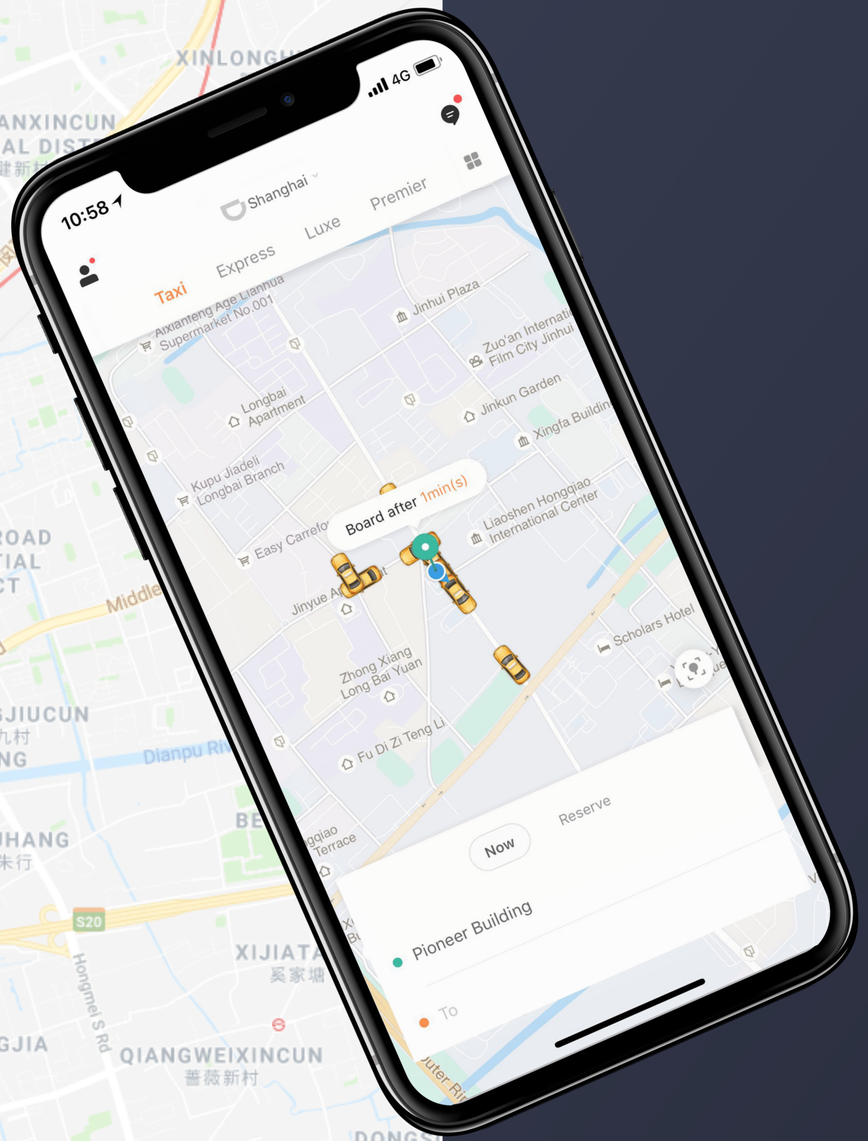


初探地图类 APP 后端那些事

空间索引算法

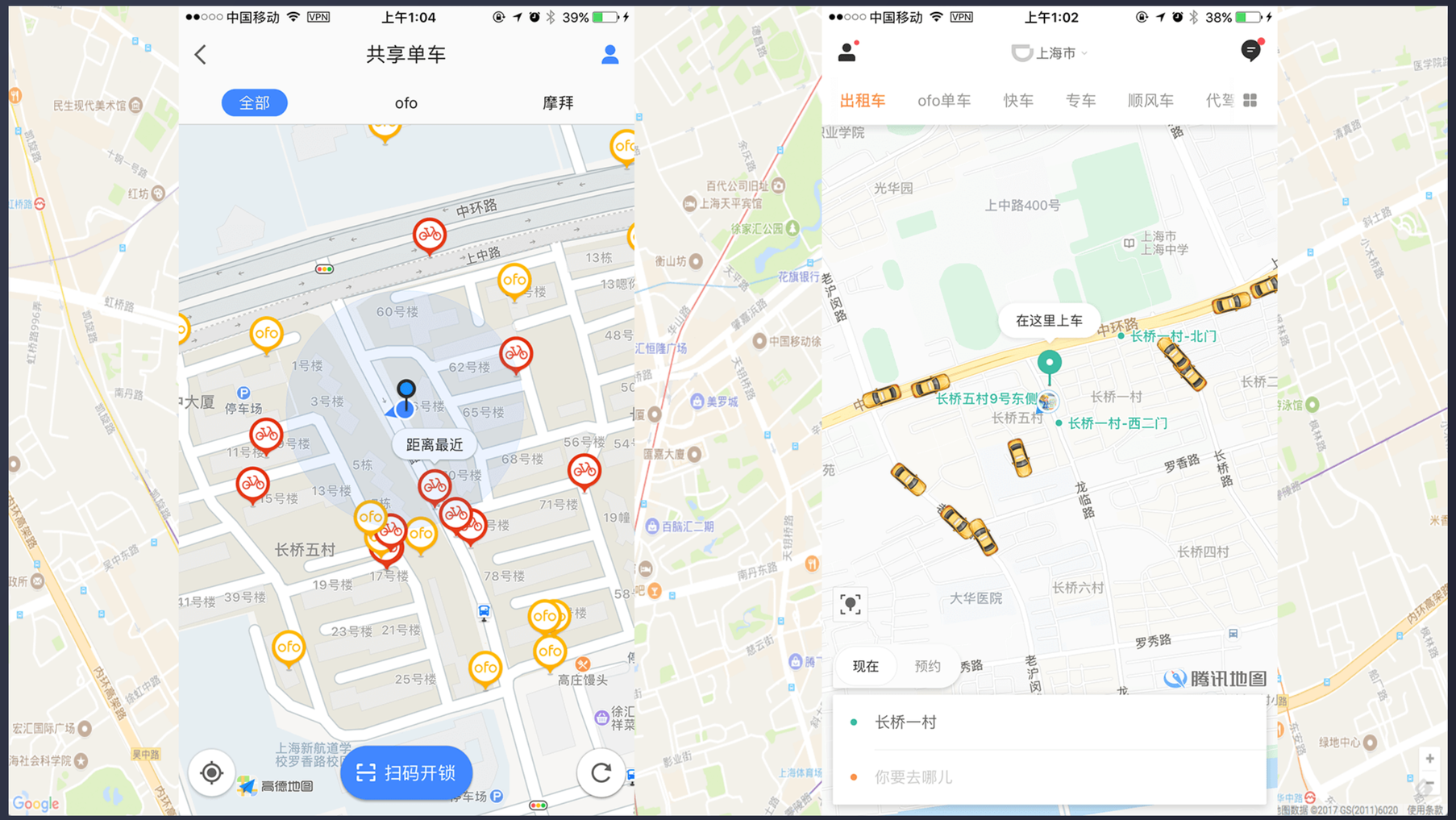
WELCOME

自我介绍

ID：一缕殇流化隐半边冰霜（花名：霜菜）

了解 iOS 开发、前端小白、后端新手
略懂 JavaScript、Go、C、C++、Objective-C、Swift

Github：@halfrost
微博：@halfrost


中国移动
上午1:04
39%
共享单车
全部
ofo
摩拜
距离最近
扫码开锁
长桥五村
中环路
上中路
高庄馒头
高德地图
Google
中国移动
上午1:02
38%
上海市
出租车
ofo单车
快车
专车
顺风车
在这里上车
长桥一村-北门
长桥五村9号东侧
长桥一村-西二门
现在
预约
长桥一村
你要去哪儿
腾讯地图

PRESENTATION AGENDA

01 Geohash

02 Space filling curve

03 Google S2

04 Application

SPATIAL INDEX

GEOHASH

NEXT SLIDE

#01

EXAMPLE

Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Base 32	0	1	2	3	4	5	6	7	8	9	b	c	d	e	f	g

Decimal	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Base 32	h	j	k	m	n	p	q	r	s	t	u	v	w	x	y	z

Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Base 36	2	3	4	5	6	7	8	9	b	B	C	d	D	F	g	G	h	H	j

Decimal	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
Base 36	J	K	l	L	M	n	N	P	q	Q	r	R	t	T	V	W	X

BASE-32 BASE-36

EXAMPLE


wtw37p
wtw37r
wtw37x
wtw37n
wtw37q
wtw37w
wtw37j
wtw37m
wtw37t
wtw37pp wtw37r0 wtw37r1 wtw37r4 wtw37r5 wtw37rh wtw37rj wtw37rm wtw37rp wtw37x0 wtw37x1
wtw37qf wtw37qg wtw37qu wtw37qv wtw37qy wtw37qz
wtw37qd wtw37qe wtw37qs wtw37qt wtw37qw wtw37qx
wtw37nr wtw37q2 wtw37q3 wtw37q6 wtw37q7 wtw37qk wtw37qm wtw37qq wtw37qr wtw37w2
wtw37q4 wtw37q5 wtw37qh wtw37qj wtw37qn wtw37qp
wtw37jz wtw37mb wtw37mc wtw37mf wtw37mg wtw37mu wtw37mv wtw37my wtw37mz wtw37tb
Google
地图数据 ©2017 GS(2011)6020 使用条款

GEOHASH (LEVEL-6)

左区间	中值	右区间	二进制结果
-90	0	90	1
0	45	90	0
0	22.5	45	1
22.5	33.75	45	0
22.5	28.125	33.75	1
28.125	30.9375	33.75	1
30.9375	32.34375	33.75	0
30.9375	31.640625	32.34375	0
30.9375	31.2890625	31.640625	0
30.9375	31.1132812	31.2890625	1
31.1132812	31.2011718	31.2890625	0
31.1132812	31.1572265	31.2011718	1
31.1572265	31.1791992	31.2011718	1
31.1791992	31.1901855	31.2011718	1
31.1901855	31.1956786	31.2011718	0

左区间	中值	右区间	二进制结果
-180	0	180	1
0	90	180	1
90	135	180	0
90	112.5	135	1
112.5	123.75	135	0
112.5	118.125	123.75	1
118.125	120.9375	123.75	1
120.9375	122.34375	123.75	0
120.9375	121.640625	122.34375	0
120.9375	121.289062	121.640625	1
121.289062	121.464844	121.640625	0
121.289062	121.376953	121.464844	1
121.376953	121.420898	121.464844	1
121.420898	121.442871	121.464844	0
121.420898	121.431885	121.442871	1

(31.1932993, 121.43960190000007)

GEOHASH (LEVEL-6)

LONGITUDE 经度

110101100101101

LATITUDE 纬度

101011000101110

Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Base 32	0	1	2	3	4	5	6	7	8	9	b	c	d	e	f	g

Decimal	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Base 32	h	j	k	m	n	p	q	r	s	t	u	v	w	x	y	z

11100 11001 11100 00011 00111 10110

28 25 28 3 7 22

wtw37q

NEXT

EXAMPLE


wtw37p
wtw37r
wtw37x
wtw37n
wtw37q
wtw37w
wtw37j
wtw37m
wtw37t
wtw37pp wtw37r0 wtw37r1 wtw37r4 wtw37r5 wtw37rh wtw37rj wtw37rm wtw37rp wtw37x0 wtw37x1
wtw37qf wtw37qg wtw37qu wtw37qv wtw37qy wtw37qz
wtw37qd wtw37qe wtw37qs wtw37qt wtw37qw wtw37qx
wtw37nr wtw37q2 wtw37q3 wtw37q6 wtw37q7 wtw37qk wtw37qm wtw37qq wtw37qr wtw37w2
wtw37q4 wtw37q5 wtw37qh wtw37qj wtw37qn wtw37qp
wtw37jz wtw37mb wtw37mc wtw37mf wtw37mg wtw37mu wtw37mv wtw37my wtw37mz wtw37tb
Google
地图数据 ©2017 GS(2011)6020 使用条款

CODE

```go
package geohash

import (
    "bytes"
)

const (
    BASE32                = "0123456789bcdefghjkmnpqrstuvwxyz"
    MAX_LATITUDE  float64 = 90
    MIN_LATITUDE  float64 = -90
    MAX_LONGITUDE float64 = 180
    MIN_LONGITUDE float64 = -180
)

var (
    bits   = []int{16, 8, 4, 2, 1}
    base32 = []byte(BASE32)
)

type Box struct {
    MinLat, MaxLat float64 // 纬度
    MinLng, MaxLng float64 // 经度
}

func (this *Box) Width() float64 {
    return this.MaxLng - this.MinLng
}

func (this *Box) Height() float64 {
    return this.MaxLat - this.MinLat
}
```

```go
// 输入值：纬度，经度，精度(geohash的长度)
// 返回geohash，以及该点所在的区域
func Encode(latitude, longitude float64, precision int) (string, *Box) {
    var geohash bytes.Buffer
    var minLat, maxLat float64 = MIN_LATITUDE, MAX_LATITUDE
    var minLng, maxLng float64 = MIN_LONGITUDE, MAX_LONGITUDE
    var mid float64 = 0

    bit, ch, length, isEven := 0, 0, 0, true
    for length < precision {
        if isEven {
            if mid = (minLng + maxLng) / 2; mid < longitude {
                ch |= bits[bit]
                minLng = mid
            } else {
                maxLng = mid
            }
        } else {
            if mid = (minLat + maxLat) / 2; mid < latitude {
                ch |= bits[bit]
                minLat = mid
            } else {
                maxLat = mid
            }
        }
        isEven = !isEven
        if bit < 4 {
            bit++
        } else {
            geohash.WriteByte(base32[ch])
            length, bit, ch = length+1, 0, 0
        }
    }
    b := &Box{
        MinLat: minLat,
        MaxLat: maxLat,
        MinLng: minLng,
        MaxLng: maxLng,
    }
    return geohash.String(), b
}
```

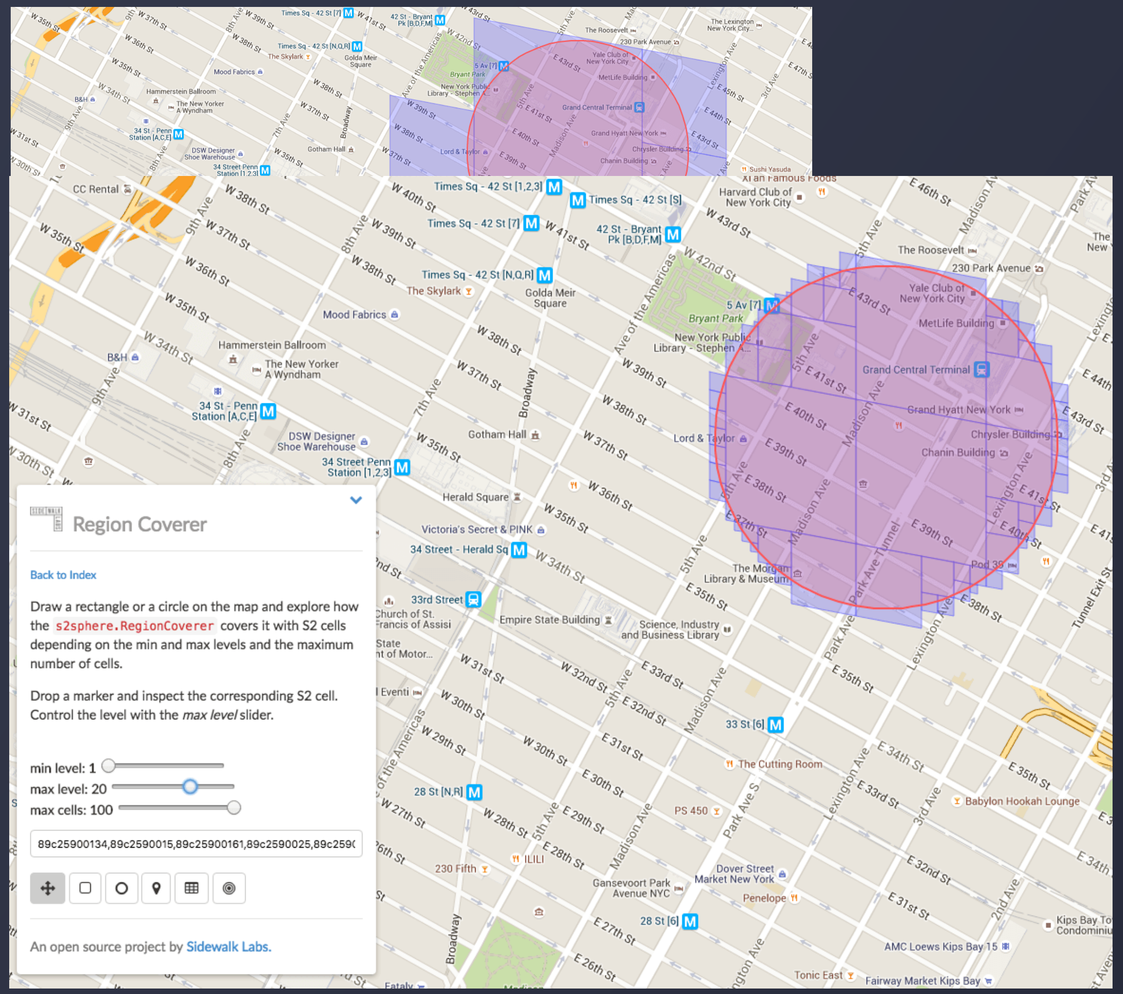
SCALE

字符串长度		cell 宽度		cell 高度
1	≤	5,000km	×	5,000km
2	≤	1,250km	×	625km
3	≤	156km	×	156km
4	≤	39.1km	×	19.5km
5	≤	4.89km	×	4.89km
6	≤	1.22km	×	0.61km
7	≤	153m	×	153m
8	≤	38.2m	×	19.1m
9	≤	4.77m	×	4.77m
10	≤	1.19m	×	0.596m
11	≤	149mm	×	149mm
12	≤	37.2mm	×	18.6mm

ERROR

Geohash 字符串长度	纬度	经度	纬度误差	经度误差	km误差
1	2	3	±23	±23	±2500
2	5	5	±2.8	±5.6	±630
3	7	8	±0.70	±0.70	±78
4	10	10	±0.087	±0.18	±20
5	12	13	±0.022	±0.022	±2.4
6	15	15	±0.0027	±0.0055	±0.61
7	17	18	±0.00068	±0.00068	±0.076
8	20	20	±0.000085	±0.00017	±0.019
9	22	23			
10	25	25			
11	27	28			
12	30	30			

SHARDING RULE



WRITE (UPDATE LOCATION)

根据经纬度计算 Shard ID 及 Cell ID
若上一次在同一个 Cell，直接更新
若上一次在不同 Cell，先删除上一个 Cell，再加入当前 Cell 的队列中。

READ (NEARBY SEARCH)

计算相交的 Shard
每个 Shard 做并行 Nearby 计算。
Geohash 只能先按照广度优先查找相应的 Cell，过滤出 Cell 中符合条件的点。

WHY

- 偶数位放经度，奇数位放纬度

  why?

- 理论基础

  why?

- 有缺点么？

  why?

