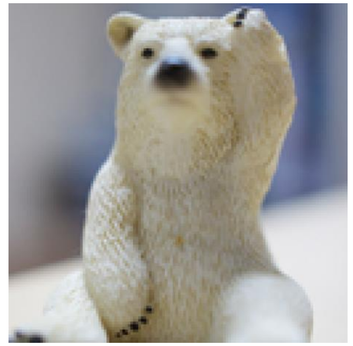
デモシーンへようこそ

～ 4KBで映像作品を作る技術 ～

Tokyo Demo Fest オーガナイザー
@kioku_systemk / @i_saint

自己紹介

奥 健太郎



@kioku_systemk

- イマジカ デジタルスケープ の デジタルリアリティラボ で
  大規模可視化システムのリードエンジニアを担当
- Tokyo Demo Fest オーガナイザー リーダー

石橋 誠也

@i_saint

- Unity Technologies Japan で主にグラフィックスを担当
- 元スクエア・エニックス。
  Tokyo Demo Fest をきっかけに Unity へ移籍

このセッションについて

最後にQ&Aの時間をとります。
質問はその際に受け付けます。

このセッションのスライドは
すぐに公開されます。

Amigaaaaaaaa!

Loading...


Loading...

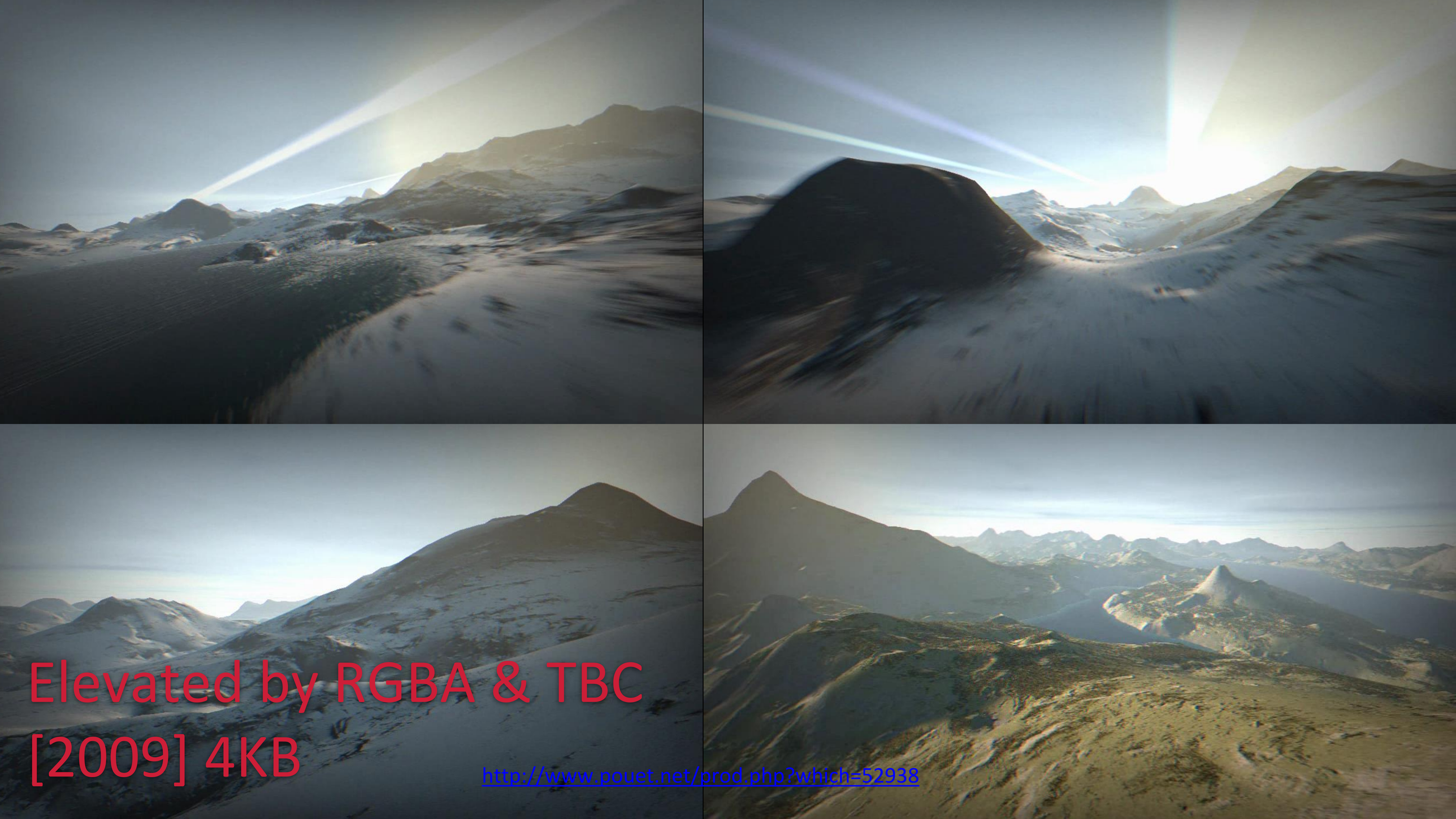

Loading...


Loading...


Loading...

