

「数学的に考える」

...

GLSL School 2024 プラスワン講義 / phi16
2025/01/11

質問・指摘等お気軽に！

自己紹介

- phi16 ('ふあい'で呼んで頂いて大丈夫です)
 - 主に VRChat か Twitter に居ます
- 一応グラフィックスエンジニアが本職
 - 仕事の話はインターネットではしませんが...
- 大学時代の専門は数理論理学とか
 - 数学科ではないんですが (情報工学科)
 - プログラミング言語自体の理論とか
- やったことのある数学の分野 (わかる人向け)
 - 自主: 集合 (素朴)・位相 (基本)・群・測度・型理論・圏
 - 講義: 線形代数・微積・複素解析 (基本)・微分幾何・計算論

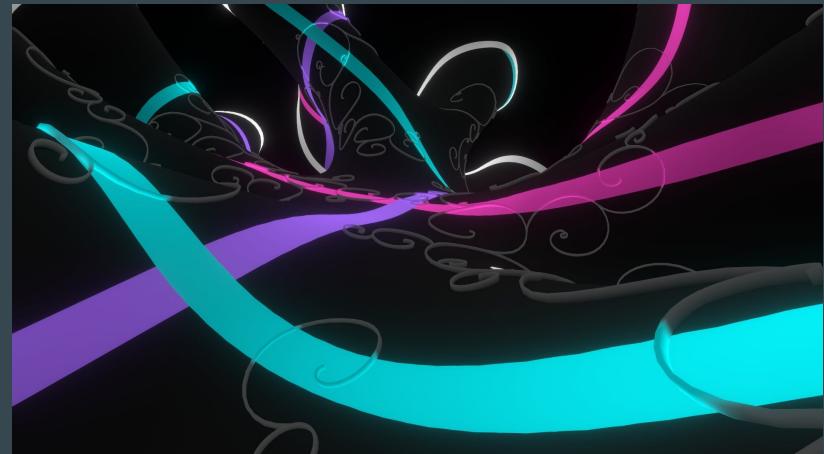


自己紹介

- 数学っぽいものを作りがち



結び目を一致させるパズル(?)



双曲平面を歩くワールド

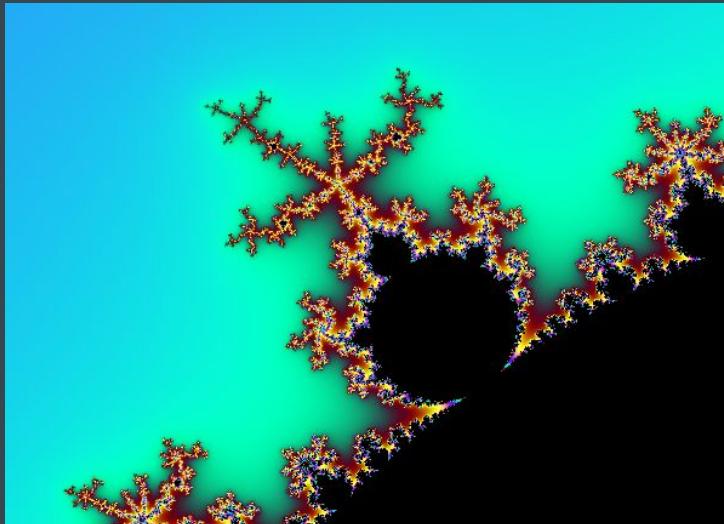
自己紹介

- 数学を使いがち

今回のテーマ

- 色々やっていると数学にぶち当たることが多い
 - そこで「わからない」「読めない」から諦めてしまいがち
 - 欲しかったテクニックがそこにあるかもしれないのに！
- 数学と仲良くなろう！
 - Wikipediaを物怖じせず面と向かって読めるように
 - 知らない単語や数式が出てきても冷静に読もうとできるように
 - 「わからないものはわからない」
 - すぐできるようにはならない。
 - わかることとわからないことを区別できるようになって初めて理解を進めることができる

「数学」



The *tensor product* of two vector spaces V and W is a vector space denoted as $V \otimes W$, together with a bilinear map $\otimes : (v, w) \mapsto v \otimes w$ from $V \times W$ to $V \otimes W$, such that, for every bilinear map $h : V \times W \rightarrow Z$, there is a *unique* linear map $\tilde{h} : V \otimes W \rightarrow Z$, such that $h = \tilde{h} \circ \otimes$ (that is, $h(v, w) = \tilde{h}(v \otimes w)$ for every $v \in V$ and $w \in W$).

https://en.wikipedia.org/wiki/Tensor_product



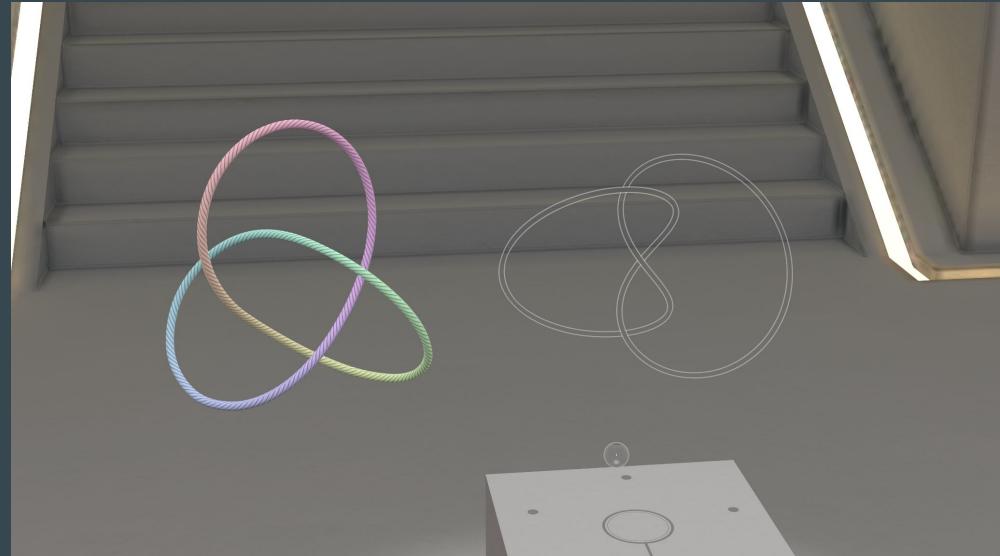
作ったものの話

もう少し自己紹介を兼ねて

- 作ったVRChatのワールドの紹介を軽くしてみます
 - Knot Puzzle
 - Bounded H^2
 - Inter-action on the Math
- メモ: VRChat の説明をする

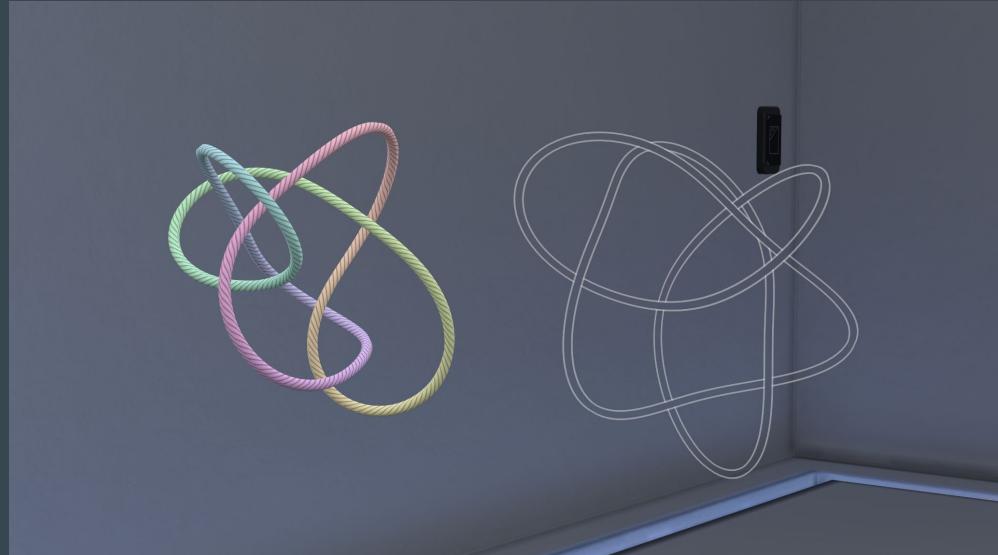
Knot Puzzle

- 結び目理論 (Knot Theory) の典型的な問題:
「2つの結び目は、変形すると一致するか？」



Knot Puzzle

- 結び目理論 (Knot Theory) の典型的な問題:
「2つの結び目は、変形すると一致するか？」



Knot Puzzle

- 結び目理論 (Knot Theory) の典型的な問題:
「2つの結び目は、変形すると一致するか？」
- VRで空中の紐を掴んで動かして一致させるパズル (?)
 - 難しさが体験できる (?)
- 紐の挙動はGPUで処理 (概ねGPGPU)
 - 近い位置にある紐からは適切に距離を取るように
 - 紐が貫通したら (スカラー三重積で計算できる) 失敗状態に遷移
 - 正誤判定は「『最も近い正解点からの距離』の最大値が0.0075未満」
 - Parallel Reduction

Bounded \mathbb{H}^2

- 変な空間 (双曲平面, \mathbb{H}^2) を歩いてみたい
 - 5直角で一周したりする (球面上の逆パターン)
 - わかりやすい言い方: サニーレタスの端っこらへん





