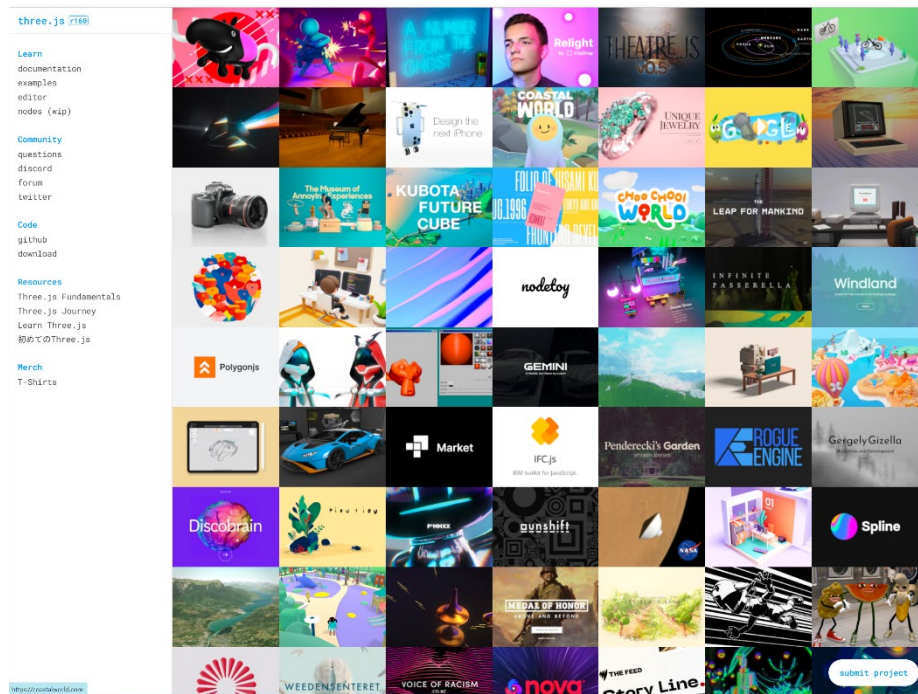


Three.js

■Three.js とは

Three.js は、Web ブラウザでプラグインなどの手間がなく 3DCG コンテンツを制作・利用できる JavaScript ライブラリ。



サンプル：<https://oreilly-japan.github.io/learning-three-js-2e-ja-support/#ch01>

◆WebGL

WebGL (Web Graphics Library) は、Web ブラウザ上で 3DCG を描画するための技術。

HTML5 に標準で搭載されたグラフィック機能 (Canvas) に JavaScript 言語で OpenGL へ描画命令を指示する **API** (application programming interface)。

この WebGL の中でも代表的なライブラリのひとつが **Three.js**

◆OpenGL と WebGL

OpenGL (Open Graphics Library) と WebGL (Web Graphics Library) は、グラフィックスプログラミングの分野で使用する二つの異なるテクノロジーで。



OpenGL

歴史 : 1992 年、グラフィックワークステーション制作会社 SGI が内部向けに開発した API。その後、この API は公開され、大型コンピュータ、PC、携帯電話、PDA に至るまで用いられている。また、家電向けの OpenGL ES (OpenGL for Embedded Systems) も実用化されている。

全体像 : OpenGL は、多くのオペレーティングシステムで利用可能なクロスプラットフォームのグラフィックス API。

使用用途 : 主に 3D グラフィックスの描画に使われ、ゲーム、CAD プログラム、仮想現実など多岐にわたるアプリケーションで使用されている。

機能の豊富さ : OpenGL は非常に強力で、広範囲なグラフィックス機能と拡張性を提供。

直接性 : ハードウェアのグラフィックス機能に直接アクセスし、高いパフォーマンスを発揮する。





WebGL

歴史：2011 年、非営利団体クロノス・グループ（Khronos Group：技術コンソーシアム）により設計、メンテナンスされた。

全体像：WebGL は、Web ブラウザ上で動作する 3D グラフィックスを描画するための API。

基盤技術：WebGL は OpenGL ES（OpenGL for Embedded Systems）に基づいている。OpenGL ES は OpenGL のサブセットで、特にモバイルデバイスや組み込みシステム向けに設計されている。

ブラウザ互換性：WebGL は HTML5 と統合され、JavaScript を介してアクセスされる。これにより、追加のプラグインなしで多くの最新ブラウザで 3D グラフィックスを実現できる。

セキュリティと利便性：ブラウザベースであるため、WebGL は「セキュリティ」と「利便性」に重点を置いている。しかし、これにより OpenGL に比べて直接的なハードウェアアクセスや機能面での制限がある。

◆2 つのソフトの関係性

技術的基盤：WebGL は OpenGL ES に基づいているため、OpenGL のコンセプトや機能の多くを継承している。

目的の違い：用途が異なるため（WebGL は Web 向け、OpenGL は一般的なアプリケーション向け）、両者は異なる制限と特性を持っている。

開発者の利点：OpenGL の経験がある開発者は、WebGL を比較的容易に学ぶことができるが、Web 環境固有の制約と機能にも適応する必要がある。

総じて、WebGL は Web ブラウザでの使用に特化した OpenGL ES の軽量バージョンと言える。それぞれが特定の環境とニーズに合わせて設計されており、強力な 3D グラフィックス機能を異なるプラットフォームで提供している。

