Pixel Round

프로젝트의 수학

픽셀 아트와 복셀 아트용 원, 타원, 구, 타원체 생성기에서 사용되는 수학적 기초에 대한 기술적이고 교육적인 문서입니다. 각 개념마다 형식적 정의, 기하학적 정당화, 그리고 프로젝트의 소스 코드와의 직접적 대응 관계를 제시합니다.

다루는 영역: 해석기하 · 이산 래스터화 · 디지털 토폴로지 · 적분학 · 선형대수 · 삼각법 · 오디오 합성.

목차

1. 개요: 문제의 수학적 성격

Pixel Round 는 **픽셀 아트**와 **복셀 아트** 를 위한 둥근 도형 생성기로, 2D 에서는 원과 타원, 3D 에서는 구와 타원체를 다룹니다. 중심 문제는 이산 래스터화 의 범주에 속합니다: 실수 $\mathbb{R}$ 위에서 해석적으로 정의된 곡선 또는 곡면이 주어졌을 때, 그것을 표현하기 위해 정규 격자 (2D 의 픽셀, 3D 의 복셀) 의 어떤 셀을 표시해야 하는지를 결정하는 문제입니다.

이 문제는 유일한 해를 가지지 않습니다. 서로 다른 포함 기준은 서로 다른 이산 실루엣을 만들어내며, 각 기준은 자체의 수학적 정당성을 가집니다. 이로 인해 본 프로젝트는 세 가지 서로 다른 2D 알고리즘, 세 가지 렌더링 모드, 세 가지 3D 셰이딩 스타일을 구현하고 있으며, 이는 이하의 절들에서 설명됩니다.

실행은 서버 컴포넌트 없이 전적으로 브라우저 안에서만 이루어지며, 이로 인해 수학적 정식화에 계산 효율성 이라는 추가 요구가 부과됩니다. 동원된 이론적 영역은 다음과 같습니다:

- 원뿔곡선과 이차곡면의 해석기하;
- 정수 산술을 사용하는 점진적 알고리즘 (Bresenham);
- 디지털 토폴로지 (4-, 6-, 26- 연결 이웃);
- 적분학과 셈 측도;
- 선형대수 (절단 평면, 원근 투영);
- 삼각법과 구면 좌표;
- 신호 합성과 기초 음향학.

2. 좌표계와 이산 픽셀 격자

캔버스는 $G_x \times G_y$ 개의 셀로 이루어진 격자입니다. 각 셀 (i, j) 는 정사각형 $[i, i+1] \times [j, j+1]$ 을 차지합니다. 연속적 관점에서 그 셀의 **중심** 은 $(i + \frac{1}{2},\ j + \frac{1}{2})$ 에 위치합니다. 이 약속은 결정적입니다: 원주방정식을 셀의 모서리 (i, j) 와 비교하는지 중심 $(i + \frac{1}{2}, j + \frac{1}{2})$ 과 비교하는지에 따라 도형에 속한다고 간주되는 셀의 집합이 달라집니다.

$$\text{셀 } (i, j) \text{ 의 중심} \ = \ (i + \tfrac{1}{2},\ \ j + \tfrac{1}{2})$$

지름 D 의 도형에 대해, 코드는 격자를 각 축당 $D + 2$ 셀로 확장합니다 (각 변에 셀 하나씩의 여백). 이로써 최외곽 픽셀 (N, S, E, W) 이 행렬의 경계에 놓이는 일

이 없도록 보장합니다. 도형의 **중심** 은 $(G_x/2,\ G_y/2)$ 에 있고, 반축 은 $r_x = D/2$ 와 $r_y = D/2$ (타원의 경우 $W/2$ 와 $H/2$) 입니다.