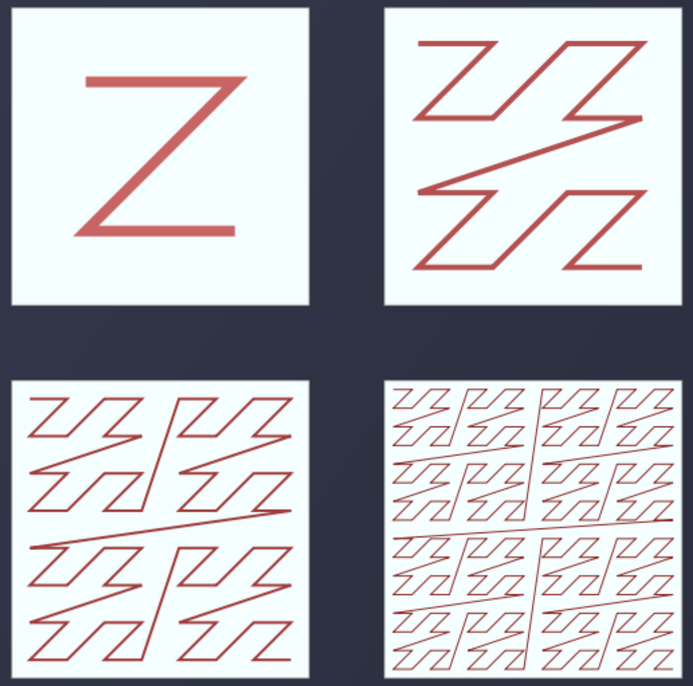
SPATIAL FILLING CURVE

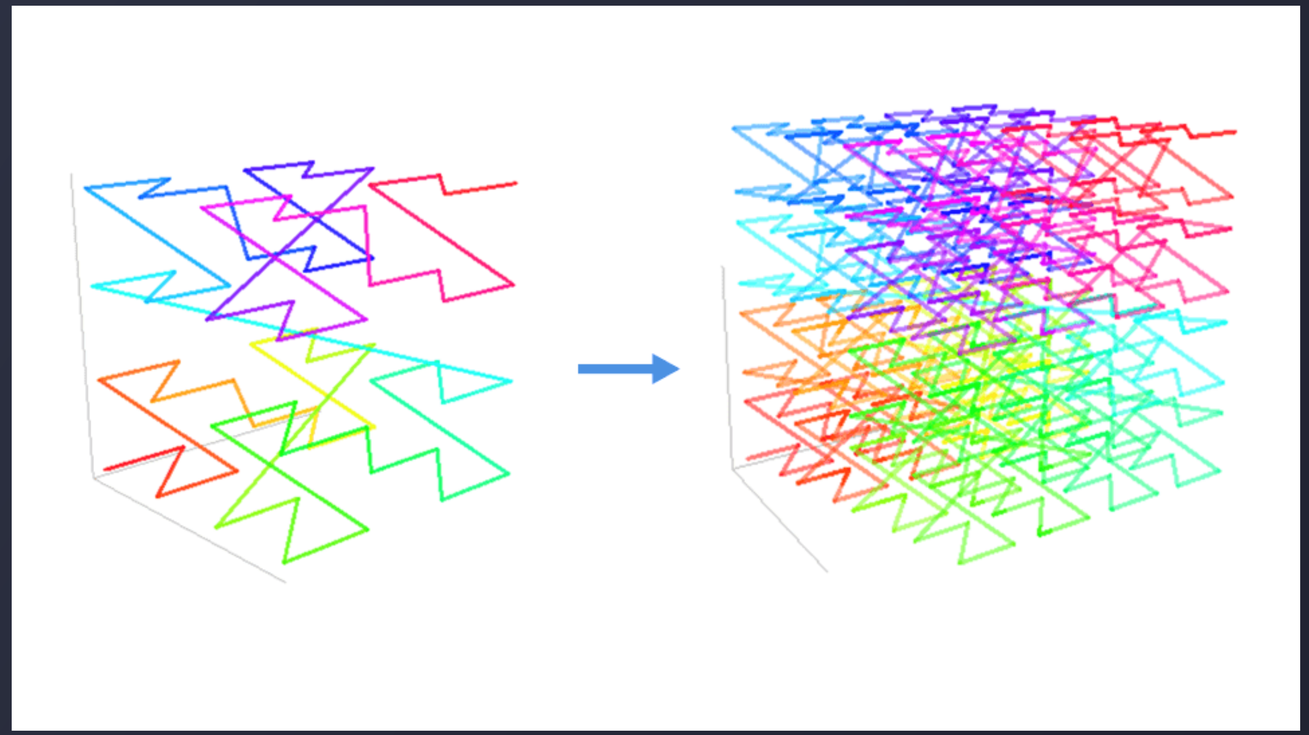
SPACE FILLING CURVE

NEXT SLIDE

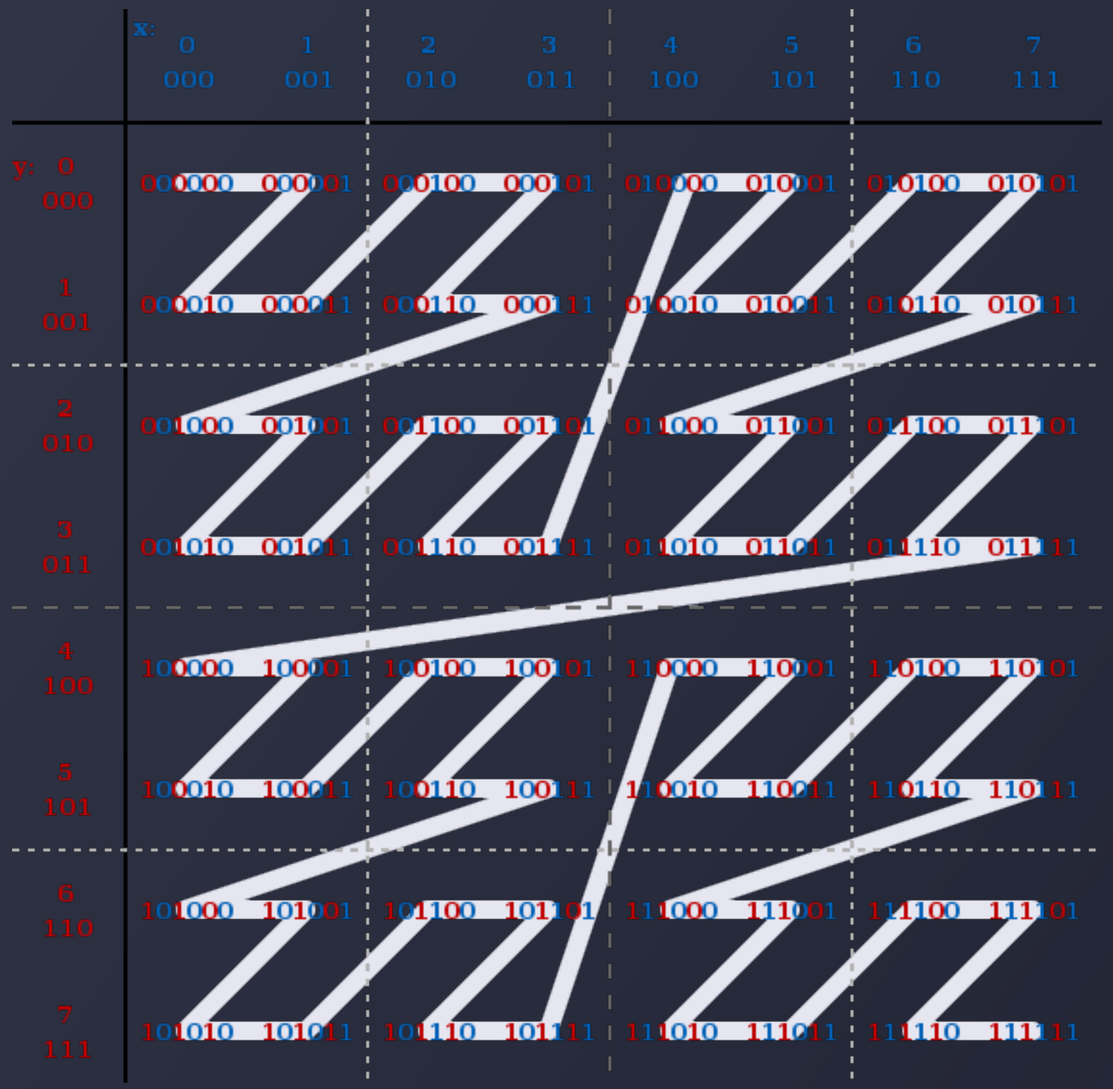
#02

Z-ORDER

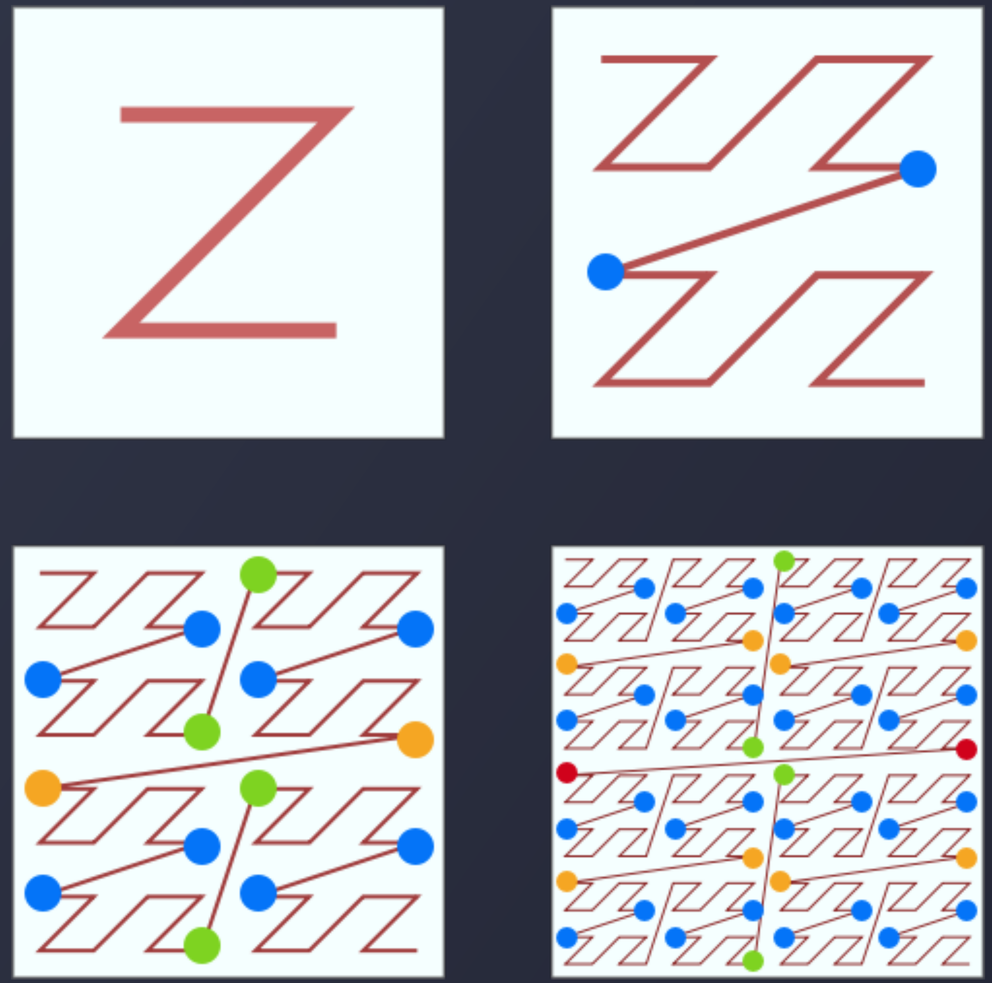




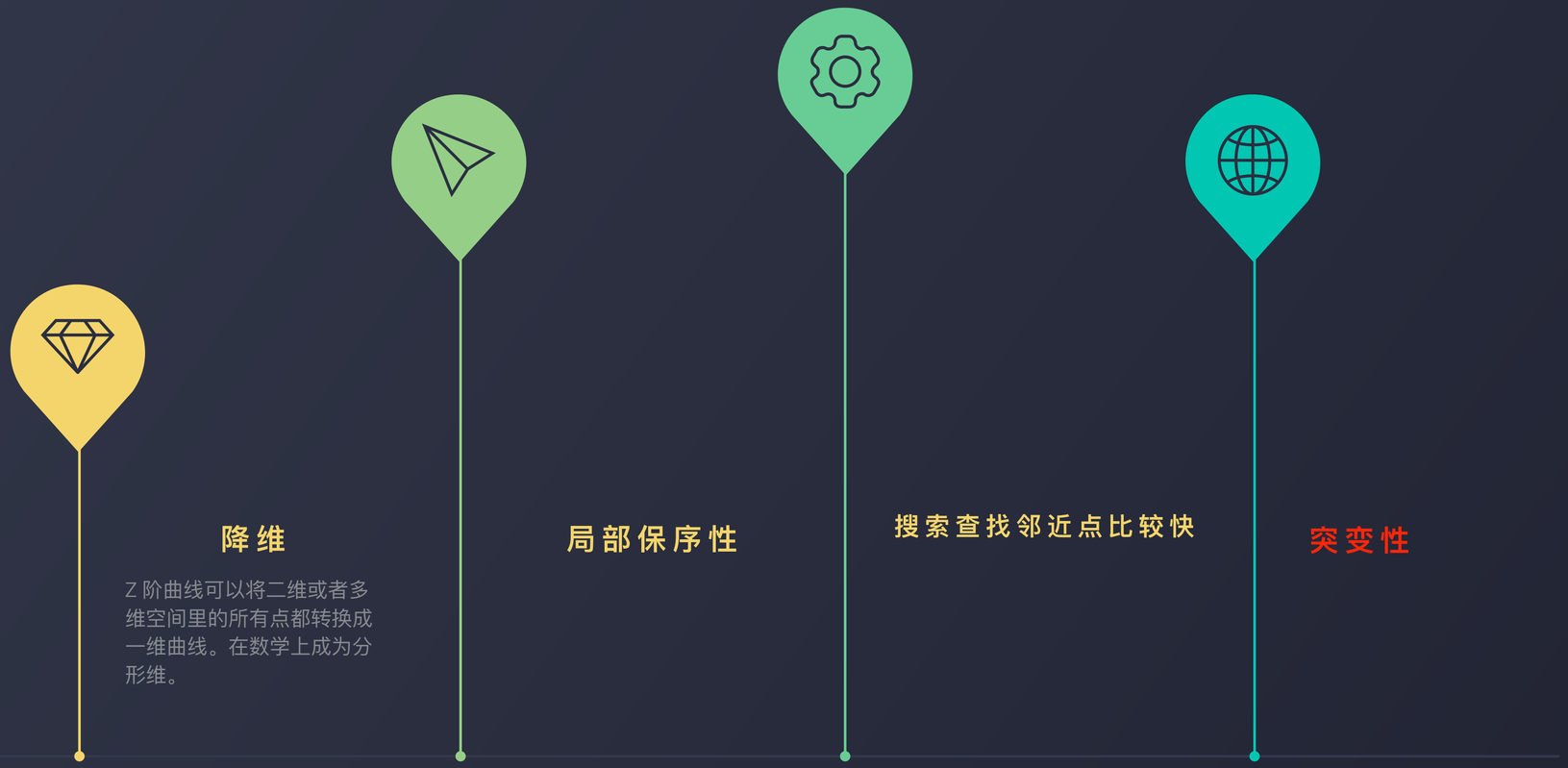
Z-ORDER


x:
0 000
1 001
2 010
3 011
4 100
5 101
6 110
7 111
y:
0 000
1 001
2 010
3 011
4 100
5 101
6 110
7 111
000000 000001 000100 000101 010000 010001 010100 010101
000010 000011 000110 000111 010010 010011 010110 010111
001000 001001 001100 001101 011000 011001 011100 011101
001010 001011 001110 001111 011010 011011 011110 011111
100000 100001 100100 100101 110000 110001 110100 110101
100010 100011 100110 100111 110010 110011 110110 110111
101000 101001 101100 101101 111000 111001 111100 111101
101010 101011 101110 101111 111010 111011 111110 111111

DISADVANTAGE



DISADVANTAGE



降维

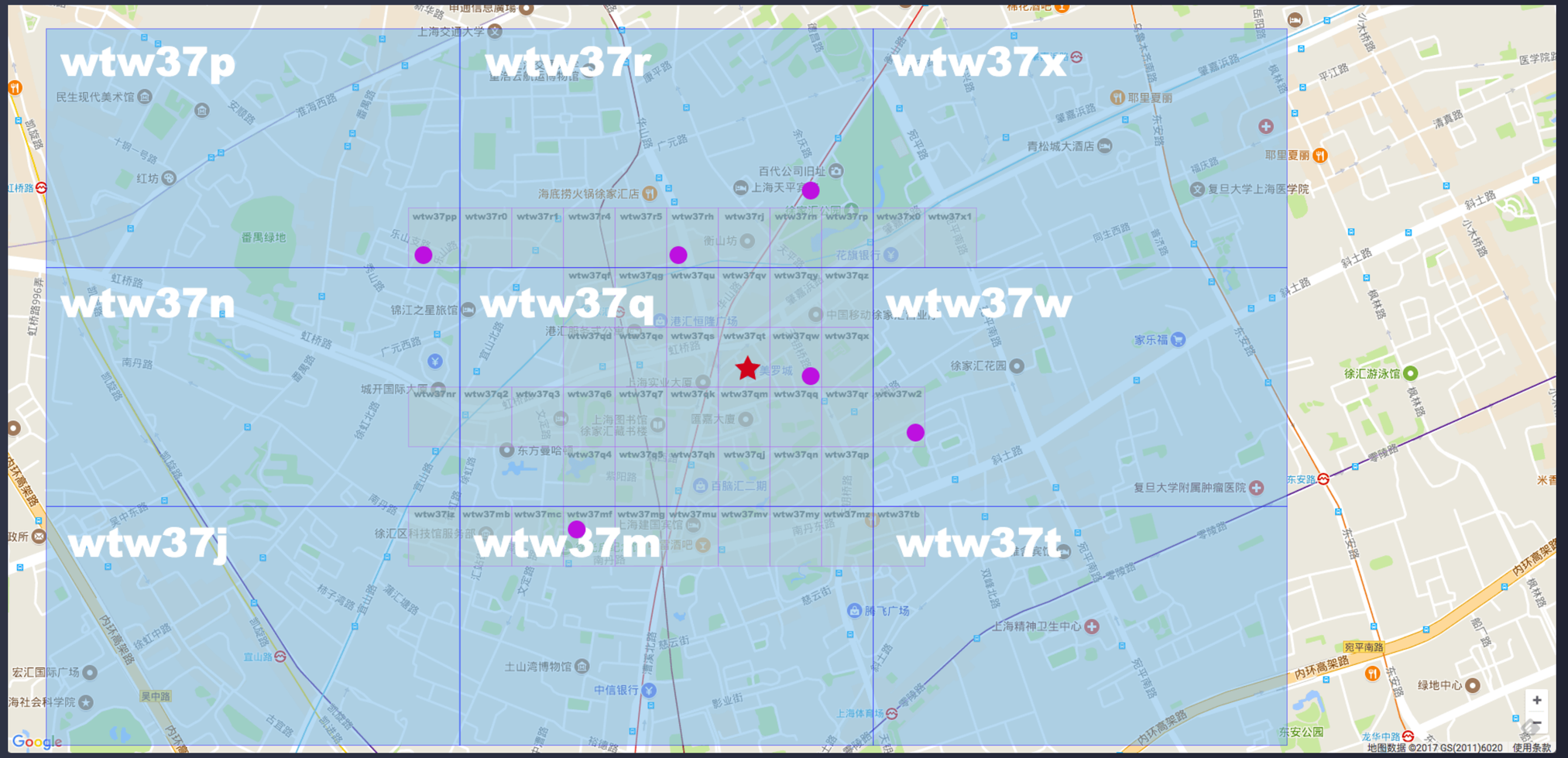
Z 阶曲线可以将二维或者多维空间里的所有点都转换成一维曲线。在数学上成为分形维。

局部保序性

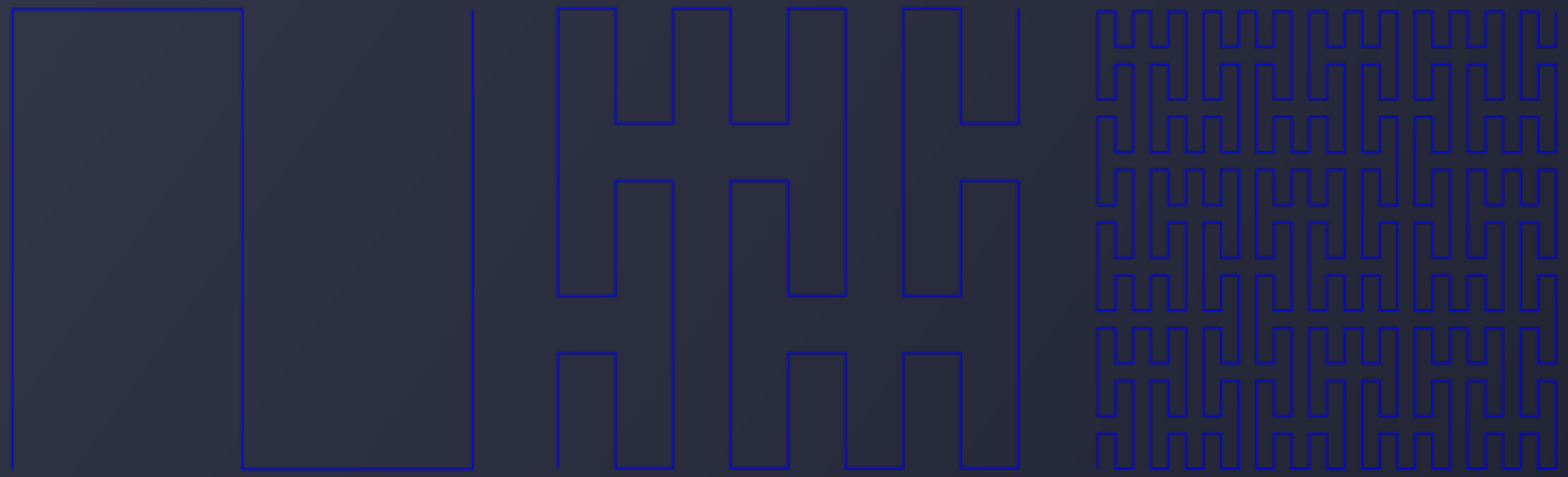
搜索查找邻近点比较快

突变性

EXAMPLE

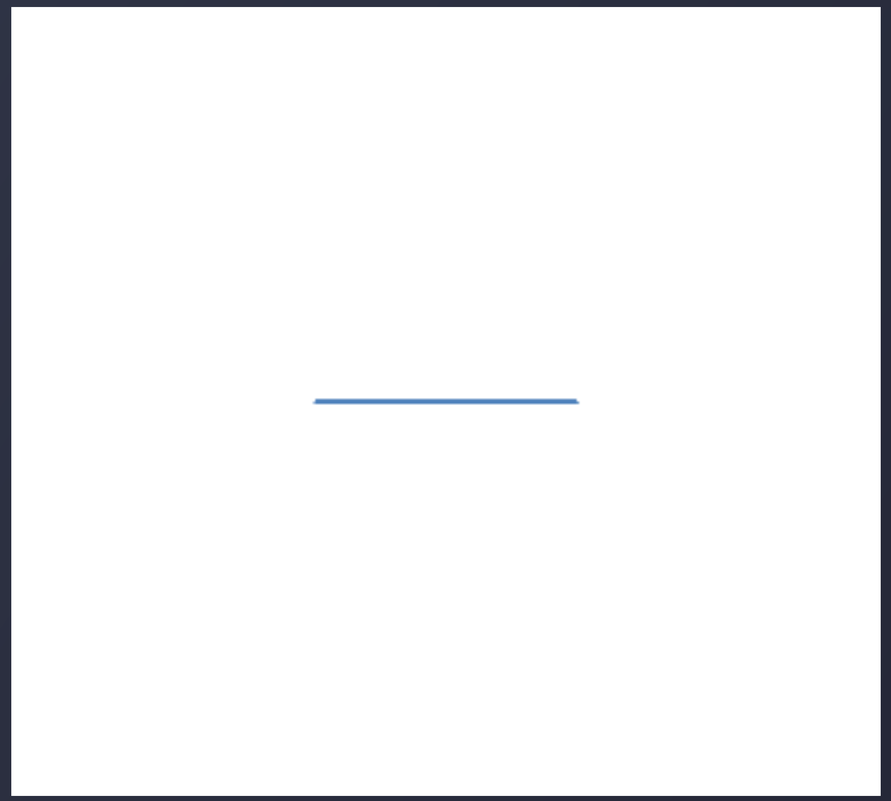

wtw37p
wtw37r
wtw37x
wtw37n
wtw37q
wtw37w
wtw37j
wtw37m
wtw37t
wtw37pp wtw37r0 wtw37r1 wtw37r4 wtw37r5 wtw37rh wtw37rj wtw37rn wtw37rp wtw37x0 wtw37x1
wtw37nr wtw37q2 wtw37q3 wtw37q6 wtw37q7 wtw37qk wtw37qm wtw37qq wtw37qr wtw37w2
wtw37jz wtw37mb wtw37mc wtw37mf wtw37mg wtw37mu wtw37mv wtw37my wtw37mz wtw37tb
Google