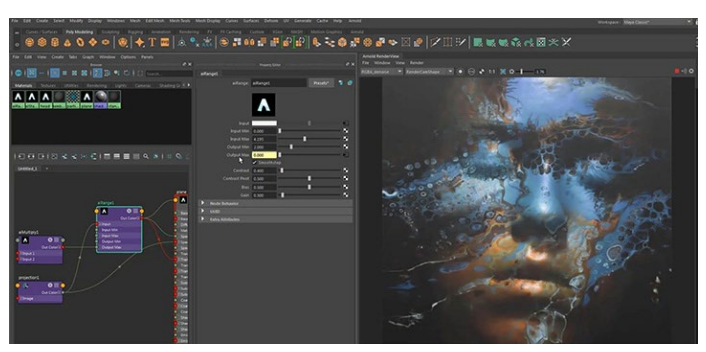
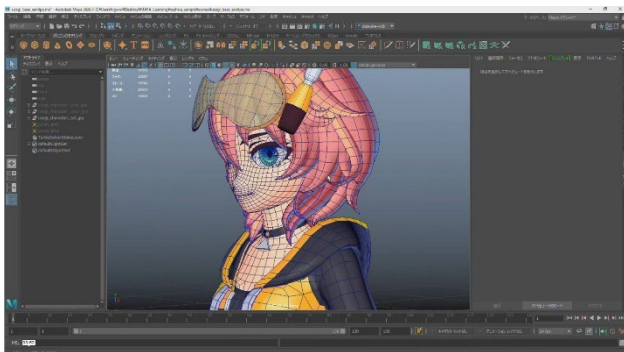
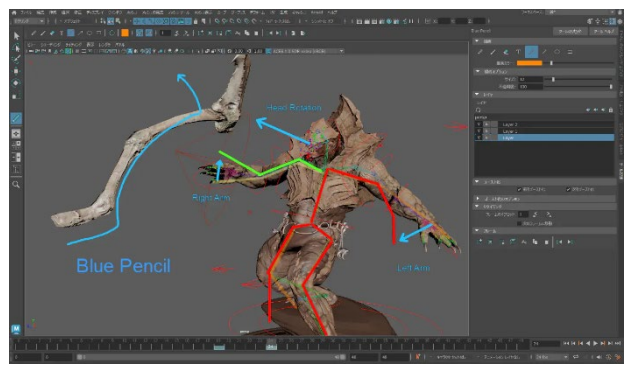
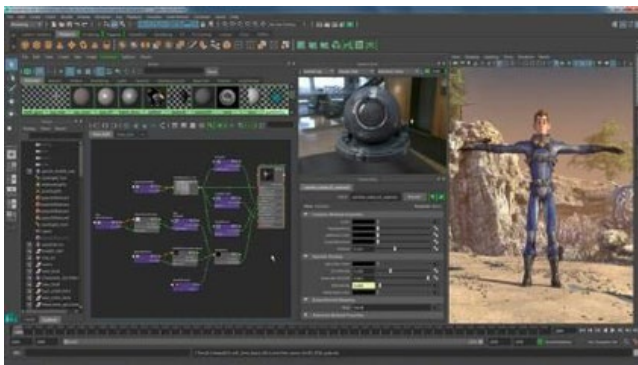
■3DCG 制作のためのソフト

◆Autodesk 社 Maya



1983 年創業の Alia Systems 社によって開発された 3DCG 制作アプリケーション。その後、2005 年にライバル会社の Autodesk 社に買収された。

Maya は、その複雑さと高度な機能により、専門的なトレーニングや学習が必要とされることもあるが、3D コンテンツ制作の分野で非常に強力なツールとして位置づけられています。



・Maya の主な特徴

モデリング： 高度なポリゴン、サブディビジョンサーフェス、NURBS モデリングツールを提供し、複雑な 3D モデルの作成を可能にしている。

アニメーション： 高度なキャラクターリギングとアニメーションツールを提供し、リアリスティックなキャラクターと動きの作成をサポート

シミュレーションとエフェクト： 力学的、流体的な物理シミュレーションを実行でき、リアルな動きやエフェクトを作成できる。

レンダリング： Arnold レンダラーを内蔵しており、高品質なビジュアライゼーションとレンダリングが

可能。

カスタマイズとスクリプト：MEL（Maya Embedded Language）や Python スクリプトを使用して、作業フローをカスタマイズし、自動化することができる。

・Maya の用途

映画とテレビ：多くのハリウッド映画の VFX（視覚効果）やアニメーション制作に広く使われている。

ゲーム開発：ゲームのキャラクターや環境のモデリング、アニメーション制作に使われている。

アーキテクチャと製品デザイン：ビジュアル化やプロトタイピングのために使われる

仮想現実（VR）と拡張現実（AR）：これらの新しいメディア形式のコンテンツ制作にも対応

◆Autodesk 社 3ds MAX

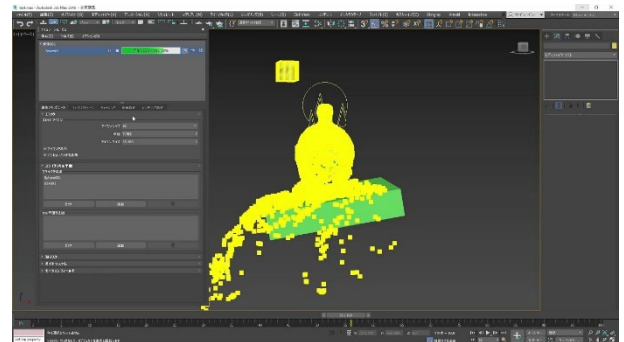


もともと建築用ソフトを開発していた Autodesk 社は、フィニッシュ画像のひとつとして 3DCG を提供しはじめ、1990 年 3D Studio DOS としてリリースを開始した。その後、名称を 3d Studio MAX とした。

