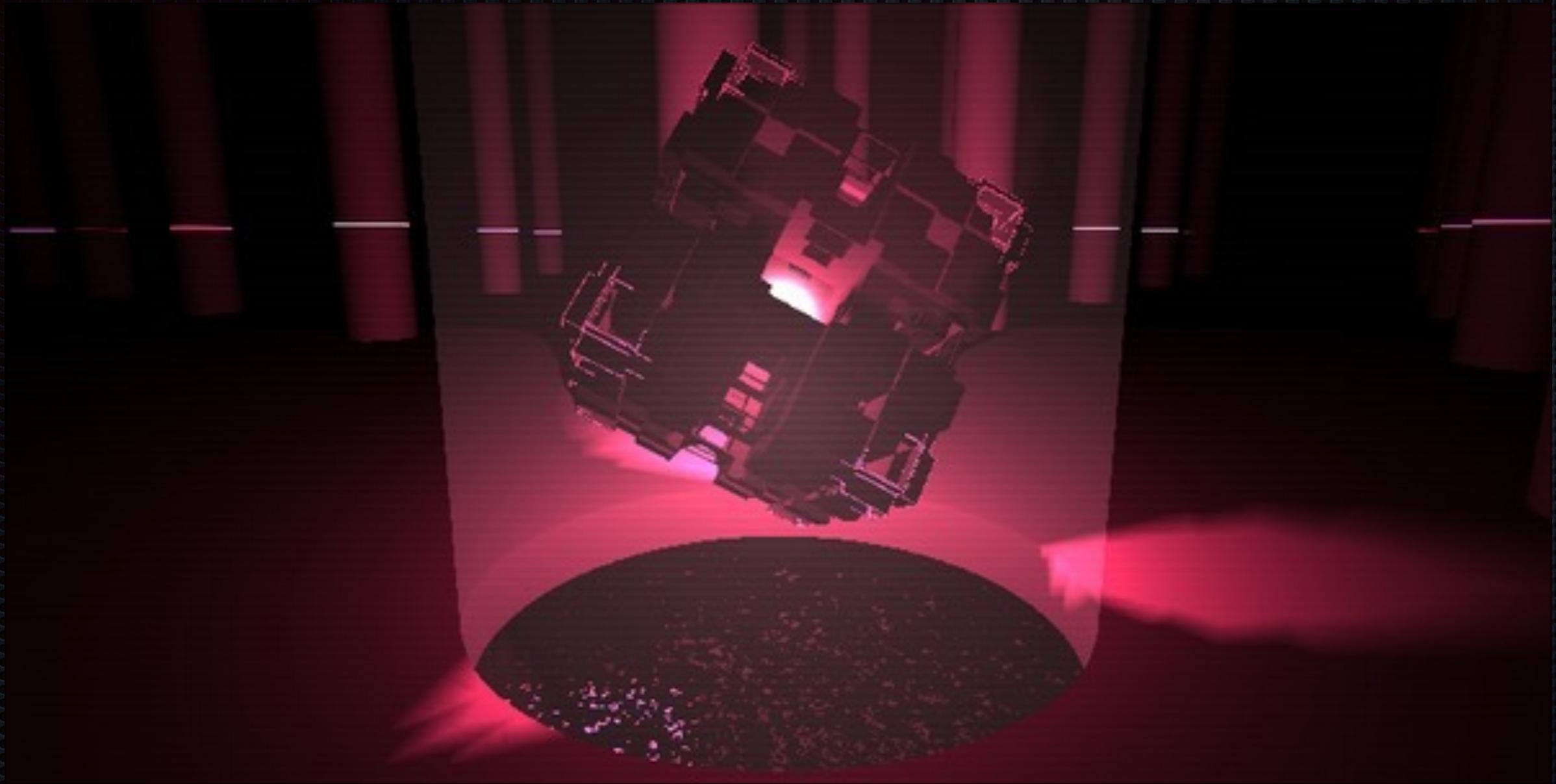


WebGL とモバイルウェブの 「これまで」と「これから」

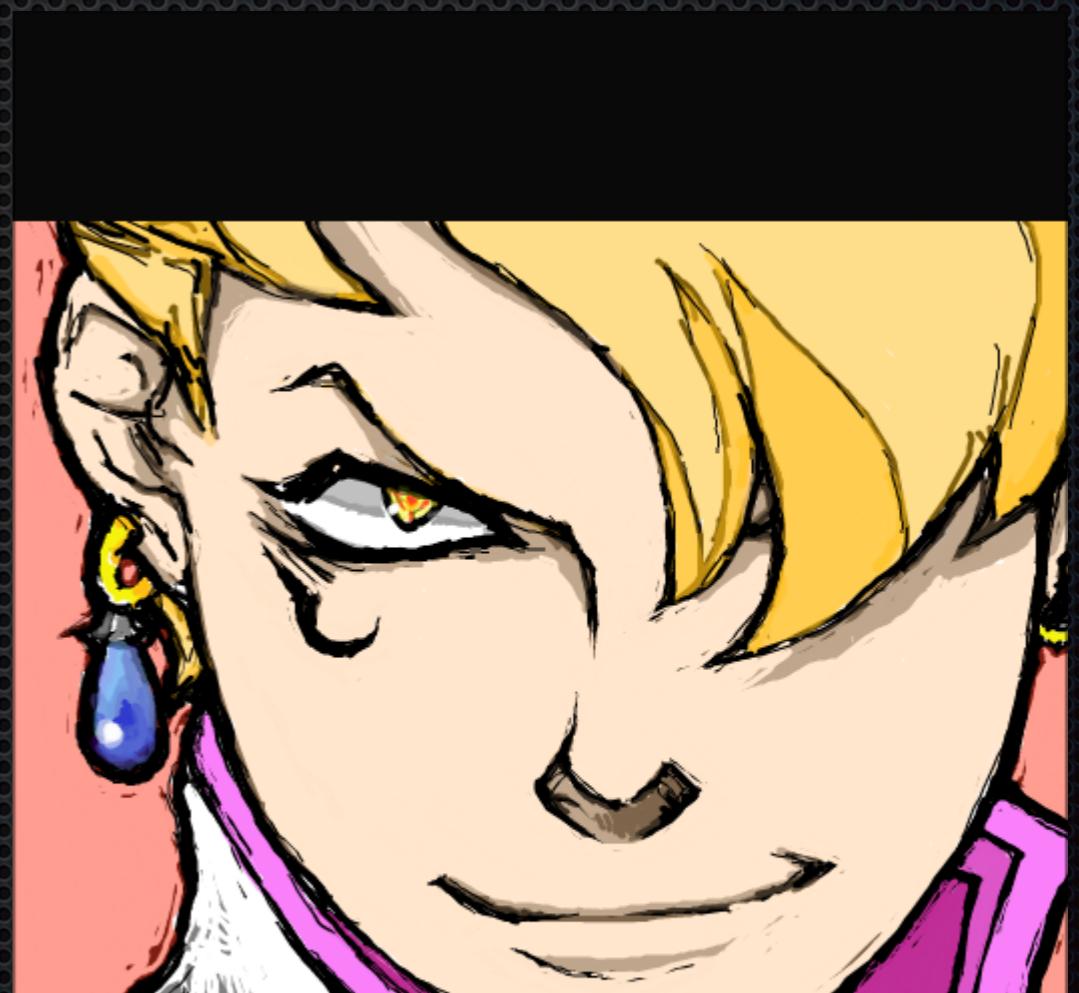
そして来たるべきWebGL 2.0へ向けて



はじめに
ご挨拶と本日のテーマ

自己紹介

- ハンドルネーム doxas
- wgl.org 運営
- WebGL 総本山 運営
