＜本日の課題＞

課題 9-1　テクスチャマッピング

サンプルコード **sample09-1** を実行し，読み込んだ画像がテクスチャとして平面（plane）に適用されることを確認しなさい。また，**GL.Texture.fromURL()** で指定する画像ファイルを変更し，同様にテクスチャとして適用されることを確認しなさい。

sample09-1(抜粋)
○シェーダの設定

```
<script id="vertexShader" type="x-shader/x-vertex">
  varying vec2 coord;    // フラグメントシェーダに渡すテクスチャ座標(uv 座標)を宣言

  void main() {
    // テクスチャ座標 (gl_TexCoord.xy) をフラグメントシェーダに渡す
    coord = gl_TexCoord.xy;
    gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
  }
</script>

<script id="fragmentShader" type="x-shader/x-fragment">
  uniform sampler2D texture;  // テクスチャデータを参照するための変数（サンプラー）
  varying vec2 coord;         // 頂点シェーダから渡されるテクスチャ座標

  void main() {
    // テクスチャ座標 (coord) を使用してテクスチャの色をピクセルの色として設定
    gl_FragColor = texture2D( texture, coord );
  }
</script>
```

頂点シェーダ: メッシュ作成時の coords オプションによりメッシュの頂点に合わせた UV 座標 (0,0)～(1,1) が定義され，それが内部的に gl_TexCoord.xy に引き渡されている。
フラグメントシェーダ： texture2D 関数により，テクスチャ画像から UV 座標に基づき色を取得し，ピクセルの色（gl_FragColor）として設定される。

texture2D(sampler2D sampler, vec2 coord)
指定した UV 座標 (coord) を基に 2D テクスチャ (sampler) から色を取得
引数: sampler: テクスチャオブジェクト (sampler2D 型)， coord: テクスチャ座標 (0.0-1.0)
戻り値: vec4 型: サンプリングされたピクセルの RGBA 色

○メッシュの作成とテクスチャの生成

```
const mesh = GL.Mesh.plane({ coords: true });  // 平面のメッシュを作成
// 外部から画像ファイルを読み込んでテクスチャを生成
const texture = GL.Texture.fromURL('texture.jpg');
```

・メッシュ作成時の {coord: true} は，UV 座標を自動的に割り当てるオプション設定
・ GL.Texture.fromURL で外部画像を読み込み，テクスチャを生成する。(画像ファイルのパスに注意)

○テクスチャのメッシュへの適用

```
texture.bind(0);        // テクスチャをテクスチャユニット 0 に関連付け（バインド）
shader.uniforms({ texture: 0 })   // テクスチャユニット 0 を割り当て
shader.draw(mesh);      // メッシュを描画
```

・texture.bind(0); で，テクスチャを ユニット 0 として設定（バインド）
・shader.uniforms({ texture: 0}); により，シェーダ内の texture に ユニット 0 を割り当て

補足）テクスチャユニット：複数のテクスチャをシェーダで同時に利用するための識別子

課題 9-2　オブジェクトへのテクスチャの適用（１）

サンプルコード **sample09-2** を実行し，読み込んだ画像がテクスチャとして立方体（cube）に適用されることを確認しなさい。また，球（sphere）に対しても同様にテクスチャを適用できることを確認しなさい。

課題 9-3　オブジェクトへのテクスチャの適用（２）

サンプルコード **sample09-2** を書き換えて，各自が準備した画像ファイルをテクスチャとして適用できることを確認しなさい。ここで，「UVchecker.png」をテクスチャ画像として用いると，オブジェクトへの UV 座標の割り当てが確認できる。

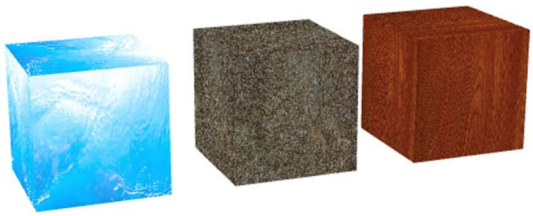
課題 9-4　オブジェクトへのテクスチャの適用（３）

サンプルコード **sample09-3** を実行し，立方体の各面に異なる画像がテクスチャとして適用されることを確認しなさい。また，フォルダ dice 内の画像を使ってサイコロを描画しなさい。ただし，サイコロの対面の数の和は必ず 7 となるものとする。

