第 5 回

3Dと
バーチャルリアリティ

現代のWebやゲームに3DCGは欠かせません。
Webアプリで3DCGを扱う方法に加えて、
スマホのセンサと組み合わせたバーチャルリアリティの
プログラミングを学びます。

フィジカルコンピューティング演習
愛知淑徳大学 人間情報学部　　國分 三輝

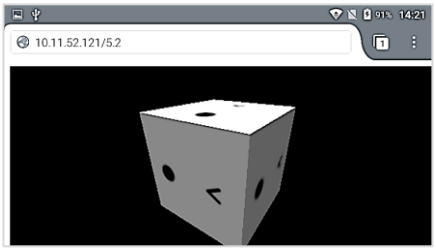
✈ 5.0 3D とバーチャルリアリティ

現代の Web やゲームに 3DCG は欠かせません。さらに市販の立体視ゴーグルやスマホのセンサを組み合わせると、話題のバーチャルリアリティ（Virtual Reality; VR）が割と簡単に実現できます。Web アプリで 3DCG を扱う方法に加えて、VR アプリの作り方を学びましょう。

5.1 JavaScript で 3DCG

➔ 3 ページ

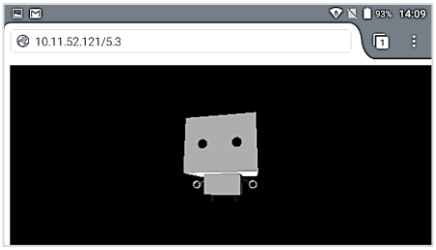
Web アプリで 3DCG を扱うにはブラウザの WebGL という機能を使います。three.js という外部ライブラリを使うことで JavaScript から WebGL にアクセスできます。二つのサンプルを通じて、three.js による 3DCG プログラミングの基礎を学びます。



5.2 外部 CG モデルの読み込み

➔ 9 ページ

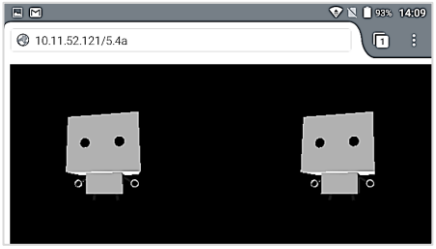
複雑な CG モデルは一般的に専門の 3DCG ソフトで作ります。ここでは、JSON 形式の CG モデルを読み込んで表示させる方法を学びます。



5.3 VR ゴーグルで立体視

➔ 11 ページ

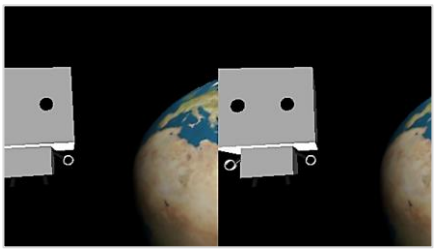
スマホを使って立体視ができるゴーグルが安価で市販されています。サイドバイサイドという方式を使ったスマホによる立体視の方法を学びます。



5.4 スマホで VR

➔ 13 ページ

ゴーグルによる立体視に加えて、スマホのジャイロセンサを使うと、3DCG の仮想空間に没入できる VR が実現できます。三つのサンプルを通してスマホで VR を実現する方法を学びます。



本日の課題

➔ 20 ページ

✈ 5.1a JavaScript で 3DCG

	PC	And	iOS
Firefox	○	○	○
Chrome	○	○	○

Web アプリで 3DCG を扱うにはブラウザの WebGL という機能を使います。three.js という外部ライブラリを使うことで JavaScript から WebGL にアクセスできます。二つのサンプル (5.1a/5.1b) を通じて、three.js による 3DCG プログラミングの基礎を学びます。

Input HTML と JavaScript を入力

- **C:￥xampp￥htdocs￥5.1a** というフォルダを作り、中に 1.1 の index.html (テンプレート) をコピーします。この index.html を Brackets で開き、以下のコードを入力しましょう。
- CampusSquare から「PHC_05_素材」をダウンロードし、「5.1a 素材」の中にある'three.js'を 5.1a フォルダに入れましょう。

※ ①や②などの丸数字は打ち込まないこと
　コメントは入力必須ではありません。

C:￥xampp￥htdocs￥**5.1a**￥ **index.html**

```
( 略 )
    <head>
        <meta charset='utf-8'>
        <meta name='viewport' content='width=device-width,initial-scale=1'>
        <title>5.1a JavaScript で 3DCG</title>
    </head>
    <body>
        <!-- 3D 空間を表示させる場所としての div 要素 -->
        <div id='screen'></div>
        <!-- JavaScript で 3D を扱うための three.js ライブラリの読み込み -->
        <script src='three.js'></script>                                    ①

        <script>
            // 描画する画面サイズを定義
            const width = window.innerWidth;    // 画面の横幅 ( ウィンドウの横幅にする )
            const height = window.innerHeight;  // 画面の高さ ( ウィンドウの高さにする )
            // シーン ( 3D 空間 ) を作成 ( 座標系は右手系 Y アップ )
            const scene = new THREE.Scene();                                    ②
            const axis = new THREE.AxisHelper(100);   // 座標軸オブジェクト ( 任意 )    ③
            scene.add(axis);                          // 座標軸をシーンに追加 ( 任意 )
            // シーンにカメラを置く
            const fov = 60;                    // 画角(field of view) ( 度 )
            const aspect = width / height;     // アスペクト比
            const near = 1;                    // ニアークリップ ( これより近い領域を非表示 )
            const far = 1000;                  // ファークリップ ( これより遠い領域を非表示 )
            const camera = new THREE.PerspectiveCamera(fov, aspect, near, far); //カメラ④
            camera.position.set(0, 0, 50);  // カメラの位置を指定
            scene.add(camera);              // カメラを追加
                                    // 次のページに続きます
```

