МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

“Санкт-Петербургский национальный исследовательский университет ИТМО”

Факультет информационных технологий и программирования

Проектная работа по дисциплине
“Дополнительные главы физики”

Скретч-голография

Выполнили студенты группы М32081:

Насибуллин Данил Наилевич

Новрогодцев Никита Павлович

Ярощук Владислав Викторович

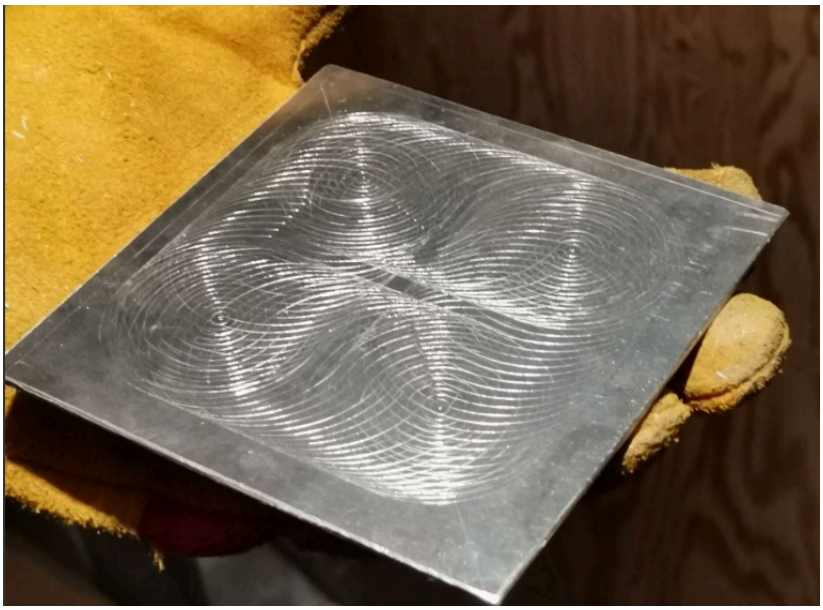
Проверил:

САНКТ-ПЕТЕРБУРГ

2020

Обзор

Набор программно-аппаратных средств для преобразования трехмерных компьютерных моделей объектов в плоский рисунок царапин отражающей пластины, демонстрирующей оптическую иллюзию трехмерного объекта при соблюдении корректных условий наблюдения.



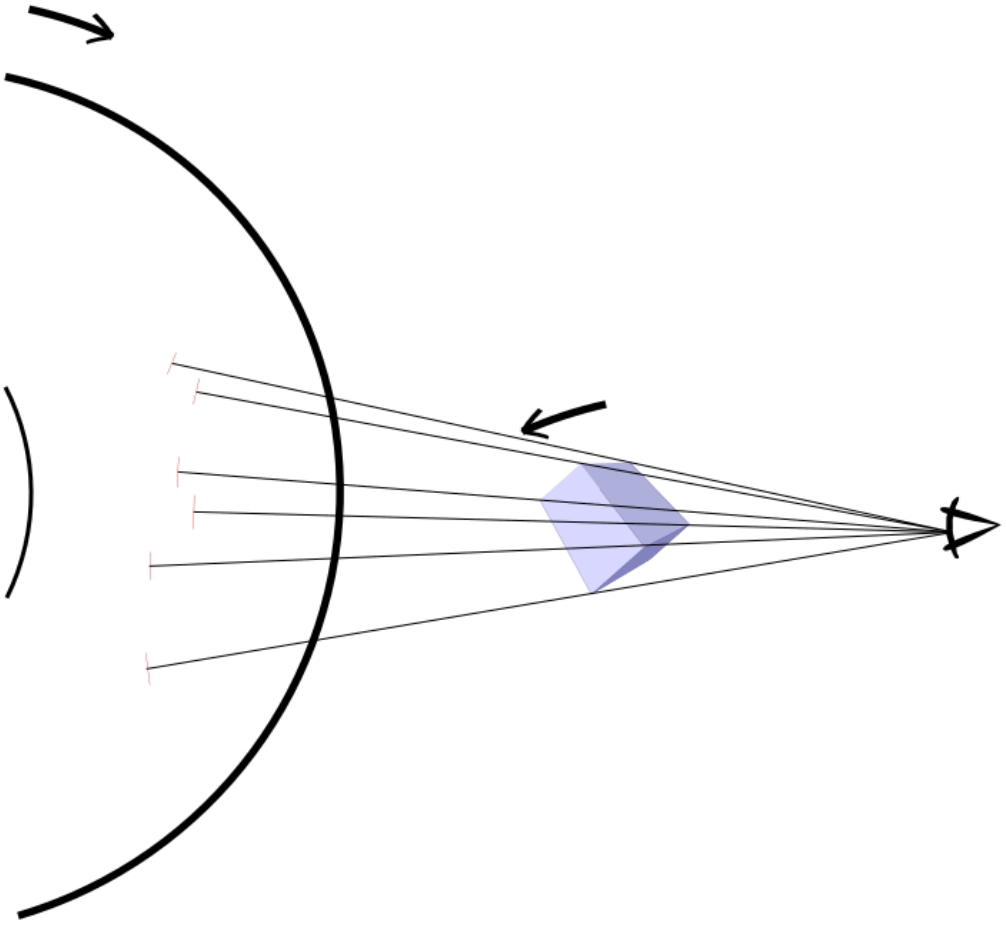
Этапы разработки

1. Изучить природу иллюзии скретч-голограмм
2. Разработать математический алгоритм трансформации трехмерного макета объекта в трехмерный рисунок царапин отражающей голографической пластины
3. Написать программную реализацию алгоритма из п.2
4. Подобрать аппаратные средства для нанесения рисунка царапин на отражающую поверхность

