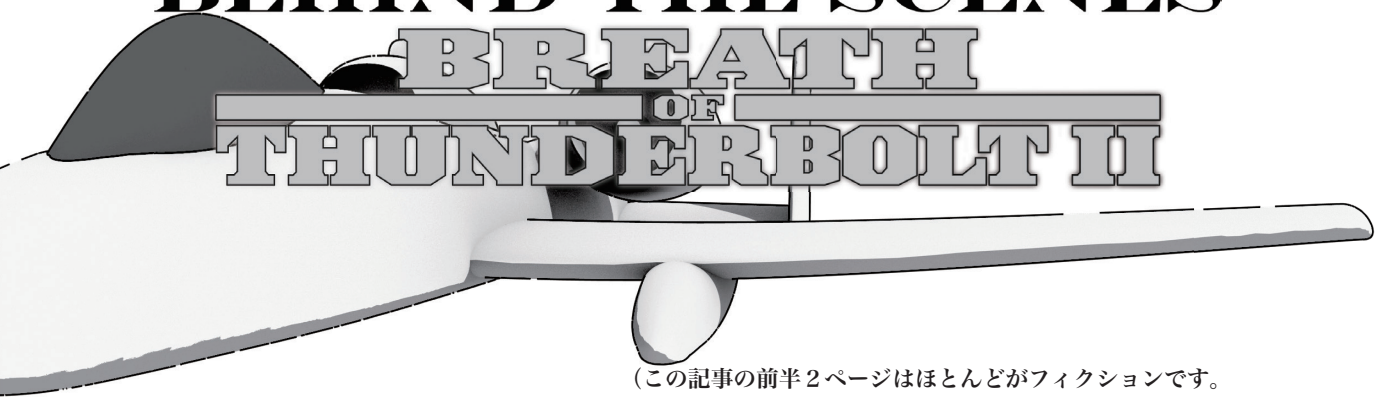
BEHIND THE SCENES



（この記事の前半２ページはほとんどがフィクションです。
実際の企業、団体、国家等とは一切関係ありません）

ストーリー

ある近未来、世界はオーストラリア大陸に墜ちた隕石被害からの復興に手を焼いていた。物語はその隕石墜落災害から一周年の日に始まる。日本・北海道に突如現れた戦車。しかもこの戦車は……おお、見よ！海中から現れたではないか！　日本のレーダー網を、そして対潜哨戒網を掻い潜ったのだ！　あまりにも常識を逸脱した事態を前に、頼りの自衛隊は駆けつけてくれそうにない。正体不明の暴力から人々を守ることができるのは、主人公の OA-10DJ だけである！　彼等はどこから来たのか、何のために来たのか。謎を明かし、世界を守るため、孤独なサンダーボルトが征く――。

OA-10DJ とは

米国での引退が進む「A-10 サンダーボルト Ⅱ」を日本で買い取り、Mitsubishi-Fairchild Inc. が様々な近代化改修を施した前線航空管制機である。目的はあくまで自衛隊での航空管制補助――即ち E-767 航空管制機の補助――であったが、Ｃ４Ｉシステムの拡充を受けてここでも引退。それを日本国内の民間軍事会社で買い上げ、再武装を施した。

21 世紀の今でも、伝説の戦車撃破王ハンス・ウルリッヒ・ルーデルの助言を受けて開発されたサンダーボルト Ⅱ が魂の輝きを失うことはない。それはルーデルがこの世に残した、陸の安寧を脅威から護る鋼の守護神なのだ。

プレイヤーが搭乗する OA-10DJ が搭載するエンジンは、通常のエンジンとは似て非なるものである。このエンジンは、日本での近代化改修時に試作されたものだ。名を、XTF34-11000 という。「11000」とは、N_2 一万一千回転までキッチリ回せ（注１）という技術者たちのメッセージだ。オリジナルよりも出力に余裕のある設計の当エンジンは、その低燃費性能から機体の作戦可能時間延長を期待された（注２）。しかしながら、この新エンジンが量産されることは無かった。燃費の良さでは群を抜いていたが、噴射速度に劣る高バイパス比ターボファンと前例のない超多段中圧圧縮機の信頼性が嫌われたのだ。十分な信頼性を持った従来型エンジンとの比較検討用に、当て馬同然に開発されたとも言われている。そんな不遇のエンジンを２機搭載した、OA-10DJ の実験機版とも言えるのが本機である。

注１）TF34 の後継機エンジン TF39 の民間版の一つである CF6-80 の N2 回転数は 9827、これと比べると実際高回転だ。

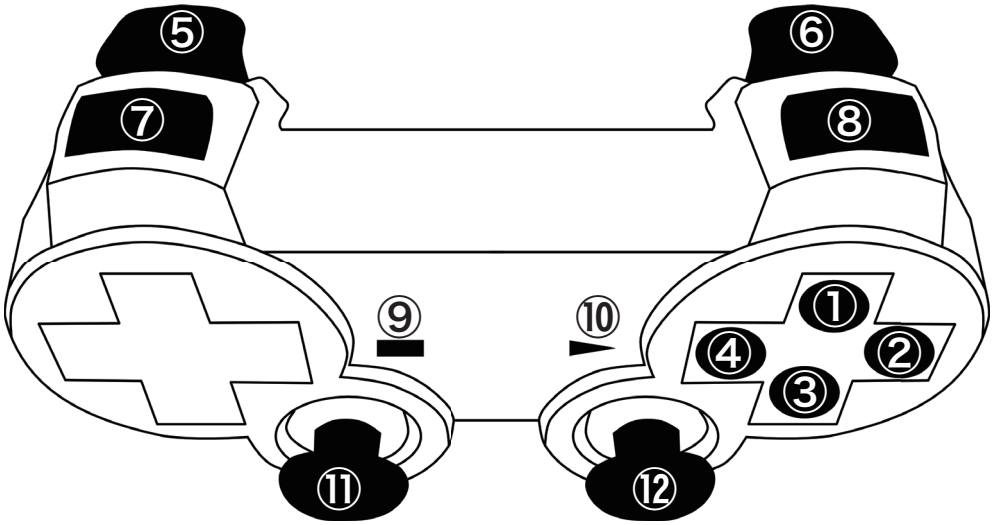
注２）E-767 のような AWACS の弱点の一つは、１機が管制可能な時間があまり長くないことにある。

製作動機（インタビュー・ウィズ・センシャより一部抜粋）

――希代のアンタイ・タンク・ヒーローとして認知された、我らがサンダーボルトⅡについて。「最初に生まれたのは mqo ファイルだった。A-10 のモデル。センシャを殺す。知っての通り、センシャ達は陸上を支配する半神のような存在。それを殺すなんて、なんたる反骨の精神か、とね。そこから全て始まった」