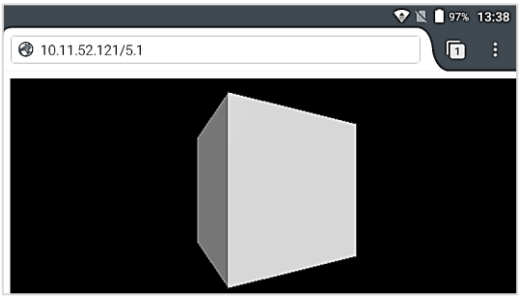
```
        // シーンに光源（照明）を置く
        const light = new THREE.DirectionalLight(0xffffff);   // 光源を作成          ⑤
        light.position.set(0, 50, 50);  // 光源の位置を指定
        scene.add(light);               // 光源を追加
        // シーンに 3D 物体（メッシュ）を置く                                          ⑥
        const geometry = new THREE.BoxGeometry(30, 30, 30); //形を指定              ⑦
        const material = new THREE.MeshPhongMaterial();     //質感を指定            ⑧
        const mesh = new THREE.Mesh(geometry, material);    //3D 物体を作成         ⑨
        mesh.position.set(0, 0, 0); //3D 物体を置く位置
        mesh.rotation.y = 0.5;      //真正面だと立体に見えないので Y 軸まわりに回す
        scene.add(mesh);            //3D 物体を追加
        // レンダラ（3D 空間を描画する機能）を準備
        const renderer = new THREE.WebGLRenderer();       // レンダラを作成        ⑩
        renderer.setSize(width, height);                 // レンダラのサイズを指定
        renderer.setClearColor(0xEEEEEE, 1.0);           // 背景色と不透明度の設定
        const screen = document.getElementById('screen'); // div 要素を取得
        screen.appendChild(renderer.domElement);         // div 内にレンダラを置く  ⑪
        // レンダリング（描画）を実行
        renderer.render(scene, camera);                                             ⑫
    </script>
（以下略）
```

Check スマホで動作確認

- Web サーバをスタートさせ、アクセス用のアドレスを調べる
  - **XAMPP Control Panel** を開き、Apache を Start
  - **コマンドプロンプト** を開き、**ipconfig** して **IPv4 アドレス** をメモする

メモ
★　　.　　.　　.

- スマホの **Firefox** で確認
  - メモしたアドレス（★）に続けて「/5.1a/」と入力して Enter
    アドレス例： **10.11.123.45/5.1a/**
  - スマホは横向き



Learn 解　説

① **`<script src='three.js'></script>`**

<script>タグに src 属性を指定して外部ライブラリを読み込みます。index.html と同じフォルダに置いた three.js ファイルを読み込んでいます。

three.js による CG プログラミングは一般的に、以下の順序で進めます。

- **シーン**（3D 空間）の作成
- シーンに**カメラ**を置く
- シーンに**光源**を置く
- シーンに**メッシュ**（3D 物体）を置く
- **レンダラ**（描画機能）を準備
- **レンダリング**（描画）を実行

② **`const scene = new THREE.Scene();`**

CG を表示するシーン（3D 空間）を作る、THREE.Scene オブジェクトです。three.js はこのように、THREE.Xxxxxx() のようにして CG の様々なオブジェクトを作ったり操作したりします。

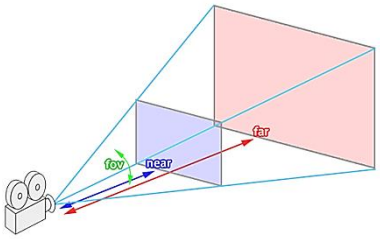
③ **`const axis = new THREE.AxisHelper(100);`**

座標軸を表示させる THREE.AxisHelper オブジェクトです。これ以下の 2 行は必須ではなく、理解のために座標軸を表示させています。X 軸（左右）が赤、Y 軸（上下）が緑、Z 軸（前後）が青の線で表示されます。次の `scene.add(axis);` で座標軸をシーンに追加しています。

④ **`const camera = new THREE.PerspectiveCamera(fov, aspect, near, far);`**

シーン内を撮影するカメラを作る THREE.PerspectiveCamera()です。パラメータはそれぞれ…

fov:　視野角（field of view)、単位は度[degree]
aspect: 縦横比
near:　どこまで近くを撮影するか
far:　どこまで遠くを撮影するか



`scene.add(camera);` でシーンにカメラを追加します。

PerspectiveCamera 以外のカメラも使えますが、これだけ知っていれば問題ありません。

⑤ **`const light = new THREE.DirectionalLight(0xffffff);`**

シーンを照らす光源（照明）を作る THREE.DirectionalLight()です。パラメータは光源の色を 16 進数で指定します。0xffffff は白色です。`scene.add(light);` でシーンに照明を追加します。

DirectionalLight は平行光源です。これら以外にも、AmbientLight（環境光源）、PointLight（点光源）、SpotLight（スポットライト）、HemisphereLight（半球光源）などが使えます。それぞれ指定するパラメータが違うので、使う場合は自分で調べてみましょう。

