Pixel Round

プロジェクトの数学

ピクセルアートおよびボクセルアート向けの円、楕円、球、楕円体ジェネレータに用いられる数学的基礎についての技術的かつ教育的なドキュメントです。各概念について、形式的定義、幾何学的根拠、およびプロジェクトのソースコードとの直接的な対応を提示します。

取り扱う分野：解析幾何 · 離散ラスタライズ · デジタルトポロジー · 積分学 · 線形代数 · 三角法 · 音声合成。

目次

1. 概要：問題の数学的性質

Pixel Round は **ピクセルアート**および **ボクセルアート**用の丸い形状を生成するツールで、2 次元では円と楕円、3 次元では球と楕円体を扱います。中心となる問題は 離散ラスタライズに属します：実数 $\mathbb{R}$ 上で解析的に定義された曲線あるいは曲面が与えられたとき、それを表現するために規則格子（2 次元ではピクセル、3 次元ではボクセル）のどのセルに印を付けるかを決定する問題です。

この問題に唯一解はありません。包含判定の基準が異なれば異なる離散シルエットが得られ、各基準には独自の数学的根拠があります。そのため本プロジェクトは、3 種類の 2D アルゴリズム、3 種類の描画モード、3 種類の 3D シェーディングスタイルを実装しており、以下の各節で詳述します。

実行はサーバを介さずブラウザ内のみで完結し、その制約から数学的定式化には 計算効率という追加要件が課されます。関連する理論分野は次のとおりです：

- 円錐曲線と二次曲面の解析幾何;
- 整数演算による漸進アルゴリズム（Bresenham）;
- デジタルトポロジー（4-、6-、26- 近傍）;
- 積分学と数え上げ測度;
- 線形代数（切断平面、透視投影）;
- 三角法と球座標;
- 信号合成と基礎音響.

2. 座標系と離散ピクセル格子

キャンバスは $G_x \times G_y$ セルから成る格子です。セル (i, j) は正方形 $[i, i{+}1] \times [j, j{+}1]$ を占めます。連続的な視点では、そのセルの **中心** は $(i + \frac{1}{2},\ j + \frac{1}{2})$ にあります。この約束事は決定的に重要です：円周をセルの 角 (i, j) と比較するか 中心 $(i + \frac{1}{2}, j + \frac{1}{2})$ と比較するかで、形状に属するとみなされるセルの集合が変わるからです。

$$\text{セル } (i,j) \text{ の中心 } = \left(i + \tfrac{1}{2},\ j + \tfrac{1}{2}\right)$$

直径 D の形状に対し、コードは格子を各軸 $D + 2$ セルに拡張します（左右に 1 セルずつの余白）。これにより最外周のピクセル（北、南、東、西）が行列の境界に乗ることがありません。形状の **中心** は $(G_x/2,\ G_y/2)$、半径は $r_x = D/2$ と

$r_y = D/2$（楕円のときは $W/2$ と $H/2$）です。

3. 基本方程式：円と楕円

プロジェクトの全理論は、原点中心・半径 r_x, r_y の楕円の陰関数式から始まります：

$$\left(\frac{x}{r_x}\right)^2 + \left(\frac{y}{r_y}\right)^2 = 1$$

$r_x = r_y = r$ のとき、楕円は半径 r の **円** に退化します。内部の点は ≤ 1、外部の点は > 1 を満たします。これがピクセルが形状に属するか否かを定める不等式です。3 つのアルゴリズムは、本質的にこの判定を格子上で適用（あるいは近似）する 3 通りの方法です。

中心を (c_x, c_y) に平行移動し、各セルの中心で評価します：

$$d_x = \frac{i + \frac{1}{2} - c_x}{r_x}, \qquad d_y = \frac{j + \frac{1}{2} - c_y}{r_y}, \qquad \text{内側} \iff d_x^2 + d_y^2 \leq 1$$