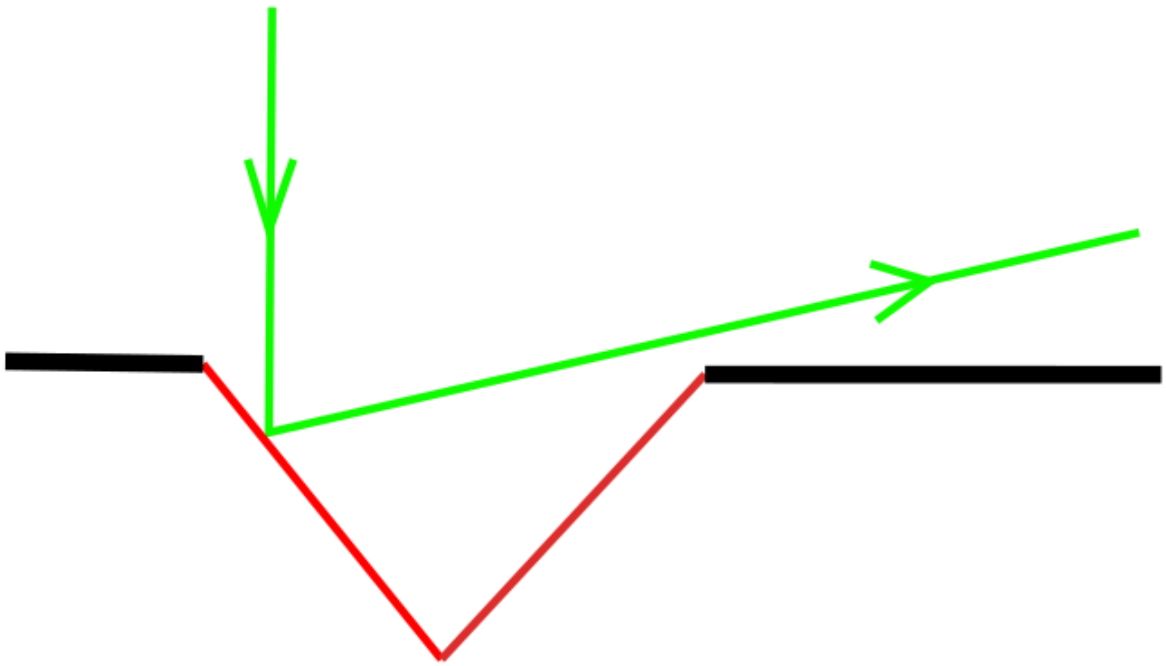
Ход работы

I.Идея реализации алгоритма записи и воспроизведения изображения объекта на пластине

Рассмотрим процесс записи изображения трехмерного объекта на дискообразную пластину: вращаем пластину и проецируем на неё все значимые точки объекта. Для каждой точки проводим царапину малой длины, перпендикулярную проведенному от наблюдателя лучу (отмечены красным цветом).



Для воспроизведения изображения проекции объекта используем источник света с параллельным пучком, расположенный над пластиной. Луч отразится от поверхности царапины и вернется к наблюдателю, воспроизводя изображение точек.



Вращая объект синхронно с пластиной, производится покадровое проецирование всех видимых для наблюдателя значимых точек объекта (точек-вершин и точек, составляющих ребра). После совершения полного оборота запись прекращается, пластина готова к воспроизведению изображения.

II.Математическая модель алгоритма, описанного в п. II хода работы

Необходимые данные об объекте:

- координаты вершин
- координаты внешних нормалей к полигонам (граням)
- границы полигонов, заданные через перечисление номеров вершин

Добавочное условие, упрощающее построение алгоритма:

- центр модели находится в начале координат
- ось вращения - ось **0z**.

Определим положение наблюдателя:

- горизонтальное расстояние от центра диска до наблюдателя **D**
- высота относительно пластины **H**

Дополнительная информация об объекте:

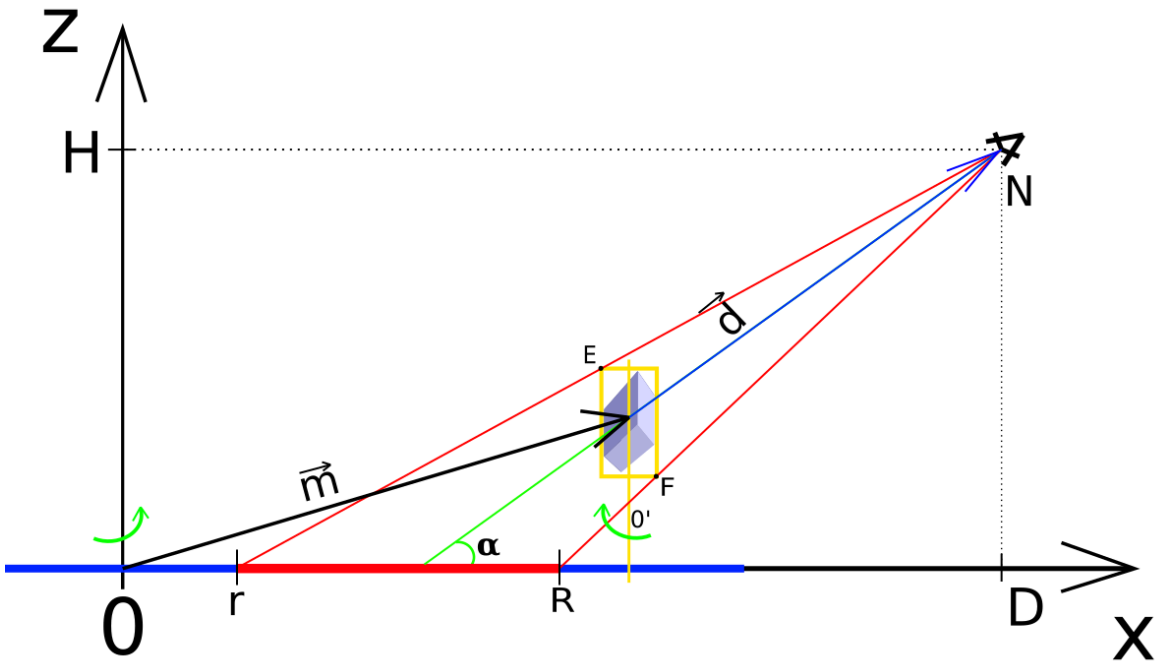
- расстояние от наблюдателя **d**
- угол между лучом наблюдения и горизонтальной поверхностью пластины **α**