光源はシーンにいくつも追加でき、光源の種類や光の当て方で 3D 物体の見え方や陰の付き方が大きく変わります。

⑥ シーンに 3D 物体（メッシュ）を置く

決まった形の 3D 物体をシーン内に置きます（他のソフトで作ったより複雑な CG モデルを置く方法は 5.2 で紹介します）。

3D 物体を置くには、「ジオメトリ（形）」というオブジェクトと、「マテリアル（質感）」というオブジェクトを作り、それらを合わせて「メッシュ」というオブジェクトを作る、という流れでコードを書きます。

⑦ `const geometry = new THREE.BoxGeometry(30, 30, 30);`

ジオメトリオブジェクトを作っています。BoxGeometry は直方体で、パラメータは(幅，高さ，奥行き)です。これ以外にも、SpereGeometry(球)、CylinderGeometry(円柱や円すい)、TorusGeometry（ドーナツ型）、PlaneGeometry（平面）など、数多くの種類があります。それぞれ指定するパラメータが違うので、使う場合は自分で調べてみましょう。

⑧ `const material = new THREE.MeshPhongMaterial();`

マテリアルオブジェクトを作っています。MeshPhongMaterial は「フォン反射材質」という質感を表現します（CG 用語なのでここでは解説はしません）。他にも様々な材質を使えますが、通常はこれを覚えておけばよいでしょう。( )内に様々なパラメータを指定して質感を調整できますが、必要に応じて調べてみてください。

⑨ `const mesh = new THREE.Mesh(geometry, material);`

⑦で作ったジオメトリ（形）と⑧で作ったマテリアル（質感）を持ったメッシュオブジェクトを作っています。少し面倒ですが、これでようやく 1 個の 3D 物体のできあがりです。最後に `scene.add(mesh);` でシーン内に 3D 物体を追加します。

⑩ `const renderer = new THREE.WebGLRenderer();`

3D 空間を描画するレンダラオブジェクトを作っています。シーンを作り、カメラ・光源・3D 物体を置いただけでは画面に何も表示されません。レンダラオブジェクトを作り、描画のしがたや描画自体の指示をしてやる必要があります。

⑪ `screen.appendChild(renderer.domElement);`

⑩で作ったレンダラオブジェクトを、HTML の div 要素の子要素として置いています。appendChild は「(　)内の要素を子要素として追加する」という意味です。domElement は、形のないレンダラオブジェクトに形を与えて、ユーザの目に見えるようにする命令です。

⑫ `renderer.render(scene, camera);`

最後に、レンダラオブジェクトに描画を指示しています。( )内にシーンオブジェクトとカメラオブジェクトを指定します。「このシーンをこのカメラで撮影して描画せよ」という意味です。

✈ 5.1b テクスチャとアニメーション

	PC	And	iOS
Firefox	○	○	○
Chrome	○	○	○

5.1a では白い箱のメッシュが出るだけでした。このメッシュにテクスチャを貼って模様をつけ、さらにメッシュを動かして少しだけアニメーションっぽくしてみましょう。

Input HTML と JavaScript を入力

- **C:￥xampp￥htdocs￥5.1a をフォルダごと複製**して、**フォルダ名を 5.1b に**します。
- index.html を Brackets で開き、以下のコードを入力しましょう。**変更は網掛けの部分**です。
- 「5.1b 素材」の中にある以下の 2 つのファイルを 5.1b フォルダに入れましょう。

'three.js'　'huchan.jpg'

※ ①や②などの丸数字は打ち込まないこと
　コメントは入力必須ではありません。

C:￥xampp￥htdocs￥**5.1b**￥ **index.html**

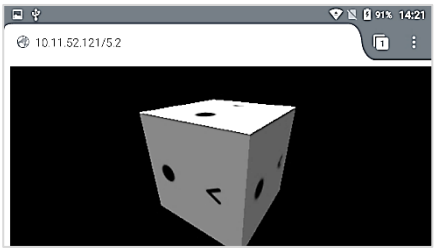
```
（略）
        <title>5.1b テクスチャとアニメーション</title>
    </head>
    <body>
        <div id='screen'></div>
        <script src='three.js'></script>

        <script>
            // 描画する画面サイズを定義
                    :
                （中略）
                    :
            // シーンに 3D 物体（Mesh）を置く
            const geometry = new THREE.BoxGeometry(30, 30, 30);   // 形を指定
            const loader = new THREE.TextureLoader();       // テクスチャを読む準備     ①
            const texture = loader.load('huchan.jpg');      // テクスチャを読み込む     ②
            const material = new THREE.MeshPhongMaterial({  // 質感を指定               ③
                    color: 0xffffff,    // 表面色の指定
                    map: texture        // 貼り付けるテクスチャの指定
            });
            const mesh = new THREE.Mesh(geometry, material);  // 3D 物体を作成
            mesh.position.set(0, 0, 0); //3D 物体を置く位置
            mesh.rotation.y = 0.5;      // 真正面だと立体に見えないので Y 軸まわりに少し回す
            scene.add(mesh);            // 3D 物体を追加
            // レンダラ（3D 空間を描画する機能）を準備
                    :
                （中略）
                    :
            screen.appendChild(renderer.domElement);        // div 内にレンダラを置く
                                    // 次のページに続きます
```

```
        // レンダリング（描画）を繰り返し実行（アニメーション）する関数
        function renderScene() {                                          ④
            mesh.rotation.y += 0.01;                  // y 軸まわりに少し回す
            mesh.rotation.x += 0.01;                  // x 軸まわりに少し回す
            renderer.render(scene, camera);           // レンダリングを実行
        }
        // レンダリング実行関数を一定間隔で繰り返し実行
        window.setInterval(renderScene, 33);                              ⑤
    </script>
（以下略）
```

