Pixel Round

项目的数学原理

本文档为像素艺术和体素艺术中圆形、椭圆、球体和椭球体生成器所采用的数学基础提供技术性与教学性的说明。每个概念都给出形式化定义、几何论证以及与项目源码的直接对应关系。

涉及领域： 解析几何 · 离散光栅化 · 数字拓扑 · 积分学 · 线性代数 · 三角学 · 音频合成。

目录

1. 概述：问题的数学本质

Pixel Round 是用于 **像素艺术**和 **体素艺术**的圆形形状生成器：二维的圆和椭圆，三维的球体和椭球体。其核心问题属于 离散光栅化的范畴：给定一条在实数集 $\mathbb{R}$ 上以解析方式定义的曲线或曲面，确定规则网格的哪些单元（二维像素或三维体素）应当被标记以表示该形状。

该问题不存在唯一解。不同的包含判据会产生不同的离散轮廓，每个判据都有其自身的数学依据。因此项目实现了三种不同的二维算法、三种渲染模式以及三种三维着色风格，并在后续章节中分别说明。

程序完全在浏览器中运行，无需服务端组件，这对数学表述提出了额外的 计算效率要求。涉及的理论领域包括：

- 圆锥曲线与二次曲面的解析几何;
- 基于整数运算的增量算法（Bresenham）;
- 数字拓扑（4-、6- 和 26- 邻接关系）;
- 积分学与计数测度;
- 线性代数（切割平面、透视投影）;
- 三角学与球坐标;
- 信号合成与基础声学.

2. 坐标系与离散像素网格

画布是 $G_x \times G_y$ 个单元构成的网格。每个单元 (i, j) 占据正方形区域 $[i, i+1] \times [j, j+1]$。在连续视角下，该单元的 **中心** 位于 $(i + \frac{1}{2},\ j + \frac{1}{2})$。这一约定至关重要：以单元 角点 (i, j) 或 中心 $(i + \frac{1}{2}, j + \frac{1}{2})$ 来比较圆周方程，会改变被判定为属于该形状的单元集合。

$$\text{单元 } (i,j) \text{ 的中心} \ = \ (i + \tfrac{1}{2},\ \ j + \tfrac{1}{2})$$

对于直径为 D 的形状，代码将网格扩展到每轴 $D + 2$ 个单元（两侧各预留一个单元的余量），保证最外层像素（东南西北）永远不会落在矩阵边界上。形状的 **中心** 位于 $(G_x/2,\ G_y/2)$，半轴为 $r_x = D/2$ 和 $r_y = D/2$（椭圆则为 $W/2$ 和 $H/2$）。

3. 基本方程：圆与椭圆

项目的全部理论均始于以原点为中心、半轴为 r_x 和 r_y 的椭圆的隐式方程：

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

当 $r_x = r_y = r$ 时，椭圆退化为半径为 r 的 **圆**。内部的点满足 ≤ 1；外部的点满足 > 1。这就是判定像素是否属于该形状的不等式。三种算法本质上是在网格上应用（或近似）该判定的三种不同方式。

将中心平移到 (c_x, c_y) 并在每个单元的中心处求值：

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

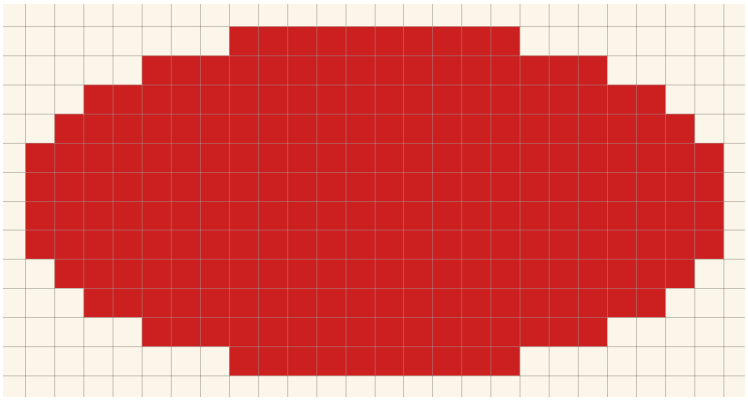

Figure 1: 离散网格上的椭圆，$W = 24$、$H = 12$。

4. 欧氏算法（距离判定）

思路。对每一行 j，解析地求出位于椭圆内部的列 i。给定 y，由椭圆方程解出 x：

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

对于纵向偏移为 $d_y = j + \frac{1}{2} - c_y$ 的行 j，椭圆在该高度的 水平半宽为：

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

若 $d_y^2/r_y^2 \geq 1$，该行不与椭圆相交，跳过。否则填充所有满足 $c_x - dxM \leq i + \frac{1}{2} \leq c_x + dxM$ 的列 i。整数边界由下式给出：