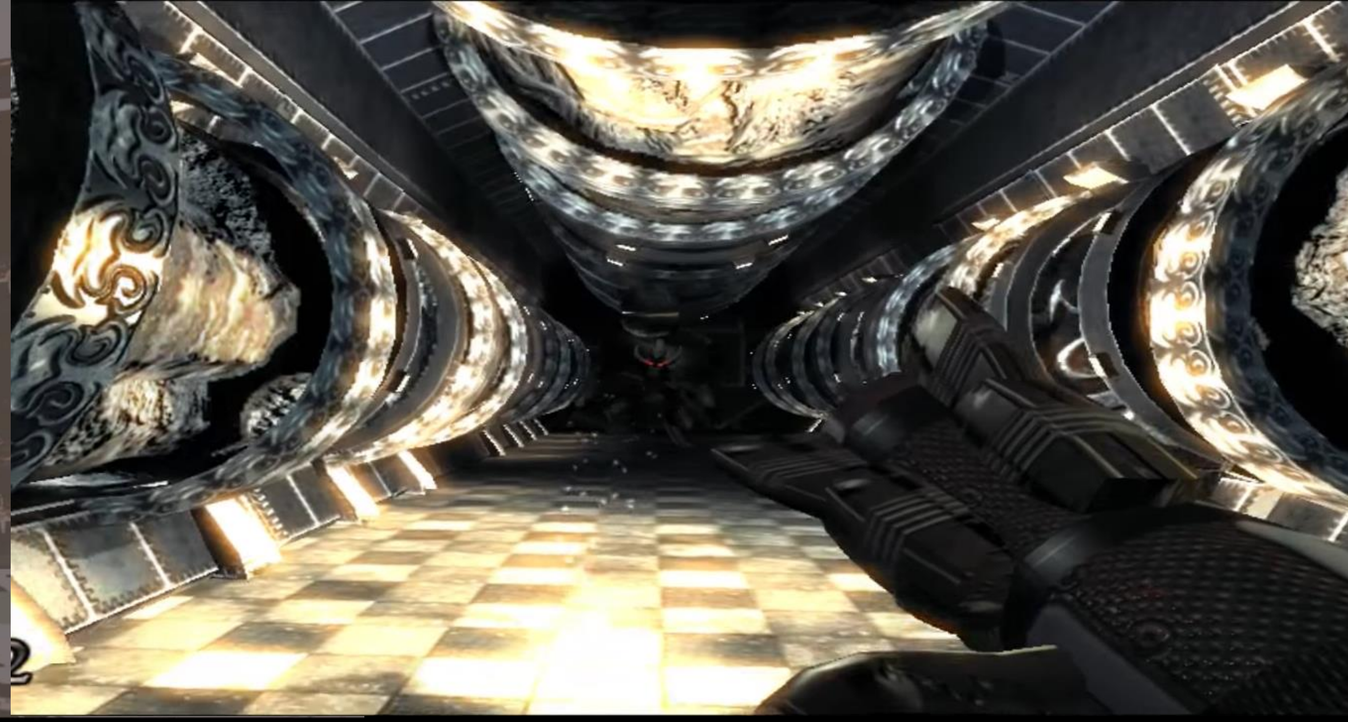
「デモシーン」
をご存知の方？

「メガデモ」
をご存知の方？


Elevated by RGBA & TBC
[2009] 4KB
http://www.pouet.net/prod.php?which=52938

fr-041: debris. by farbrausch [2007] 177KB

http://www.pouet.net/prod.php?which=30244



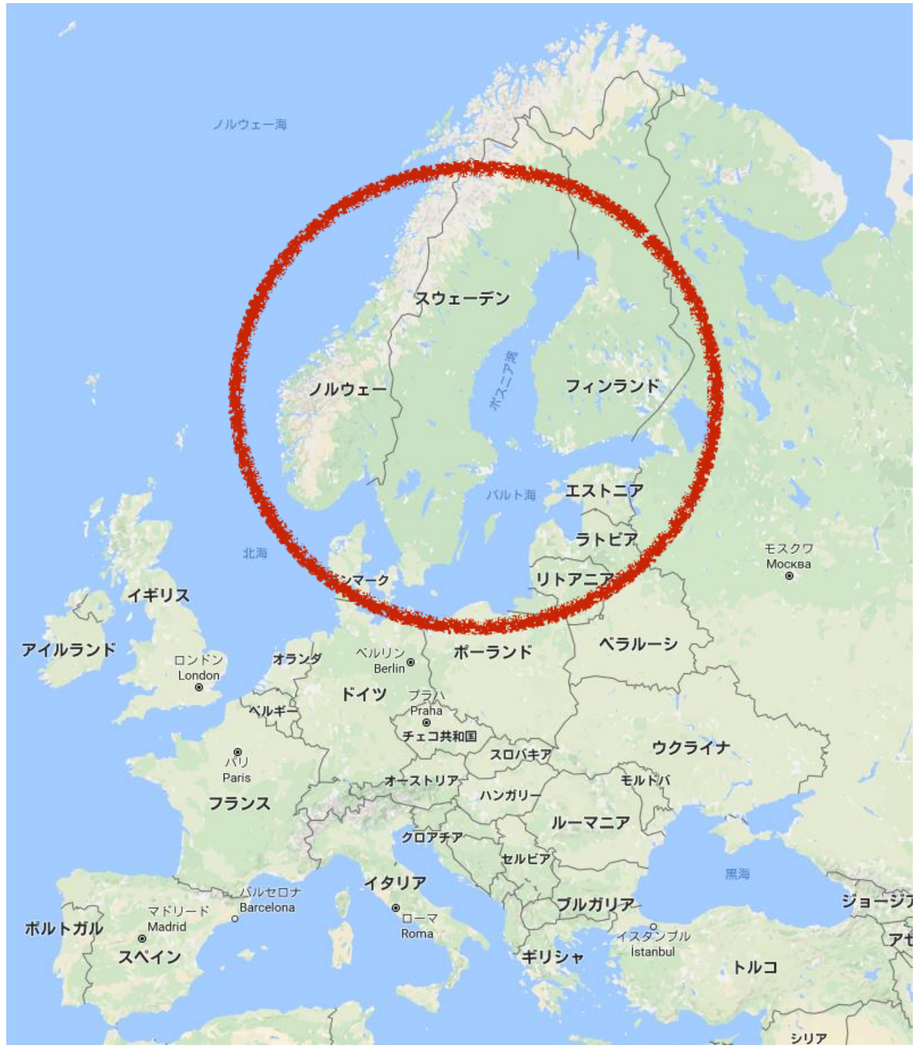
.kkreiger by farbrausch

[2004] 96KB

http://www.pouet.net/prod.php?which=12036

デモシーンは
どこからきたのか
？

デモシーンの起源



(c) Google

- 1980年ごろ、ヨーロッパ(北欧)を起源としたコンピュータカルチャー
- 外は寒いので、家にひきこもってコンピューターで遊ぶ人が多かった？

デモシーンの起源

- 1980年代 日本ではファミコン、MSX
- ヨーロッパではApple II, Commodore 64, Amigaなどのマシンが主流

The party 1992
http://www.slengpung.com/

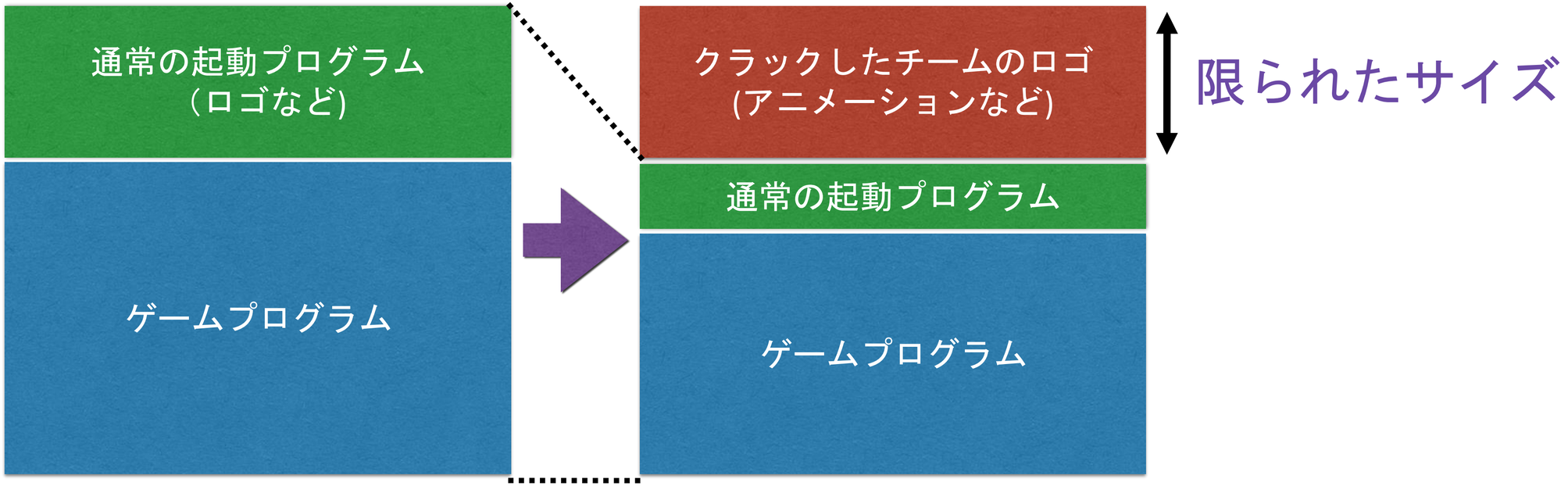
画像はwikipediaより引用

デモシーンの起源

RAZOR 1911 Amiga Cracktro [Birds of Prey]
https://www.youtube.com/watch?v=cHUMQ37cGro

