Playing the MIDI output

Verovio can produce basic MIDI files and this feature is also available in the JavaScript toolkit. It can be used to play an MEI file directly in the browser as demonstrated in this tutorial.

Add a MIDI player

MIDI playback is not built-in to the web browser, nor available directly in Verovio. This means that we need to add a MIDI player to our page. For this tutorial we are going to use MIDIjs. You need to add a `<script>` tag with the following `src` attribute:

```
https://www.midijs.net/lib/midi.js
```

We also need buttons to handle the start and stop playing events. Add them a the top of the body above the notation div:

HTML / JAVASCRIPT

```
  <button id="playMIDI">Play</button>
  <button id="stopMIDI">Stop</button>
```

You also need to make sure they do something when clicking on them:

JAVASCRIPT

```
document.getElementById("playMIDI").addEventListener("click", playMIDIHandler);
document.getElementById("stopMIDI").addEventListener("click", stopMIDIHandler);
```

We now need to define what actually happens when the user clicks. That is, defining the `playMIDIHandler` and `stopMIDIHandler` functions we just bound to the button event listeners. We can scaffold them with:

JAVASCRIPT

```
const playMIDIHandler = function () {
  // do something to start playing
}

const stopMIDIHandler = function () {
  // do something to stop playing
}
```

At this stage they do nothing. To start playing, we need to get the MIDI data produced by Verovio and pass it to the player. We will use the Verovio `renderToMIDI` method that returns a MIDI file encoded as a base64 string, which we can pass to MIDIjs:

JAVASCRIPT

```
// Get the MIDI file from the Verovio toolkit
let base64midi = tk.renderToMIDI();
// Add the data URL prefixes describing the content
let midiString = 'data:audio/midi;base64,' + base64midi;
// Pass it to play to MIDIjs
MIDIjs.play(midiString);
```

Stop playing is even simpler. You only need to tell MIDIjs to do so:

JAVASCRIPT

```
MIDIjs.stop();
```

The examples above will be followed to write the body of the `playMIDIHandler` and `stopMIDIHandler` functions.

Highlighting the notes while playing

The MIDIjs player provides us with a callback function that gives us the current playback time. We can use this for highlighting the notes as the MIDI file plays! Each time the callback function is called, we can highlight the notes that are currently played, and automatically move to the next page if necessary.

You need to start by defining a callback function, similar to the button event we wrote earlier:

JAVASCRIPT

```
const midiHightlightingHandler = function (event) {
  // Do something everytime the callback function is called
}
```

You will notice that the function has an `event` parameter that will give us information about the current event. What we need to use is `event.time` that indicates the current playing time in seconds. We are going to use this and the Verovio `getElementsAtTime` method that retrieves all elements being played at a given time in order to obtain the list of notes being played:

JAVASCRIPT

```
// Get elements at a time in milliseconds (time from the player is in seconds)
let currentElements = tk.getElementsAtTime(event.time * 1000);
```

Now we should check that a page number was set. This is just to ensure we do not end up in an undefined state; for example, if the file is not loaded, or if we asked for elements that do not exist. If the page is 0, something went wrong and we should return:

JAVASCRIPT

```
if (currentElements.page == 0) return;
```

We should also check that we are currently rendering the correct page. If not, we should load it first:

JAVASCRIPT

```
if (currentElements.page != currentPage) {
  currentPage = currentElements.page;
  document.getElementById("notation").innerHTML = tk.renderToSVG(currentPage);
}
```

To do the highlighting of the notes we are going to use a CSS rule to be defined in the `style.css` file. A simple way to do it is to add a class `playing` to be applied to `g.notes` and that changes the color:

```
g.note.playing {
  fill: crimson;
}
```

Now we can actually highlight the notes. To do so, we are going to loop over the list of notes listed in `currentElements` and simply add the `playing` class to them:

JAVASCRIPT

```
// Get all notes playing and set the class
for (note of currentElements.notes) {
  let noteElement = document.getElementById(note);
  if (noteElement) noteElement.classList.add("playing");
}
```

Finally, we need to bind the MIDIjs player with the callback function we have defined. This will be done with:

JAVASCRIPT

```
MIDIjs.player_callback = midiHightlightingHandler;
```

This should not work! However, if it does, you will notice that we also need to de-highlight the notes that are not played anymore, otherwise they will stay highlighted. This should be done at the beginning of the `midiHightlightingHandler` callback function by removing the `playing` class to the notes that currently have it:

JAVASCRIPT

```
// Remove the attribute 'playing' of all notes previously playing
let playingNotes = document.querySelectorAll('g.note.playing');
for (let playingNote of playingNotes) playingNote.classList.remove("playing");
```

The code above needs to be placed within the body of the `midiHighlightingHandler` function.