- WebGL スクール主催



今日のテーマ

まず前半は

WebGL とモバイルウェブの 「これまで」と「これから」

- 「これまで」の WebGL 実装や、そのモバイル環境における位置づけがどのようなものだったか
- 「これから」のモバイルとウェブ、そしてモバイル環境における WebGL の立ち位置はどのように変化していくのか

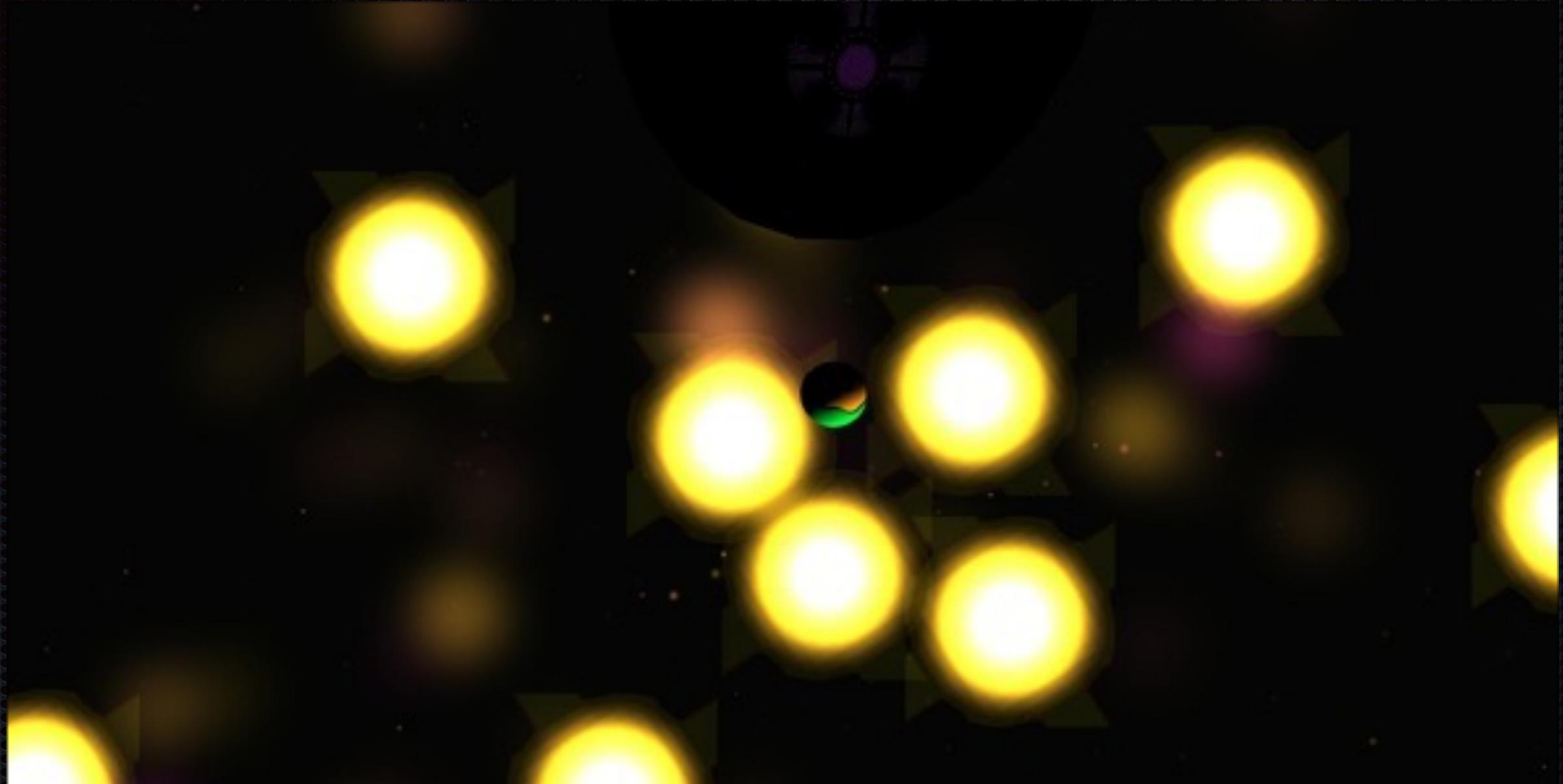
後半は.....