Также дополнительно необходимые данные:

- число шагов поворота **n** модели на проекции, тогда угол поворота равен $360°/n$
- шаг между точками, составляющими ребро (ребро разбивается на точки и проецируется точечно)
- длина отражающей царапины

Предварительное определение параметров проекции

Из условия, до начала преобразований центр объекта находится в начале координат



- Определение координат вектора перемещения $\vec{m}$

$$\vec{m} = \begin{pmatrix} D - d \cdot cos\alpha \\ 0 \\ H - d \cdot sin\alpha \end{pmatrix}$$

- Определение границ проекции - радиусы **r** и **R**

Впишем весь объект в цилиндр (обозначен жёлтым цветом), таким образом установив максимально возможные границы объекта при вращении.

При обработке входных данных сохраним значения (*V - множество вершин модели)*:

$$z_{max}$$

$$z_{min}$$

$$xy_{max} = \sqrt{max(|x_i|)^2 + max(|y_i|)^2}, i \in [1, |V|]$$

Тогда с учетом перемещения всех точек объекта на вектор $\vec{m}$, для проекции цилиндра на плоскость Oxz получим координаты точек E, F:

$$E \begin{pmatrix} x_{\vec{m}} - xy_{max} \\ 0 \\ z_{\vec{m}} + z_{max} \end{pmatrix}$$

$$F\begin{pmatrix} x_{\vec{m}} + xy_{max} \\ 0 \\ z_{\vec{m}} + z_{min} \end{pmatrix}$$

Координаты наблюдателя:

$$N\begin{pmatrix} D \\ 0 \\ H \end{pmatrix}$$

Из уравнений прямых, содержащих отрезки NE и NF, найдем значения **r** и **R**:

$$r = \frac{H \cdot (x_E - D)}{H - z_E} - D = \frac{H \cdot (x_{\vec{m}} - xy_{max} - D)}{H - z_{\vec{m}} - z_{max}} = \frac{H \cdot (-xy_{max} - d \cdot cos\alpha)}{d \cdot sin\alpha - z_{max}}$$

$$R = \frac{H \cdot (x_F - D)}{H - z_F} - D = \frac{H \cdot (x_{\vec{m}} + xy_{max} - D)}{H - z_{\vec{m}} - z_{min}} = \frac{H \cdot (xy_{max} - d \cdot cos\alpha)}{d \cdot sin\alpha - z_{min}}$$

Оценка видимой части модели, фильтрация ребер (точек, составляющих ребра) для последующего проецирования

Для проецирования все ребра модели разбиваются на точки с некоторым шагом и проверяется видимость каждой точки. Для последующих вычислений понадобятся координаты вектора $\vec{d}$

$$\vec{d} = \begin{pmatrix} d \cdot cos\alpha \\ 0 \\ d \cdot sin\alpha \end{pmatrix}$$

Для дальнейшей работы производится **перемещение всей модели на вектор** $\vec{m}$ (из начала координат к месту мнимого отображения).

Так как для каждого полигона трехмерной модели известны внешние нормали, видимые полигоны можно отфильтровать.

Для полностью выпуклых моделей видимые полигоны можно отфильтровать, сравнивая угол ϕ между внешней нормалью к полигону и вектором $\vec{d}$: он не должен превышать $90°$, а значит достаточное условие для отбора полигонов:

$$cos\phi = \frac{\vec{n}_i \vec{d}}{|\vec{n}_i| \cdot |\vec{d}|} \in [0, \pi/2)$$

Где $\vec{n_i}$- нормаль к *i*-му полигону. Ребра, ограничивающие отобранные полигоны, точно видимые, их можно проецировать на абразивный диск. Время работы алгоритма $O(1)$для одного полигона , тогда для проверки всех полигонов $O(|P|)$, где P - множество полигонов. Проверка производится однократно для всей модели.

Для моделей, имеющих вогнутые поверхности вышеописанного условия будет недостаточно: ***полигоны, повернутые к наблюдателю, могут перекрывать друг друга***. Решение данной проблемы - трассировка каждой точки ребра (отобранного по алгоритму выше) к наблюдателю, и проверка пересечения с плоскостями полигонов на пути трассировки.

Рассмотрим алгоритм на примере точки V_k и произвольного полигона.

Зная координаты нормали $\vec{n}$ к полигону и координаты точки V_i, лежащей в его плоскости, строим уравнение плоскости:

$$x_n \cdot x + y_n \cdot y + z_n \cdot z + (x_n \cdot x_{V_i} + y_n \cdot y_{V_i} + z_n \cdot z_{V_i}) = 0$$

Перегруппируем с переименованием коэффициектов:

$$A \cdot x + B \cdot y + C \cdot z + D = 0$$

Имеем трассирующий луч от наблюдателя N к точке V_k, лежащий на прямой с направляющим вектором:

$$\vec{V_kN} = \vec{r} = \begin{pmatrix} D - x_{V_k} \\ -y_{V_k} \\ H - z_{V_k} \end{pmatrix}$$

Тогда каноническое уравнение прямой V_kN:

$$\frac{x - x_N}{x_{\vec{r}}} = \frac{y - y_N}{y_{\vec{r}}} = \frac{z - z_N}{z_{\vec{r}}}$$

Подставим значения:

$$\frac{x - D}{D - x_{V_k}} = \frac{y}{y_{V_k}} = \frac{z - H}{H - z_{V_k}} = t$$