Check スマホで動作確認

- スマホの Firefox で確認
  - アドレス例： **10.11.123.45/5.1b/**



Learn 解　説

① **const loader = new THREE.TextureLoader();**

テクスチャ画像を読み込む THREE.TextureLoader オブジェクトを作ります。

② **const texture = loader.load('huchan.jpg');**

①で作った TextureLoader で画像ファイルを読み込みます。

③ **const material = new THREE.MeshPhongMaterial(…);**

5.1a ではこの MeshPhongMaterial の( )内に何も指定しませんでしたが、今回は中括弧{ }の中で color と map という属性を指定しています。color はメッシュの表面の色、map はテクスチャです。map に②で読み込んだテクスチャを指定することで、メッシュにテクスチャを貼り付けられます。

見やすいようにサンプルコードでは改行して 6 行に渡っていますがが、プログラム的にはこれで 1 行の命令です。

④ **function renderScene() { … }**

メッシュを少し動かして描画、また少し動かして描画…を繰り返すことで、アニメーションさせることができます。この「少し動かして描画」を行う関数を定義しておきます。

⑤ **window.setInterval(renderScene, 33);**

前にも出た、一定間隔で関数を実行させる setInterval(関数, 実行間隔[ミリ秒])です。今回は 33 ミリ秒間隔で（つまり 1 秒間に約 30 回）④を実行して、アニメーションさせています。

✈ 5.2 外部 CG モデルの読み込み

	PC	And	iOS
Firefox	○	○	○
Chrome	○	○	○

複雑な CG モデルは一般的に専門の 3DCG ソフトで作ります。ここでは、JSON 形式※の CG モデルを読み込んで表示させる方法を学びます。

※ JavaScript Object Notation の略。JavaScript で最も扱いやすい形式です。無償の CG ソフト Blender などで作ることができます。three.js ではより一般的な OBJ 形式などの CG モデルも読み込めますが、JSON 形式を使うのが最も簡単です。Blender や JSON 形式モデルの作り方は、必要に応じて自分で調べてみましょう。

Input HTML と JavaScript を入力

- **C:￥xampp￥htdocs￥5.1b をフォルダごと複製**して、**フォルダ名を 5.2 に**します。
- index.html を Brackets で開き、以下のコードを入力しましょう。**変更は網掛けの部分**です。
- 「5.2 素材」の中にある以下の 2 つのファイルを 5.2 フォルダに入れましょう。

'three.js'　　'huchan.json'

※ ①や②などの丸数字は打ち込まないこと
　コメントは入力必須ではありません。

C:￥xampp￥htdocs￥**5.2**￥ **index.html**

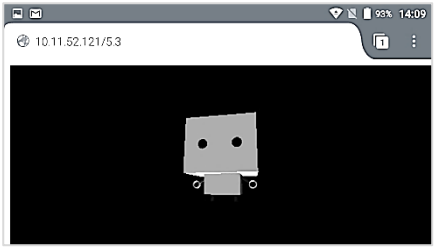
```
（略）
        <title>5.2 外部 CG モデルの読み込み</title>
    </head>
    <body>
        <div id='screen'></div>
        <script src='three.js'></script>
        <script>
            // 描画する画面サイズを定義
                    :
                （中略）
                    :
            // シーンに光源（照明）を置く
            const light = new THREE.DirectionalLight(0xffffff);   // 光源を作成
            light.position.set(0, 50, 50);  // 光源の位置を指定
            scene.add(light);               // 光源を追加
            // シーンに 3DCG モデルを置く（「シーンに 3D 物体を置く」の部分を以下のように変更）
            let mesh;                                          // モデルを入れる箱          ①
            const loader = new THREE.JSONLoader();             // モデルを読む準備          ②
            loader.load('huchan.json', addModel);              // モデルを読み込む          ③
            function addModel(geometry, material) {            // 読み込み完了時の処理
                const faceMat = new THREE.MeshFaceMaterial(material); // 質感             ④
                mesh= new THREE.Mesh(geometry, faceMat);       // 3D 物体を作成             ⑤
                mesh.position.set(0, 0, 0);                    // モデルを置く位置
                mesh.scale.set(10, 10, 10);                    // モデルの大きさ            ⑥
                scene.add(mesh);                               // モデルを追加
            }
```

// 次のページに続きます

```
            // レンダラ（3D 空間を描画する機能）を準備
                  :
               （中略）
                  :
            // レンダリング実行関数を一定間隔で繰り返し実行
            window.setInterval(renderScene, 33);
        </script>
（以下略）
```

Check スマホで動作確認

- スマホの Firefox で確認
  - アドレス例： **10.11.123.45/5.2/**



Learn 解　説

① **let mesh;**

mesh の中身は後で決める（mesh に値を再代入する）ので、const ではなく let で定義します。

② **const loader = new THREE.JSONLoader();**

JSON 形式のファイルを読み込む THREE.JSONLoader オブジェクトです。

③ **loader.load('huchan.json', addModel);**

②の JSONLoader を使って CG モデルを読み込んでいます。( )内には、モデルのファイル名と、モデルの読み込みが完了したときに実行する関数（ここではすぐ下の addModel）を指定します。