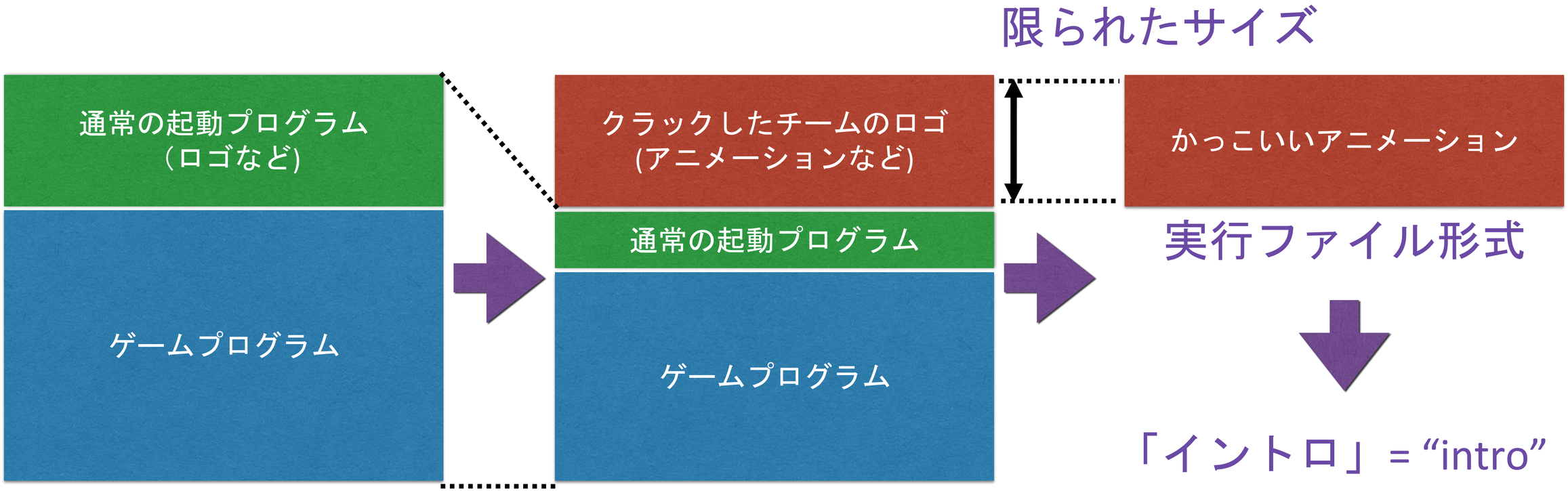
製品のゲーム
が起動

デモについて



プログラムのサイズを変えないように
元のプログラムコードは圧縮など行う。

デモシーンの起源



デモシーンの始まり!!

デモの種類について

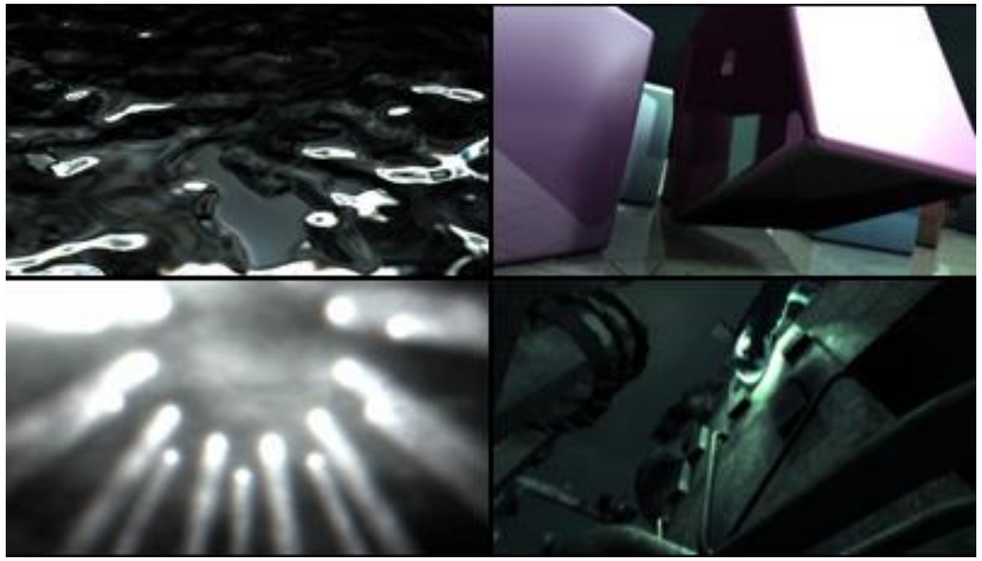
いずれも実行可能形式

- デモの種類は以下のものがあります。
- Demo
  - サイズは128MB～256MB程度
    (その時のルールによる)
- Intro
  - サイズ制限があるもの – 4KB, 64KB
    またはそれ以下のサイズなど
- 4K Executable Graphics
  - 4KBの実行ファイルで
    1枚の静止画を表示する
- Invitation Demo
  - デモパーティを開催する際に、
    作成する招待デモ
- Wild Demo
  - 特殊なハードウェア、
    市販ゲーム機のデモなど、
    上記に当てはまらない
    カテゴリーの作品など

デモの種類について

Demo (～256MB程度)

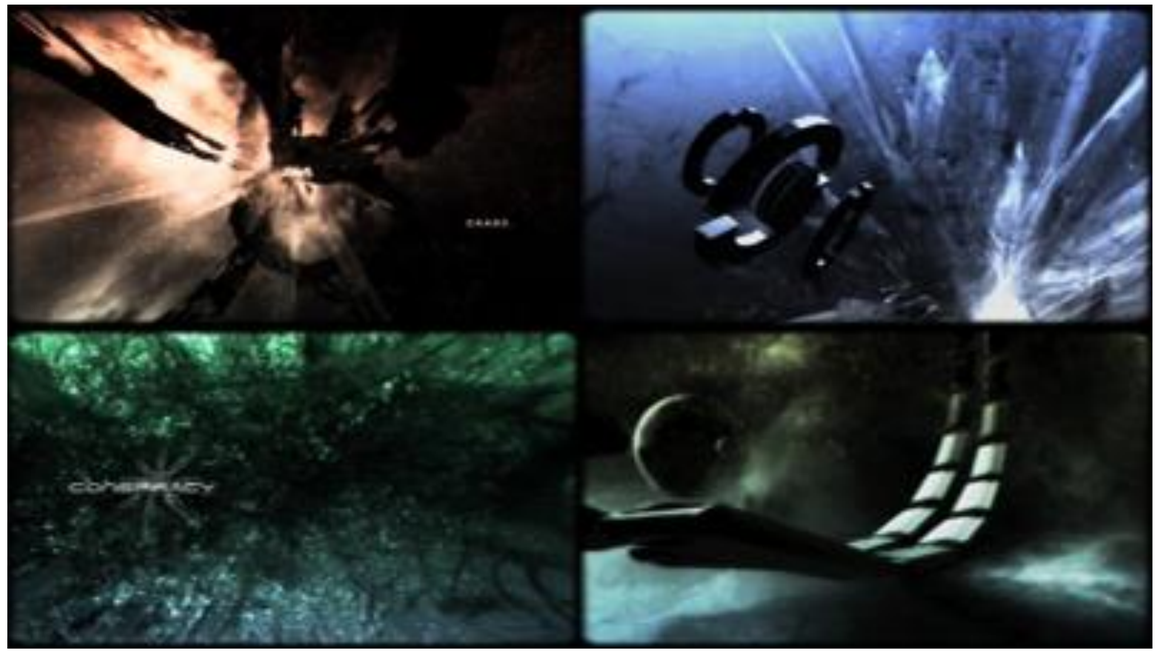
Lifeforce by ASD [2007]
http://www.pouet.net/prod.php?which=31571
https://www.youtube.com/watch?v=VacQErWAuMU



Stargazer by Orb & Andromeda [2008]
http://www.pouet.net/prod.php?which=31571
https://www.youtube.com/watch?v=VacQErWAuMU

デモの種類について

64KB INTRO



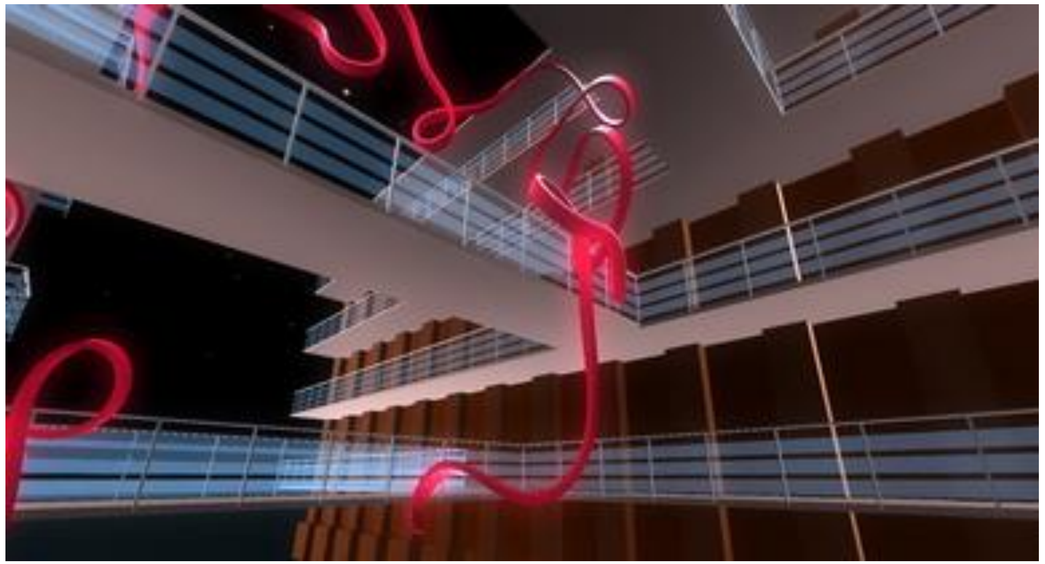
Chaos Theory by Conspiracy [2006]
http://www.pouet.net/prod.php?which=25774
http://www.youtube.com/watch?v=gfk5Mqy3gpA