Перепишем в параметрическом виде:

$$\begin{cases} x = D + t \cdot (D - x_{V_k}) \\ y = t \cdot (y_{V_k}) \\ z = H + t \cdot (H - z_{V_k}) \end{cases}$$

Подставим в уравнение плоскости, задающей полигон, и решим относительно t:

$$A \cdot (D + t \cdot (D - x_{V_k})) + B \cdot (t \cdot (y_{V_k})) + C \cdot (H + t \cdot (H - z_{V_k})) + D = 0$$

$$t = -\frac{A \cdot D + C \cdot H + D}{A \cdot (D - x_{V_k}) + B \cdot y_{V_k} + C \cdot (H - z_{V_k})}$$

Если знаменатель равен 0, то луч от наблюдателя параллелен плоскости полигона, что исключается достаточным условием видимости полигонов для выпуклых моделей.

Вычислив параметр ***t*** и подставив в систему параметрических уравнений прямой, **получим точку Z пересечения прямой и плоскости**.

Так как данная точка должна лежать между наблюдателем и проверяемой точкой V_k, введем дополнительное условие: угол β между векторами $\vec{V_kZ}$ и $\vec{V_kN}$ не должен превышать $90°$, а значит:

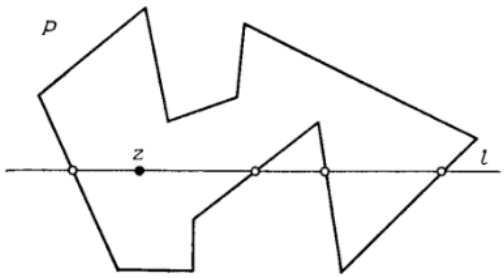
$$cos\beta = \frac{\vec{V_kN} \cdot \vec{V_kZ}}{|\vec{V_kN}| \cdot |\vec{V_kZ}|} \in [0, 1)$$

Далее необходимо проверить, **находится ли Z в части плоскости, ограниченной полигоном**.

Для проверки используется метод, описанный в книге *“Вычислительная геометрия: Введение” Препарта Ф., Шеймос М, раздел 2.2 с.57.*

Для проверки принадлежности точки ***Z*** некоторой замкнутой области предлагается провести из данной точки произвольный луч, и проверить количество пересечений луча с ломаной, ограничивающей область. Количество пересечений должно быть *нечетно*, в таком случае точка лежит внутри области. Для приведенного выше примера число пересечений для левого луча равно 1, для правого 3. Также, **необходимым условием** является **непересечение прямой, на которой лежит луч, с вершинами многоугольника**. Если условие не

выполняется, необходимо произвести малый поворот луча относительно точки z и произвести пересчёт.



Произведём аналитическое описание алгоритма: пусть секущая прямая l имеет направляющий вектор:

$$\vec{ZU} = \begin{pmatrix} x_U - x_Z \\ y_U - y_Z \\ z_U - z_Z \end{pmatrix} = \begin{pmatrix} x_{\vec{ZU}} \\ y_{\vec{ZU}} \\ z_{\vec{ZU}} \end{pmatrix}$$

где U - случайно выбранная точка в плоскости, координаты можно выбрать случайным образом, используя уравнение плоскости полигона. Тогда каноническое уравнение прямой:

$$\frac{x - x_Z}{x_{\vec{ZU}}} = \frac{y - y_Z}{y_{\vec{ZU}}} = \frac{z - z_Z}{z_{\vec{ZU}}}$$

Подставив координаты вершин, ограничивающих полигон, в каноническое уравнение, проверим их принадлежность секущей. Если хотя бы одна вершина лежит на секущей, меняем точку U и производим пересечет. Далее, когда ни одна вершина не лежит в на секущей, проверяем пересечение A, B секущей и отрезков-ребер, ограничивающих полигон. Для отрезка AB между вершинами имеем отрезок, лежащий на прямой с направляющим вектором $\vec{AB}$.
Каноническое уравнение прямой:

$$\frac{x - x_A}{x_{\vec{AB}}} = \frac{y - y_A}{y_{\vec{AB}}} = \frac{z - z_A}{z_{\vec{AB}}}$$

Условие **коллинеарности** направляющих векторов прямых:

$$\frac{x_{\vec{ZU}}}{x_{\vec{AB}}} = \frac{y_{\vec{ZU}}}{y_{\vec{AB}}} = \frac{z_{\vec{ZU}}}{z_{\vec{AB}}}$$

Если выполняется, прямые либо параллельны, либо совпадают. Случай, когда прямые совпадают, исключен (секущая не пересекает ни одну из вершин). Значит пересечения нет.

Если условие коллинеарности не выполняется, то прямые имеют точку пересечения. Для поиска точки пересечения перепишем уравнения прямых в каноническом виде:

$$\begin{cases} x = x_Z + t \cdot x_{\vec{ZU}} \\ y = y_Z + t \cdot y_{\vec{ZU}} \\ z = z_Z + t \cdot z_{\vec{ZU}} \end{cases}$$

$$\begin{cases} x = x_A + k \cdot x_{\vec{AB}} \\ y = y_A + k \cdot y_{\vec{AB}} \\ z = z_A + k \cdot z_{\vec{AB}} \end{cases}$$

Координаты точки пересечения должны удовлетворять обеим системам, приравняем соответствующие координаты и найдем один из коэффициентов *t* или *k*:

$$\begin{cases} x_Z + t \cdot x_{\vec{ZU}} = x_A + k \cdot x_{\vec{AB}} \\ y_Z + t \cdot y_{\vec{ZU}} = y_A + k \cdot y_{\vec{AB}} \\ z_Z + t \cdot z_{\vec{ZU}} = z_A + k \cdot z_{\vec{AB}} \end{cases}$$

Имеем систему трех уравнений с двумя неизвестными. Решение относительно ***t***, из первого и второго уравнения системы:

$$t = \frac{y_{\vec{AB}} \cdot (x_A - x_Z) + x_{\vec{AB}} \cdot (y_Z - y_A)}{x_{\vec{ZU}} \cdot y_{\vec{AB}} - y_{\vec{ZU}} \cdot x_{\vec{AB}}}$$

Подставив значение ***t*** в систему параметрических уравнений секущей, получим координаты точки пересечения K. Условие нахождения точки внутри отрезка:

$$AB = AK + KB$$

Значения длин отрезков вычисляются из координат точек A, B, K.