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

说明。 $+\frac{1}{2}$ 项来自于约定：索引为 i 的单元的中心位于 $i + \frac{1}{2}$。函数 $\lceil \cdot \rceil$ 与 $\lfloor \cdot \rfloor$ 将连续区间转换为整数索引，确保只有 **中心** 落在椭圆内部的单元被纳入。该方案给出最接近连续曲线的离散近似，因此产生视觉上最平滑的轮廓。但对于较小的直径，结果在像素艺术风格上可能不如其他算法明显。

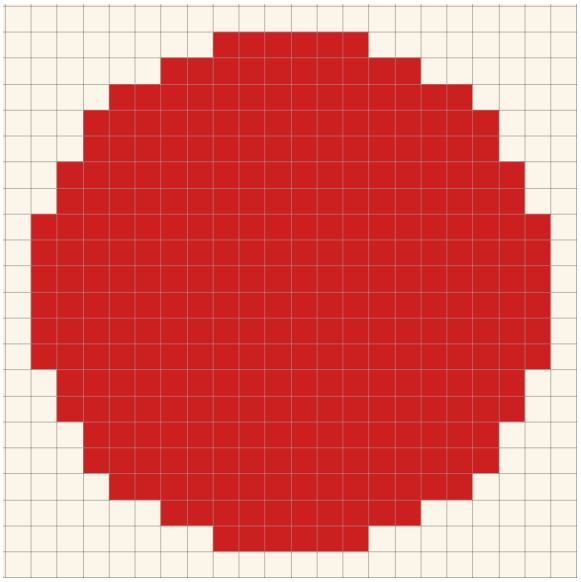

Figure 2: 欧几里得算法，$D = 20$。

5. Bresenham 算法（整数中点法）

Bresenham 算法 1965 年首先用于绘制直线段，随后被推广到圆，并由 Pitteway 与 Kappel 进一步推广到任意椭圆。其显著特点是不使用浮点运算和开方：下一个像素的判定仅依赖 **整数加法与比较**，通过一个 判定函数判断两候选像素之间的中点位于解析曲线的哪一侧。

5.1 圆的情形

对于整数半径 R 的圆，从 $(x{=}0,\ y{=}R)$ 出发，初始判定参数为：

$$d_0 \ = \ 1 - R$$

在 0 到 45° 的八分圆中，每一步：

$$\begin{cases} \text{若} d < 0: & \text{下一个为} (x{+}1,\ y), \quad d \mathrel{+}= 2x + 3 \\ \text{若} d \geq 0: & \text{下一个为} (x{+}1,\ y{-}1), \quad d \mathrel{+}= 2(x - y) + 5 \end{cases}$$

说明。 函数 d 在每次迭代时近似圆的隐式方程在两候选像素之间 **中点** 处的值：

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

d 的符号指示该中点位于圆的内部（此时仅水平前进）还是外部（此时还需向下递减纵坐标）。圆的八个八分圆由对计算出的八分圆作二面体群 D_4 的对称变换得到，对应代码中 `span(...)` 的八次调用。所产生的轨迹呈现出光滑曲线离散近似所特有的阶梯状图案。

