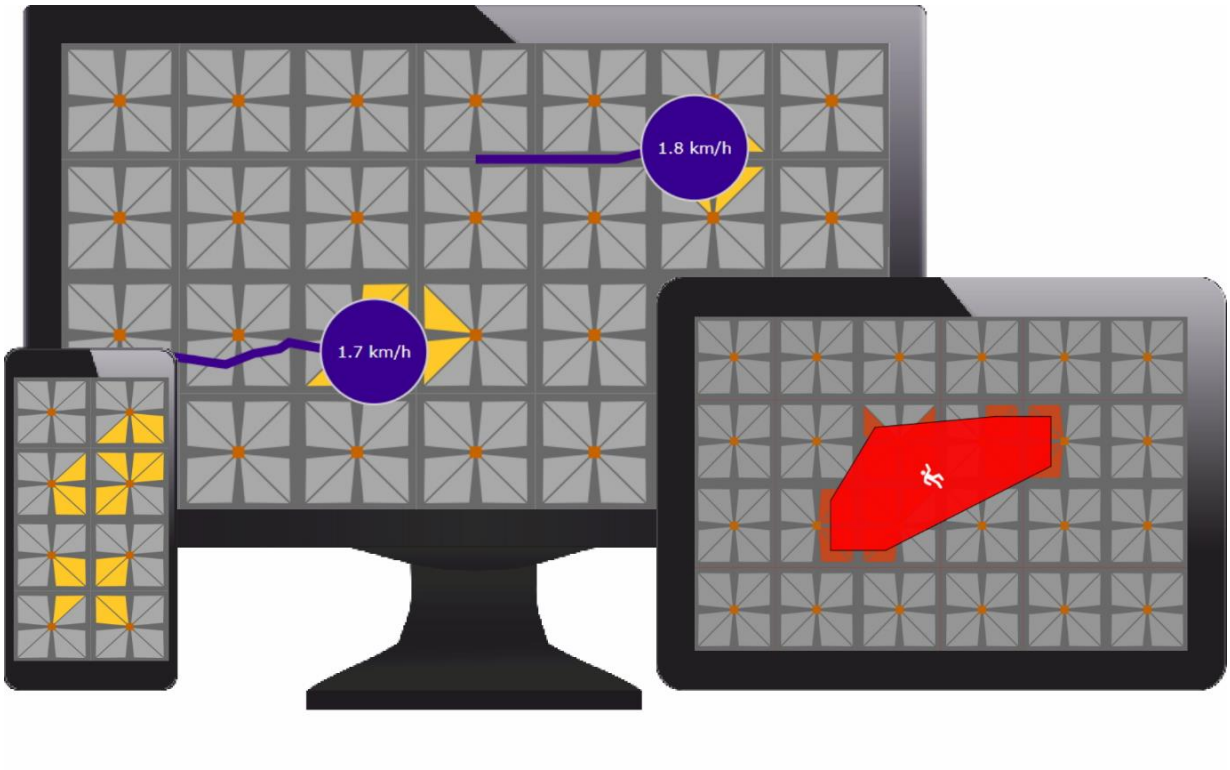
SensFloor® by FUTURE SHAPE



User Manual

SensFloor® Demo Kit

Future-Shape GmbH . Altlaufstr. 34 . D-85635 Hoehenkirchen-Siegertsbrunn
Phone +49 8102 89638-0, Fax: +49 8102 89638-99, sales@future-shape.com, www.future-shape.com

Contents

SensFloor® Demo Kit

SensFloor® is a sensitive floor that measures proximity, much in the same way as a smartphone touchscreen, albeit in a much larger area. It is based on a textile underlay, with embedded microelectronics that wirelessly send information about the state of each sensor field to a central receiver.

The SensFloor® Demo Kit allows you to better understand how SensFloor® works, and to explore different features already built in the smart receiver, the SE10.

You can choose to use the SensFloor® Web App out of the box to display your SensFloor® installation or build your own application by taking advantage of the SensFloor® API.

What's inside

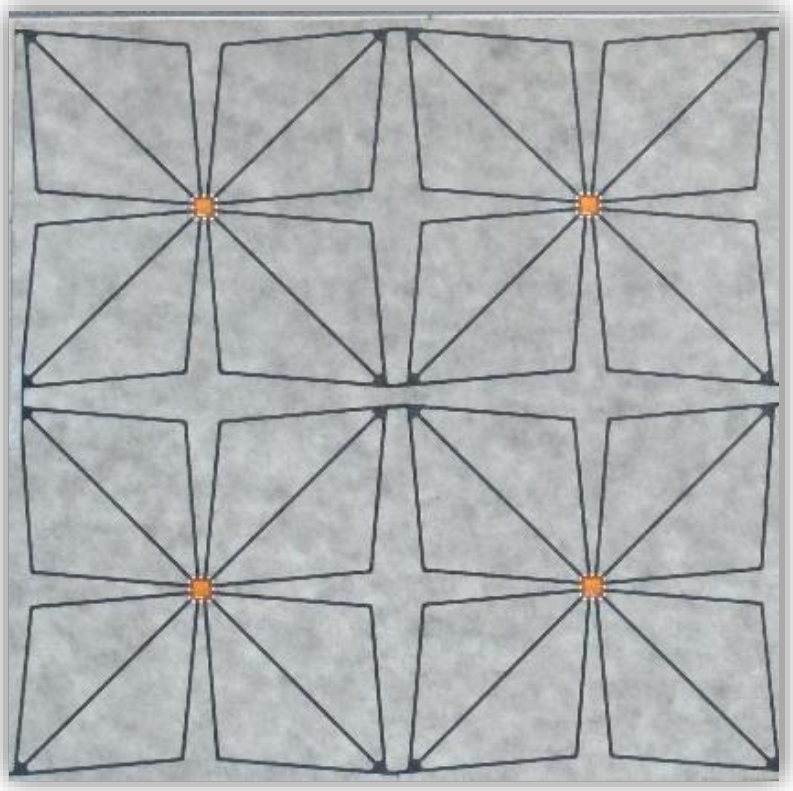


Figure 1: SensFloor® underlay.

Figure 4: SE10 Transceiver.

Figure 3: SE10 power adapter.

Figure 2: SensFloor® power adapter.

Setup

The SensFloor® Demo Kit allows you to use your own devices (i.e. smartphone, tablet, laptop) to access SensFloor® data. To do so, simply connect to the SE10 via Wi-Fi or LAN, open your browser of choice and navigate to the SensFloor® Web App. See below for more details.

Wi-Fi

The SE10 Transceiver generates a Wi-Fi Access Point that you can connect to directly. Such as with a home router, simply connect to this network, using the SSID and Password found on the label.

Figure 5: SE10 label

After successfully connecting to the SE10's network, you can access the SensFloor® Web App by opening a web browser (i.e. Google Chrome) and typing the following URL in the address field:

http://192.168.5.5:8000

Ethernet

If you wish to access the SensFloor® Web App through your own local network, you can do so by connecting an Ethernet LAN cable to the SE10.

You now need to find out which IP address was assigned to the SE10. You can do this either by checking the IP address table entries on your Router/Switch, or by opening a terminal window directly on the SE10. For the latter, connect a monitor, and a USB mouse and keyboard to the SE10. Now press Alt + F4 on your keyboard to exit the full screen browser view and open a terminal window by navigating to the top left menu of the Raspbian OS or by pressing CTRL+ALT+T on your keyboard. Once open, type "ifconfig eth0" and note down the IP address, which is found on the inet addr field (example output: "inet addr: 192.168.100.105").