「まずサンダーボルトⅡと、あの恐ろしい復讐者のガトリング砲が生まれた。コウチ＝サンがチョーフから送って来た exe ファイルを見て、私は稲妻に打たれたようになった。なにかが私たち 2 人の脳に降りて来たような感覚だった」

「すぐに、彼の永遠のライバルであるレッドセンシャ（編注：T-72 か）やツングースカが生まれた。昔から彼らヴィランを知っていたような感覚だったよ。とても、とても不思議だった」



操縦方法

本ゲームを起動したその瞬間から、プレイヤーは皆が A-10 神に仕える司祭である。そして、一部の者は未だ A-10 学校どころか、飛行機学校に足を踏み入れたことのないヒヨッコであろう。よって、司祭の常識として、以下にサンダーボルトⅡを操る術を記そう。

言うまでもないことだが、**左スティックが操縦桿**である。基本の操作はまずここからだ。

①**ロックオン切り替え**：標的を切り替えることが出来る。的確に狙いを定め、迅速にこれを撃破すべし。

③**機銃発射**：アヴェンジャーを撃ちこむ。これが君達が扱う唯一にして最高の武器である。

④**広域地図表示**：地図を見て、戦況を見極めよ。広い視野を持つことが勝利の鍵だ。

⑤⑥**減速／加速**：敵との相対速度を冷静に合わせよ。速すぎれば敵を撃てず、遅すぎれば自分が撃たれる。

⑦⑧**左／右ヨーイング**：機体を水平方向に旋回させる。目標との水平位置を正確に合わせる際に活用せよ。

⑩**ポーズ**：所謂一時停止である。

⑪**自動速度制御**：パイロットの指定した目標速度に合わせ、自動で加減速する。
減速／加速ボタンを押すことで、目標速度は変更することも出来る。

⑫**視点切替**：HUD 視点・後方視点を切り替えることが出来る。汝の為したいように為すが良い。

ボタン配置は全てが変更可能だ。操作しやすいように各種ボタン配置を切り替えるのも良いだろう。

「マニューバの秘訣は３つのＫ！　気合い！　気合い！！　気合い！！！　それで AllRight ！！」

……醒めちまったこの会誌に……

……熱いのは……

俺達の MAKING……

3D の技術周りについて

ここでは、このゲームに使われている 3D 計算の技術について書くことにします。

多少、幾何学や線形代数の考え方が必要になるかもしれないので、そこはご容赦下さい(岡部いさく風)。

ロール・ピッチ・ヨーを得る

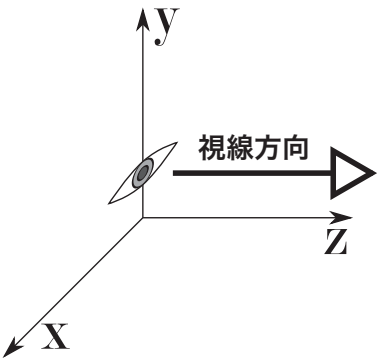
飛行機の操縦で重要な角度（車でも重要だったりするけど)、**ロール角・ピッチ角・ヨー角**は基本的に画面に表示しなければならないでしょう。ですが、これはそれほど容易なことではないのです。

3D での回転表現については、私の 2 年前の記事に詳しく書いたので、そちらを参照して欲しいです。PDF 版が X680x0 同好会のページから閲覧できるので、URL を載せておきます。[1]

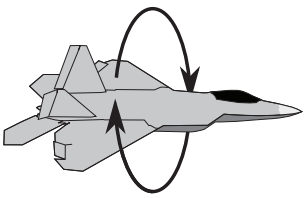
このゲームでは、機体がどのような回転をしているかという情報は、全て四元数で管理しています（四元数については [1] の 3 章をご覧下さい)。

このままでは画面に反映されないため、オイラー角（カルダン角）に変換してモデルの回転状態を与えます（カルダン角については [1] の 2 章を（ry、カルダン駆動とは関係ないので注意)。この四元数→オイラー角の変換と、四元数→ロール・ピッチ・ヨー角の変換は、実は本質的には一緒です。

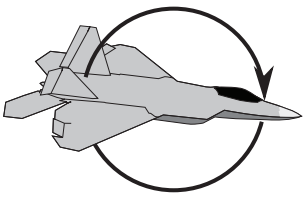
座標系はどちらでも同じ結果となるのですが、仮にプログラミングでよく用いられる左手系を使いましょう。あなたは、Z 軸方向を向いているとします。その時に、右方向が X 軸の正で、上方向が Y 軸の正です。この状態で、Z 軸回転→ X 軸回転→ Y 軸回転を行うオイラー角の変換がそのままロール・ピッチ・ヨーとなります。Z 軸の回転角が自機の傾き（ロール）を表し、X 軸の回転角が自機の仰角･俯角（ピッチ)、Y 軸の回転角が自機の向いている方角（ヨー）を表していることになるわけです。カルダン角が X 軸回転→ Y 軸回転→ Z 軸回転を行うオイラー角の変換ですから、ほとんど一緒ということが分かると思います。(左手系なので、Z 軸の回転角が正の時は左にバンクしている状態で、X 軸の回転角が正の時



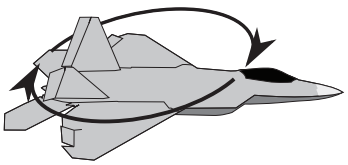
左手系



ロール方向の回転



ピッチ方向の回転



ヨー方向の回転

はダイブする状態、Y 軸の回転角が正の時は面舵（注３）を取っている状態になります）

考え方は、変換元と変換先の両方を 4×4 の行列にしてしまい、成分の比較を行う方程式を解くというものです。

それでは実際に計算してみましょう。

四元数は任意軸回転行列を用いることで行列にすることが出来ます。[1] の３章と４章を見て下さい。まず、単位ベクトル $\vec{u}$ を中心に θ 回転させる四元数 q を考えます。

$$q = [q_w; (q_x, q_y, q_z)] \tag{1}$$

$$\begin{aligned} q_x &= u_x \sin\frac{\theta}{2} && (2)\\ q_y &= u_y \sin\frac{\theta}{2} && (3)\\ q_z &= u_z \sin\frac{\theta}{2} && (4)\\ q_w &= \cos\frac{\theta}{2} && (5) \end{aligned}$$