3. 기본 방정식: 원과 타원

프로젝트의 모든 이론은 원점을 중심으로 반축이 r_x, r_y 인 타원의 음함수 방정식에서 시작합니다:

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

$r_x = r_y = r$ 일 때, 타원은 반지름 r 의 **원** 으로 퇴화됩니다. 내부 의 점들은 ≤ 1 을 만족하고; 외부 의 점들은 > 1 을 만족합니다. 이것이 픽셀이 도형에 속하는지 여부를 정의하는 부등식입니다. 세 알고리즘은 본질적으로 이 판정을 격자 위에서 적용 (또는 근사) 하는 세 가지 다른 방법입니다.

중심을 (c_x, c_y) 로 평행이동하고 각 셀의 중심에서 평가하면:

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

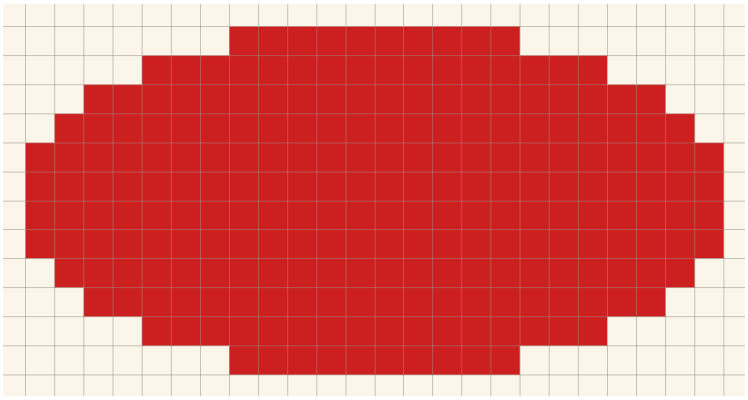

Figure 1: 이산 격자 위의 타원, $W = 24$ 및 $H = 12$.

4. 유클리드 알고리즘 (거리 판정)

아이디어. 각 행 j 에 대해 타원 내부에 있는 열 i 를 해석적으로 구합니다. y 가 주어졌을 때 타원 방정식을 x 에 대해 풀면:

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

수직 변위가 $d_y = j + \frac{1}{2} - c_y$ 인 행 j 에서 타원의 수평 반폭 은:

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

$d_y^2 / r_y^2 \geq 1$ 이면 그 행은 타원과 교차하지 않으므로 건너뜁니다. 그렇지 않으면 $c_x - dxM \leq i + \frac{1}{2} \leq c_x + dxM$ 을 만족하는 모든 열 i 를 채웁니다. 정수 경계는 다음에서 나옵니다:

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

설명. $+\frac{1}{2}$ 항은 인덱스 i 의 셀의 중심이 $i + \frac{1}{2}$ 에 있다는 약속에서 비롯됩니다. 함수 $\lceil \cdot \rceil$ 과 $\lfloor \cdot \rfloor$ 는 연속 구간을 정수 인덱스로 변환하여, **중심** 이 타원의 내부에 속하는 셀만이 포함되도록 보장합니다. 이 정식화는 연속 곡선에 가장 가까운 이산 근사를 산출하며, 따라서 시각적으로 가장 부드러운 실루엣을 만듭니다. 그러나 작은 지름에서는 결과가 다른 알고리즘과 비교하여 픽셀 아트적 특성이 덜 두드러질 수 있습니다.

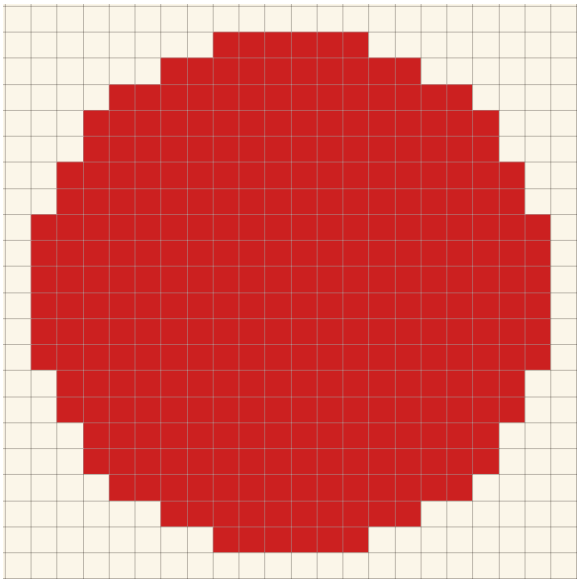

Figure 2: 유클리드 알고리즘, $D = 20$.

