Pixel Round

Математика проекта

Техническая и учебная документация математических основ, применяемых в генераторе кругов, эллипсов, сфер и эллипсоидов для пиксель-арта и воксель-арта. Для каждого понятия приводятся формальное определение, геометрическое обоснование и прямое соответствие с исходным кодом проекта.

Рассматриваемые области: аналитическая геометрия · дискретная растеризация · цифровая топология · интегральное исчисление · линейная алгебра · тригонометрия · синтез звука.

Содержание

1. Обзор: математическая природа задачи

Pixel Round — это генератор скруглённых форм для **пиксель-арта** и **воксель-арта**: окружностей и эллипсов в 2D, сфер и эллипсоидов в 3D. Центральная задача относится к классу *дискретной растеризации*: для кривой или поверхности, определённой аналитически на действительных числах $\mathbb{R}$, требуется определить, какие ячейки регулярной сетки (пиксели в 2D, воксели в 3D) должны быть отмечены для её представления.

Эта задача не имеет единственного решения. Различные критерии включения порождают различные дискретные силуэты, и каждый критерий имеет собственное математическое обоснование. По этой причине проект реализует три различных 2D-алгоритма, три режима отрисовки и три стиля 3D-затенения, которые описаны в последующих разделах.

Выполнение происходит полностью в браузере, без серверной части, что налагает на математическую формулировку дополнительное требование *вычислительной эффективности*. Используемые теоретические области:

- аналитическая геометрия конических сечений и квадрик;
- инкрементальные алгоритмы с целочисленной арифметикой (Брезенхэм);
- цифровая топология (4-, 6- и 26-связные окрестности);
- интегральное исчисление и считающая мера;
- линейная алгебра (секущие плоскости, перспективная проекция);
- тригонометрия и сферические координаты;
- синтез сигналов и основы акустики.

2. Система координат и дискретная сетка пикселей

Холст — это сетка из $G_x \times G_y$ ячеек. Каждая ячейка (i, j) занимает квадрат $[i, i+1] \times [j, j+1]$. В непрерывном описании **центр** ячейки находится в $(i + \frac{1}{2},\ j + \frac{1}{2})$. Это соглашение принципиально: сравнение окружности с *углом* (i, j) или с *центром* $(i + \frac{1}{2}, j + \frac{1}{2})$ изменяет множество ячеек, считающихся принадлежащими форме.

$$\text{центр ячейки } (i, j) \;=\; (i + \tfrac{1}{2},\ \ j + \tfrac{1}{2})$$

Для формы диаметра D код расширяет сетку до $D+2$ ячеек по каждой оси (поле в одну ячейку с каждой стороны). Это обеспечивает, что крайние пиксели (С, Ю, В, З) не оказываются на границе матрицы. **Центр** формы находится в $(G_x/2,\ G_y/2)$, а *полуоси* равны $r_x = D/2$ и $r_y = D/2$ (или $W/2$ и $H/2$ для эллипса).

3. Основное уравнение: окружность и эллипс

Вся теория проекта начинается с неявного уравнения эллипса с центром в начале координат и полуосями r_x и r_y:

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

Когда $r_x = r_y = r$, эллипс вырождается в **окружность** радиуса r. Точки *внутренней* области удовлетворяют ≤ 1; точки *внешней* области — > 1. Это и есть неравенство, определяющее принадлежность пикселя форме. Три алгоритма по существу представляют собой три различных способа применять (или приближать) этот тест на сетке.

