Web開発フローと親和するWeb3Dライブラリ

―Web開発のためのWebGLフレームワーク“Grimoire.js”―

1.　背景

近年登場した新しいWeb技術の一つ、“WebGL”によってWeb上では今まで成し得なかった高度な映像表現が可能になった。一方、この技術は元々Web技術界隈で使われていたものから大きく離れた、ゲーム開発などのグラフィクスエンジニアが用いる技術に近い。そのため、今まではWebエンジニアが使いこなせず、現実的なコストでWebGLを用いたWeb開発ができなかった。

ここで、我々はWebエンジニアとグラフィクスエンジニアの仲介に立つフレームワークを作る事で、新たなWeb表現を用いることを現実的にすることを可能にしていく。

2.　目的

本プロジェクトでは、大きく異なるそれぞれのパラダイムに属する、Webエンジニアとグラフィクスエンジニアに対して、それぞれに適切なAPIを提供して、結果的に意識せずともこの全く異なる種類のエンジニアが共同作業できるようなフレームワークを開発する。

このフレームワークをベースにして、さまざまなWebGLを用いた表現を実現するための拡張機能を構築し、Webが文字や画像に次ぐ新たな表現の筆としてWebGLを獲得する未来を創造する。

3.　開発の内容

本プロジェクトで開発したフレームワーク“Grimoire.js”の主な特徴の一つが、対象となるエンジニアによって用いるAPIが異なるという点である。Webエンジニアとグラフィクスエンジニアに対し、それぞれ慣れ親しんでいるAPIを提供して、結果的にお互いの特徴を意識しないで済む共同作業環境を提供しなくてはならない。

そこで本プロジェクトではまず、この2つのAPIとその相互関係を定義するような仕様を策定して、それらを管理するだけのフレームワークのコアとなる部分を開発した。

- WebエンジニアサイドのAPI

WebエンジニアサイドのAPIとしては、まず初期状態を定義するための、HTMLに近い役割をWebGL上で実現するためのXMLベースの仕様GOMLを策定した（図1）。

さらに、JavaScript側のAPIとして最も基本的な値の変更とイベントベースの処理を、既存のWebライブラリとして最も有名なjQueryに近い書き方で記述できるようにするための仕様をまとめたAPIを開発した（図2）。

```
<goml width="fit" height="fit">
    <geometry name="cylinder" type="cylinder" />
    <scene>
        <camera class="camera" near="0.01" far="40.0" aspect="1.5" fovy="45d">
            <camera.components>
                <MouseCameraControl/>
            </camera.components>
        </camera>
        <mesh geometry="cylinder" position="0,0,0" scale="3,3,3" color="red"/>
    </scene>
</goml>
```

図1: WebエンジニアサイドのAPIの例（GOML）

```
var mesh = gr("#canvas")("mesh");

mesh.on("mouseenter", function() {
  mesh.setAttribute("diffuse", "blue");
});

mesh.on("mouseleave", function() {
  mesh.setAttribute("diffuse", "orange");
});
```

図2: WebエンジニアサイドのAPIの例（JavaScript）

- **グラフィクスエンジニアサイドのAPI**

一方、グラフィクスエンジニアサイドにはWebエンジニアとは異なり、Unityなどのモダンなゲームエンジンのアーキテクチャに近いAPIを提供した。さらにこのAPIにより、記述される内容がそのままWebエンジニアサイドから操作可能となる設計となった（図3）。

これらの二つの手法にまたがるフレームワークの構造は、WebGLに限定せず、思想の異なるAPIを共存させるための設計手法の一つとして、価値があるものである。さらに本プロジェクトではこの仕組み上で最適かつ汎用的なWebGLのレンダラライブラリを構築した（図4）。