そして来るべき WebGL 2.0へ向けて

- 次世代の規格となる WebGL 2.0 が登場することによってこれまでとなにが変わるのが
- WebGL 2.0 が登場することを踏まえて、今何を見据えるべきなのか

対象受講者

- WebGL に興味がある（実装経験問わず）
- 3D プログラミングに興味がある
- モバイル環境における WebGL の現状が知りたい
- 今後の WebGL がどうなっていくのか気になる
- あんまりガチな 3D 談義はしません（笑）
- WebGL について概要を知る、というつもりで、楽な気持ちでご覧いただければと思います



WebGL とは？
そもそもそいつは何者なのか

WebGL

WebGL

- ブラウザからネイティブアプリケーションのように低レベルな 3D API を利用できるため、GPU を活用した高速な描画処理を行うことができる JavaScript の API
- あくまでも JavaScript の API なので、ブラウザの実装により挙動や性能が異なる
- モバイル向けの軽量 OpenGL 実装である OpenGL ES がベースになっている

WebGL

- OpenGL ES を使ったことがあれば、ほとんど同じような感覚で実装を行うことが可能
- 言い換えると、WebGL を通じて得た知識や技術は、ネイティブアプリケーション向けの API を扱う上でもけして無駄にはならない
- ネイティブアプリ開発にもつながっていく可能性がありつつ、ブラウザとテキストエディタさえあれば気軽に開発が始められるため、3D プログラミングの学習や入門に最適