Now that you know the IP address of the SE10 on your local network, you can access the SensFloor® Web App by opening a web browser (i.e. Google Chrome) and typing this IP and Port in the URL address field:

http://192.168.100.105:8000

HDMI

In addition to network access, the SensFloor® Web App can be visualized directly with an external monitor connected to the SE10's HDMI port.

Notes:

- Please note that the SensFloor® Web App can only be reached either via Wi-Fi or LAN at one time. That is, if you connect to the SE10's Wi-Fi but plug in an Ethernet LAN cable, you won't be able to access the app via Wi-Fi at http://192.168.5.5:8000. The LAN connection takes precedence.
- When changing from Wi-Fi to LAN or vice-versa, please restart the SE10 by disconnecting and reconnecting the power cable.
- If using the HDMI output port to connect to an external monitor, insert the HDMI cable before the power cable.
- After powering up the SE10, please allow up to one minute for the SensFloor® Web App to start.

SensFloor® Web App

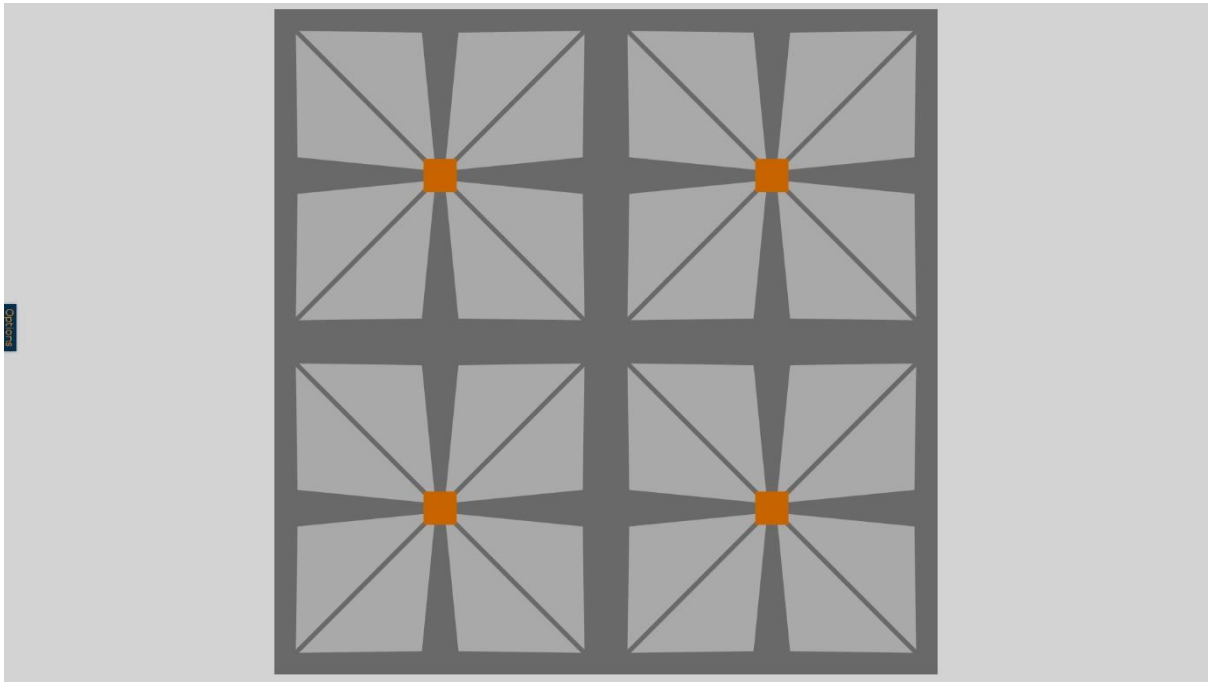


Figure 6: SensFloor® Web App.

The SensFloor® Web App shows you in real time how SensFloor® can be used to track persons and detect falls. Through interacting with the visualization options menu, you can intuitively see how, by progressively processing and abstracting the raw sensor data, higher level functions such as centroid extraction, clustering, objects (persons) detection, objects tracking, step detection, step counting, and fall detection can be achieved.

You can use any of these abstraction levels to build your own application upon, by using the SensFloor® API (see next chapter).

To toggle the visualization options menu, click on the “Options” button on the left side of the screen. Additionally, on touch-enabled devices, you can swipe on the left side of the page from-left-to-right or from-right-to-left, to show and hide the menu respectively.

Tracking

Here you find the options for visualizing the various tracking abstraction levels. The main goal of this function is to track persons standing or walking on top of SensFloor®.

SensFloor

This option displays the SensFloor® ground plan. You may want to hide it if, for instance, you just wish to display the objects over an empty background.

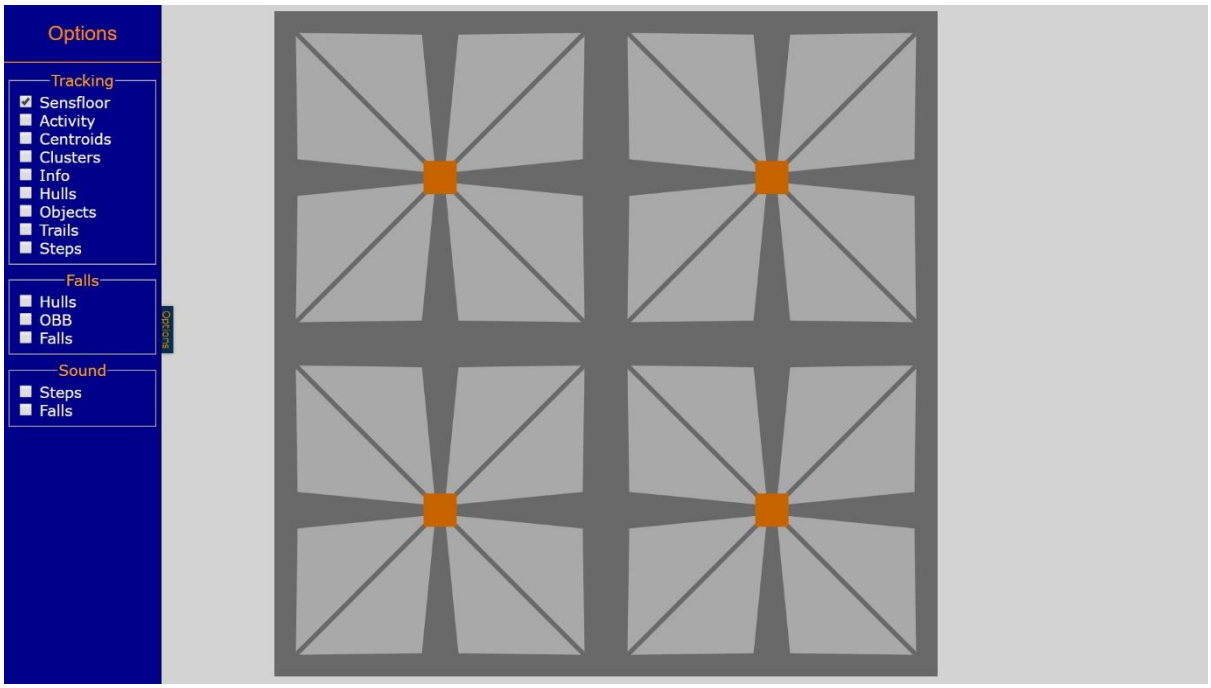


Figure 7: Options menu.

