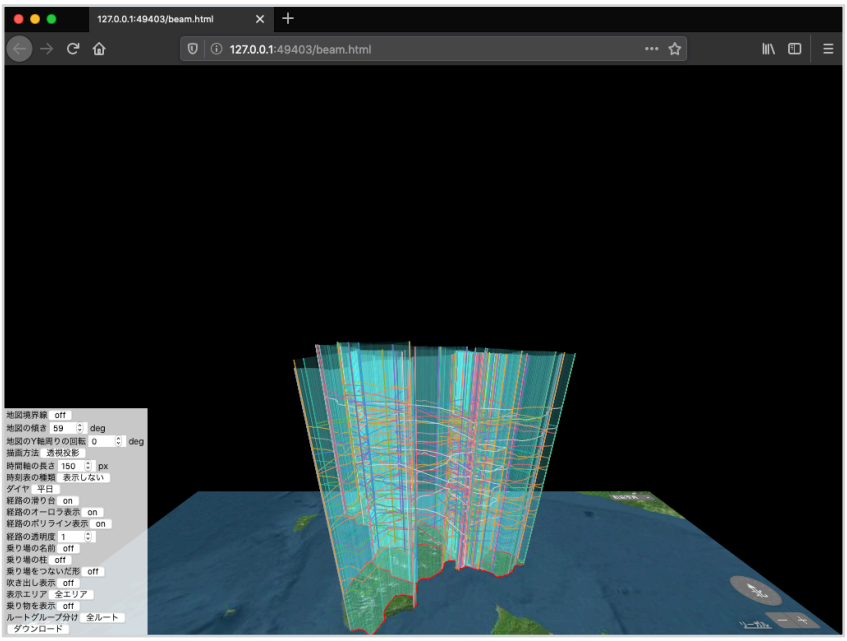
GSLetterNeo vol.137
2019年12月

Webブラウザにおける2Dコンテンツ上での透視投影変換を用いた3D表現（１）

松原 伸人 matubara@sra.co.jp

はじめに

テキストや画像や動画像に地図などWebブラウザ上のコンテンツの多くは二次元データです。今回は、Webブラウザ上の2Dコンテンツに透視投影変換を適用することで、3D的に表示する方法を紹介します。既にある2Dコンテンツを3D的に表示できると、画像や地図の上に付加的な情報を立体的に重ねた表現ができます。オープンデータの１つとして、公共交通の運行情報が **GTFS** で公開されるようになってきています。次の画像は、開発中の、運行情報を地図上に表示するプロトタイプです。



この画像は、プロトタイプを行うにあたり、共同研究を行っている公立はこだて未来大学の協力により、函館バスから提供いただいた運行情報を GTFS に変換して使用させていただきました。

北海道函館市で運行している函館バスの全運行経路を表示しています。地図上を這うようにして描かれている赤い線は、路線バスの各系統が走行する運行経路で、始発駅から停車する順に停留所を直線で結んで描いています。各運行経路の始発駅と終着駅の場所を垂直に線を描いて表しています。この垂直線は、上端が0時で下端が24時の時間軸も表しています。各系統は日ごとや曜日ごとなどで、始発駅を何時に発車して途中の駅に何時に止まって終着駅に何時に着くかといった旅程が組まれています。各運行経路上に描かれている線はその系統の旅程を表しています。上の方を走っている線は午前中に