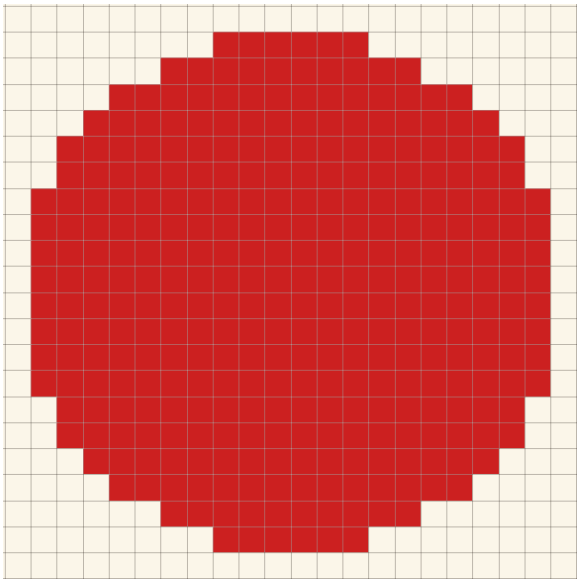

Figure 3: Bresenham 算法，$D = 20$。

5.2 椭圆的情形（两个区域）

对于半轴为 a, b 的椭圆，算法将第一象限以切线斜率为 45° 的点为界划分为 **两个区域**：

区域 I: $2b^2x < 2a^2y \quad (|dy/dx| < 1),$ 区域 II: 否则

初始判定参数为：

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

每次迭代时，p 通过预先计算的增量更新（乘以 4 后全部为整数）。结果是每像素最少的算术运算量：一次符号判断和两次加法。代码在由 r_x 和 r_y 计算 a, b 时使用 `Math.floor`，以避免当 r_y 非整数时在 *bounding box* 之外多填一行。

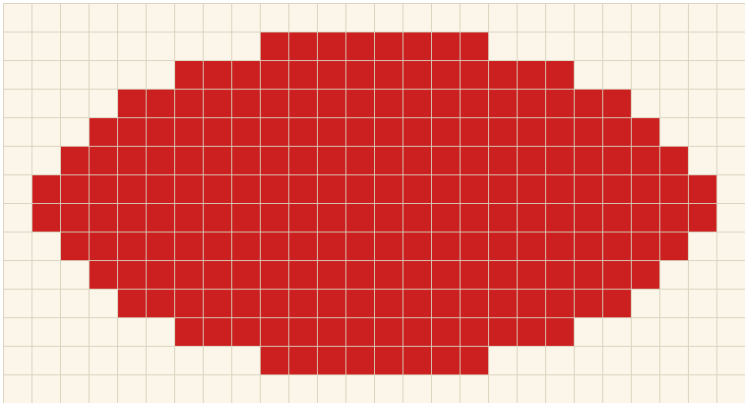

Figure 4: Bresenham 算法用于椭圆 $W = 24$、$H = 12$。

6. 阈值算法（Threshold / 角点覆盖）

该算法的判定标准为：单元的四角中是否至少有一个位于椭圆内？对每个单元 (i, j)，代码寻找距形状中心最近的角点（中心在单元边上的投影）：

$$n_x = \mathrm{clamp}(c_x,\, i,\, i{+}1), \quad n_y = \mathrm{clamp}(c_y,\, j,\, j{+}1)$$
$$\text{内部} \iff \left(\frac{n_x - c_x}{r_x}\right)^2 + \left(\frac{n_y - c_y}{r_y}\right)^2 \le 0{,}94$$

取值 0,94 为略小于 1,0 的 经验阈值，其作用是消除在四个基本方向（N、S、E、W）出现的 1 像素 切线尖刺伪影——此时曲线恰好与单元的一整条边相切。若不作此降阈处理，算法会额外纳入一个对形状视觉表示无实质改善的单元。