Activity

This is the option that directly shows the raw SensFloor® data for your installation. When enabled, the application draws the current capacitance for each SensFloor® patch sensor field (triangular shapes). The color changes from yellow (less capacitance) to red (more capacitance).

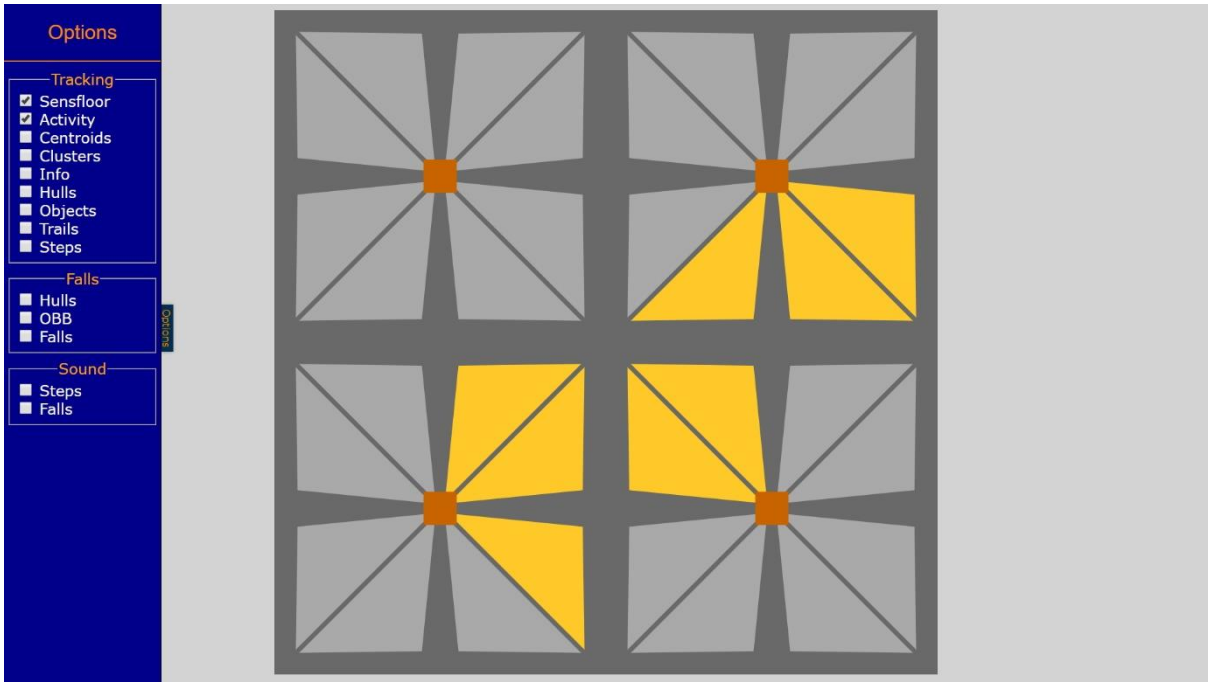


Figure 8: SensFloor® activity displayed as colored sensor fields. These shift from yellow to red as the capacitance increases.

Centroids

This option displays the center of mass for each active triangular-shaped sensor field. The more capacitance a field has, the more visible the center of mass is. This is used to abstract the geometry from the sensor fields onto a single (x, y) point, which is facilitates later data processing.

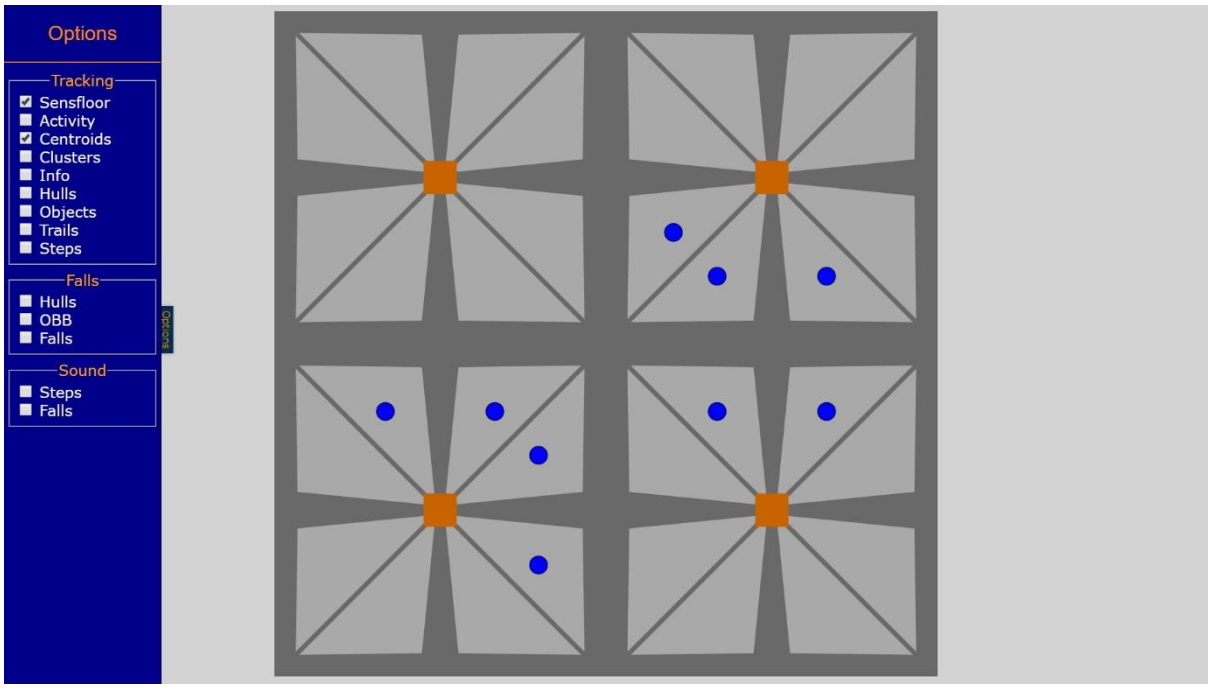


Figure 9: Centroid points of sensor fields. The more capacitance is present, the more visible the centroids are.

Clusters

The clusters represent groups of close-by centroids that became active (sensor fields above a certain capacitance) at roughly the same time. A cluster can represent, for instance, a foot of a person walking on the floor. This is, however, not guaranteed to always be the case, depending on floor resolution, ground plan configuration, and foot/shoe size.

Info

This shows information about each cluster. Namely: age – lifetime of the cluster in seconds; size – number of centroids grouped; weight – average capacitance of the centroids [140 – 255]; area – size of the cluster in square meters; spread – standard deviation of the centroids.