Heaven7 by Exceed [2000]
http://www.pouet.net/prod.php?which=5
https://www.youtube.com/watch?v=rNqpD3Mg9hY

デモの種類について

4KB INTRO

cdak by Quite & orange [2010]
http://www.pouet.net/prod.php?which=55758
https://www.youtube.com/watch?v=RCh3Q08HMfs



Atrium by TBC & Loonies [2008]
http://www.pouet.net/prod.php?which=50063
https://www.youtube.com/watch?v=jlRoOqRs0bs

デモの種類について

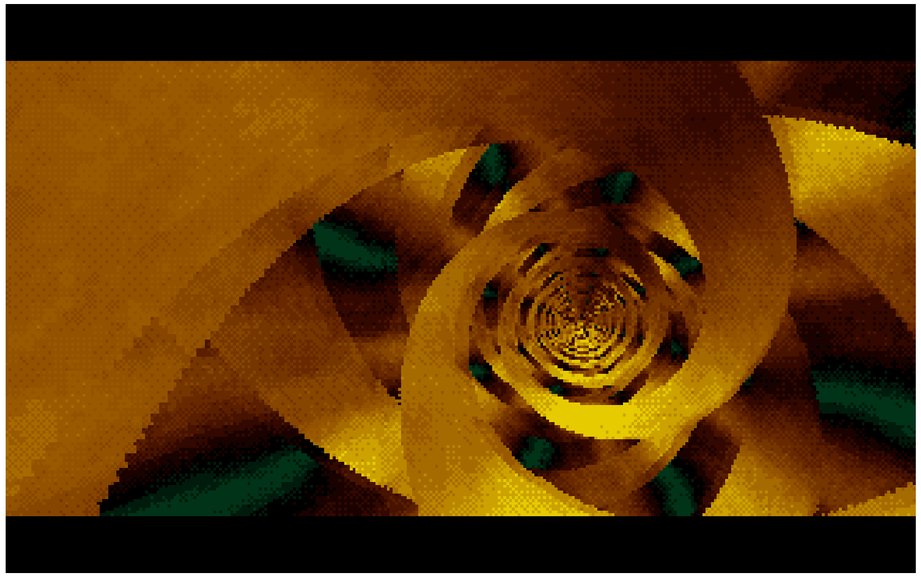
4KB EXECUTABLE GRAPHICS

ixaleno by RGBA [2008]
http://www.pouet.net/prod.php?which=50068

Burj Babil by Loonies [2010]
http://www.pouet.net/prod.php?which=54544

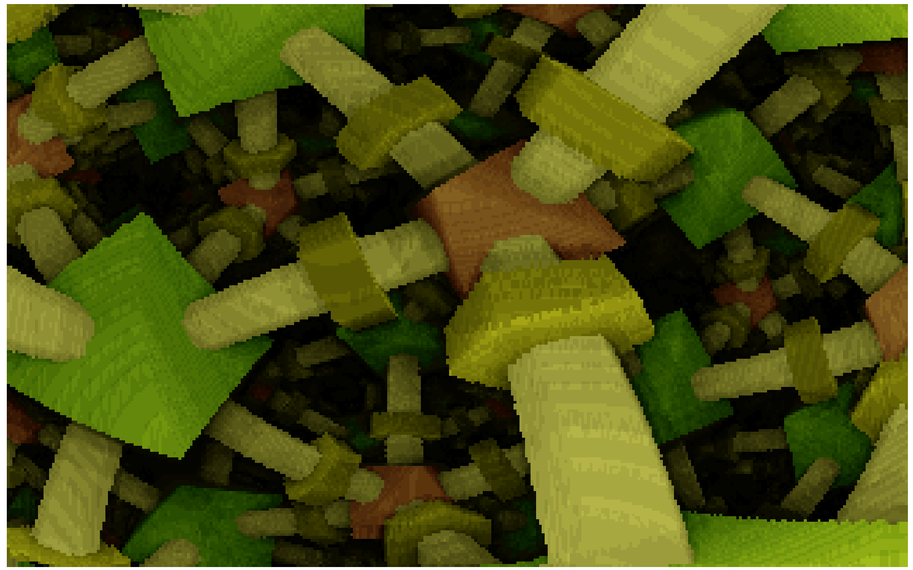
デモの種類について

256B INTRO



tube by 3sc [2001] (256バイト)
http://www.pouet.net/prod.php?which=3397

https://www.youtube.com/watch?v=f1joQfp78Yo



Puls by Řrřola [2009] (256バイト)
http://www.pouet.net/prod.php?which=53816

https://www.youtube.com/watch?v=R35UuntQQF8

デモについて

```
B0 13 CD 10 68 00 A0 07 8C C8 80 C4 10 8E E0 31
C9 BA C8 03 89 C8 EE 42 D0 F8 78 07 EE F6 E0 C1
E8 06 EE B0 00 EE 79 08 28 C8 D0 E8 EE D0 E8 EE
89 CB 64 88 1F E2 DA 89 CB 01 C8 D3 C0 88 C6 C0
FE 05 10 F2 64 12 97 FF 00 D0 EA 64 88 17 F6 D7
64 88 17 E2 E2 DB E3 D9 EE 80 C7 08 BF 04 02 D8
45 F4 57 BA B0 FF BD 60 FF BE FC 01 DF 44 D6 89
2C DF 04 89 14 DF 04 B1 02 D9 C3 D9 FB D9 C2 D8
C9 D9 C4 D8 CB DE E9 D9 CB DE CA DE CB DE C2 D9
CA E2 E6 D9 C1 DC C8 D9 C1 DC C8 DE C1 D9 FA DE
FB D9 F3 DE 4C FC DF 1C DE 4C FC DF 5C 01 8B 34
8D 00 00 E0 24 40 B0 FB 74 0F C1 E6 02 8D 00 28
E0 B0 F0 79 04 D1 E6 B0 D0 64 02 00 00 05 47 45
81 FD A0 00 75 93 42 83 FA 50 75 8A 5E BF 00 19
B5 64 F3 A5 B5 C8 4E C0 3C 02 E2 FA E4 60 98 48
0F 85 65 FF B0 03 CD 10 29 00 C3 3C 62 61 7A 65
```

tube by 3sc [2001] (256バイト)
http://www.pouet.net/prod.php?which=3397

https://www.youtube.com/watch?v=f1joQfp78Yo

```
B0 13 53 BA C8 03 CD 10 88 D8 84 CB 7A 05 F6 E8
C1 E8 07 F6 EB 88 E0 EE B2 C9 E2 EC B1 03 4B 75
E9 68 CE 9F 07 B7 56 DB E3 83 00 58 DF 00 D9 FB
DC F9 DF 00 D8 0C D9 FE DE 0C 66 5A 06 55 60 89
1F 8B 05 DF 05 F7 E8 29 17 4F 7B F5 DF 07 66 81
05 CD CC 00 00 D8 CC D9 C0 D8 CE D9 CA D8 CD DC
EA D8 CE DE C1 D9 CA 47 7B EB 4F 6B 10 0A DF 19
89 17 02 34 00 FB 73 F6 99 B4 E6 11 D9 E8 15 00
28 CC D5 04 04 46 89 45 FC 61 45 26 88 02 75 AE
E4 60 48 75 92 B3 00 8B 29 D3 FD 31 D5 01 2F 00
FB 73 F4 D6 DF 10 51 D3 E9 80 C5 25 8B 10 F7 18
69 E8 00 80 2B 29 79 02 F7 DD D1 ED 01 EA 89 2F
00 FB 73 EC 39 CA 40 72 25 B3 02 7A DF 2B 10 40
2B 10 80 EE 60 6B D2 0D 8B 14 70 02 40 99 2B 2F
79 02 F7 DD 01 EA 8B 2F 00 FB 73 F2 39 CA 59 19
D2 F5 18 D1 10 D1 80 F9 06 73 04 00 D4 75 96 C3
```