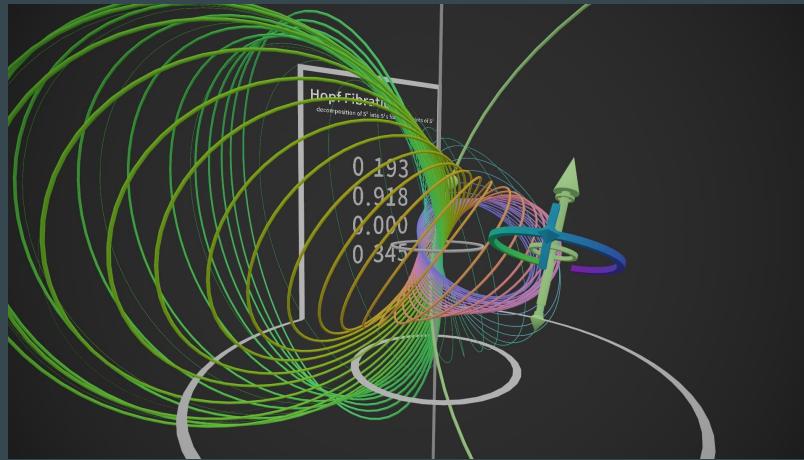
Bounded \mathbb{H}^2

- 変な空間 (双曲平面, \mathbb{H}^2) を歩いてみたい
 - 5直角で一周したりする (球面上の逆パターン)
 - わかりやすい言い方: サニーレタスの端っこらへん
- Gyrovector space
 - 加法が非結合的 ($(a + b) + c \neq a + (b + c)$ となることがある)
 - ある程度の等式が成り立つだけマジ
- 歩行はCPU、描画はGPUで処理
 - 歩くとメッシュがぐにぐに変形する
 - 空間形状はテクスチャに保存されている



Inter-action on the Math

- 可視化できそうな数学トピックをいくつか置いたワールド
- 概ね GPU 処理



何がしたかったのか

- 面白いものが作りたい！
- 数学を調べてると面白いものがいっぱい見つかる
- そういうものたちを見やすい形で共有したい
 - そのために数学やGPUを活用する

何がしたかったのか

- 面白いものが作りたい！
- □□を調べてると面白いものがいっぱい見つかる
- そういうものたちを見やすい形で共有したい
 - そのために数学やGPUを活用する
- やりたいことをやるために
 - 正確な計算・求めている挙動・高速な処理・大量の描画

→ 目的に依らない！

数学の捉え方

数学は言語

- 言語はモノの考え方を誘導する
 - 言語(日本語などに限らず、JavaScriptとか「技術用語」もそう)によって私達の「考えやすさ」「伝えやすさ」は影響を受ける
 - 数学という言語によって考えやすくなる領域がある
 - この副作用が「欲しいもの」
 - 知らない言語で書かれてることがわからないのは**当然**
 - 数学をやらなければ数学はできない
- モノゴトを記述する方法は一つではない
 - 日本語でもJavaScriptで書いても良い
 - 数学でも良い...けど?

数学の良いところ

- 定義は展開できる
 - 言葉の意味がそれ違うと思ったら、それを冷静に確認できる
 - 厳格なコミュニケーションに最適
- 証明は検証できる
 - 「できること」が定められている
 - →「間違ったこと」をしていないことを確認できる
- 想像力に囚われない
 - 自分が知らないことでも、数式を弄することで結論を確認できる
 - 想像に引っ張られて間違えたりしない
 - 正確には、間違っても教えてくれる

「わかる」とは？(諸説)

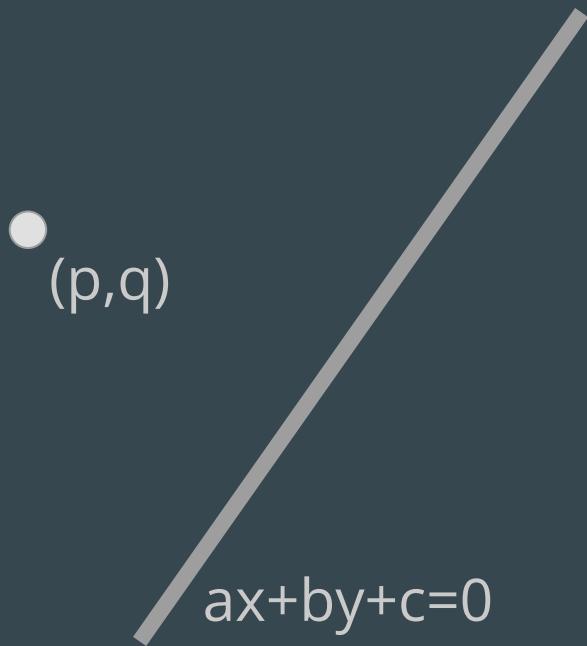
- 「わかる」=「計算できること」「証明できること」
 - 「もしもその分野について、特に困らずに正しい推論が行えるのであれば、あなたはわかっている。」
- 「でももっと良い計算方法があるかもしれない」
 - そんなのは無数にある
 - 全ての数学がわかるなんてことは無い
 - 無いです。
 - こういう考え方があるらしい、と知ったときに学びに向かえば良い
 - やりすぎるとWikipediaが読み終わらないのでゆっくりと…
 - 「わかっている」ことに違いはないのです。