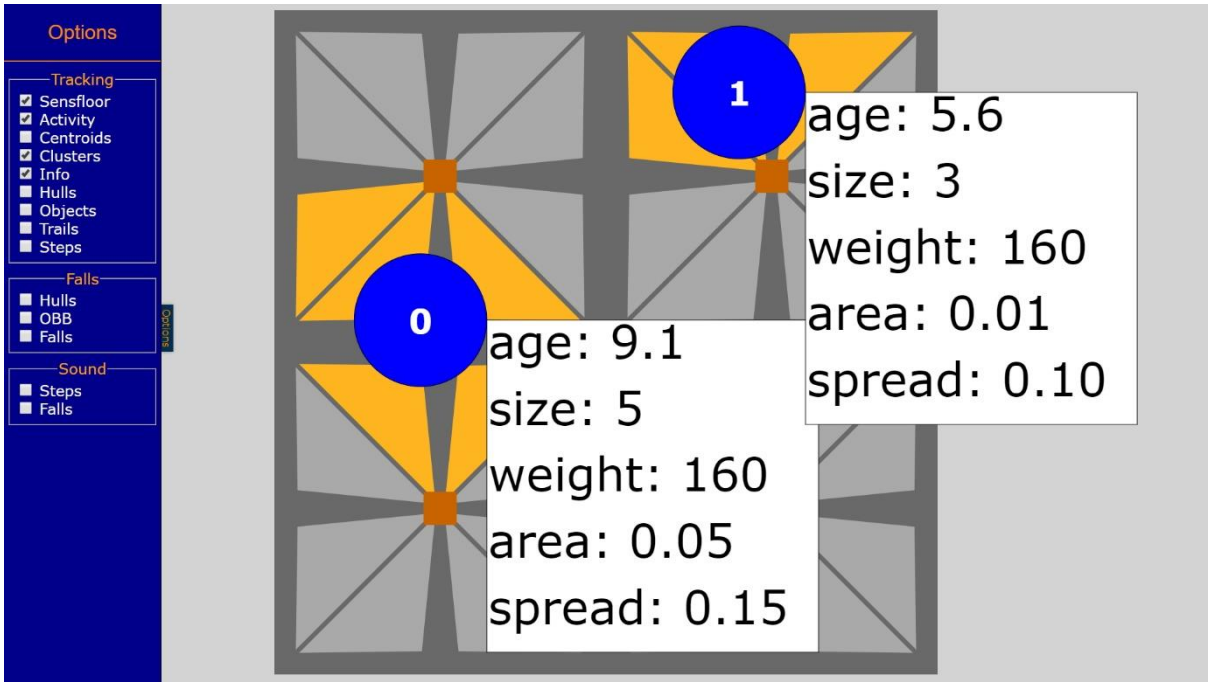


Figure 10: Clusters are formed from groups of active points.

Hulls

Shows the convex hull of the cluster, by connecting the “outer-most” centroids from the group and drawing a polygon.

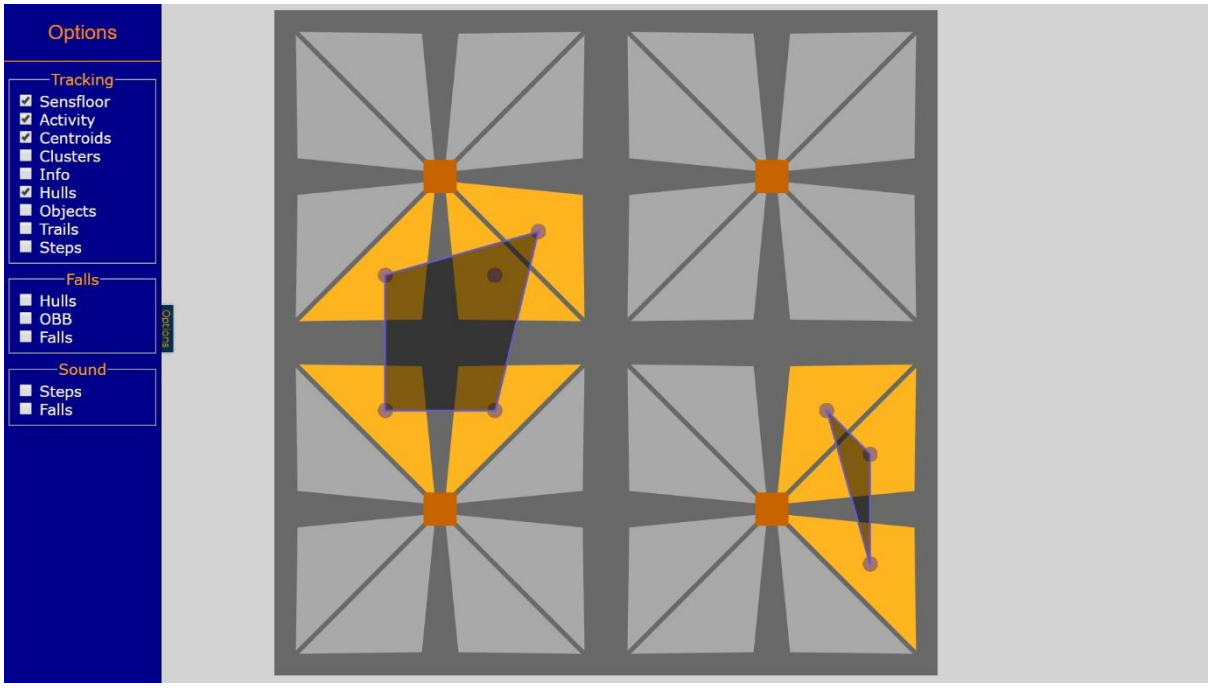


Figure 11: Convex hulls convey how large each cluster is.

Objects

Enabling this option displays the objects (persons) detected on top of SensFloor®. Each object is classified by grouping nearby clusters together. At times, one object can split into two, when the system thinks that there are two persons standing/walking close together.

Additionally, displayed on each object, is its speed in km/h. The color of an object is light blue when static (0 km/h) and shifts from dark blue to red as the speed increases.

Trails

With the trails enabled, each object leaves behind a trail which represents its trajectory in the last two seconds. You can use this to easily visualize how the system tracks individual objects.