3DCG を描画するレンダラーには、NVIDIA 社が提供する 3D レンダリングエンジンを使用していたが、現在（2018 以降）は物理ベースの 3D レンダリングアプリケーション Arnold が使用されている。（これは Maya も同じ）

その柔軟性と強力なモデリング、レンダリング、アニメーション機能で知られており、プロフェッショナルな 3D アーティストやデザイナーにとって重要なツールの一つです。その直感的なインターフェースと強力な機能により、3D グラフィックスの分野で高い評価を受けています。初心者から上級者まで幅広いユーザーに対応できる設計となっており、特に建築ビジュアライゼーションやゲーム開発の分野で人気があります。





・3ds Max の主な特徴

モデリング： ポリゴンベースのモデリング機能は非常に強力で、複雑な形状やデザインを作成することができる。

アニメーション： キャラクターリギングやアニメーションツールが充実しており、リアルな動きを持つキャラクターやオブジェクトの作成が可能。

レンダリング： 高品質なレンダリングエンジン（Arnold）を内蔵し、リアルタイムレンダリングやフォトリアリスティックなイメージを生成できる。

シミュレーションとエフェクト： 力学、粒子、流体などの物理ベースのシミュレーションをサポートしている。

スクリプトとプラグイン： Maxscript や Python スクリプトを使用したカスタマイズや自動化が可能で、豊富なサードパーティ製プラグインも利用できる。

3ds Max の用途

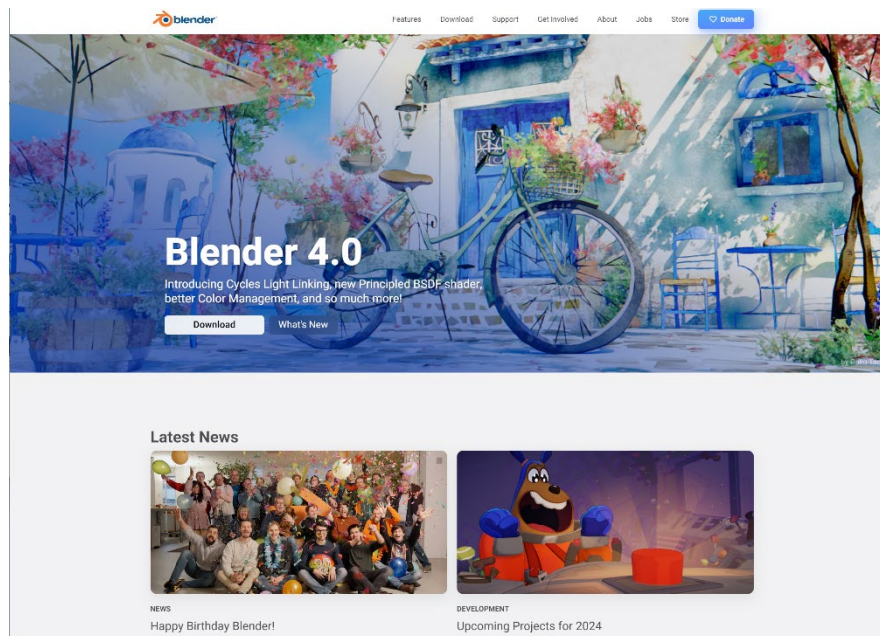
建築ビジュアライゼーション： 建築やインテリアデザインの 3D ビジュアライゼーションに広く使用されている。

ゲーム開発： ゲームアセットの作成、キャラクターモデリング、環境デザインなどに利用されている。

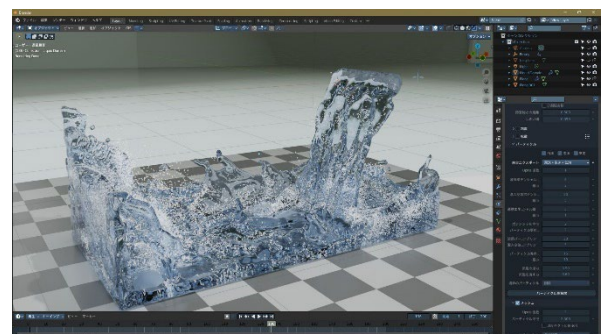
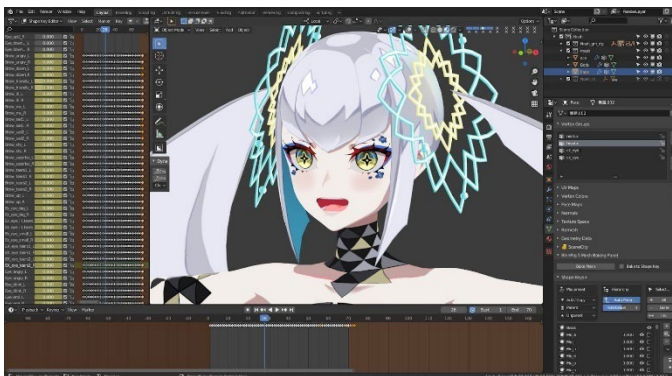
映画とテレビ： VFX やアニメーション制作において、特に 3D モデルやアニメーションの作成に使用される。

