Easy Leap Motion Gestures and Starter Kit V1.0

Support: carlos.fernandez.musoles@gmail.com

Introduction

If you are reading this you probably don't need me to tell you how awesome and promising Leap Motion is. From Virtual Reality applications to fun casual games, the possibilities are endless. Now, it truly comes down to developers' imagination!

Easy Leap Motion Gestures (ELMG for now on) is a plugin that process the input received from the device and lets you detect a collection of gestures far beyond the predefined ones. It is as easy as start listening for gestures, and then the plugin will send you notifications whenever they happen, together with some data to make them more useful (things like position, state, and so on).

The plugin comes also with a Starter Kit scene (*StarterKitDemo* scene) that includes a simple example of how to control a 3D object solely with Leap Motion: rotate, move and scale using just your hand.

The second scene (ELMGdemo scene) gives an example of how to control the plugin. Give it a try and study it carefully to better understand the plugin.

Setting up

First thing you must do after importing the package is to copy the Leap Motion libraries to your Unity project. For convenience I have included the libraries for Mac 32 and 64 bits, but please do make sure you install the proper ones –more information **here.**

To start using the plugin you just need to register a listener for the gestures, switch the ones you want on and that's it! The demo scene that comes with the package shows you how easy this is:

- Register listener for all gestures (one listener for all)

```
ELGManager.GestureRecognised += onGestureRecognised;
```

- Activate the gesture

```
ELGManager.circleGestureRegistered = true;
ELGManager.keytapGestureRegistered = true;
ELGManager.screentapGestureRegistered = true;
…
```

Your listener method will need to have a *EasyLeapGesture* object as a parameter, which contains data about the gesture that the plugin is informing you about:

- *Type*: EasyLeapGestureType enum that describes the kind of gesture
- *State*: EasyLeapGestureState enum that describes the state of the gesture (STATEINVALID, STATESTART, STATEUPDATE, STATESTOP)
- *Id*: unique int that identifies the gesture
- *Duration*: in miliseconds
- *Frame*: Leap.Frame object that contained the gesture
- *Position:* Where in the detection zone the gesture happened (3D space)

```
void onGestureRecognised(EasyLeapGesture gesture) {
            // Do what you like with the gesture
    }
```

Using the plugin

As we stated above, to start using gestures you need to register an individual listener for all gestures AND activate each one. This dual process gives further control to the developer, since you can **register** all gestures when you start the app, but then **switch on only** the ones your app needs at any given time. This is the recommended approach to **avoid noise data** (and to reduce computation). Do not unregister and re-register the listener during gameplay as this is far less efficient than activating and deactivating them (setting the gesture property Boolean to false/true).

For instance (as shown in the demo scene), you would register your gesture listener on Start() function:

```
void Start () {
        ELGManager.GestureRecognised += onGestureRecognised;

}
```

Then, it may not make sense for your app to be listening to all of them in the menus. Perhaps you only want ScreenTap and KeyTap, so when your menus are loaded, you would activate only those:

```
void ActivateGesturesForMenus () {

        ELGManager.circleGestureRegistered = false;
        ELGManager.keytapGestureRegistered = true;
        ELGManager.screentapGestureRegistered = true;
        ELGManager.swipeGestureRegistered = false;
        ELGManager.numbersGestureRegistered = false;
        ELGManager.closeFistRegistered = false;
        ELGManager.openFistRegistered = false;
        ELGManager.pushGestureRegistered = false;
        ELGManager.pullGestureRegistered = false;
        ELGManager.doubleInwardsSwipeGestureRegistered = false;
        ELGManager.doubleOutwardsSwipeGestureRegistered = false;
        ELGManager.clapGestureRegistered = false;
        ELGManager.twoFingerKeytapRegistered = false;
        ELGManager.threeFingerKeytapRegistered = false;
        ELGManager.twoFingerScreentapRegistered = false;
        ELGManager.threeFingerScreentapRegistered = false;
        ELGManager.steeringWheelRegistered = false;
}
```

Once you load up the first level, perhaps those gestures are not relevant to your game play, and you only want Circle,, DoubleInwardsSwipe and Clap, hence:

```
void ActivateGesturesForLevel () {

        ELGManager.circleGestureRegistered = true;
        ELGManager.keytapGestureRegistered = false;
        ELGManager.screentapGestureRegistered = false;
        ELGManager.swipeGestureRegistered = false;
        ELGManager.numbersGestureRegistered = false;
        ELGManager.closeFistRegistered = false;
        ELGManager.openFistRegistered = false;
        ELGManager.pushGestureRegistered = false;
        ELGManager.pullGestureRegistered = false;
        ELGManager.doubleInwardsSwipeGestureRegistered = true;
        ELGManager.doubleOutwardsSwipeGestureRegistered = false;
        ELGManager.clapGestureRegistered = true;
        ELGManager.twoFingerKeytapRegistered = false;
        ELGManager.threeFingerKeytapRegistered = false;
        ELGManager.twoFingerScreentapRegistered = false;
        ELGManager.threeFingerScreentapRegistered = false;
        ELGManager.steeringWheelRegistered = false;
}
```