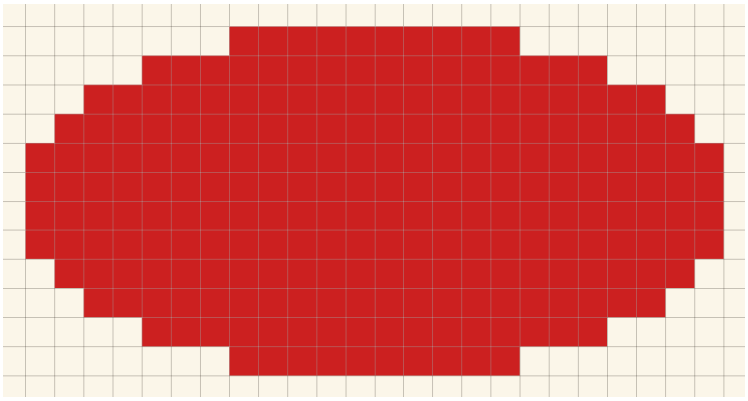

Figure 1: 離散格子上の楕円。$W = 24$、$H = 12$。

4. ユークリッド法（距離判定）

着想。 各行 j について、楕円内部にある列 i を解析的に求めます。y を与えて楕円方程式を x について解くと：

$$x \;=\; c_x \;\pm\; r_x \sqrt{1 - \left(\frac{y - c_y}{r_y}\right)^2}$$

垂直変位が $d_y = j + \frac{1}{2} - c_y$ の行 j における楕円の 水平半幅は：

$$dxM \;=\; r_x \sqrt{1 - \frac{d_y^2}{r_y^2}}$$

$d_y^2 / r_y^2 \geq 1$ なら、その行は楕円と交わらず飛ばされます。そうでなければ $c_x - dxM \leq i + \frac{1}{2} \leq c_x + dxM$ を満たすすべての列 i を塗ります。整数境界は次式で与えられます：

$$lo = \lceil c_x - dxM - \tfrac{1}{2} \rceil, \qquad hi = \lfloor c_x + dxM - \tfrac{1}{2} \rfloor$$

根拠。 $+\frac{1}{2}$ の項は、インデックス i のセルの中心が $i + \frac{1}{2}$ にあるという約束に由来します。$\lceil \cdot \rceil$ と $\lfloor \cdot \rfloor$ が連続区間を整数インデックスへ変換し、**中心** が楕円内部に属するセルだけが含まれることを保証します。この定式化は連続曲線に最も近い離散近似を与え、結果として最も滑らかなシルエットになります。ただし直径が小さい場合、他のアルゴリズムに比べてピクセルアートらしさが弱く感じられることがあります。

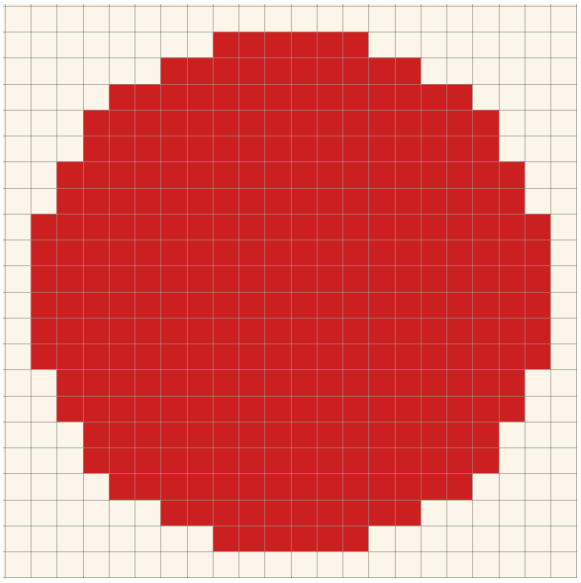

Figure 2: ユークリッド・アルゴリズム、$D = 20$。

5. Bresenham 法（整数中点法）

Bresenham のアルゴリズムは 1965 年に線分描画のために発表され、後に円へ拡張され、Pitteway と Kappel により任意の楕円へと一般化されました。その特徴的な性質は、浮動小数点演算と平方根計算を不要にする点にあります：次のピクセルの決定には **整数の加算と比較** のみを用い、2 つの候補ピクセルの中点が解析曲線のどちら側にあるかを判定する 判定関数を介して行われます。