数学の言葉で表す

例: 点と線の距離



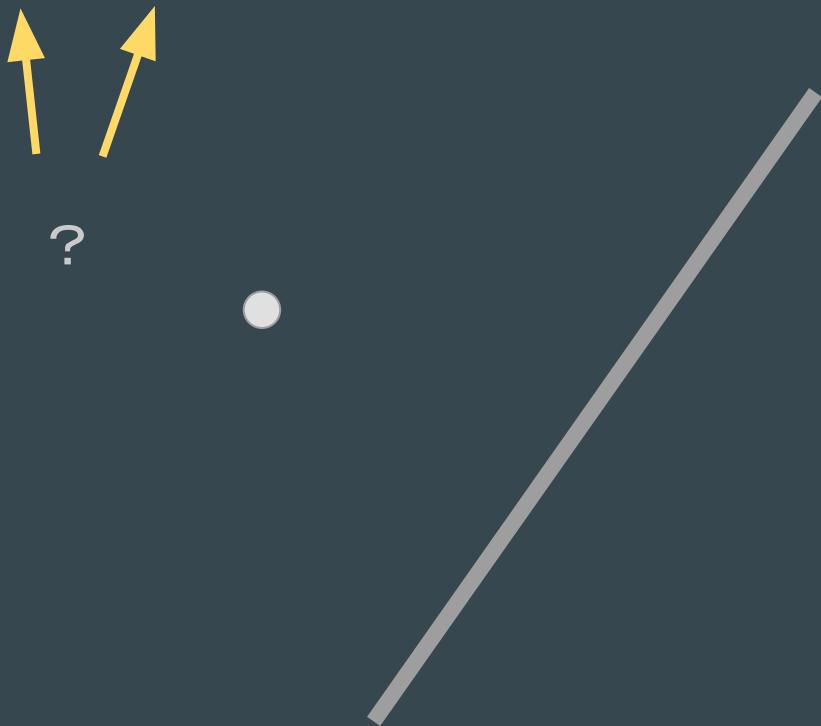
例: 点と線の距離



$$d = \frac{|ap + bq + c|}{\sqrt{a^2 + b^2}}$$

↑見覚えがある？

例: 点と線の距離



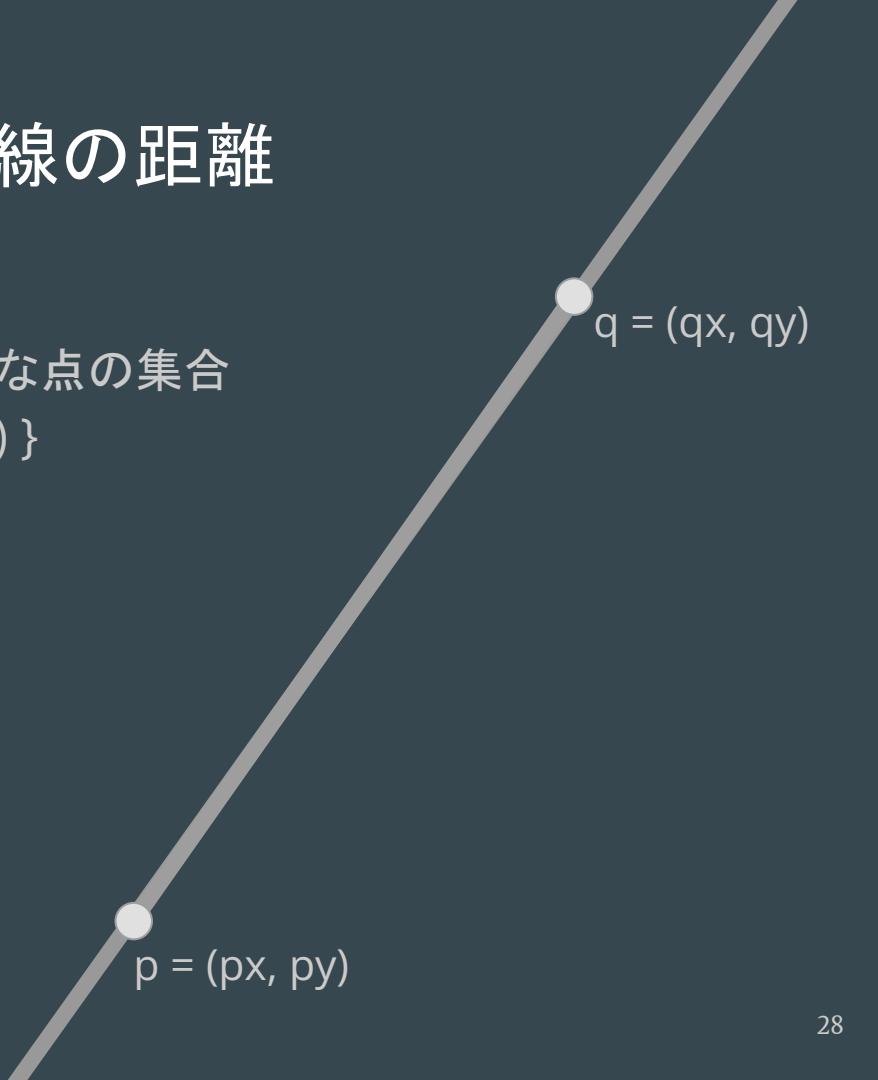
例: 2次元平面 \mathbb{R}^2 における点と線の距離

- $\mathbb{R}^2 = \{ (x,y) \mid x, y \in \mathbb{R} \}$
 - x と y を実数として、「 (x,y) 」として表現されるものの集まり
 - 実数というのは... float みたいなものです
 - これを点と呼ぶ(ことにする)
 - $(0,0)$ は点である
 - $(1.2, 3.4)$ は点である
 - 逆に言えば、点は (p,q) という形式で表せる
 - ここで p, q は実数
- 線とは?
 - 点の集まりで、まっすぐなもの
 - まっすぐとは?

← あなたが決めるのです！

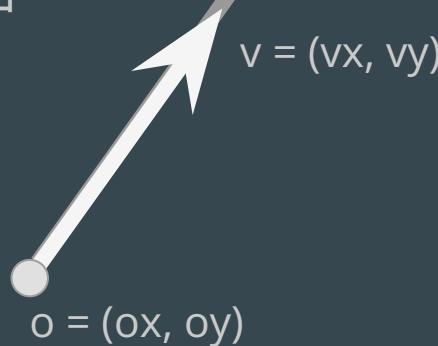
例: 2次元平面 \mathbb{R}^2 における点と線の距離

- 線とは...
 - 点 $p, q (p \neq q)$ の “あいだ” にあるような点の集合
 - $\{ x \in \mathbb{R}^2 \mid \exists \alpha. x = p + \alpha \times (q-p) \}$
 - $0 \leqq \alpha \leqq 1$ なら線分
 - $\alpha \in \mathbb{R}$ なら直線



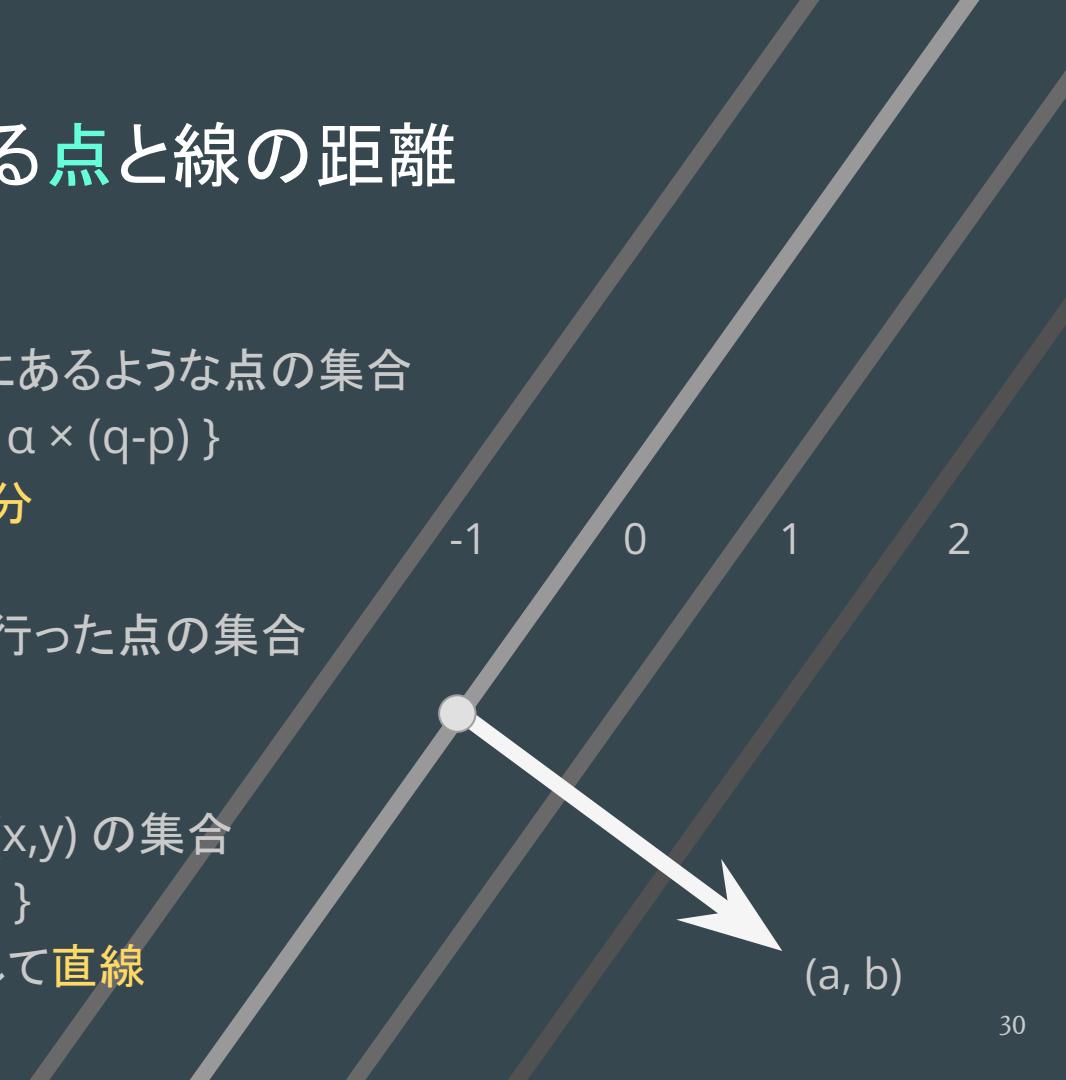
例: 2次元平面 \mathbb{R}^2 における点と線の距離

- 線とは...
 - 点 $p, q (p \neq q)$ の “あいだ” にあるような点の集合
 - $\{ x \in \mathbb{R}^2 \mid \exists \alpha. x = p + \alpha \times (q-p) \}$
 - $0 \leqq \alpha \leqq 1$ なら線分
 - $\alpha \in \mathbb{R}$ なら直線
 - 点 o から v の方向に進んで行った点の集合
 - $\{ o + t \times v \mid t \in \mathbb{R} \}$
 - $t \geqq 0$ なら半直線



例: 2次元平面 \mathbb{R}^2 における点と線の距離

- 線とは...
 - 点 $p, q (p \neq q)$ の “あいだ” にあるような点の集合
 - $\{ x \in \mathbb{R}^2 \mid \exists a. x = p + a \times (q-p) \}$
 - $0 \leqq a \leqq 1$ なら線分
 - $a \in \mathbb{R}$ なら直線
 - 点 o から v の方向に進んで行った点の集合
 - $\{ o + t \times v \mid t \in \mathbb{R} \}$
 - $t \geqq 0$ なら半直線
 - $ax + by + c = 0$ で表せる点 (x,y) の集合
 - $\{ (x,y) \mid ax + by + c = 0 \}$
 - 平面を1次元分潰して直線

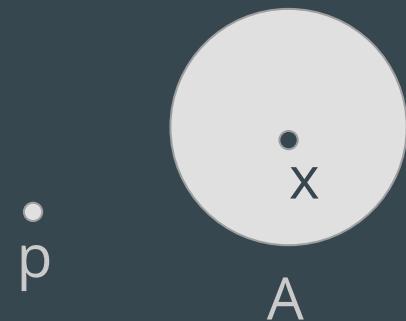


教訓

- 自然言語で同じ言葉だとしても数学上の表現はいろいろある
 - 観点の数だけある
 - (今まででは「点の集合」だったけど...)
 - 「1点」が直線を表すこともある (射影空間)
 - 「ベクトル対」が直線を表すこともある (Plücker coordinates)
 - しかし同じものであることもわかる！
 - 式変形によってお互いを変換できる
 - お互いに変換できるならそれは同じものである！
 - 同型の考え方