Сместив центр в (c_x, c_y) и вычисляя в центре каждой ячейки:

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

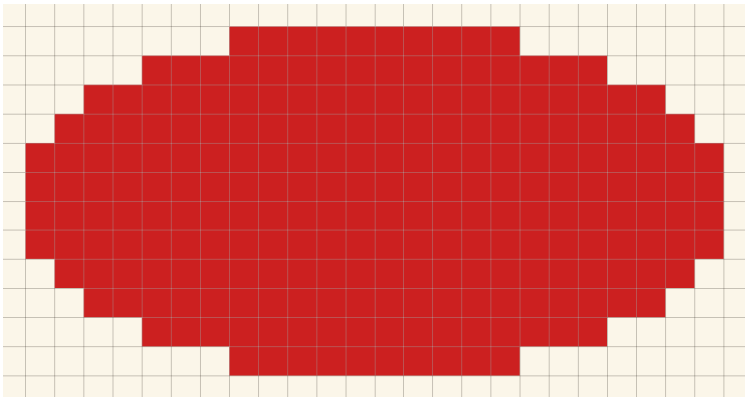

Рис. 1: Эллипс с $W = 24$ и $H = 12$ на дискретной сетке.

4. Евклидов алгоритм (тест расстояния)

Идея. Для каждой строки j аналитически найти столбцы i, лежащие внутри эллипса. Решая уравнение эллипса относительно x при данном y:

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

Для строки j с вертикальным смещением $d_y = j + \frac{1}{2} - c_y$ *горизонтальная полуширина* эллипса на этой высоте равна:

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

Если $d_y^2 / r_y^2 \geq 1$, строка не пересекает эллипс и пропускается. Иначе заполняются все столбцы i, для которых $c_x - dxM \leq i + \frac{1}{2} \leq c_x + dxM$. Целочисленные границы получаются как:

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

Обоснование. Слагаемое $+\frac{1}{2}$ возникает из соглашения, согласно которому центр ячейки с индексом i находится в $i + \frac{1}{2}$. Функции $\lceil \cdot \rceil$ и $\lfloor \cdot \rfloor$ преобразуют непрерывный интервал в целочисленные индексы, гарантируя, что включаются только те ячейки, **центр** которых принадлежит внутренности эллипса. Эта формулировка даёт дискретное приближение, наиболее близкое к непрерывной кривой, и, следовательно, силуэт с наибольшей визуальной гладкостью. Однако для малых диаметров результат может проявлять менее выраженный пиксель-арт-характер по сравнению с другими алгоритмами.

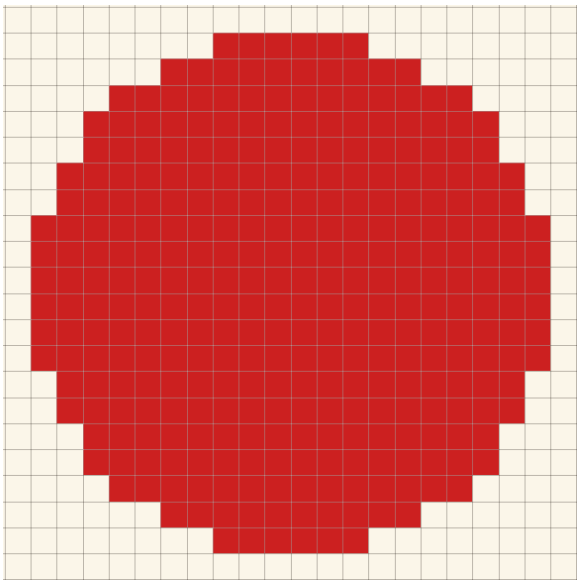

Рис. 2: Евклидов алгоритм при $D = 20$.

5. Алгоритм Брезенхэма (целочисленная середина)

Алгоритм Брезенхэма, опубликованный в 1965 году для рисования отрезков, был впоследствии расширен на окружности и обобщён Питтуэем и Каппелем для произвольных эллипсов. Его отличительное свойство — отказ от операций с плавающей точкой и извлечения квадратного корня: определение следующего пикселя использует только **сложения и сравнения целых чисел** посредством *функции решения*, оценивающей, с какой стороны аналитической кривой расположена середина между двумя пикселями-кандидатами.