5. Bresenham 알고리즘 (정수 중점)

1965 년 선분 그리기를 위해 발표된 Bresenham 알고리즘은 이후 원으로 확장되었으며, Pitteway 와 Kappel 에 의해 임의의 타원으로 일반화되었습니다. 그 두드러진 특징은 부동소수점 연산과 제곱근 계산을 사용하지 않는다는 점입니다: 다음 픽셀의 결정은 오직 **정수의 덧셈과 비교** 만을 사용하며, 두 후보 픽셀 사이의 중점이 해석 곡선의 어느 쪽에 있는지를 평가하는 판정 함수 를 통해 이루어집니다.

5.1 원의 경우

정수 반지름 R 인 원에 대해, $(x{=}0,\, y{=}R)$ 에서 초기 판정 매개변수와 함께 시작합니다:

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

0 부터 45° 사이의 8 분원에서 각 단계마다:

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

설명. 함수 d 는 반복마다 두 후보 픽셀 사이의 **중점** 에서 평가된 원의 음함수 방정식의 값을 근사합니다:

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

d 의 부호는 중점이 원의 내부에 있는지 (이 경우 수평으로만 전진) 외부에 있는지 (이 경우 수직 좌표도 감소) 를 알려줍니다. 원의 8 개 8 분원은 계산된 8 분원에 이면체 군 D_4 의 대칭을 적용하여 얻어지며, 이는 코드 상으로 `span(...)` 의 8 회 호출에 해당합니다. 그 결과 만들어지는 자취는 매끄러운 곡선의 이산 근사에 특유한 계단형 패턴을 나타냅니다.

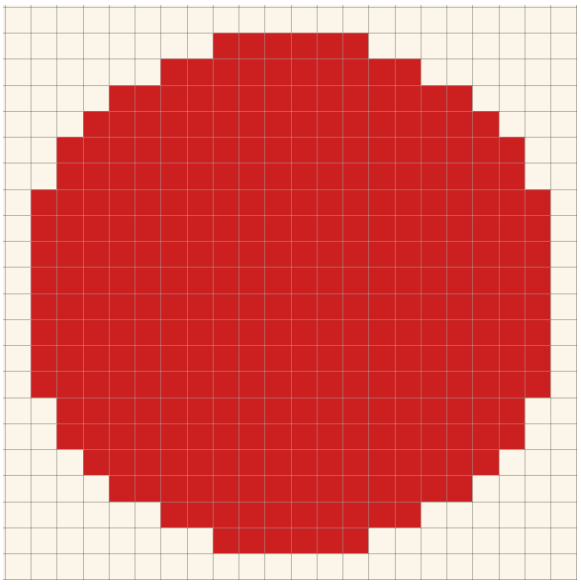

Figure 3: Bresenham 알고리즘, $D = 20$.

5.2 타원의 경우 (두 영역)

반축이 a, b 인 타원에 대해, 알고리즘은 접선의 기울기가 $45°$ 인 점을 경계로 제 1 사분면을 **두 영역** 으로 나눕니다:

$$\text{영역 I: } 2b^2x < 2a^2y \quad (|dy/dx| < 1), \qquad \text{영역 II: 그 외}$$

초기 판정 매개변수는:

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

각 반복마다 p 는 미리 계산된 증분으로 갱신됩니다 (4 를 곱한 뒤에는 모두 정수). 픽셀당 산술량은 가능한 최소입니다: 부호 판정 1 회와 덧셈 2 회. 코드는 r_y 가 정수가 아닐 때 *bounding box* 바깥에 여분의 행이 추가되지 않도록, r_x, r_y 로부터 a, b 를 계산할 때 `Math.floor` 를 사용합니다.