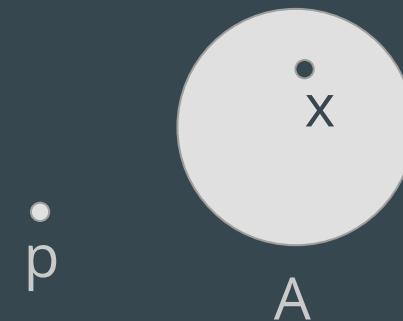
点と直線の距離

- 点 p と点の集合 A との距離は $\min\{ d(p, x) \mid x \in A \}$



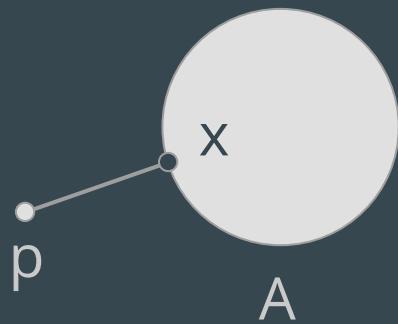
点と直線の距離

- 点 p と点の集合 A との距離は $\min\{ d(p, x) \mid x \in A \}$



点と直線の距離

- 点 p と点の集合 A との距離は $\min\{ d(p, x) \mid x \in A \}$



点と直線の距離

- 点 p と点の集合 A との距離は $\min\{ d(p, x) \mid x \in A \}$

- ということに決めました(昔の人が、そして私達も)
 - 何故なら、この定義は“うまくいく”から

- 「点と直線の距離」を計算してみよう！

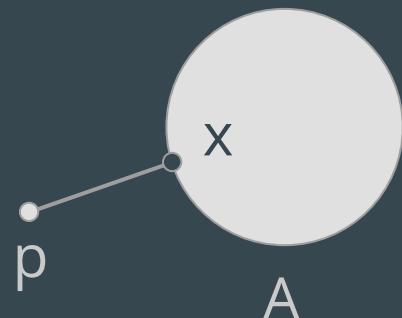
- $A = \{ (x,y) \mid ax + by + c = 0 \}$

- (距離) = $\min\{ d(p, x) \mid x \in A \}$

$$= \min\{ \sqrt{(p_x - x)^2 + (p_y - y)^2} \mid (x,y) \in \mathbb{R}^2, ax + by + c = 0 \}$$

$$= \sqrt{\min\{ (p_x - x)^2 + (p_y - y)^2 \mid (x,y) \in \mathbb{R}^2, ax + by + c = 0 \}}$$

$$= \dots$$



計算 (続)

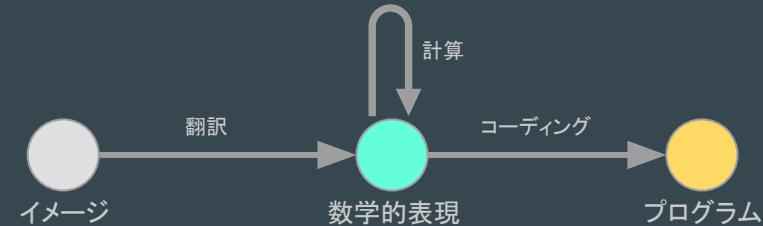
- $d^2 = \min\{ (p_x - x)^2 + (p_y - y)^2 \mid (x, y) \in \mathbb{R}^2, ax + by + c = 0 \}$
 - p_x, p_y, a, b, c は与えられたパラメータ
- この時点でもはや「点」も「直線」も関係ない
 - 「イメージ」は完全に数学の言葉に変換できた
- あとは計算するだけ
 - ?

計算 (続)

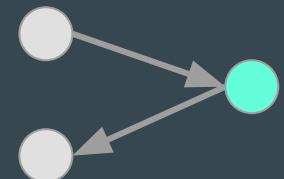
- $d^2 = \min\{ (p_x - x)^2 + (p_y - y)^2 \mid (x, y) \in \mathbb{R}^2, ax + by + c = 0 \}$
 - p_x, p_y, a, b, c は与えられたパラメータ
- この時点でもはや「点」も「直線」も関係ない
 - 「イメージ」は完全に数学の言葉に変換できた
- あとは計算するだけ
 - 拘束条件付き最適化問題 → ラグランジュの未定乗数法
 - もう ChatGPT とかに投げても良い
 - 一昔前は Wolfram | Alpha に投げてたんですが...

数学の言葉に翻訳する

- イメージを自然言語ではなく数学の言葉で書く
 - プログラミングの言葉でも良いけど数学の言葉の方が「強い」
 - 表現力が高い・検証可能
 - → そこから計算可能な形式へ



- 面白い現象: 数学の言葉から逆にイメージが出てくることがある
 - シンプルな拘束条件の問題だな～
→ これは点と直線の距離として解釈できるかも、みたいな
 - シンプレックス法とか



例: 射影変換 (Projection Matrix とは何だったのか)

- 物体情報をカメラから見た視界に変換する
 - まっすぐな線はまっすぐな線に変換される
 - (または点になる)



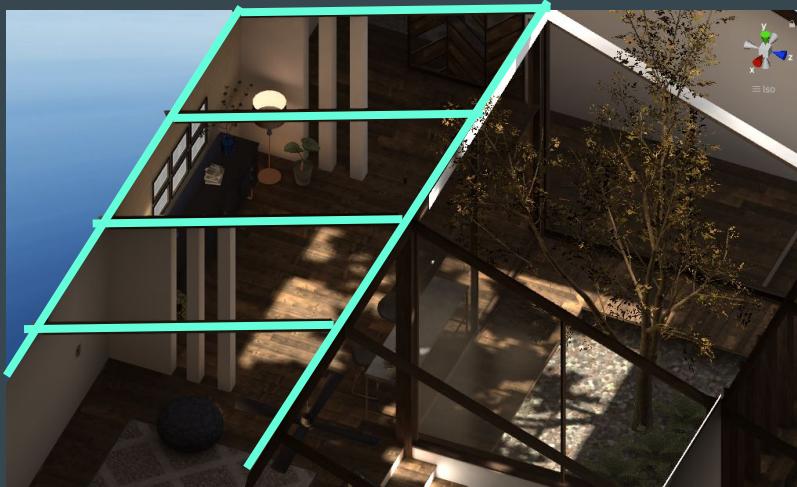
例: 射影変換

- 物体情報をカメラから見た視界に変換する
 - まっすぐな線はまっすぐな線に変換される
 - (または点になる)



例: 射影変換

- 物体情報をカメラから見た視界に変換する
 - まっすぐな線はまっすぐな線に変換される
 - (または点になる)



例: 射影変換

- 物体情報をカメラから見た視界に変換する
 - まっすぐな線はまっすぐな線に変換される
 - (または点になる)
- この時点でこの変換は射影変換であることがわかる
 - 同次座標系上の行列として表現できることがわかる
 - データとしてはただの数の列！
- もしも平行な線が平行な線に変換されるなら
 - 単なる線形変換であることがわかる

人生に役立つ知識

- まっすぐを扱っていたらどこかしらに**線形性**がある
 - 線形性の扱い方を学ぶなら → 線形代数学
- 思いも寄らないところにまっすぐがあつたりする
 - 力ク力クしてくるくらいなら区分線形
 - 非線形でも(なめらかなら)小さく見れば線形
 - フーリエ変換も線形変換
 - 光も線形(雑な言い方)

物事を数学で観る

もう少し表現寄りの話

画像を回す

- 画像って?
 - バイナリデータ列...
 - テンソル... (グレースケールなら行列...)
 - 実数列とサイズの情報...
 - 30度回転した画像を計算するには?
 - ピクセルの間の扱いは...?
 - サイズは...?
 - むむむ...

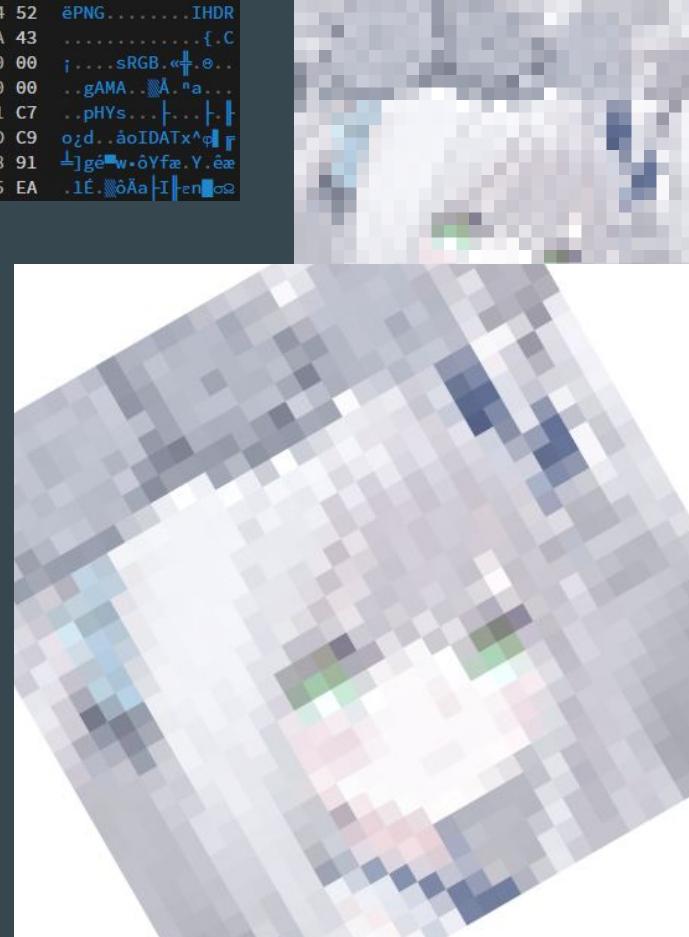


```
[[189 189 189 ... 216 217 218]
 [190 190 188 ... 216 217 218]
 [190 189 189 ... 216 215 218]
 ...
 [253 252 252 ... 184 182 182]
 [252 252 252 ... 183 183 183]
 [251 252 253 ... 183 182 182]]
```