GTFS
静的な GTFS の概要 | Static Transit | Google Developers

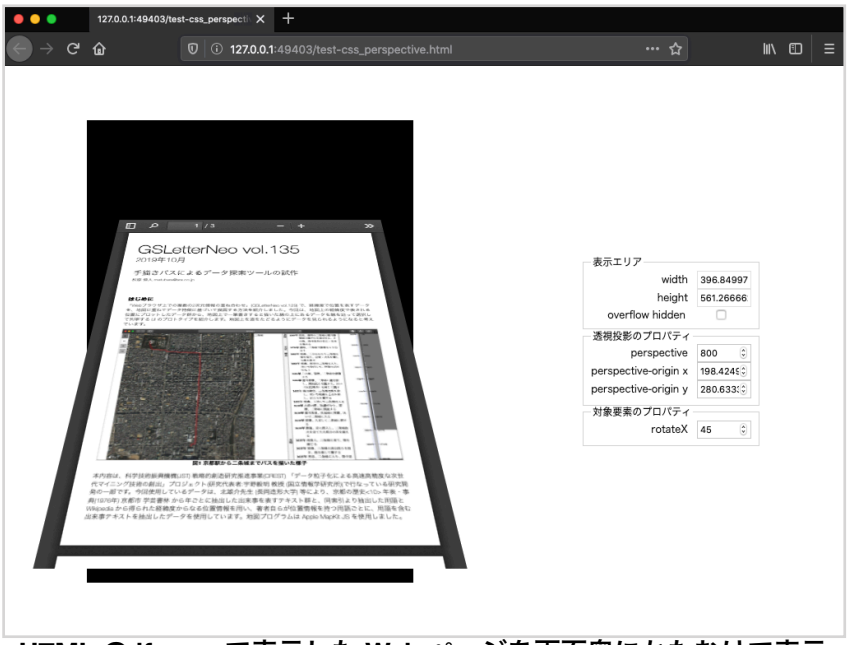
走行するとか、朝方と夕方に走行本数が多いとか、多くの系統が走っているエリアがどこか、などといったことがわかります。

地図には、Web ページに組み込んで表示できるインタラクティブな地図フレームワークである Apple MapKit JS を使用しています。運行情報の描画には HTML の Canvas の CanvasRenderingContext2D を用いています。**Apple MapKit JS** を用いて表示した HTML エレメントに、CSS の **perspective** と **transform** を設定することで、透視投影変換を行い、画面奥に向かって59度かたむけて表示しています。

2Dコンテンツの3D表示

次のプログラムは、同様の方法を HTML の **iframe** に適用し、Web ページを画面奥にかたむけて表示する例です。以降で説明するように、透視投影での3D表示は、投影変換プログラムを書かなくても HTML と CSS を書くだけで実現できます。次の URL を Web ブラウザで開いて試せるようにしています。

https://www3.sra.co.jp/ktl/nobutomatsubara/test-css_perspective.html



HTML の iframe で表示した Web ページを画面奥にかたむけて表示

上記の URL を Web ブラウザで開いて Web インスペクタでプログラムを見ることができます。また、同じプログラム例を本号の末尾に掲載しています。プログラムをコピーして HTML ファイルを作成すれば、上記の URL と同じプログラムを手元の Web ブラウザでも動かせます。

画面左の縦長の黒塗りのエリアが perspective を設定して透視投影により3D空間になっています。画面右には表示エリアの大きさと、透視投影のプロパティと対象要素のプロパティを表示しています。**透視投影のプロパティ**にある perspective および perspective-origin は、現在の設定値を表示しています。設定値を変えると、CSS の値が変わり透視投影変換の結果が変わります。**perspective プロパティ**は、ピクセルなどで指定する、ユーザの視点から消失点までの距離です。距離が近いと手前のものが手前に大きく表示され、距離が長いと消失点に向かって遠ざかった方向に表示されるよう、遠近感が変わります。**perspective-origin** を設定すると透視投影の消失点の位置を設定できます。perspective-origin は、使用する表示エリアの横方向および縦方向