PEANO CURVE



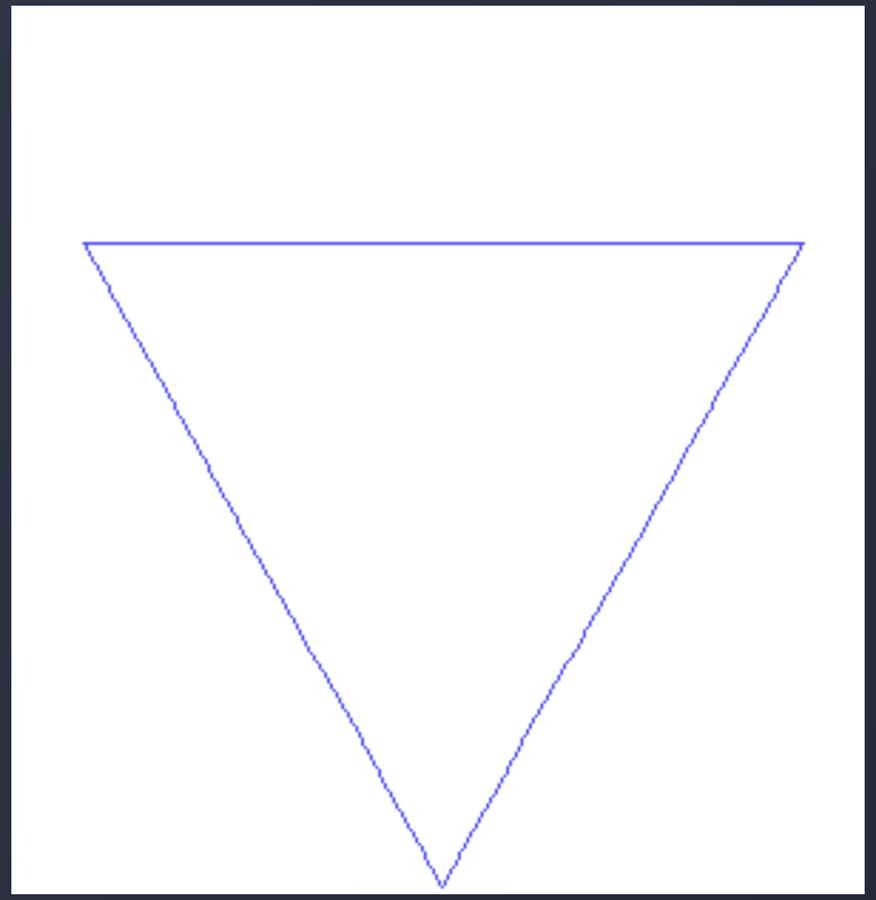
皮亚诺曲线是一条连续的但处处不可导的曲线。

DRAGON CURVE

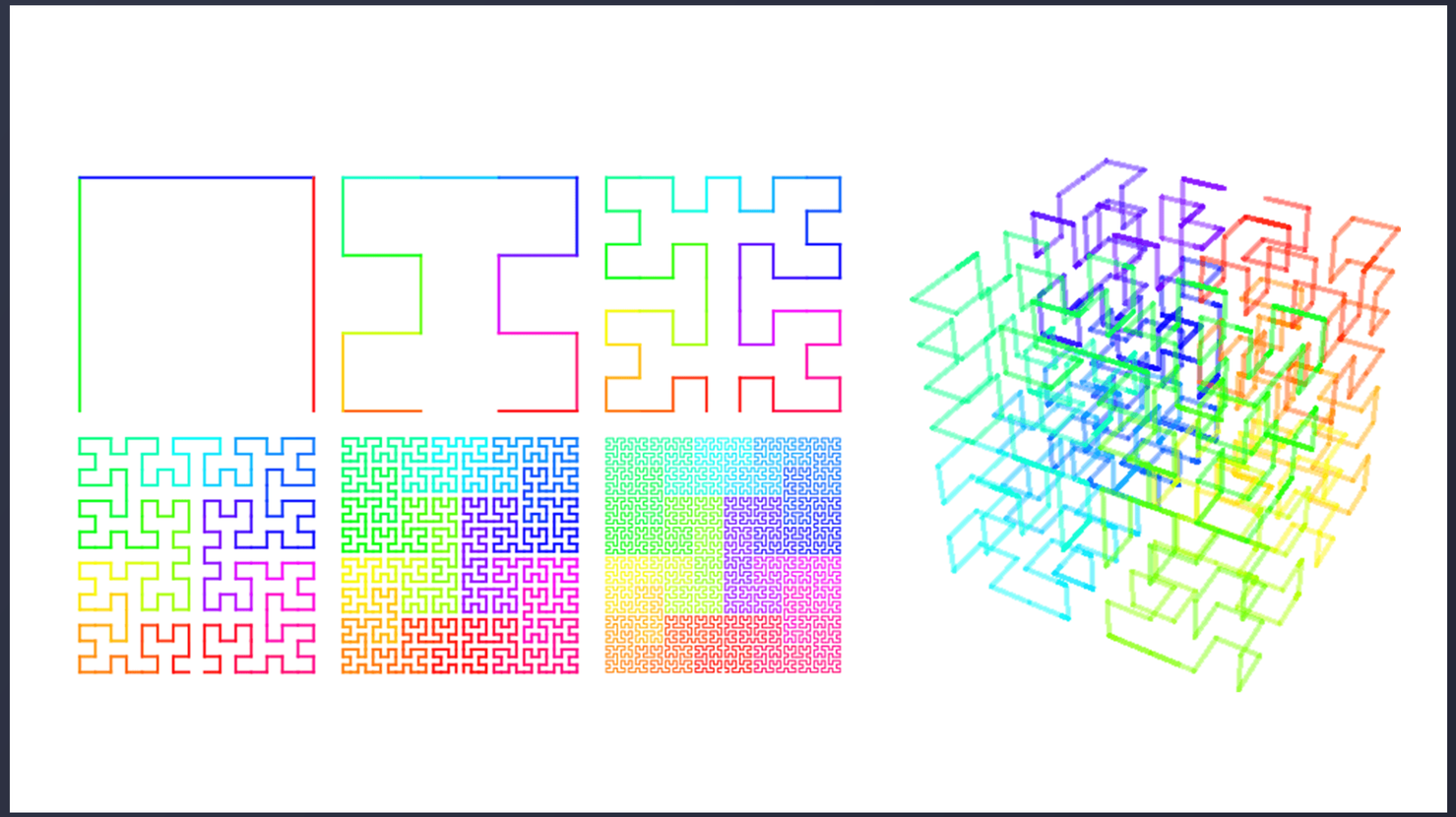


GOSPER CURVE

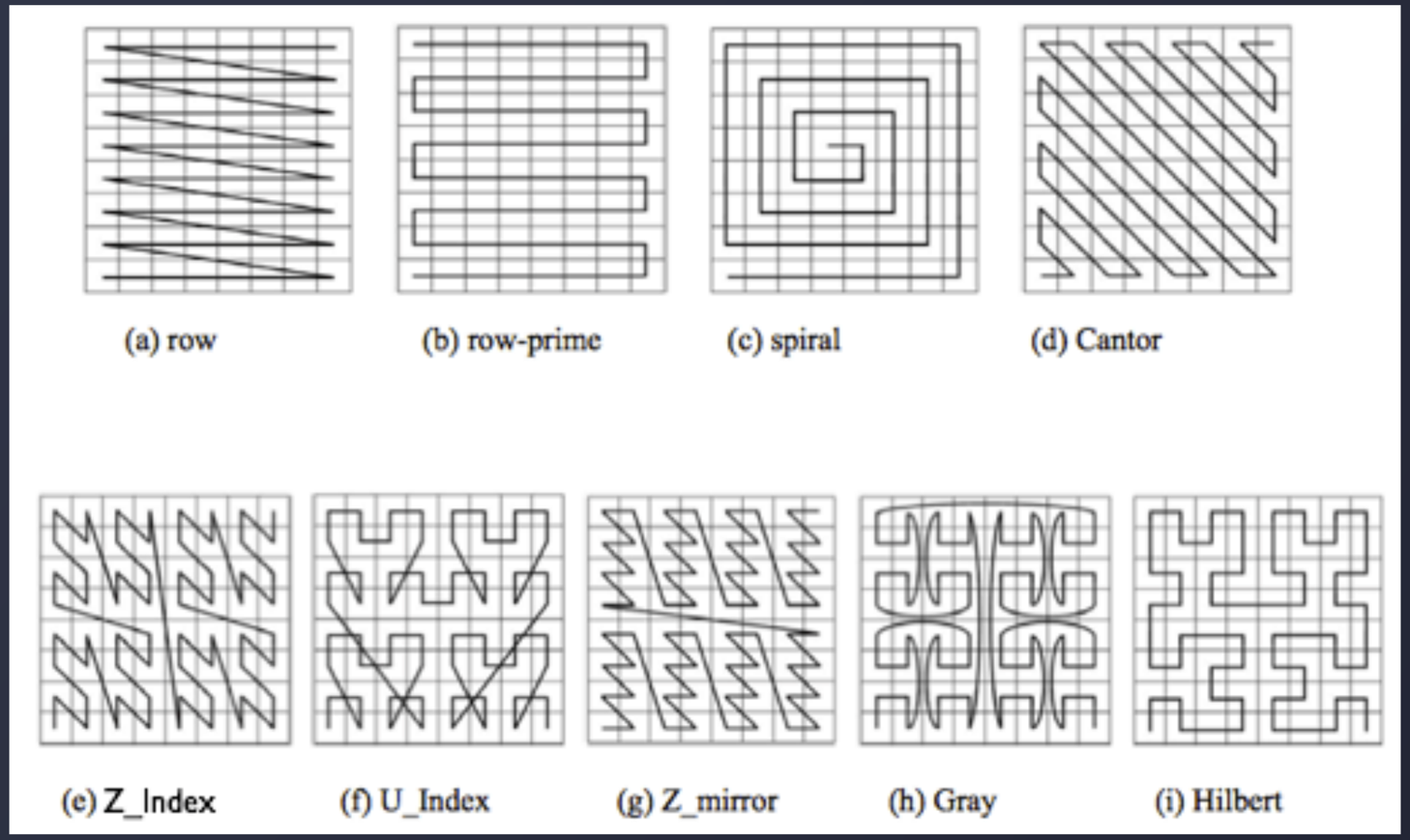
KOCH CURVE



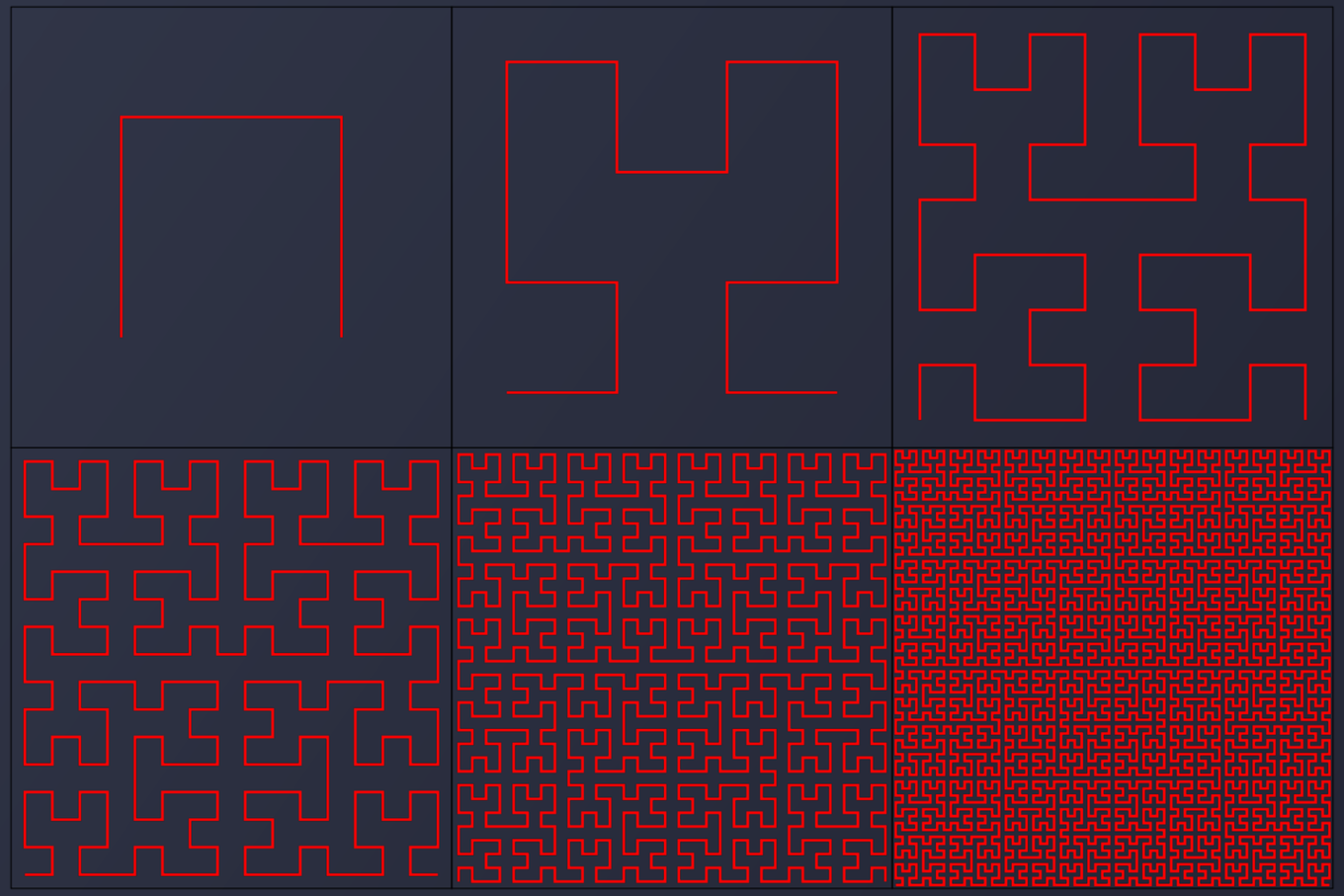
MOORE CURVE



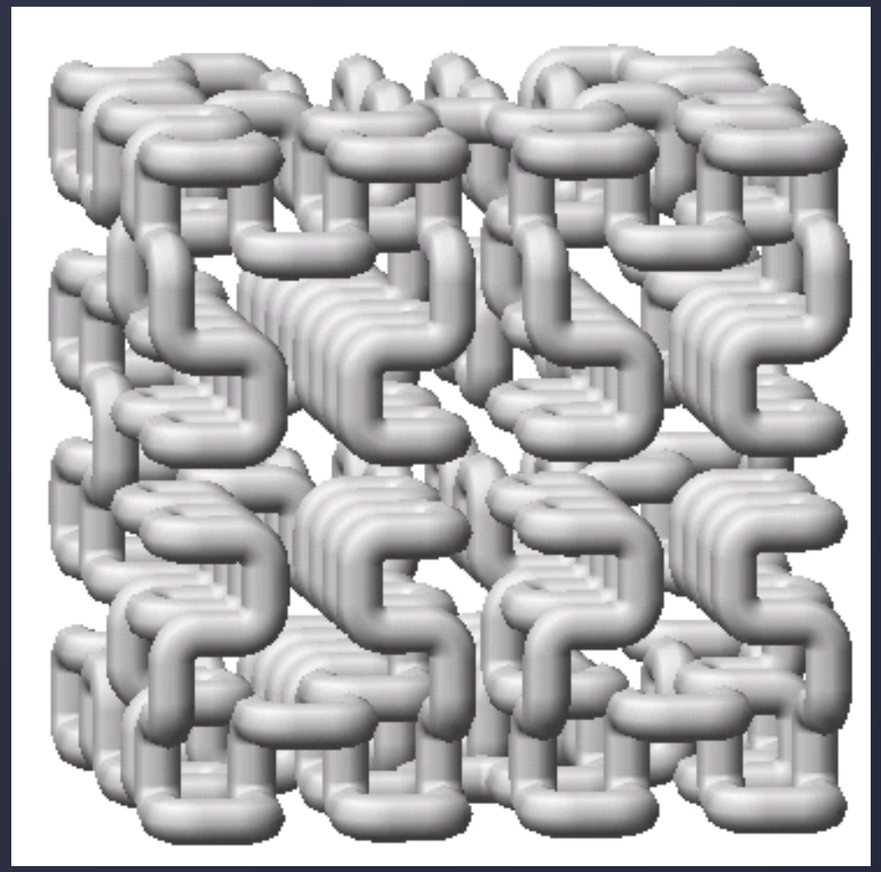
SIERPIŃSKI CURVE



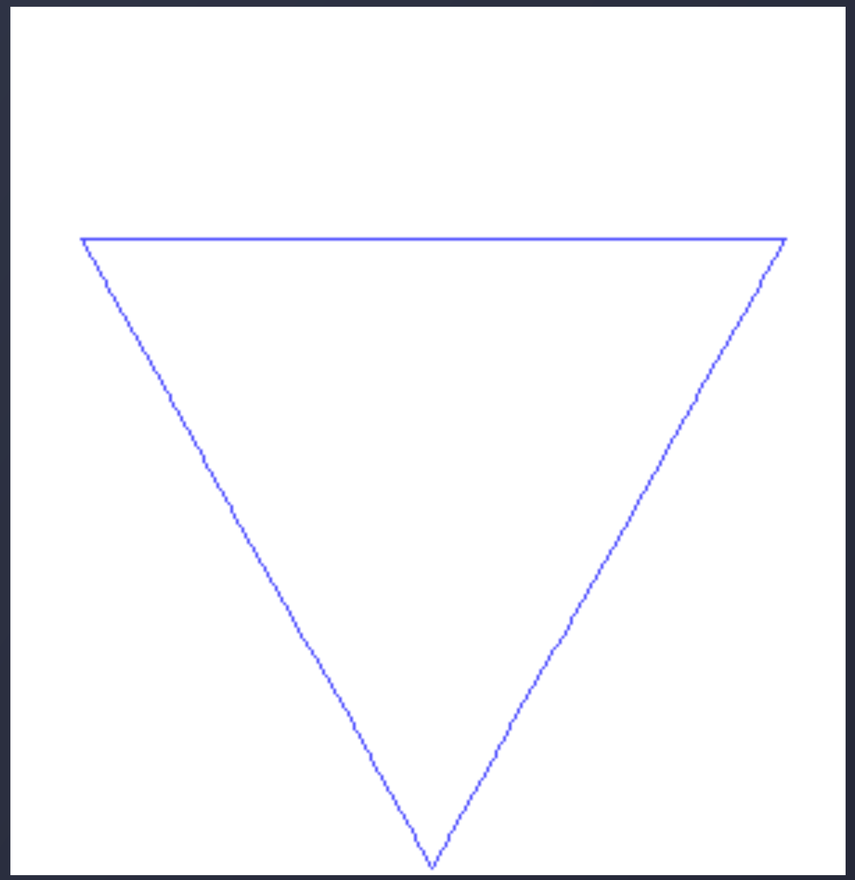
HILBERT CURVE



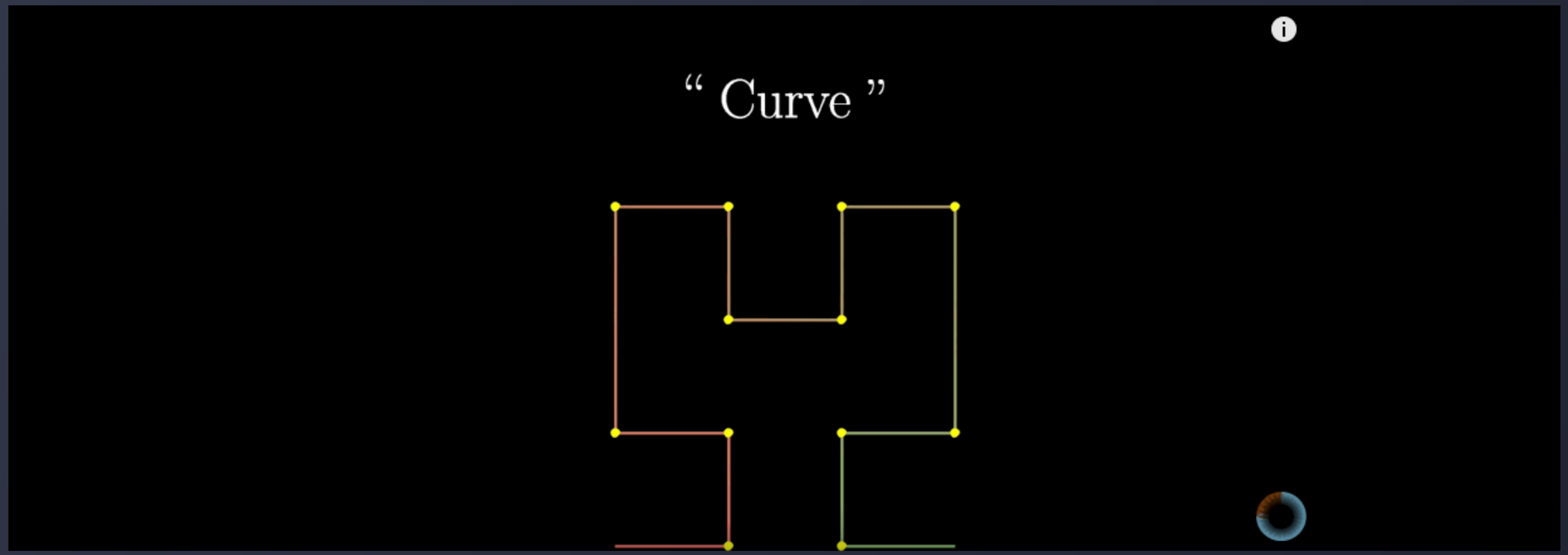
HILBERT CURVE



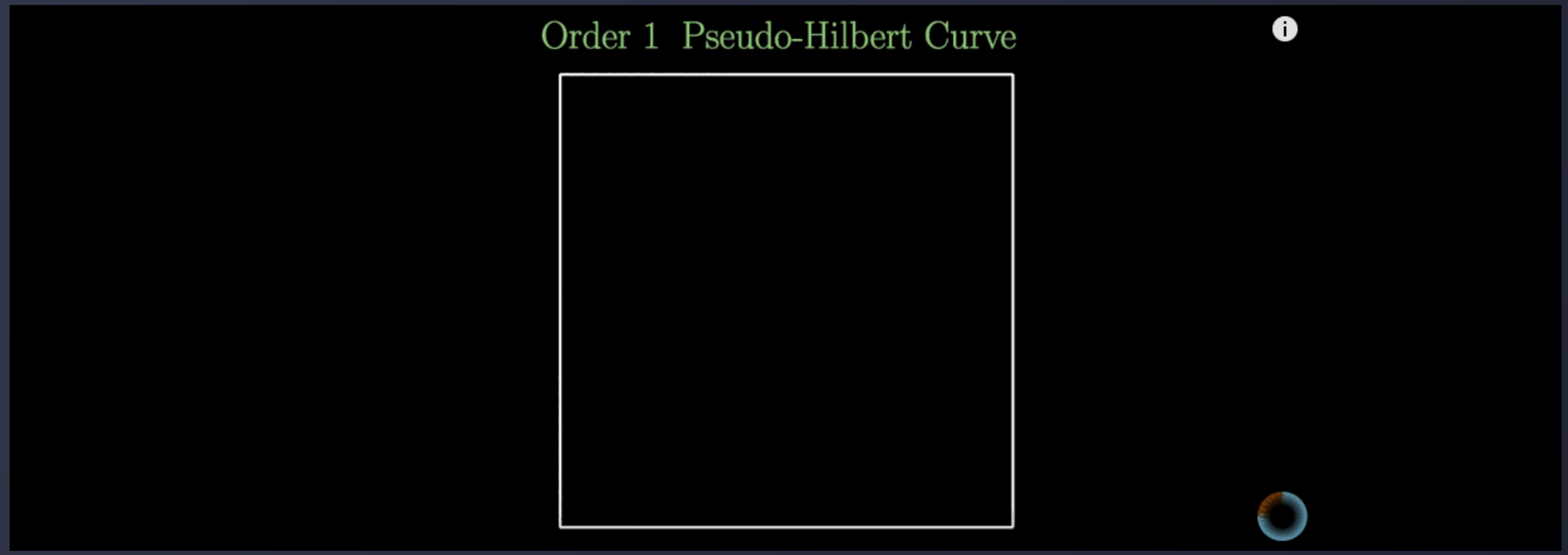
HAUSDORFF FRACTALS DIMENSION



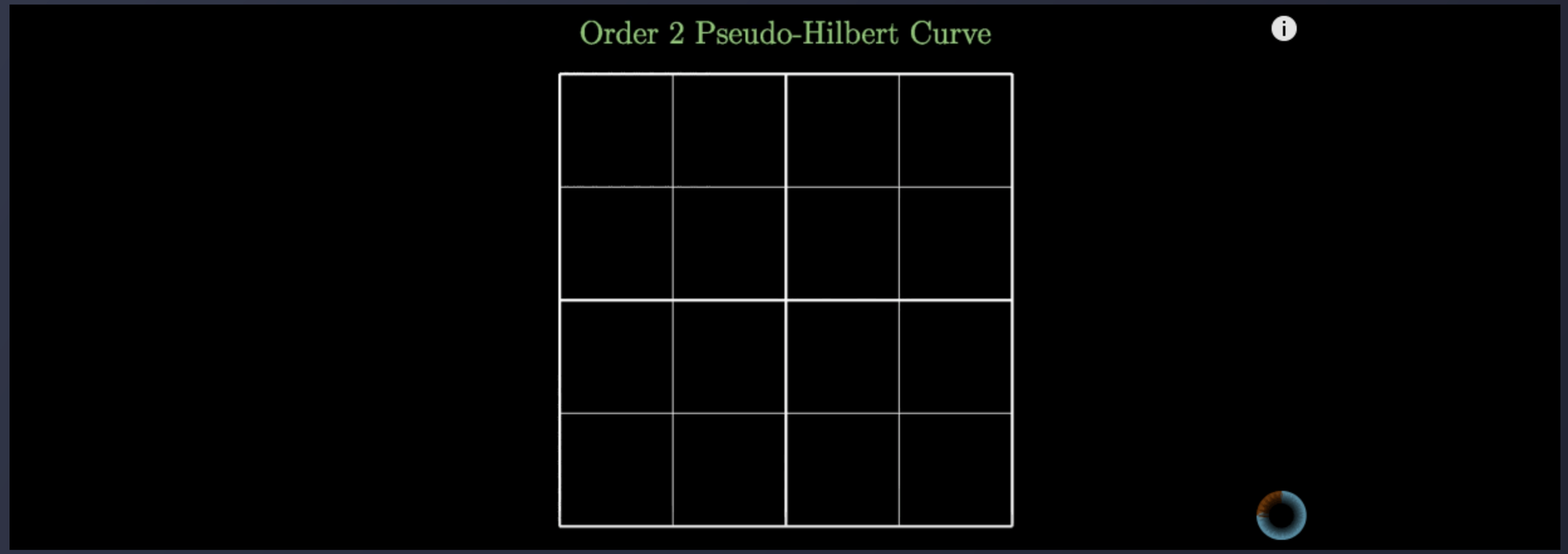
HILBERT CURVE



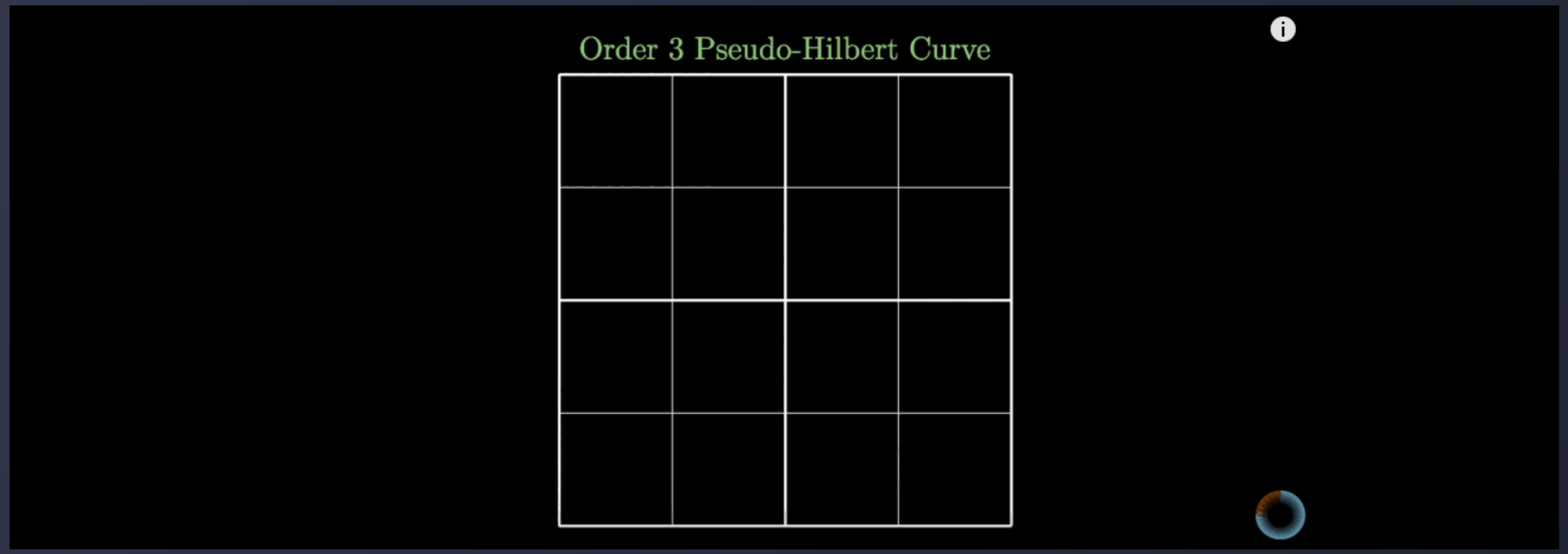
HILBERT CURVE



