KINECTED BROWSER

Version 1.0.0.6 – November, 2012 – release notes rev. B
For the latest version of this document, see http://go.microsoft.com/fwlink/?LinkId=272161.

INTRODUCTION

Kinected Browser allows Web page authors to create immersive interactive experiences inside the browser. The platform provides high and low level access to the Kinect device to enable a variety of designs.

The goal of Kinected Browser is to provide a platform for experimentation with Natural User Interface in the browser. This package is experimental quality and should not be used for production applications.

Features

1. Scriptable high level DOM events for joint activity such as leftHandOver, rightKneeMove, etc.
2. Access to full skeleton tracking
3. Support for multiple and custom methods for mapping physical space into the browser
4. Access to the depth stream with built in methods for drawing to HTML5 Canvas

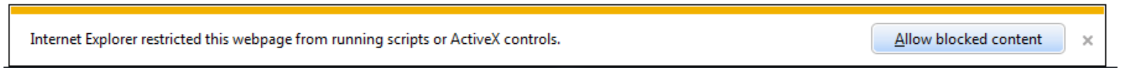
SYSTEM REQUIREMENTS

Kinected Browser works on PCs running Microsoft Windows 7 and 8, x86 and x64 platforms, using Internet Explorer 9 and 10. On x64 platforms, only x86 Internet Explorer is supported. For Windows 8, the browser must be in Desktop mode as the full-screen experience does not support ActiveX controls. Users must have a Kinect for Windows (K4W) device connected to their PC and the Kinect drivers version 1.6 or higher installed.

GETTING STARTED

Follow the steps below to get started developing with the toolkit.

1. Download and run the installer to copy the component and samples onto your computer and register the ActiveX object.
2. In the directory you chose, locate the sample file TestPage.html and open it with Internet Explorer.
3. Your Internet Explorer security settings may cause the “gold bar” below to appear. If it does, click *Allow blocked content* to continue.



Internet Explorer restricted this webpage from running scripts or ActiveX controls.

4. After a few moments, you should see be able to see the depth image and skeleton displayed in the large boxes. You may need to step away from your Kinect as the demo does not have near or seated modes enabled by default.

ARCHITECTURE

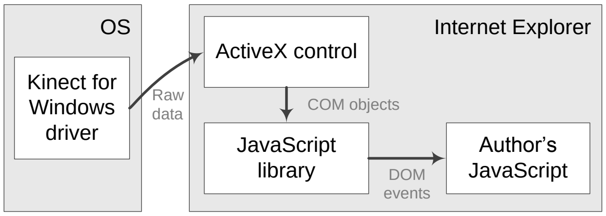


Figure 1. High level architectural diagram

Kinected Browser consists of two main components: an ActiveX control that acts as an intermediary between the K4W hardware and Internet Explorer, and a JavaScript library that processes events from the K4W hardware into JavaScript-friendly data and events. We hope that by providing most of the logic in the JavaScript layer, developers can alter the code and experiment with ways to make their Web applications even more amazing.

The ActiveX control is registered by the installer. The JavaScript library, NUIScript.js, is installed into the same location as the DLL.

KINECTED BROWSER API

INITIALIZATION

To use the Kinect in a Web page, include two script tags, one referencing jQuery, and one referencing NUIScript.js, found in the installation directory:

```
<script type="text/javascript" src="http://ajax.aspnetcdn.com/ajax/jquery/jquery- ↵
   1.8.1.min.js"></script>
<script type="text/javascript" src="NUIScript.js"></script>
```

In your custom script, use the `Kinect` class to work with the Kinect device. Before the system can respond to any events, it must be initialized. The initialization process adds an object tag that instantiates the ActiveX control to the document and calls the Kinect SDK's initialization routines. To initialize, call the `init` method on the `Kinect` class. Method `init` takes an optional dictionary of options described below:

`Kinect.init([`*`options`*`])`

`options` (optional) Dictionary of properties that configure Kinect-browser interaction.
You may specify any of the following key/value pairs. Keys and values are case sensitive.