5.1 円の場合

整数半径 R の円に対し、$(x{=}0,\, y{=}R)$ から初期判定パラメータで開始します：

$$d_0 \;=\; 1 - R$$

0 から 45° の 8 分の 1 円内の各ステップで：

$$\begin{cases} \text{もし} d < 0: & \text{次は} (x{+}1,\; y), \quad d \mathrel{+}= 2x + 3 \\ \text{もし} d \geq 0: & \text{次は} (x{+}1,\; y{-}1), \quad d \mathrel{+}= 2(x - y) + 5 \end{cases}$$

根拠。 関数 d は反復ごとに、2 つの候補ピクセルの **中点** における円の陰関数式の値を近似します：

$$F\!\left(x + 1,\; y - \tfrac{1}{2}\right) \;=\; (x+1)^2 + \left(y - \tfrac{1}{2}\right)^2 - R^2$$

d の符号は、中点が円の内側にある（この場合は水平方向のみに進む）か、外側にある（この場合は垂直座標も減らす）かを示します。円の 8 つの 8 分の 1 円は、計算された 8 分の 1 円に対し二面体群 D_4 の対称変換を施して得られ、コード上は `span(...)` の 8 回の呼び出しに対応します。得られる軌跡は、滑らかな曲線を離散近似する際に特有の階段状パターンを示します。

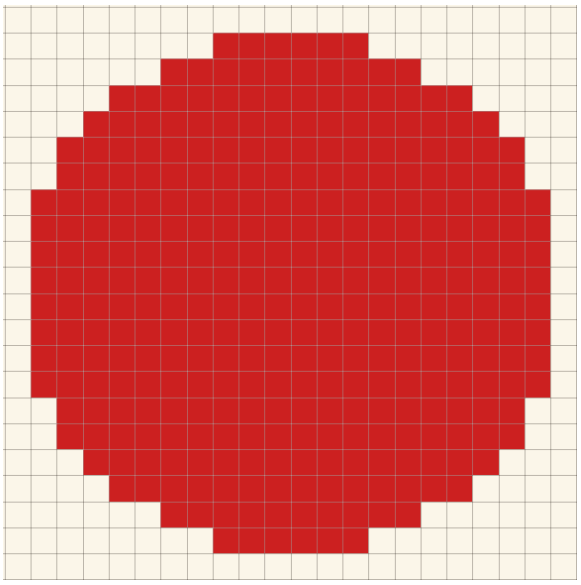

Figure 3: Bresenham アルゴリズム、$D = 20$。

5.2 楕円の場合（2 つの領域）

半径 a, b の楕円に対し、アルゴリズムは第 1 象限を、接線が 45° をなす点を境に **2 つの領域** に分割します：

$$\text{領域 I: } 2b^2x < 2a^2y \quad (|dy/dx| < 1), \qquad \text{領域 II: それ以外}$$

初期判定パラメータは：

$$p_1 = b^2 - a^2b + \tfrac{a^2}{4},$$
$$p_2 = b^2\left(x + \tfrac{1}{2}\right)^2 + a^2(y-1)^2 - a^2b^2.$$

各反復で p は事前計算した増分により更新されます（4 を掛けた後はすべて整数）。1 ピクセルあたりの演算量は最小限です：符号判定 *1* 回と加算 *2* 回。コードは、r_y が整数でないときに *bounding box* の外に余分な行を追加しないよう、r_x, r_y から a, b を求める際に `Math.floor` を用います。