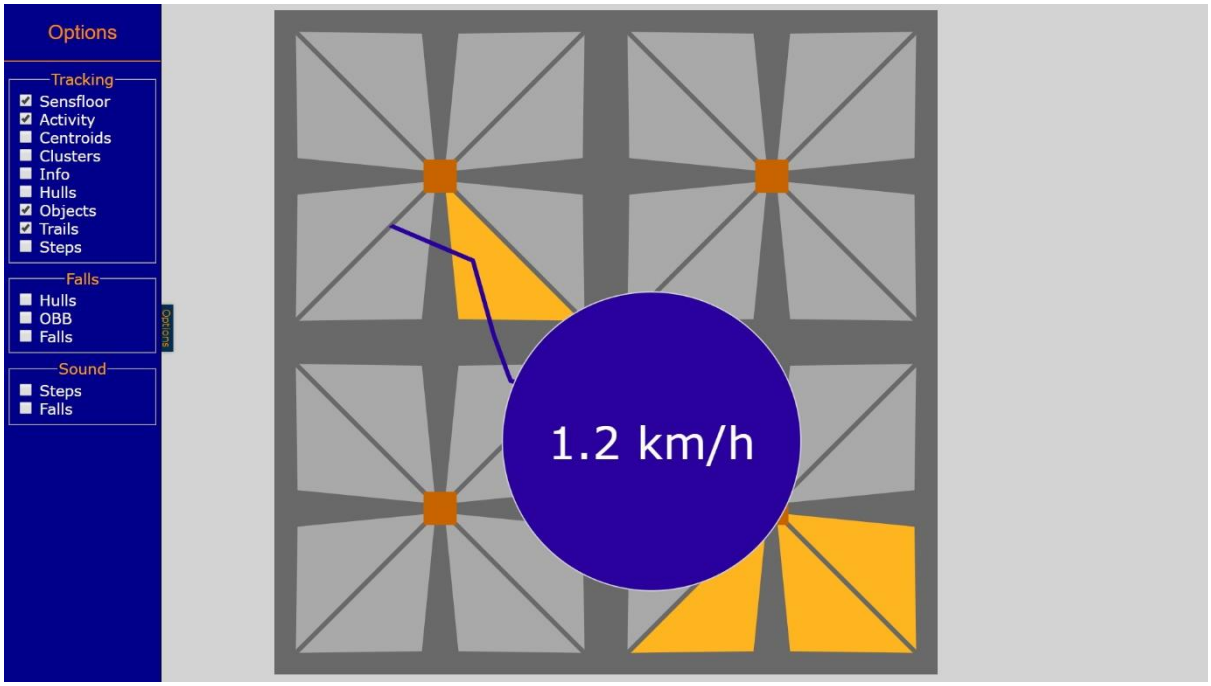


Figure 12: Objects are formed from clusters and display multiple persons standing or walking on SensFloor®.

Steps

The steps option shows a short outwards-expanding and dissipating wave where a step was detected on the floor. Note that steps detected near an object are associated with the latter. Step detection performance varies depending on the floor resolution as well as the persons gait and speed.

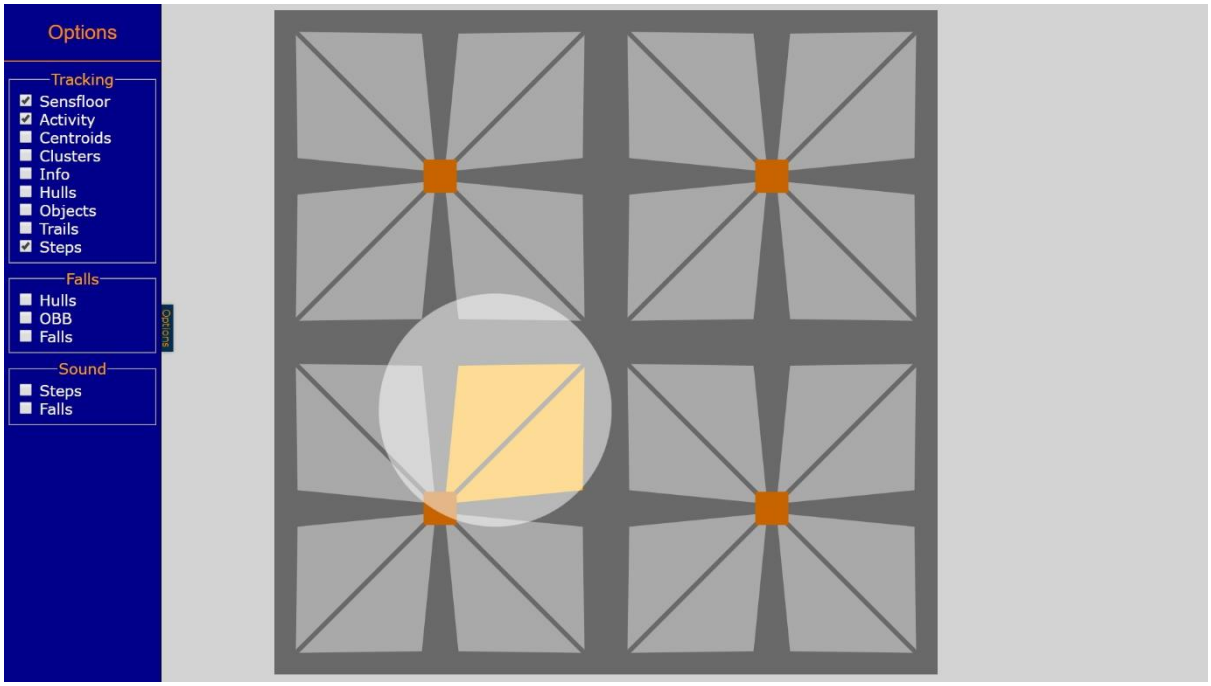


Figure 13: Detected steps quickly fade away from the detection point.

Fall detection

This sub-menu contains options for displaying falls detected on SensFloor®.

Hulls

Similarly to the hulls on the tracking options, the falls hulls display the convex hull as part of the fall detection algorithm. This part of the algorithm groups active nearby centroids together to be analyzed further. To note that, even when a fall is not considered active, these hulls will still be formed by groups of centroids as "possible falls".

OBB

This option (Oriented Bounding Box) draws a green box around the hulls, to make it easier to extract information about the possible fall orientation, location, shape, etc.

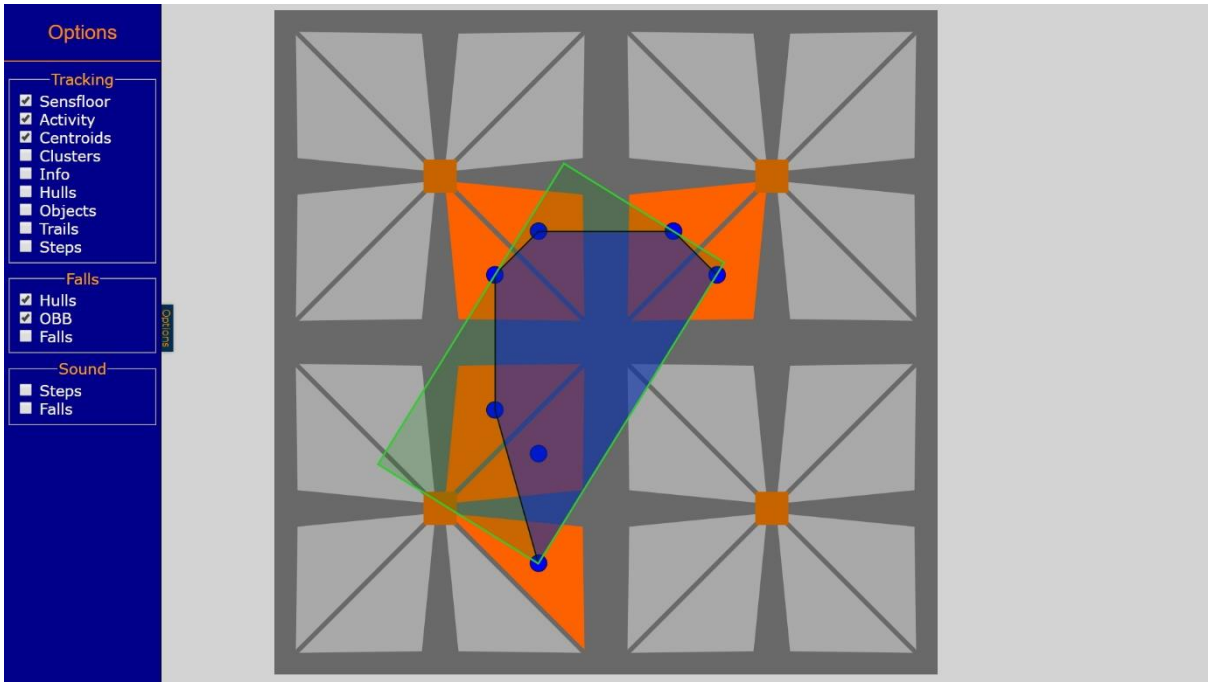


Figure 14: Convex hulls and oriented bounding boxes help visualize the how fall detection algorithm groups points to extract falls.