在三种算法中，对于相同的直径 D，本算法所产生的轮廓面积最大，因为其包含条件最为宽松：单元只要有一个角点满足椭圆不等式，即整体被填充。

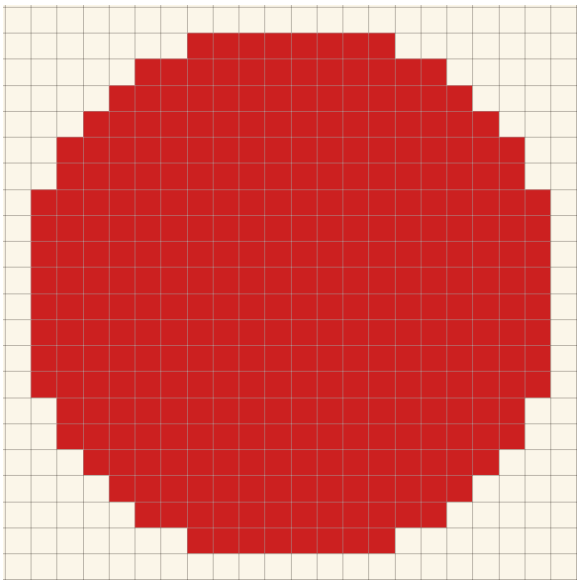

Figure 5: 阈值算法，$D = 20$。

7. 渲染模式：填充、细、粗

在拥有标记内部单元的矩阵 $f[j][i] = 1$ 后，项目通过 **离散拓扑** 推导出三种视觉风格：

7.1 填充

返回未修改的 f。全部内部像素。

7.2 细（outline）

若一个像素已被填充 且至少有一个正交邻居（N、S、E、W）位于形状 外部，则它属于细轮廓。用逻辑符号表示：

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

几何意义：这是形状的 **离散边界**，对应于拓扑边界 $\partial F = F \cap \overline{F^{c}}$ 的离散类比。

7.3 粗（thick）

粗模式在边界像素之上再加入 **对角桥**：同时拥有一个水平边界邻居 和一个垂直边界邻居的内部单元。这填补了细模式留下的 1 像素对角空隙，适用于轮廓须在场

景中保持稳定（无 45° 漏洞）的情形。

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

8. 离散面积的计算

函数 `area2D` 仅对 f 中标记的单元进行 **计数**。这就是 计数测度，是对椭圆指示函数的 Riemann 积分的离散近似：

$$\underbrace{\pi\, r_x\, r_y}_{\text{连续面积}} \qquad \longleftrightarrow \qquad \underbrace{\sum_{j=0}^{G_y-1} \sum_{i=0}^{G_x-1} f(i,j)}_{\text{离散面积}}$$

当 D 较大时，离散面积/连续面积 $\to 1$。当 D 较小时，算法之间的差异明显可见：Threshold 倾向于高估；欧氏算法接近理想面积；Bresenham 介于两者之间。

9. 由二维到三维：椭球体方程

在三维中，基本形状是以 (c_x, c_y, c_z) 为中心、半轴为 r_x, r_y, r_z 的 **椭球体**。其方程是椭圆方程的自然推广：

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

当 $r_x = r_y = r_z$ 时，化为 **球**。在每个体素的中心求值：

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

10. 体素化与体积计算

椭球体的连续体积是积分学的经典结果，可通过对圆盘积分得到：

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

离散版本遍历箱形区域 $D_x \times D_y \times D_z$ 中的每个体素，当 $a_x^2 + a_y^2 + a_z^2 \le 1$ 时计数器递增。即 `voxelVolume` 算法。

壳（shell）与内部。在 *filled* 模式下，项目仅显示至少有一个 6-邻居位于保留集合之外的体素（即从外表面或切割面可见的体素）。在 *thin* 模式下，仅显示完整椭球体的 **表面**（厚 1 体素）。