Puls by Řrřola [2009] (256バイト)
http://www.pouet.net/prod.php?which=53816

https://www.youtube.com/watch?v=R35UuntQQF8

デモの種類について

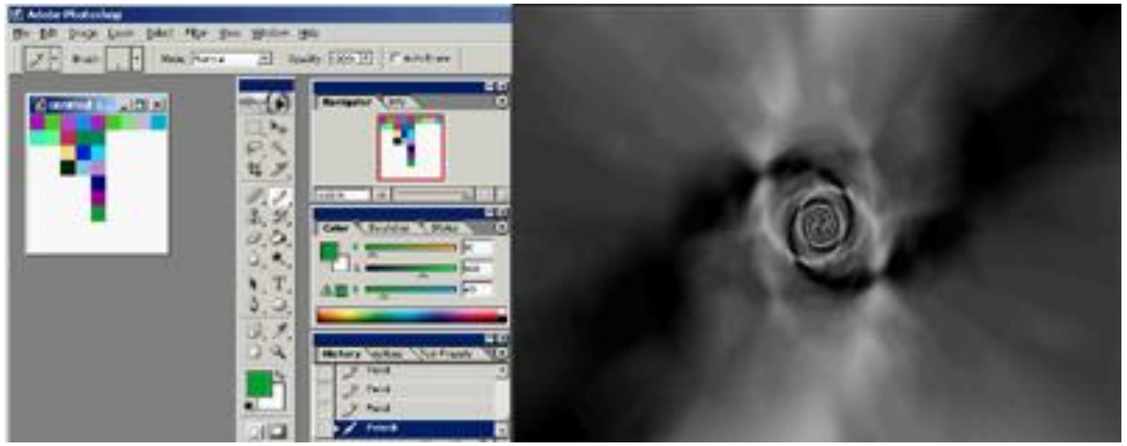
Invitation demo

YouShould by Haujobb [2010]
http://www.pouet.net/prod.php?which=54920
https://www.youtube.com/watch?v=U9Q4YE6b9dI

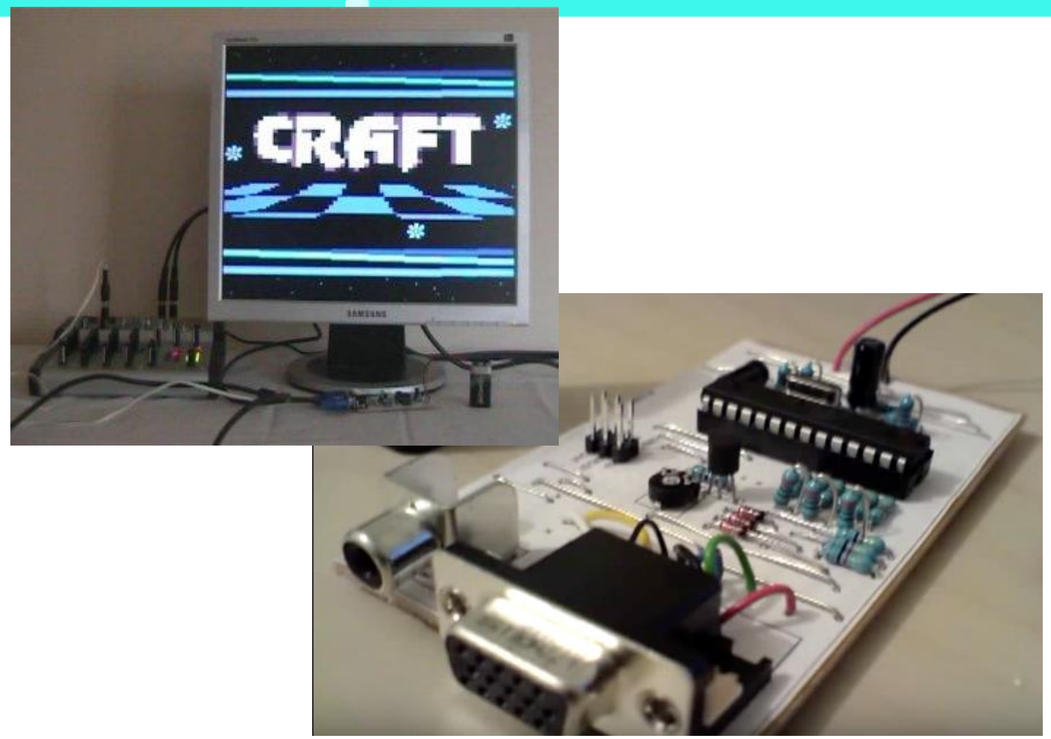
“Kings of the Playground” - Evoke 2004 64k invite by Equinox [2004]
http://www.pouet.net/prod.php?which=12790
https://www.youtube.com/watch?v=YQ1uDEbhWO8

デモの種類について

Wild demo



real pixel coding by RGBA [2011]
http://www.pouet.net/prod.php?which=56814
https://www.youtube.com/watch?v=cfYTZBJ2N9g



Craft by lft [2008]
http://www.pouet.net/prod.php?which=50141
https://www.youtube.com/watch?v=sNCqrylNY-0