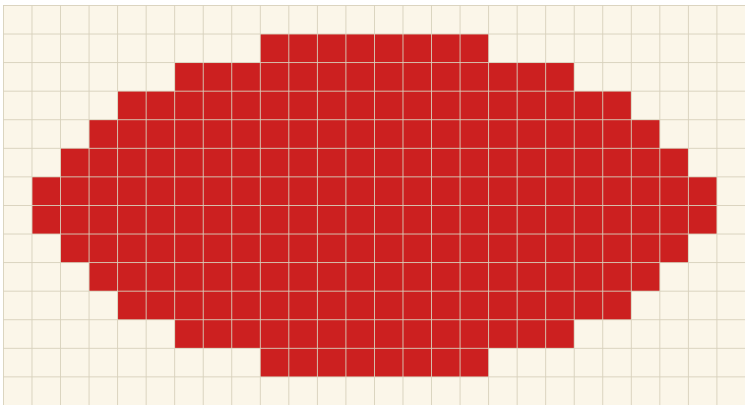

Figure 4: Bresenham 알고리즘을 타원 ($W = 24$, $H = 12$) 에 적용.

6. 임계값 알고리즘 (Threshold / 모서리 포함)

이 알고리즘은 다음을 묻습니다: 셀의 모서리 중에 타원 안에 있는 것이 있는가? 각 셀 (i, j) 에 대해 코드는 도형의 중심에 가장 가까운 모서리 (셀의 변에 대한 중심의 사영) 를 찾습니다:

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

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

값 $0{,}94$ 는 $1{,}0$ 보다 약간 작은 경험적 임계값 입니다. 그 역할은 사방위 (N, S, E, W) 에서 발생하는 1 픽셀짜리 접선 스파이크 아티팩트를 제거하는 것입니다. 이때 곡선은 셀의 한 변 전체와 접합니다. 이 감소 처리가 없으면 알고리즘은 도형의 시각적 표현을 개선하지 않는 추가 셀까지 포함하게 됩니다.

세 알고리즘 중에서 동일한 지름 D 에 대해 가장 넓은 면적의 실루엣을 만드는 것이 이 알고리즘입니다. 포함 조건이 가장 느슨하기 때문입니다: 셀의 모서리 하나만이라도 타원의 부등식을 만족시키면 셀 전체가 채워집니다.

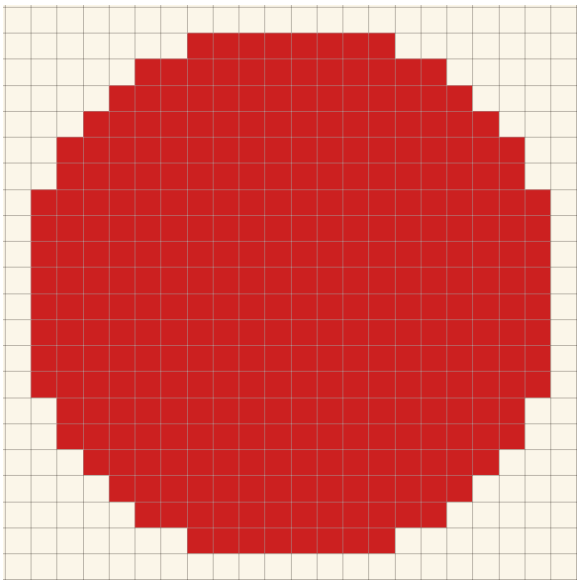

Figure 5: 임계값 알고리즘, $D = 20$.

7. 렌더링 모드: 채움, 가는, 두꺼운

내부 셀들을 표시하는 행렬 $f[j][i] = 1$ 을 얻은 뒤, 프로젝트는 **이산 토폴로지** 에 의해 세 가지 시각 스타일을 유도합니다:

7.1 채움

f 를 그대로 반환합니다. 모든 내부 픽셀.