④ **const faceMat = new THREE.MeshFaceMaterial(material);**

モデルの質感の設定です。5.1 では MeshPhongMaterial という設定を使いましたが、これだと、せっかく CG ソフトで設定した質感（色やテクスチャ）が損なわれてしまいます。MeshFaceMaterial という質感の設定を使うと、CG ソフトで設定した質感のまま表示できます。

⑤ **mesh = new THREE.Mesh(geometry, faceMat);**

ジオメトリ（形）とマテリアル（質感）を指定してメッシュを作るという手順は、定形のモデルでも、外部 CG モデルでも、同じです。

⑥ **mesh.scale.set(10, 10, 10);**

CG ソフトで作ったときのサイズ感と、three.js で表示されるサイズは異なります。試しながら拡大縮小させて適当なサイズに調整します。( )内は（幅，高さ，奥行き）です。

✈ 5.3 VR ゴーグルで立体視

	PC	And	iOS
Firefox	○	○	○
Chrome	○	○	○

スマホを使って立体視ができるゴーグルが安価で市販されています。サイドバイサイドという方式※を使ったスマホによる立体視の方法を学びます。

※ 画面を左右に分割して、画面の左側に左目用の映像、右側に右目用の映像を表示する立体視の方式。他の方式に比べてコンテンツの作成や管理、機器の構成が簡単にできるため、Youtube の 3D 動画や、多くの VR ゴーグルがこの方式を採用しています。

Input HTML と JavaScript を入力

- **C:￥xampp￥htdocs￥5.2 をフォルダごと複製**して、**フォルダ名を 5.3 に**します。
- index.html を Brackets で開き、以下のコードを入力しましょう。**変更は網掛けの部分**です。
- 「5.3 素材」の中にある以下の 3 つのファイルを 5.3 フォルダに入れましょう。

'three.js'　'huchan.json'　'StereoEffect.js'

※ ①や②などの丸数字は打ち込まないこと
　コメントは入力必須ではありません。

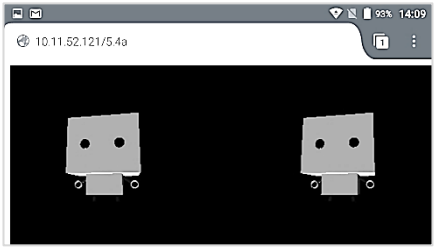
C:￥xampp￥htdocs￥**5.3**￥ **index.html**

```
（略）
        <title>5.3 VR ゴーグルで立体視</title>
    </head>
    <body>
        <div id='screen'></div>
        <script src='three.js'></script>
        <!-- サイドバイサイド立体視を可能にする StereoEffect.js ライブラリの読み込み -->
        <script src='StereoEffect.js'></script>                                    ①
        <script>
            // 描画する画面サイズを定義
                    :
                （中略）
                    :
            // レンダラ（3D 空間を描画する機能）を準備
            const renderer = new THREE.WebGLRenderer();       // レンダラを作成
            renderer.setSize(width, height);                  // レンダラのサイズを指定
            renderer.setClearColor(0xEEEEEE, 1.0);            // 背景色と不透明度の設定
            const screen = document.getElementById('screen'); // div 要素を取得
            screen.appendChild(renderer.domElement);          // div 内にレンダラを置く
            // レンダラをステレオ化する
            const stereo = new THREE.StereoEffect(renderer); //ステレオ効果オブジェクト ②
            stereo.setSize(width, height);                    // ステレオレンダラのサイズ
            // レンダリング（描画）を繰り返し実行（アニメーション）する関数
            function renderScene() {
                mesh.rotation.y += 0.01;                      // y 軸まわりに少し回す
                mesh.rotation.x += 0.01;                      // x 軸まわりに少し回す
                                    // 次のページに続きます
```

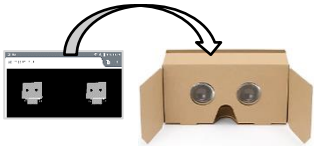
```
            //renderer.render(scene, camera);          // この行はコメントアウトする
            stereo.render(scene, camera);              // ステレオでレンダリング実行 ③
        }
        // レンダリング実行関数を一定間隔で繰り返し実行
        window.setInterval(renderScene, 33);
    </script>
( 以下略 )
```

Check スマホで動作確認

- スマホの Firefox で確認
  - アドレス例： **10.11.123.45/5.3/**



- スマホを VR ゴーグルに入れて覗いてみる



Learn 解　説

① **`<script src="StereoEffect.js"></script>`**

three.js でサイドバイサイド立体視を簡単に実現する'StereoEffect.js'というライブラリを読み込んでいます。

② **`const stereo = new THREE.StereoEffect(renderer);`**

5.2 まで使ってきたレンダラオブジェクトを、THREE.StereoEffect オブジェクトに渡してやるだけで、サイドバイサイドになって (左目用の映像と右目用の映像を作ってくれて、左右に表示されるように合成されて) 戻ってくるという、超簡単な仕組みです。

③ **`stereo.render(scene, camera);`**

通常の renderer (THREE.WebGLRenderer) の代わりに、stereo レンダラ (THREE.StereoEffect) を使って描画させています。これだけで、サイドバイサイド立体視のできあがりです。

✈ 5.4a スマホで VR

	PC	And	iOS
Firefox	×	○	○
Chrome	×	○	○

ゴーグルによる立体視に加えて、スマホのセンサを使うと、3DCG の仮想空間に没入できる VR が実現できます。三つのサンプルを通して、スマホで VR を実現する方法を学びます。