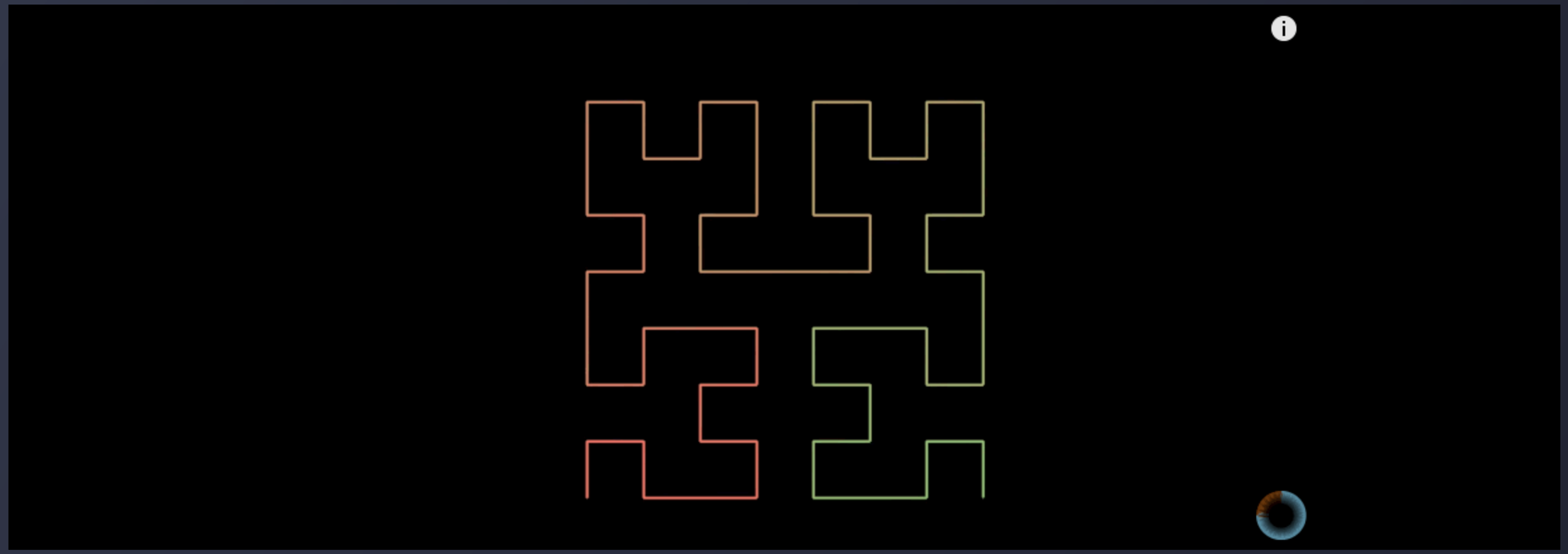
HILBERT CURVE



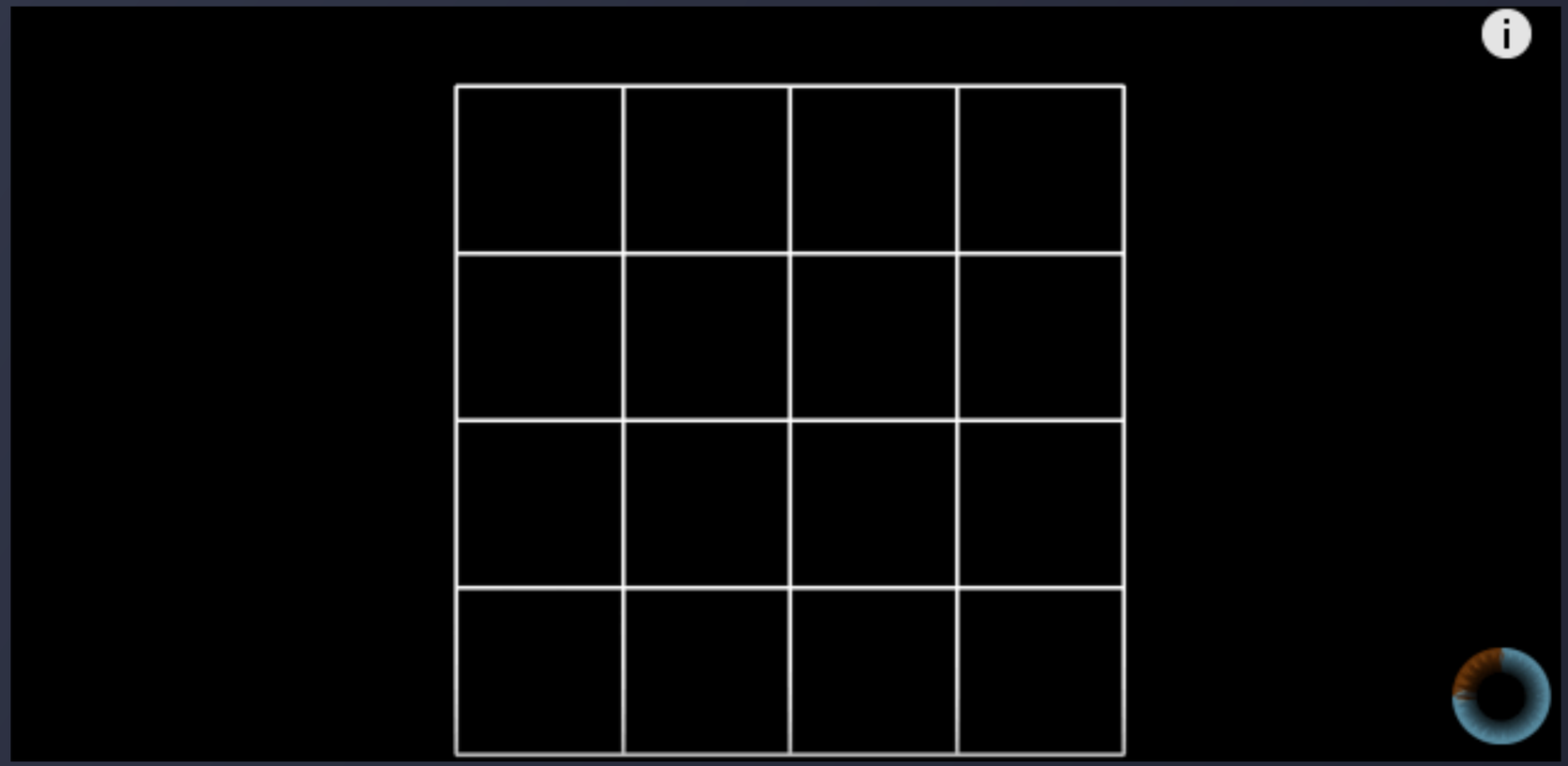
HILBERT CURVE



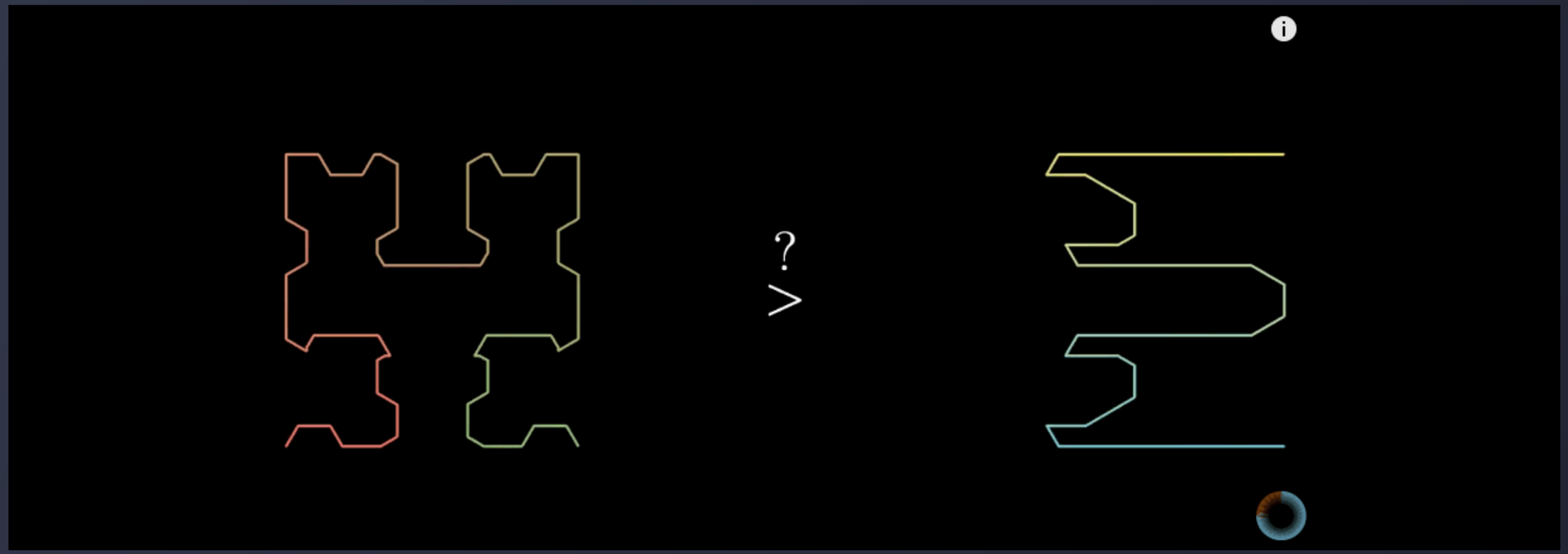
HILBERT CURVE



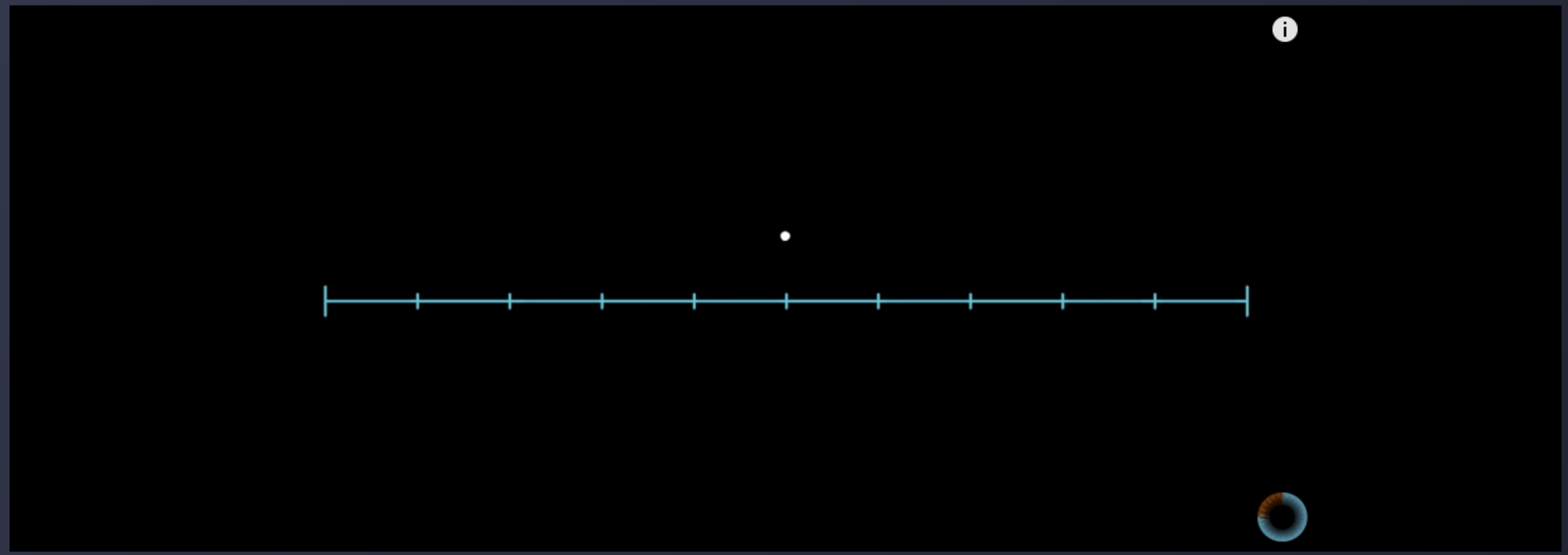
HILBERT CURVE



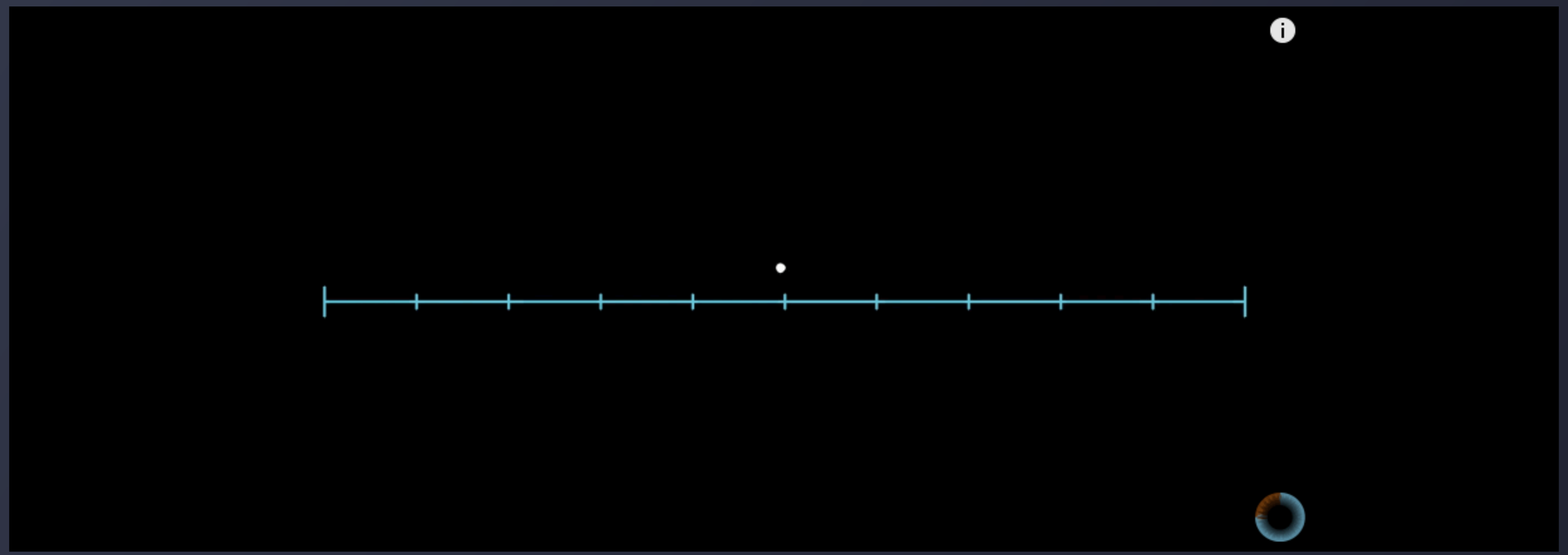
HILBERT CURVE



HILBERT CURVE



HILBERT CURVE



CHARACTERISTIC

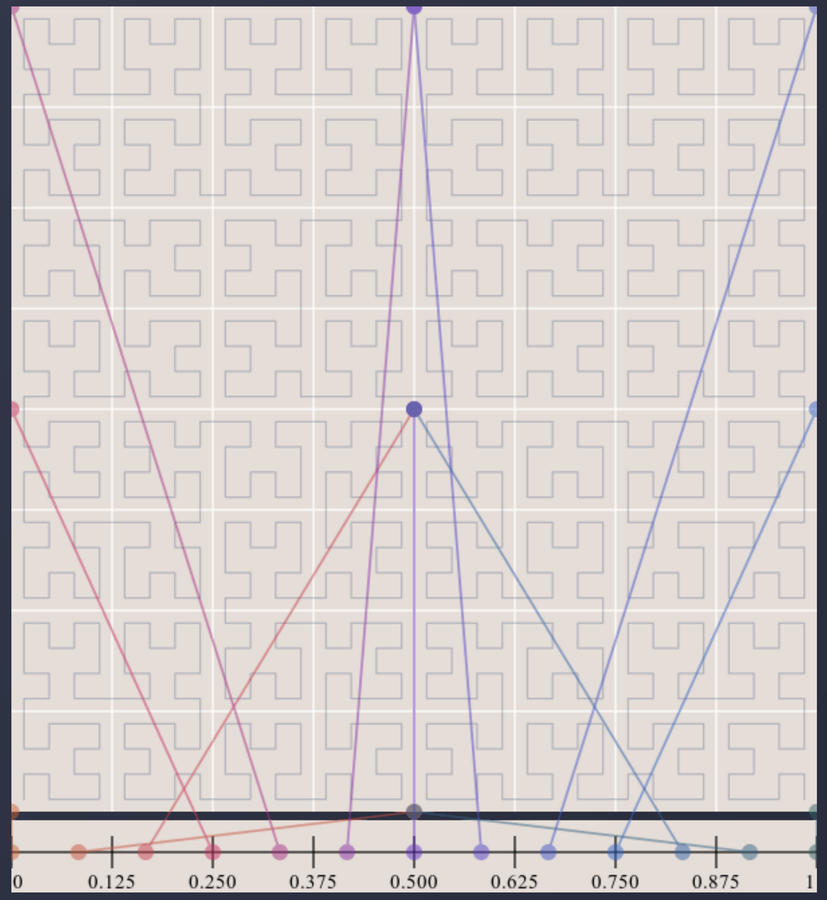
降维

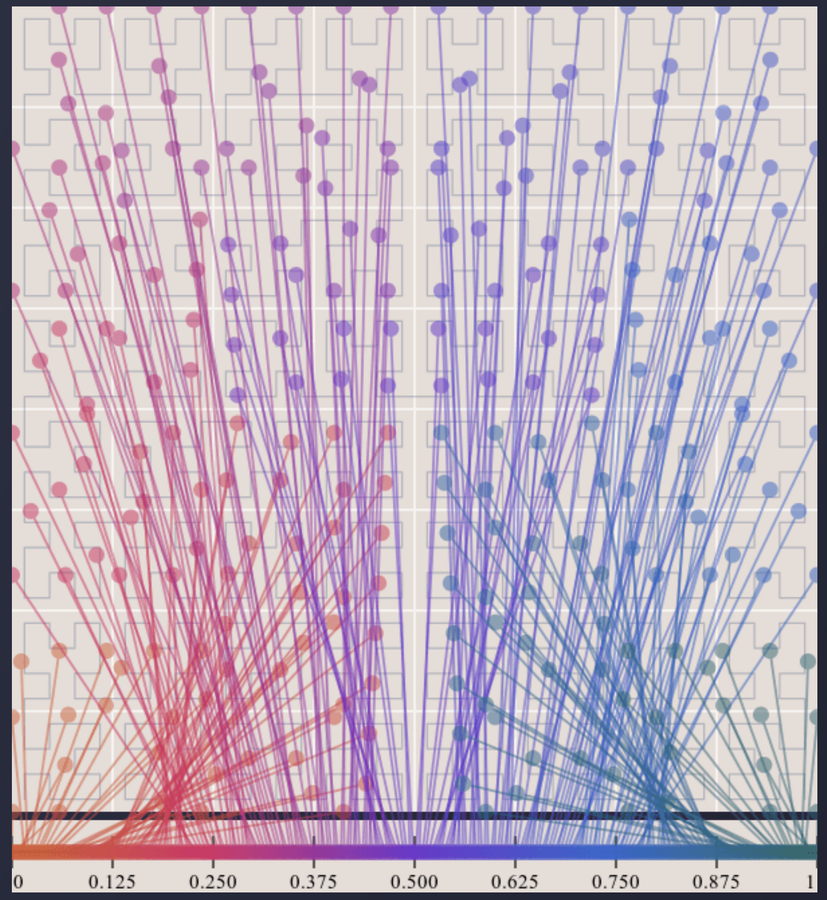
稳定

连续

希尔伯特曲线是连续的，所以能保证一定可以填满空间。连续性是需要数学证明的。具体证明方法这里就不细说了，感兴趣的可以点文章末尾一篇关于希尔伯特曲线的论文，那里有连续性的证明。

CONTINUITY


0
0.125
0.250
0.375
0.500
0.625
0.750
0.875
1


0
0.125
0.250
0.375
0.500
0.625
0.750
0.875
1

THANKS FOR YOUR ATTENTION!

