第 26 回日本バーチャルリアリティ学会大会論文集（2021 年 9 月）

Virtual Reality 空間における HTML に組み込み可能な新規属性を用いた 360 度サイト作成手法の提案

Method for Creating 360 Degree Website Using New Embeddable Attributes of HTML in Virtual Reality Space

新井響 [1)]，坂本雄児 [2)]
Hibiki ARAI, Yuji SAKAMOTO

1) 北海道大学大学院 情報科学院 （〒060-0814 札幌市北区北 14 条西 9 丁目, baseballbear-0606@eis.hokudai.ac.jp）

2) 北海道大学大学院 情報科学研究院 （〒060-0814 札幌市北区北 14 条西 9 丁目, yuji@ist.hokudai.ac.jp ）

概要：本稿では，HTML の属性に着目し，新たに Virtual Reality 空間向けの属性を追加した 360 度サイト作成手法を提案する．提案手法は，既存の 360 度サイト作成手法よりも容易に，従来の HTML では対応できない 3 次元表現を可能にする． また， 本来のサイトの構成を崩さず従来の 2 次元ブラウザでの閲覧も可能である. 本手法の有効性を確認するため， 既存手法の一つである A-Frame との比較評価を行った.

キーワード：作業支援・評価，ユーザインターフェース，HTML

1. はじめに

昨今，エンターテインメントや医療などの分野において, Virtual Reality(VR) の応用が進んでいる．ここで VR とは，見かけや名目上現実とは異なるが，実質的に現実と同等の機能を有するもの，またはその環境を作り出す技術やシステムを指す．さまざまな感覚刺激をユーザーに与えることにより人工的に現実感を発生させることを可能としている VR だが，その応用が進歩する中，一般に使用される VR 上での Web ブラウザは仮想ウィンドウによるデスクトップのミラーリングといった必要最低限の機能に留まっている現状がある．

仮想ウィンドウの研究として, Toyama ら[1] は並列 Web ブラウジングに欠かせない Web ページの検索を支援するために，様々な可視化・インタラクション手法を活用した Web ブラウザ「VRowser」を提案した．その中の実験において, VR 環境のランドマークに記憶しやすさが依存していることを発見しており, 3 次元表現による効果が期待できることを示した.

VR 上でのサイトの 3 次元表現がすでにいくつかの分野で実現されている. VR/AR クリエイティブプラットフォーム「STYLY」 では，株式会社パルコが VR/AR でのショッピング[2]を，バーチャル住宅展示場 「MY HOME MARKET」[3] では，いくつかの建築会社が VR 空間上での内見システムを実際の営業として用いている．

このような VR サイトコンテンツの作成を手助けする手法には, Mozilla が開発した“A-Frame”[4] という Web ブラウザ上で動作する 3D と VR 開発のための WebVR フレームワークなどがある[5]. A-Frame は，A-Frame の JSON ファイルを HTML 内に組み込むだけで WebVR の機能を実装することができ，内部では three.js が使われている. <a-img> や<a-text> 等を用いて, VR に対応させることが可能であり, VR コンテンツを簡単に実装できる．しかしこの手法では，既存の HTML を用いたサイトとは別に, VR 専用のサイトを作成する必要がある．特に個人での作成には敷居が高い.

そこで，本研究では HTML の属性に着目した既存の HTML を引き継いで作成できる新たな 360 度サイト作成手法を提案する.

2. 研究の手法

本研究では VR 空間上に対応する新規属性を作成し，それを読み取ることで 3 次元空間に文章や動画像を出力するシステムを提案する. DOM や正規表現, Xpath を用いて HTML パーサを駆使することで, HTML のタグや属性とそれに属するテキストや写真を抽出している.