工業デザイン： 製品の 3D プロトタイピングやビジュアル化にも使われる。

◆Autodesk社 Maya



オープンソースの 3DCG ソフト。3D モデリング、アニメーション、レンダリング、動画編集、シミュレーション、ゲーム制作などの多岐にわたる機能を提供しており、その利用範囲はプロフェッショナルレベルの作品制作から趣味のプロジェクトまで広がっている。



・Blender の主な特徴

モデリング： ポリゴンベースのモデリングツールに加え、スカルプティング機能も備えており、高度な 3D モデルの作成が可能。

アニメーション： キャラクターリギング、アニメーション、モーションキャプチャー処理など、高度なアニメーションツールを提供してる。

レンダリング： 強力なレンダリングエンジン（Cycles、Eevee）を内蔵し、リアルタイムレンダリングやフォトリアリスティックなビジュアライゼーションが可能。

シミュレーション： 流体、煙、火、布、髪の毛などの物理ベースのシミュレーションをサポートしている。

動画編集： ビデオ編集機能も搭載しており、3D プロジェクトに加えて動画編集も行える。

スクリプトとカスタマイズ： Python を用いたスクリプトによる高度なカスタマイズが可能で、機能の拡張や自動化が行える。

Blender の用途

3D アートとビジュアライゼーション： 美術作品、ビジュアルエフェクト、アーキテクチャビジュアライゼーションなどに使用される。

映画とアニメーション： 短編から長編映画まで、アニメーション制作に幅広く利用されている。

ゲーム開発： 3D アセットの作成やゲームエンジンへの統合が可能。

教育とトレーニング： オープンソースであるため、教育機関での 3D グラフィックスやアニメーションのトレーニングツールとしても人気がある。

趣味と個人プロジェクト： 無料かつプロフェッショナルな機能を備えているため、趣味で 3D アートを楽しむユーザーにも適している。



◆Autodesk 社 Maya

1986 年からイーフロンティア社が提供している日本製の 3DCG ツール。(現在はフォーラム 8 社が継続) プロフェッショナルから趣味のユーザーまで幅広い層に利用されており、特にアーキテクチャ、インテリアデザイン、製品デザインの分野での使用が目立ちます。



・Shade3D の主な特徴

モデリング： 高度なポリゴンモデリングツール、NURBS モデリング、サブディビジョンサーフェスなど、多様なモデリング手法をサポートしています。

レンダリング： 高品質なレンダリング機能を備えており、フォトリアリスティックなビジュアライゼーションを生成できます。

アニメーション： 基本的なアニメーションツールを提供し、動きのある 3D シーンの作成が可能です。

UV マッピングとテクスチャリング： 効率的な UV マッピングツールとテクスチャリング機能を備えており、リアルな素材感を表現できます。

シミュレーション： 基本的な物理シミュレーション機能も提供しています。

・Shade3D の用途

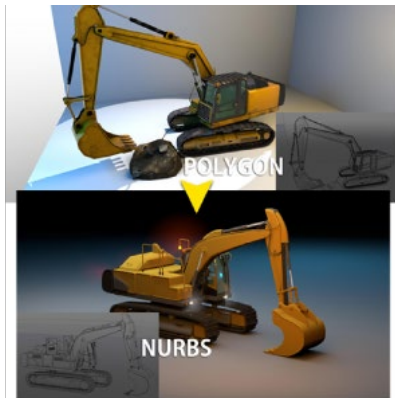
アーキテクチャビジュアライゼーション： 建築モデルの作成やリアルなビジュアライゼーションに適しています。

インテリアデザイン： 室内空間の 3D モデリングとビジュアライゼーションに使用されます。

製品デザイン： 工業製品や家具などのデザインプロセスに利用されることがあります。

教育用途：3D モデリングとレンダリングの基本を学ぶための教育ツールとしても使用されます。

趣味と個人プロジェクト：アマチュアアーティストや趣味のユーザーにも人気があり、個人的な 3D プロジェクトに利用されます。

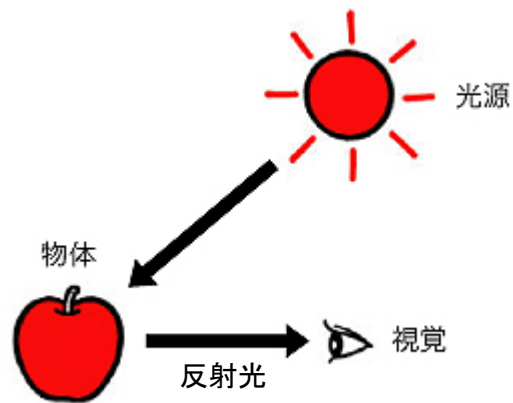


■CG 基礎

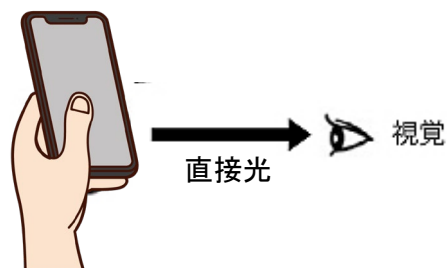
Three.js にしても、その他の 3DCG ツールにしても、3DCG の基礎と用語を知らないとは制作ができません。そこで、CG の基礎です。