Input HTML と JavaScript を入力

- **C:￥xampp￥htdocs￥5.3 をフォルダごと複製**して、**フォルダ名を 5.4a に**します。
- index.html を Brackets で開き、以下のコードを入力しましょう。**変更は網掛けの部分**です。
- 「5.4a 素材」の中にある以下の 4 つのファイルを 5.4a フォルダに入れましょう。

'three.js'　'huchan.json'　'StereoEffect.js'
'DeviceOrientationControls.js'

※ ①や②などの丸数字は打ち込まないこと
　コメントは入力必須ではありません。

C:￥xampp￥htdocs￥5.4a￥ index.html

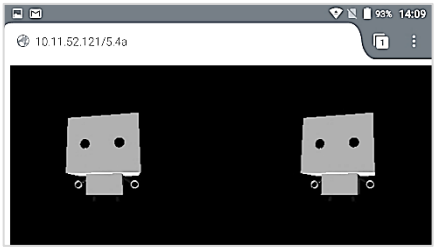
```
（略）
        <title>5.4a スマホで VR</title>
    </head>
    <body>
        <div id='screen'></div>
        <script src='three.js'></script>
        <script src="StereoEffect.js"></script>
        <!-- スマホの動きで VR する DeviceOrientationControls.js ライブラリの読み込み -->
        <script src="DeviceOrientationControls.js"></script>                    ①
        <script>
            // 描画する画面サイズを定義
                   :
               （中略）
                   :
            // レンダラをステレオ化する
            const stereo = new THREE.StereoEffect(renderer);  // ステレオ効果オブジェクト
            stereo.setSize(width, height);                    // ステレオレンダラのサイズ
            // スマホのセンサで視点を操作
            // ジャイロセンサの deviceorientation イベントを定義
            window.addEventListener('deviceorientation', gyroControl);          ②
            // deviceorientation イベントが発生した時に呼ばれる関数
            function gyroControl(event) {
                // コントローラで camera オブジェクトの向きを操作する
                const ctrl = new THREE.DeviceOrientationControls(camera, true);  ③
                ctrl.connect(); // コントローラに接続                            ④
                ctrl.update();  // コントローラの情報を更新
            }
                                    // 次のページに続きます
```

```
            // レンダリング（描画）を繰り返し実行（アニメーション）する関数
            function renderScene() {
                mesh.rotation.y += 0.01;                    // y 軸まわりに少し回す
                mesh.rotation.x += 0.01;                    // x 軸まわりに少し回す
                stereo.render(scene, camera);               // ステレをでレンダリング実行
            }
（以下略）
```

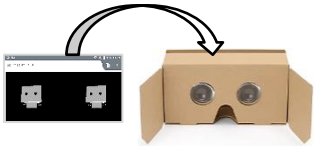
Check スマホで動作確認

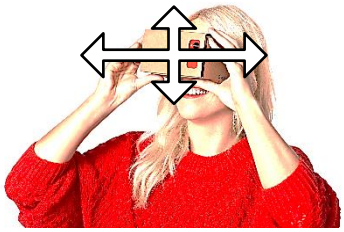
- スマホの Chrome で確認※1
  - アドレス例： **10.11.123.45/5.4a/**

※1 Firefox でも動作しますが、授業で使うスマホとの相性のため Chrome を使います。



- スマホを VR ゴーグルに入れて覗いてみる、首を上下左右に回してみる





Learn 解　説

① **`<script src="DeviceOrientationControls.js"></script>`**

スマホの動きでシーン内のカメラの向きを変えるためのライブラリを読み込みます。

② **`window.addEventListener('deviceorientation', gyroControl);`**

4.2 で使ったジャイロセンサのイベントです。スマホが上下左右に動いたら（ジャイロセンサの値が変化したら）gyroControl という関数を実行して、シーン内のカメラの向きを変えます。

③ **`const ctrl = new THREE.DeviceOrientationControls(camera, true);`**

ジャイロセンサの値でシーン内のカメラの向きを制御する THREE.DeviceOrientationControls オブジェクトです。④の `ctrl.connect()` と `ctrl.update()` とセットで使います。

④ **`ctrl.connect();` および `ctrl.update();`**

ジャイロセンサの値をカメラにつないで（connect）、カメラの向きを更新（update）するという命令です。そういう仕様ですので「必ずセットで使う」と思っておけば構いません。

✈ 5.4b スマホで VR（複数の物体）

	PC	And	iOS
Firefox	×	○	○
Chrome	×	○	○

3D モデルは一つだけでなく、シーン内に複数置くことができます。せっかくの VR ですから、いろいろな場所に物体を置き、見回すことを楽しむようにしましょう。

Input HTML と JavaScript を入力

- **C:￥xampp￥htdocs￥5.4a をフォルダごと複製**して、**フォルダ名を 5.4b に**します。
- index.html を Brackets で開き、以下のコードを入力しましょう。**変更は網掛けの部分**です。
- 「5.4b 素材」の中にある以下の 5 つのファイルを 5.4b フォルダに入れましょう。

'three.js'　'huchan.json'　'StereoEffect.js'
'DeviceOrientationControls.js'　'earthmap.jpg'