Full example

Open this example in a new window.

HTML / JAVASCRIPT

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width">
        <title>MIDI playback</title>
        <!-- A stylesheet for modifying the appearance of the notes being played -->
        <link href="midi.css" rel="stylesheet" type="text/css" />
        <!-- Verovio -->
        <script src="https://www.verovio.org/javascript/develop/verovio-toolkit-wasm.js" defer></script>
        <!-- A JavaScript MIDI player -->
        <script src='https://www.midijs.net/lib/midi.js'></script>
    </head>
    <body>
        <button id="playMIDI">Play</button>
        <button id="stopMIDI">Stop</button>
        <div id="notation"></div>
        <script>
            /**
                We need to wait for the whole page to load before we try to
                work with Verovio.
            **/
            document.addEventListener("DOMContentLoaded", (event) => {
                verovio.module.onRuntimeInitialized = function () {
                    // This line initializes the Verovio toolkit
                    const tk = new verovio.toolkit();

                    tk.setOptions({
                        pageWidth: document.body.clientWidth,
                        pageHeight: document.body.clientHeight,
                        scaleToPageSize: true,
                    });

                    // The current page, which will change when playing through the piece
                    let currentPage = 1;

                    /**
                     The handler to start playing the file
                    **/
                    const playMIDIHandler = function () {
                        // Get the MIDI file from the Verovio toolkit
                        let base64midi = tk.renderToMIDI();
                        // Add the data URL prefixes describing the content
                        let midiString = 'data:audio/midi;base64,' + base64midi;
                        // Pass it to play to MIDIjs
                        MIDIjs.play(midiString);
                    }

                    /**
                     The handler to stop playing the file
                    **/
                    const stopMIDIHandler = function () {
                        MIDIjs.stop();
                    }

                    const midiHightlightingHandler = function (event) {
                        // Remove the attribute 'playing' of all notes previously playing
                        let playingNotes = document.querySelectorAll('g.note.playing');
                        for (let playingNote of playingNotes) playingNote.classList.remove("playing");

                        // Get elements at a time in milliseconds (time from the player is in seconds)
```

```
                        let currentElements = tk.getElementsAtTime(event.time * 1000);

                        if (currentElements.page == 0) return;

                        if (currentElements.page != currentPage) {
                            currentPage = currentElements.page;
                            document.getElementById("notation").innerHTML = tk.renderToSVG(currentPage);
                        }

                        // Get all notes playing and set the class
                        for (note of currentElements.notes) {
                            let noteElement = document.getElementById(note);
                            if (noteElement) noteElement.classList.add("playing");
                        }
                    }

                    /**
                        Wire up the buttons to actually work.
                    */
                    document.getElementById("playMIDI").addEventListener("click", playMIDIHandler);
                    document.getElementById("stopMIDI").addEventListener("click", stopMIDIHandler);
                    /**
                     Set the function as message callback
                    */
                    MIDIjs.player_callback = midiHightlightingHandler;

                    // This line fetches the MEI file we want to render...
                    fetch("https://www.verovio.org/examples/downloads/Schubert_Lindenbaum.mei")
                    // ... then receives the response and "unpacks" the MEI from it
                    .then((response) => response.text())
                    .then((meiXML) => {
                        // ... then we can load the data into Verovio ...
                        tk.loadData(meiXML);
                        // ... and generate the SVG for the first page ...
                        let svg = tk.renderToSVG(1);
                        // ... and finally gets the <div> element with the ID we specified,
                        // and sets the content (innerHTML) to the SVG that we just generated.
                        document.getElementById("notation").innerHTML = svg;
                    });
                }
            });
        </script>
    </body>
</html>
```