本手法では，新規属性として Dpos, Drot, Dpol, win を提案しており，既存の HTML のタグではカバーできない 3 次元表現を設定できるようにしている．図 1 に示すように,

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>提案手法説明</title>
</head>
<body bgcolor="#fdffba">

    <h1 Dpos="0,1,1">見出し</h1>

    <div class="article" style="background-color:#EDF7FF;width:240px;padding:10px;border:1px solid #ccc">
        <div class="headline" Dpos="-0.6,0.72,1.0"><h3>DposとDrot</h3></div>
        <div Dpos="-0.1,0.13,0.3">
            ここ全体はDposでの位置
        </div>
        <div Dpos="-0.1,0.1,0.3" Drot="45,125,3">
            ここだけDrotで傾き変更
        </div>
    </div>

    <div class="article" style="background-color:#EDF7FF;width:280px;padding:10px;border:1px solid #ccc">
        <div class="headline" Dpol="1,75,130"><h3>Dpol</h3></div>
        <div class="botl" Dpol="0.3,82,105">
            ここ全体をDpolで位置、傾きを調整
        </div>
    </div>

    <div class="article" style="background-color:#EDF7FF;width:320px;padding:10px;border:1px solid #ccc">
        <div class="headline" Dpos="-0.5,0,0.64" win="188,33,-0.2,-0.36,2.0,0,0,0"><h3>win</h3></div>
        <div class="botl" Dpos="-0.06,-0.02,0.3">
            ウィンドウ表示行い、ここ全体をまとめる
        </div>
    </div>

</body>
</html>
```

図 1: 提案手法の HTML ソース

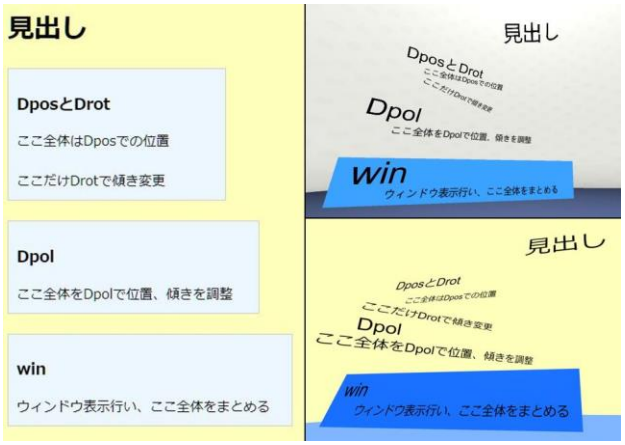

図 2 : 2 次元ディスプレイでの出力(左).
提案手法での出力(右上).A-Frame での出力(右下)

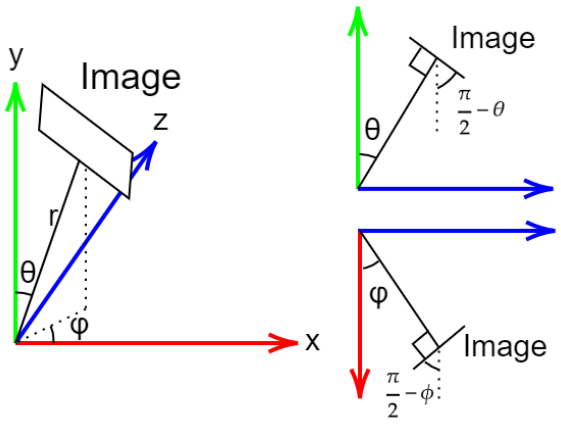

図 3:コンテンツの角度変換

既存の HTML の属性に新規属性を設定する．提案する新規属性のソースには色をつけており，次節で各属性の詳細を示す．この新規属性は，既存のブラウザでは無視されるため，従来の 2 次元ブラウザで閲覧する際のデザインに干渉することなくシステム上で 3 次元表現が可能になる．つまり，既存の HTML に新たにこれらの新規属性を書き加えるだけで従来の 2 次元ブラウザでのサイトと 360 度サイトの両立を可能にしている．これを図 2 に示す．

2.1 Dpos(図 1-赤線)

表示するオブジェクトの位置を x[m], y[m], z[m] で指定する．使い方は のようになる．原点は自身の頭の位置となる．

2.2 Drot(図 1-緑線)

表示するオブジェクトの回転を x[°], y[°], z[°] で指定する．回転方法は x 軸 y 軸 z 軸の順となる．使い方は のようになる．オブジェクトの中心が回転中心となる．

2.3 Dpol(図 1-青線)

表示するオブジェクトの位置を極座標 r[m],θ[°], φ[°] で指定する．回転軸はθ が x 軸, φ が y 軸となる．使い方は <div Dpol="r,θ,φ"> のようになる．利用者が感覚的にコンテンツの位置を決める場合は，直行座標系(x, y, z) よりも極座標系(r,θ,φ) の方が管理がしやすい可能性があるため実装した．極座標系から直行座標系への変換式(1)を用いることで変換を行い，位置を決定する．

$$\begin{cases} x = r\sin\theta\cos\varphi \\ y = r\cos\theta \\ z = r\sin\theta\sin\varphi \end{cases} \quad (1)$$

$$0 < r, 0 \le \theta \le \pi, 0 \le \varphi \le 2\pi$$