Key	Default	Description
`skeletonCanvas`	`undefined`	HTML Canvas Element. Specifies the canvas to which the system draws skeletons.
`depthCanvas`	`undefined`	HTML Canvas Element. Specifies the canvas to which the system draws depth data.
`drawSkeleton`	`false`	Boolean. When `skeletonCanvas` is set and `drawSkeleton` is true, the system draws the skeletons to the specified canvas.
`drawDepth`	`false`	Boolean. When `depthCanvas` is set and `drawDepth` is

		true, the system draws the depth image to the specified canvas.
`mapper`	`"viewport"`	String. Either `"egocentric"` or `"viewport"`. The **viewport** mapper maps the Kinect's camera to the viewport of the browser through a constant linear transformation. The result is that skeleton coordinates on one side of the physical space will map to screen pixels on the corresponding portion of the browser viewport. The **egocentric** mapper maps all points it believes to be reachable by the first skeleton it finds. The linear transformation dynamically adjusts to the estimated reach of the player. The goal is to allow all pixels in the viewport to be reachable by the player. This is useful to allow players to reach any pixel without them having to move their feet (a lateral translation).
`depth`	`true`	Allow reading the depth image from the device
`color`	`false`	(reserved for future use) Allow reading the color image from the device
`skeletons`	`true`	Track skeleton positions in the frame
`audio`	`false`	Enable sound source localization
`nearmode`	`false`	Track skeletons in "near mode"
`seated`	`false`	Track skeletons in seated position

SHUTTING DOWN

The ActiveX control of Kinected Browser creates a few threads and maintains some state. Developers should shut down the Kinected Browser system when the page unloads, cleaning up threads and uninitializing the Kinect SDK. The `Kinect.shutdown` method performs this cleanup. Using jQuery, one can easily bind to the beforeunload event and call `Kinect.shutdown()`.

```
$(document).on("beforeunload", Kinect.shutdown);
```

Please note that it may take several seconds to shut down the system.

SKELETON DOM EVENTS

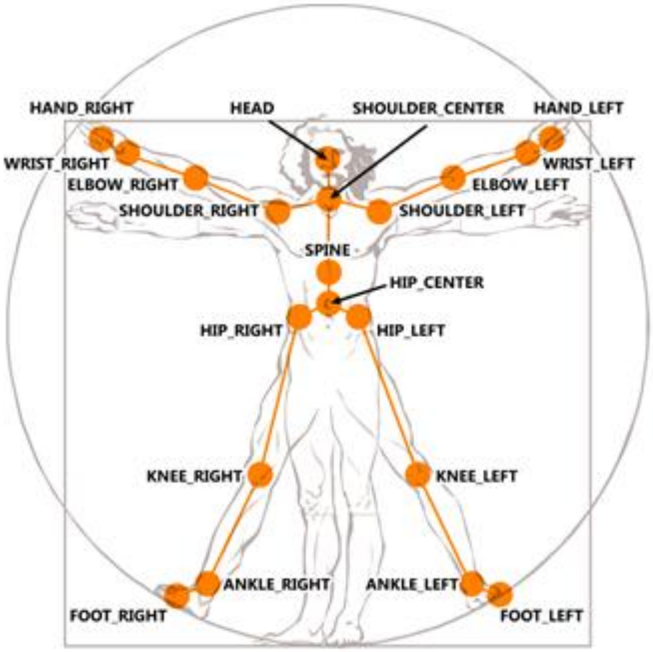
Kinected Browser events on DOM elements by dispatching through jQuery's event dispatch system. To have your JavaScript respond to Kinect events, you must do two steps:

1. Add the `kinect_enable` CSS class to any element that should receive over/out events. This class is required to reduce the set of elements that the system tracks when computing over/out event signaling.
2. Subscribe to the event using jQuery's *on()* method :
   $(*selector*).on("*eventName*.kinect", handler);

Currently, the package supports joint over, joint out, and joint move events. For each *joint*, the following three event names are defined: *joint*Over, *joint*Out, and *joint*Move. These function just like the existing mouseOver, mouseOut, and mouseMove events in modern browsers. Use the figure and table below and to determine the correct event names to subscribe.

In some cases, developers may wish to show interface elements that indicate where joints are. For example, two DIV elements can be absolutely positioned to show where the left and right hands are with respect to the page (this is done in the sample TestPage.html code). However, these elements should not receive events themselves. Tag such elements with the CSS class `kinect_ignore`.