この四元数を式（1）のように表すとき、成分は式（2）～（5）のようになります。

これを [1] の式（26）に代入すると、

$$R_Q = \begin{bmatrix} {u_x}^2(1-\cos\theta)+\cos\theta & u_xu_y(1-\cos\theta)+u_z\sin\theta & u_xu_z(1-\cos\theta)-u_y\sin\theta \\ u_xu_y(1-\cos\theta)-u_z\sin\theta & {u_y}^2(1-\cos\theta)+\cos\theta & u_yu_z(1-\cos\theta)+u_x\sin\theta \\ u_xu_z(1-\cos\theta)+u_y\sin\theta & u_yu_z(1-\cos\theta)-u_x\sin\theta & {u_z}^2(1-\cos\theta)+\cos\theta \end{bmatrix} \tag{6}$$

となります。この回転行列はベクトルに対して右から掛け算をするタイプです。もし左から掛け算をしたい場合は、転置をして下さい。

次に、カルダン角の変換行列（X 軸回転→ Y 軸回転→ Z 軸回転）を計算します。行列を右から掛け算するので、[1] の式（7）と同じです。これを [1] の式（4）～（6）を代入すると、

$$R_E = \begin{bmatrix} \cos\theta_y\cos\theta_z & \cos\theta_y\sin\theta_z & -\sin\theta_y \\ \sin\theta_x\sin\theta_y\cos\theta_z-\cos\theta_x\sin\theta_z & \sin\theta_x\sin\theta_y\sin\theta_z+\cos\theta_x\cos\theta_z & \sin\theta_x\cos\theta_y \\ \cos\theta_x\sin\theta_y\cos\theta_z+\sin\theta_x\sin\theta_z & \cos\theta_x\sin\theta_y\sin\theta_z-\sin\theta_x\cos\theta_z & \cos\theta_x\cos\theta_y \end{bmatrix} \tag{7}$$

となります。

この式（6）と（7）の右辺が一致するので、各成分から次のようになります。

$$\begin{aligned} \tan\theta_x &= \frac{\sin\theta_x\cos\theta_y}{\cos\theta_x\cos\theta_y} = \frac{R_{E23}}{R_{E33}} = \frac{R_{Q23}}{R_{Q33}} = \frac{u_yu_z(1-\cos\theta)+u_x\sin\theta}{{u_z}^2(1-\cos\theta)+\cos\theta} && (8)\\ -\sin\theta_y &= R_{E13} = R_{Q13} = u_xu_z(1-\cos\theta)-u_y\sin\theta && (9)\\ \tan\theta_z &= \frac{\cos\theta_y\sin\theta_z}{\cos\theta_y\cos\theta_z} = \frac{R_{E12}}{R_{E11}} = \frac{R_{Q12}}{R_{Q11}} = \frac{u_xu_y(1-\cos\theta)+u_z\sin\theta}{{u_x}^2(1-\cos\theta)+\cos\theta} && (10) \end{aligned}$$

この時、式（2）～（5）から次のことが得られます。

$$u_xu_y(1-\cos\theta) = \frac{q_xq_y}{\sin^2\frac{\theta}{2}}(1-\cos\theta) = 2q_xq_y \tag{11}$$

$$\begin{aligned} &\because \cos\theta = \cos^2\tfrac{\theta}{2} - \sin^2\tfrac{\theta}{2} = 1 - 2\sin^2\tfrac{\theta}{2} \\ &\Leftrightarrow \sin^2\tfrac{\theta}{2} = \tfrac{1}{2}(1-\cos\theta) \end{aligned}$$

注３）船舶において三時方向・スターボード（右舷方向）へ舵をとること。

同様にして、

$$u_y u_z(1-\cos\theta) = 2q_y q_z \tag{12}$$
$$u_x u_z(1-\cos\theta) = 2q_x q_z \tag{13}$$
$$u_x{}^2(1-\cos\theta) = 2q_x{}^2 \tag{14}$$
$$u_z{}^2(1-\cos\theta) = 2q_z{}^2 \tag{15}$$

次に、

$$u_x \sin\theta = \frac{q_x}{\sin\frac{\theta}{2}} = 2q_x q_w \tag{16}$$
$$\because \sin\theta = 2\sin\frac{\theta}{2}\cos\frac{\theta}{2} = 2\sin\frac{\theta}{2}q_w$$

同様にして、

$$u_y \sin\theta = 2q_y q_w \tag{17}$$
$$u_z \sin\theta = 2q_z q_w \tag{18}$$

最後に、

$$\cos\theta = \cos^2\frac{\theta}{2} - \sin^2\frac{\theta}{2} = q_w{}^2 - (q_x{}^2 + q_y{}^2 + q_z{}^2) \tag{19}$$
$$\because q_x{}^2 + q_y{}^2 + q_z{}^2 = \sin^2\frac{\theta}{2}(u_x{}^2 + u_y{}^2 + u_z{}^2) = \sin^2\frac{\theta}{2}$$

以上より、X軸回転→Y軸回転→Z軸回転で表されるオイラー角へ四元数から変換を行う場合は、式(8)～(10)に式(11)～(19)を代入して、

$$\theta_x = \tan^{-1}\left(-\frac{2(q_y q_z + q_x q_w)}{-q_x{}^2 - q_y{}^2 + q_z{}^2 + q_w{}^2}\right) \tag{20}$$
$$\theta_y = \sin^{-1}\left(2(q_y q_w - q_x q_z)\right) \tag{21}$$
$$\theta_z = \tan^{-1}\left(-\frac{2(q_x q_y + q_z q_w)}{q_x{}^2 - q_y{}^2 - q_z{}^2 + q_w{}^2}\right) \tag{22}$$

となります。同じように、ロール・ピッチ・ヨー回転を得る変換を考えます。Z軸回転→X軸回転→Y軸回転なのですから、

式(6)と(23)の右辺が一致することから、

$$R_{E2} = \begin{bmatrix} \sin\theta_x\sin\theta_y\sin\theta_z + \cos\theta_y\cos\theta_z & \cos\theta_x\sin\theta_z & \sin\theta_x\cos\theta_y\sin\theta_z - \sin\theta_y\cos\theta_z \\ \sin\theta_x\sin\theta_y\cos\theta_z - \cos\theta_y\sin\theta_z & \cos\theta_x\cos\theta_z & \sin\theta_x\cos\theta_y\cos\theta_z + \sin\theta_y\sin\theta_z \\ \cos\theta_x\sin\theta_y & -\sin\theta_x & \cos\theta_x\cos\theta_y \end{bmatrix} \tag{23}$$

となります。

$$\theta_x = \sin^{-1}\left(2(q_x q_w - q_y q_z)\right) \tag{24}$$
$$\theta_y = \tan^{-1}\left(-\frac{2(q_x q_z + q_y q_w)}{-q_x{}^2 - q_y{}^2 + q_z{}^2 + q_w{}^2}\right) \tag{25}$$
$$\theta_z = \tan^{-1}\left(-\frac{2(q_x q_y + q_z q_w)}{-q_x{}^2 + q_y{}^2 - q_z{}^2 + q_w{}^2}\right) \tag{26}$$