7.2 가는 (outline)

픽셀이 채워져 있고 또한 도형의 외부 에 직교 이웃 (N, S, E, W) 중 적어도 하나를 가지면 가는 윤곽에 속합니다. 논리적 표기로:

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

기하학적으로는: 도형의 **이산 경계** 이며, 이는 위상 경계 $\partial F = F \cap \overline{F^{\mathrm{c}}}$ 의 이산 대응물입니다.

7.3 두꺼운 (thick)

두꺼운 모드는 경계 픽셀에 **대각 다리** 를 추가합니다: 수평 경계 이웃과 수직 경계 이웃을 동시에 가지는 내부 셀입니다. 이는 가는 모드가 열어 두는 1 픽셀의 대각 빈틈을 메워, 윤곽이 장면 안에서 안정적으로 보여야 할 때 ($45°$ 의 구멍이

없도록) 유용합니다.

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

8. 이산 면적의 계산

함수 `area2D` 는 단지 f 에서 표시된 셀들을 **셉니다**. 이것은 셈 측도 로서, 타원 지시함수의 리만 적분을 근사합니다:

$$\underbrace{\pi\, r_x\, r_y}_{\text{연속 면적}} \qquad \longleftrightarrow \qquad \underbrace{\sum_{j=0}^{G_y-1} \sum_{i=0}^{G_x-1} f(i,j)}_{\text{이산 면적}}$$

D 가 크면 이산 면적/연속 면적 $\to 1$ 입니다. D 가 작으면 알고리즘 간의 차이가 시각적으로 드러납니다: Threshold 는 과대평가하는 경향이 있고, 유클리드는 이상적 면적에 가까이 유지되며, Bresenham 은 그 사이 어디쯤에 위치합니다.

9. 2D 에서 3D 로: 타원체의 방정식

3D 에서 기본 도형은 (c_x, c_y, c_z) 를 중심으로 반축 r_x, r_y, r_z 를 가지는 **타원체** 입니다. 그 방정식은 타원 방정식의 자연스러운 일반화입니다:

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

$r_x = r_y = r_z$ 일 때 **구** 로 돌아갑니다. 각 복셀의 중심에서 평가하면:

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

10. 복셀화와 부피 계산

타원체의 연속 부피는 적분학의 고전적 결과로, 원판을 통한 적분으로 얻어집니다:

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

이산 버전은 상자 $D_x \times D_y \times D_z$ 의 모든 복셀을 순회하며 $a_x^2 + a_y^2 + a_z^2 \le 1$ 일 때 카운터를 증가시킵니다. 이것이 `voxelVolume` 알고리즘입니다.

껍질 (shell) vs. 내부. *filled* 모드에서 프로젝트는 보존된 집합 바깥에 6- 이웃을 적어도 하나 가지는 복셀들만 표시합니다 (즉, 외부 표면이나 절단면에서 보이는 복셀들). *thin* 모드에서는 완전한 타원체의 **표면** 만 표시됩니다 (두께 1 복셀).