さらに、本プロジェクトの期間中ではWebGLと一緒に利用するであろう様々な機能についてプラグイン開発を行った。例としては、「シェーディング」、「モデル読み込み」、「物理エンジン」、「文字レンダリング」など一般的に必要とされるものが優先して開発した（図5）。

```
gr.registerComponent("Rotate", {
  attributes: {
    speed: {
      default: 0.2,
      converter: "Number",
    },
  },
  $mount() {
    this.transform = this.node.getComponent("Transform");
    this.current = 0;
  },
  $update() {
    this.current += this.getAttribute("speed");
    this.transform.rotation = `y(${this.current})`;
  }
});
```

図3: グラフィクスエンジニアサイドのAPIの例

図4: レンダラプラグインを利用して描画した宇宙のエフェクト

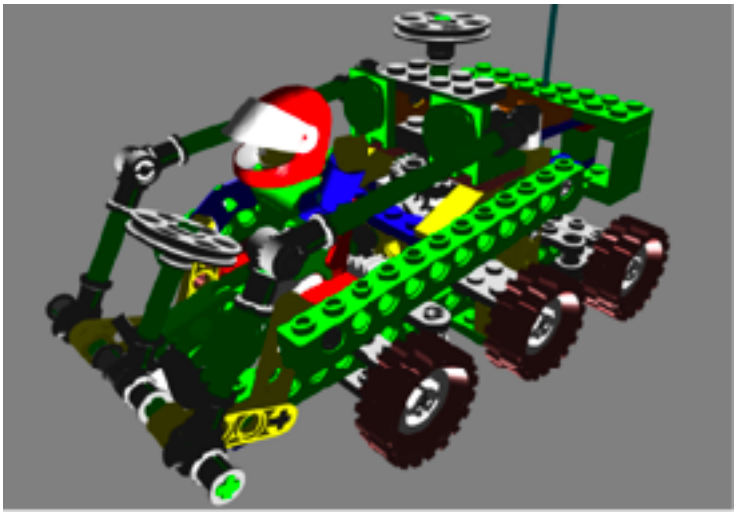

図5: モデル読み込みプラグインを用いたモデル表示

4.　従来の技術（または機能）との相違

WebGLをラップして使いやすい形で提供しているライブラリは多数存在するが、実際にはグラフィクス系のプログラムに慣れているエンジニアが、Web上でもコンテンツを開発できるようにするためのライブラリとして存在している場合が多い。本ライブラリは、これに加えてWebエンジニアの存在を想定し、これらのエンジニアが同時に存在するようなワークフローを想定した。これによって、Webエンジニアが開発するWebサービスにおいて、現実的な開発コストでWebGLが活用できるようになると推測される。

さらに、グラフィクス系のエンジニアによって開発された本ライブラリにおけるコンポーネント群も、近年のWeb開発で用いられるnpmなどのパッケージ配信システムや、jsのscriptファイル単体でのプラグインとして配布することが極めて容易であるため、Web上での3D表現として再利用可能なリソース群が増える未来を提供できる。

5.　期待される効果

Grimoire.jsを用いることでWeb開発フローの中に自然と WebGL表現を組み込むことが可能になる。これによって、実際の開発の中でWebGLを扱うコストが削減され、WebGLを利用したインタラクティブなWebサービスが増加されることが期待される。また、Webエンジニアとグラフィクスエンジニアの連携が容易になる。

6.　普及（または活用）の見通し

本プロジェクト期間終了後もハンズオンを開催することが決まっている。コミュニティ活動として、以降も利用者増加を目指していくとともにフィードバックを求め、ライブラリ開発を発展させていく。

特に、ライブラリの普及を目指すターゲットを日本国内に限定せず世界中に普及するように、ライブラリのドキュメントの整備を続け、日本初のオープンソースソフトウェアの成功例の一つとして、新たにこのライブラリの名前を広げたいと画策している。

7. クリエータ名（所属）

石井 翔（東京理科大学）
秋澤 一史（東京理科大学）
大谷 拓海（東京理科大学）
城山 賢人（東京理科大学）

関連URL

https://grimoire.gl/
https://github.com/GrimoireGL/GrimoireJS/