Falls

This is the main fall visualization option. When checked, and if a fall is detected, it will omit objects and clusters, and show a blinking red hull of where the fall is detected, as well as a red background. When a fall is not detected anymore, the objects will reappear after around two seconds.

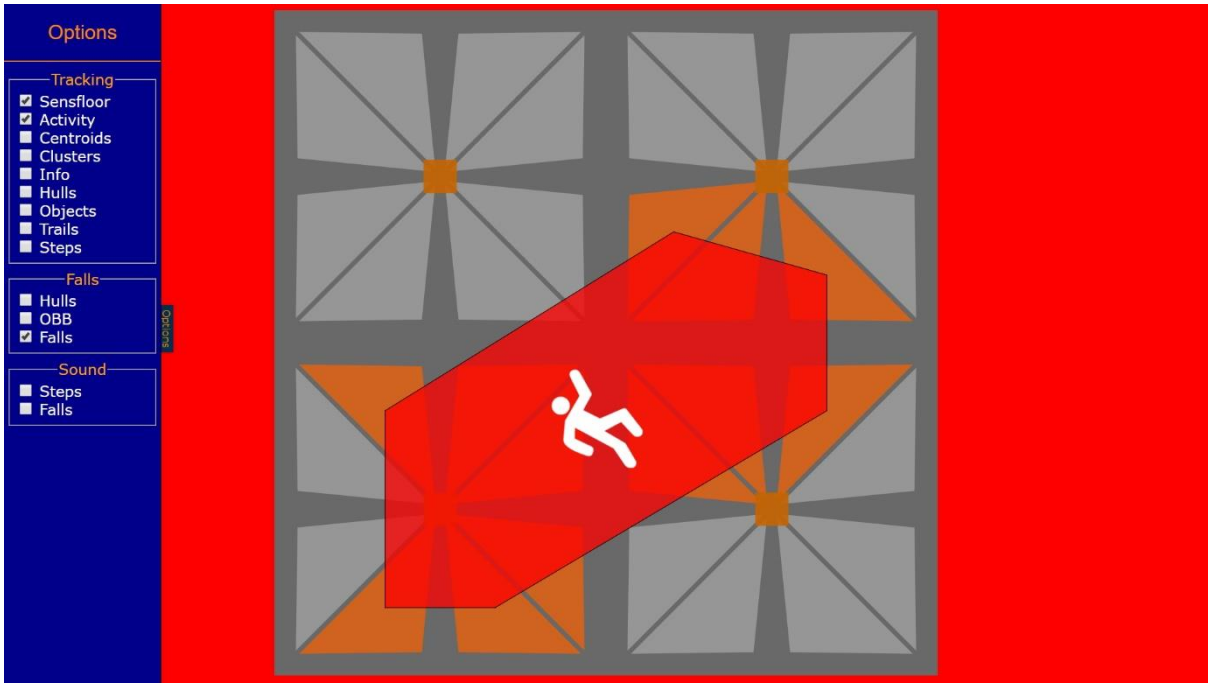


Figure 15: When a fall is detected, a blinking red polygon encompasses the area, and the whole background turns red.

Sound

You can enable sounds for two events detected on SensFloor®.

Steps

When enabled, up to six different short sounds will be played each time a step is detected. Steps associated with different persons (objects) will have different sounds.

Falls

When fall sounds are enabled and a fall is active, a warning sound will be continuously played.

SensFloor® API

The SensFloor® API allows you to build your own application by taking advantage of the different levels of pre-processed data it provides. For instance, you may wish to know if there's any activity on the floor at all; or count the number of persons in the area; or take an action when a fall is detected.

The API is reachable via web sockets, namely via socket.io (see https://socket.io/).

Socket.io clients exist for Node.js and for the browser in JavaScript. Furthermore, other implementations in other languages are available (see https://github.com/socketio/socket.io).

Data from the API are provided in the form of events. While some events are triggered asynchronously (ASYNC) when a SensFloor® message is received by the SE10, the majority is generated synchronously, roughly around every 100ms (SYNC).

How to use

To use to the API, you need to know the URL of the SE10. In case you connected via Wi-Fi, the URL shall be: http://192.168.5.5:8000. If you connected via Ethernet, the URL is dependent on your IP, for instance: http://192.168.100.105:8000. See the *Setup* section for more details.

Using JavaScript on the browser, you can connect to the API and get data as follows:

```
<script src="http://192.168.5.5:8000/socket.io/socket.io.js"></script>
<script>
	var client = io.connect('http://192.168.5.5:8000');
	client.on('connect', function() {
		console.log('connected');
	});

	client.on('messages-raw', function(messages) {
		for (var i = 0; i < messages.length; i++) {
			console.log(messages[i]);
		}
	});
</script>
```

Events

There are several events that can be accessed through the SensFloor® API, depending on the abstraction level you need to build your own application. As stated above, these events are either triggered by SensFloor® messages (ASYNC) or artificially triggered every 100ms by the system (SYNC).

To use data from each event, simply register a listener for the event name and retrieve the data (See the *Examples* sub-section for more details).

‘messages-raw’

Description

This event sends out every SensFloor® message received, without any processing. It pre-appends a timestamp to the array, in milliseconds since the epoch.

Type

ASYNC

Data

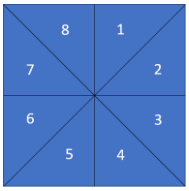
Array of time-stamped full SensFloor messages.

Format

[
    [TIME, BYTE0, BYTE1, ..., BYTE15],
    [TIME, BYTE0, BYTE1, ..., BYTE15],
    ...
]

TIME	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
TIME	GID0	GID1	X	Y	DC	DC	DC	DC	FIELD1	FIELD2	FIELD3	FIELD4	FIELD5	FIELD6	FIELD7	FIELD8

Time: Milliseconds since the epoch
GID1,0: Two bytes comprising the Group ID of the installation
X, Y: Position of the SensFloor® patch
DC: Don’t care
FIELD1…8: “Capacitance” value for each sensor field, between 128 and 255



Data example

```
[ [1476180138377, 3, 65, 7, 2, 179, 0, 17, 2, 175, 142, 128, 127, 128, 127, 132, 130] ]
```

‘new-activity-on-fields’

Description

This event sends out messages every time at least one of the sensor fields has a new capacitance value. It can be used to update the capacitance of individual fields in your application, instead of the whole patch at one time.

Type

ASYNC

Data

Array of time-stamped new capacitance values per field.

Format

[
    [TIME, GID, X, Y, FIELD, CAP],
    [TIME, GID, X, Y, FIELD, CAP],
    ...
]

Data example

```
[ [1476189894888, 833, 6, 3, 1, 129] ]
```

‘step’

Description

This event is triggered every time a new step is detected on SensFloor® and provides information about the location of the step. Step detection performance is dependent on gait and floor resolution.

Type

ASYNC

Data

Two numbers containing the location of the step.

Format

x, y

Data example

```
0.398, 0.667
```

‘activity-update’

Description

This event sends out information about which points (centroids of the sensor fields) changed capacitance and state roughly in the last 100 milliseconds.