お疲れ様でした。簡単に結論だけまとめてしまいましょう。四元数→オイラー角の変換は、式(20)～(22)を使いましょう。オイラー角を使った変換方法を用いるライブラリはとても多い(注4)ので、この計算は意外と多用すると思います。

注4）例えば、DXライブラリのMV1SetRotationXYZという関数にはこのオイラー角を与える必要があります。

さらに、機体のピッチ・ロール・ヨーを得たいときは、式（24）～（26）を使いましょう。θ_xがピッチ角、θ_yがヨー角、θ_zがロール角になります。

3D の座標から自力で 2D の座標を得る

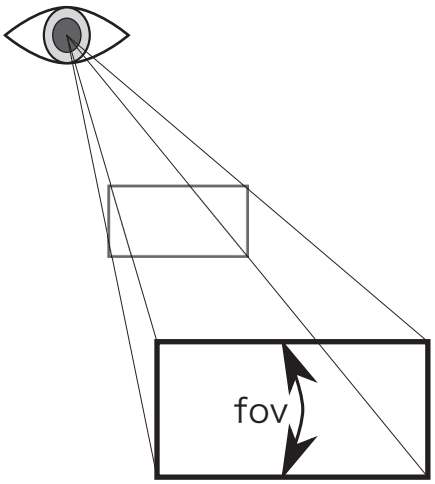
HUD などを実装する場合、3D モデルを描画している上で、そのモデルと対応するような画像を画面上に 2D で表示したいという状況が生まれます。たとえば、敵の周りを四角く囲いたい、敵が画面上に表示されているかを知りたい、というときです。そういう場合には、ライブラリなどで 3D モデルが画面に表示される工程を、自分で行わなければなりません。

前提として、モデルがどの位置に・どのような回転状態で・どのような大きさで表示されるかを表す変換行列によって、3D モデルの各頂点が変換されますが、これは省略します。この変換で、3D 空間上での座標が得られます。その座標が画面に映るときに、どこに表示するかをこれから計算してゆきましょう。

まずは、その座標がカメラから見てどの位置にあるかを計算します。つまり、カメラからの相対座標です。カメラの基準座標の取り方・3D 空間の座標の取り方などは、後述の Projection 行列・ViewPort 行列をどのように作るかで幾らでも変更できます。しかし、ここでは説明のために定義することにします。本質さえ理解してしまえば、右手系・左手系の違いなどは些細なものだと感じるようになるでしょう。Projection 行列をライブラリから得るような場合でも、その行列がどのような変換を行っているかが分かれば、それに合うように適宜正負を反転させていくだけで対応できるでしょう。

左手系座標において、カメラが Z 軸の正の方向を向いており、上が Y 軸の正の方向で、右が X 軸の正の方向を向いていることにします。これを Projection 行列は、X-Y 平面上にカメラの写しているものを射影するものと定義します。

このとき必要になるのは、視野角（Field of View）です。この視野角というのは、カメラが写すことができる角度を示します。たとえば、鳥の目などは広角で広い視野角を持っています。この視野角 *fov* は普通垂直方向についての場合が多いです。一般的な視野角としては、45 度が用いられています。



視野角

そのまま縦横同じ視野角を用いてしまうと、正方形な範囲しか得られず、ワイドな画面に射影するときに横に伸びてしまいます。そこで、画面の縦と横の比率であるアスペクト比 aspect が必要になります。これは普通、横幅を高さで割ったものになります。16：9 のワイドディスプレイでは、この値は 16/9=1.777……になります。

さらに、どの範囲の距離（正確には距離ではなく Z 軸方向での遠さ）を表示するかを決める必要があります。最も近い点Z_nと最も遠い点Z_fです。この二つの値は、3D の座標系がどのようなスケーリングで用いられているかによるため、非常にまちまちです。

以上の値から Projection 行列を作ってみましょう。視野角の直線上に存在する場合は、-1.0 か 1.0 となるようにすれば良いことになります。二等辺三角形になるわけですから、Z で割り算してしまえばよいわけです。Z 軸に関しても範囲内かどうかという変換（さらに言うと、そのモデルが他のモデルと比べてカメラと近いか遠いかを判断することにも用いられます。 この値を並べたものを Z バッファなどと呼び、

モデルの描画順に関与します）が必要となるため、ベクトルの W 成分にベクトルの Z 成分を格納し、ベクトル全体を W で割るという方法が用いられます。

このような変換は次のようにあらわすことができます。

$$x' = \frac{x}{aspect \tan\left(\frac{fov}{2}\right)} \tag{27}$$

$$y' = \frac{y}{\tan\left(\frac{fov}{2}\right)} \tag{28}$$

$$z' = \frac{(z - Z_n)Z_f}{Z_f - Z_n} \tag{29}$$

$$w' = z \tag{30}$$

このとき（*x′/w′*, *y′/w′*）= (-1, -1）となる場所がカメラにとっての左上になり、（*x′/w′*, *y′/w′*）= (1, 1）が右下になります。さらに、*z′/w′* = 0 となる場所が画面に映る最も近い場所であり、*z′/w′* = 1 は最も遠い場所になります。

式（27）～（30）をベクトルに対して右から掛け算をするような行列で表現する場合は、次のようになります。

$$\begin{bmatrix} x' & y' & z' & w' \end{bmatrix} = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{aspect \tan\left(\frac{fov}{2}\right)} & 0 & 0 & 0 \\ 0 & \frac{1}{\tan\left(\frac{fov}{2}\right)} & 0 & 0 \\ 0 & 0 & \frac{Z_f}{Z_f - Z_n} & 1 \\ 0 & 0 & -\frac{Z_n Z_f}{Z_f - Z_n} & 0 \end{bmatrix} \tag{31}$$

これが Projection 行列そのものになります。

式（31）で変換されたベクトルから画面の座標に変換してみましょう。上記のように、（*x′/w′*, *y′/w′*）= (-1, -1）が画面の左上になり、（*x′/w′*, *y′/w′*）= (1, 1）が右下となるため、画面の幅を width・高さを height とすると、次で計算できます。

$$x_d = \frac{\left(1 + \frac{x'}{w'}\right) width}{2} \tag{32}$$

$$y_d = \frac{\left(1 - \frac{y'}{w'}\right) height}{2} \tag{33}$$

式（32）(33）を 4×4 行列に拡張したものが ViewPort 行列になります。このとき、Z 成分と W 成分はそのまま渡します。

$$\begin{bmatrix} x_d & y_d & z_d & 1 \end{bmatrix} = \begin{bmatrix} \frac{x'}{w'} & \frac{y'}{w'} & \frac{z'}{w'} & \frac{w'}{w'} \end{bmatrix} \begin{bmatrix} \frac{width}{2} & 0 & 0 & 0 \\ 0 & -\frac{height}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \frac{width}{2} & \frac{height}{2} & 0 & 1 \end{bmatrix} \tag{34}$$