5.1 Случай окружности

Для окружности целочисленного радиуса R начинаем из $(x{=}0,\ y{=}R)$ с начальным параметром решения:

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

На каждом шаге (в октанте от 0 до $45°$):

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

Обоснование. Функция d на каждой итерации приближает значение неявного уравнения окружности в **средней точке** между двумя пикселями-кандидатами:

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

Знак d показывает, лежит ли средняя точка внутри окружности (в этом случае продвигаемся только по горизонтали) или вне (в этом случае также уменьшается вертикальная координата). Восемь октантов окружности получаются применением симметрий диэдральной группы D_4 к вычисленному октанту, что в коде соответствует восьми вызовам `span(...)`. Получаемый след имеет ступенчатый узор, характерный для дискретных приближений гладких кривых.

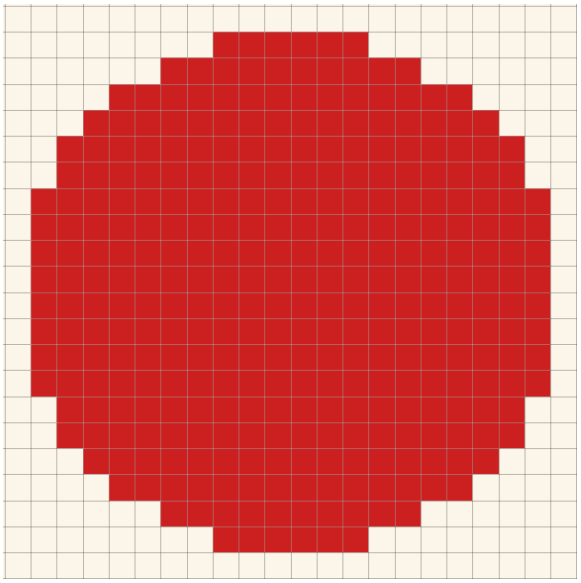

Рис. 3: Алгоритм Брезенхэма при $D = 20$.

5.2 Эллиптический случай (две области)

Для эллипса с полуосями a, b алгоритм делит первый квадрант на **две области**, разграниченные точкой, в которой касательная имеет угол $45°$:

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

Начальные параметры решения:

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

На каждой итерации p обновляется заранее вычисленными приращениями (все целые после умножения на 4). Результат — минимально возможная арифметика на пиксель: *одна проверка знака и два сложения*. Код использует `Math.floor` при вычислении a и b из r_x и r_y, чтобы избежать добавления лишней строки за пределами *bounding box* при нецелом r_y.

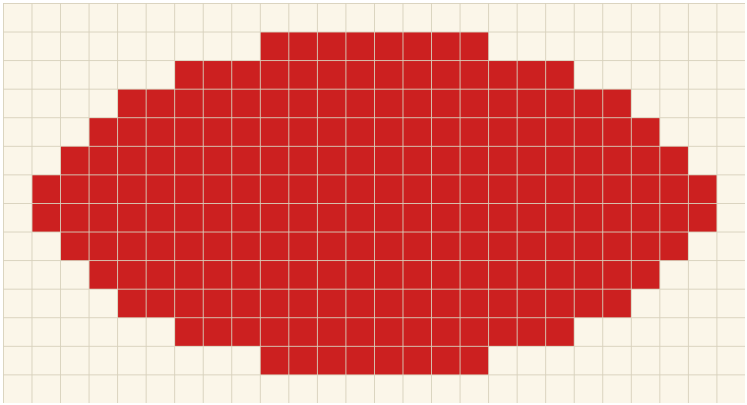

Рис. 4: Алгоритм Брезенхэма применённый к эллипсу $W = 24$, $H = 12$.

6. Пороговый алгоритм (Threshold / покрытие угла)

Этот алгоритм задаёт вопрос: *есть ли хотя бы один угол ячейки внутри эллипса?* Для каждой ячейки (i, j) код находит угол, ближайший к центру формы (проекция центра на рёбра ячейки):

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

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

Значение $0{,}94$ — *эмпирический порог*, чуть меньше $1{,}0$. Его роль — устранить артефакт, известный как *касательный пик* в 1 пиксель, возникающий в четырёх кардинальных точках (С, Ю, В, З), где кривая касается целого ребра ячейки. Без этого снижения алгоритм включал бы дополнительную ячейку, вклад которой не улучшает визуальное представление формы.