```
<!DOCTYPE html>
<head>
    <meta charset=utf-8" >
    <title>提案手法説明:A-FRAME</title>
    <script src="https://aframe.io/releases/1.2.0/aframe.min.js"></script>
</head>
<a-scene background="color:#fdffba">
    <a-plane color="#70a0ff"
             width="50" height="50"
             position="0 0 -3"
             rotation="-90 0 0"></a-plane>
    <a-entity mb-text position="0.0 4.5 -2" scale="4.0 4.0 1.0" data-text="見出し"></a-entity>

    <a-entity mb-text position="-1.1 2.7 -1" data-text="DposとDrot"></a-entity>
    <a-entity mb-text position="-1.0 2.5 -1" scale="0.6 0.6 1.0" data-text="ここ全体はDposでの位置"></a-entity>
    <a-entity mb-text position="-1.0 2.25 -1" rotation="0.0 30.0 0.0" scale="0.6 0.6 1.0" data-text="ここだけDrotで傾き変更"></a-entity>

    <a-entity mb-text position="-1.44 2.19 -1" rotation="20.0 30.0 0.0" data-text="Dpol"></a-entity>
    <a-entity mb-text position="-0.8 1.95 -1" rotation="20.0 30.0 0.0" scale="0.6 0.6 1.0" data-text="ここ全体をDpolで位置、傾きを調整"></a-entity>