Далее необходимо отфильтровать точки, лежащие относительно Z по одну сторону на секущей l. Для этого отберем только удовлетворяющие $\vec{ZK} \uparrow\uparrow \vec{ZU}$, что эквивалентно выполнению хотя бы одного из условий:

$$sign(x_{\vec{ZK}}) = sign(x_{\vec{ZU}}) \neq 0$$

$$sign(y_{\vec{ZK}}) = sign(y_{\vec{ZU}}) \neq 0$$

$$sign(z_{\vec{ZK}}) = sign(z_{\vec{ZU}}) \neq 0$$, где

$$sign(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{if } x = 0 \\ -1, & \text{if } x < 0 \end{cases}$$

Если точка Z лежит внутри полигона, значит имеем пересечение трассирующего луча из точки V_k и полигона на пути к наблюдателю, а значит точка V_k перекрывается полигоном и невидима для наблюдателя. Тогда игнорируем её.

Таким образом, время работы алгоритма поиска принадлежности точки полигону $O(n)$, *n* - количество сторон многоугольника, представляющего полигон. Время работы алгоритма для одной проецируемой точки $O(n \cdot |P|) = O(|P|)$. Опустить *n* возможно так как $n \in \{3, 4\}$ для типичного случая полигона.

Итоговая асимптотика:

- для выпуклой модели: $O(|P| + k)$, где k - число проецируемых точек видимой части модели
- для невыпуклой модели: $O(k \cdot |P|)$, где *k* - число проецируемых точек всей модели

Проецирование видимых точек, отобранных алгоритмом фильтрации. Осуществление поворота модели и проекции для заполнения всей поверхности пластины

Пусть абразивная пластина находится в плоскости ***0xy***, с центром в начале координат. Если точка V_k отобрана для проецирования на абразивную пластину, находим точку пересечения W_k прямой V_kN с плоскостью ***0xy***. Из канонического уравнения прямой получим координаты W_k:

$$\begin{cases} x_{W_k} = \frac{H \cdot (D - x_{V_k})}{z_{V_k} - H} \\ y_{W_k} = \frac{H \cdot y_{V_k}}{z_{V_k} - H} \\ z_{W_k} = 0 \end{cases}$$

Проецирование можно производить сразу при отборе соответствующих точек. Далее, после проецирования всех точек одного состояния необходимо повернуть модель относительно оси **O'**, а проекции на абразивном диске относительно оси **0z**.

Необходим полный поворот модели за **n** заданных шагов, тогда угол единичного поворота $\phi = 360°/n$.

Чтобы уменьшить количество вычислений, будем хранить угол, на который модель *уже* повернута. Для начального состояния $\theta = 0°$. После одного шага $\theta = 360°/n$, после двух шагов $\theta = 2 \cdot (360°/n)$ и т.д.

Имеем матрицу поворота относительно оси ***z*** на угол γ***:***

$$M_z(\gamma) = \begin{pmatrix} cos\gamma & -sin\gamma & 0 \\ sin\gamma & cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Тогда новые координаты точки будут:

$$\begin{pmatrix} x' & y' & z' \end{pmatrix} = \begin{pmatrix} x & y & z \end{pmatrix} \cdot M_z(\gamma)$$

$$\begin{pmatrix} x' & y' & z' \end{pmatrix} = \begin{pmatrix} x \cdot cos\gamma + y \cdot sin\gamma & -x \cdot sin\gamma + y \cdot cos\gamma & z \end{pmatrix}$$

Чтобы не производить пересчет всех точек абразивного диска каждый раз, можно *сразу при вычислении координат* проекции W_k точки V_k *поворачивать точку* W_k *относительно оси* ***0z*** на угол $(-\theta)$.

Для упрощения вычислений, исходные координаты модели, расположенной в начале координат, сохраняются. Для вычисления k-й проекции модель (координаты вершин) модель в начале координат поворачивается на соответствующий номеру проекции угол θ относительно оси **0z**, а после переносится в точку проецирования на вектор $\vec{m}$.

Таким образом, производим покадровое проецирование всей модели, сделав её полный оборот за **n** шагов.

Расчет координат отражающих царапин, соответствующих каждой спроецированной точке

Пусть для угла $(-\theta)$ имеем уже повернутую проекцию W'_k некоторой точки модели.

Также, построим проекцию положения наблюдателя в плоскости пластины и произведем поворот проекции на угол $(-\theta)$, обозначим данную точку как N'_{0xy} .

Дальнейшее решение можно рассматривать в двумерном пространстве, так как для всех точек координата z. Найдем координаты вектора $W'_k\vec{N}'_{0xy} = \vec{s}$, как разность координат концов. Далее находим вектор $\vec{q}$, перпендикулярный $\vec{s}$, и нормируем его. Условие перпендикулярности векторов:

$$x_{\vec{q}} \cdot x_{\vec{s}} \cdot + y_{\vec{q}} \cdot y_{\vec{s}} \cdot = 0$$

Зададим произвольную точку $x_{\vec{q}} = const$, тогда:

$$y_{\vec{q}} = -\frac{x_{\vec{q}} \cdot x_{\vec{s}}}{y_{\vec{s}}}$$

Нормируем:

$$\vec{q}_0 = \begin{pmatrix} \frac{x_{\vec{q}}}{|\vec{q}|} \\ \\ \frac{y_{\vec{q}}}{|\vec{q}|} \end{pmatrix}$$