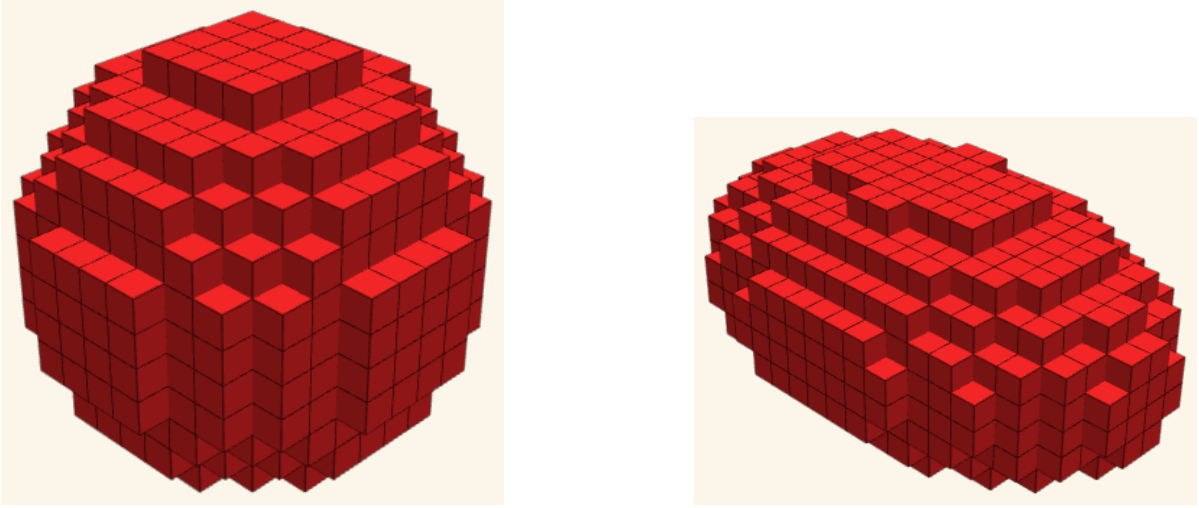

Figure 6: 복셀화된 구 ($D = 12$) 와 타원체 ($W = 20$, $H = 10$, $D = 12$).

11. 기하학적 절단: 평면과 45° 대각 절단

Pixel Round 는 **세 개의 평면** 으로 입체를 절단할 수 있습니다. 각 절단은 복셀을 버리는 선형 부등식 입니다:

$$\begin{array}{lll} \text{X 축:} & x \ge cut & \Rightarrow \text{ 버려짐} \\ \text{Y 축:} & y \ge cut & \Rightarrow \text{ 버려짐} \\ \text{대각:} & x + y \ge cut & \Rightarrow \text{ 버려짐} \end{array}$$

X, Y 평면은 좌표축에 수직이며, 법선 벡터는 각각 $(1,0,0)$ 과 $(0,1,0)$ 입니다. 대각 평면 은 법선 벡터가 $(1,1,0)/\sqrt{2}$ 이며 XY 평면 안에서 45° 로 절단합니다: 수학적으로는 평면족 $x + y = k$ 입니다. $D_x \times D_y$ 상자에서 $x + y$ 의 최댓값은 $D_x + D_y$ 이므로, 대각 절단 슬라이더의 범위는 $D_x + D_y$ 이고, X 와 Y 의 범위는 각각 D_x 와 D_y 입니다. 절단은 **비례적** 입니다: 코드는 비율 $cutPct$ 를 저장하며 한계값을 다음과 같이 다시 계산합니다

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

이로써 사용자가 축을 전환하거나 크기를 조정해도 한 축에서의 50% 는 그대로 50% 로 유지됩니다.

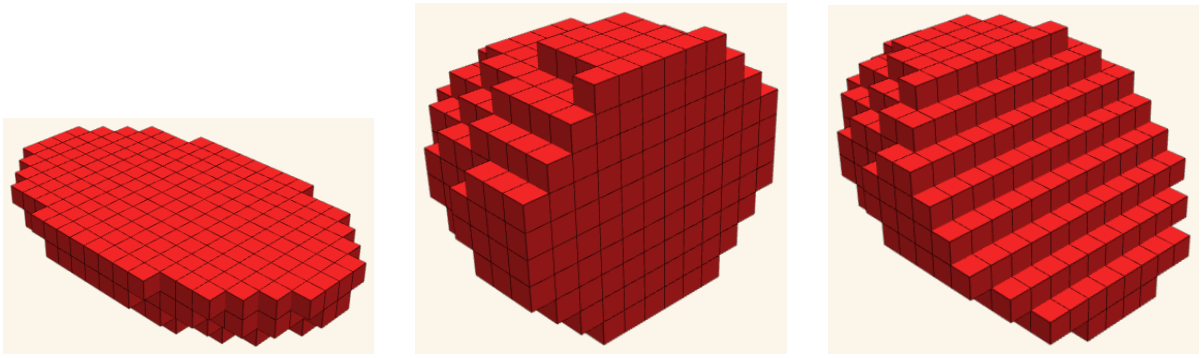

Figure 7: Y, X, 대각선 축에서의 50% 평면 절단 (왼쪽에서 오른쪽).

