＜本日の課題＞

課題 8-1　シェーダの利用

サンプルコード **sample08-1** を実行し，シェーダを使って立方体が描画されることを確認しなさい。

sample08-1(抜粋)

```
<script id="vertexShader" type="x-shader/x-vertex">
  void main() {
    gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
  }
</script>

<script id="fragmentShader" type="x-shader/x-fragment">
  void main() {
    gl_FragColor = vec4(1.0, 1.0, 1.0, 1.0);
  }
</script>
```

```
  const mesh = GL.Mesh.cube();
  const shader = new GL.Shader('vertexShader','fragmentShader');
```

```
  shader.draw(mesh);
```

課題 8-2　シェーディング（ライティング）

サンプルコード **sample08-2** を実行し，光源を与えた時の立方体の様子を確認しなさい。また，**sample08-1** の立方体と比較，確認しなさい。

sample08-2(抜粋)

```
<script id="vertexShader" type="x-shader/x-vertex">
  varying vec3 normal;
  void main() {
    normal = gl_NormalMatrix * gl_Normal;
    gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
  }
</script>

<script id="fragmentShader" type="x-shader/x-fragment">
  varying vec3 normal;
  void main() {
    vec3 lightDir = normalize(vec3(1.0, 1.0, 1.0));
    float intensity = max( dot(normalize(normal), lightDir), 0.0);
    gl_FragColor = vec4(intensity, intensity, intensity, 1.0);
  }
</script>
```

```
  const mesh = GL.Mesh.cube( {normals:  true} );
  const shader = new GL.Shader('vertexShader','fragmentShader');
```

課題 8-3　法線ベクトルの役割

サンプルコード **sample08-2** において，mesh 生成時のオプション　normals（法線ベクトルの生成）を false に変更すると，どのような変化があるか確認しなさい。また，なぜその様になるのか考えなさい。

課題 8-4　メッシュの生成

サンプルコード **sample08-2** において，mesh 生成時のオプションを変更し，どのような変化があるかを確認しなさい。また，立方体（cube）以外の 球（sphere），平面（plane）についても同様に確認しなさい。

図形	メソッド	使用できるオプション
立方体	GL.Mesh.cube()	normal: 法線ベクトルを生成するかを指定（true で生成） coords: テクスチャ座標を生成するかを指定（true で生成） colors: 頂点カラーを生成するかを指定（true で生成） 以上は共通オプション（他の図形でも使用可）
球	GL.Mesh.sphere()	detail: 球の分割数を指定（デフォルトは 6）
平面	GL.Mesh.plane()	detailX: X 方向の分割数を指定（デフォルトは 6） detailY: Y 方向の分割数を指定（デフォルトは 6） detail: detailX と detailY の両方を同じ値に設定

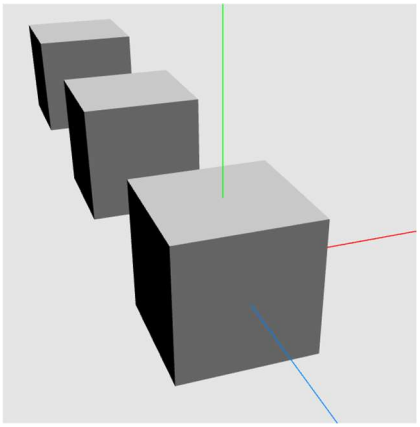
オプションの使用例）　`const mesh = GL.Mesh.sphere( { normals: true, detail: 30 } );`

→ 球について，各面の法線ベクトルを生成，分割数は 30 に指定

○課題 8-5　シェーダを使ったシーンの作成

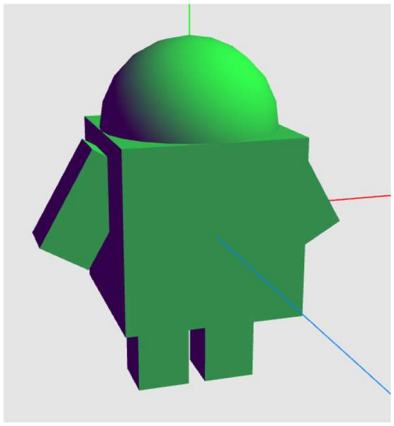
次に示すシーンを描画しなさい。ここで，オブジェクトの色や大きさ，間隔は任意とする。