Среди трёх алгоритмов этот даёт силуэт наибольшей площади при том же диаметре D, поскольку условие включения наименее ограничительно: достаточно, чтобы один угол ячейки удовлетворял неравенству эллипса, и вся ячейка заполняется.

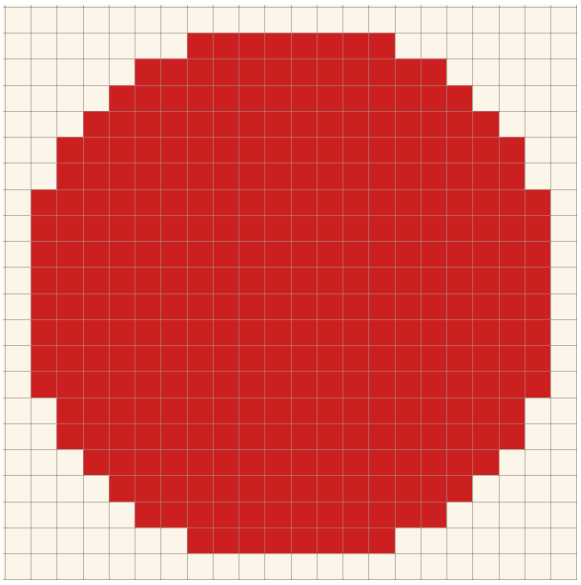

Рис. 5: Пороговый алгоритм при $D = 20$.

7. Режимы отрисовки: Заполнено, Тонкая, Толстая

Имея матрицу $f[j][i] = 1$ для внутренних ячеек, проект выводит три визуальных стиля посредством **дискретной топологии**:

7.1 Заполнено

Возвращает f без изменений. Все внутренние пиксели.

7.2 Тонкая (outline)

Пиксель принадлежит тонкому контуру, если он заполнен *и* имеет хотя бы одного ортогонального соседа (С, Ю, В, З) *вне* формы. В логической записи:

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

Геометрически: это **дискретная граница** формы, дискретный аналог топологической границы $\partial F = F \cap \overline{F^{\mathrm{c}}}$.

7.3 Толстая (thick)

Толстый режим добавляет к граничным пикселям **диагональные мосты**: внутренние ячейки, у которых одновременно есть горизонтальный *и* вертикальный граничный сосед. Это закрывает диагонали в 1 пиксель, которые тонкий

режим оставляет открытыми; полезно, когда контур должен выглядеть устойчиво в сцене (без отверстий под $45°$).

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

8. Вычисление дискретной площади

Функция `area2D` просто **подсчитывает** ячейки, отмеченные в f. Это *считающая мера*, приближающая интеграл Римана от индикаторной функции эллипса:

$$\underbrace{\pi\, r_x\, r_y}_{\text{непрерывная площадь}} \quad \longleftrightarrow \quad \underbrace{\sum_{j=0}^{G_y-1} \sum_{i=0}^{G_x-1} f(i,j)}_{\text{дискретная площадь}}$$

При большом D дискретная площадь/непрерывная площадь $\to 1$. При малом D различие между алгоритмами заметно: Threshold склонен к завышению; Евклидов остаётся близко к идеальной площади; Брезенхэм находится где-то посередине.

9. От 2D к 3D: уравнение эллипсоида

В 3D основная форма — **эллипсоид** с центром в (c_x, c_y, c_z) и полуосями r_x, r_y, r_z. Его уравнение — естественное обобщение уравнения эллипса:

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

При $r_x = r_y = r_z$ возвращаемся к **сфере**. Вычисляя в центре каждого вокселя:

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

10. Вокселизация и вычисление объёма

Непрерывный объём эллипсоида — классический результат интегрального исчисления, получаемый интегрированием по дискам:

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

Дискретная версия проходит по каждому вокселю коробки $D_x \times D_y \times D_z$ и увеличивает счётчик при $a_x^2 + a_y^2 + a_z^2 \le 1$. Это алгоритм `voxelVolume`.