MapKit JS
MapKit JS | Apple Developer Documentation

CanvasRenderingContext2D
CanvasRenderingContext2D - Web API | MDN

perspective
perspective - CSS | MDN

transform
transform - CSS | MDN

iframe
インラインフレーム要素 - HTML | MDN

perspective-origin
perspective-origin - CSS | MDN

の位置で指定します。**対象要素のプロパティ**にある rotateX は、iframe の画面奥方向への傾きの角度を表しています。数値を変えると、指定した角度に傾きが変わります。画面奥への傾きは、CSS の transform に rotateX 関数を指定して設定できます。プログラム例では、rotateX 欄への入力値を cssTransformRotateX 関数が受け取り、target という id が設定してある iframe のスタイルの --rotationX **カスタムプロパティ**を setProperty を用いて変更しています。perspective と perspective-origin もそれぞれ同様の方法で cssPerspective 関数と cssPerspectiveOrigin にそれぞれ実装しています。**表示エリア**の overflow hidden にチェックを入れると、黒いエリアからはみ出ている部分が表示されなくなります。これも CSS の **overflow** プロパティに hidden を設定するだけで表現できます。JavaScript 部分を省いて HTML と CSS だけにすると次のような40行程度の短いプログラムで実現できます。

https://www3.sra.co.jp/ktl/nobutomatsubara/test-css_perspective-css_only.html

カスタムプロパティ

CSS カスタムプロパティ (変数) の使用 - CSS | MDN

overflow

overflow - CSS | MDN

プロトタイピング tips

先に示したプログラムでは画面右側に、今回紹介した3D表示方法に関係する CSS のプロパティを表示して、画面上で各プロパティを変更できるようにしています。これは、どれくらいの傾きにすると見やすいかとか、消失点の距離はどれくらいにすると見やすいかとか、見た目の違いを素早く試せるようにするために、ソフトウェアのプロトタイプで必要に応じて入れるようにしているコードです。コードにして残しておくことにより、素早く試せるだけでなく、後から振り返ってコードの主旨を思い出しやすくなると思います。画面右側部分がない2つ目のプログラムだけでも、web インスペクタ上に直接 JavaScript で同様のコードが書けるので、素早く試すことができる、という目的は果たせていますが、web インスペクタを閉じるとコードが消えて残りません。**Jupyter Notebook** は、Web インスペクタ上で行うような、ちょっと試すために書くようなコードの断片も記録して残しやすい、コードとメモを記録できるプログラミング環境です。一連のコードを文脈ごととか断片的に分けて書いて、断片それぞれにメモをつけたりする感じにノートを書くようにプログラムしていけます。おもに python 言語でのプログラミングを行う環境ですが、HTML や JavaScript も書けるようになっています。

Jupyter Notebook

Project Jupyter

おわりに

今回は、CSS を用いた透視投影変換による 3D 表現について紹介しました。次回は、冒頭で紹介した事例のように、3D 的に情報を付加して重ねて表示する方法を紹介する予定です。

GSLetterNeo vol.137
発行日 2019年12月20日
発行者 株式会社 ＳＲＡ 先端技術研究所
編集者 土屋 正人
バックナンバー https://www.sra.co.jp/gsletter/