    <a-plane color="#1561f9"
             width="2.0" height="0.4"
             position="-0.95 1.64 -1.1"
             rotation="0 0 0"></a-plane>
    <a-entity mb-text position="-1.5 1.7 -1" data-text="win"></a-entity>
    <a-entity mb-text position="-0.75 1.56 -1" scale="0.6 0.6 1.0" data-text="ウィンドウ表示行い、ここ全体をまとめる"></a-entity>
</a-scene>
</body>
```

図 4: A-Frame の HTML ソース

また，このときオブジェクトの向きをカメラ方向にするために傾きをつけるようにする．図 3 より各軸に対するオブジェクトの角度は以下の式(2) のようになる．

$$\begin{cases} rot_X = \theta - \dfrac{\pi}{2} \\ rot_Y = \dfrac{\pi}{2} - \varphi \\ rot_Z = 0 \end{cases} \qquad (2)$$

$$0 < r, 0 \leq \theta \leq \pi, 0 \leq \varphi \leq 2\pi$$

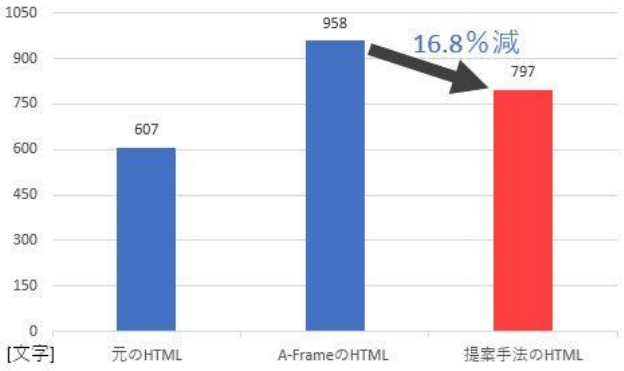

図 5:実験 1 のそれぞれの文字数比較

2.4 win(図 1-黄線)

テキスト等を見やすくするためのウィンドウ表示を可能にする.ウィンドウのサイズ(width,height)や位置(x,y,z),傾き(X,Y,Z)を指定できる．使い方は のようになる.

3. 実験結果

既存の手法である A-Frame と今回の提案手法において出力結果と実装に必要な文字数を比べることで評価を行った．実際のコンテンツの構造部分である<body>間で文字数比較を行っている．システムは Unity2019.3.7f1 を利用して作成し，プログラムは HTML パーサである Html Agility Pack (HAP)を軸に C# で書かれている.

3.1 実験 1

コード量の少ないサイトでの比較を行った．A-Frame の HTML ソースが図 4，提案手法の HTML ソースが図 1 に示されており，それぞれの出力は図 2 のようになった．各 HTML ソースの文字数を図 5 に示す.

元の文字数より，A-Frame の文字数 が 57.8 %，提案手法の文字数が 31.3 %増加している．図 2 より，同様の 3 次元表現を A-Frame と比較した場合では，提案手法は 16.8 %の文字数の削減に成功した．また，元の HTML との差異を比べると，A-Frame は 876 文字，提案手法では 190 文字が新たに書き加えたコードであり，A-Frame に比べて約 0.22 倍の記述量で相似な 3 次元表現を表すことができた.

3.2 実験 2

コード量の多いサイトでの比較を行った．提案手法では図 6 のような出力になった．実際に運用しているサイトであるため，元の HTML には CSS 等が含まれている．各 HTML ソースの文字数を図 7 に示す.

元の文字数より，A-Frame の文字数が 58.4 %，提案手法の文字数が 18.7 %増加している．実験 1 と同様に A-Frame と比較した場合では，提案手法は 27.7 %の文字数の削減に成功した．新たに書き加えたコード量でも A-Frame が 3068 文字，提案手法が 456 文字で 3 次元表現を表すことができ，約 0.15 倍の記述量で作成できた.

図 6:実験 2 の元 HTML 出力(左).提案手法の出力(右)

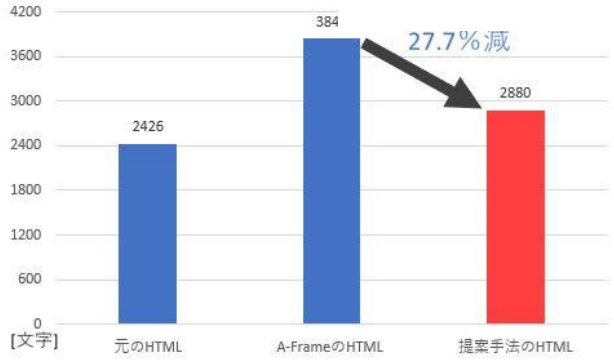

図 7: 実験 2 のそれぞれの文字数比較

4. 結論・まとめ

既存の HTML に新しい属性を追加する拡張を行うことで既存手法である A-Frame より少ないコード量で 360 度サイトの作成が可能であることを確認した．また，既存手法と異なり，新たに HTML のコードを書き直す必要がないため，非常に簡単に 360 度サイトの作成が可能である．通常の HTML レンダリングも問題なく行われており，従来の 2 次元ブラウザとの互換性を保ちつつ 3 次元表現を可能とするサイト作成を行うことが可能である．しかし本手法では,テキストと画像の出力とハイパーリンクによる画面遷移といった最低限の機能しか持ち合わせていない．複雑な HTML や CSS には対応していないため，今後は本手法に対応するタグや属性を増やしていくことが課題である．

参考文献

[1] Adrian H. Hoppe, Florian van de Camp, Rainer Stiefelhagen, “Enabling Interaction with Arbitrary 2D Applications in Virtual Environments”, Springer, Cham , 2020.

[2] STYLY, https://styly.cc/ja/news/parco_shopping/, (2021/7/28 アクセス).

[3] MY HOME MARKET, https://myhomemarket.jp/, (021/7/28 アクセス).

[4] A-FRAME, https://aframe.io/, (2021/7/28 アクセス).

[5] Solange Gomes Santos, Jorge C. S. Cardoso, “Web-based Virtual Reality with A-Frame”, CISTI, 2019.