Оболочка (shell) vs. внутренность. В режиме *filled* проект отображает только воксели, имеющие хотя бы одного 6-соседа вне сохранённого множества (то есть воксели, видимые с внешней поверхности или с грани среза). В режиме *thin* отображается только **поверхность** полного эллипсоида (толщиной 1 воксель).

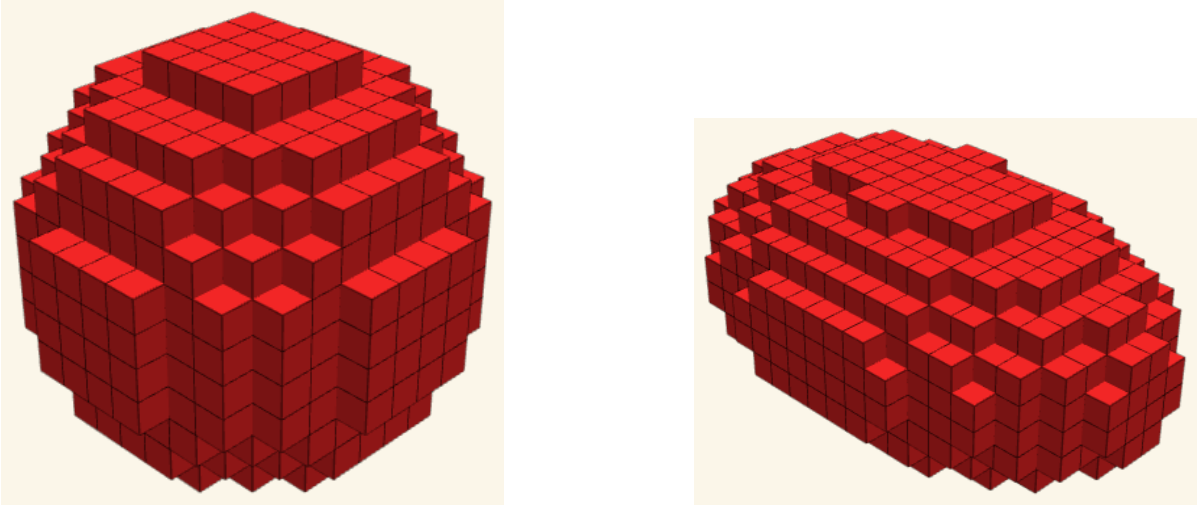

Рис. 6: Воксельная сфера ($D = 12$) и эллипсоид ($W = 20,\ H = 10,\ D = 12$).

11. Геометрические срезы: плоскости и диагональный срез $45°$

Pixel Round позволяет срезать тело **тремя плоскостями**. Каждый срез — это *линейное неравенство*, отбрасывающее воксели:

Ось X:	$x \ge cut$	$\Rightarrow$	отброшено
Ось Y:	$y \ge cut$	$\Rightarrow$	отброшено
Диагональ:	$x + y \ge cut$	$\Rightarrow$	отброшено

Плоскости X и Y перпендикулярны осям координат, с нормалями $(1, 0, 0)$ и $(0, 1, 0)$. *Диагональная плоскость* имеет нормаль $(1, 1, 0)/\sqrt{2}$ и срезает под углом $45°$ в плоскости XY: математически это семейство плоскостей $x + y = k$. Поскольку максимум $x + y$ в коробке $D_x \times D_y$ равен $D_x + D_y$, ползунок диагонального среза имеет диапазон $D_x + D_y$, тогда как X и Y — диапазоны D_x и D_y соответственно. Срез **пропорциональный**: код хранит долю *cutPct* и пересчитывает предел как

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

так что 50% по одной оси остаются 50% при переключении оси или изменении размера.