なかなかシンプルだったでしょう。

（文責：T08　河内）

――――――！とても宇宙的な！――――――

――――――！初心者向な記事催す！――――――

２Ｄ／３Ｄグラフィックについての Tips

予め申し上げておきますが、私の書く記事の全ては少し Google を使えば簡単に調べられる内容です。さて、ここで主役として扱うアプリケーションは２つ。

- **Javie**
- **Blender+Freestyle**

まず、Javie は本記事の筆者がマカーの貧乏人であることから使っている、**映像編集ソフトウェア**です。断言します。ほとんどの人はこれをわざわざ選ぶ必要はありません。Windows にはフリーウェア NiVE という選択肢がありますし、また十分な資金力がある場合は映像編集ソフトウェアの最高峰の一つである Adobe After Effects（以下 AE）を選ぶことが出来ます。**本記事で記述した全てのことは NiVE や AE でならより簡単に、より短時間で行うことが出来る**でしょう。そもそもエフェクトを作るならそれに特化したアプリケーションが存在します。しかし今回は Javie を前提として記述します。筆者がマカーなので。

Blender は **3D コンピューターグラフィックスソフトウェア**です。フリーウェアながら非常に強力な機能を取り揃える一方で、操作性には強い癖があります。実際、筆者ごときでは到底使いこなせるものではなく、本記事では**既存のモデルを見た目よく出力するためだけに使われる存在**です。今回はそのためにアドオンの Freestyle を使用しています。

ほのおのにおいしみついて　Javie

ちょっとだけむせるんじゃ。Javie は、今回エフェクト製作のために活躍してもらいました。完成図は図１のような爆発画像となります。実際のゲームでは、敵機が爆発四散するときにこのような画像が多く散らばる形で使用されています。細かい説明はともかく、まずは図２のようにコンポジションを新規作成しましょう。

図１：ゲーム中と同じタイプの画像

続いてコンポジションの設定に関するウインドウが表示されます。「幅：」を 480 にし、名前は適当にわかりやすくつけましょう。そして、「デュレーション」には初期状態で「0;01;00;02」という値が入っているはずですから、これを「1」に書き換えます。残りの項目は今回関係ないので放置。そして「Finish」を押します。

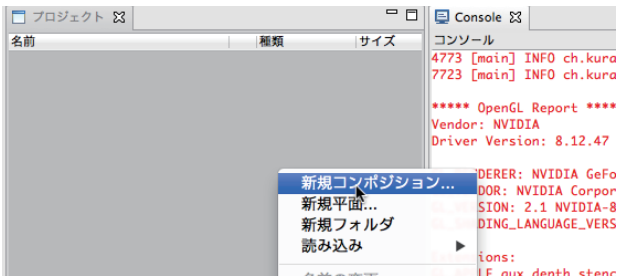


図２：画面左上部の空白を右クリックして、コンポジションを作成

丁寧な解説書ならここで各種画面の説明とかするんでしょうが、スペースが勿体無いので適当に説明しましょう。画面右上が

ビデオ画面、左上が素材一覧、下半分がタイムライン。基本的には左上に D&D してインポートした素材をタイムラインに投げ込んでアレコレすることで、編集は進みます。

コンポジション作成時と同様にコンテキストメニューを開き、上から２番めの「新規平面」を選択しましょう。サイズは今回はコンポジションと同じ 480×480、カラーはオレンジ色としましょう。この色が後々爆発の色を決めますから、悩んでもいいでしょう。今回は写実的爆発が必要なのでオレンジ色としますが、例えばアヤシイ緑色の粒子の爆発が欲しい時とか、ニーズに合わせて変更はしていくべきですね。

先ほど作った平面を２度、タイムラインに投げ込みます。続いて、２つあるうち上の平面の（ここではオレンジ色の）四角の左側にある三角（▶方向）を押します。内容が展開されて（▼方向）、「エフェクト」「トランスフォーム」が並ぶはずです。「エフェクト」を右クリック、図３のように「エフェクト」「ノイズ＆グレイン」「フラクタルノイズ」と選びましょう。黒白の煙っぽいものが見えるようになったはずです。

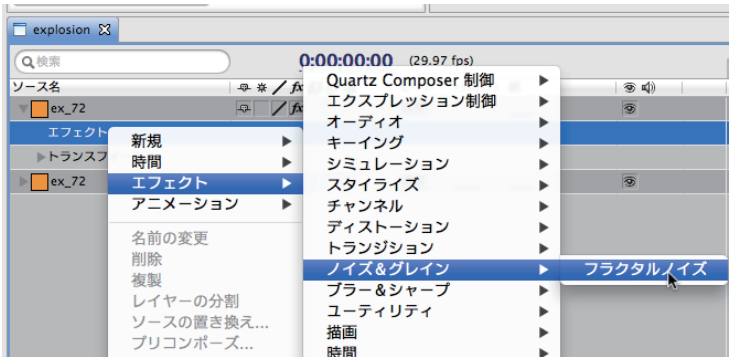


図３：フラクタルノイズを上側のレイヤーへ

続いて先ほどフラクタルノイズを適用した平面の「モード」を「通常」から「リニアライト」へ（図４）。オレンジ色の光る、ちょっと爆発っぽくなった煙が見られるでしょうか。さあ、一気に調整して仕上げてしまいましょう。

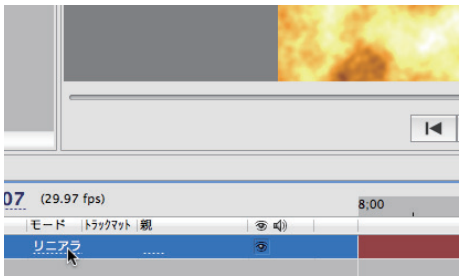


図４：「リニアライト」だけでちょっとそれっぽく？

タイムラインには「▶フラクタルノイズ」という表記が見えるはずです。先ほどと同様に▶をクリック。すると、フラクタルの種類・反転・コントラスト・明るさ……とメニューが並びます。このへんも調整すれば色々と変化を楽しめますが、今回は「コントラスト」「明るさ」「トランスフォーム／オフセット」のみを弄ることとしましょう。大雑把に方針を述べますと、１。150 ～ 200 程度にコントラストを上げる。２。10 前後に明るさを上昇させる。３。「それっぽい形の爆炎」を探し当てられるまでオフセットを動かす。以上です。いくつか手順を増やすと３を省くことが出来る場合がありますが、ここでは扱いません。