12. 3D 두꺼운 모드: 슬라이스에 의한 입체화

3D *thick* 모드에서 프로젝트는 세 축을 따라 **모든 슬라이스** 에 2D 두꺼운 알고리즘을 적용합니다: z 마다 XY 슬라이스, y 마다 XZ 슬라이스, x 마다 YZ 슬라이스. 그 다음 결과들의 합집합 을 취합니다. 기하학적 동기는 기밀성입니다: 껍질의 두께를 두 배로 만들지 않으면서 모든 대각선을 막는 복셀의 최소 집합입니다. 형식적으로 $S_z(z)$ 를 높이 z 의 XY 슬라이스라 하고 T 를 2D 두꺼운 연산자라 하면:

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

결과는 이후 절단으로 보존된 집합과 교차됩니다. 이론적 배경은 **디지털 토폴로지** 입니다: 두꺼운 연산자는 껍질의 *26-* 연결성 을 보장하며 (대각 이웃을 포함), 대각으로 인접한 복셀이 면으로 접촉하지 않는 Minecraft 의 건축에 유용합니다.

13. 3D 카메라: 구면 좌표

3D 카메라는 거리 r 로 객체를 공전하며 두 개의 각도로 매개화됩니다 — θ (수평면 안의 방위각) 과 φ (수직 축에서 측정한 극각). 이는 구면 좌표의 **물리학적 약속** 이며, 직교 좌표 (*three.js* 가 이해하는 형식) 로의 변환은 다음과 같습니다:

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

초기 위치: $\theta = \pi/4$ (45°, 등각투상 시각) 과 $\varphi = \pi/3$ (북극에서 60°, 적도보다 약간 위). 마우스 드래그는 θ, φ 를 직접 증가시키며; 휠/핀치는 r 을 증가시킵니다. 이 매개화의 단순성은 사원수나 명시적 행렬 없이 자유로운 회전을 가능하게 합니다.

14. 자동 줌: 둘러싸는 구와 FOV

사용자가 도형의 크기를 바꾸거나 카메라를 재설정하면, 프로젝트는 어떤 각도에서도 객체가 화면에 맞도록 카메라의 **이상적 거리** 를 다시 계산합니다. 발상은 객체를 반지름 R 의 구 (AABB, axis-aligned bounding box 의 대각선의 절반) 안에 가두는 것입니다:

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

구는 **회전에 대해 불변한 투영** 을 가지므로, 한 방향에서 화면에 맞으면 모든 방향에서 맞습니다. 카메라의 수직 FOV ($fov_V = 35°$, 라디안 단위) 가 주어지면 반지름 R 의 구를 화면에 담는 데 필요한 거리는:

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

수평 FOV 는 캔버스의 종횡비 ($aspect =$ 너비/높이) 로부터 다음과 같이 구해집니다:

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

최종 거리는 $dist_V$ 와 $dist_H$ 의 최댓값에 1,20 을 곱한 값입니다 (시각적 여백 20%). 도형이 절단되어 있을 때 코드는 D_x 나 D_y 를 남은 크기로 줄여서, 75% 절단된 구가 화면 구석에서 작아 보이지 않도록 합니다.

15. 셰이딩과 RGB 색 산술

세 가지 3D 스타일 (*Classic*, *Smooth*, *Blocks*) 은 각 복셀의 세 가시면 (윗면, 빛받는 면, 그림자 면) 에 적용되는 **곱셈 인자** 로 구분됩니다. 함수 `shadeHex(hex, factor)` 는 16 진값을 (R, G, B) 로 파싱하고 각 채널에 인자를 곱한 뒤 $[0, 255]$ 로 클램프합니다:

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

이것은 이상적인 람베르트 조명 함수의 선형 근사 입니다,

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

세 인자가 각각 면의 법선과 고정 광원 방향이 이루는 각의 코사인에 해당할 때 위 식이 성립합니다. *Smooth* 에서는 인자들이 가깝습니다 (면들이 비슷한 색); *Classic* 에서는 인자들의 차이가 큽니다 (강한 대비). *Blocks* 는 추가로 복셀에 기하학적 함몰을 도입합니다 — 각 면을 일정량 안쪽으로 평행이동시킵니다.