1）並んだ 3 つの立方体



Hint: `shader.draw(cube);` を 3 回記述する

2）オブジェクトを組み合わせただけのロボット



・頭は球，胴体は立方体，腕・脚は直方体で構成

・各部位の色，大きさ，配置は任意

Hint: 直方体は `gl.scale(x,y,z);` を使って表現

ひな形として kadai08-5.html を準備したので利用してください。

課題 8-6 光の表現（Phong の反射モデル）

サンプルコード **sample08-3 の 1〜3** を実行し，CG における光の表現法（Phong の反射モデル）について確認しなさい。

sample08-3-3(フラグメントシェーダの抜粋)

```
varying vec3 vNormal;     // 頂点の単位法線ベクトル
varying vec3 vPosition;   // モデルビュー座標系での頂点位置

void main() {
  vec3 ambientLight = vec3(0.0, 0.0, 1.0) * 0.5;  // 環境光
  vec3 lightPosition = vec3(10.0, 10.0, 10.0);    // 光源の位置
  vec3 diffLight = vec3(1.0, 0.0, 0.0);           // 拡散反射光
  vec3 specLight = vec3(1.0, 1.0, 1.0);           // 鏡面反射光
  vec3 viewPosition = vec3(0.0, 0.0, 10.0);       // 視点の位置

  // 拡散反射の強度を算出
  vec3 lightDir = normalize(lightPosition - vPosition);
  float diff = max(dot(vNormal, lightDir), 0.0);
  vec3 diffuse = diffLight * diff;

  // 鏡面反射の強度を算出
  vec3 viewDir = normalize(viewPosition - vPosition);
  vec3 reflectDir = reflect(-lightDir, vNormal);
  float spec = pow(max(dot(viewDir, reflectDir), 0.0), 16.0);
  vec3 specular = specLight * spec;

  // 環境光、拡散反射、鏡面反射を合成
  vec3 color = ambientLight + diffuse + specular;
  gl_FragColor = vec4(color, 1.0);
}
```

課題 8-7 光の表現

サンプルコード **sample08-3-3** について，次の操作をしたときに球の見た目がどのように変わるかを確認しなさい。

1）環境光を**緑色 (vec3(0.0, 1.0, 0.0))** で，**強度を 0.2** に変更する。

2）光源の位置を **真上(vec3(0.0, 10.0, 0.0))**，**逆方向(vec3(-10.0, -10.0, 10.0))**に配置する。

3）鏡面反射の指数（現在は 16.0）を，**4.0（粗い）**，**32.0（滑らか）**に変更する。

4）拡散反射光の色を，**黄色，紫色**に変更する。

5）視点の位置を，**右側 (10.0, 0.0, 10.0)**，**上側(0.0, 10.0, 10.0)** に移動する。

課題 8-8　シェーダへのデータ渡し（１）

サンプルコード **sample08-4** を実行し，uniform を利用してシェーダに光源の情報を渡すことができることを確認しなさい。

シェーダにデータを渡す手順

① シェーダの冒頭部分に **uniform** を使ってデータを受け取るための変数を定義する。

sample08-4(フラグメントシェーダの冒頭部分の抜粋)

```
// 環境光と拡散反射光の設定を外部から受け取る
uniform vec3 ambientLight;    // 環境光の設定
uniform vec3 diffLight;       // 光源の色（拡散反射光）

void main() {
    ・・・
}
```

② シェーダを作成した後に **shader 名.uniforms({変数名: 値, ・・・ })** の形でデータを渡す。

sample08-4(抜粋)

```
// シェーダに光源の情報を渡す
shader.uniforms({
    ambientLight: [0.1, 0.1, 0.1], // 環境光 (vec3)
    diffLight: [0.6, 0.0, 0.0]     // 拡散反射光 (vec3)
});
// シェーダで球を描画
shader.draw(mesh);
```

○課題 8-9　シェーダへのデータ渡し（２）

質感の異なる２つの球を表示するプログラムを作成しなさい。

Hint: 環境光，拡散反射光に加えて他のパラメータも設定できるようにして，それぞれ値を変えて呼び出す。

以上