SLIDE:
        ClutterGestureSlideEvent *slide_event = (ClutterGestureSlideEvent
        *)event;
        …..
        break;

    case GESTURE_ROTATE:
        ClutterGestureRotateEvent *rotate_event = (ClutterGestureRotateEvent
        *)event;
        …..
        break;

    case GESTURE_NAVIGATE:
        ClutterGestureNavigateEvent *navigate_event =
        (ClutterGestureNavigateEvent *)event;
        …..
        break;

}
```

Work Flow

1. Create an instance

*ClutterGesture *gesture = NULL;*
*gesture = clutter_gesture_new (ClutterActor *actor);*

2. Configure the instance
   *clutter_gesture_set_gesture_mask (gesture, ClutterActor *actor_A, GESTURE_MASK_PINCH | GESTURE_MASK_SLIDE);*

3. Connect callbacks for gesture signals.
   g_signal_connect (gesture, "gesture-pinch-event", G_CALLBACK(gesture_pinch_cb), your_data);
   g_signal_connect (gesture, "gesture-slide-event", G_CALLBACK(gesture_slide_cb), your_data);