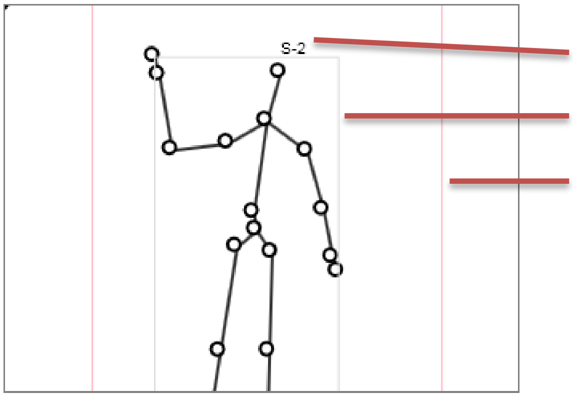
K4W SDK Constant	Kinected Browser event prefix
`HIP_CENTER`	`centerHip`
`SPINE`	`spine`
`SHOULDER_CENTER`	`centerShoulder`
`HEAD`	`head`
`SHOULDER_LEFT`	`leftShoulder`
`ELBOW_LEFT`	`leftElbow`
`WRIST_LEFT`	`leftWrist`
`HAND_LEFT`	`leftHand`
`SHOULDER_RIGHT`	`rightShoulder`
`ELBOW_RIGHT`	`rightElbow`
`WRIST_RIGHT`	`rightWrist`
`HAND_RIGHT`	`rightHand`
`HIP_LEFT`	`leftHip`
`KNEE_LEFT`	`leftKnee`
`ANKLE_LEFT`	`leftAnkle`
`FOOT_LEFT`	`leftFoot`
`HIP_RIGHT`	`rightHip`
`KNEE_RIGHT`	`rightKnee`
`ANKLE_RIGHT`	`rightAnkle`
`FOOT_RIGHT`	`rightFoot`



In addition to the individual joints, the system supports mapping multiple joints to a single virtual joint. This makes it simpler for developers to subscribe to events about a joint on either side of the body. For example, the *leftHand* and *rightHand* family of events also trigger a *hand* family of events; that is, *leftHandOver* and *rightHandOver* both trigger the *handOver* event if either hand moves over the target element and neither hand is currently over the target. This avoids firing the event twice.

DRAWING SKELETONS

For debugging, it can be helpful to see the locations of the tracked skeletons. The `drawSkeleton` and `skeletonCanvas` options passed to `Kinect.init()` combine to enable drawing skeletons to the canvas on each new frame. The diagram below shows a skeleton drawn to a canvas. In addition to the skeleton, the system draws two bounding boxes: one which bounds the skeleton itself and one which shows the “reachable” bounding box – given the length of the arms and position of the player, this shows where a player is likely able to reach.



Skeleton ID

Bounding box (grey box)

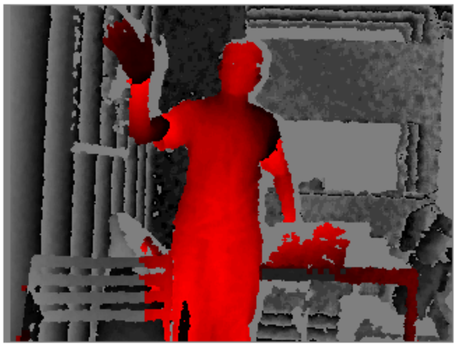
Reachable bounding box (pink box)

EVENT PROPERTIES

Event handlers receive a single object with properties about the event.

`offsetX`	(int) X-coordinate of relative position within the element
`offsetY`	(int) Y-coordinate of relative position within the element
`clientX`	(int) X-coordinate in the browser's viewport
`clientY`	(int) Y-coordinate in the browser's viewport
`depth`	(int) Distance in millimeters from the Kinect device to the joint
`srcElement`	(Object) Original element that triggered the event
`timeStamp`	(Date) time of the event
`player_id`	(int) player ID

DEPTH DATA



Kinected Browser supports access to the depth stream of the Kinect hardware. Developers can choose to use the depth image in two different ways. The system can draw the frames to an HTML5 Canvas specified by the developer by specifying the the `drawDepthCanvas` flag and `depthCanvas` property when initializing. This procedure is more efficient than calling the `getDepthData` method as the depth data is drawn to the canvas directly using native code. Pixels are drawn in a unique color per player whose brightness corresponds to the depth. Background depth pixels are drawn in grayscale.

