2017 年 9 月 10 日

Leap Motion and Web

https://developer.leapmotion.com/documentation/javascript/api/Leap_Classes.html

徐　丙鉄

LeapMotion コントローラ→Leap Service →【WebSocket サーバー】←Browser

↓

API

↓

Leap アプリ

Leap Motion コントローラから**トラッキングデータ**を取得するバックグラウンドで実行されるアプリを Leap Service と呼ぶ。この Leap Service は、モーショントラッキングデータをネイティブ API で提供するとともに，デフォルトで localhost(127.0.0.1)のポート 6437 で WebSocket サーバを立て，WebSocket を通じても提供する。

㈱インテリジェントテクノロジーの技術情報ブログ：http://iti.hatenablog.jp/entry/2013/11/28/172240

WebSocket サーバにアクセスすると，モーショントラッキングデータが JSON 形式で毎秒 100 フレームで返ってくる。一方，ブラウザは最高で 60fps で画面を更新するので，LeapJS コントローラはブラウザの animation loop に基づいた animation frame を生成する。

確認

Chrome アドオン Simple WebSocket Client で LeapMotion の WebSocket サーバにアクセス

ws://localhost:6437

Leap Motion コントローラーの frame objects (JSON)


```
{
    "currentFrameRate": 30.0015,
    "gestures": [],
    "hands": [],
    "id": 68820,
    "interactionBox": {
        "center": [0, 200, 0],
        "size": [221.418, 221.418, 154.742]
    },
    "pointables": [],
    "r": [
        [-0.197554, -0.902611, 0.382446],
        [0.0582653,  0.378634, 0.923711],
        [-0.978559,  0.204766,-0.0222098]
    ],
    "s": -4.74618,
    "t": [357.555, -373.131, 1857.62],
    "timestamp": 8613448358
}
```


Leap.js　　CDN：https://developer-archive.leapmotion.com/javascript

Leap Motion WebSocket サーバの JSON 出力を Leap Motion ネイティブ API と一貫性のある構造に変換する JavaScript API である。

サンプル

Leap Gallary JavaScript: https://gallery.leapmotion.com/category/javascript/

Controller

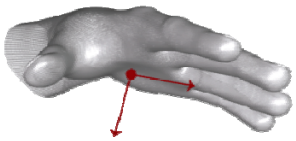
計測単位

Distance:	millimeters
Time:	microseconds (unless otherwise noted)
Speed:	millimeters/second
Angle:	radians

計測データ:The Frame object is essentially the root of the Leap Motion data model.

Frame

Hand: palmNormal, direction



Pointable

tipPosition, direction

Finger: tipPosition, direction
(Finger extends Pointable)

Gesture

```
var controller = Leap.loop({enableGestures: true}, function(frame){
  if(frame.valid && frame.gestures.length > 0){
    frame.gestures.forEach(function(gesture){
        switch (gesture.type){
          case "circle":
              console.log("Circle Gesture");
              break;
          case "keyTap":
              console.log("Key Tap Gesture");
              break;
          case "screenTap":
              console.log("Screen Tap Gesture");
              break;
          case "swipe":
              console.log("Swipe Gesture");
              break;
        }
    });
  }
});
```

loop 関数

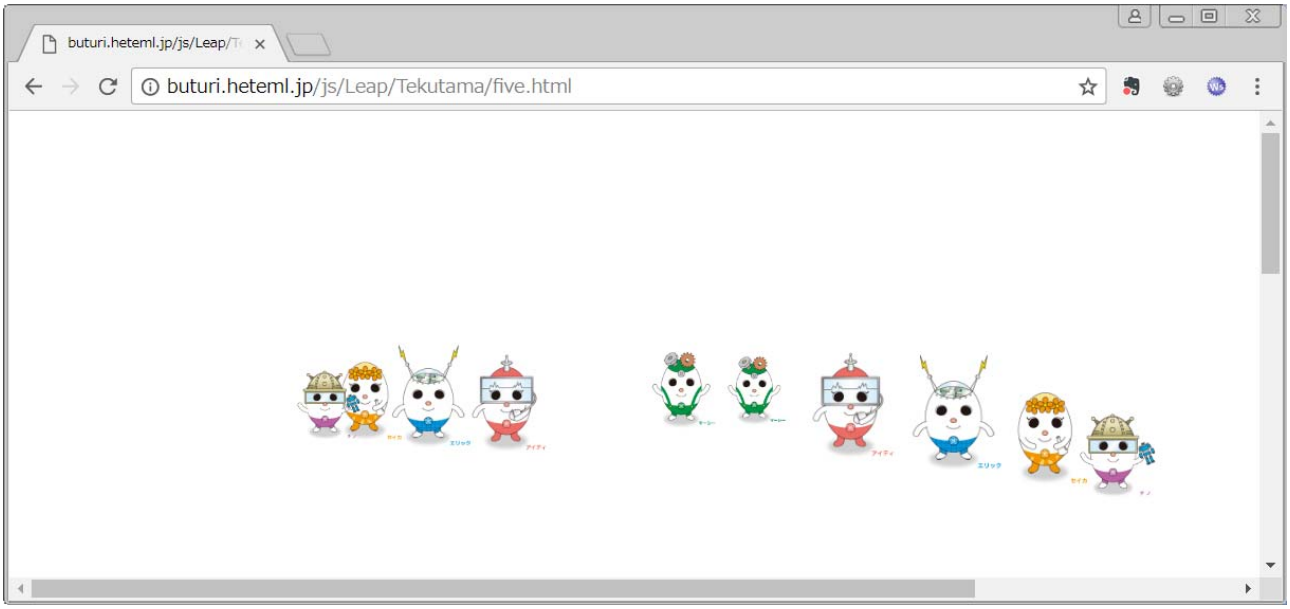
Controller　Leap.loop(options, callback)