どこまでできる？

どこまでできる？

- ・ とは言え所詮はブラウザの API に過ぎないわけで、大したことではないのでは？
- ・ ブラウザを中継するわけだからパフォーマンスが著しく落ちるのでは？

どこまでできる？

- 論より証拠、作例を見て実際に体感してみましょう

独断と偏見による

WebGL 作例紹介

APEXvj

- ◆ SoundCloud やローカル音源などを用いることができるサウンドビジュアライザ
- ◆ 描画パターンの種類の豊富さと演出のハイレベルさが際立つ

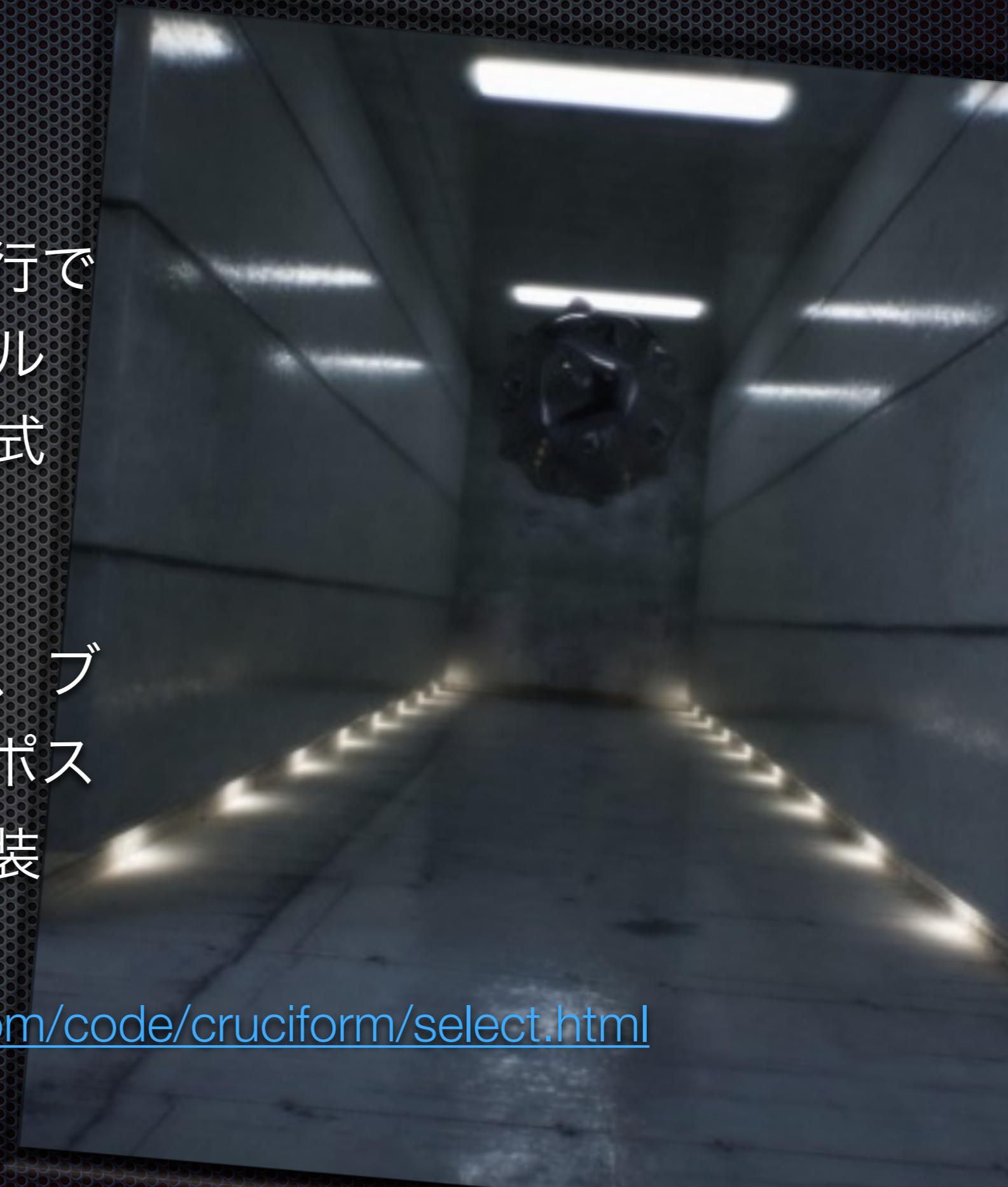
<http://www.apexvj.com/v3/>



Cru·Ci·form

- 様々なデバイスで実行できるように負荷レベルを最初に選択する方式を採用
- 被写界深度やノイズ、ブルームに FXAA などポストプロセスを多数実装

<http://www.clicktorelease.com/code/cruciform/select.html>



SpaceLamb

- ・ 全体的に丁寧に、しかし惜しみなく WebGL を使い倒していく印象のゲーム
- ・ ポストエフェクトはとても美しいが瞬間に FPS が下がる程度に容赦がない