ANY QUESTIONS?

NEXT SLIDE

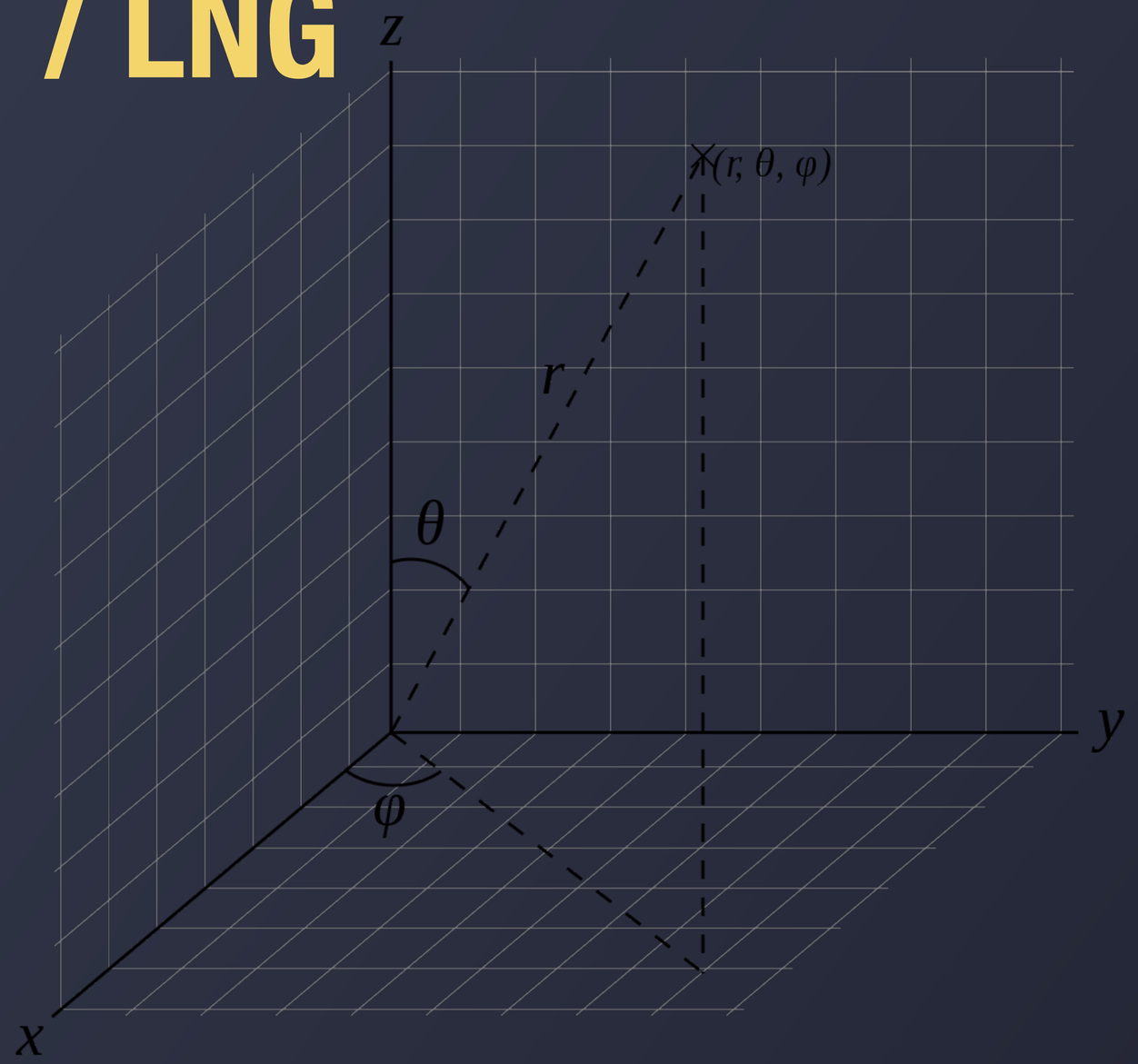
SPATIAL INDEX

GOOGLE S2

NEXT SLIDE

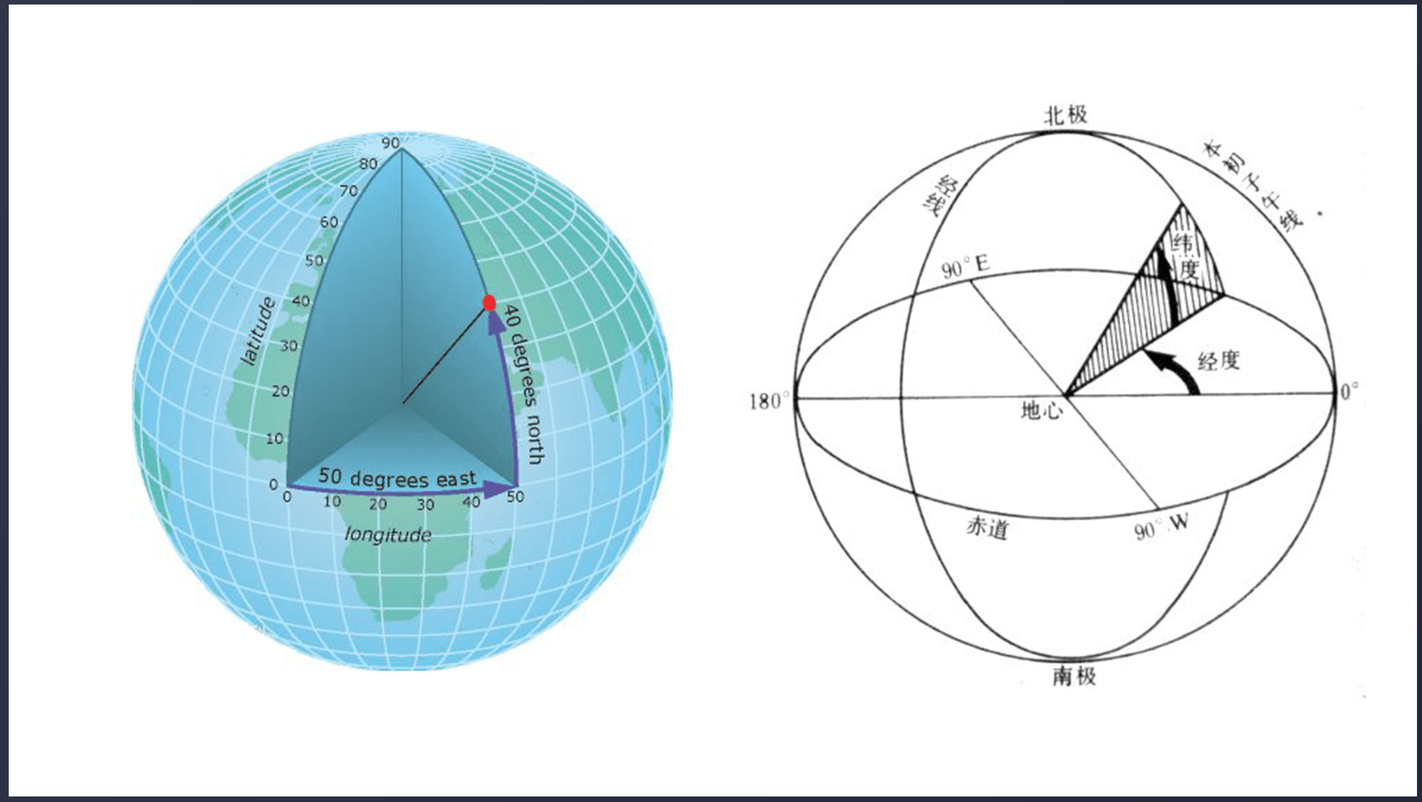
#03

LAT / LNG



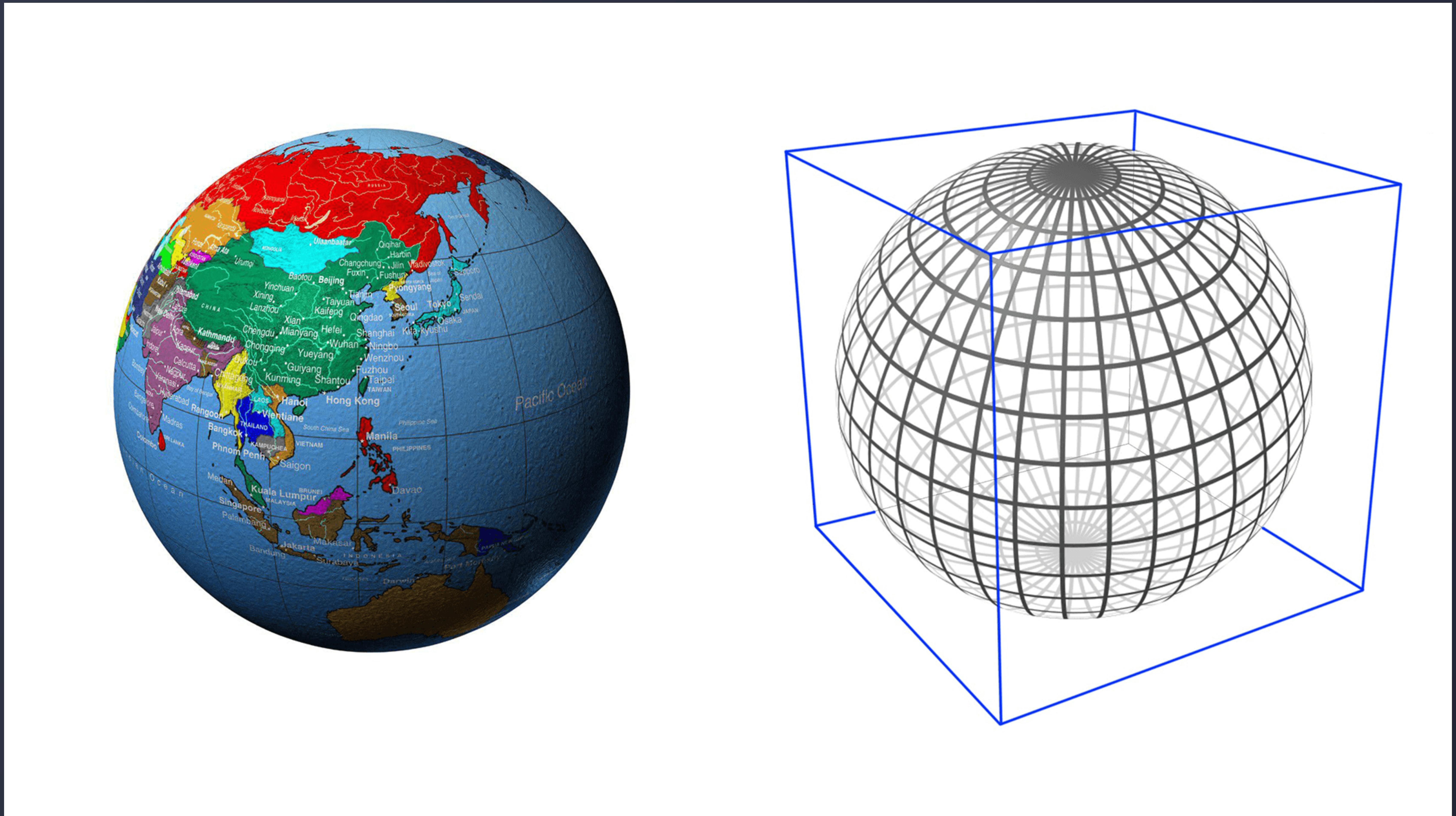
$$x = r * \sin\theta * \cos\phi$$
$$y = r * \sin\theta * \sin\phi$$
$$z = r * \cos\theta$$

LAT / LNG



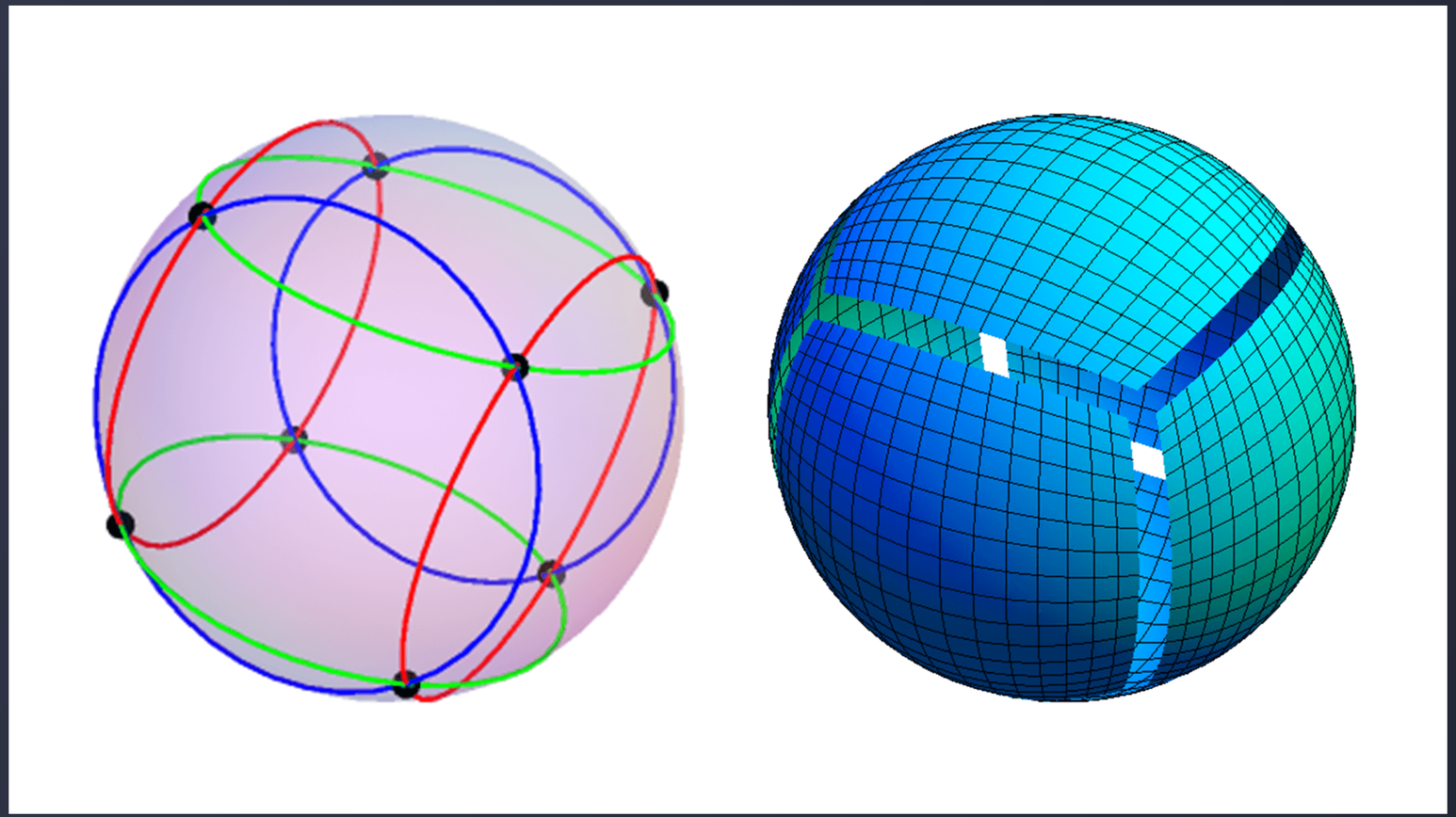
s(lat,lng) -> f(x,y,z)

PROJECTION

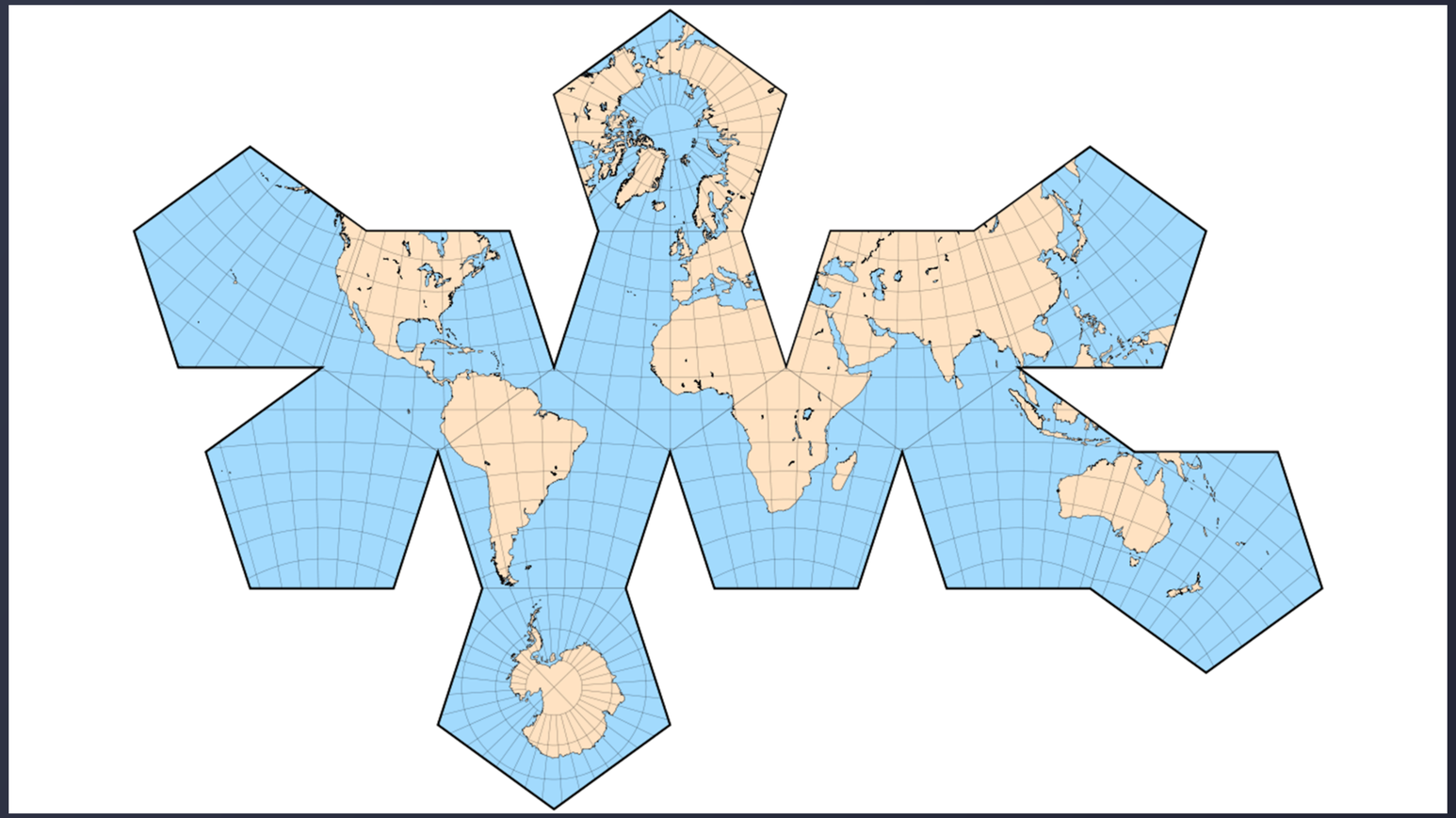


Pacific Ocean
Hong Kong
Manila
Bangkok
Hanoi
Vientiane
Rangoon
Phnom Penh
Saigon
Kuala Lumpur
Jakarta
Beijing
Seoul
Tokyo
Shanghai
Kathmandu