デモについて

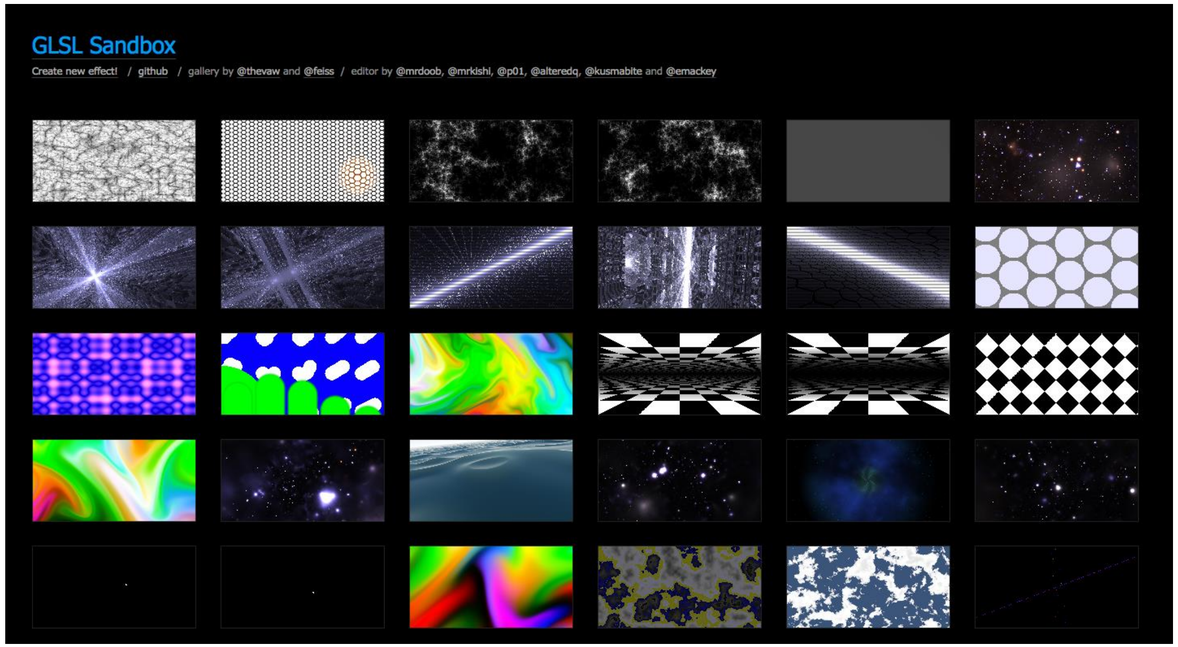
poüet.net (英語)
http://www.pouet.net

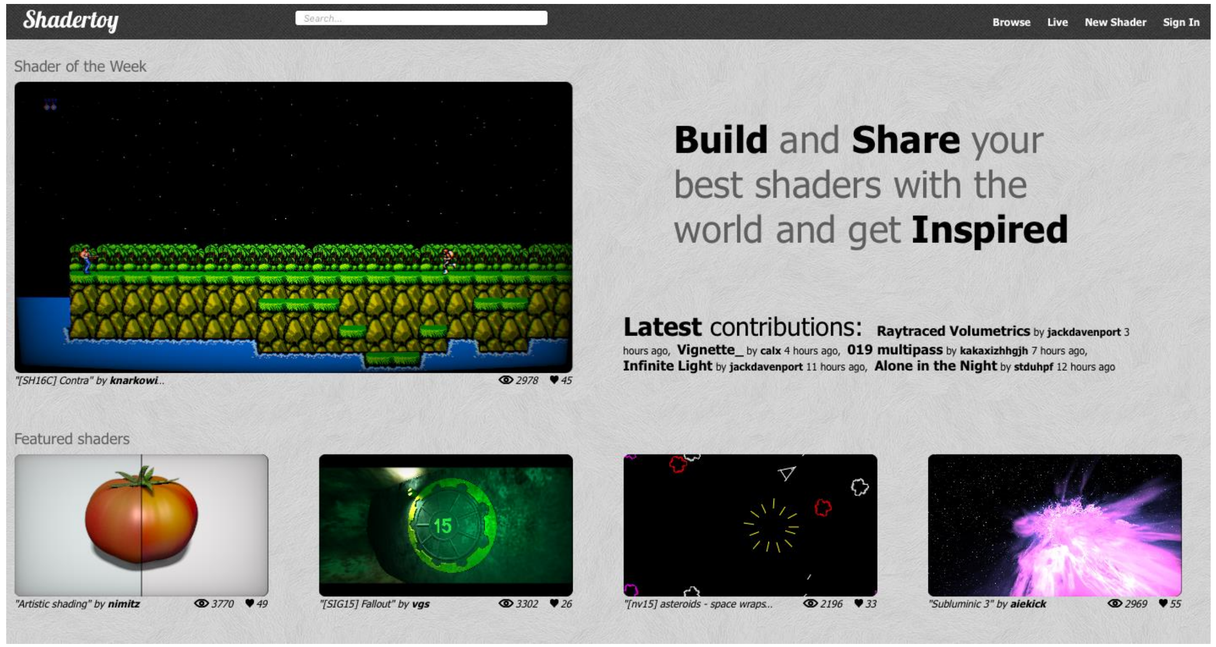
curio (英語)
http://curio.scene.org/

メガデモ[MEGADEMO] ダウンロード (日本語)
http://megademo.nobody.jp (2002-2005)

demo99 return of the pixel (日本語)
http://demo99z.web.fc2.com/

ブラウザデモへの発展





- GLSL Sandbox (英語)
  http://glslsandbox.com

- Shader toy (英語)
  https://www.shadertoy.com

デモパーティについて

デモパーティについて

みんなでコンピュータ
とゲームを
持ち寄って遊ぶ

→

LANパーティ

→

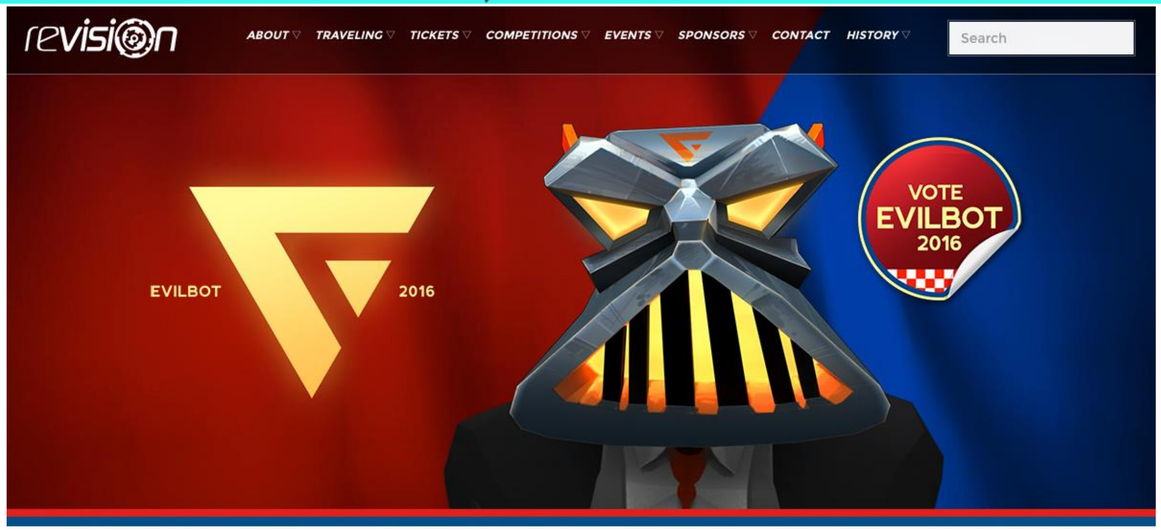
デモパーティ

- るむる
- するべ飲く
- 出見食をつー
- ををルをナ
- モモザ一達ミ
- デデピビ友セ
- ・
- ・
- ・
- ・
- ・

→

e-sportsの大会

デモパーティについて



https://2016.revision-party.net

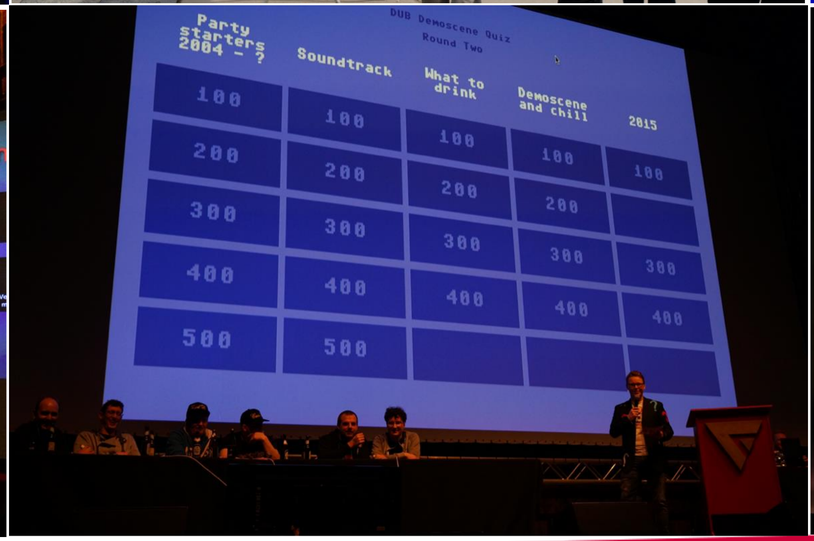
- イースターの祝日に開催
- 1000人以上が参加
- 4日間開催
- ヨーロッパ各国から参加

Revision 2016のビジター席

デモパーティについて


PC 64K INTRO


NEVER
BE OFFLINE
viprinet®
AGAIN.
Assumption
• Linear motion
• Ignore occlusion


DUB Democscene Quiz
Round Two
Party starters 2004 - ?
Soundtrack
What to drink
Democscene and chill
2015
100
200
300
400
500