16. 오디오: 주파수와 음계

인터페이스의 효과음은 WebAudio 로 실시간 합성됩니다. 각 효과음은 **순수한 사인파** 입니다:

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

프로젝트가 선택한 f (Hz) 의 값들은 임의가 아닙니다 — 그것들은 *12* 평균율 의 음을 근사합니다. 거기서 각 반음은 주파수에 $2^{1/12} \approx 1{,}0595$ 를 곱합니다:

효과음	주파수 (Hz)	가까운 음	용도
click	620	≈ D♯5 / E♭5	버튼
hover	1100	≈ C♯6	데스크톱 호버
tick	1300	≈ E6	슬라이더 단계
ok	520 → 720 → 920	상행 아르페지오	리셋 / 다운로드
pop	680 + 980	완전 5 도 음정	토글
error	200	≈ G♯3	잘못된 동작

완전 5 도 (비 3 : 2): *pop* 효과음은 680 Hz 와 980 Hz 를 섞습니다; $980/680 \approx 1{,}44$ (순정 완전 5 도라면 1,5) 로, 협화적으로 들릴 만큼 가깝습니다. ***ok* 효과음의 아르페지오** 는 잇따른 비가 단 3 도에 가깝습니다: $520 \to 720$ (비 1,38, 평균율 단 3 도 1,2 를 약간 넓힌 값) 와 $720 \to 920$, 결론적으로 들리는 상행 진행을 형성합니다. 포락선의 길이는 12 ms 에서 160 ms 사이이며 최고 이득은 0,035 (약 −29 dBFS), 장시간 사용 시 청각 피로를 피하기 위해 의도적으로 낮춘 수준입니다.

17. 결론

본 문서는 Pixel Round 에서 사용되는 수학적 기초를 체계적으로 기술하였습니다. 약 1000 줄의 JavaScript 코드 안에서 본 프로젝트는 여러 분야로부터의 기여를 통합합니다:

- **해석기하**: 타원과 타원체의 음함수 방정식을 일차 포함 판정 기준으로 사용합니다.
- **고전적 래스터화 알고리즘**: 중점 형식의 Bresenham, 그리고 타원에 대한 Pitteway/Kappel 확장.
- **디지털 토폴로지**: 이산 경계 연산자, 4-, 6-, 26- 연결성 개념과 슬라이스에 의한 합성.
- **적분학**: 타원체 부피의 삼중 적분 표현과 그 이산 대응 (복셀에 대한 셈 측도).

- **선형대수**: 선형 부등식으로 정의되는 절단 평면과 3 차원 원근 투영.
- **삼각법**: 구면 좌표에서의 카메라 매개화 및 FOV 에 따른 거리 자동 조정.
- **신호 합성**: 순수한 사인파와 12 평균율 근사.

본 연구가 이끌어 내는 중심 관찰은 다음과 같습니다: 이산 격자 위에서는 연속 곡선의 **유일하게 올바른 표현이 존재하지 않습니다**. 본 문서에서 제시된 각 알고리즘은 셀 포함 기준에 대한 서로 다른 수학적 정의에 대응하며, 각 정의는 고유의 형식적 성질을 가지는 실루엣을 만들어냅니다 — 유클리드 알고리즘의 매끄러움, Bresenham 의 계단 패턴, 모서리 포함 기준의 더 넓은 덮개. 따라서 알고리즘의 선택은 의도하는 시각적 표현 목표와 결과 도형에 요구되는 정량적 특성을 함께 고려해야 하는 기술적 결정입니다.