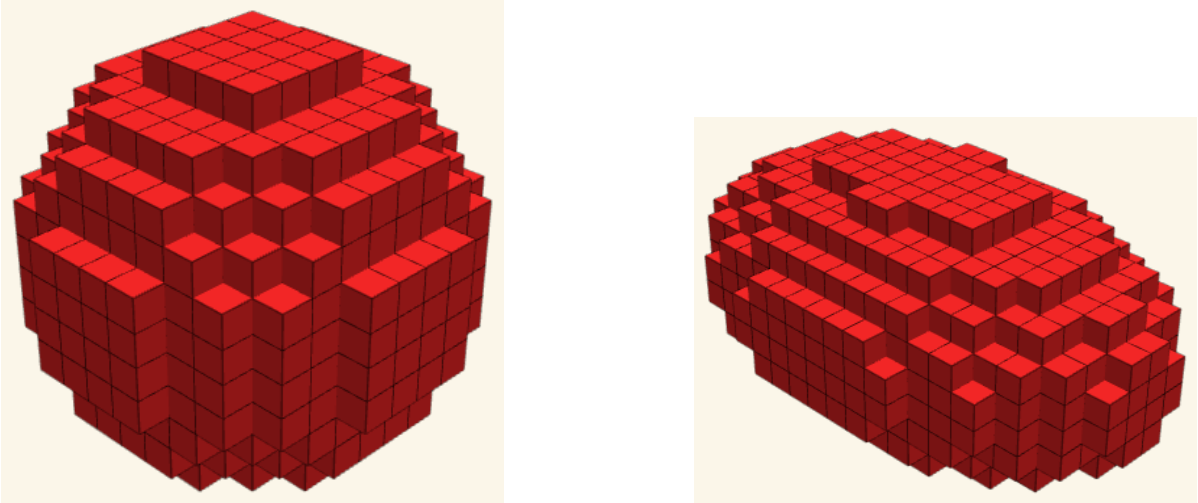

Figure 6: 球（$D = 12$）与椭球（$W = 20$, $H = 10$, $D = 12$）的体素化。

11. 几何切割：平面与 45° 对角切割

Pixel Round 允许使用 **三个平面** 对实体进行切割。每个切割都是一个抛弃体素的线性不等式：

$$\begin{array}{lll} \text{X 轴:} & x \geq cut & \Rightarrow \text{被舍弃} \\ \text{Y 轴:} & y \geq cut & \Rightarrow \text{被舍弃} \\ \text{对角:} & x + y \geq cut & \Rightarrow \text{被舍弃} \end{array}$$

X、Y 平面分别垂直于坐标轴，法向量为 $(1, 0, 0)$ 和 $(0, 1, 0)$。对角平面的法向量为 $(1, 1, 0)/\sqrt{2}$，在 XY 平面上以 45° 切割：在数学上即平面族 $x + y = k$。由于 $x + y$ 在尺寸为 $D_x \times D_y$ 的箱体中的最大值为 $D_x + D_y$，故对角切割的滑块范围为 $D_x + D_y$，而 X 与 Y 的范围分别为 D_x 与 D_y。切割按 **比例** 进行：代码保存百分比 $cutPct$ 并按以下方式重新计算阈值

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

因此用户切换坐标轴或调整尺寸时，某轴上的 50% 仍保持 50%。

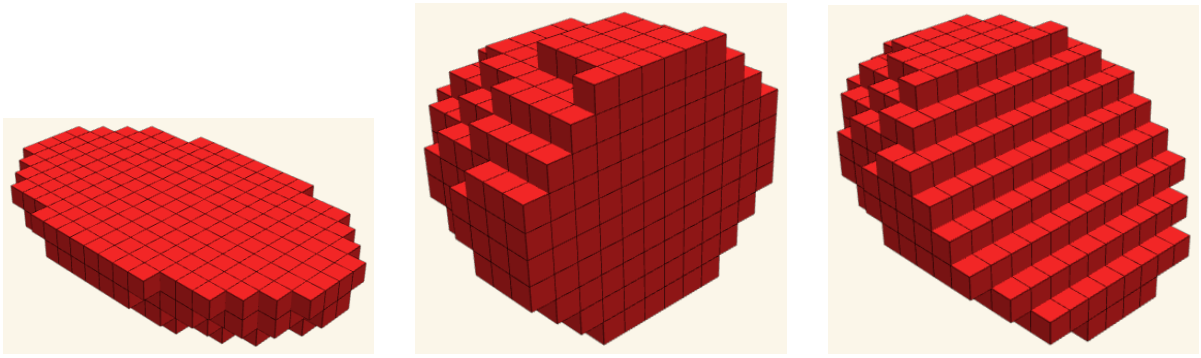

Figure 7: Y、X、对角线三个方向上的 50% 平面切割（从左到右）。

12. 三维粗模式：按切片实现的立体化

在三维 *thick* 模式下，项目对沿三轴的 **所有切片** 应用二维粗算法：每个 z 对应一张 XY 切片，每个 y 对应一张 XZ 切片，每个 x 对应一张 YZ 切片，并对结果取 并集。其几何动机是 密封性：用最少的体素堵住所有对角空隙而不使壳厚度加倍。形式化地，记 $S_z(z)$ 为高度 z 处的 XY 切片，T 为二维粗运算符：

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

随后将结果与被切割保留的集合取交。其理论依据是 **数字拓扑**：粗运算符保证壳的 *26*-连通性（含对角邻居），便于在 Minecraft 中搭建（其中对角相邻的体素并不通过面接触）。

13. 三维相机：球坐标

三维相机以距离 r 绕物体环行，并由两个角度参数化——θ（方位角，水平面内）与 φ（极角，从纵轴起算）。这是球坐标的 **物理学约定**；转换为笛卡尔坐标（*three.js* 所采用的形式）为：

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

初始位置：$\theta = \pi/4$（45°，等轴测视角），$\varphi = \pi/3$（60°，自北极稍偏赤道）。鼠标拖动直接增量 θ 和 φ；滚轮 / 捏合增量 r。参数化的简单性使得无需四元数或显式矩阵即可实现自由旋转。