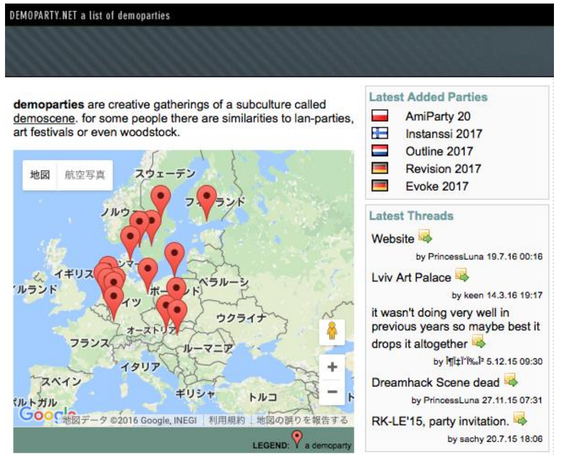

METEORIKS
CELEBRATING THE DEMOSCENE

デモパーティについて

- デモパーティの様子はslengpungというサイトから写真を見ることができます。
- デモパーティによってはストリーミング配信もあります。

http://www.slengpung.com

- 開催予定のデモパーティのスケジュールはdemoparty.netに登録されています。

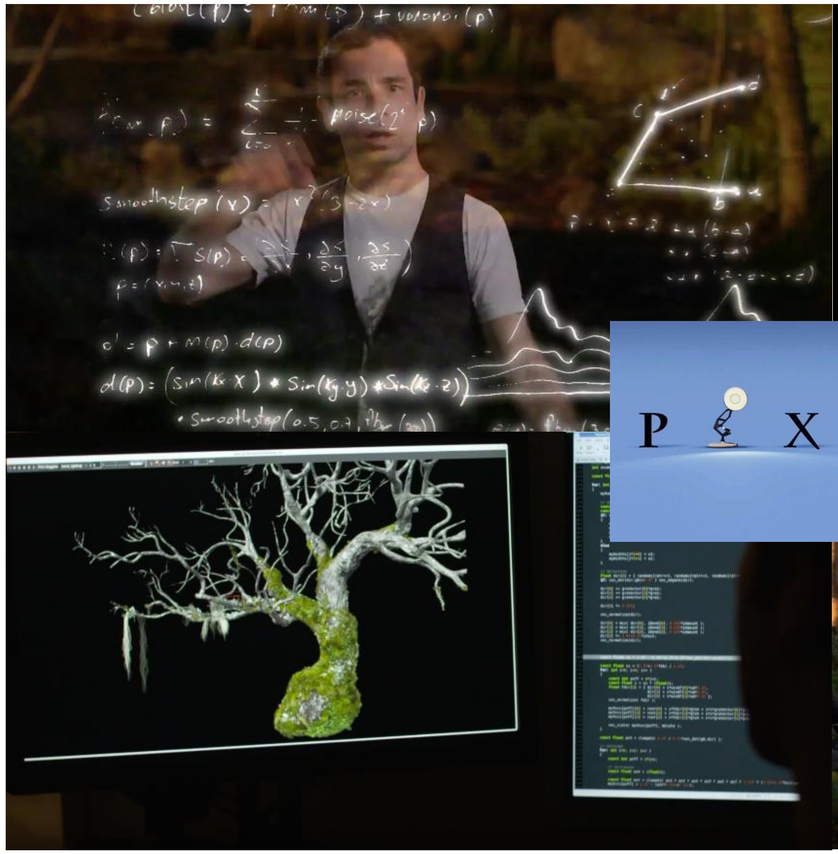


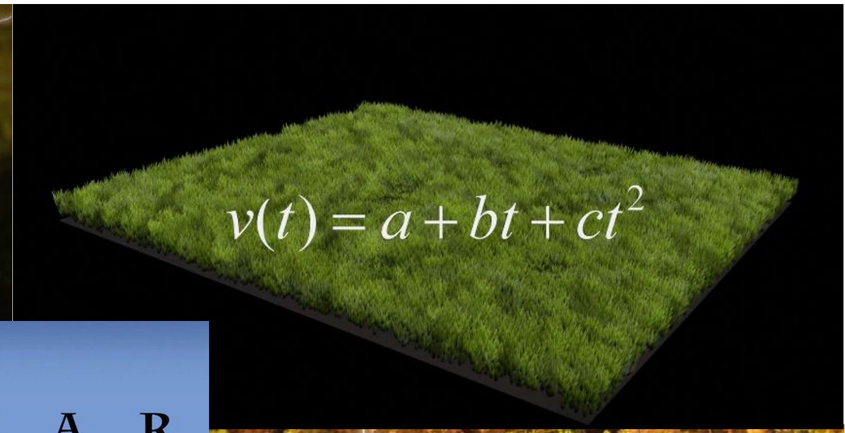
http://www.demoparty.net

デモシーナーの活躍

デモシーナーの活躍

[@aras_p](https://twitter.com/aras_p)





- ヨーロッパでは、デモシーンを経験した多くの人々は、リアルタイムCGを活用する企業で活躍しています。

- Inigo Quilez - Pixar Wonder Moss [英語]
- https://www.youtube.com/watch?v=Z_Vk3Yn-wCk

デモシーナーの活躍

Tim Sweeney
the founder of Epic Games

https://twitter.com/timsweeneyepic/status/639920825878085632

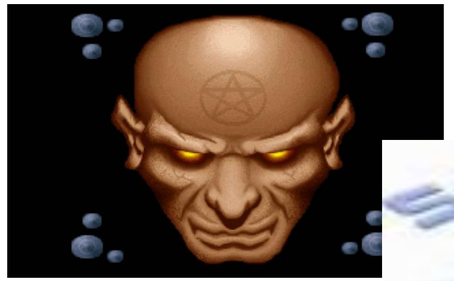
John Carmack
CTO of Oculus VR [英語]

@QuakeCon 2011 Keynote

https://www.youtube.com/watch?v=4zgYG-_ha28&feature=player_detailpage#t=4827s

- 海外では、技術のトップがデモシーンについて言及していることもあります。

デモグループの活躍



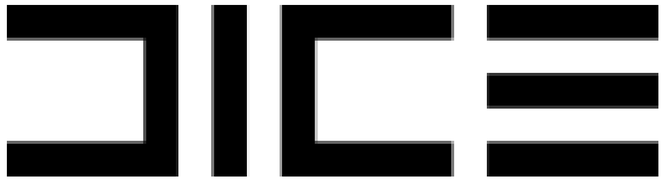
Second Reality by Future Crew [1993]
https://www.youtube.com/watch?v=rFv7mHTf0nA

3DMark2001 by Futuremark [2001]
https://www.youtube.com/watch?v=yg7Vk4b0Ook

FUTUREMARK®
a UL company

http://www.futuremark.com

The Silents [TSL]
http://www.pouet.net/groups.php?which=197



https://ja.wikipedia.org/wiki/EA_Digital_Illusions_CE より

https://www.youtube.com/watch?v=iRkZcTg1JWU

6octaves
The voice of 6octaves

Let your passion guide you

http://6octaves.blogspot.jp/p/demoscene.html

http://6octaves.blogspot.jp/search/label/Demoscene%20is%20still%20ON

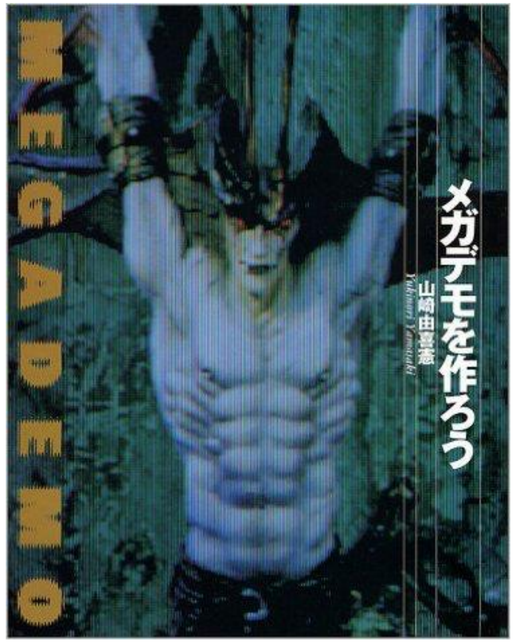
日本のデモシーンについて

TOKYODEMOFEST2016

Tokyo Demo Fest について

日本のデモシーンについて

黎明期



メガデモを作ろう[1997]
https://www.amazon.co.jp/dp/4797303522

メガデモを語るスレッド

メガデモダウンロード (2002-2005)