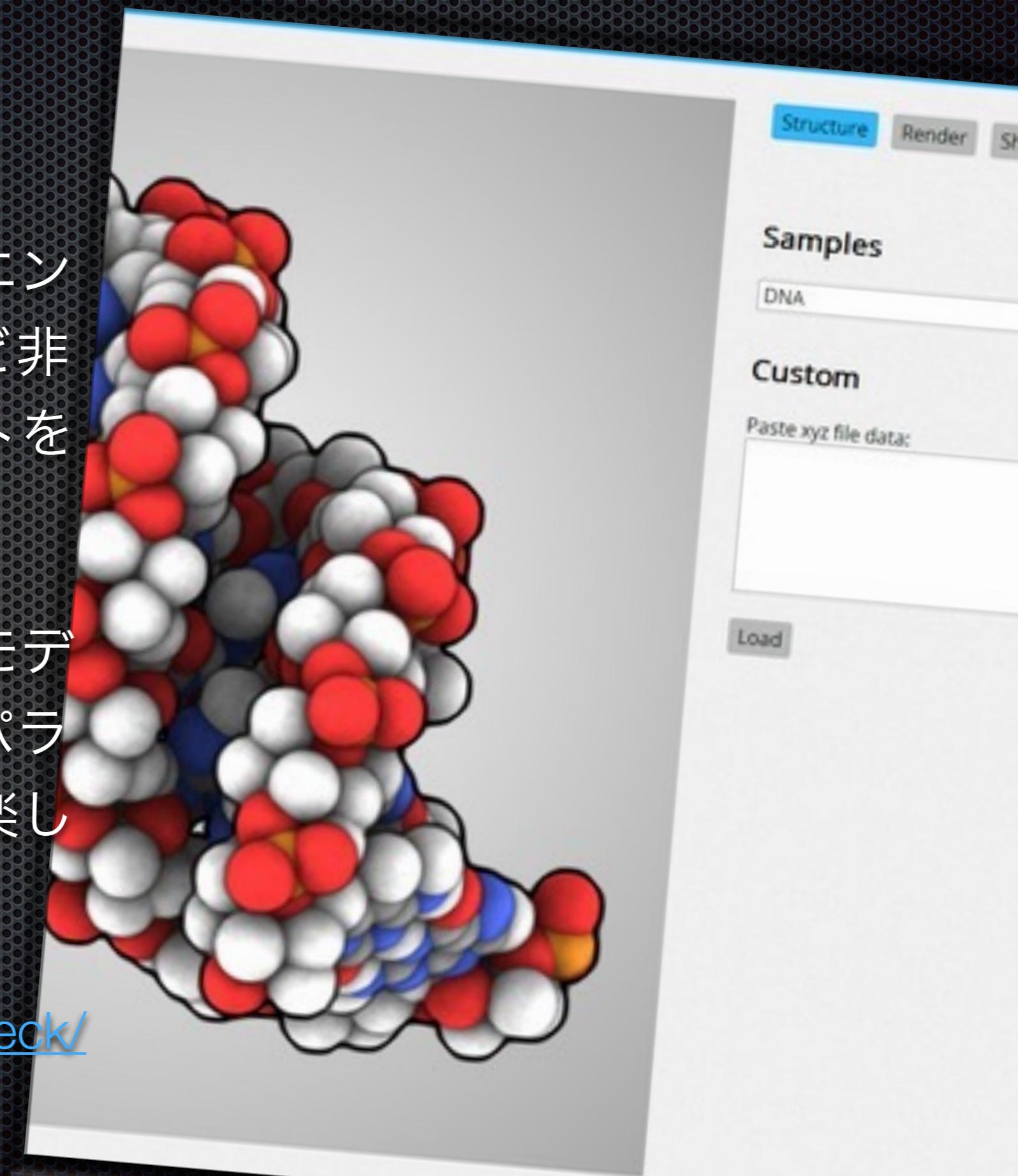
<http://spacelamb.12wave.com/>



Speck

- 被写界深度やアンビエントオクルージョンなど非常に多彩なエフェクトを持つ
- プリセットで様々なモデルが用意されていてパラメータいじるだけで楽しい

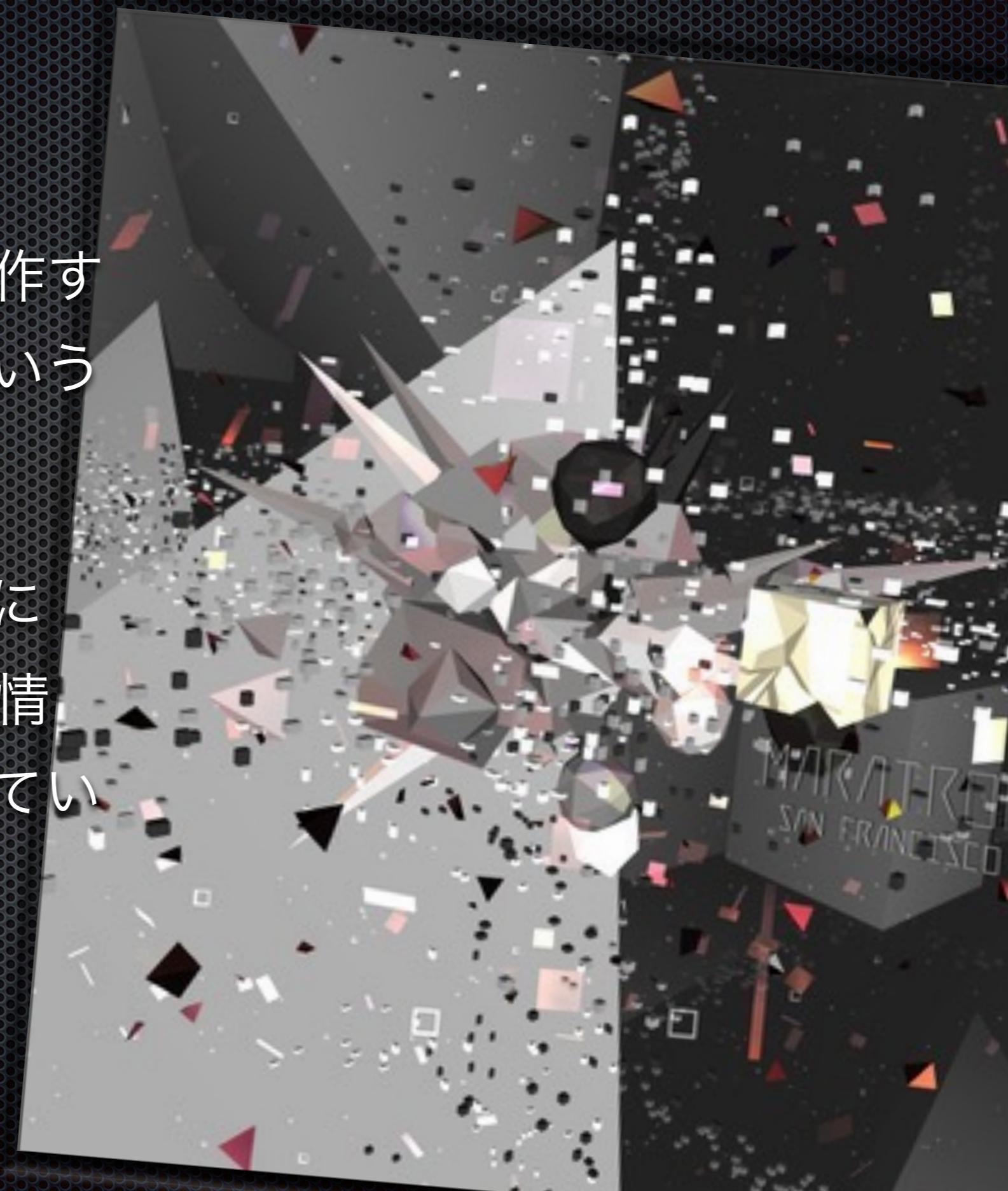
<http://wwwtyro.github.io/speck/>



Maratropa

- Twitter と連携して動作するウェブならではという印象の強いデモ作品
- 大量のオブジェクトに Twitter のアカウント情報などが紐付けられている

<http://maratropa.com/>



DENNIS

- ◆ すべてのモデルデータをプロシージャルに生成しており毎回異なるMVになる
- ◆ スマートフォンでの閲覧も考慮されておりジャイロで視点変更が可能

<http://www.dennis.video/>



要点をまとめると？

要点をまとめると？

- PC で動かす分にはかなり高コストな描画でも割と動かせる状況になってきている
- プログラマブルシェーダなので一般によく利用されているシェーダテクニックが使える
- Web ならではのインタラクティブなコンテンツや動的なオンラインデータの取得と相性が良い