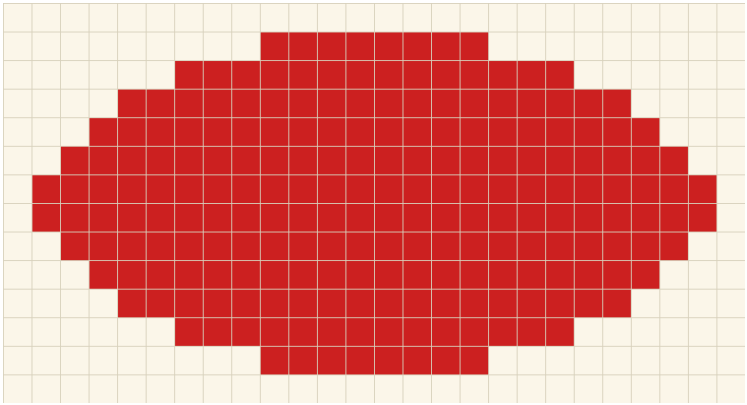

Figure 4: Bresenham アルゴリズムを楕円（$W = 24$、$H = 12$）に適用。

6. しきい値アルゴリズム（Threshold / 角の被覆）

このアルゴリズムは次を問います：セルの角のいずれかが楕円内にあるか？各セル (i, j) について、形状の中心に最も近い角点（中心のセル辺への射影）を求めます：

$$n_x = \text{clamp}(c_x,\, i,\, i{+}1), \quad n_y = \text{clamp}(c_y,\, j,\, j{+}1)$$

$$\text{内側} \iff \left(\frac{n_x - c_x}{r_x}\right)^2 + \left(\frac{n_y - c_y}{r_y}\right)^2 \le 0{,}94$$

値 0,94 は 1,0 よりわずかに小さい 経験的しきい値で、その役割は四方位（N、S、E、W）に現れる 1 ピクセルの 接線スパイクアーティファクトを除去することです。曲線がセルの 1 辺と接する箇所では、この低下処理なしでは形状の視覚的表現を改善しない余計なセルが含まれてしまいます。

3 つのアルゴリズムのうち、同じ直径 D で最大の面積を持つシルエットを生成するのが本アルゴリズムです。包含条件が最も緩く、セルの角の 1 つでも楕円不等式を満たせばセル全体が塗られるためです。

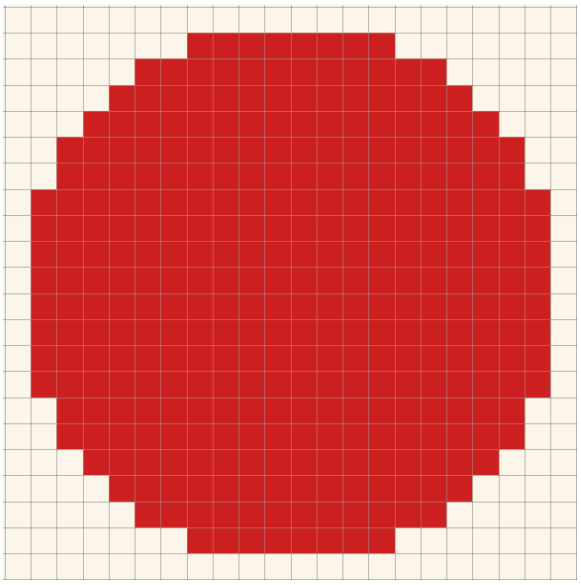

Figure 5: 閾値アルゴリズム、$D = 20$。

7. 描画モード：塗りつぶし、細、太

内部セルに対して $f[j][i] = 1$ となる行列を得たうえで、プロジェクトは **離散トポロジー** により 3 つの視覚スタイルを導出します：

7.1 塗りつぶし

f をそのまま返します。すべての内部ピクセル。

7.2 細（outline）

ピクセルは塗りつぶされており かつ形状の 外に直交近傍（N、S、E、W）を少なくとも 1 つ持つとき、細い輪郭に属します。論理記法では：

$$b(i,j) \;=\; f(i,j) \;\wedge\; \big(\neg f(i{-}1,j) \vee \neg f(i{+}1,j) \vee \neg f(i,j{-}1) \vee \neg f(i,j{+}1)\big)$$