お問い合わせ
gsneo@sra.co.jp
〒171-8513 東京都豊島区南池袋2-32-8
株式会社SRA

2Dコンテンツ（iframe）を3D表示するプログラム例

```
<html>

<body>
<!-- 3D表示用スタイル -->
    <style>
        html,
        body {
            margin: 0;
            padding: 0;
        }

        body {
            width: 100%;
            height: 100%;
            overflow: hidden;
            display: flex;
            justify-content: space-around;
            align-items: center;
        }

        #threeDarea {
            background-color: black;

            --perspective: 800px;
            --perspective-origin: 50% 50%;
            perspective: var(--perspective);
            perspective-origin: var(--perspective-origin);
        }

        #target {
            width: calc(210mm / 2);
            height: calc(297mm / 2);
            border: none;

            --rotationX: 45deg;
            transform: rotateX(var(--rotationX));
        }
    </style>

<!-- 画面右側のオプション表示用 -->
    <style>
        fieldset {
            font-size: 12px;
            line-height: 12px;
            border: solid 1px lightgray;
            margin: 2px;
            display: flex;
            flex-direction: column;
            align-items: flex-end;
        }

        fieldset input {
            width: 6em;
            margin-left: 1em;
            box-shadow: none;
        }

        fieldset input:read-only {
            border: none;
            color: gray;
        }
    </style>
    <div id="threeDarea">
        <iframe id="target" src="https://www2.sra.co.jp/Portals/0/files/gsletter/pdf/GSLetterNeoVol135.pdf"></iframe>
    </div>
    <div class="options">
        <fieldset>
            <legend>表示エリア</legend>

            <label>width<input type="text" readonly id="threeDareaWidth"></label>
            <label>height<input type="text" readonly id="threeDareaHeight"></label>
            <label>overflow hidden<input type="checkbox" id="threeDareaOverflowHidden"></label>
        </fieldset>
        <fieldset>
            <legend>透視投影のプロパティ</legend>

            <label>perspective<input type="number" value="800" id="perspective"></label>
            <label>perspective-origin x<input type="number" value="0" id="perspectiveOriginX"></label>
            <label>perspective-origin y<input type="number" value="0" id="perspectiveOriginY"></label>
        </fieldset>
        <fieldset>
            <legend>対象要素のプロパティ</legend>

            <label>rotateX<input type="number" value="45" id="transformRotateX"></label>
        </fieldset>
    </div>
    <script>
        function cssPerspective(element, pxValue) {
            let cssString = pxValue + 'px';
```

```
            styleName = '--perspective'
        element.style.setProperty(styleName, cssString)
    }

    function cssPerspectiveOrigin(element, x, y) {
        let xPosition = x + 'px',
            yPosition = y + 'px',
            cssString = [xPosition, yPosition].join(' '),
            styleName = '--perspective-origin'
        element.style.setProperty(styleName, cssString)
    }

    function cssTransformRotateX(element, degrees) {
        let cssString = degrees + 'deg',
            styleName = '--rotationX'
        element.style.setProperty(styleName, cssString)
    }

    function cssOverflowHidden(element, aBoolean) {
        let cssString = aBoolean ? 'hidden' : 'visible',
            styleName = 'overflow'
        element.style.setProperty(styleName, cssString)
    }

    let threeDarea = document.getElementById('threeDarea'),
        threeDareaOverflowHidden = document.getElementById('threeDareaOverflowHidden'),
        rect = threeDarea.getBoundingClientRect()

    document.getElementById('threeDareaWidth').value = rect.width
    document.getElementById('threeDareaHeight').value = rect.height
    threeDareaOverflowHidden.addEventListener('change', (evt) => {
        cssOverflowHidden(threeDarea, threeDareaOverflowHidden.checked)
    })

    let perspective = document.getElementById('perspective'),
        perspectiveOriginX = document.getElementById('perspectiveOriginX'),
        perspectiveOriginY = document.getElementById('perspectiveOriginY')

    perspectiveOriginX.value = rect.width / 2
    perspectiveOriginY.value = rect.height / 2
    cssPerspectiveOrigin(threeDarea, perspectiveOriginX.value, perspectiveOriginY.value)

    perspective.addEventListener('change', (evt) => {
        cssPerspective(threeDarea, evt.target.value)
    })
    perspectiveOriginX.addEventListener('change', (evt) => {
        cssPerspectiveOrigin(threeDarea, perspectiveOriginX.value, perspectiveOriginY.value)
    })
    perspectiveOriginY.addEventListener('change', (evt) => {
        cssPerspectiveOrigin(threeDarea, perspectiveOriginX.value, perspectiveOriginY.value)
    })

    let targetElement = document.getElementById('target')
    let transformRotateX = document.getElementById('transformRotateX')

    transformRotateX.addEventListener('change', (evt) => {
        cssTransformRotateX(targetElement, transformRotateX.value)
    })
</script>
</body>

</html>
```