※ ①や②などの丸数字は打ち込まないこと
　コメントは入力必須ではありません。

C:￥xampp￥htdocs￥5.4b￥ **index.html**

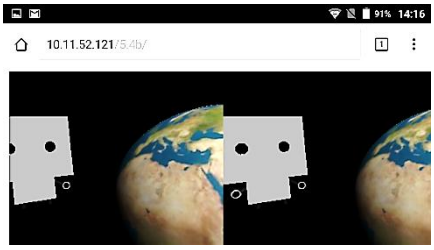
```
（略）
        <title>5.4b スマホで VR（複数の物体）</title>
    </head>
    <body>
        <div id='screen'></div>
        <script src='three.js'></script>
        <script src="StereoEffect.js"></script>
        <script src="DeviceOrientationControls.js"></script>
        <script>
            // 描画する画面サイズを定義
                    :
                （中略）
                    :
            // シーンに 3DCG モデルを置く
                    :
                （中略）
                    :
            // シーンに球を置く                                                    ①
            const geoGlobe = new THREE.SphereGeometry(50, 32, 32);    // 球を指定
            const texLoader = new THREE.TextureLoader();      // テクスチャを読む準備
            const texture = texLoader.load('earthmap.jpg');   // テクスチャを読み込む
            const matGlobe = new THREE.MeshPhongMaterial({map: texture}); // マテリアル
            const globe = new THREE.Mesh(geoGlobe, matGlobe); // 球を作成
            globe.position.set(80, -30, 0); // 球を置く位置
            scene.add(globe);                 // 球を追加
            // レンダラ（3D 空間を描画する機能）を準備
            const renderer = new THREE.WebGLRenderer();     // レンダラを作成
            renderer.setSize(width, height);                // レンダラのサイズを指定
                                    // 次のページに続きます
```

```
            renderer.setClearColor(0x000000, 1.0);          // 背景色と不透明度の設定    ②
            const screen = document.getElementById('screen'); // div 要素を取得
            screen.appendChild(renderer.domElement);        // div 内にレンダラを置く
                    :
                ( 中略 )
                    :
            // レンダリング ( 描画 ) を繰り返し実行 ( アニメーション ) する関数
            function renderScene() {
                mesh.rotation.y += 0.01;                    // y 軸まわりに少し回す
                mesh.rotation.x += 0.01;                    // x 軸まわりに少し回す
                globe.rotation.y += 0.005;                  // 球を y 軸まわりに回す      ③
                stereo.render(scene, camera);               // ステレをでレンダリング実行
            }
( 以下略 )
```

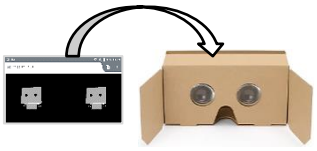
Check スマホで動作確認

- スマホの Chrome で確認[※1]
  - アドレス例： **10.11.123.45/5.4b/**

※1 Firefox でも動作しますが、授業で使うスマホとの相性のため Chrome を使います。



- スマホを VR ゴーグルに入れて覗いてみる、首を上下左右に回してみる



Learn 解　説

①　シーンに球を置く

5.1a の要領でシーンに 3D 物体を置いています。THREE.SphereGeometry は球状の物体を作ります。( )内のパラメータは(球の半径，横方向のポリゴン分割数，縦方向のポリゴン分割数)です。

また、5.1b の要領で、この球体に地球のテクスチャ (earthmap.jpg) を貼り付けています。

②　renderer.setClearColor(0x000000, 1.0);

レンダラの背景色を 0x000000 (真っ黒) にして、宇宙に浮かんでいる雰囲気にしています。

③　globe.rotation.y += 0.005;

①で作った球を回転させています。

✈ 5.4c スマホで VR（全画面化）

	PC	And	iOS
Firefox	×	○	○
Chrome	×	○	○

VR っぽくなってきましたが、ここまでの例では、ブラウザのアドレスバーなどが見えてしまい没入感が出ません。ブラウザを全画面化して、VR アプリを仕上げましょう。

Input HTML と JavaScript を入力

- **C:￥xampp￥htdocs￥5.4b をフォルダごと複製**して、**フォルダ名を 5.4c に**します。
- index.html を Brackets で開き、以下のコードを入力しましょう。**変更は網掛けの部分**です。
- フォルダ内に以下の 5 つのファイルがあることも確認しておきましょう。

'three.js' 'huchan.json' 'StereoEffect.js' 'DeviceOrientationControls.js' 'earthmap.jpg'