幾何学的には、形状の **離散境界** であり、位相的境界 $\partial F = F \cap \overline{F^{\mathrm{c}}}$ の離散版に相当します。

7.3 太（thick）

太モードでは境界ピクセルに加え、**対角ブリッジ** を追加します。これは水平境界の隣接 かつ垂直境界の隣接を同時に持つ内部セルです。これにより細モードで残る 1 ピクセルの対角穴が埋まり、輪郭をシーン中で安定させる用途（45° の

隙間がない）に適します。

$$\begin{aligned} t(i,j) &= b(i,j) \ \vee \ \big(f(i,j) \ \wedge \ h_H \ \wedge \ h_V\big), \\ h_H &= b(i{-}1,j) \ \vee \ b(i{+}1,j), \\ h_V &= b(i,j{-}1) \ \vee \ b(i,j{+}1). \end{aligned}$$

8. 離散面積の計算

関数 `area2D` は f で印付けられたセルを単に **数える** だけです。これは 数え上げ測度であり、楕円の指示関数の Riemann 積分の近似となります：

$$\underbrace{\pi\, r_x\, r_y}_{\text{連続面積}} \qquad \longleftrightarrow \qquad \underbrace{\sum_{j=0}^{G_y-1} \sum_{i=0}^{G_x-1} f(i,j)}_{\text{離散面積}}$$

D が大きいとき 離散面積/連続面積 $\to 1$。D が小さいときはアルゴリズム間の差が顕著です：Threshold は過大評価する傾向、ユークリッド法は理想面積に近い、Bresenham はその中間に位置します。

9. 2D から 3D へ：楕円体の方程式

3D では基本形状は、(c_x, c_y, c_z) を中心とし半径 r_x, r_y, r_z を持つ **楕円体** です。その方程式は楕円の自然な一般化です：

$$\left(\frac{x}{r_x}\right)^2 + \left(\frac{y}{r_y}\right)^2 + \left(\frac{z}{r_z}\right)^2 \ = \ 1$$

$r_x = r_y = r_z$ のとき **球** に戻ります。各ボクセルの中心で評価すると：

$$a_\xi = \frac{\xi + \frac{1}{2} - c_\xi}{r_\xi} \ \ (\xi \in \{x,y,z\}), \qquad \text{内側} \iff a_x^2 + a_y^2 + a_z^2 \le 1$$

10. ボクセル化と体積計算

楕円体の連続体積は積分学の古典的結果で、円盤による積分で得られます：

$$V \ = \ \frac{4}{3}\,\pi\, r_x\, r_y\, r_z$$

離散版は箱 $D_x \times D_y \times D_z$ の各ボクセルを走査し、$a_x^2 + a_y^2 + a_z^2 \le 1$ のときカウンタを増やします。これが `voxelVolume` のアルゴリズムです。

シェル（shell）と内部。 *filled* モードでは、保持された集合の外に少なくとも 1 つの 6-近傍を持つボクセル（つまり外面または切断面から見えるボクセル）のみを表示します。*thin* モードでは、完全な楕円体の **表面** のみ（厚さ 1 ボクセル）を表示します。