画像を回す

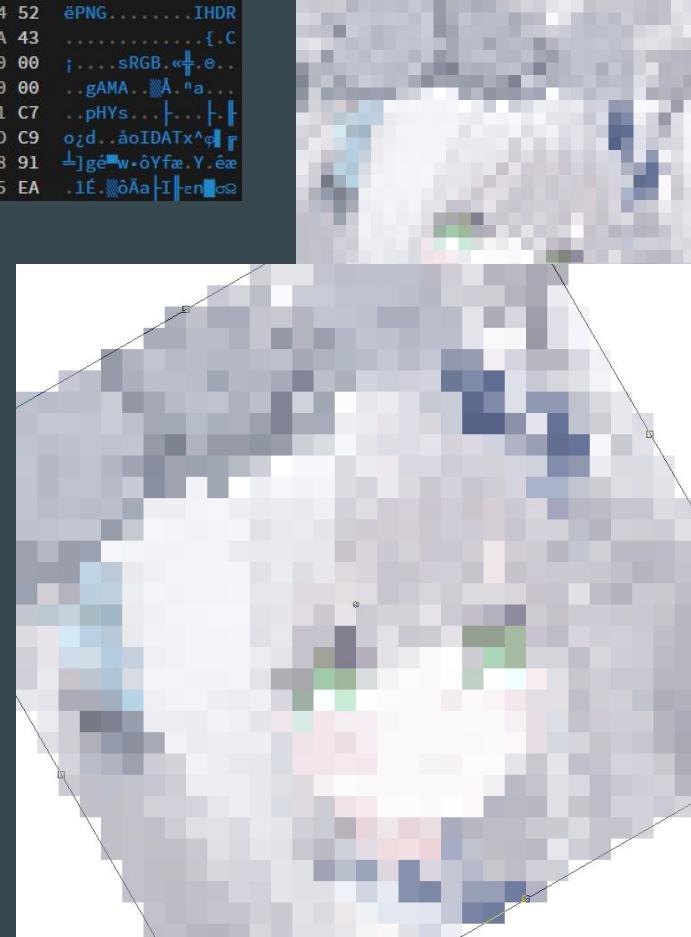
- 画像って?
 - バイナリデータ列...
 - テンソル... (グレースケールなら行列...)
 - 実数列とサイズの情報...
 - 30度回転した画像を計算するには?
 - ピクセルの間の扱いは...?
 - サイズは...?
 - むむむ...

00000000	89 50 4E 47 0D 0A 1A 0A 00 00 00 00 00 0D 49 48 44 52	�PNG.....IHDR
00000010	00 00 02 00 00 00 02 00 08 02 00 00 00 7B 1A 43sRGB..e...
00000020	AD 00 00 00 01 73 52 47 42 00 AE CE 1C E9 00 00	.gAMA..a...
00000030	00 04 67 41 4D 41 00 00 B1 8F 0B FC 61 05 00 00	..phYS...
00000040	00 09 70 48 59 73 00 00 0E C3 00 00 0E C3 01 C7	o.oIDATx ^q
00000050	6F A8 64 00 00 86 6F 49 44 41 54 78 5E ED DD C9	.oIDATx ^q
00000060	CF 5D 67 82 DF 77 FE 93 59 66 91 01 59 04 88 91	.lgeW..oYfA.Y..����
00000070	04 31 90 04 B1 93 8E 61 C3 49 C7 EE 6E DB E5 EA	.1E..����.H..en...os



画像を回す

- 画像って?
 - バイナリデータ列...
 - テンソル... (グレースケールなら行列...)
 - 実数列とサイズの情報...
 - 30度回転した画像を計算するには?
 - ピクセルの間の扱いは...?
 - サイズは...?
 - むむむ...



画像を回す

- 画像って？
 - バイナリデータ列...
 - テンソル... (グレースケールなら行列...)
 - 実数列とサイズの情報...
 - 位置を与えると色を返す関数 ($f: \mathbb{R}^2 \rightarrow \mathbb{R}^3$)
- 30度回転した画像を計算するには?
 - 新たな関数 $g(x, y) = f(x \cos 30^\circ + y \sin 30^\circ, x \sin 30^\circ - y \cos 30^\circ)$ を考えれば良い (回転が逆向きなことに注意)
- 数学的表现は「合成」がしやすい
 - そういうものは見通しが良い (具体表現に依らない)

texture2Dのことです！

$$\mathbb{R}^2 \xrightarrow{\text{回転}} \mathbb{R}^2 \xrightarrow{f} \mathbb{R}^3$$

じゃあ、画像って？

- まず写像 ($\mathbb{R}^2 \rightarrow \mathbb{R}^3$) があった、と思う
 - 解像度、無限！
 - 色域、無限！
- しかしデータ量も無限...
- そこで近似を行う
 - 範囲を矩形に制限する
 - 色の最小・最大値を制限する
 - 空間を離散化する(ピクセルの登場、標本化)
 - 色を離散化する(量子化)
- 有限に収まった！ ← これが「画像」...と考えることもできる

じゃあ、画像って？

- 「本当に欲しかったものは最初から $\mathbb{R}^2 \rightarrow \mathbb{R}^3$ だった」
- シェーダを用いるとかなり "この感覚" で処理が書ける
 - 数学的記述を素直に反映させることができる

画像を写像として扱うには



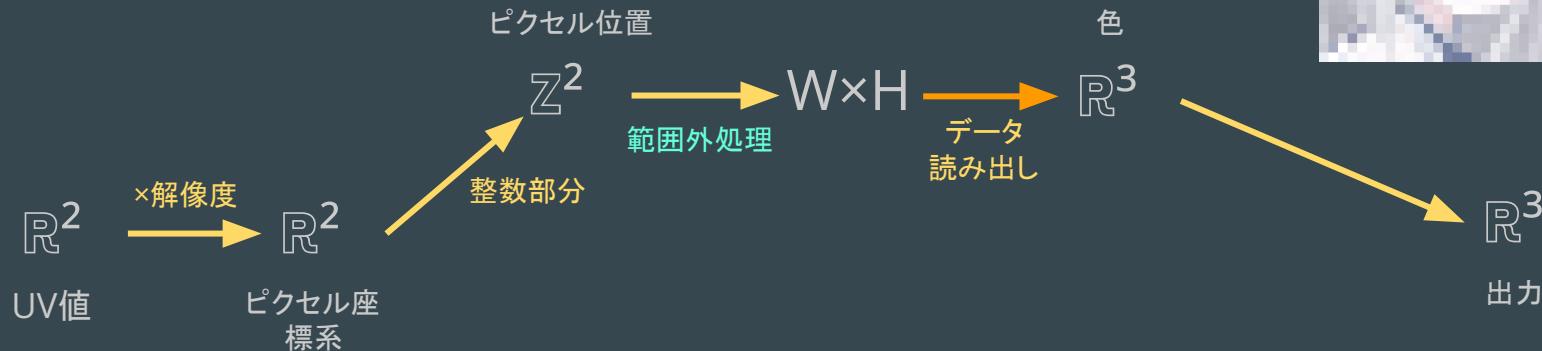
画像を写像として扱うには



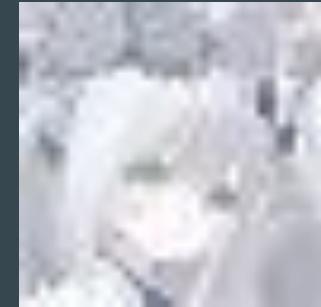
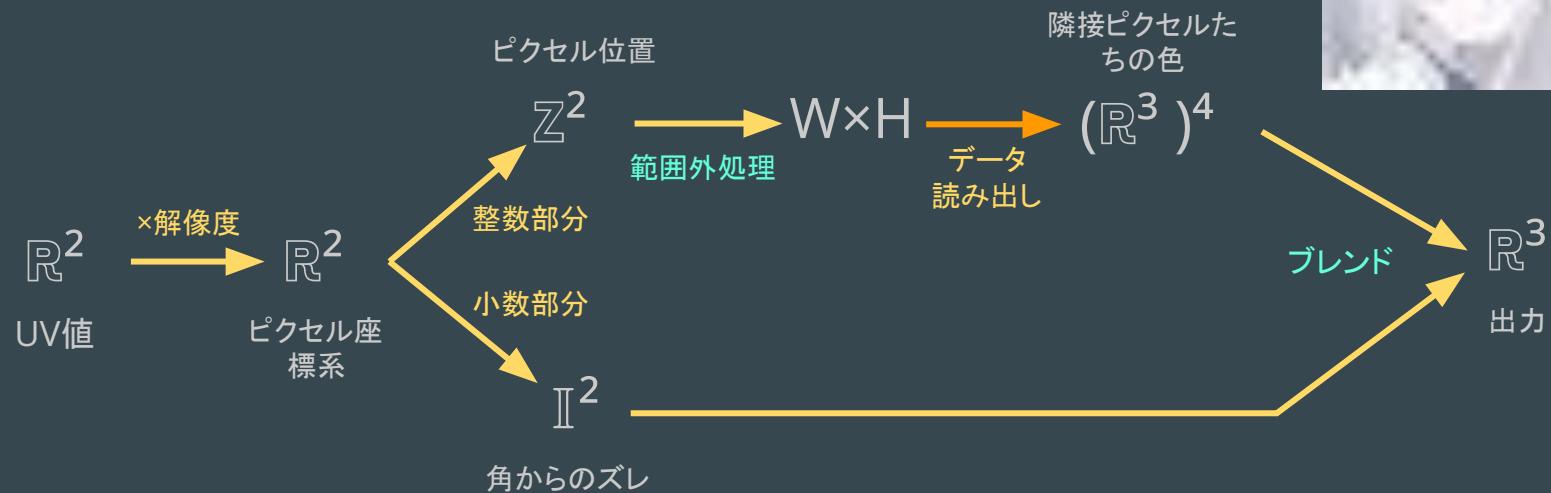
画像を写像として扱うには



?