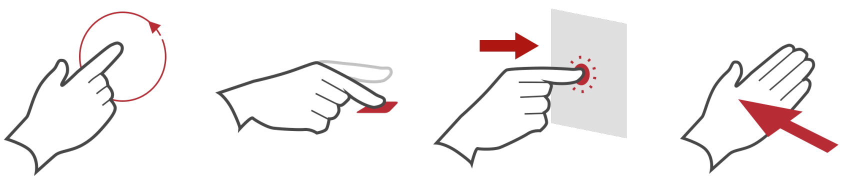
Gestures

ELMG builds up on the number of gestures recognized by the Leap Motion library to offer up to 17 different types of gestures (and 26 unique gestures).

Some of the parameters can be adjusted by modifying the static values on EasyLeapGesture struct. This is the complete list, but we'll mention in the gesture description what they affect.

```
public struct EasyLeapGesture {
        …
        // configurable settings
        public static float MaxAngleFinger = 45f; // max angle to consider a pointable
a finger
        public static float MinDistanceFinger = 25f; // min distance to consider a
pointable a finger (away from hand)
        public static float MaxAnglePalm = 25f; // max angle to consider a hand
horizontal
        public static float MaxFieldPalm = 180f; // max x and z to read palm pos
        public static float PullRecoveryTime = 0.2f; // min time in between pull
gestures
        public static float PushRecoveryTime = 0.2f; // min time in between push
gestures
        public static float MinPushPullVelocity = 350f; // min velocity to recognise
push pull gestures
        public static float DoubleInwardsRecoveryTime = 0.2f; // min time in between
double inwards swipe gestures
        public static float DoubleOutwardsRecoveryTime = 0.2f; // min time in between
double outwards swipe gestures
        public static float MinSwipeVelocity = 350f; // min velocity to recognise
double swipe gestures
        public static float MaxPalmDistance = 120f; // max distance between palms to
consider together -double swipe
        public static float MinClapVelocity = 350f; // min velocity to recognise a clap
gesture
        public static float MaxPalmClapDistance = 90f; // max distance between palms to
consider together -clap
        public static float ClapRecoveryTime = 0.15f; // min time in between claps
}
```

Default gestures



To activate, set ELGManager.circleGestureRegistered, ELGManager.keytapGestureRegistered, ELGManager.screentapGestureRegistered, ELGManager.swipeGestureRegistered to true

These are the Circle, KeyTap, ScreenTap and Swipe.

Two and three fingers KeyTap

To activate, set ELGManager.twoFingerKeytapRegistered / ELGManager.threeFingerKeytapRegistered to true

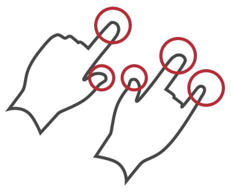
These two gestures are identical to KeyTap, but using 2 or 3 fingers.

Two and three fingers ScreenTap

To activate, set ELGManager.twoFingerScreentapRegistered / ELGManager.threeFingerScreentapRegistered to true

These two gestures are identical to ScreenTap, but using 2 or 3 fingers.

Numbers



To activate, set ELGManager.numbersGestureRegistered to true

It counts the number of extended fingers in the detection zone –up to 10. You can form a 4 with two hands (1+3 or 2+2) or with any 4 fingers on a single hand.

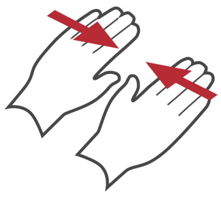
Note that there might be some noise in the sensor and sometimes you may get spikes of, say, ONE when trying to detect a TWO. This is quite natural and difficult to avoid, so we have introduced a duration parameter in the numbers gesture so you can be sure the gesture is not a one off random spike, but a purposeful gesture shown by the user (eg. You can use 0.5s as a threshold, or any other that fits your application).

While the gesture is detected, your listener will get a gesture per frame (including duration).

EasyLeapGesture relevant parameters:

- MaxAngleFinger: angle between your palm and an extended finger. If the angle is less than the value set here, the finger is not considered extended (and won't count to the total)
- MinDistanceFinger: distance between the finger tip and the palm position to consider a pointable a finger (sometimes you get noise near the palm).

Double inwards swipe



To activate, set ELGManager.doubleInwardSwipeGestureRegistered to true.

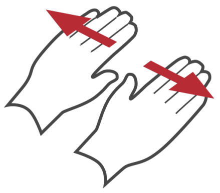
The gesture is detected once, at the end of the swiping movement (when both hands get together). Palms need to be parallel to the floor.