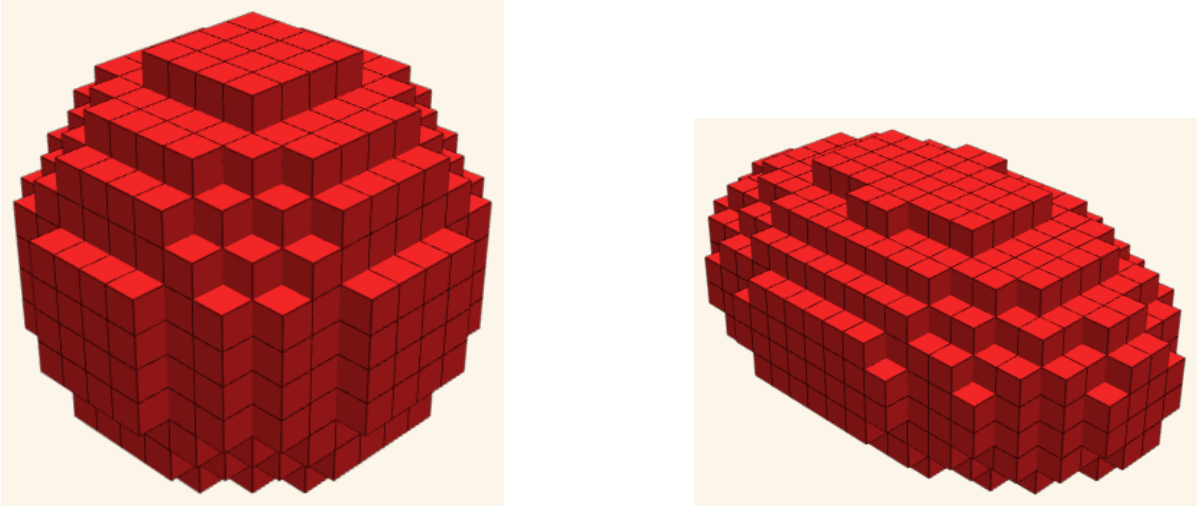

Figure 6: ボクセル化された球（$D = 12$）と楕円体（$W = 20$, $H = 10$, $D = 12$）。

11. 幾何学的切断：平面と 45° の対角切断

Pixel Round では **3 つの平面** で立体を切断できます。各切断はボクセルを除外する 線形不等式です：

$$\begin{aligned} &\text{X 軸:} && x \geq cut && \Rightarrow \text{ 除外} \\ &\text{Y 軸:} && y \geq cut && \Rightarrow \text{ 除外} \\ &\text{対角:} && x + y \geq cut && \Rightarrow \text{ 除外} \end{aligned}$$

X、Y の平面は座標軸に垂直で、法線ベクトルはそれぞれ $(1, 0, 0)$、$(0, 1, 0)$ です。対角平面の法線ベクトルは $(1, 1, 0)/\sqrt{2}$ で、XY 平面内で 45° に切ります：数学的には平面族 $x + y = k$ です。$D_x \times D_y$ の箱における $x + y$ の最大値は $D_x + D_y$ なので、対角切断スライダーの幅は $D_x + D_y$、X と Y はそれぞれ D_x、D_y です。切断は **比例的** です：コードは割合 $cutPct$ を保持し、しきい値を次式で再計算します