Найдем координаты точек K, L начала и конца царапины. Для этого переместим точку W'_k на $\pm\vec{q}_0 \cdot u$, где u - половина длины царапины:

$$K \begin{pmatrix} x_{W'_k} + \frac{x_{\vec{q}}}{|\vec{q}|} \cdot u \\ \\ y_{W'_k} + \frac{y_{\vec{q}}}{|\vec{q}|} \cdot u \end{pmatrix}$$

$$L \begin{pmatrix} x_{W'_k} - \frac{x_{\vec{q}}}{|\vec{q}|} \cdot u \\ \\ y_{W'_k} - \frac{y_{\vec{q}}}{|\vec{q}|} \cdot u \end{pmatrix}$$

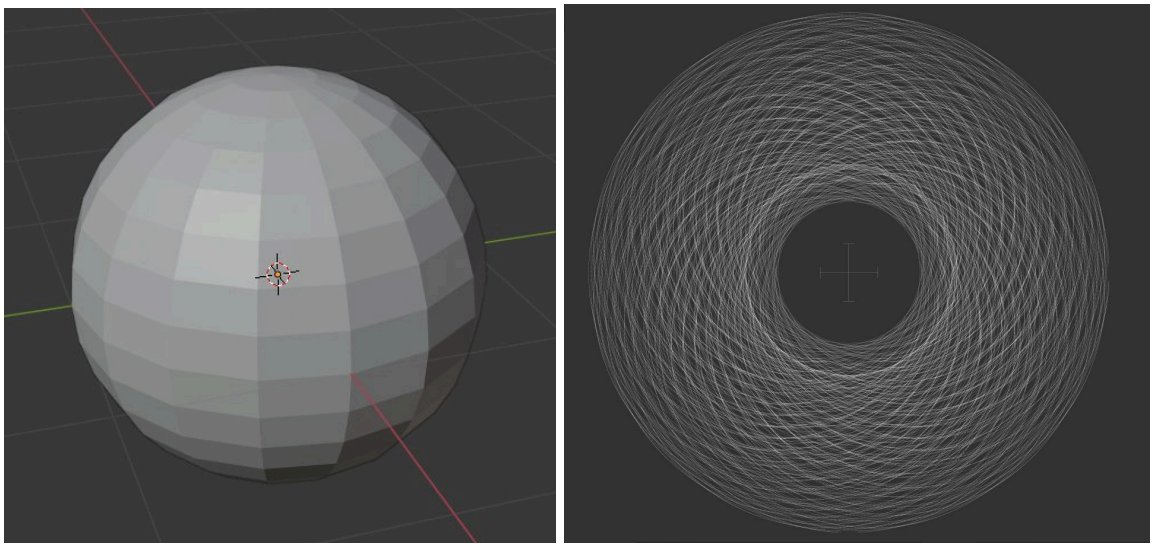
Повторяем для каждой точки, получаем рисунок царапин на плоскости пластины

III.Реализация алгоритма

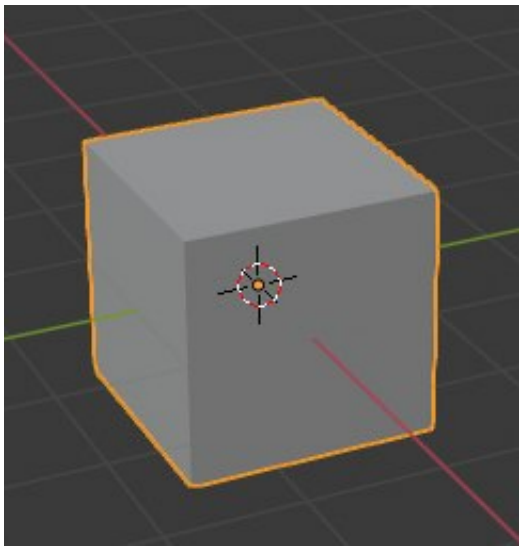
Адаптированный алгоритм реализован на языке C++ с использованием стандартной библиотеки, выпущена первая версия с интерфейсом командной строки (CLI), преобразующая *.obj-файл трехмерной модели в *.svg-файл с рисунком царапин: https://github.com/shchuko/ScratchedHologramFrom3D

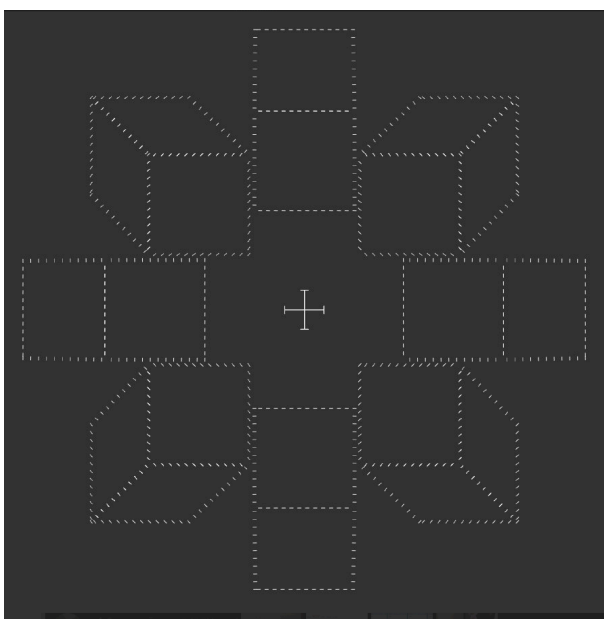
```
shchuko@shchuko-pc ~/P/c/S/c/bin (master)> ./ScratchedHologramFrom3DApp
Scratched Hologram Builder v0.0.1
Supported input formats: *.obj (Z Up, -X Forward). Supported output formats: *.svg.
Note: [length units] are related to input 3D-object units.
Options:
    --help                          - Display this help, disables all options except this one
-i, --in           [arg] {required} - Set input file name
-o, --out          [arg]            - Set output file name. Default: scratch_out.svg
-f, --write-force                   - Overwrite output file if exists
-d, --viewer-dist [arg] {required} - Distance between object and viewer in [length units]
-h, --plane-dist  [arg] {required} - Distance between object and projection plane [length units]
-a, --angle       [arg] {required} - Angle between viewer and object center (default: degrees)
    --rad                           - Read angle as radians
-s, --spacing     [arg]            - Spacing width in [length units]. Default 0.0
-n, --steps       [arg] {required} - Calculation steps number
-c, --convex                        - Perform calculations for convex 3d-model (faster)
    --width       [arg]            - Set canvas width in px. Default: 1024px
    --height      [arg]            - Set canvas height in px. Default: 1024px
    --line-width  [arg]            - Set line width in px. Default: 1px
    --no-confirm                    - Skip operations confirmation
-x, --scratch-len [arg] {required} - Scratch length [length units]
-k, --step-len    [arg] {required} - Step between two scratches on the edge projection [length units]
```