○課題 9-5 テクスチャを使ったシーンの作成

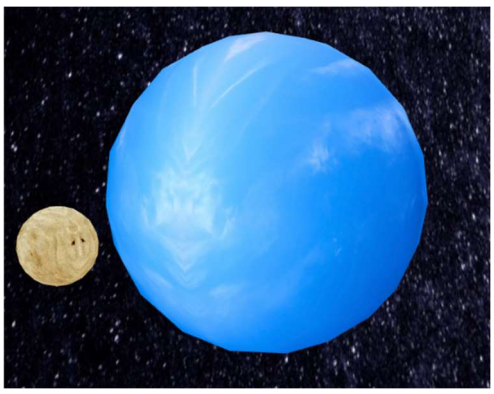
次に示すシーンを描画しなさい。ここで，オブジェクトの大きさ，配置は問題文の条件の範囲で任意とする。また，テクスチャ画像は各自で準備すること。**（課題提出時には使用した画像ファイルも一緒に提出！）**

1）異なるテクスチャを適用した 3 つの立方体



2）地面に配置された立方体

3）背景が星空の自転する惑星と公転する衛星



4）各自でシーンを考え実装する。
ただし，テクスチャを適用したオブジェクトを 3 つ以上使うこと

・ひな形として，kadai09-5.html を準備したので必要な人は利用してください。

課題 9-6　レイトレーシング

レイトレーシングのサンプルコード **raytracing** について，次の課題に取り組みなさい。

1）サンプルコード **raytracing** を実行し，画面に描画されるシーンを確認しなさい。特に，光の反射や影について観察してください。

2）球の位置（**sphereCenter**）を次のように変更したときシーンがどのように変化するか確認しなさい。

- 球をカメラに近づける　(0.0, 1.6, −2.0)
- 球をカメラから遠ざける　(0.0, 1.6, 5.0)
- 球を左側に移動する　(−2.0, 1.6, 0.0)

3）光源の位置（**lightPos**）を次のように変更したときシーンがどのように変化するか確認しなさい。

- 光源を高くする　(5.0, 20.0, 5.0)
- 光源を低くする　(5.0, 2.0, 5.0)
- 光源を左側に移動する　(−5.0, 10.0, 5.0)

4）球の半径（**sphereRadius**）を次のように変更したときシーンがどのように変化するか確認しなさい。

- 半径を小さくする　0.5
- 半径を大きくする　3.0
- 半径をさらに大きくする　5.0

5）上記の結果を踏まえて，レイトレーシングとはどのような技術か調べなさい。

変更箇所：gl.ondraw 内の shader.uniforms（シェーダに渡す変数の設定）

```
shader.uniforms({
  eye: tracer.eye, // カメラ位置
  ray00: tracer.getRayForPixel(0, h), // 左下レイ
  ray10: tracer.getRayForPixel(w, h), // 右下レイ
  ray01: tracer.getRayForPixel(0, 0), // 左上レイ
  ray11: tracer.getRayForPixel(w, 0), // 右上レイ
  lightPos: [5.0, 10.0, 5.0],     // 光源の位置
  sphereCenter: [0.0, 1.6, 0.0], // 球の中心
  sphereRadius: 1.5              // 球の半径
});
```

補足）ブラウザ（Chrome）でのローカルファイルの読み込み

GoogleChrome ではセキュリティ上の理由でローカルファイルの読込みに制限がかかっており，テクスチャ画像が表示されない。この場合，起動オプションに「`-allow-file-access-from-files`」を付けることで表示ができるようになる。

Windows

1）Google Chrome のショートカットを新規作成

2）新規作成したショートカットを右クリック→プロパティを開く

3）「リンク先」のパスのあとに空白を開けて「`-allow-file-access-from-files`」と入力

Mac

1）Finder で「アプリケーション」→「ユーティリティ」→「ターミナル」を開く。

2）Google Chrome を起動するためのコマンドを入力し，実行する。

```
open -a Google¥ Chrome --args -allow-file-access-from-files
```

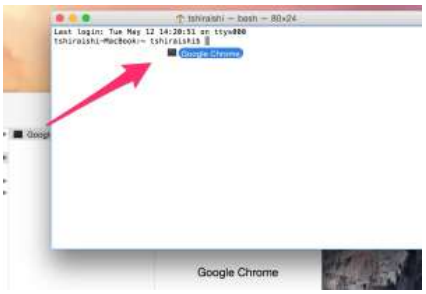
もしくは

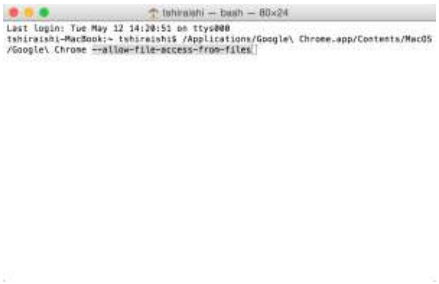
1）Chrome のアプリの「パッケージの内容を表示」

2）「Contents/MacOS/Google Chrome」を，ターミナルにドラッグ＆ドロップ

3）そこまでのパスが自動的に入るので，

その後に半角スペースをあけて「`-allow-file-access-from-files`」をつけて Enter





なお，　VSCode の拡張機能（LivePreview など）によるプレビューを使う場合は，上記設定は不要です。

以上