$$cut \;=\; cutPct \cdot \max_{\text{軸}}$$

これによりユーザが軸を切り替えたりサイズを変更しても、ある軸での 50% は引き続き 50% のままになります。

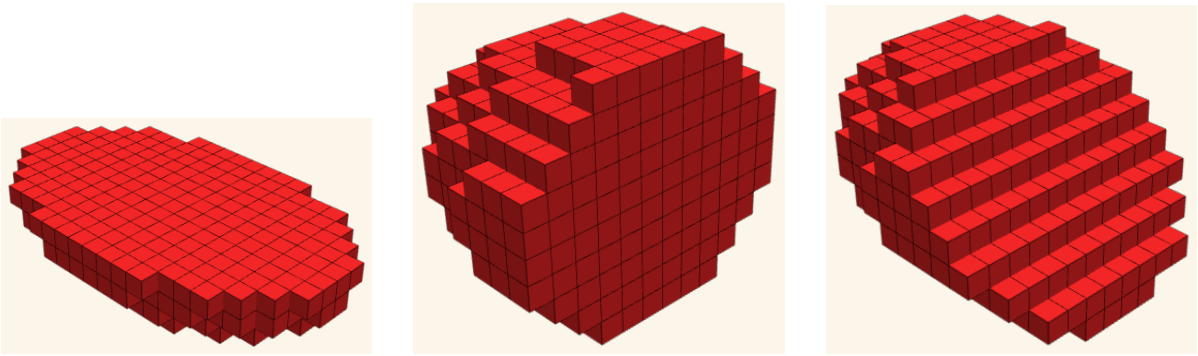

Figure 7: Y、X、対角線軸での 50% 平面切断（左から右）。

12. 3D 太モード：スライスによる立体化

3D *thick* モードでは、3 軸方向の **すべてのスライス** に 2D 太アルゴリズムを適用します：z ごとに XY スライス、y ごとに XZ スライス、x ごとに YZ スライス。続いて結果の 和集合を取ります。幾何学的動機は気密性です：シェル厚を倍にせずに、すべての対角を塞ぐボクセルの最小集合を取ります。形式的には、$S_z(z)$ を高さ z における XY スライス、T を 2D 太演算子として：

$$thick3D = \bigcup_z T(S_z(z)) \cup \bigcup_y T(S_y(y)) \cup \bigcup_x T(S_x(x))$$

結果はその後、切断で保持された集合との共通部分が取られます。背後にある理論は **デジタルトポロジー** です：太演算子はシェルの *26*-連結性（対角を含む近傍）を保証し、対角に隣接するボクセルが面で接触しない Minecraft における構築に有用です。

13. 3D カメラ：球座標

3D カメラは距離 r で対象を周回し、2 つの角度で記述されます　θ（方位角、水平面内）と φ（極角、垂直軸からの角度）。これは球座標の **物理学流の慣習** であり、直交座標（*three.js* が扱う形式）への変換は次のとおりです：

$$\begin{aligned} x &= r\,\sin(\varphi)\,\cos(\theta), \\ y &= r\,\cos(\varphi), \\ z &= r\,\sin(\varphi)\,\sin(\theta). \end{aligned}$$

初期位置：$\theta = \pi/4$（45°、等角投影風）、$\varphi = \pi/3$（北極から 60°、赤道のやや上）。マウスドラッグで θ、φ を直接増減；ホイール／ピンチで r を増減。このパラメータ化の簡潔さにより、四元数や明示的な行列を用いずに自由な回転が可能になります。

14. 自動ズーム：包囲球と FOV

ユーザが図形のサイズを変えたりカメラをリセットすると、プロジェクトは任意の角度で対象がビューに収まるよう、カメラの **理想距離** を再計算します。発想は対象を 半径 R の球で包むことです（AABB すなわち axis-aligned bounding box の対角線の半分）：

$$R \;=\; \sqrt{(D_x/2)^2 + (D_y/2)^2 + (D_z/2)^2}$$

球は **回転に対し不変な投影** を持つため、ある向きでビューに収まれば任意の向きでも収まります。カメラの垂直 FOV を $fov_V = 35°$（ラジアン）とすると、半径 R の球を収めるのに必要な距離は：

$$dist_V \;=\; \frac{R}{\tan(fov_V/2)}$$

水平 FOV はキャンバスのアスペクト比（$aspect$ = 幅/高さ）から：

$$fov_H \;=\; 2\,\arctan\!\big(\,\tan(fov_V/2)\cdot aspect\,\big)$$

最終距離は $dist_V$ と $dist_H$ の最大値に 1,20 を乗じた値（視覚マージン 20%）です。図形が切断されている場合、コードは D_x または D_y を残存サイズに縮めるので、75% 切断した球が画面の隅に小さく現れることはありません。