◆見えると言うことは

私たちが見えると言うことは、「光源」から発出された光が、「物体」にあたり、その「反射光」が「目」に入ることでおきる現象です。

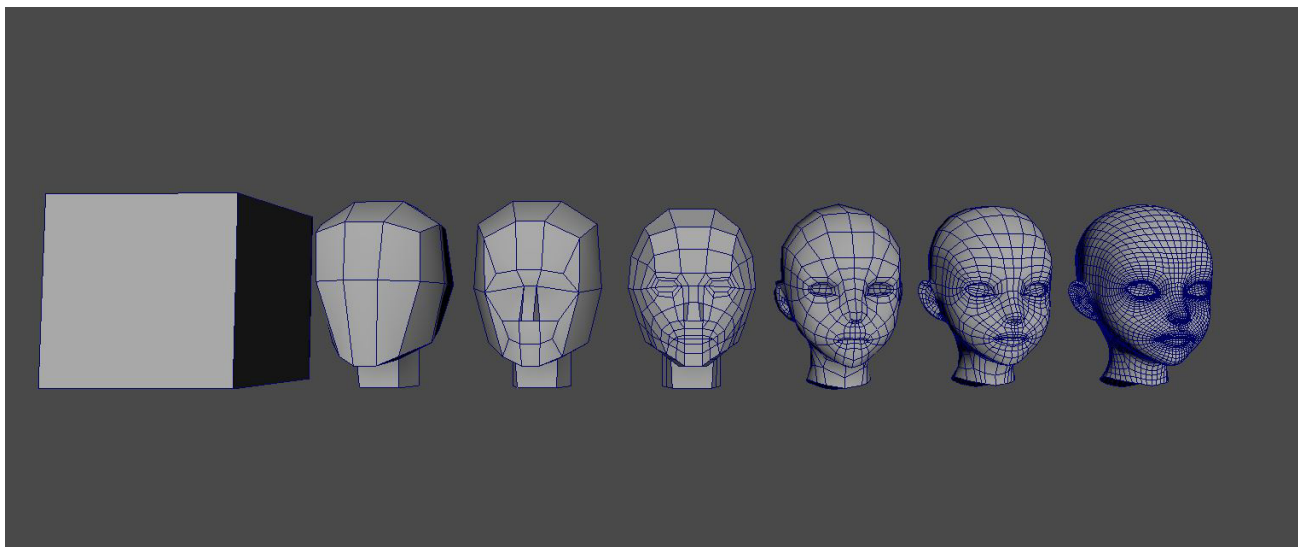


有史以来続いてきたこの物理現象に革命が起きている。
それは、物体を必要とせず、ディスプレイなどから発せられる直接光を見て、視覚情報を得ている。



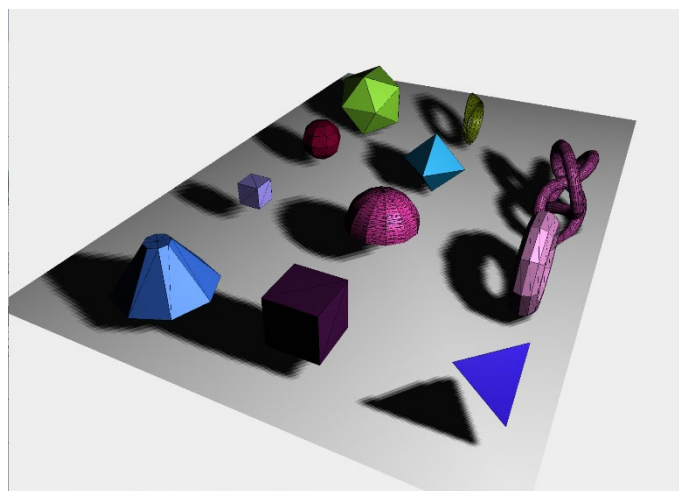
◆モデリング、プリミティブ

3DCG で形を作る事を **モデリング** と呼びます。



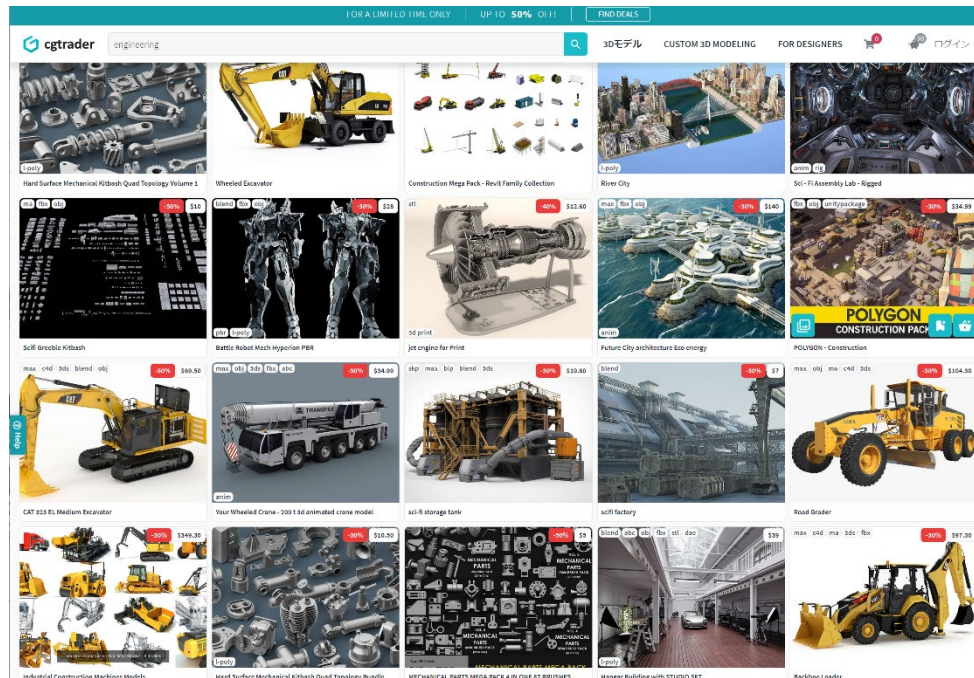
制作の中で最も重要な部分で、時間もかかります。線を細かく分割修正して、面で形を作って行きます。