モバイル環境では？

モバイル環境では？

- 対応コンテンツの数自体がまだ少ないものの、徐々にその数は増えてきている
- モバイル向けとなると特にハードウェア性能やブラウザ間での挙動や性能の差を吸収するのが大変
- iOS+Safari はかなりマシで、Android が組み込みブラウザのシェアが高いうえに、その種類がとても多いのでかなりカオス

モバイル環境では？

- 現状では残念ながら、対象を iOS8 以降を搭載した iPhone などに限定せざるを得ないケースが多い
- Android は 5.x 系であれば既定のブラウザが Chromium になっているので、工場出荷時の状態で WebGL を動作させられるが、なにせ日本では特にシェアが少ない傾向が顕著



モバイルと WebGL

その可能性と今後の展望とは

モバイルだからこそ？

モバイルだからこそ？

- モバイル環境では PC では扱うことのできないジャイロセンサーなどの多彩なセンサーを利用できる
- 現在はジャイロセンサーを活用した VR 的な利用法が比較的活発に研究されている印象

モバイルだからこそ？

- モバイル機器に備わっているセンサーはなにもジャイロセンサーばかりではない
- たとえば OS やブラウザを限定するものの、照度センサーが利用できれば周囲の明るさを WebGL の世界に持ち込んでライティングを行うなど、PC 環境とは異なる、ちょっと変わった活用方法も考えられる

モバイルだからこそ？

- ♦ でもそれって.....ネイティブアプリでやればいいことなんじゃないの？
- ♦ WebGL をあえて使う理由やメリットが果たしてあるの？

ネイティブアプリとWebGL

ネイティブアプリと WebGL

- ネイティブアプリにしかできないこと、というのは年々少なくなってきており、多くのことが javascript から行えるようになっている（WebView を利用したハイブリッドアプリなども最近は多い）
- WebGL を利用したリッチなコンテンツを、あたかもネイティブアプリのように動かせる環境は、現時点でもかなり整いつつある

WebGL アプリのメリット

WebGL アプリのメリット

- 実装の中心が WebGL であれば、バージョンアップやバグフィックスのたびにアプリの再申請を行うなどの手続きが必要なく、単にサーバー上の実装を置き換えてあげるだけでよい
- つまり WebGL の実装部分についてはユーザー側の作業としての更新作業は一切いらない
- 完全にウェブベースならインストールも不要でホーム画面にアイコン（ショートカット）を置くだけ

WebGL アプリのメリット

- ブラウザや GPU との相性問題はもちろんあるものの、javascript ベースなのでマルチプラットフォーム向けに開発が行える
- ほかの javascript API や CSS などとも連携できるため、たとえばインターフェース部分は CSS で構築しコアなレンダリングだけを WebGL で、なんてこともできる

WebGL アプリのデメリット

WebGL アプリのデメリット

- 端末による挙動のバラつきがかなり激しく、ブラウザの種類による違いが大きい
- つまりいわゆる普通のウェブページと同じようなクロスブラウザ対策が必要になる
- モバイルハードウェアでは純粹にパワー不足になりがちで、それ専用に最適化が必要
- バッテリーがモリモリ減る

WebGL アプリのデメリット

- 事前にアプリケーションとしてダウンロードやインストールといった作業は必要ない代わりに、実行するたびにリソースをダウンロードしなくてはならない
- データをキャッシュする系の技術を利用するという手もあるが、それを嫌がるユーザーは恐らく多い
- その辺り節度を持って実装しないと WebGL そのものが悪として認知されるという事態に.....



WebGL とどう向き合うか

モバイル全盛時代の WebGL 実装とは

WebGL のメリットを活かす

WebGL のメリットを活かす

- アプリケーションのアップデートやバグフィックスに対応しやすいというウェブベースのメリットを活用した、スピード感のある開発
- リアルタイム性の高い情報の表示に特化した WebGL の活用を（生活密着型の情報アプリなど）

WebGL のメリットを活かす

- javascript ベースの実装なのでフロントエンドやウェブ関連のスキルを転用しやすい特性を活かし、フロント寄りの人たちをさらにフルスタックの道へと追い込む（笑）
- 上記は冗談だが、スキルを転用できる上に WebGL 実装ならモバイルだけでなく PC 向けのページの実装にもほぼそのまま流用できる点は非常に大きなメリットであり、学習したことは無駄にならない

これからの WebGL

これからの WebGL

- もちろん PC 向けにハイエンドな実装を行う方向性はずっと残っていくが、ベースラインを「モバイルでも動作する」というところに据えれば、一度実装したものはほぼどんな環境でも動くことになる
- 数年のうちに対応 OS や対応ブラウザがほぼ全体を占めるようになり、ウェブにも、アプリにも、積極的に組み込んでいける唯一の、本当の意味でのマルチプラットフォームな存在へ

これからの WebGL

- ただし使うべきシーンは慎重に見極めてやる必要があり、リッチな演出が重要な意味を持つゲームなどの場合、もっと低レベルな部分を直接扱える専用の API を利用したほうがよい場合ももちろんある
- iOS なら Metal、Android なら Vulkan など
- PC 向けのウェブ実装なら WebAssembly の活用なども今後は面白い展開を見せそう