A point (field) changing state means that its capacitance crossed a certain threshold (set to 140 by default). That is, a point can either be active or inactive.

Type

SYNC

Data

An object containing information on which points changed state or capacitance value since the last sampling period.

Format

```
{
        flags: {
                newCapacitance,
                newState

              },
        newCapacitance,
        newState

}
```

.flags.newCapacitance [boolean] - true if at least one point has a new capacitance value.
.flags.newState [boolean] - true if at least one point has a new state (ON/OFF = Active/Inactive).

.newCapacitance [array] - points that have a new capacitance value.

point {

.**id**: number of the point

.**capacitance**: value of the field capacitance [127 – 255]

.**x**: world position on the x axis, in meters

.**y**: world position on the y axis, in meters

}

.newState [array] - points that have a new state.

point {

.**id**: number of the point

.**isActive**: true if the point is active (True/False)

.**birthTime**: When the point became active, in milliseconds since epoch

.**x**: world position on the x axis, in meters

.**y**: world position on the y axis, in meters

}

Data examples

- Point 4 becomes active

```
activity: {
    flags: {
        newCapacitance: true,
        newState: true
    },
    newCapacitance: [ {id: 4, capacitance: 150, x: 0.167, y: 0.102} ],
    newState: [ {id: 4, isActive: true, birthTime: 1476372959279, x: 0.167, y: 0.102} ]

}
```

- Point 12 has a new capacitance value but did not change state

```
activity: {
    flags: {
        newCapacitance: true,
        newState: false
    },
    newCapacitance: [ {id: 12, capacitance: 176, x: 0.667, y: 0.102} ],
    newState: []
}
```

‘clusters-update’

Description

This event provides information about clusters of active points (centroids of sensor fields). Each cluster is comprised of at least one point and has a unique ID assigned to it. In simple terms, active points close to each other, that became active within a short period of time, form a cluster. Conversely, two points that are either far away from each other, or became active at very distinct times, will form two separate clusters.

Type
SYNC

Data
An array containing the current clusters.

Format
[cluster, cluster, ...]

```
cluster: {
      id,
      birthTime,
      x,
      y,
      xWeighted,
      yWeighted,
      age,
      weight,
      size,
      hull,
      area,
      spread
}
```

.id [number] – cluster id.
.birthTime [number] - when the cluster was created, in milliseconds since epoch.
.**x** [number] – cluster world position on the x axis, in meters.
.**y** [number] – cluster world position on the y axis, in meters.
.**xWeigthed** [number] - capacitance-weighted position on the x axis, in meters.
.**yWeigthed** [number] - capacitance-weighted position on the y axis, in meters.
.**age** [number] – time since the cluster was created, in milliseconds.
.**weight** [number] - average capacitance value of the cluster's points.
.**size** [number] - amount of points belonging to the cluster.
.**hull** [array] - array of vertices (x, y) that define the convex hull of the cluster.
.**area** [number] - area of the convex hull, in square meters.
.**spread** [number] - cluster points' standard deviation, in meters.

Data example
Array with two clusters

```
[ { id: 0,
    birthTime: 1479827359404,
    x: 0.7310000000000001,
    y: 0.7310000000000001,
    xWeighted: 0.7310000000000001,
    yWeighted: 0.7310000000000001,
    averageBirthTime: 1479827359460,
    age: 11337,
    weight: 150,
    size: 2,
```

```
    hull: [ [Object], [Object] ],
    area: 0,
    spread: 0.132 },
{   id: 1,
    birthTime: 1479827365235,
    x: 2.5989999999999998,
    y: 1.247,
    xWeighted: 2.5989999999999998,
    yWeighted: 1.247,
    averageBirthTime: 1479827365266,
    age: 5506,
    weight: 150,
    size: 3,
    hull: [ [Object], [Object], [Object] ],
    area: 0.02177999999999991,
    spread: 0.2498719672152121 } ]
```


'objects-update'

Description

The objects event provides information about the objects (normally associated to persons) that are detected on SensFloor®. Objects are born from groups of clusters. The location of an object is estimated from the current location of its clusters and a simple movement prediction system. Furthermore, each object keeps track of how many steps were detected in its vicinity.

Type

SYNC

Data

An array containing the current objects.

Data Format

[object, object, ...]

```
object: {
      id,
      birthTime,
      birthX,
      birthY,
      x,
      y,
      age,
      mV,
      aV,
      steps
}
```

.id [number] – object id.
.birthTime [number] - when the object was created, in milliseconds since epoch.
.**birthX** [number] – world position of where the object was created on the x axis, in meters.
.**birthX** [number] – world position of where the object was created on the y axis, in meters.
.**x** [number] – object world position on the x axis, in meters.

.**y** [number] – object world position on the y axis, in meters.
.**age** [number] – time since the object was created, in milliseconds.
.**mV** [number] – magnitude component of the velocity vector for the object, in meters per second.
.**aV** [number] – angle component of the velocity vector for the object, in degrees.
.**steps** [number] – number of steps detected near the object.

Data
An array containing the current objects.

Data example
Array with one object

```
[ { id: 0,
    birthTime: 1479828662153,
    birthX: 0.7310000000000001,
    birthY: 0.7310000000000001,
    x: 1.124036889454593,
    y: 1.2421012866769736,
    age: 2100,
    mV: 0.3223749077759499,
    aV: 52.439715720681185,
    steps: 2 } ]
```

'falls-detected'

Description
This event contains information about falls detected on SensFloor®. It provides details about the fall, such as its location, the polygon that circumscribes the area of the fall, and an Oriented Bounding Box (OBB) for that polygon.

Type
SYNC

Data
Two arrays containing the current falls. The first array contains information about the bounding boxes. The second about the convex hulls.

Data Format
[OBBInfo, OBBInfo, …], [hullsInfo, hullsInfo, …]

```
OBBInfo: {
      points,
      centroid
}
```

.points [array] – the four vertices defining the bounding box for the fall.
      point[array]: world position of the vertex, in meters [x, y].
.centroid [array] – world position of the box centroid, in meters [x, y].

```
hullsInfo: [points, centroid]
```

points [array] – the polygon vertices defining the convex hull of the fall.
point[array]: world position of the vertex, in meters [x, y].
centroid [array] – world position of the convex hull centroid, in meters [x, y].

Data example
One fall detected.

```
[ { points: [ [Object], [Object], [Object], [Object] ],
    centroid: [ 0.605, 0.434 ]
} ]

[ [ [ [Object], [Object], [Object], [Object], [Object] ],
    [ 0.589, 0.484 ] ]
]
```

Examples

This section provides you with three examples of how the SensFloor® API can be access through JavaScript in the browser.

To see each example, create an empty HTML file, paste the code you see below, and save the file (e.g. Example1.html). Afterwards, open the file you just created in your browser of choice and you will be able to see different information when you walk or fall on top of SensFloor®. Make sure you are connected to the SE10, either via Wi-Fi or via LAN. For the latter case, change the two IP addresses on each example to match the SE10's IP address on your local network.