画像を写像として扱うには



!

画像を写像として扱うには

- 「近似」を解釈しなおす必要がある
 - ピクセルとピクセルの間はどうする?
 - → **Texture Filtering** の設定
 - 範囲の外側はどうする?
 - → **Wrap Mode** の設定
- 機能が何故あるのか(必要なのか)のかがわかる
 - 「こんな機能があるんだ」ではなく、「これが無いと困るよね」

写像を画像にするには？

- うまく写像をピクセルデータにできるだろうか？
 - 中央でサンプルした色をピクセル色にしてみる

写像を画像にするには？

- うまく写像をピクセルデータにできるだろうか？
 - 中央でサンプルした色をピクセル色にしてみる



写像を画像にするには？

- うまく写像をピクセルデータにできるだろうか？
 - 中央でサンプルした色をピクセル色にしてみる



写像を画像にするには？

- うまく写像をピクセルデータにできるだろうか？
 - 中央でサンプルした色をピクセル色にしてみる



写像を画像にするには？

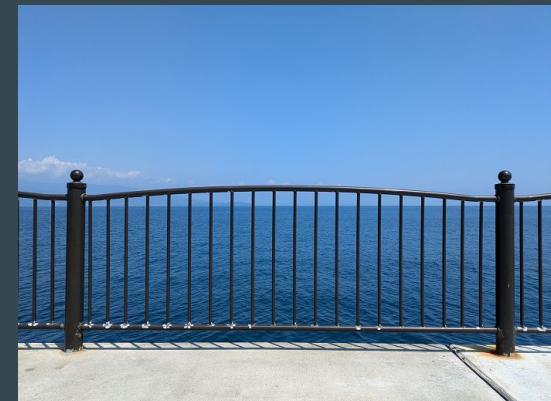
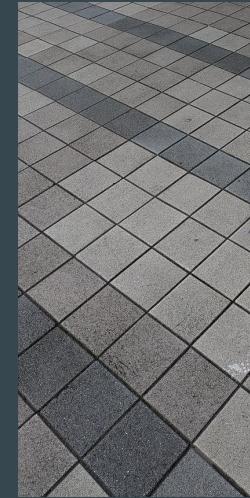
- うまく写像をピクセルデータにできるだろうか？
 - 中央でサンプルした色をピクセル色にしてみる
 - ここには恣意性がある
 - 「数学的に自然」じゃない
 - 別にピクセル左上を選んでも問題は起きない
 - つまり「最も良い選択」がない
 - 柔軟性があると、どこかに皺寄せが来ている...
 - 数学的直感
- 実際、1ピクセルにたくさんの画素が含まれていると...
 - ジャギ・モアレが発生

写像を画像にするには？

- 自然な選択: ピクセル範囲での**積分値**を表示する！
 - ピクセルを選択しない！
 - 計算が大変 → **MipMap** による高速な近似
- 複雑な機能も、数学側から解釈することでわかりやすくなるかも
 - 柔軟な捉え方ができるようになるかも
 - 各所で新たな機能を覚えるよりも、数学をやったほうが早い
 - 気がする

対称性と群作用

- 対称性はどこにでもある
 - 複雑に見えるものを作るときに
基本となるもの



対称性と群作用

- 対称性はどこにでもある
 - 複雑に見えるものを作るとときに基本となるもの
- 対称性といえば → 群
 - 対称性があるものに関する「形を変えない操作」を司る
 - 形を変えない操作を連ねても形を変えない (演算子の存在)
 - 「何もしない」のも形を変えない (単位元の存在)
 - 逆回しができる (逆元の存在)
 - 正確には、「対称性は群の作用として記述することができる」
 - 群作用によって扱えるものを、対称性と呼ぶ



対称性と群作用

- 対称性はどこにでもある
 - 複雑に見えるものを作るとときに基本となるもの
- 対称性といえば → 群
 - 対称性があるものに関する「形を変えない操作」を司る
 - 形を変えない操作を連ねても形を変えない (演算子の存在)
 - 「何もしない」のも形を変えない (単位元の存在)
 - 逆回しができる (逆元の存在)
 - 正確には、「対称性は群の作用として記述することができる」
 - 群作用によって扱えるものを、対称性と呼ぶ



対称性と群作用

- 対称性はどこにでもある
 - 複雑に見えるものを作るとときに基本となるもの
- 対称性といえば → 群
 - 対称性があるものに関する「形を変えない操作」を司る
 - 形を変えない操作を連ねても形を変えない(演算子の存在)
 - 「何もしない」のも形を変えない(単位元の存在)
 - 逆回しができる(逆元の存在)
 - 正確には、「対称性は群の作用として記述することができる」
 - 群作用によって扱えるものを、対称性と呼ぶ

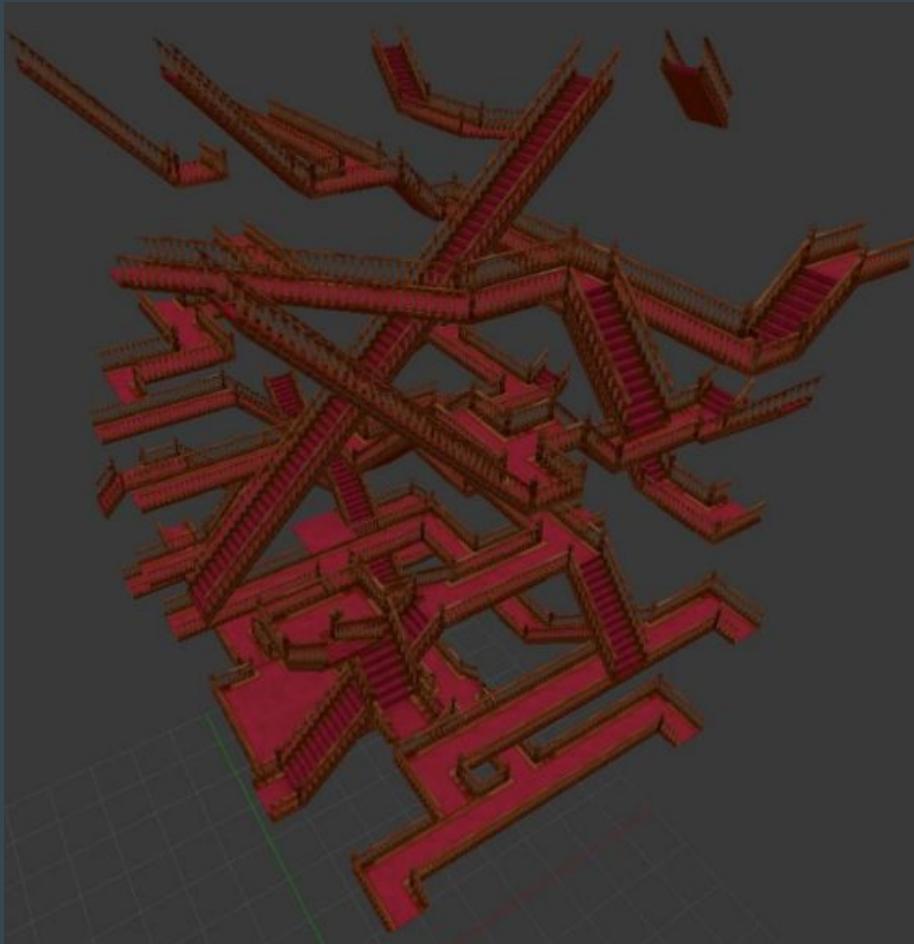
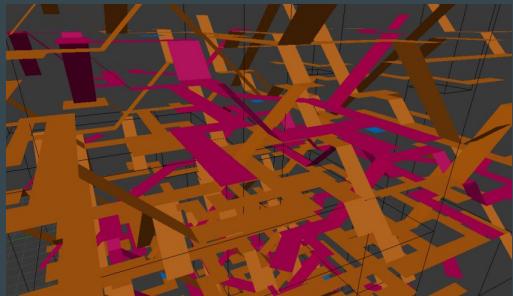
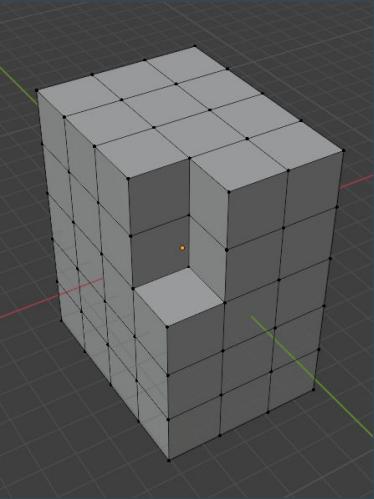