要点まとめ

モバイルウェブと WebGL

モバイルウェブと WebGL

- モバイル環境においても WebGL を使っていける環境は整いつつあり、今後もその流れは止まらない
- マルチプラットフォームを目指すなら有用な選択肢となり得る
- スピード感のある開発を目指すならウェブベースのモバイルアプリ開発に挑戦してみよう

モバイルウェブと WebGL

- 既に WebGL は物珍しいだけの存在では無くなりつつある。ただ WebGL を使えばいいということではなく、適材適所、WebGL のメリットを最大限活かした実装を
- ウェブベースの、勉強を始めやすい WebGL を活用してハイエンドグラフィックス技術を磨き、ウェブに、アプリに、転用できるスキルを身につけよう



WebGL 2.0

次世代 WebGL その正体とは？

要は何者 WebGL 2.0

要は何者 WebGL 2.0

- ❖ 文字通り、現行の WebGL 1.0 の次期バージョン
- ❖ OpenGL ES 3.0 をベースにしており、WebGL で利用するシェーダ言語である GLSL についても、これに倣う形で GLSL ES 3.0 に格上げされている

要は何者 WebGL 2.0

- OpenGL ES 3.0 は、OpenGL ES 2.0 への後方互換を残しているものの、GLSL ES 3.0 は、現行の 1.0 と比較するとほとんど別物というくらい変わっている
- WebGL 2.0 では GLSL ES 3.0 を使用すると明示的に宣言した場合を除いて、GLSL ES 1.0 で記述されるとみなす仕組みになっている

WebGL 2.0 の新機能

WebGL 2.0 の新機能

- **MRT (Multiple Render Targets)**

一度のドローコールで、複数のカラーバッファへレンダリングする。

遅延レンダリングなどの技術に利用されるもので現行の 3DCG シーンの多くで活躍している WebGL 2.0 の目玉とも言える新機能。

たった一度のドローコールだけで複数のカラーバッファに色と法線を別々に出力することなどが可能。

WebGL 2.0 の新機能

- **Transform feedback**

GPU から VBO に値を書き出すことができる。

これにより、GPU から VBO に書き出しそれを GPU で読み出す、ということが行えるようになり、CPU を介さずに GPU だけで演算を完結させることができる。

いわゆる GPGPU 的な使い方など、さまざまな GPU を駆使した処理に活用できる。

WebGL 2.0 の新機能

- **VAO (Vertex Array Object)**

WebGL 1.0 の頃から拡張機能として提供されていた機能が標準化。

VBO や IBO といったバッファ類のバインド作業を大幅に効率化させる機能で、まるで構造体のように複数のバッファをまとめてバインドしたり解除したりできるようになる。

WebGL 2.0 の新機能

■ Texture 関連

新しいフォーマットが大量に追加され、やっといっぽしのテクスチャ関連フォーマットが揃ったような格好に.....なると思われる。

3D テクスチャや Float テクスチャなどが既定の状態で利用できるようになれば、かなりできることの幅が広がる。

WebGL 2.0 の新機能

- その他にもたくさんの変更が

正方行列以外の行列を Uniform 変数として利用できたり、Uniform 変数を構造体化させたり、GLSL のバージョンが上がったことに対応できるよう、かなりたくさんの変更が掛かっている。

WebGL 1.0 の頃は拡張機能だったものがいくつか格上げされて標準に組み込まれている。

WebGL 2.0 で何が変わる？

WebGL 2.0 で何が変わる？

- OpenGL ES 3.0 ベースの機能が揃ったことで、ほんの少し現行の 3DCG API に近づいた
- OpenGL ES 3.0 がそうであるように、ジオメトリシェーダなどは利用できない
- 厳しい見方をすれば、当たり前のことがちょっとだけ新しく使えるようになっただけで、まだまだ時代遅れな実装とも言える

WebGL 2.0 で何が変わる？

- ♦ それでも WebGL 1.0 と比べればできることの幅が大きく広がり、さらに高品位なレンダリングを行うための基盤ができるることは間違いない
- ♦ デスクトップアプリケーション並のグラフィックスを、黒魔術的な実装をせずに再現できるようになった（深度値を 8 bit の RGBA に分割して焼くとか、そういう怪しいことしなくて良くなる！）