For direct access to the data, use the `getDepthData` method. Using the depth stream will cause additional CPU load. Since copying over 153Kb of data multiple times per second across the native code-JavaScript boundary is costly, accessing the depth stream through JavaScript will be slow. Currently the system supplies the image bytes as elements in a VBArray, which wraps the underlying COM SAFEARRAY pointer (access to the SAFEARRAY type is not directly supported in Internet Explorer). In the future, we plan to add support for refilling a single typed array, now supported by Internet Explorer 10, as a substantial performance improvement.

```
Kinect.getDepthData()
```

Returns: VBArray object containing depth data. Call the `getItem` method on the returned VBArray instance to get the depth data for each pixel. The depth in millimeters is the left 13 bits of each value; the player id is the right three bits of each value. That is, for the depth at pixel (28, 105):

```
var ddata = Kinect.getDepthData();
var px = ddata.getItem(105 * 320 + 28);
var depth = px >>> 3;
var player = px & 7;
```

CAMERA ANGLE

You can control the tilt angle of the camera subject to restrictions of the underlying Kinect SDK. That is, the tilt angle cannot be changed too many times in a short time period, and the angle is restricted to a range of -27 to 27 degrees off of horizontal.

To get the camera angle, call:

```
Kinect.GetTiltAngle();
```

To set the camera angle, call:

```
Kinect.SetTiltAngle(deg);
```

Where *deg* is the absolute angle in degrees. Negative values tilt the sensor below the horizontal plane of its base; positive values tilt above.

NOTE: Currently you cannot call this method immediately after calling `Kinect.init`. Instead, you should create a callback that executes more than 250ms after calling `Kinect.init`. This can be accomplished using `setTimeout()`.

SOUND SOURCE LOCALIZATION

Kinected Browser supports the sound source localization of the Kinect device. To activate this function, pass the pair `"audio": true` to the options when calling `Kinect.init`. The included sample code SoundSourceLoc.html has a working example.

The source angle and confidence are available by polling the following methods. Refer to the C++ documentation for notes on precision, accuracy, and range information.

```
Kinect.Audio.GetSourceAngle()
```

Returns: the angle (in degrees) from which the device believes the sound emanates. The angle is relative to the z-axis, which is perpendicular to the Kinect sensor.

```
Kinect.Audio.GetSourceConfidence()
```

Returns: the confidence from 0.0 to 1.0; higher is better.

SAMPLES

The toolkit includes two samples:

TestPage.html, a simple test page that displays the skeleton tracking, depth canvas, and has a `DIV` element that responds to `handOver` and `handOut` events.

SouncdSourceLoc.html demonstrates how to use sound source localization.

FURTHER INFORMATION

Stay up to date with the latest developments at http://go.microsoft.com/fwlink/?LinkId=271616.

This work was originally published at the 2012 ACM Interactive Tabletop and Surface conference: Liebling, D.J., and Morris, M.R. Kinected Browser: Depth Camera Interaction for the Web. *Proceedings of the 2012 ACM Conference on Interactive Tabletops and Surfaces*. November, 2012.

We welcome your comments! Please send your feedback to kbfb@microsoft.com.

KNOWN ISSUES

1. Kinected Browser can only run in one tab and one process at a time. The Kinect for Windows SDK limits control of the device to a single process.
2. Including jQuery from a local path (on the local computer) instead of a remote URL can cause the ActiveX control to not initialize properly.
3. Certain audio devices may cause issues which block echo cancellation and sound source localization from functioning.
4. Polygonal imagemap areas are not currently supported. Developers can modify the code in NUIScript.js to add support for polygons by using a standard polygonal hit testing routine.
5. The depth canvas is only drawn when skeleton tracking is enabled and at least one skeleton is tracked. Developers can modify the code in NUIScript.js to call the `drawDepthCanvas` method manually.
6. In seated mode, skeletons drawn to the canvas show an extraneous line from the SHOULDER_CENTER joint to (0, 0).

Please feel free to send bug reports to kbfb@microsoft.com.

LICENSE

Microsoft

INDEX