対称性と群作用

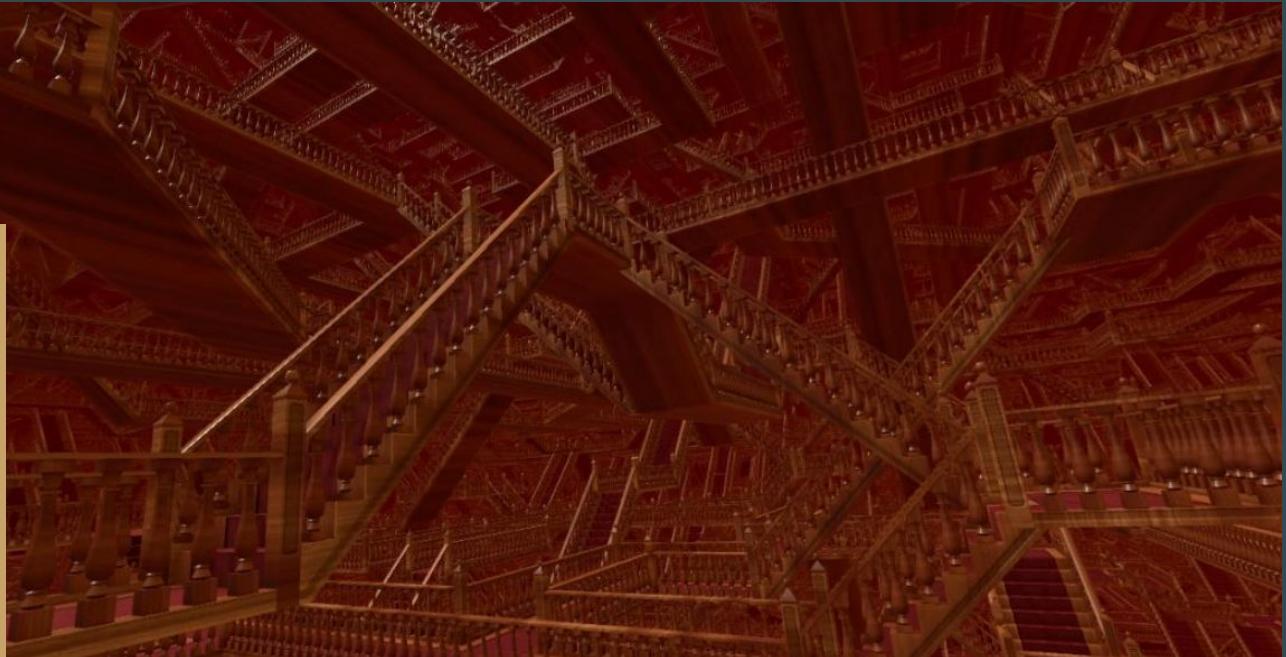
- 典型的な例
 - 二面体群 D_8
 - さつきのドーナツ
 - 整数の加法 $(\mathbb{Z}, +)$
 - 平行移動した繰り返し
 - 2次元格子 $(\mathbb{Z}^2, +)$
 - 上にも右にも繰り返し



昔作ったもの



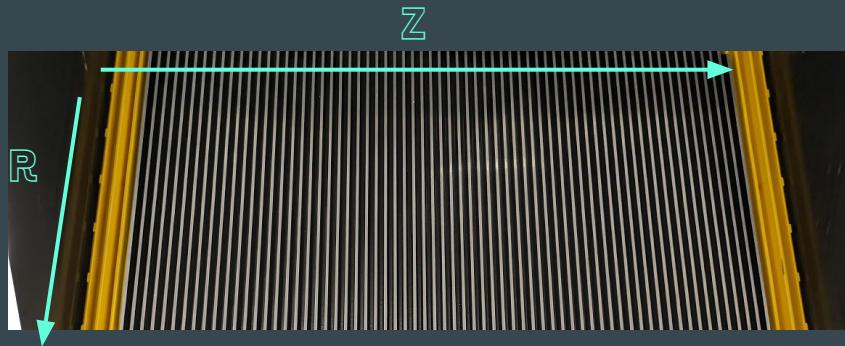
昔作ったもの



格子の一単位 (平行六面体)

対称性と群作用

- 典型的な例
 - 二面体群 D_8
 - さつきのドーナツ
 - 整数の加法 $(\mathbb{Z}, +)$
 - 平行移動した繰り返し
 - 2次元格子 $(\mathbb{Z}^2, +)$
 - 上にも右にも繰り返し
 - 実数の加法 $(\mathbb{R}, +)$
 - 円の回転 (\mathbb{S}^1, \times)
- 対称性に見えるものが増えるかも？



対称性と群作用

- 対称性があると → すごい わけじゃない

対称性と群作用

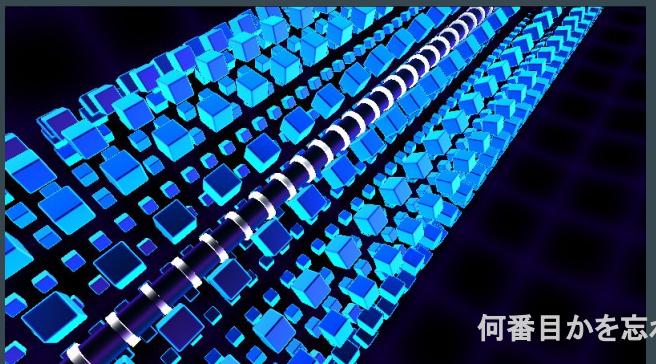
- 対称性があると → 単一パーツのことだけ考えれば良い！
 - 1個作ってコピペで増やせば良い
 - 輪郭から回転体を作れば良い
- シェーダで書くときは?
 - 1個作って空間内に増やすのではなく
 - 空間を折り畳んで1つだったことにする
- 空間内における違いを忘れる → 各パーツが同じになる
 - 極端な例: 座標を全部忘れる → 全部同じ色！

「嬉しい」といいます



対称性と群作用

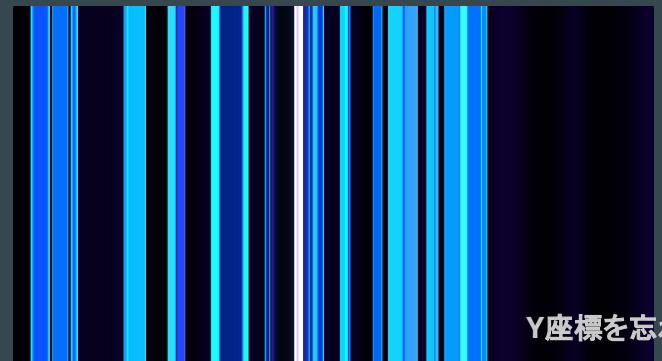
- 対称性があると → 単一パーツのことだけ考えれば良い！
 - 1個作ってコピペで増やせば良い
 - 輪郭から回転体を作れば良い
- シェーダで書くときは?
 - 1個作って空間内に増やすのではなく
 - 空間を折り畳んで1つだったことにする
- 空間内における違いを忘れる → 各パーツが同じになる
 - 極端な例: 座標を全部忘れる → 全部同じ色！



対称性と群作用

「嬉しい」といいます

- 対称性があると → 単一パーツのことだけ考えれば良い！
 - 1個作ってコピペで増やせば良い
 - 輪郭から回転体を作れば良い
- シエーダで書くときは？
 - 1個作って空間内に増やすのではなく
 - 空間を折り畳んで1つだったことにする
- 空間内における違いを忘れる → 各パーツが同じになる
 - 極端な例：座標を全部忘れる → 全部同じ色！