タイムライン上の Shift を押しながらの選択で２つとも選んでやり、コンテキストメニューから「プリコンポーズ」を押します。適当に新規コンポジションの名前を与え、「OK」。これを行った時点で、色調以外の調整はほとんど不可になるとお考えください。「これで良いな」と納得できる段階で実行しましょう。

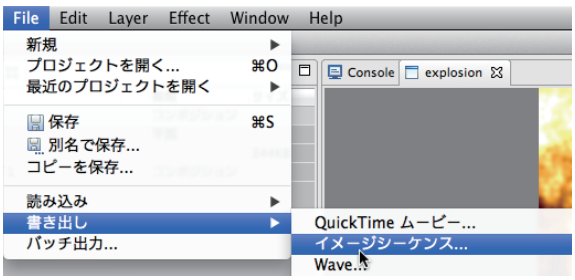


図５：Javie 上の作業はこれでおしまい

では、「File」「書き出し」「イメージシーケンス...」と選び、適当にファイル名を付けて書き出しましょう（図５）。この時、ファイル名は「（任意）**.png**」という形で、png までをセットで記述しましょう。さもないと、ビットマッ

プで吐き出されてしまいます。ビットマップは透明を扱えませんので。

では、作業の場を Gimp に移します。もちろん、利用可能ならば Photoshop がベターです。ペイント？キミは帰りなさい。ここでは Gimp の使い方は既知であるとします。一応図は掲載しますが。Gimp で画像を開き、図６のように自由選択ツールで爆発の概形を切り取りましょう。この時、暗い部分をなぞるようにすると良いでしょう。また、この時必ず「境界をぼかす」をオンにしましょう。「半径」の値は 40 以上で適当に取りましょう。作った境界で画像をコピーし、「ファイル (F)」「画像の生成 (C)」「クリップボードから (T)」と選択。切り取られた爆発部分だけを取り出せるでしょう。これを保存すると、図１のようになるはずです。これで完了です。お疲れ様でした。

でも、普通の Windows ユーザーならこんなことしてないで、エフェクト作成用ソフト使いましょうね？

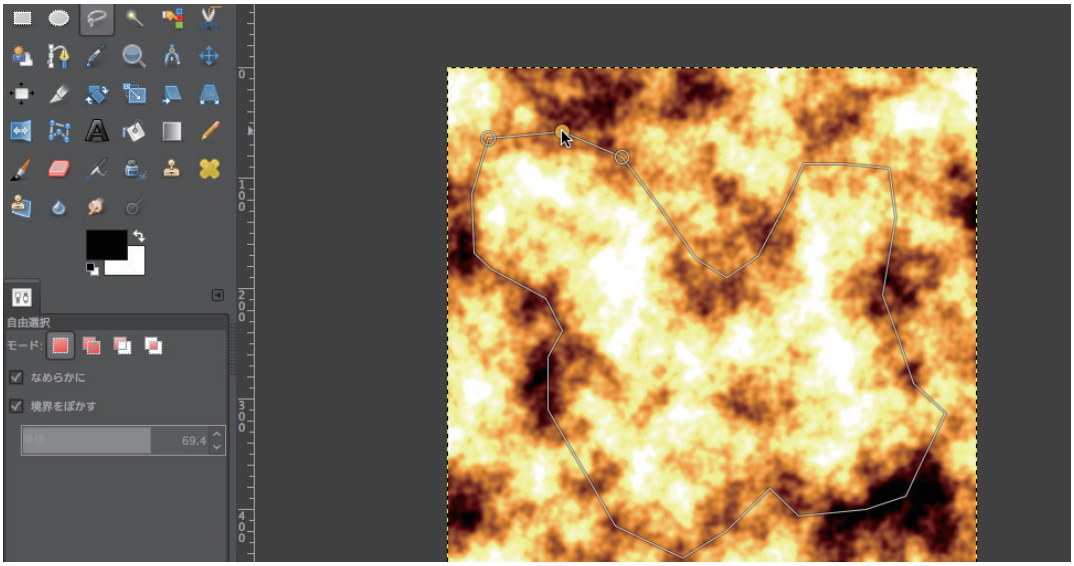


図６：このように　きりとるのだ

アニメじゃない　アニメじゃない　不思議な Blender

3D のことさ。さて、当記事ではアプリケーションの日本語化とトゥーン風画像作りのみに触れます。操作に関してはあまり詳しく説明しません。公式運営の Blender Wiki ならばほぼ確実に詳細で正確な Blender についての情報が入手できます。操作やテクニックなどのわかりやすさを重点する場合は「CG 制作」様の Blender の項（http://cg.xyamu.net/Blender/）の閲覧をオススメします。それでもわからない場合はググる。

日本語化

では、Blender を起動。ウインドウ左下の立方体のボタンを押し、「User Preferences」を選択します（図７）。「Interface」「Editing」と並ぶタブから「System」を選択し、画面中央から少し右下に位置するであろう「International Fonts」のチェックを入れます。続いて「Language:」からなんとか「Japanese（日本語）」を探し出し選択、また「Translate:」下の

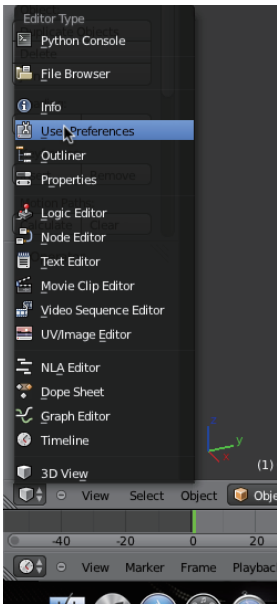


図７：「環境設定」へ

「Interface」「Tooltips」「New Data」も全て押下しましょう（図８）。そして、ウインドウ左下の「ユーザー設定の保存」をクリック（図９）。これで日本語化は完了です。

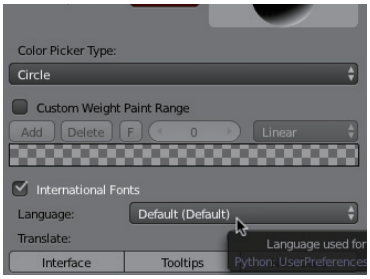


図８：言語設定を変える

以上の手順により行った日本語化は多くの場合はユーザーに分かりやすさをもたらします。一方で、Blender の発展的使い方を求めた場合、そのテクニックは多くが英語で記述された文章や動画の中にあります。逆に言うと、日本語の文献はあまりありません。ゆえに、下手に日本語化を行うとかえって何が起こっているのかわからなくなる場合もあります。この際は、先ほど図８で設定した「International Fonts」（日本語では「ローカライズ」）のチェックを外し、英語に戻しましょう。