超個人的見解

WebGL が向かうその先

WebGL が向かうその先

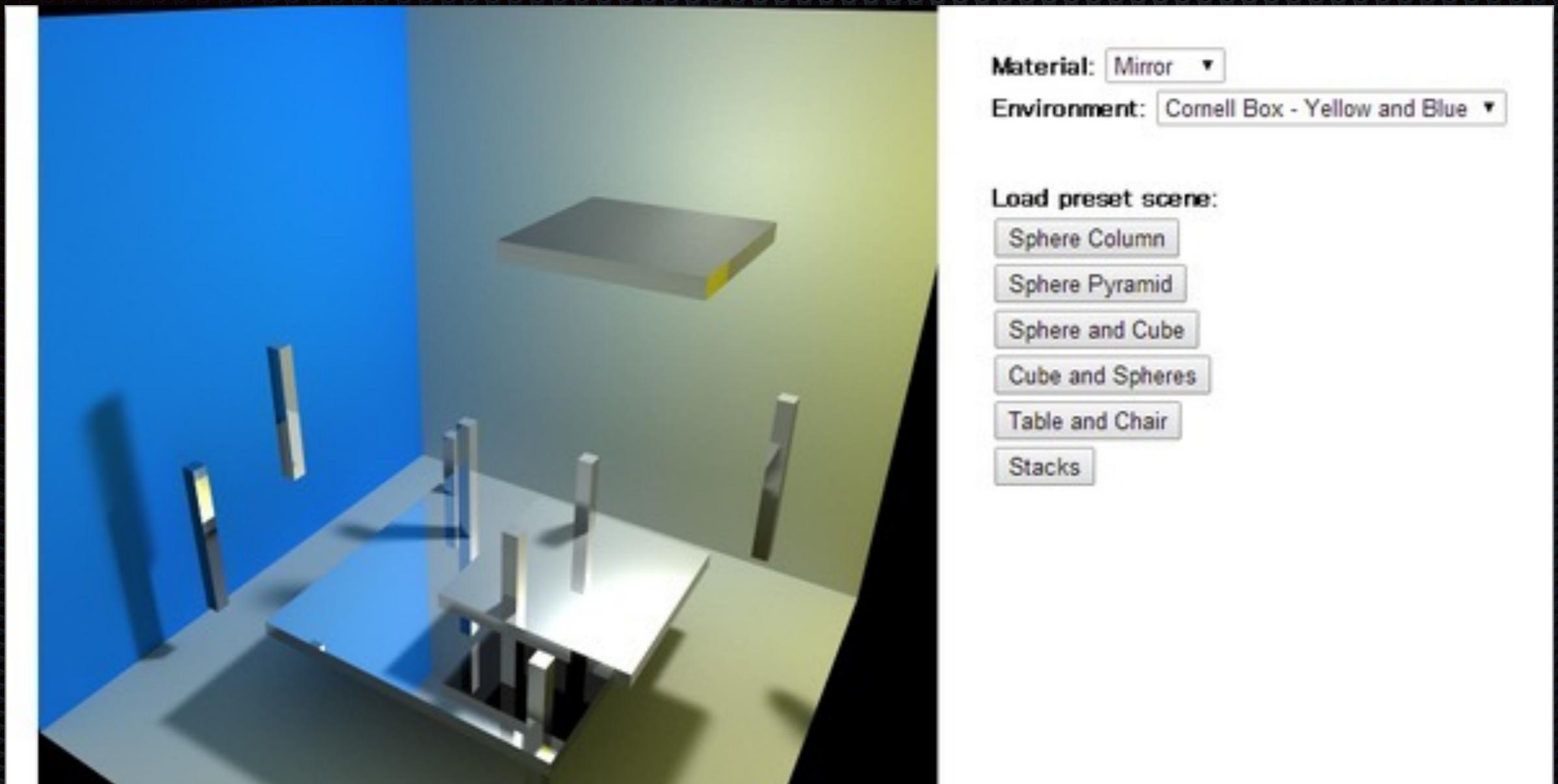
- Metal や Vulkan などの新しい API が登場している昨今、WebGL は今後どこへ向かっていくのか
- 当然、いつの日か、上記のようなより低レベルな API を操作できるようになっていくのだろうけれど、それはかなり先のことになるのではないかと予想
- それは 3DCG 畑のなかでも群を抜いて複雑なプラットフォームを持つウェブに生きる WebGL の宿命

WebGL が向かうその先

- 主要なゲームエンジンが HTML5 出力をサポートしている現在の状況から見ても、WebGL との付き合い方はなにもスクラッチ記述することだけではなくなっていくと予想（特にゲームなどのエンターテイメント系ジャンル）
- 少し時代遅れな実装であることが幸いし、WebGL で扱う技術の多くは既に利用されているリソースを参考にすることができるので勉強しやすい

WebGL が向かうその先

- WebGL は開発環境の構築の簡単さや、実行してすぐにブラウザ上で結果を確認できるメリットがあり、やはり依然として 3DCG を勉強するためのプラットフォームとしては間違いなく最強
- ゲームエンジン製の作品もいいが、リソース容量の問題やシェーダエフェクトのクオリティアップのために、スクラッチで記述できるスキルは必要であり続けるため勉強が無駄になることはない



最後に
WebGL、やらないか

なぜ WebGL にこだわるのか

なぜ WebGL にこだわるのか

- ウェブという開かれた世界で 3DCG API を叩いて高品質な 3D シーンをレンダリングできる幸せ
- エンターテイメントだけでなく、研究開発やプロモーション、メディアとしての表現など、あらゆるジャンルに広く利用できる技術であること
- なにより学習がしやすく、手軽であること

なぜ WebGL にこだわるのか

- コンシューマ開発など、技術の漏洩などの観点から情報をオープンにすることが難しい分野があることはもちろん理解しているけれど、ウェブにもっともっと有用な 3D 技術を広げていきたい
- 3DCG に出会う若い才能を増やしていくことが結果的に業界全体を底上げしていくはずだし、そのための取っ掛かりとして、学習を始めやすい WebGL が果たせる役割は大きい信じている

なぜ WebGL にこだわるのか

- フロントエンドの人たちにも新しいジャンルに挑戦してほしいし、逆に、今までウェブでは活躍する土俵が無かった人たちにも、オープンなウェブで活躍していってほしい
- 難しい問題はたくさんあるが、初めて 3DCG を描画できたときのあの嬉しい気持ちや感動した経験を、ひとりでも多くの人に感じてほしいからこそ、これからも WebGL を推し広めていく所存です

WebGL やらないか

WebGL、やりましょう！

ありがとうございました！