Example 1 – Printing raw data

```
<!DOCTYPE html>
<html>
    <head>
        <title>SensFloor - Example 1 - Raw Data</title>
        <script src="http://192.168.5.5:8000/socket.io/socket.io.js"></script>
    </head>
    <body>
        <textarea id=logger></textarea>
        <script>
            var client = io.connect('http://192.168.5.5:8000');

            client.on('connect', function() {
                console.log('connected');
            });

            client.on('messages-raw', function(messages) {
                for (var i = 0; i < messages.length; i++) {
                    var date = new Date(messages[i][0]);
                    var timeStamp = date.getFullYear() + '-' +
                        ('0' + (date.getMonth() + 1)).slice(-2) + '-' +
                        ('0' + date.getDate()).slice(-2) + '-' +
                        ('0' + date.getHours()).slice(-2) + ':' +
                        ('0' + date.getMinutes()).slice(-2) + ':' +
                        ('0' + date.getSeconds()).slice(-2) + ':' +
                        ('00' + date.getMilliseconds()).slice(-3);

                    var messageString = timeStamp;
                    for (b = 1; b <= 16; b++) {
                        var hexByte = ('0' + (messages[i][b] &
                         0xFF).toString(16)).slice(-2).toUpperCase();
                        messageString += ' ' + hexByte;
                    }
                    var logger = document.getElementById('logger');
                    logger.innerHTML += messageString + '&#010;';
                    logger.scrollTop = logger.scrollHeight;
                }
```

```
            });
        </script>
        <style>
            body {
                background-color: lightgrey;
            }
            #logger {
                width: 90%;
                height: 90%;
                top: 5%;
                left: 5%;
                position: absolute;
                resize: none;
                font-size: 2.25vw;
            }
        </style>
    </body>
</html>
```

Example 2 – Tracking persons and speed

```
<!DOCTYPE html>
<html>
    <head>
        <title>SensFloor - Example 2 - Tracking persons and speed</title>
        <script src="http://192.168.5.5:8000/socket.io/socket.io.js"></script>
    </head>
    <body>
        <div id=number-persons></div>
        <div id=static-persons></div>
        <div id=moving-persons></div>
        <div id=max-speed></div>
        <div id=speed-record></div>
        <script>
            var client = io.connect('http://192.168.5.5:8000');
            var speedRecord = 0;

            client.on('connect', function() {
                console.log('connected');
            });

            client.on('objects-update', function(objects) {
                var nPersons = nStanding = nWalking = maxSpeed = 0;
                for (var i = 0; i < objects.length; i++) {
                    nPersons++;
                    if (objects[i].mV > 0) {
                        nWalking++
                    }
                    else {
```

```
                    nStanding++;
                }
                if (objects[i].mV > maxSpeed) {
                    maxSpeed = objects[i].mV;
                }
                if (objects[i].mV > speedRecord) {
                    speedRecord = objects[i].mV;
                }
            }
            document.getElementById('number-persons').innerHTML = nPersons + ' Person(s)';
            document.getElementById('static-persons').innerHTML = nStanding + ' Standing';
            document.getElementById('moving-persons').innerHTML = nWalking +  ' Walking';
            document.getElementById('max-speed').innerHTML = 'Max Speed: ' +
            (maxSpeed * 3.6).toFixed(1) + ' km/h';
            document.getElementById('speed-record').innerHTML = 'Speed Record: ' +
            (speedRecord * 3.6).toFixed(1) + ' km/h';
        })
    </script>
    <style>
        div {
            width: 40%;
            height: 20%;
            position: absolute;
            font-size: 4vw;
            border: 1px solid black;
            display: flex;
            align-items: center;
            justify-content: center;
            background-color: lightgrey;
        }
        #number-persons {
            top: 5%;
            left: 5%;
            width: 90%;
            height: 25%;
            font-size: 10vw;
        }
        #static-persons {
            left: 5%;
            top: 40%;
        }
        #moving-persons {
            left: 55%;
            top: 40%;
        }
        #max-speed {
            left: 5%;
            top: 70%;
```

```
                color: blue;
            }
            #speed-record {
                left: 55%;
                top: 70%;
                color: red;
            }
        </style>
    </body>
</html>
```

Example 3 – Detecting falls

```
<!DOCTYPE html>
<html>
    <head>
        <title>SensFloor - Example 3 - Detecting falls</title>
        <script src="http://192.168.5.5:8000/socket.io/socket.io.js"></script>
    </head>
    <body>
        <div id=current-fall></div>
        <div id=last-fall>No previous falls detected</div>
        <script>
            var client = io.connect('http://192.168.5.5:8000');
            var lastFallDate = '';
            var lastFallTime = '';

            client.on('connect', function() {
                console.log('connected');
            });

            client.on('falls-detected', function(fallsOBBs, fallsHulls) {
                if (fallsOBBs.length > 0) {
                    var date = new Date();
                    var fallDate =  date.getFullYear() + '-' +
                                    ('0' + (date.getMonth() + 1)).slice(-2) + '-' +
                                    ('0' + date.getDate()).slice(-2);

                    var fallTime =  ('0' + date.getHours()).slice(-2) + ':' +
                                    ('0' + date.getMinutes()).slice(-2) + ':' +
                                    ('0' + date.getSeconds()).slice(-2);
                    document.getElementById('current-fall').innerHTML = 'FALL DETECTED !';
                    document.getElementById('last-fall').innerHTML = 'Last fall detected on ' +
                    fallDate + ' at ' + fallTime;
                    document.getElementById('current-fall').style.color = "red";
                    document.getElementById('last-fall').style.color = "red";
                    document.body.style.background = 'red';
                }
                else {
```

```
                    document.getElementById('current-fall').innerHTML = 'No falls';
                    document.getElementById('current-fall').style.color = "black";
                    document.body.style.background = 'none';
                }
            });
        </script>
        <style>
            div {
                width: 90%;
                left: 5%;
                position: absolute;
                border: 1px solid black;
                display: flex;
                align-items: center;
                justify-content: center;
                background-color: lightgrey;
            }
            #current-fall {
                height: 50%;
                top: 5%;
                font-size: 10vw;
            }
            #last-fall {
                height: 25%;
                top: 70%;
                font-size: 4.5vw;
            }
        </style>
    </body>
</html>
```

F.A.Q.

Q1. I connected the SE10 to the monitor via HDMI, but nothing is displayed.

Please connect the HDMI cable before the power cable and allow up to one minute for the SensFloor® Web App to start.

Q2. How do I connect to the SE10 Wi-Fi ?

You can find the SSID and Password in the SE10's label. See the *Setup* section for more details.

Q3. I can see the SensFloor® Web App but nothing happens when I step on the floor.

Please make sure that SensFloor® is powered on. If still nothing is displayed, reboot the SE10 and try again.

Q4. How far away can the SE10 be from the SensFloor® ?

Messages can be received up to around 15 meters. This range can be higher or lower, depending on several environmental factors, such as the building layout and obstacles between the floor and the receiver.

Q5. I connected the SE10 to my LAN network and now can't use the SensFloor® Web App/API via Wi-Fi.

Please note that only LAN or Wi-Fi can be used at one time. See the *Setup* section for more details.

Q6. The SensFloor® Web App doesn't work well on my browser.

If you can't see the SensFloor® Web App being displayed properly, please try another browser. We recommend the latest Google Chrome.

Q7. How many persons can SensFloor® detect at one time ?

Around 2 persons per square meter.

Q8. I tried the HTML examples but don't see anything.

Please make sure you are connected to the SE10, either via Wi-Fi or LAN. In addition, make sure that the IP in the examples matches the SE10's IP (should be the case for Wi-Fi).