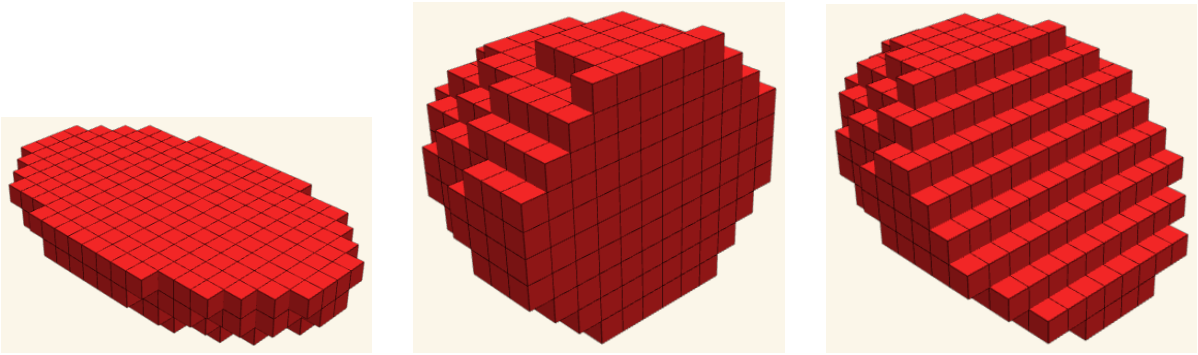

Рис. 7: Плоские сечения 50% по осям Y, X и диагонали (слева направо).

12. Толстый 3D-режим: триангуляция срезами

Для режима *thick* в 3D проект применяет 2D-толстый алгоритм ко **всем срезам** вдоль трёх осей: XY для каждого z, XZ для каждого y, YZ для каждого x. Затем берётся *объединение* результатов. Геометрическая мотивация — герметичность: минимальное множество вокселей, перекрывающее все диагонали без удвоения толщины оболочки. Формально, обозначая $S_z(z)$ срез XY на высоте z и T — 2D-толстый оператор:

$$\mathit{thick3D} \;=\; \bigcup_{z} T(S_z(z)) \;\cup\; \bigcup_{y} T(S_y(y)) \;\cup\; \bigcup_{x} T(S_x(x))$$

Результат затем пересекается с множеством, сохранённым срезом. Лежащая в основе теория — **цифровая топология**: толстый оператор гарантирует *26-связность* оболочки (соседи, включая диагональных), что полезно для построения в Minecraft, где диагональные воксели не соприкасаются гранями.

13. 3D-камера: сферические координаты

3D-камера обращается вокруг объекта на расстоянии r с двумя углами — θ (азимут в горизонтальной плоскости) и φ (полярный угол от вертикальной оси). Это **физическое соглашение** сферических координат, а преобразование в декартовы (форма, принимаемая *three.js*) имеет вид:

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

Начальное положение: $\theta = \pi/4$ ($45°$, изометрическая перспектива) и $\varphi = \pi/3$ ($60°$ от северного полюса, чуть выше экватора). Перетаскивание мыши непосредственно увеличивает θ и φ; колесо/щипок увеличивает r. Простота этой параметризации позволяет свободно вращать сцену без кватернионов или явных матриц.

14. Авто-зум: ограничивающая сфера и FOV

Когда пользователь меняет размер фигуры или сбрасывает камеру, проект пересчитывает **идеальное расстояние** камеры так, чтобы объект помещался в кадр при любом угле. Идея — заключить объект в *сферу радиуса* R (половина диагонали AABB, axis-aligned bounding box):

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

Сфера имеет **проекцию, инвариантную при вращении**, поэтому если она помещается в кадр в одной ориентации, то помещается во всех. При вертикальном FOV камеры ($fov_V = 35°$ в радианах) расстояние, необходимое для размещения сферы радиуса R:

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

Горизонтальный FOV получается из соотношения сторон холста (*aspect* = ширина/высота):

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

Итоговое расстояние — максимум из $dist_V$ и $dist_H$, умноженный на $1{,}20$ (20% визуального запаса). Когда фигура срезана, код уменьшает D_x или D_y до оставшегося размера, чтобы сфера, обрезанная на 75%, не оказывалась крошечной в углу.