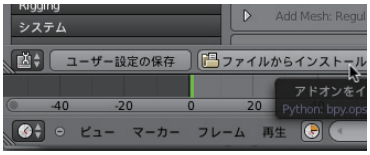


図９：ユーザー設定の保存をお忘れなく

すごく適当な仕組みのお話

今回取り上げる Freestyle は、モデルの輪郭線を描いてくれるアドオンです。実は、Blender にはアニメ塗りっぽいことをする機能は昔から存在する一方、アニメっぽい線を描画することは簡単ではありませんでした。そして、この線の有無が見栄えに大きな差を及ぼします。例を挙げましょう（図 10）。今回は、ある自動車のタイヤ部分をトゥーンレンダリングしてみました。左半分は Freestyle 未使用、右半分が Freestyle を使用したものです。左側ではホイールの輪郭がまったく見えずわかりづらい。その一方右側ではその線がために輪郭を判別することが出来ます。これが Freestyle の力なのです！

図 10：輪郭線の有無

カメラ調整

モデルは各々でご用意の上、以下の作業をお試しください。モデルの頂点だの辺だのを変更することはせず、ただ絵を作る、撮影することのみを説明します。

今回使用するモデルは戦車「T-72」（図 11）。現代においてはロシアでもっとも有名な戦車であり、やられ役として最も活躍している戦車です。それはフィクション・ノンフィクションを問うことはありません。このゲームでも主敵として大活躍です。さて、視点を回してみましょう。Blender では、マウス中ボタンを押しながらマウスを動かすと、視点を回転させることが出来ます。また、Shift キーを押しながらマウスを動かすと視点を移動させることが出来、マウスホイールで視点を拡大／縮小することが出来ます。これら３つの操作を使って、何はともあれモデルの写りが良い見え方を探してみましょう。完了したら、Ctrl+Alt+ テンキーの 0 を押すか、図 12（次ページ）のように「ビュー」「視点を揃える」「現在の視点にカメラを合わせる」と操作していきます。すると、画面中央部以外が暗くなり、また画面中央の明るく残った部分と暗い部分の境界には破線が引かれることで

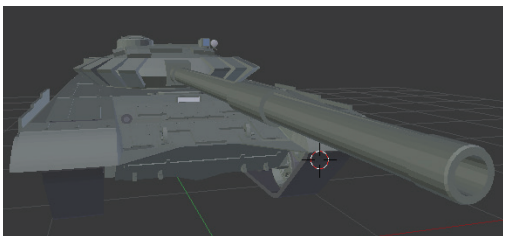


図 11：T-72

しょう。この破線の内側が、レンダリングを行った時に画面に映るものとなります。つまり、視野であります。Shift+F キーで、「フライモード」に入ります。WASD で前後に動く、マウスで視点方向を変えるなど……所謂 FPS ゲームに近い操作感でカメラを操作することが出来ます。左クリックを押すとフライモードは終了します。とりあえず、試しに一枚レンダリングしてみましょう。F12 キーを押してください。

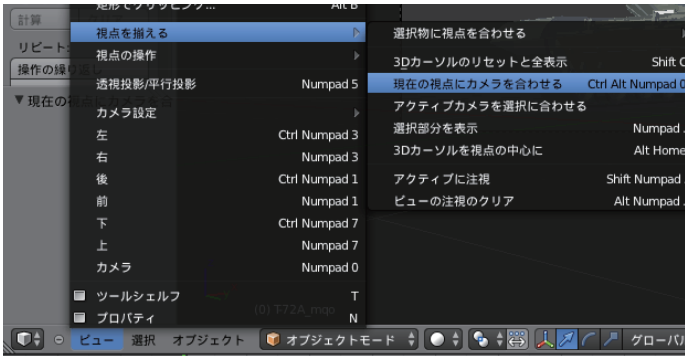


図 12：最もわかりやすいカメラの設定方法

図 13：ほぼ初期状態レンダリング

時間はかからないでしょう。では、出来上がった図を早速……（図 13）暗い！　なんか暗い！　下半分何も見えないじゃん！

レイ・トレースの結果が間違っているとかそんなことはありません。この絵がここまで暗い主な理由は、この戦車を照らす光が戦車左上空に浮かぶポイントライトだけであり、そしてこの照明がかなり強い影を創りだすように設定させていることにあります。では、照明を弄って少し明るい絵を目指してみましょう。

光と闇の調整

Das Marchen des……もとい、まずは現在使われている照明を見てみましょう。視点を引き、図 14 のような照明アイコンを探し出し、右クリックします。続いて、画面右下からランプに関する設定を表示しましょう（図 15）。「▼ランプ」となっているところが見えるでしょうか。すると、ここに「ポイント」が設定されていることがわかると思います。これは豆電球のような照明で、遠くに行けば行くほどだんだん光が減衰する性質があります（ただし、今回の場合は減衰の影響を考えなければならないほど遠くに物体は存在しません）。今回のようにアイコンとして使う、写実性よりもくっきりさが絵に求められる場合ではポイントライトよりサンライトが適するでしょう。こちらには減衰がありませんが、一方で指向性のある光です。ポイントライトは無指向性でした。それと、影も調整しましょう。ランプのメニューの一番下から一個上、「影」という設定項目があります。初期状態では真っ黒の部分をクリックすると、図 16 のように色の設定項目が現れます。RGB に直接値を（0 ～ 1 の間で）入力してもいいでしょうし、目で見ながら色を変更しても良いでしょう。今回の例では、R=G=B=0.152 としました。続いて、ランプの位置と角度も変更しましょう。ランプからは現在、図 15 のように赤・緑・青色の矢印が出ているはずです。この矢印をそれぞれクリックして動かすことによ

図 14：照明アイコン

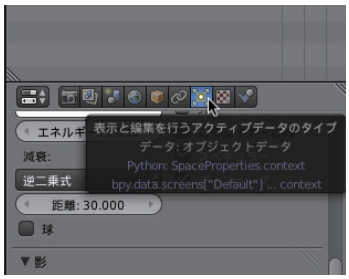


図 15：Lamp 設定

図 16：影色設定

り、XYZ それぞれの方向への照明の位置変更が可能です。また、このままキーボードの R を押す、もしくは図 17 のマウスポインタの位置にある弧のマークを押すことで角度の変更が出来ます。今回はとにかく図としてのわかりやすさ重視ですから、視線とほぼ同じ方向に光が差すように照明を設定しました。