モデリングでスタートになるのがプリミティブと呼ばれる立体です。Three.js の中には 11 種類のプリミティブがあります。



<https://oreilly-japan.github.io/learning-three-js-2e-ja-support/chapter-02/04-geometries.html>

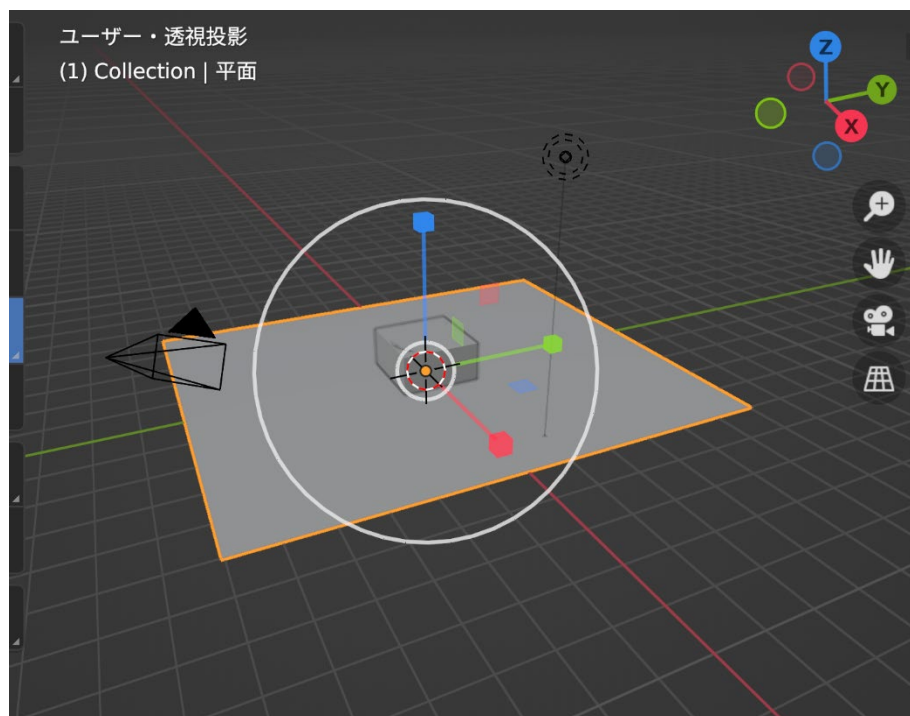
最近では、「写真」や「動画」を有料・無料サイトから手に入れるように、3DCG の世界でも、モデリングを販売しているサイトがあり、既存のものはその制作に時間を費やすのではなく、そうしたサイトから入手しています。



@cgtrader

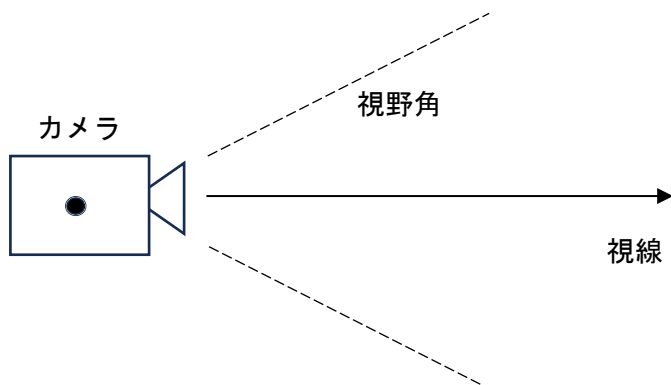
◆3DCGの要素

3DCG ツールで基本となる要素は「ライト」、「オブジェクト」、「カメラ」の3点。



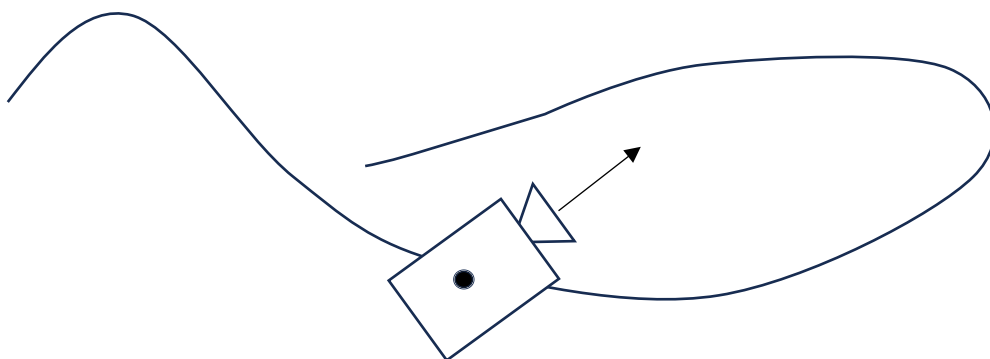
◆カメラ

カメラは一派に使われているカメラと変わりはない。ただ、大きな違いと言えば、そのカメラが自由に動けること。その際に、重要となるのは、カメラ本体の「位置」と「向き」、そしてカメラの「視線」方向。