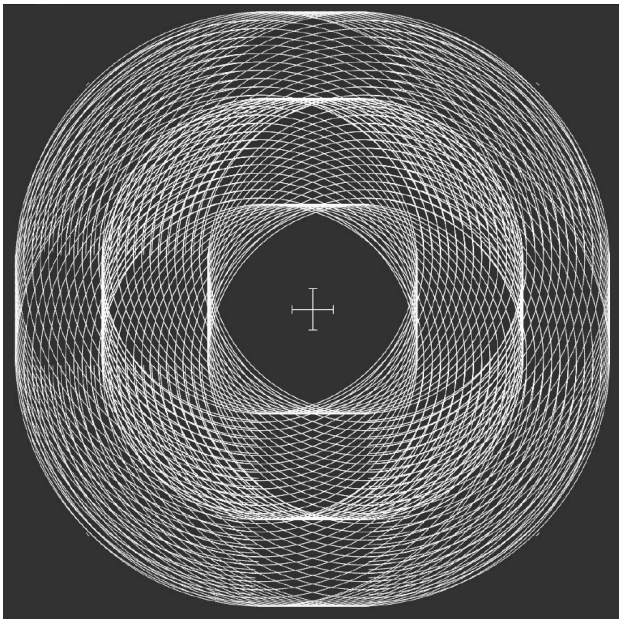
Примеры работы алгоритма:



Модель сферы, большое число итераций алгоритма





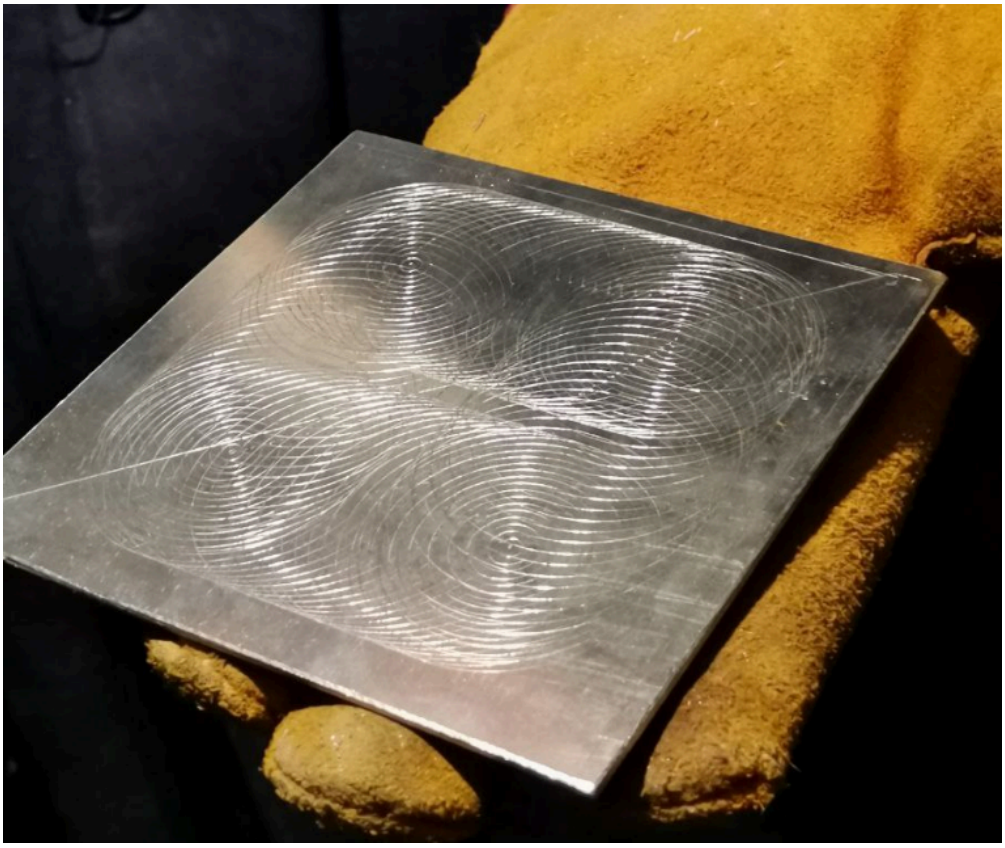


Модель куба, малое и большое число итераций алгоритма

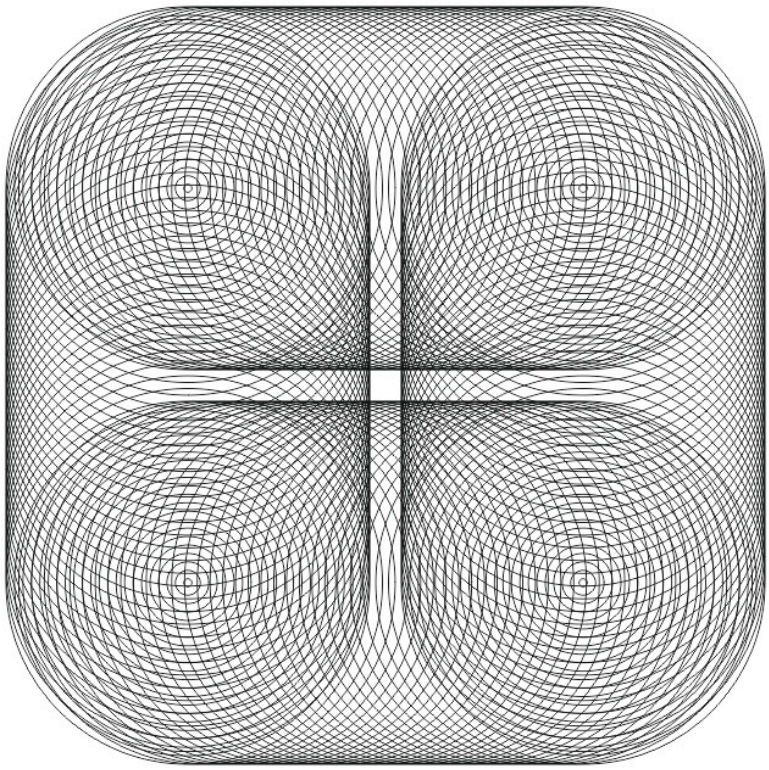
На текущем этапе разработка все еще ведется, так как требуются исправления и улучшения. Значительным недостатком текущей версии является дискретность проекций точек, представленных в виде отражающих отрезков царапин. Более оптимально использовать интерполирующие кривые, проходящие через точки проекций, что упростит дальнейшую работу с полученным рисунком царапин.

IV.Тестовые образцы гравировки голографической пластинки

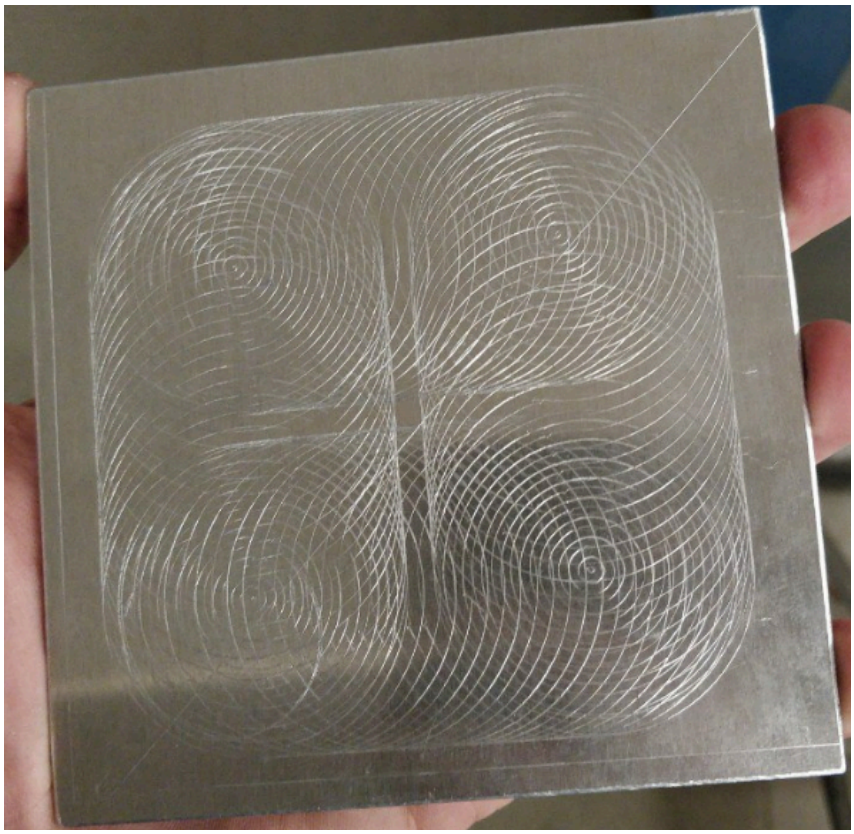
Используя аппаратные возможности ЧПУ-станка Roland-MDX40 лаборатории ФабЛаб Университета ИТМО, была получена первая голографическая пластинка с моделью куба, изготовленная по скретч-технологии из листа алюминия:



Входной рисунок царапин, использованный в качестве макета:



Царапины, перенесенные на пластину:



Для резки изготовлена специализированная насадка для ЧПУ-станка, заменяющая стандартный шпиндель. Первая пробная версия приведена на фото ниже:

Насадка обеспечивает постоянный прижим твердосплавной чертилки по металлу к материалу, позволяя игнорировать незначительные неровности. Из недостатков текущей версии можно отметить существенный люфт чертилки в горизонтальной плоскости, и при этом плохое скольжение чертилки в направляющих отверстиях. Также использование резинового натяжителя является временным решением. В дальнейшем насадка будет доработана, деревянные направляющие будут заменены на вертикальный подшипник, вместо натяжительной резинки будут использованы металлические пружины или различные утяжелители, а также добавлена возможность регулировки усилия прижима и смены режущих наконечников, что позволит увеличить число поддерживаемых для резки царапин материалов.

Планы дальнейшей разработки проекта

1. Доработать ПО, добавить интерполяцию отрезков царапин кривыми, добавить графический пользовательский интерфейс (GUI);
2. Доработать насадку для ЧПУ, повысить точность резки путем минимизации горизонтальных люфтов чертилки, добавить возможность резки по различным видам материалов (прим. винил);
3. Провести испытания резки по различным видам материалов, выбрать наиболее оптимальный (качество голографического эффекта, сложность изготовления, цена материала);
4. Изготовить голографические пластинки с более сложными трехмерными моделями.