The loop() function sets up the Leap controller and WebSocket connection and invokes the specified callback function on a regular update interval. You do not need to create your own controller when using this method. The most recent frame of Leap data is passed to your callback function.

frame: the controller supplying tracking data to this loop function

```
var controller = Leap.loop({ enableGestures:true },   function(frame){
      var currentFrame = frame;
      var previousFrame = controller.frame(1);
      var tenFramesBack = controller.frame(10);
});
```

Tekutama Fingers:　http://buturi.heteml.jp/js/Leap/Tekutama/five.html




```
<!DOCTYPE html>
<html>
<head>
<script src="https://js.leapmotion.com/leap-0.6.4.min.js"></script>
<script src="https://js.leapmotion.com/leap-plugins-0.1.12.min.js"></script>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script>
$(function(){
    var myCanvas = document.getElementById("myCanvas");
    var ctx = myCanvas.getContext("2d");
    var w =1280;            var h= 800;
    var imgW = 206;         var imgH = 240;
    var imageIds =   ["#kikai","#jyoho","#denshi","#kagaku","#robo","#kennchiku"];
    var imageIds2 = ["#kikai2","#jyoho2","#denshi2","#kagaku2","#robo2","#kennchiku2"];

    Leap.loop(function(frame){
            if(!frame.pointables.length) return;
            ctx.clearRect(0, 0, w, h);
            for(var hi = 0; hi < frame.hands.length; hi++){
                var hand = frame.hands[hi];
                var pointables = hand.pointables;
                    for( var pi=0; pi < pointables.length; pi++){
                        var pointable = pointables[pi];
                        var tip = pointable.tipPosition;
                        if(!tip) return;
                        showTekutamaAtTip(tip, hi, pi);
                    }
                }
    });

    function showTekutamaAtTip(tip, hi, pi){
                        // hi: hands[hi]左右の手を区別, pi: pointables[pi]指を区別
        var x = tip[0]*2 + w/2;       //tip[0] x 座標
```

```
        var y = -tip[1] + h/2;        // tip[1] y 座標：上下方向
        var zoom = tip[2];              zoom = 200 -zoom;           // tip[2] z 座標：前後
        //画像のサイズを z 座標（前後）で変更
        var W = 0.0015*imgW*zoom;        var H = 0.0015*imgH*zoom;
        if(hi == 0){
                $(imageIds[pi]).css({width: W, height: H});
                $(imageIds[pi]).css({left:(x-0.5*W),top:(y-0.5*H)});
        }else{
                $(imageIds2[pi]).css({width: W, height: H});
                $(imageIds2[pi]).css({left:(x-0.5*W),top:(y-0.5*H)});
        }
    }

});
</script>
<style>img{position: absolute;} </style>
</head>
<body>
<canvas id ="myCanvas" width="1200" height="800"></canvas>
    <img src="imgs/jyoho.gif" id="jyoho">
    <img src="imgs/denshi.gif" id="denshi">
    <img src="imgs/kagaku.gif" id="kagaku">
    <img src="imgs/robo.gif" id="robo">
    <img src="imgs/kenchiku.gif" id="kenchiku">
    <img src="imgs/kikai.gif" id="kikai">

    <img src="imgs/jyoho.gif" id="jyoho2">
    <img src="imgs/denshi.gif" id="denshi2">
    <img src="imgs/kagaku.gif" id="kagaku2">
    <img src="imgs/robo.gif" id="robo2">
    <img src="imgs/kenchiku.gif" id="kenchiku2">
    <img src="imgs/kikai.gif" id="kikai2">
</body>
</html>
```