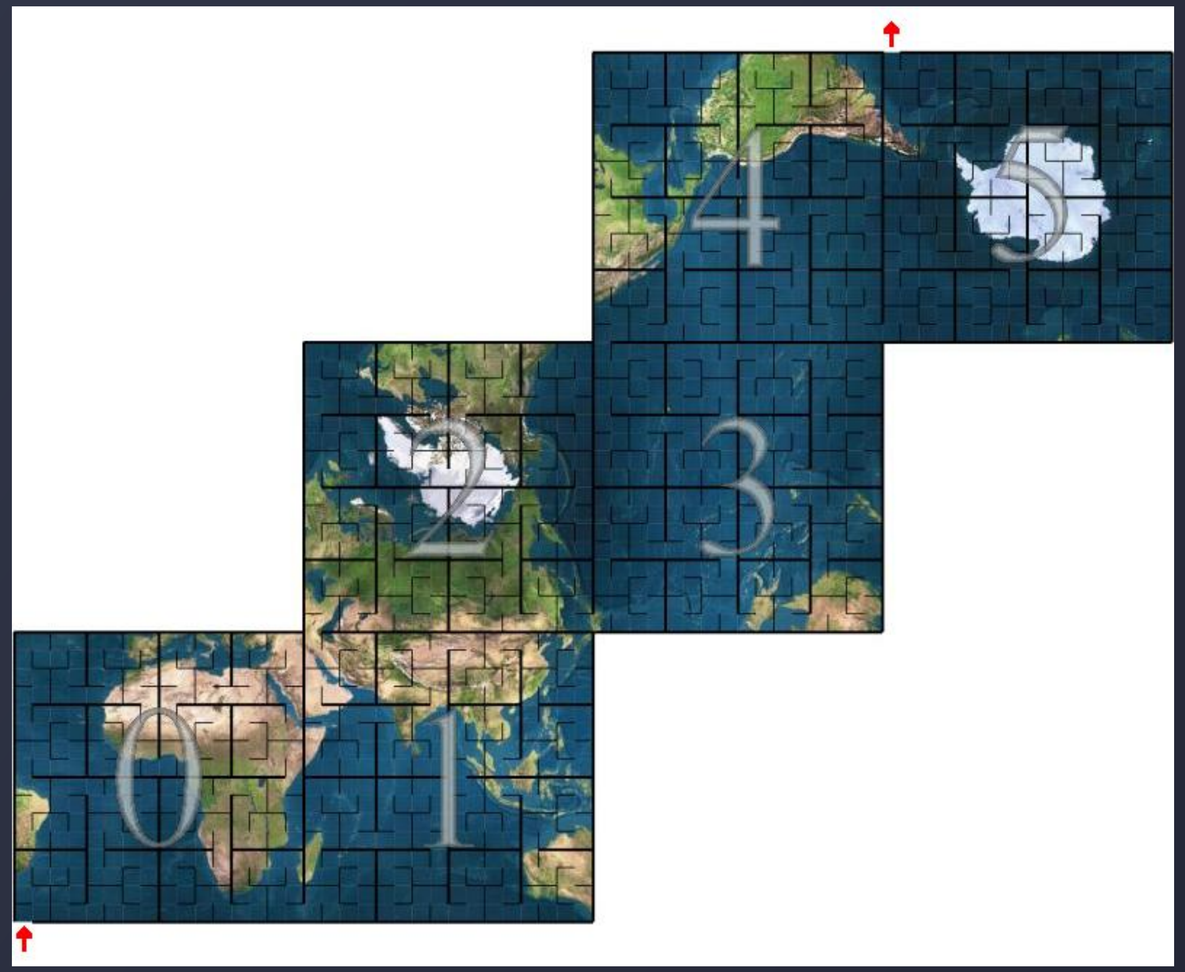
PROJECTION



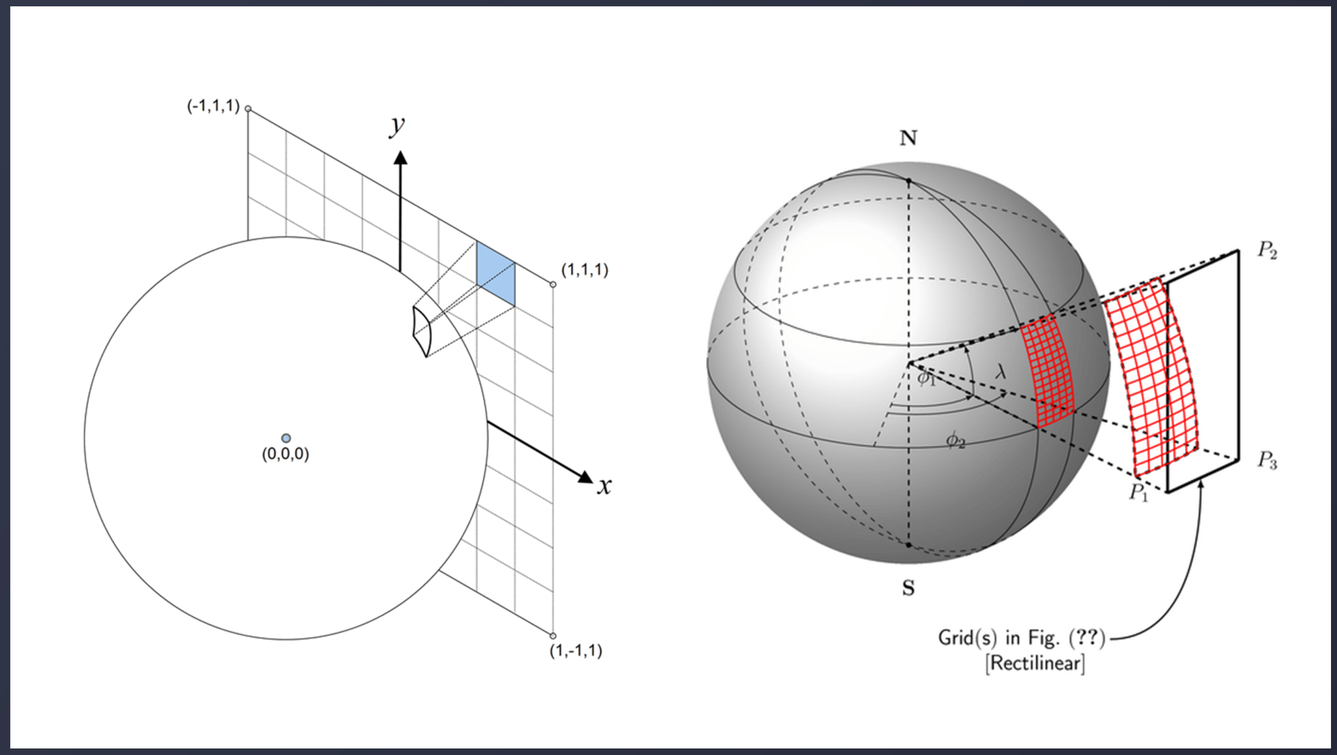
FRACTAL

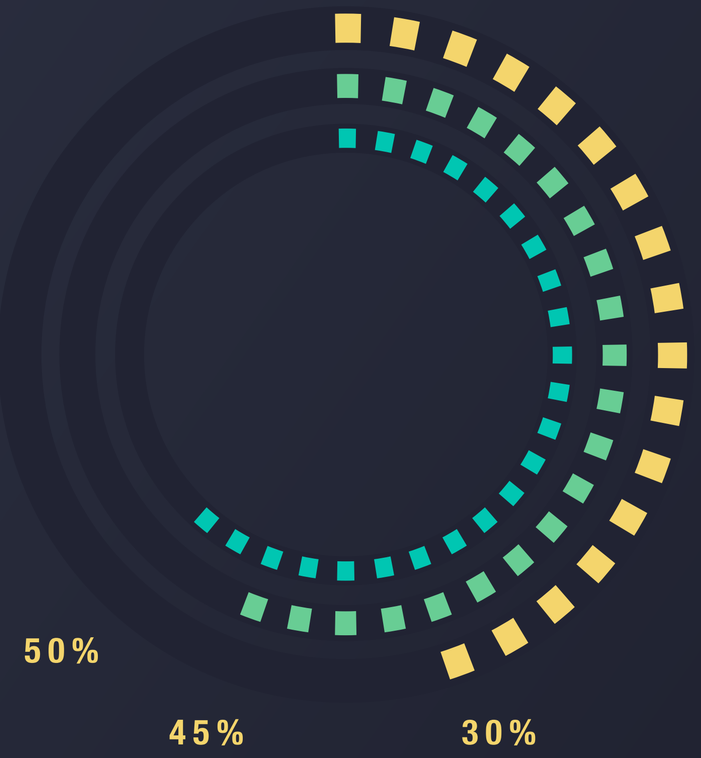


FRACTAL

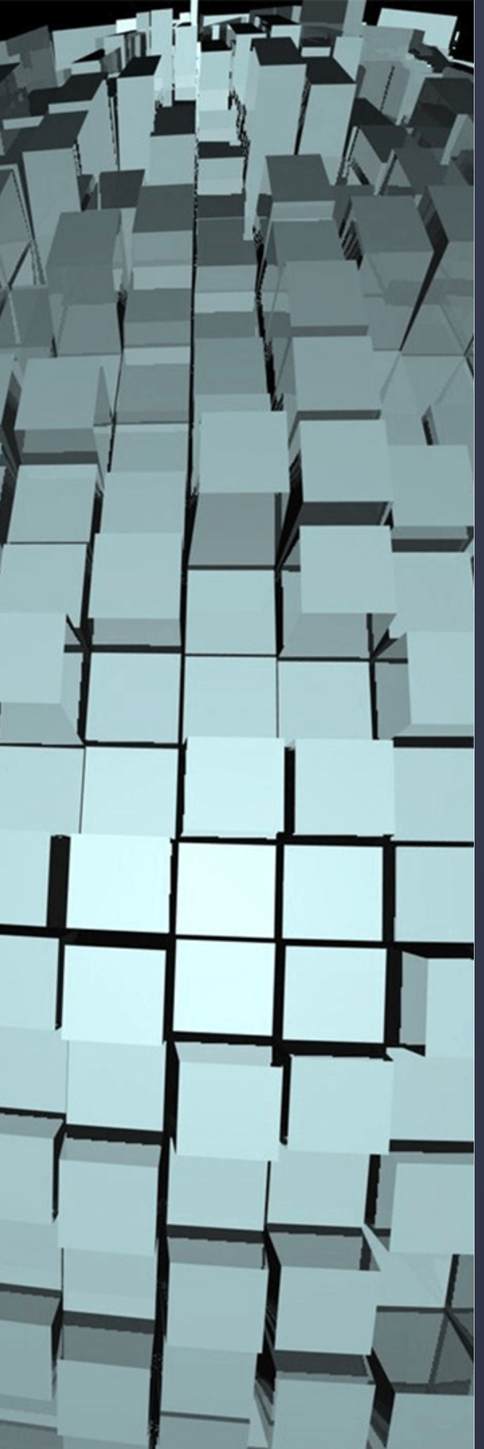


FRACTAL

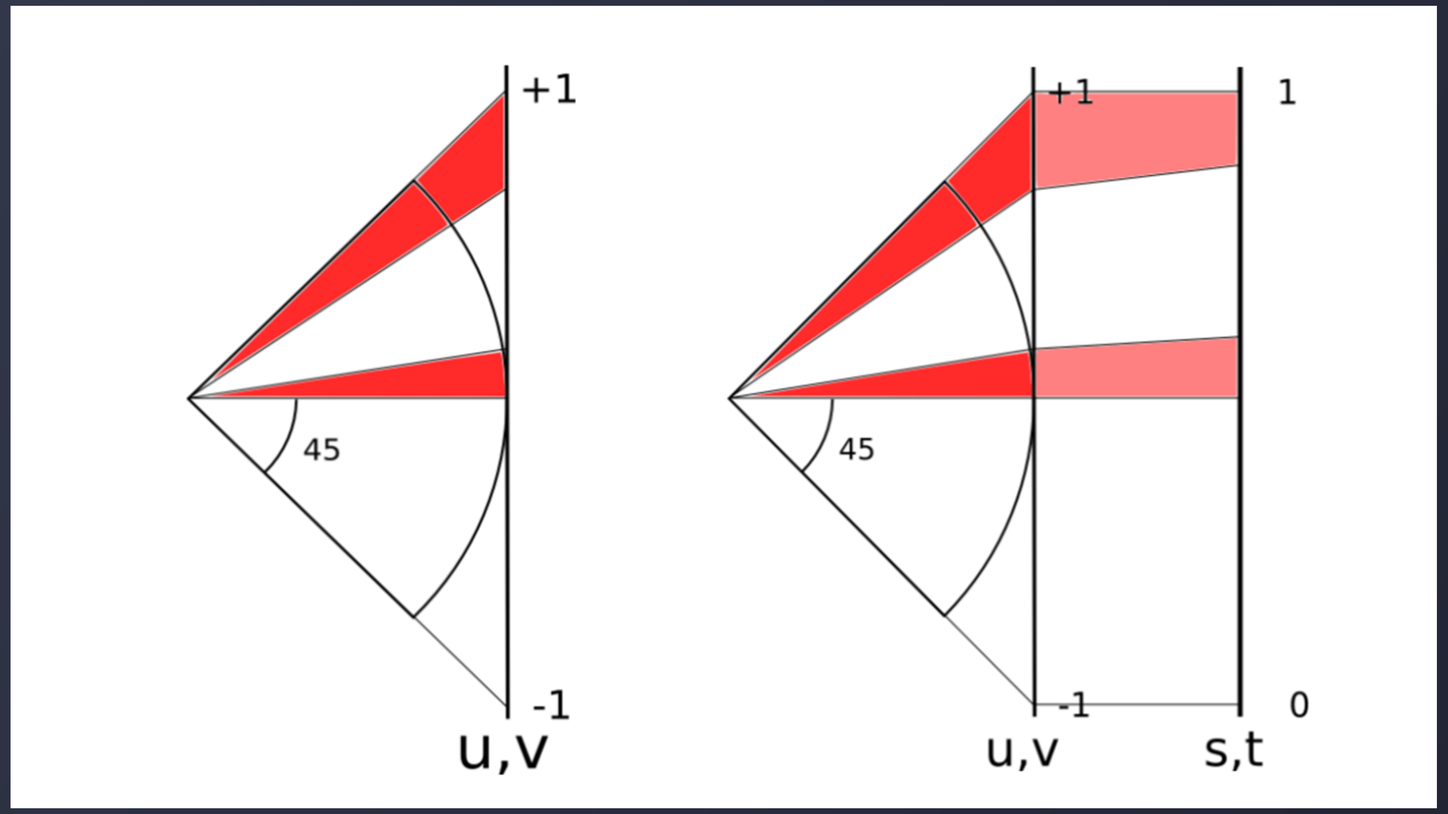




f(x,y,z) -> g(face,u,v)



FIXED



g(face,u,v) -> h(face,s,t)

PROGRAM

1

线性变换

	面积比率	边比率	对角线比率	ToPointRaw	ToPoint	FromPoint
线性变换	5.200	2.117	2.959	0.020	0.087	0.085
tan()变换	1.414	1.414	1.704	0.237	0.299	0.258
二次变换	2.082	1.802	1.932	0.033	0.096	0.108

2

tan() 三角变换

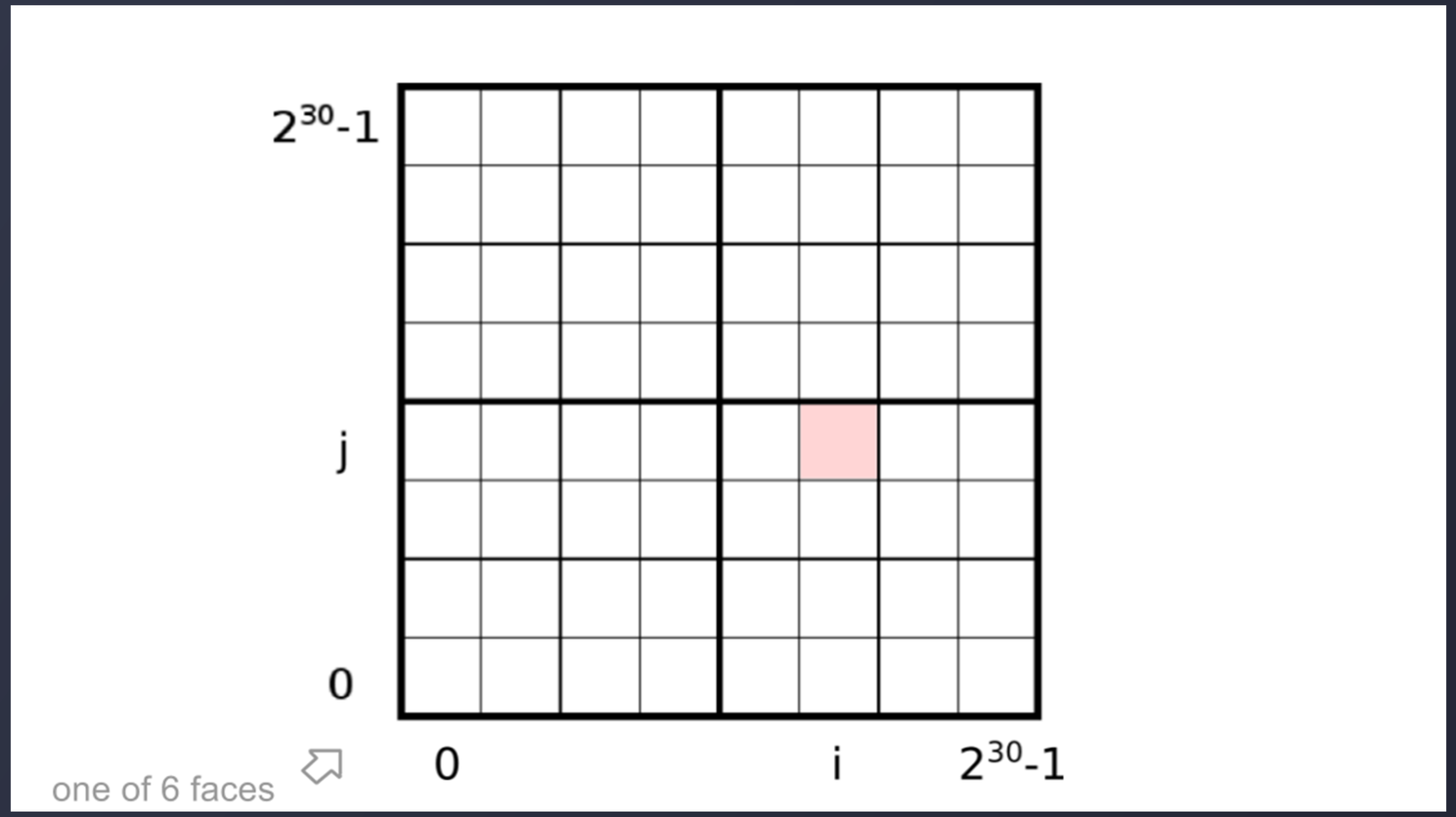
3

二次变换

u >= 0， u = 0.5 * sqrt(1 + 3*u)

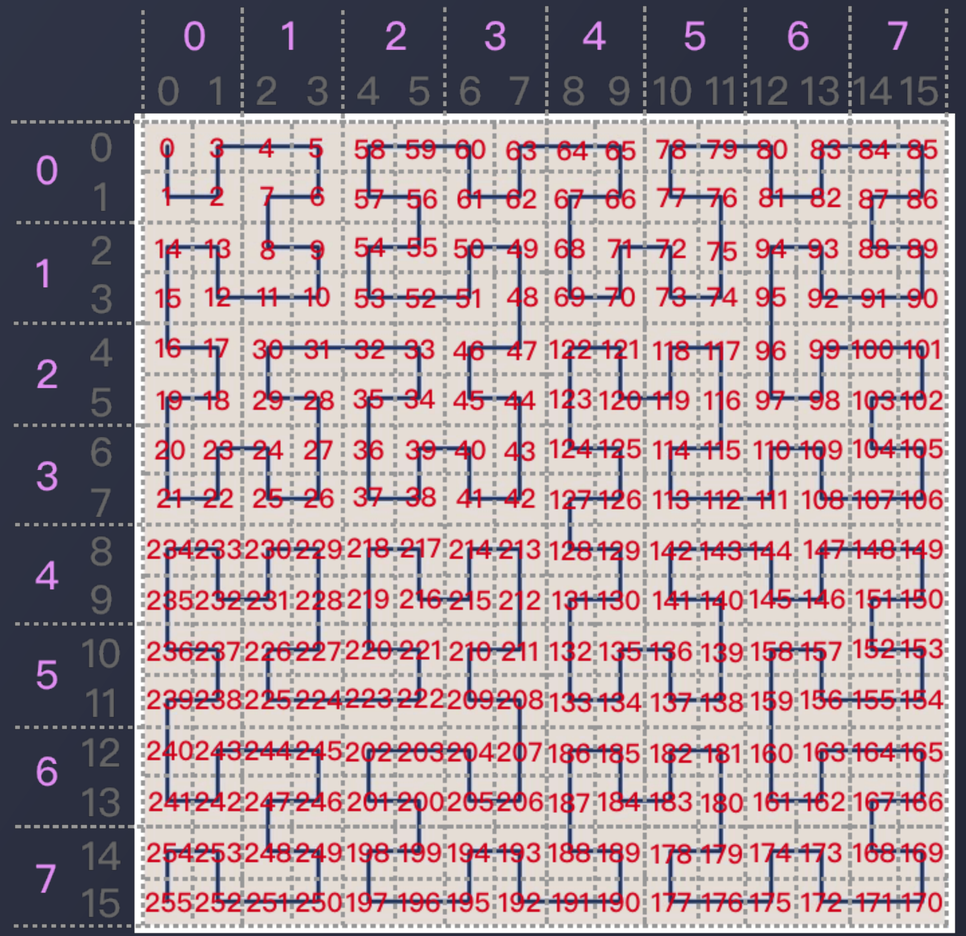
u < 0， u = 1 - 0.5 * sqrt(1 - 3*u)

TRANSFORM

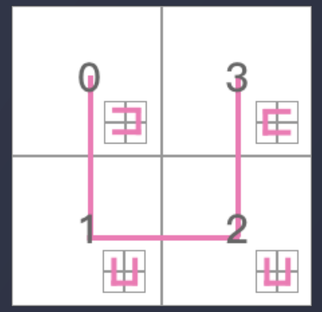


h(face,s,t) -> H(face,i,j)

