Polling-based Joystick API

The current proposal for accessing Joysticks from Mozilla is an event-based one, mirroring keyboard and mouse input. https://wiki.mozilla.org/JoystickAPI

At 60fps, with all the of the analog buttons, triggers, vibration, gyroscope on modern game pad (PS3, Xbox 360, and others) there will easily be many events within each 16ms frame.

With so many events, it's both inefficient, as many events will only be handled in order to update the "current state", and complex to correctly maintain and update the state, which will have to be written in every application.

On the positive side, events and polling are not mutually exclusive. While it seems like the primary use-case for gamepad input is for games, which would prefer polling, there may be some uses where the event-based is better, say for using the gamepad as an input device to advance frames in a presentation.

Polling proposal sketch

navigator.joystick.getFeatures(index)

Returns Object containing information about details of hardware.

numAxes	integral Number >= 0
numButtons	integral Number >= 0
numTriggers	integral Number >= 0
numAccelerometers	integral Number >= 0

Note voice, vibration, and other extended features of controllers to be handled via cooperation with other future APIs. It would be nice to ensure that there's a trivial mapping between the index used for identification here, and in those other APIs.

The number of buttons, triggers, axes, and accelerometers depends on the connected hardware.

navigator.joystick.getState(index)

Returns null if there is no joystick currently connected at that index, otherwise an Object containing:

timestamp	monotonically increasing number to determine if hardware has been polled since previous getState() call
inputs	Array of Numbers indexed by constants defined below.

.inputs is indexed by the following constants:

2D Axes

These are typically thumbsticks or full hand joysticks. Axes are indexed by constants named
`navigator.joystick.XAXIS_0`
`navigator.joystick.YAXIS_0`
`navigator.joystick.XAXIS_1`
`navigator.joystick.YAXIS_1`
…
up to `XAXIS_0` + getFeatures(i).numAxes - 1.

Note that for all pairs of symbolic axis names, `XAXIS_0` + 1 = `YAXIS_0`, and `YAXIS_0` + 1 = `XAXIS_1`, etc.

All axis values should be normalized to [-1.0 .. 1.0]. On a standard joystick controller, left and up should be -1.0, and right and down 1.0.

Buttons

Indexed by constants named
`navigator.joystick.BUTTON_0`
`navigator.joystick.BUTTON_1`
…
up to `BUTTON_0` + getFeatures(i).numButtons - 1

Note that `BUTTON_0` + 1 == `BUTTON_1`, etc.

Button values should be normalized to the range [0.0 .. 1.0], where 0.0 means fully unpressed, and 1.0 means fully pressed.

Triggers

Indexed by constants named
`navigator.joystick.TRIGGER_0`
`navigator.joystick.TRIGGER_1`
…
up to `TRIGGER_0` + getFeatures(i).numTriggers - 1

Note that `TRIGGER_0` + 1 == `TRIGGER_1`, etc.

Trigger values should be normalized to the range [0.0 .. 1.0], where 0.0 means fully unpressed, and 1.0 means fully pressed.

Accelerometers

Indexed by constants named
`navigator.joystick.ACCELEROMETER_0`
`navigator.joystick.ACCELEROMETER_1`
…
up to `ACCELEROMETER_0` + getFeatures(i).numAccelerometers - 1

Note that `ACCELEROMETER_0` + 1 == `ACCELEROMETER_1`, etc.

Accelerometer values should be normalized and separated into individual axes as necessary. Each axis value should be normalized to the range [-1.0 .. 1.0].

Discussion

It is intended that these values are not interpreted, dead-zoned, etc. in any way and that interpretation is handled at a higher level. All values are returned as normalized floating point numbers even if the device only supports digital input.

Mapping of buttons and triggers is undefined, but buttons should appear in decreasing order of importance, so that the primary button is element 0, the secondary button is element 1, and so on, depending on the capabilities of the device. todo; This is pretty vague. Canonical orderings for popular controllers are probably necessary to avoid a big cross-browser mess. It would be nice to include definitions like `X360_A` == `BUTTON_0`, `PS3_CIRCLE` == `BUTTON_1`, and so on at the risk of getting overly specific.

The device need not be synchronously polled for this call (hence the timestamp field), but it should be polled as frequently as makes sense on the target platform, preferably to allow 60 fps or greater input sampling.

The values are stored in an array indexed by constants (rather than named fields, or separate arrays for each type of input) to allow for the common use of remapping, and iterating over all values to find deltas. It is intended that future devices will require new constants and types to be added, but this should be balanced by not having too many different input “types” so that in general games are able work with buttons, triggers, etc. without knowing about the specific details of the joystick.

Output to the vibration motors of the joysticks is omitted from the API with the thought that vibration could be shared with a future API that would vibrate things other than joysticks (esp. phones).

todo; It is not yet clear if implementations may need an open/close API around getState() to optimize access to the hardware. It would be nice to avoid if possible.

It’d be helpful for games to have navigator.keyboard.getState() and navigator.mouse.getState() to go along with navigator.joystick.getState() too.