＜カメラの位置、向き、視線＞

- ・カメラを固定して、カメラの向きを変える
- ・カメラの向きを固定して、カメラを移動する
- ・カメラを移動して、同時に視線の方向も変える
(鳥のように動きながら、何かに着目する)



＜カメラの画角＞

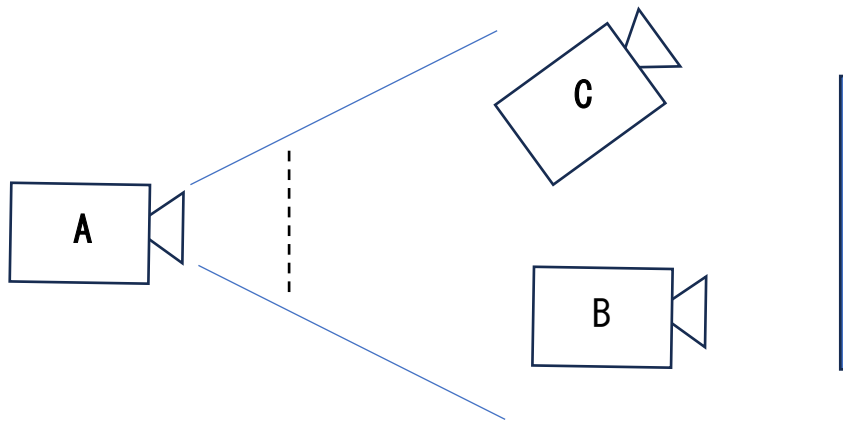
カメラはレンズサイズによって、切り取られる範囲が異なる。望遠レンズであればあるほど、狭い範囲しか切り取られない。



極端な例としては、180 度近くの角度を画面の中に取り入れる「魚眼レンズ」もある。

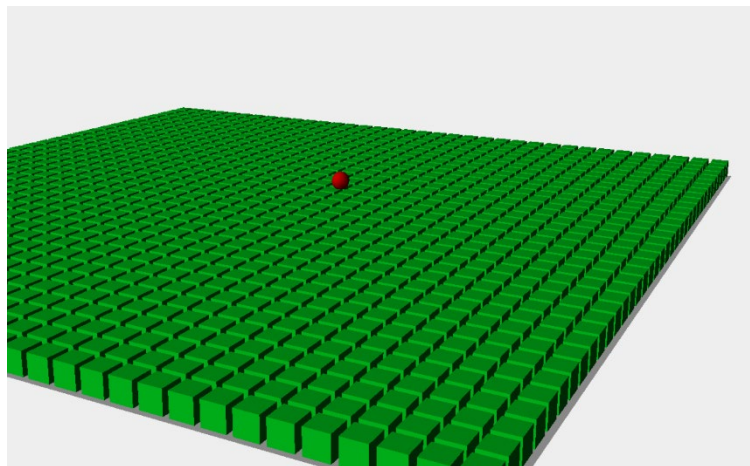


<カメラの向きと範囲>



(同じレンズだとして) **カメラ A** は、遠くに引きすぎているので、オブジェクトの周囲まで映り込む。**カメラ B** は、オブジェクトの一部分だけが映る。これは、カメラ A の位置でも望遠にすることで可能でもある。**カメラ C** はオブジェクトが画角に入っていないので、何も映らない。

また、**カメラ A** で、オブジェクトまたはスクリーンから切り取られる範囲を「**ウィンド**」(window ; 点線部)と呼ぶ。その切り取られた部分を「**クリッピング**」(clipping)と呼ぶ。



<https://oreilly-japan.github.io/learning-three-js-2e-ja-support/chapter-02/08-cameras-lookat.html>