この状態でレンダリングすると、見た目が遙かに良いことが確認できるでしょう。しかし、影が光に対して少しずつ深くなっていっています。当然の事で、これはモデルの材質が現在ランバートというものに設定されているからです。ともかく、このランバートをトゥーンにしてやらぬことには、今回我々が望む 2 階調の結果は得られません。それでは地味な作業を始めましょう。

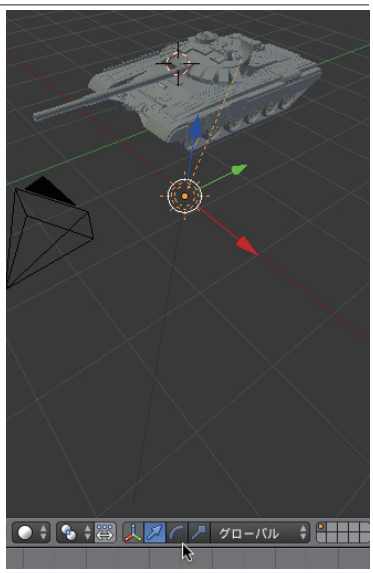


図 17：照明方向設定

材質変更ジゴク

図 18 を御覧ください。この図と同様のものは、一般的な Blender の画面では右側に存在するはずです。「T-72_mqo」の部分には各々で準備したモデルのファイル名が入っているはずです。その左、⊕のようなマークを押すと、モデル内のオブジェクトがずらりと展開されます。それらを選び、画面右下側プロパティーのタブ群から「マテリアル」（図 18 では右から 4 番目の金属球のようなアイコン）を選択します。すると、図では「本体色」とある部分に、そのオブジェクトに属する材質が一覧で表示されます。トゥーン化を行うためにすることは、それらを一つ一つ選び、「▼ディフューズ」とあるところの「ランバート」だの「ミンナート」だのを全てトゥーンにし、サイズを 1.5 ～ 2.0 程度に設定し、スムーズを 0.1 以下に下げること。そして、「▼スペキュラー」とある部分の「強度」を 0 まで落とすこと、この 2 つです。これを、トゥーンに設定されていないディフューズが駆逐されるまで繰り返します。このまま仕上げます。Freestyle による線付けです。

図 18：材質設定

Freestyle

Freestyle はありがたいことになかなか秀逸な初期設定を持ちます。もちろん、Blender の機能である以上、細かい設定もいくらでもできますが。先ほど「マテリアル」を選んだのと同じように、一番左側「レンダー」を選び、その中から Freestyle を探し出し、これにチェックをいれます。続いてその隣のタブ、「レンダーレイヤー」に移りましょう。こちらにも「Freestyle：以下いくつもの設定項目がありますが、最もベーシックな使い方では、その「Freestyle」の中身だけで構いません。この角度より鋭い角には線が描かれるという閾値になる「クリース」、線がうまくつながらない時にオンにするとこれが解消するかもしれない「面のスムーズさ」チェックボックスです。とにかく、レンダリングにかけてみましょう。おや、既にほとんど文句のつけようがありませんね

（図 19）。ということは、あまりにもあっさりですが、このサンプルはここで完成ということでいいでしょう。ここからは、いくつかの Tips をご紹介して、記事をおしまいにしましょう。

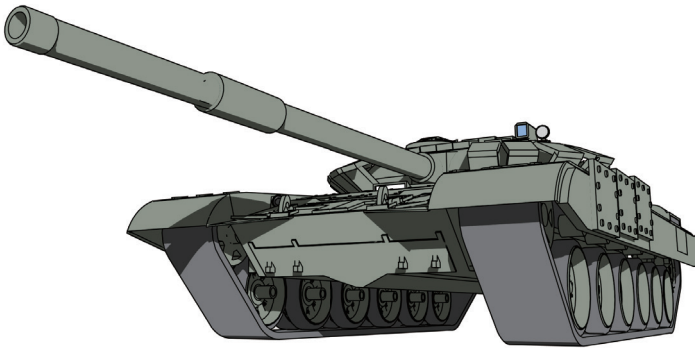


図 19：ほぼ初期状態 Freestyle レンダリング

細かなこと

- Freestyle の線を描く処理は、Blender による「塗り」が完了してから行われます。よって、レンダリング結果にまだ線が描かれていなかったとしても、それは多くの場合不具合でもなんでもなく処理が完了していないだけです。
- 「プロパティ」の「レンダー」には解像度の設定があります。つまり出力サイズの設定です。「縦」「横」「大きさのパーセンテージ」と並びます。「値とか色とか、これでいいのかな」というテストの間はそのパーセンテージを 50% とか、25% とかに絞っておくことをオススメします。色々弄ったモデルのレンダリング結果が、超長時間をかけて悲惨なことになったら辛いですし。
- 「ここ、ちょっとだけカクカクしてるんだけどなんとかならんかな」と思ったら、その部位を右クリックで選択し「シェーディング」の「スムーズ」を押してみましょう（図 20）。改善するかもしれません。ただし、これを有機物に使うと物体が溶ける可能性がありますので注意しましょう。
- スムーズだと溶けるんだけど滑らかにしたい！　という時はモディファイアの辺分離（EdgeSplit）と細分割曲面（Subsurf）が役に立つかもしれません。プロパティにモディファイアー—というタブがあり、ここから設定できます。詳細はおググりください。
- Freestyle の出力結果の画像に、角なんかありゃしないところに断片的に線が描かれた場合は、クリース角度設定の下にある「カリング」チェックボックスをオンにしてみましょう。解決する場合があります。
- 最初から説明しておけ、レベルの話ですが、アドオンを追加することによって Blender は実に様々なモデルを読み込むことが出来ます。今のところ、直接の読み込みが難しいのは Google 3D のモデルぐらいでしょう。例えば、Metasequoia 用のインポート・アドオンはインストールしておいて損することは無いでしょう。またこのインポーターの作者は MMD 用のインポーターまで作っていらっしゃいます。MMD を触る場合は、これも有用です。

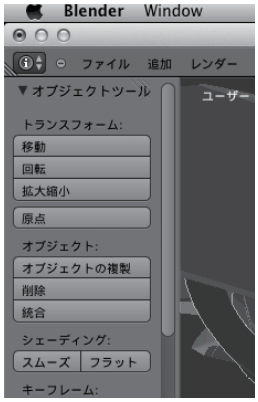


図 20：シェーディング設定

最後として

とっつきにくさに定評のある 3D という世界でもさらにとっつきにくい分類に入る Blender ですが、そのうちの限定的な機能を使う分には私のような無学の人間でもでもなんとかなりました。ゲーム開発も、これからはだんだんと 3D の時代かもしれません。それを先取りできる技術を部分的にでも会得できるなら、それはあった方がかっこいいじゃん。

（文責：C09　古田）