※ ①や②などの丸数字は打ち込まないこと
　コメントは入力必須ではありません。

C:￥xampp￥htdocs￥**5.4c**￥ **index.html**

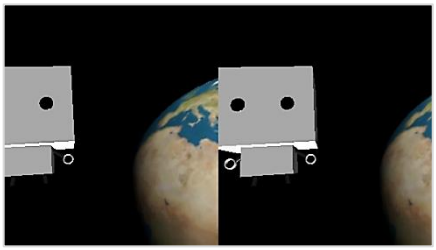
```
（略）
        <title>5.4c スマホで VR（全画面化）</title>
        <style>    /* ブラウザのウィンドウ全体に描画するための設定 */                    ①
            body {
                margin: 0px;
                overflow: hidden;
            }
            #screen {
                position: absolute;
                top: 0;
                left: 0;
                right: 0;
                bottom: 0;
            }
        </style>
    </head>
    <body>
        <div id='screen'></div>
        <script src='three.js'></script>
        <script src="StereoEffect.js"></script>
        <script src="DeviceOrientationControls.js"></script>
        <script>
            // 描画する画面サイズを定義
            let width = window.innerWidth;      // 画面の横幅（ウィンドウの横幅にする） ②
            let height = window.innerHeight;    // 画面の高さ（ウィンドウの高さにする）
            // シーン（3D 空間）を作成（座標系は右手系 Y アップ）
            const scene = new THREE.Scene();
            // const axis = new THREE.AxisHelper(100);  // 座標軸はコメントアウトして
            // scene.add(axis);                         // 無くしてしまう
                                    // 次のページに続きます
```

```
            // シーンにカメラを置く
                  :
              ( 中略 )
                  :
            // スマホのセンサで視点を操作
            // deviceorientation イベントを定義
            window.addEventListener('deviceorientation', gyroControl);
            // deviceorientation イベントが発生した時に呼ばれる関数
            function gyroControl(event) {
                // コントローラで camera オブジェクトの視点を操作する
                const ctrl = new THREE.DeviceOrientationControls(camera, true);
                ctrl.connect(); // コントローラに接続
                ctrl.update();  // コントローラの情報を更新
            }
            // 全画面化 ( 画面をタップすると全画面化 )
            screen.addEventListener('click', fullscreen);   // 画面クリックイベント      ③
            // 全画面化する関数 ( ブラウザによって命令が異なる )
            function fullscreen() {
                screen.webkitRequestFullscreen();   // Chrome, Safari, Opera の場合     ④
                // screen.mozRequestFullScreen();   // Firefox の場合
                // screen.msRequestFullscreen();    // IE の場合
                // screen.requestFullscreen();      // 本来の仕様
            }
            // 画面サイズが変わった時の処理
            window.addEventListener('resize', resize);  // リサイズイベント           ⑤
            function resize() {                                                      ⑥
                width = window.innerWidth;          // 新しい画面の横幅を取得
                height = window.innerHeight;        // 新しい画面の高さを取得
                camera.aspect = width / height;     // カメラのアスペクト比を再設定
                renderer.setSize(width, height);    // レンダラのサイズを再設定
                stereo.setSize(width, height);      // ステレオレンダラのサイズを再設定
            }
            // レンダリング ( 描画 ) を繰り返し実行 ( アニメーション ) する関数
            function renderScene() {
( 以下略 )
```

Check スマホで動作確認

- スマホの Chrome で確認※1
  - アドレス例： **10.11.123.45/5.4c/**

※1 Firefox でも動作しますが、授業で使うスマホとの相性のため Chrome を使います。



- スマホを VR ゴーグルに入れて覗いてみる、首を上下左右に回してみる

Learn 解　説

①　/* ブラウザのウィンドウ全体に描画するための設定 */

全画面にした時の画面の見た目の調整です。どんな意味なのかは自分で調べてみましょう。

②　let width = window.innerWidth; と let height = window.innerHeight;

全画面化の処理⑥で width と height に値を再代入するので、const ではなく let で宣言します。

③　screen.addEventListener('click', fullscreen);

画面をタップした時に全画面化するための click イベントの定義です。タップされると、すぐ下の fullscreen 関数が実行されます。

④　screen.webkitRequestFullscreen();

全画面化させる requestFullscreen という命令です。現状、仕様上は requestFullscreen として定義されていますが、まだブラウザのメーカーごとに統一されていない命令です。今後は統一されるはずですが、現状は利用するブラウザによって異なる命令を使う必要があります。

⑤　window.addEventListener('resize', resize);

画面サイズが変更されたときに発生する resize イベントの定義です。③の requestFullscreen によって画面サイズが変わると resize イベントが発生し、すぐ下の resize 関数を実行します。

⑥　function resize() { … } 内の処理

画面サイズが変わったら、three.js で作った様々なオブジェクトのサイズを変える必要があります。具体的には、カメラの縦横比、レンダラ、および、ステレオレンダラのサイズです。これらを再設定しています。

Break コラム⑮　VR の新定番？　A-Frame

この授業では three.js を使った 3D と VR のプログラミングを紹介しました。three.js は少し難しい印象を持ったかもしれませんが、3DCG の基本を理解できると考えて紹介しました。

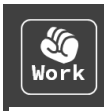
実は近ごろ（2016 年 2 月）、もっと簡単に 3D や VR を実現できる、A-Frame というライブラリが現れました。Firefox ブラウザを作っている Mozilla 財団が作っているライブラリで、今後の普及が大いに期待できるライブラリです。

3D や VR に興味を持ったら、ぜひ A-Frame についても調べてみて、作ってみましょう！

← https://aframe.io/

データを持ち帰る

- 本日作成したデータは全て USB メモリなどにコピーして持ち帰りましょう
  - C:￥xampp￥htdocs 内の以下のフォルダ（7 個）
  - 5.1a/b　5.2　5.3　5.4a/b/c



本日の課題

① 5.1b、5.2、5.3 のコードを完成させる（可能ならスマホで動作確認する）

・ 資料のコードをもとに自分なりのアレンジを加えても構いません。その際は②の readme.txt の所感にアレンジ内容を書いてください。効果的なアレンジであれば成績評価に加点します。

② readme.txt というテキストファイルを作り、以下を書く

・ 学籍番号 と 氏名
・ 所感　（考えたこと、感想、応用アイデアなど。字数不問だが数行は書いてほしい。）
・ 質問　（無ければ不要）

【提出方法】

・ ①のフォルダ 3 個と、②の「readme.txt」1 個、全てをまとめて zip 圧縮
・ zip のファイル名は「学籍番号.zip」とする　（例：12345nhu.zip）
・ CampusSquare のレポート「フィジカルコンピューティング **05**」から提出

【提出締切】　11 月 10 日（木）　15:00（遅れてしまった場合は担当教員に相談のこと）