◆ライト



ライト（照明）の要素には、位置、照明の種類、光の強さ、光の減衰率、向き、色などがある。



このガス灯の場合、高さ 3m、点光源、100W（1520 ルーメン）、2m90%曲線、拡散光、昼白色

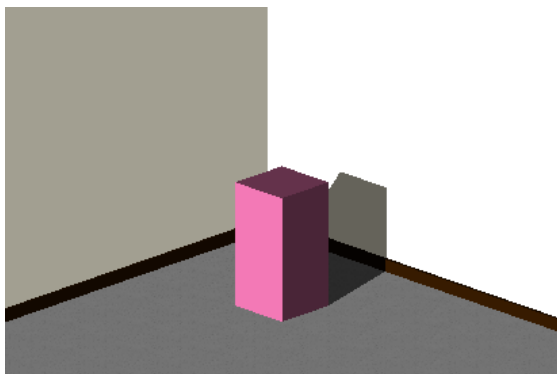
・光源の種類

点光源、線光源、面光源

面光源	点光源
発光面積の大きい光源 幅や長さの大きい光源	発光面積の小さい光源 例…ハロゲン電球
拡散光	指向性の強い光
	
かげと光沢感が得られない	かげと光沢感が得られる

・光源の形状

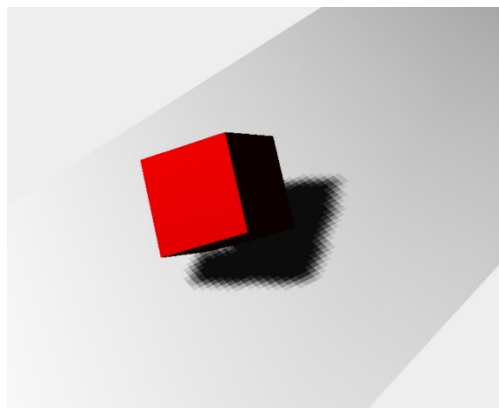
平行光線、拡散光





・アンビエントライト

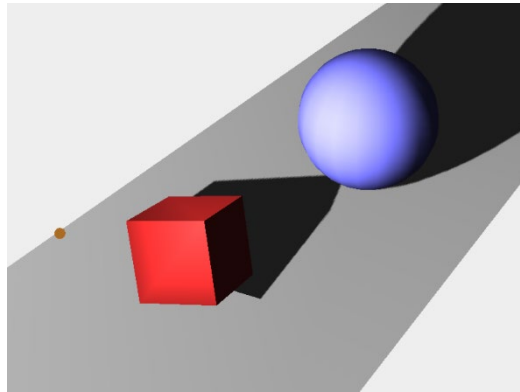
CG では基本的なライトとして「アンビエントライト」(Ambient Light) は、特定の光ではなく、全体を照らす照明。



<https://oreilly-japan.github.io/learning-three-js-2e-ja-support/chapter-03/01-ambient-light.html>

- ・ スポットライト

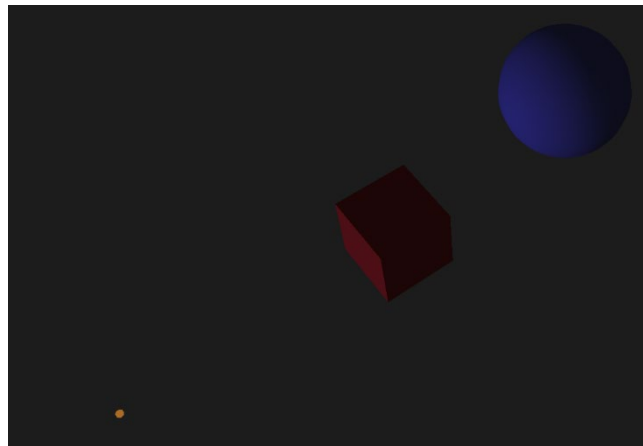
点光源で、方向性を待つ。多くの場合、拡散光。



<https://oreilly-japan.github.io/learning-three-js-2e-ja-support/chapter-03/03-spot-light.html>

- ・ ディレクショナルライト

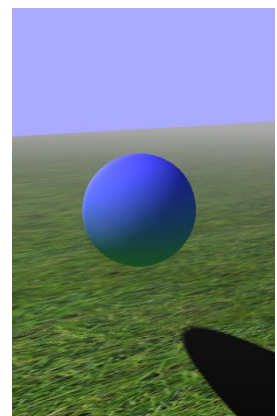
文字通り、方向性を持った光



<https://oreilly-japan.github.io/learning-three-js-2e-ja-support/chapter-03/04-directional-light.html>

- ・ ヘミスフェアライト

Hemisphere Light。天空と地面の2つが作り出す光を柔らかく表現できる。



<https://oreilly-japan.github.io/learning-three-js-2e-ja-support/chapter-03/05-hemisphere-light.html>

■基本コード

◆html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Three.js Example</title>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/three.js/r128/three.min.js"></script
>
</head>
<body>
  <canvas id="myCanvas"></canvas>
  <script src="3dcg.js"></script> <!-- ここに JavaScript ファイルをリンク -->
</body>
</html>
```

◆Three.js コード

ファイル名は 3dcg.js

```
function init() {
  const width = 960;
  const height = 540;

  // レンダラーを作成
  const renderer = new THREE.WebGLRenderer({
    canvas: document.querySelector('#myCanvas')
  });
  renderer.setSize(width, height);
  renderer.setPixelRatio(window.devicePixelRatio);

  // シーンを作成
  const scene = new THREE.Scene();

  // カメラを作成
  const camera = new THREE.PerspectiveCamera(45, width / height, 1, 10000);
  camera.position.set(0, 0, 1000);
```



```
// 箱を作成
const geometry = new THREE.BoxGeometry(500, 500, 500);
const material = new THREE.MeshStandardMaterial({color: 0x0000FF});
const box = new THREE.Mesh(geometry, material);
scene.add(box);

// 平行光源
const light = new THREE.DirectionalLight(0xFFFFFF);
light.intensity = 2; // 光の強さを倍に
light.position.set(1, 1, 1);
scene.add(light);

// 初回実行
tick();

function tick() {
    requestAnimationFrame(tick);

    // 箱を回転させる
    box.rotation.x += 0.01;
    box.rotation.y += 0.01;

    // レンダリング
    renderer.render(scene, camera);
}
window.addEventListener('DOMContentLoaded', init);
```