Y座標を忘れた

対称性と群作用

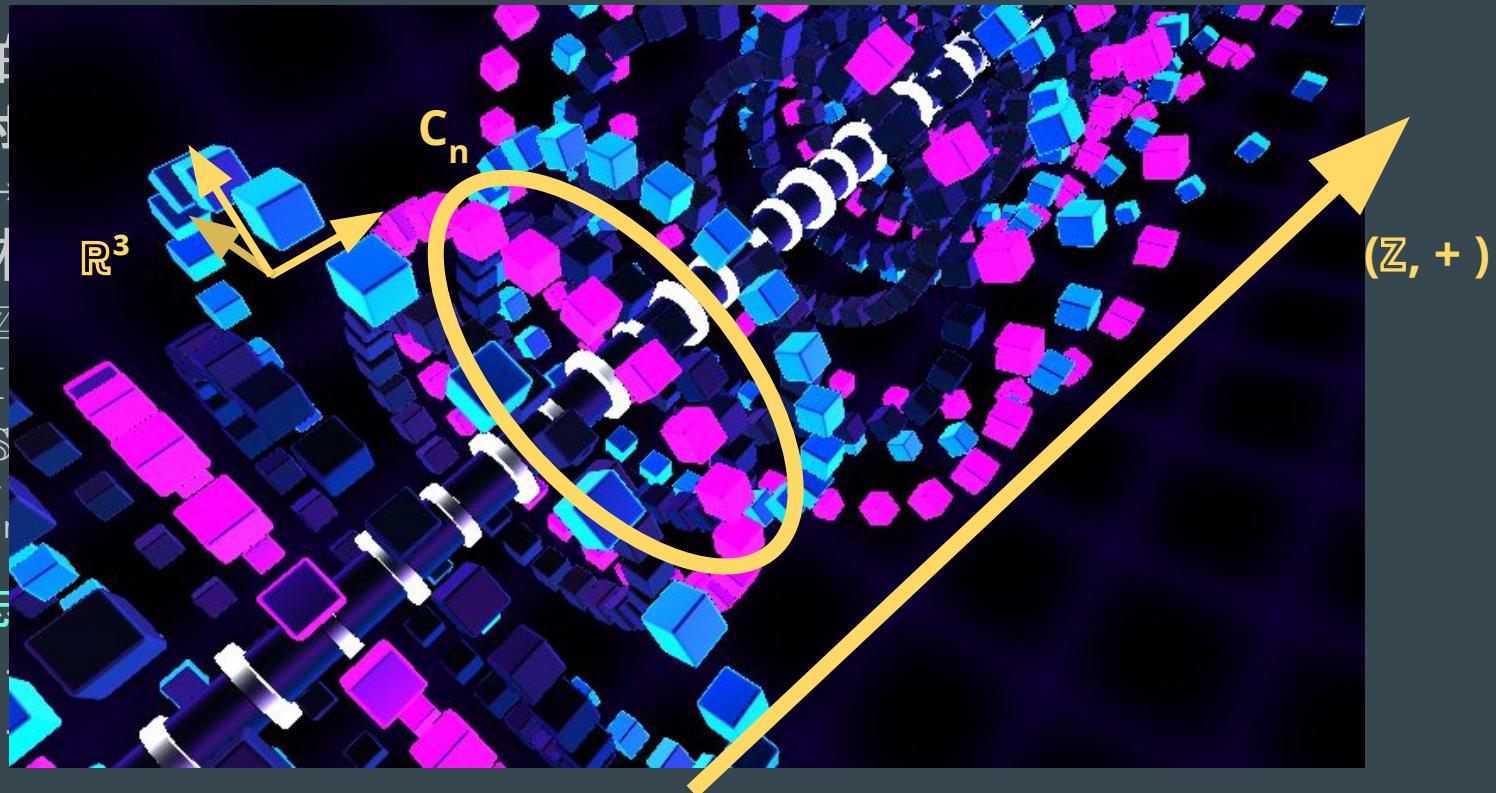
- 群作用に沿って忘れる
 - $(\mathbb{Z}, +)$ (平行移動) を忘れる → `frac(x)` や `mod(x, m)` だけを使う
 - $(\mathbb{Z}^2, +)$ (2次元格子) を忘れる → `frac(p.xy)` だけを使う
 - $(\mathbb{R}, +)$ (1つの軸) を忘れる → 使わなければ良い
 - (\mathbb{S}^1, \times) (円の回転) を忘れる → `length(p.xy)` だけを使う
 - C_n (回転対称) を忘れる → `polar-mod`, `pmod` って呼ばれてるみたい
- 何もしなければ対称！
 - 何かを使うことで私達は対称性を崩している

対称性と群作用

- 實用的には?
 - 対称すぎても面白くない ...
 - → 基本を対称形として、それぞれの違いを少しだけ思い出す
- 群の構造に沿って分解する
 - $(\mathbb{Z}, +)$ なら $(\text{floor}(x), \text{frac}(x))$
 - $(\mathbb{I}, +)$ なら $(\text{frac}(x), \text{floor}(x))$
 - (\mathbb{S}^1, \times) なら $(\text{normalize}(p.xy), \text{length}(p.xy))$ (atan2 でも良い)
 - C_n なら pmod の計算中に出てくる index (floor の結果) を思い出す
 - 忘れたものと忘れなかったものを組み合わせると元に戻せる
 - $x = \text{floor}(x) + \text{frac}(x)$
 - $p = \text{normalize}(p.xy) \times \text{length}(p.xy)$

対称性と群作用

- 実用的
- 対称
- 一元化
- 群の構造
- $(\mathbb{Z}, +)$
- (\mathbb{H}, \cdot)
- (S^1, \cdot)
- C_n
- 忘れ



対称性と群作用

- (これは別に数学的に面白い話というわけではない)
 - 準同型で送っただけ
 - 直積に分解しただけ
 - (双曲平面 \mathbb{H}^2 のタイル張りとかだと色々話題はあります)
- (何か思考の助けになることがあれば、くらい)
 - 私はこう考えるのが好き、という程度で受け取ってもらえば

どう数学をやるか

数式をどうやって読む？

- 文法がある
 - 要素の組み合わせでできている
 - それに定義がある
 - 典型的な記法がある
 - よく省略される部分もある
 - 文脈から意味を読み取る必要がある
- 自然言語部分もなんだかんだ大事
 - この式で何がしたいのか
 - 定義？計算？具体例？補足？
 - この式が何を表しているのか
 - 成り立ってほしい条件か？常に成り立つべき条件か？定義か？

慣れは必要...

数式をどうやって読む？

- 例: たまにある四元数の定義
 - 実数 a, b, c, d を使って $q = ai + bj + ck + d$ と表せるもの。
 - ここで i, j, k は $i^2 = j^2 = k^2 = ijk = -1$ を満たす。
- これで本当に「定義」になっているか？

数式をどうやって読む？

- 例: たまにある四元数の定義
 - 実数 a, b, c, d を使って $q = ai + bj + ck + d$ と表せるもの。
 - ここで i, j, k は $i^2 = j^2 = k^2 = ijk = -1$ を満たす。
- 書いてないこと(1)
 - ここでの + や (省略された) \times は結果的に普通の演算のように扱えるものの本当は演算子ではない(これ以上計算できない)
 - 「形式的に定義する」という言い方
 - 1 は本当は $0i + 0j + 0k + 1$ と書かなきゃいけない
 - 何なら $(0,0,0,1)$ でも良い
 - $a+b = b+a$ で、 $pi + qj = (p+q)i$ で、 $p(qi) = (pq)i$ と定義するからいろいろあって(実数の)1を(四元数の)1と書いて良い。

数式をどうやって読む？

- 例: たまにある四元数の定義
 - 実数 a, b, c, d を使って $q = ai + bj + ck + d$ と表せるもの。
 - ここで i, j, k は $i^2 = j^2 = k^2 = ijk = -1$ を満たす。
- 書いてないこと (2)
 - i, j, k に関する演算規則はこの等式によって確定する、からこれだけで定義が済ませて良い
 - 加法は明らか
 - 乗法は 3×3 通りが定まっていれば良い
 - $ij = ij \times (-k^2) = -(ijk)k = -(-1)k = k$
 - $ji = (-i^2) \times ji \times (-j^2) = i(ij)(ij)j = ik^2j = -ij = -k$
 - $ik = (-j^2) \times ik = -j(ji)k = jk^2 = -j$ など... 確かに定まっている

全く明らかではない



数式をどうやって読む？

- 例: たまにある四元数の定義
 - 実数 a, b, c, d を使って $q = ai + bj + ck + d$ と表せるもの。
 - ここで i, j, k は $i^2 = j^2 = k^2 = ijk = -1$ を満たす。
- 書いてないことがたくさんある
 - 最初から細部まで完璧に理解するような学び方は難しい
 - 人間向きではない
 - 一旦前提として受け入れる
 - 例や計算を通して雰囲気から認識していく方が良いかも
 - 間違いを正しながら進んでいこう
 - 練習問題や証明の理解が大事
 - 慣れてくると あ~いつものね みたいになってくる

数学の学び方

- ゼミ (Seminar) が典型的
 - 複数人で本を順番に担当を決めて読んでいく
 - 前の内容がわからないと次がわからなくなる → 全員の理解を確認！
 - 1冊を半年～数年とかそういうペース
- 単に数学書を「読む」だけではわからない
 - わかっていることを確認する為に人に伝えるのはかなり良い
 - 説明できなければわかっていない
 - 質問に答えられなければわかっていない
 - (質問が的外れなら的外れであることを指摘できるべき)
- わからない言葉を一度受け入れる
 - いつか困るから、そのときに考える

数学の学び方

- まとめ
 - 気軽に雰囲気から読んでみるのも面白いのではないかと思う
 - 人と一緒に読むと尚良い
 - 今ならある程度はChatGPTが全部答えてくれそうです
 - 気になったときに詳細を掘りに行くという気持ちで読む
 - わかってない部分をわかっていると強い
 - Wikipediaが便利
 - とりあえず先を読む
 - 定義より性質を見たほうが意味がわかりやすい
 - 自然言語部分をちゃんと読んだほうがわかりやすい
 - Wikipediaだと足りないところ...
- (個人の気持ちですが...)

終わりに

総括

- 数学と仲良くなろう！
 - 物事を理解しやすくなるかも
 - 物事を考えやすくなるかも
 - 物事を表現しやすくなるかも
 - 新たな視点に出会えるかも
- シェーダ的プログラミングとも仲が良い！
 - 正確には宣言的/関数型的プログラミングと仲が良い
- おすすめです！