14. 自动缩放：包围球与视场角

当用户改变图形尺寸或重置相机时，项目重新计算相机的 **理想距离**，使物体在任意视角下都能完整呈现。其思路是用 半径为 R 的球将物体包络（AABB 即 axis-aligned bounding box 对角线的一半）：

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

球的 **投影在旋转下不变**，故只要在某一朝向能放下，则任意朝向均可。设相机的竖直 FOV（弧度制）为 $fov_V = 35°$，要将半径为 R 的球框入所需的距离为：

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

水平 FOV 由画布宽高比（$aspect =$ 宽/高）按下式得到：

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

最终距离取 $dist_V$ 与 $dist_H$ 中的较大者，再乘以 1,20（视觉留白 20%）。当图形被切割时，代码将 D_x 或 D_y 缩到剩余尺寸，避免切去 75% 的球在画面角落显得过小。

15. 着色（shading）与 RGB 颜色运算

三种三维风格（*Classic*、*Smooth*、*Blocks*）由作用于每个体素三个可见面的 **乘性因子**（顶面、受光侧、背光侧）加以区分。函数 `shadeHex(hex, factor)` 将十六进制解析为 (R, G, B) 并对每个通道相乘，结果钳制于 $[0, 255]$：

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

上述变换是理想 Lambertian 照明函数的 线性近似，

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

当三个因子分别等于每个面的法向量与固定光源方向的夹角余弦时即恢复 Lambertian 模型。在 *Smooth* 中三个因子相近（各面相似）；在 *Classic* 中差异较大（对比强烈）。*Blocks* 另在每个体素中引入几何凹陷——各面整体向内平移一恒定距离。

16. 音频：频率与音阶

界面音效通过 WebAudio 实时合成。每个音效都是一段 **纯正弦波**：

$$s(t) = A\cdot\sin(2\pi\, f\, t)$$

项目所选的频率 f（Hz）并非任意——它们近似 十二平均律的音名，其中每个半音将频率乘以 $2^{1/12} \approx 1{,}0595$：

音效	频率 (Hz)	最近音名	用途
click	620	≈ D#5 / E♭5	按钮
hover	1100	≈ C#6	桌面端悬停
tick	1300	≈ E6	滑块步进
ok	520 → 720 → 920	上行琶音	重置 / 下载
pop	680 + 980	纯五度音程	切换
error	200	≈ G#3	无效操作

纯五度（比值 $3:2$）：*pop* 音效混合 680 Hz 与 980 Hz；$980/680 \approx 1{,}44$（纯五度应为 1,5），足以听起来协和。***ok* 音效的琶音**相邻频率比近似小三度：520 → 720（比值 1,38，接近平均律小三度 1,2 稍扩）以及 720 → 920，从而产生具有结束感的上行级进。包络时长 12 至 160 ms，峰值增益 0,035（约 −29 dBFS），为避免长时使用造成听觉疲劳而刻意压低音量。

17. 结语

本文档以系统化的方式描述了 Pixel Round 所使用的数学基础。在约一千行 JavaScript 代码中，项目整合了来自多个领域的贡献：

- **解析几何**：椭圆与椭球体的隐式方程作为主要的包含判据。
- **经典光栅化算法**：中点形式的 Bresenham，及 Pitteway/Kappel 对椭圆的推广。
- **数字拓扑**：离散边界算子，4-、6- 和 26- 连通性概念，以及按切片的合成。
- **积分学**：椭球体体积的三重积分表示及其离散对应（体素上的计数测度）。
- **线性代数**：由线性不等式定义的切割平面与三维透视投影。
- **三角学**：球坐标下的相机参数化以及视场角驱动的距离自动调整。
- **信号合成**：纯正弦波与对十二平均律的近似。

本研究允许得出的核心观察是：在离散网格上，**并不存在唯一正确的** 连续曲线表示方式。本文中所介绍的每一种算法都对应于单元包含判据的一种不同数学定义，并由此产生一种具有自身形式属性的轮廓——欧氏算法的平滑感、Bresenham 的阶梯感、角点覆盖算法的更大覆盖面。算法的选择因此是一个技术决策，须结合预期的视觉表现目标与所期望的形状度量特征加以判断。