HILBERT CURVE

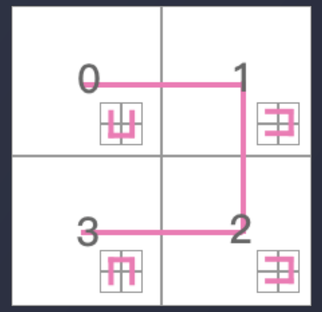


HILBERT CURVE



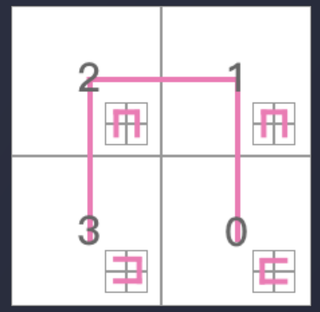
标准顺序

图0



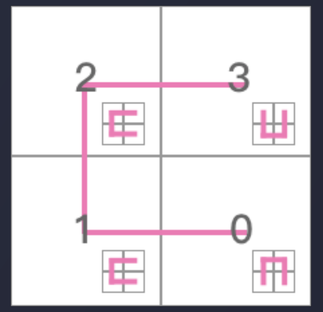
轴旋转

图1



上下倒置

图2

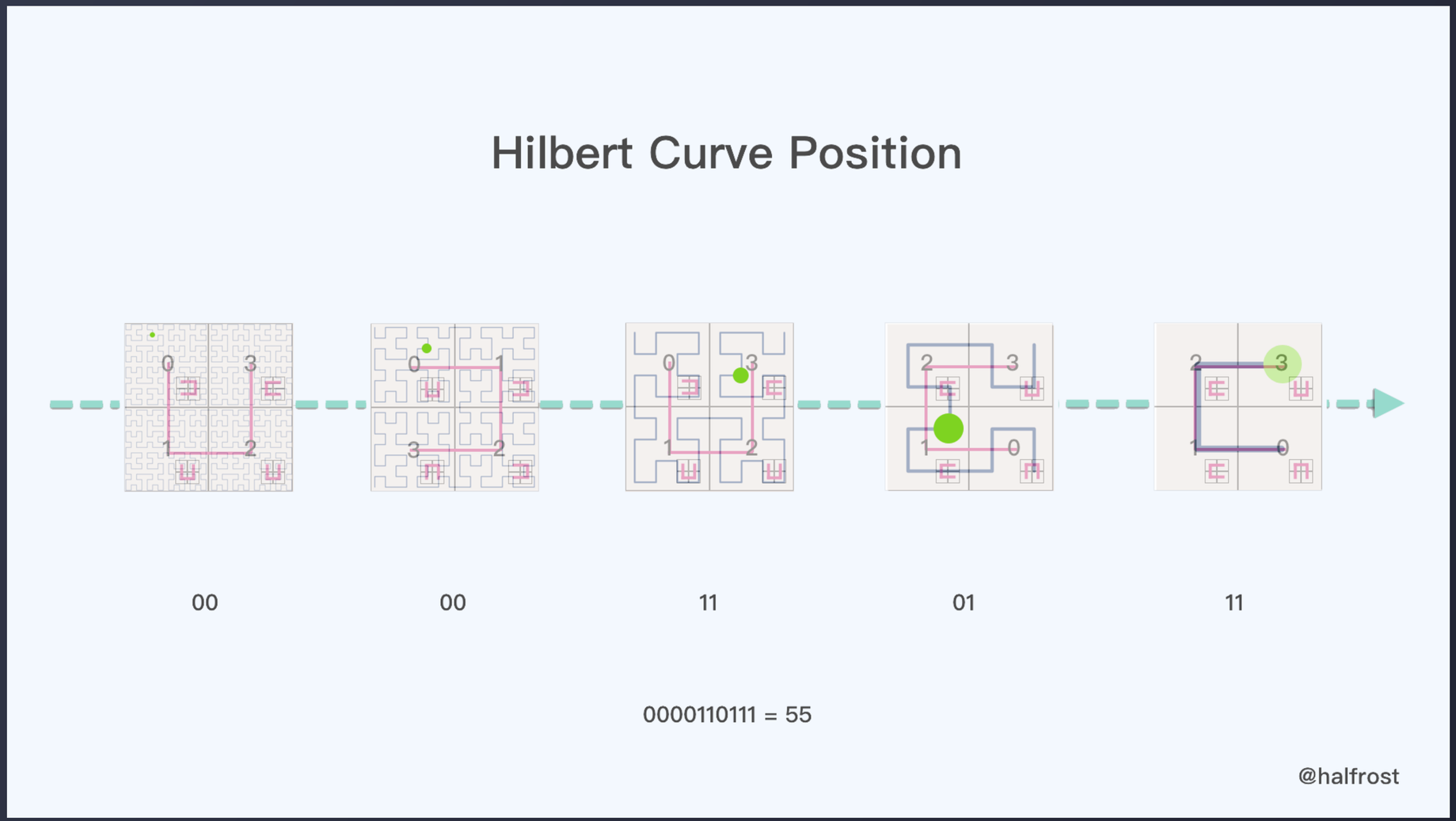


轴旋转左右倒置

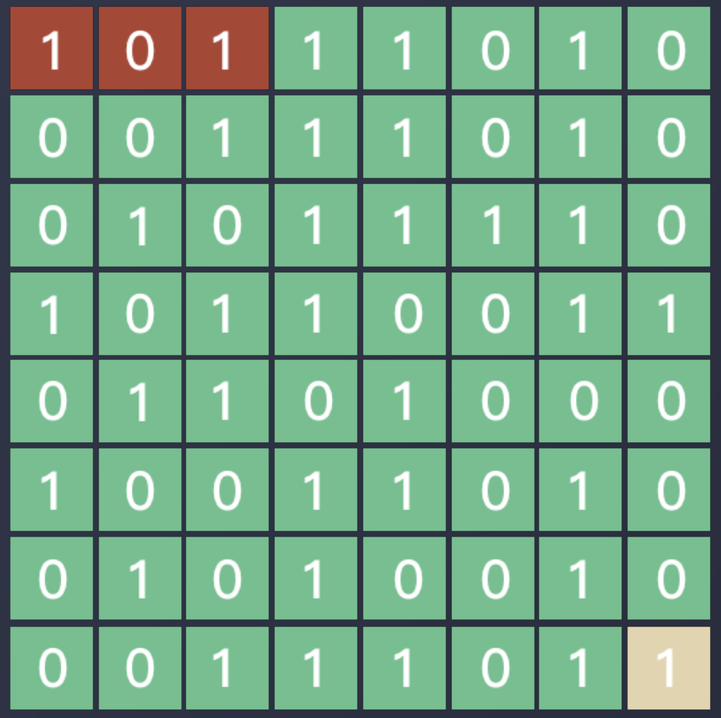
图3

@halfrost

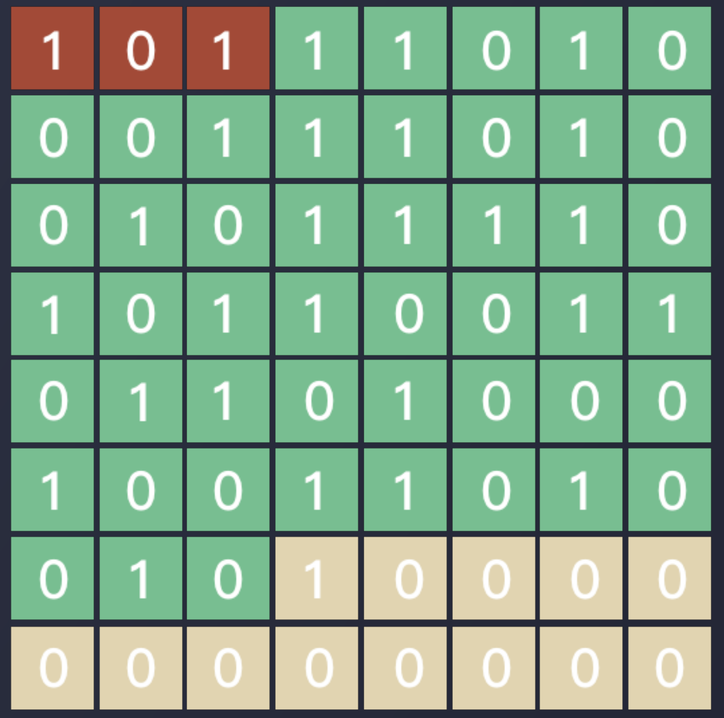
HILBERT CURVE



HILBERT CURVE LEVEL



[0,2^30−1] * [0,2^30−1]

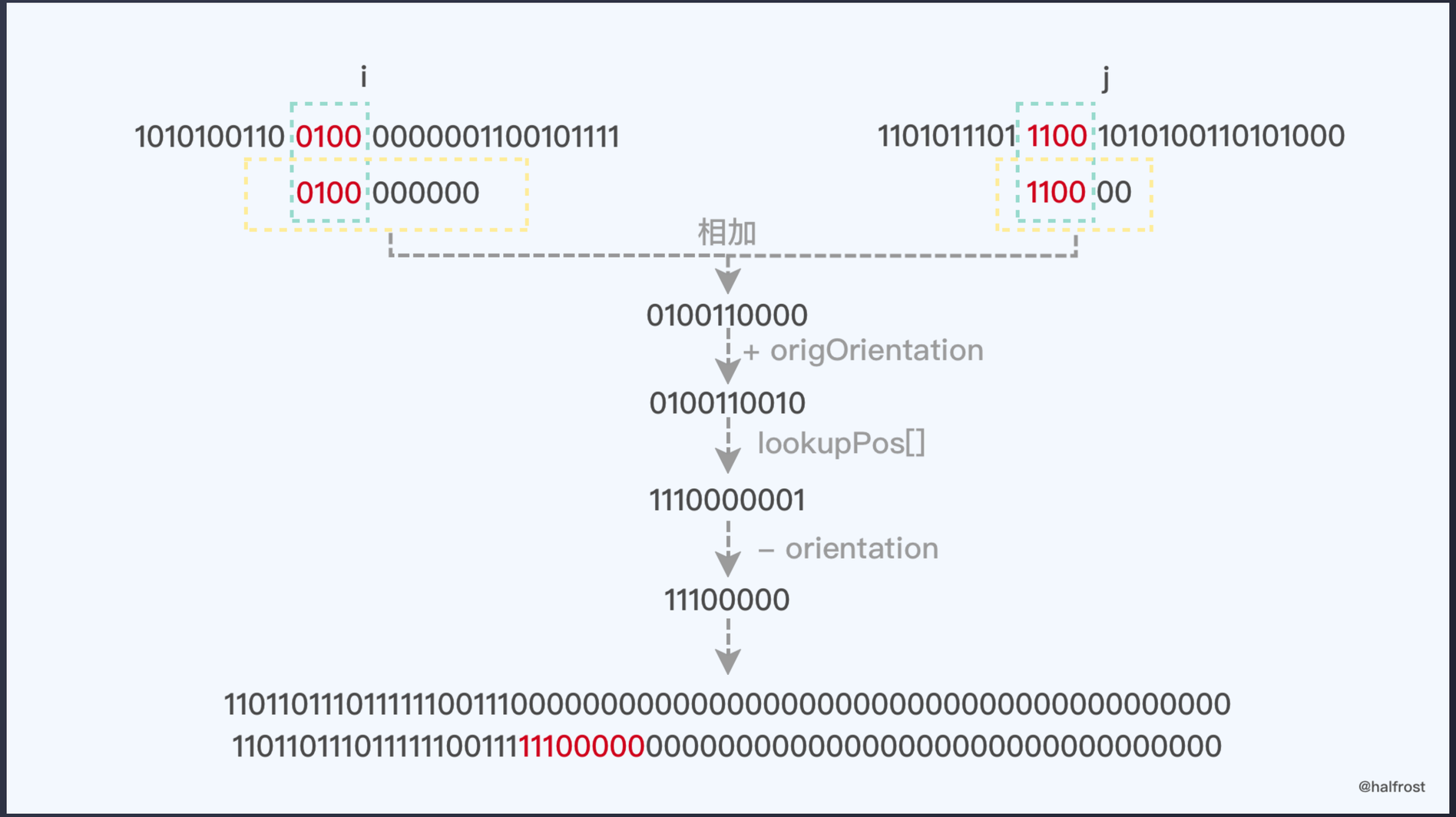


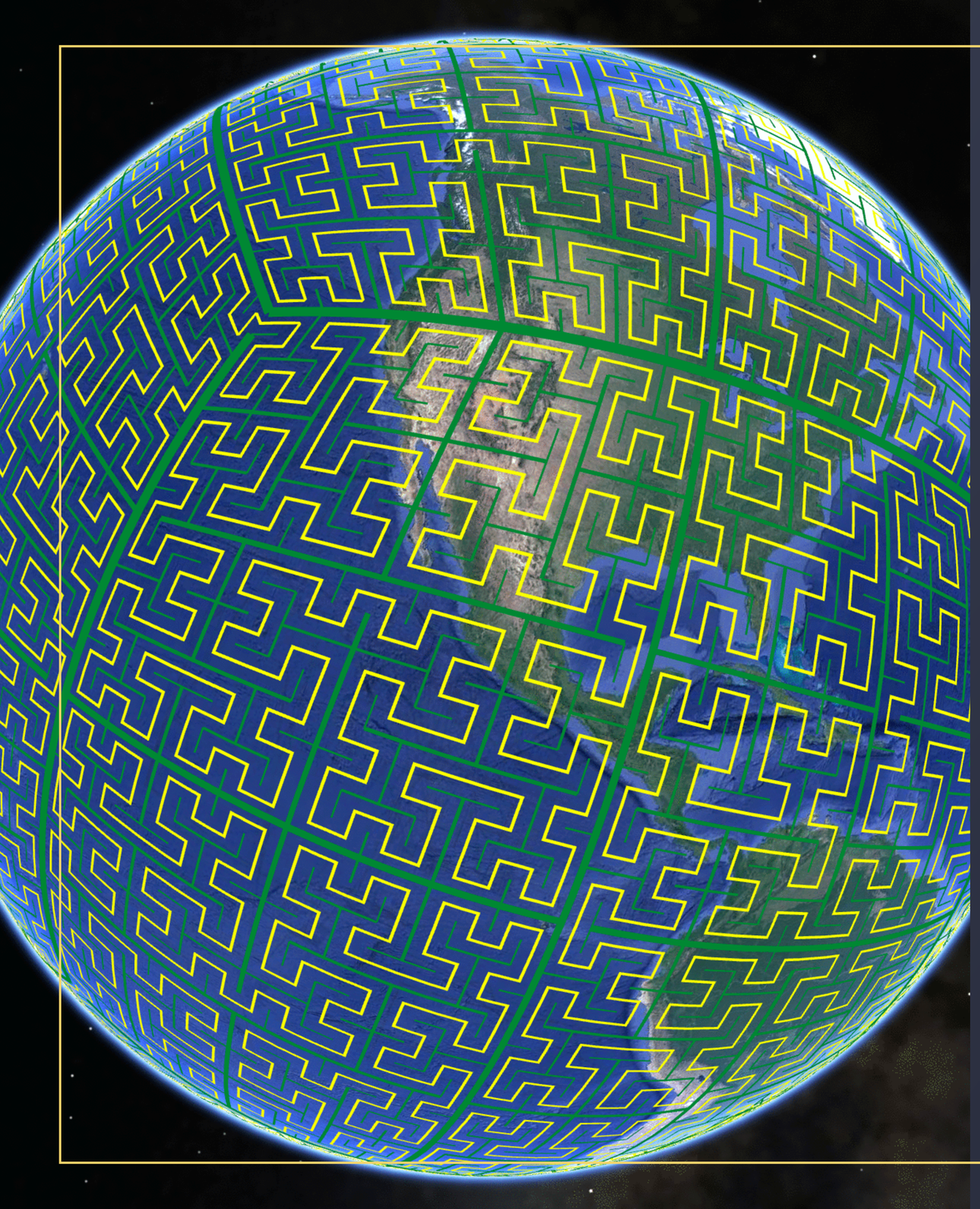
[0,2^24−1] * [0,2^24−1]

H(face,i,j) -> CellID

@halfrost

CELL ID


i
1010100110 0100 0000001100101111
0100 000000
j
1101011101 1100 1010100110101000
1100 00
相加
0100110000
+ origOrientation
0100110010
lookupPos[]
1110000001
– orientation
11100000
11011011101111110011100000000000000000000000000000000000000000
11011011101111110011111100000000000000000000000000000000000000
@halfrost



GOOGLE S2

S(lat,lng) -> f(x,y,z) -> g(face,u,v) ->
h(face,s,t) -> H(face,i,j) -> CellID

NEXT SLIDE

level	min area	max area	average area	units	Random cell 1 (UK) min edge length	Random cell 1 (UK) max edge length	Random cell 2 (US) min edge length	Random cell 2 (US) max edge length	Number of cells
00	85011012.19	85011012.19	85011012.19	km^2	7842 km	7842 km	7842 km	7842 km	6
01	21252753.05	21252753.05	21252753.05	km^2	3921 km	5004 km	3921 km	5004 km	24
02	4919708.23	6026521.16	5313188.26	km^2	1825 km	2489 km	1825 km	2489 km	96
03	1055377.48	1646455.50	1328297.07	km^2	840 km	1167 km	1130 km	1310 km	384
04	231564.06	413918.15	332074.27	km^2	432 km	609 km	579 km	636 km	1536
05	53798.67	104297.91	83018.57	km^2	210 km	298 km	287 km	315 km	6K
06	12948.81	26113.30	20754.64	km^2	108 km	151 km	143 km	156 km	24K
07	3175.44	6529.09	5188.66	km^2	54 km	76 km	72 km	78 km	98K
08	786.20	1632.45	1297.17	km^2	27 km	38 km	36 km	39 km	393K
09	195.59	408.12	324.29	km^2	14 km	19 km	18 km	20 km	1573K
10	48.78	102.03	81.07	km^2	7 km	9 km	9 km	10 km	6M
11	12.18	25.51	20.27	km^2	3 km	5 km	4 km	5 km	25M
12	3.04	6.38	5.07	km^2	1699 m	2 km	2 km	2 km	100M
13	0.76	1.59	1.27	km^2	850 m	1185 m	1123 m	1225 m	402M
14	0.19	0.40	0.32	km^2	425 m	593 m	562 m	613 m	1610M
15	47520.30	99638.93	79172.67	m^2	212 m	296 m	281 m	306 m	6B
16	11880.08	24909.73	19793.17	m^2	106 m	148 m	140 m	153 m	25B
17	2970.02	6227.43	4948.29	m^2	53 m	74 m	70 m	77 m	103B
18	742.50	1556.86	1237.07	m^2	27 m	37 m	35 m	38 m	412B
19	185.63	389.21	309.27	m^2	13 m	19 m	18 m	19 m	1649B
20	46.41	97.30	77.32	m^2	7 m	9 m	9 m	10 m	7T
21	11.60	24.33	19.33	m^2	3 m	5 m	4 m	5 m	26T
22	2.90	6.08	4.83	m^2	166 cm	2 m	2 m	2 m	105T
23	0.73	1.52	1.21	m^2	83 cm	116 cm	110 cm	120 cm	422T
24	0.18	0.38	0.30	m^2	41 cm	58 cm	55 cm	60 cm	1689T
25	453.19	950.23	755.05	cm^2	21 cm	29 cm	27 cm	30 cm	7e15
26	113.30	237.56	188.76	cm^2	10 cm	14 cm	14 cm	15 cm	27e15
27	28.32	59.39	47.19	cm^2	5 cm	7 cm	7 cm	7 cm	108e15
28	7.08	14.85	11.80	cm^2	2 cm	4 cm	3 cm	4 cm	432e15
29	1.77	3.71	2.95	cm^2	12 mm	18 mm	17 mm	18 mm	1729e15
30	0.44	0.93	0.74	cm^2	6 mm	9 mm	8 mm	9 mm	7e18

GEOHASH VS GOOGLE S2

LEVEL 精细度

Geohash 有12级，从5000km 到 3.7cm。中间每一级的变化比较大。有时候可能选择上一级会大很多，选择下一级又会小一些。比如选择字符串长度为4，它对应的 cell 宽度是39.1km，需求可能是50km，那么选择字符串长度为5，对应的 cell 宽度就变成了156km，瞬间又大了3倍了。Geohash 需要 12 bytes 存储

突变性

S2 有30级，从 0.7cm² 到 85,000,000km² 。S2 的存储只需要一个 uint64 即可存下

几何计算

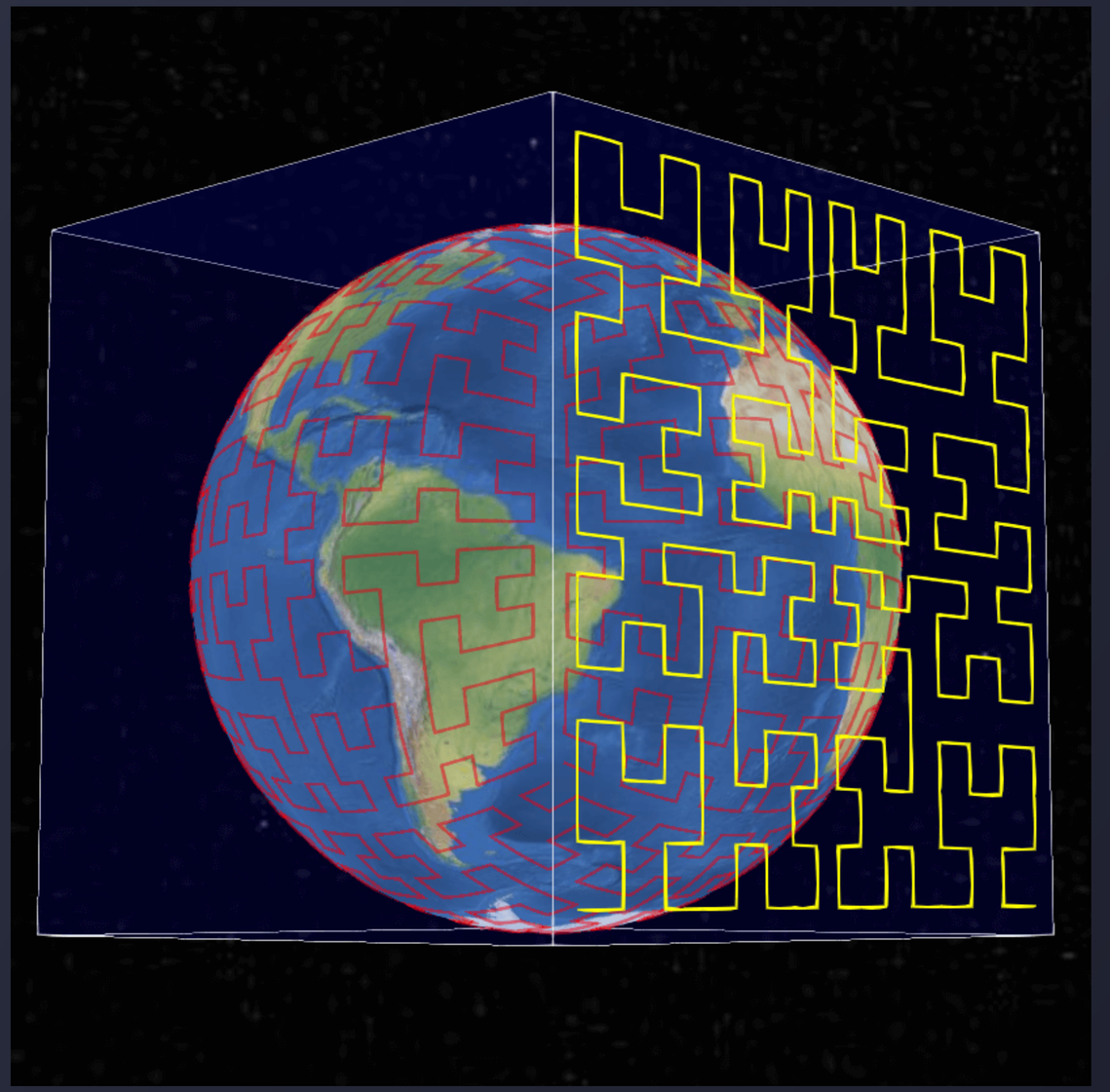
各种向量计算，面积计算，多边形覆盖，距离问题，球面球体上的问题

多边形覆盖

S2 还能解决多边形覆盖的问题

GOOGLE S2

1. 涉及到角度，间隔，纬度经度点，单位矢量等的表示，以及对这些类型的各种操作。
2. 单位球体上的几何形状，如球冠（“圆盘”），纬度 - 经度矩形，折线和多边形。
3. 支持点，折线和多边形的任意集合的强大的构造操作（例如联合）和布尔谓词（例如，包含）。
4. 对点，折线和多边形的集合进行快速的内存索引。
5. 针对测量距离和查找附近物体的算法。
6. 用于捕捉和简化几何的稳健算法（该算法具有精度和拓扑保证）。
7. 用于测试几何对象之间关系的有效且精确的数学谓词的集合。
8. 支持空间索引，包括将区域近似为离散“S2单元”的集合。此功能可以轻松构建大型分布式空间索引。