15. Затенение (shading) и арифметика цвета RGB

Три 3D-стиля (*Classic*, *Smooth*, *Blocks*) различаются **множителями**, применяемыми к трём видимым граням каждого вокселя (верх, освещённая сторона, тёмная сторона). Функция `shadeHex(hex, factor)` разбирает hex на (R, G, B) и умножает каждый канал, ограничивая значениями в $[0, 255]$:

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

Это *линейное приближение* идеальной функции освещения Ламберта,

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

когда три множителя соответствуют косинусам угла между нормалью каждой грани и фиксированным направлением света. В *Smooth* множители близки (одинаковые грани); в *Classic* множители различаются сильнее (сильный контраст). *Blocks* вдобавок вводит геометрическое углубление в воксели — постоянный сдвиг каждой грани внутрь.

16. Аудио: частоты и музыкальный ряд

Звуки интерфейса синтезируются в реальном времени через WebAudio. Каждый звук — это **чистая синусоидальная волна**:

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

Выбранные проектом значения f (в Гц) не случайны — они приближают музыкальные ноты *12-тоновой равномерной темперации*, в которой каждый полутон умножает частоту на $2^{1/12} \approx 1{,}0595$:

Звук	Частота (Гц)	Ближайшая нота	Назначение
click	620	≈ D#5 / E♭5	кнопки
hover	1100	≈ C#6	наведение на десктопе
tick	1300	≈ E6	шаги ползунка
ok	$520 \to 720 \to 920$	восходящее арпеджио	сброс / скачивание
pop	$680 + 980$	интервал чистой квинты	переключения
error	200	≈ G#3	недопустимое действие

Чистая квинта (отношение $3:2$): звук *pop* смешивает 680 Гц и 980 Гц; $980/680 \approx 1{,}44$ (чистая квинта была бы $1{,}5$), достаточно для консонантного звучания. **Арпеджио звука *ok*** даёт последовательные отношения, близкие к терциям: $520 \to 720$ (отношение $1{,}38$, близко к темперированной малой терции $1{,}2$, немного расширенной) и $720 \to 920$, конфигурация, формирующая восходящую каденцию, воспринимаемую как заключительную. Огибающая имеет длительность от 12 до 160 мс и пиковое усиление $0{,}035$ (около -29 dBFS), уровень намеренно низок, чтобы избежать слуховой усталости при длительном использовании.

17. Заключение

Настоящий документ систематически описал математические основы, применяемые в Pixel Round. Приблизительно в тысяче строк кода JavaScript проект интегрирует вклад нескольких областей:

- **Аналитическая геометрия**: неявные уравнения эллипса и эллипсоида как первичный критерий принадлежности.
- **Классические алгоритмы растеризации**: Брезенхэм в формулировке средней точки с расширением Питтуэя/Каппеля для эллипсов.
- **Цифровая топология**: операторы дискретной границы, понятия 4-, 6- и 26-связности и композиция срезами.
- **Интегральное исчисление**: объём эллипсоида как тройной интеграл и его дискретный аналог (считающая мера по вокселям).
- **Линейная алгебра**: секущие плоскости, заданные линейными неравенствами, и трёхмерная перспективная проекция.
- **Тригонометрия**: параметризация камеры в сферических координатах и автоматическая регулировка расстояния в зависимости от FOV.
- **Синтез сигналов**: чистые синусоидальные волны и приближение равномерной 12-тоновой темперации.

Центральное наблюдение, которое позволяет сформулировать данное исследование, таково: на дискретной сетке **не существует единственного правильного представления** непрерывной кривой. Каждый представленный в документе алгоритм соответствует отдельному математическому определению критерия включения ячеек, и каждое определение порождает силуэт со своими формальными свойствами — гладкость в Евклидовом алгоритме, ступенчатый узор Брезенхэма, более широкое покрытие в критерии покрытия угла. Выбор алгоритма, следовательно, — техническое решение, которое должно учитывать предполагаемую визуальную цель и желаемые метрические характеристики результирующей формы.