15. シェーディングと RGB 色演算

3 つの 3D スタイル（*Classic*、*Smooth*、*Blocks*）は、各ボクセルの 3 つの可視面（上面、明側、暗側）に適用される **乗算因子** により区別されます。関数 `shadeHex(hex, factor)` は 16 進値を (R, G, B) にパースし、各チャネルを掛けて $[0, 255]$ にクランプします：

$$R' = \mathrm{clamp}(R \cdot f), \quad G' = \mathrm{clamp}(G \cdot f), \quad B' = \mathrm{clamp}(B \cdot f)$$

これは理想 Lambertian 照明関数の 線形近似です：

$$I \;=\; \max(0,\; \mathbf{n}\cdot\mathbf{l})\cdot \quad ,$$

3 つの因子が、各面の法線と固定光源方向のなす角の余弦に対応するときに上式が回復されます。*Smooth* では因子が近接し（面同士が似た色）、*Classic* では差が大きく（強いコントラスト）。*Blocks* はさらにボクセルへの幾何的な凹みを導入します　各面を一定量内側に並進させます。

16. 音声：周波数と音階

インターフェースの効果音は WebAudio によりリアルタイム合成されます。各効果音は **純粋な正弦波** です：

$$s(t) \;=\; A \cdot \sin\!\left(2\pi\, f\, t\right)$$

プロジェクトが選んだ f（Hz）の値は任意ではありません　*12* 平均律の音名に近似します。半音ごとに周波数が $2^{1/12} \approx 1{,}0595$ 倍されます：

効果音	周波数 (Hz)	近い音名	用途
click	620	≈ D#5 / E♭5	ボタン
hover	1100	≈ C#6	デスクトップのホバー
tick	1300	≈ E6	スライダーのステップ
ok	520 → 720 → 920	上行アルペジオ	リセット／ダウンロード
pop	680 + 980	完全 5 度の音程	トグル
error	200	≈ G#3	無効な操作

完全 5 度（比 3 : 2）：*pop* 音は 680 Hz と 980 Hz を混ぜます；980/680 ≈ 1,44（純正完全 5 度なら 1,5）で、協和に聞こえます。***ok* 音のアルペジオ**の連続比は 3 度に近いです：520 → 720（比 1,38、平均律短 3 度 1,2 を少し広げた値）と 720 → 920。この構成が結びの感覚を与える上行カデンツを生みます。エンベロープの長さは 12 〜160 ms、最大ゲインは 0,035（およそ −29 dBFS）で、長時間使用での聴覚疲労を避けるため意図的に低く抑えられています。

17. 結論

本書は Pixel Round で用いる数学的基礎を体系的に記述してきました。約 1000 行の JavaScript コードにおいて、本プロジェクトは複数の分野からの寄与を統合しています：

- **解析幾何**: 楕円および楕円体の陰関数式を主たる帰属判定基準とする。
- **古典的ラスタライズアルゴリズム**: 中点形式の Bresenham と楕円への Pitteway/Kappel 拡張。
- **デジタルトポロジー**: 離散境界作用素、4-、6-、26- 連結性の概念、スライス合成。
- **積分学**: 楕円体体積の三重積分表示およびその離散対応（ボクセル上の数え上げ測度）。

- **線形代数**: 線形不等式により定義される切断平面と 3 次元透視投影。
- **三角法**: 球座標によるカメラの記述と、FOV に応じた距離の自動調整。
- **信号合成**: 純粋な正弦波と 12 平均律の近似。

本書から導かれる中心的観察は次の通りです：離散格子上では、連続曲線に対する **唯一正しい表現は存在しません**。本書で示した各アルゴリズムは、セル包含基準の数学的定義それぞれに対応し、独自の形式的性質を持つシルエットを生み出します　ユークリッド法での滑らかさ、Bresenham の階段パターン、角の被覆基準の広い覆い。アルゴリズムの選択は、意図する視覚的表現と結果形状に求められる定量的特性を勘案して下す技術的判断です。