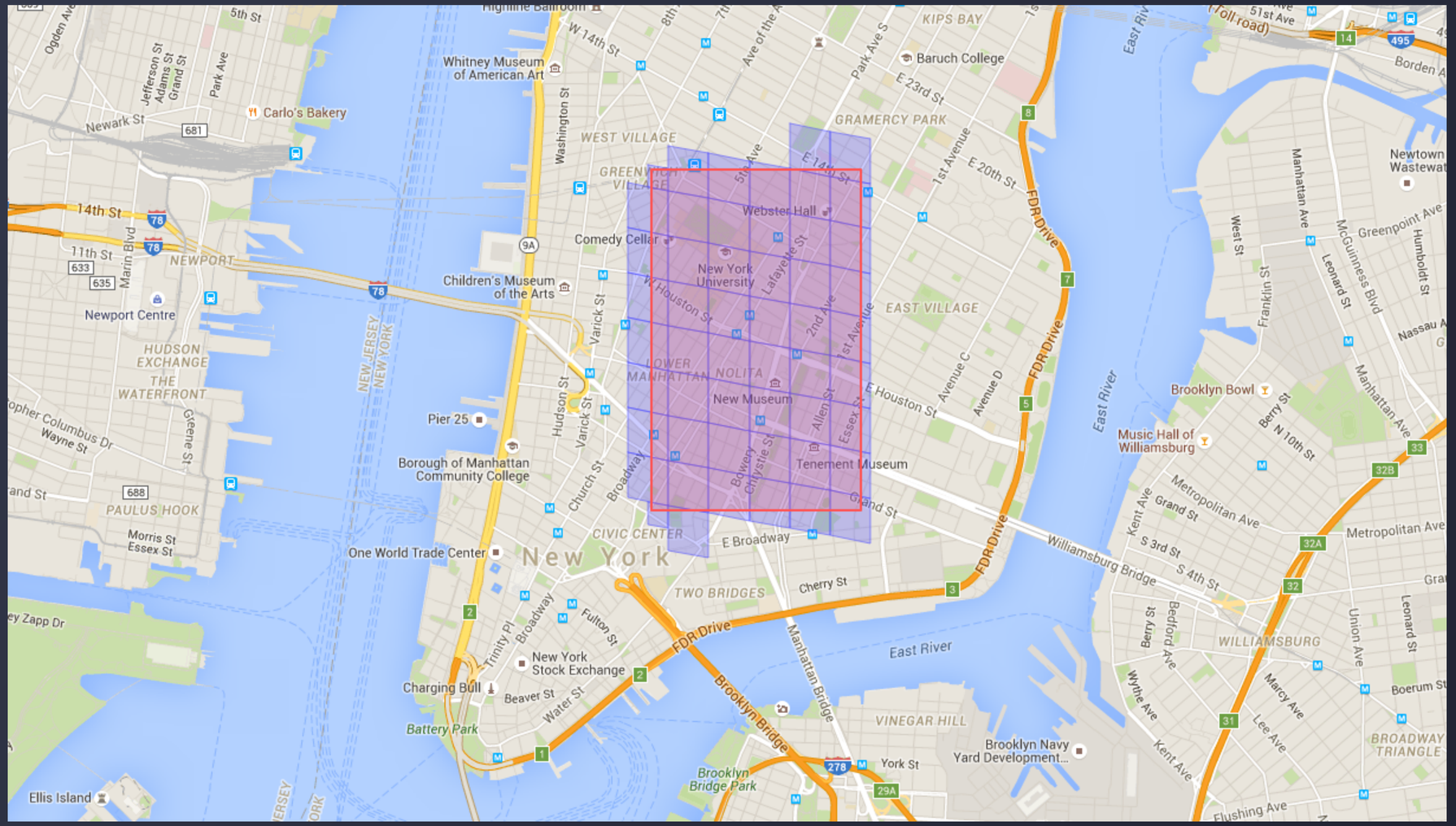
FINAL

APPLICATION

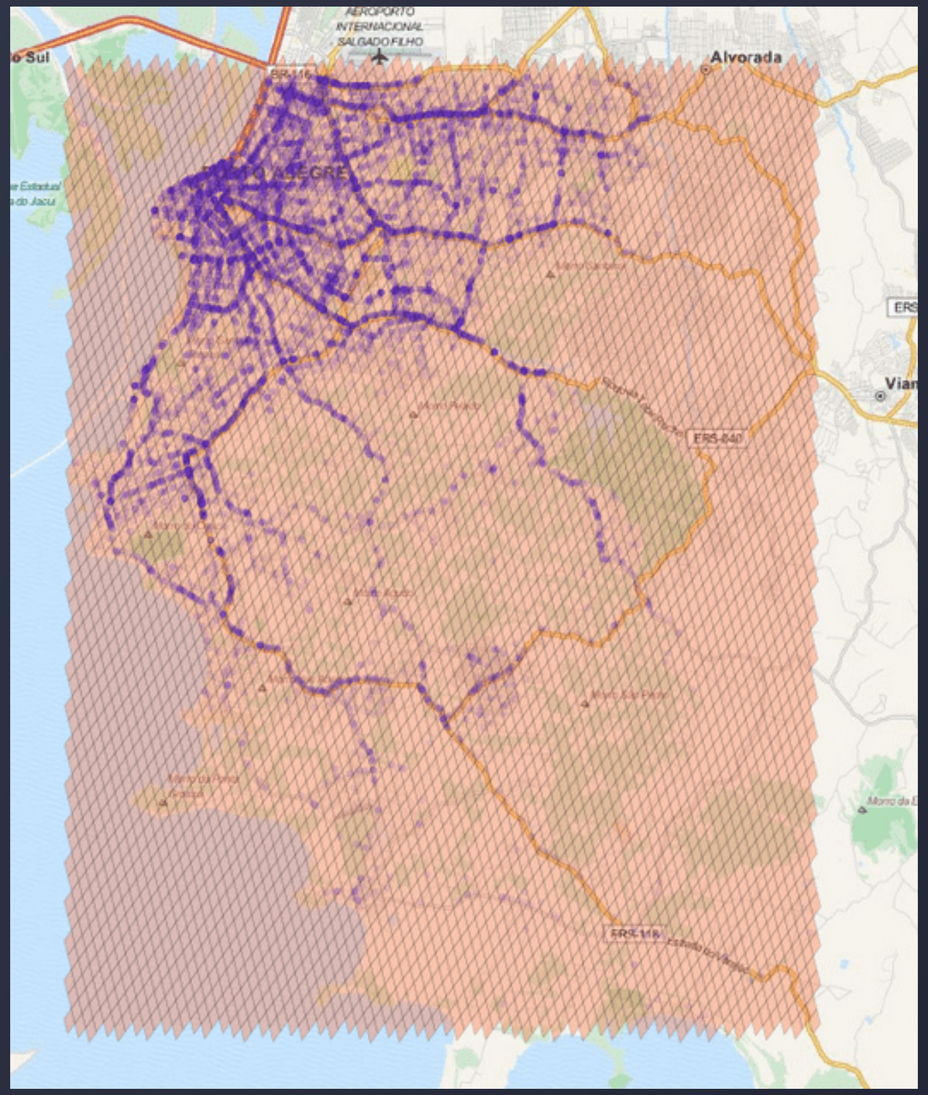
NEXT SLIDE

#04

APPLICATION



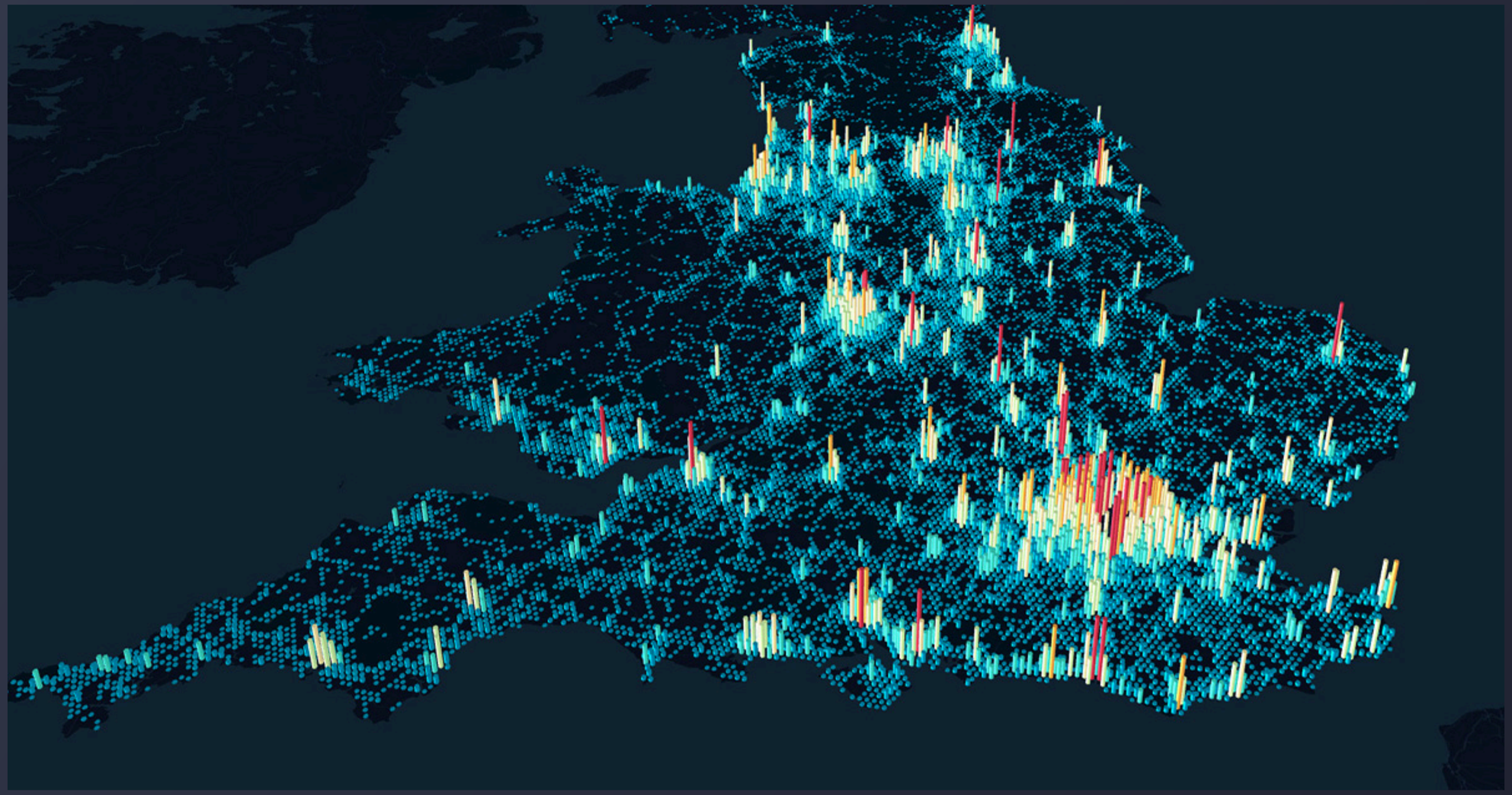
APPLICATION



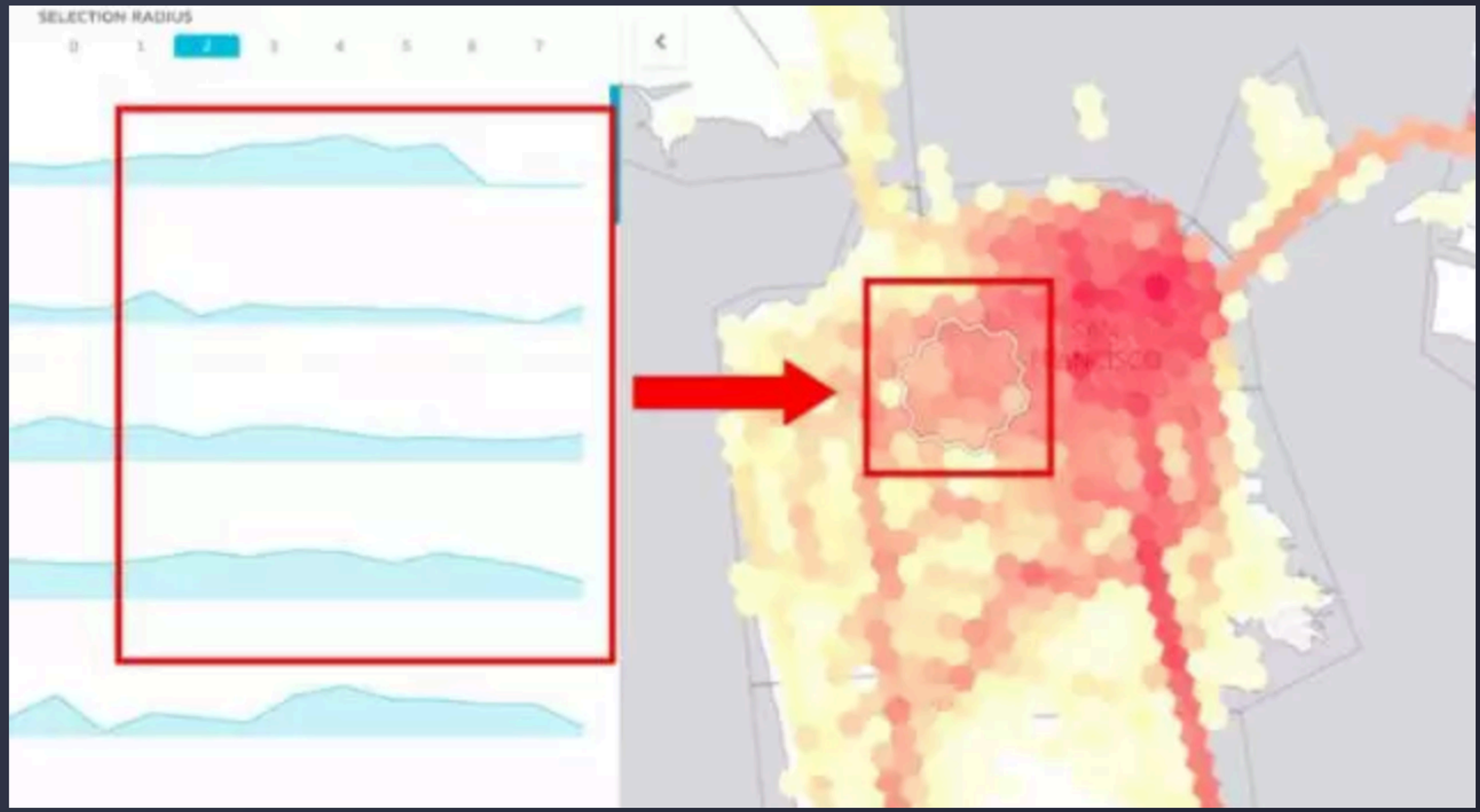
APPLICATION

APPLICATION

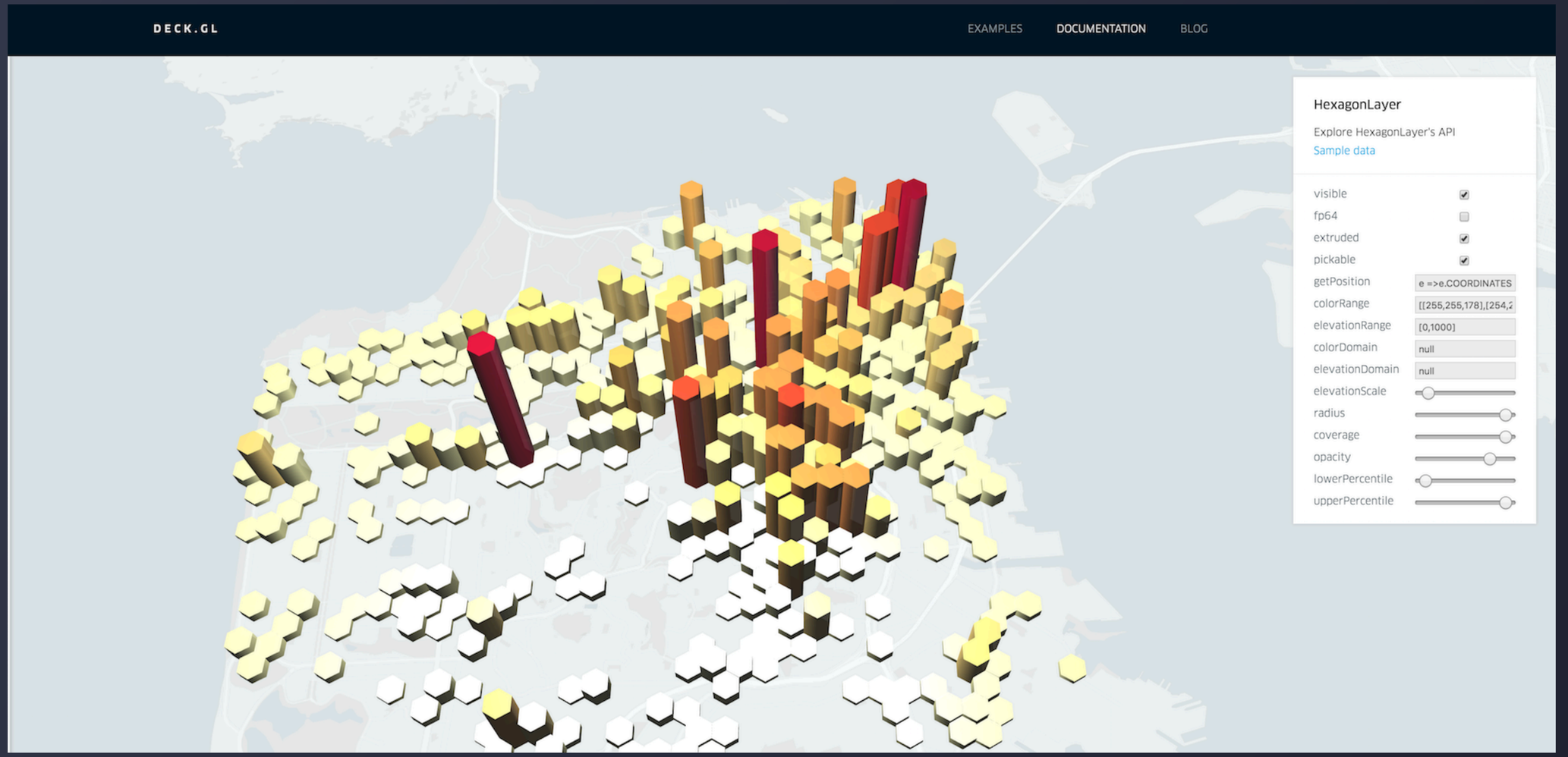
APPLICATION



APPLICATION



APPLICATION



APPLICATION

流量是每秒钟大概数万条消息，一天大概是几亿，并且每条消息包含几十个字段

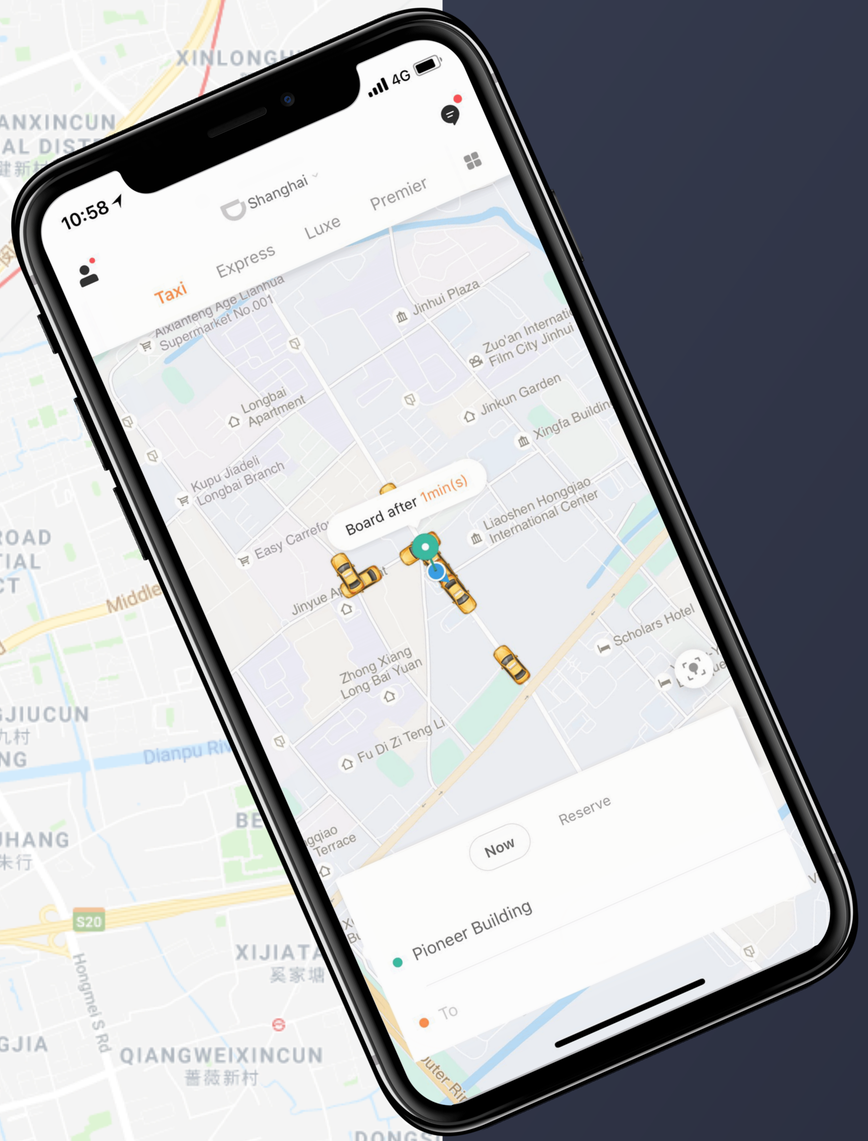
1.支持时序和地理空间的切片
2.支持大流量数据
3.支持秒级(毫秒级？)查询
4.支持原始数据查询

ElasticSearch + Kafka

ONE MORE THING

空间搜索	golang/geo	如何理解 n 维空间和 n 维时空 高效的多维空间点索引算法 — Geohash 和 Google S2 Google S2 中的 CellID 是如何生成的？ Google S2 中的四叉树求 LCA 最近公共祖先 神奇的德布鲁因序列 四叉树上如何求希尔伯特曲线的邻居？ Google S2 是如何解决空间覆盖最优解问题的? --- Code <T> share keynote

https://github.com/halfrost/Halfrost-Field



THANKS

BYEBYE