EasyLeapGesture relevant parameters:

- DoubleInwardsRecoveryTime: Min time between double inwards swipe gestures (to avoid getting a single gesture repeatedly)
- MinSwipeVelocity: Min velocity your hands have to have in order to consider a double swipe gesture
- MaxPalmDistance: Max distance between palms to consider both hands together

- MaxAnglePalm: Max angle to consider a hand parallel to the floor

Double outwards swipe

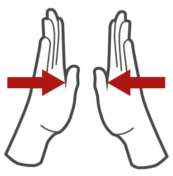


To activate, set ELGManager.doubleOutwardSwipeGestureRegistered to true.

The gesture is detected once, at the end of the swiping movement (when both hands get separated). Palms need to be parallel to the floor.

EasyLeapGesture relevant parameters:
- DoubleOutwardsRecoveryTime: Min time between double outwards swipe gestures (to avoid getting a single gesture repeatedly)
- MinSwipeVelocity: Min velocity your hands have to have in order to consider a double swipe gesture
- MaxPalmDistance: Max distance between palms to consider both hands together
- MaxAnglePalm: Max angle to consider a hand parallel to the floor

Clap

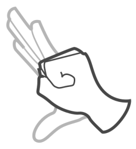


To activate, set ELGManager.clapGestureRegistered to true.

The gesture is detected once, when both hands get together.

EasyLeapGesture relevant parameters:
- ClapRecoveryTime: Min time between clap gestures (to avoid getting a single gesture repeatedly)
- MinClapVelocity: Min velocity your hands have to have in order to consider a clap gesture
- MaxPalmClapDistance: Max distance between palms to consider both hands together

Close fist

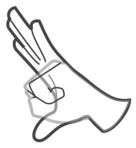


To activate, set ELGManager.closeFistRegistered to true.

The gesture is recorded over time:

- When is first detected (open hand parallel to the floor), the plugin will notify your listener of the gesture, assigning a State of STATESTART.
- From there until the gesture is finished (or aborted) your listener will get a CLOSE_FIST gesture with State STATEUPDATE per frame (including duration)
- Once the gesture is finished (close fist with hand parallel to the floor) or cancelled (hand moves out of the detection zone) your listener will get a CLOSE_FIST gesture with State STATESTOP.

Open fist



To activate, set ELGManager.openFistRegistered to true.

The gesture is recorded over time:

- When is first detected (close fist with hand parallel to the floor)), the plugin will notify your listener of the gesture, assigning a State of STATESTART.
- From there until the gesture is finished (or aborted) your listener will get a OPEN_FIST gesture with State STATEUPDATE per frame (including duration)
- Once the gesture is finished (open hand parallel to the floor or cancelled (hand moves out of the detection zone) your listener will get a OPEN_FIST gesture with State STATESTOP.

Push

To activate, set ELGManager.pushGestureRegistered to true.

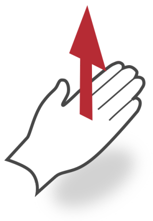
The gesture is detected once, when your hand reaches the necessary downwards velocity (palm needs to be parallel to the ground).

EasyLeapGesture relevant parameters:

- PushRecoveryTime: Min time between push gestures (to avoid getting a single gesture repeatedly)
- MaxAnglePalm: Max angle to consider a hand parallel to the floor

- MinPushPullVelocity: Min velocity to recognize push / pull gestures

Pull

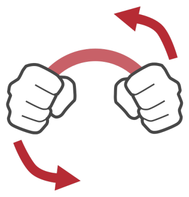


To activate, set ELGManager.pullGestureRegistered to true.

The gesture is detected once, when your hand reaches the necessary upwards velocity (palm needs to be parallel to the ground).

EasyLeapGesture relevant parameters:
- PullRecoveryTime: Min time between pull gestures (to avoid getting a single gesture repeatedly)
- MaxAnglePalm: Max angle to consider a hand parallel to the floor
- MinPushPullVelocity: Min velocity to recognize push / pull gestures

Steering Wheel (beta)



This gesture is still being polished, but it can be sufficient for some users.

To activate, set ELGManager.steeringWheelRegistered to true.

The gesture is detected whenever the user has two closed fists in the scene. Technically it does not matter whether the user is holding a virtual steering wheel, but the gesture will work fine when he/she does. The EasyLeapGesture passed on to the listener has the following properties:
- Type: EasyLeapGestureType.STEERING_WHEEL
- State: always EasyLeapGestureState.STATESTOP
- Duration: 0
- Position: the x and y properties give the steering angle*, whereas the z property indicates the distance in depth** (z coordinate) in between hands -this could be useful for some games.

*the steering angle is given in degrees (not radians). A positive angle indicates left hand is higher than right hand (right steering), whereas a negative angle indicates left hand is lower (left steering).

**z depth is still experimental and may return odd values

More gestures?

If you feel your game needs a particular gesture that is not yet implemented, you need only ask! Send me a note to carlos.fernandez.musoles@gmail.com and I'll be more than happy to try to add it to the ELMG collection.