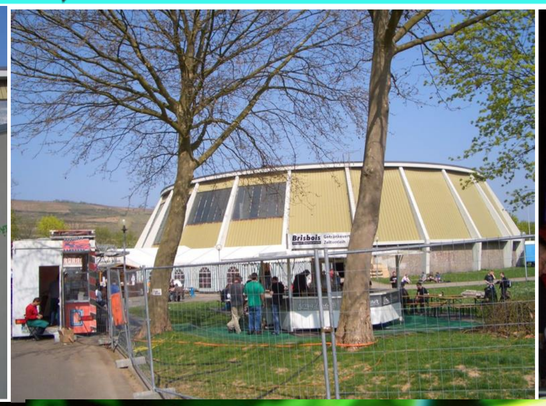
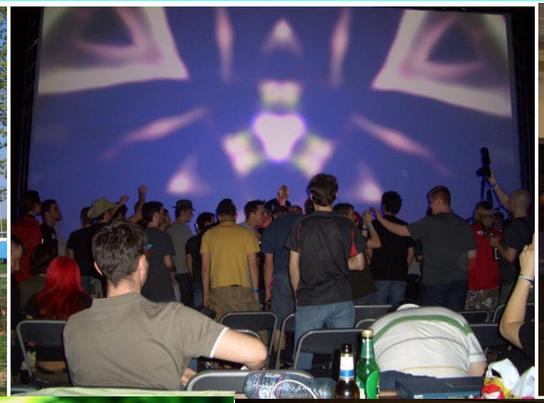
http://megademo.nobody.jp

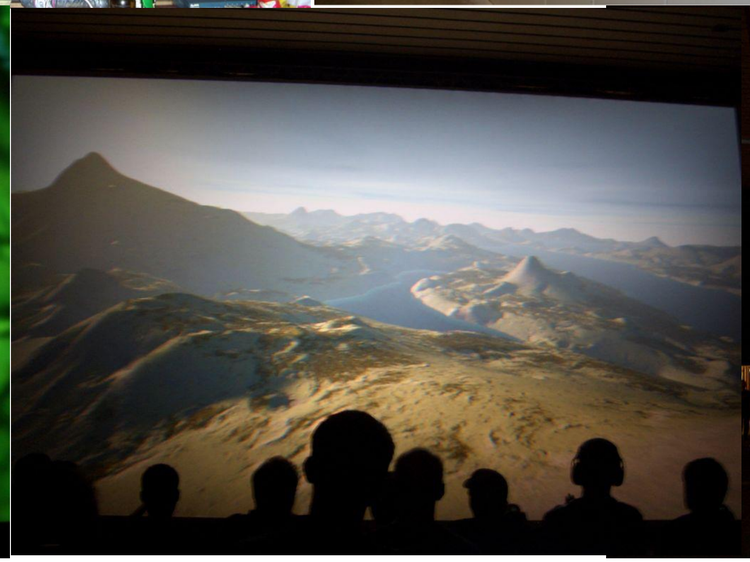
2chparty

オンライン投稿型

- [2001]
- [2005]
- [2005 summer]
- [2006]
- [2006 summer]
- [2007]
- [2007summer]
- [2009]

http://www.pouet.net/search.php?what=2chparty&type=party

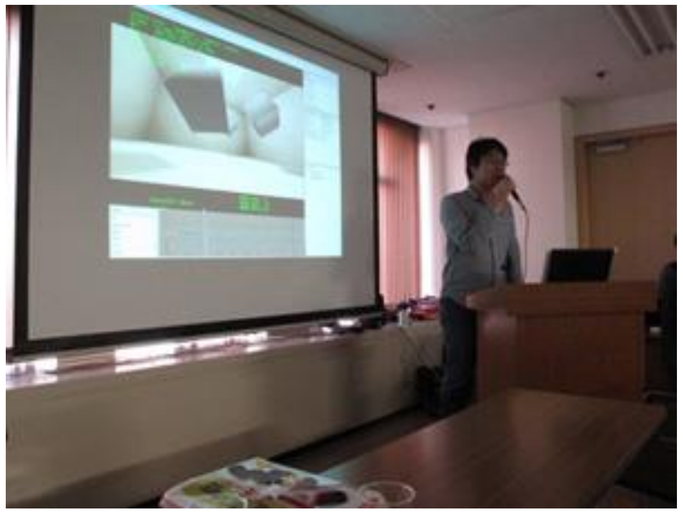




Breakpoint 2009 in Germany

Tokyo Demo Festについて

- 日本初のデモパーティ
- すみだ産業会館
- 参加者50人
- 12のデモエントリー
- 4つのセミナー
- http://tokyodemofest.jp/2011/



http://kagamin.net/hole/after/aftertdf2011.htm

Tokyo Demo Festについて

TOKYO.DEMO.FEST

potentially highly demoactive zone

[2012 - 2015]

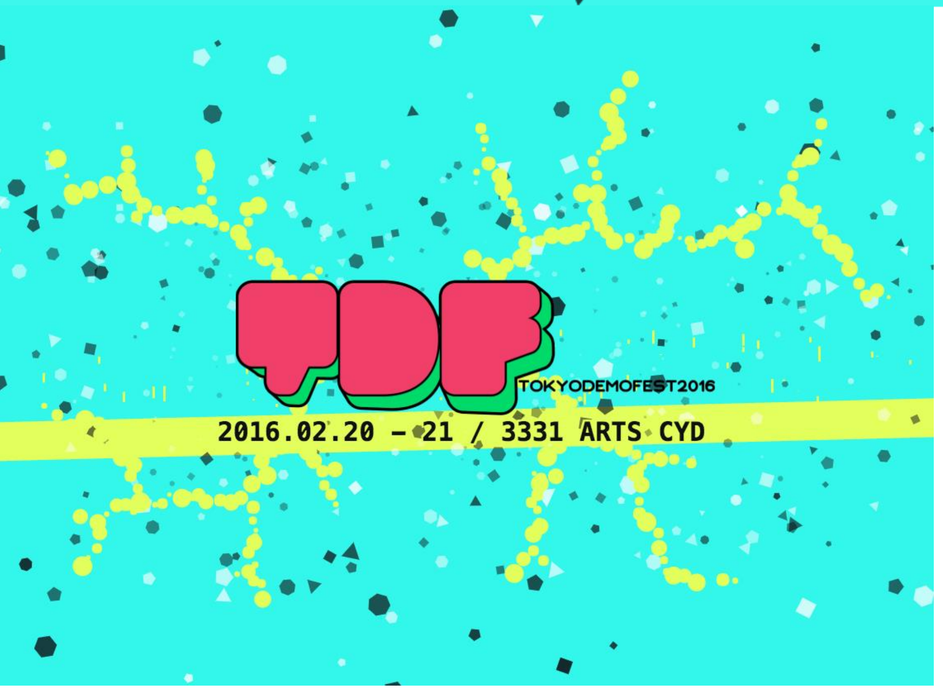
Tokyo Demo Fest

Iidabashi, Feb 9-10, 2013

Tokyo Demo Fest 2014

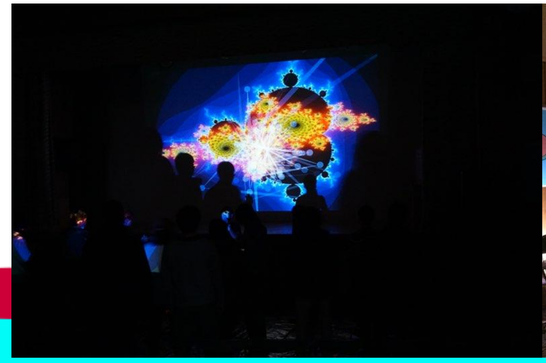
JPN TOKYO DEMO FEST 2015
パーティで会いましょう

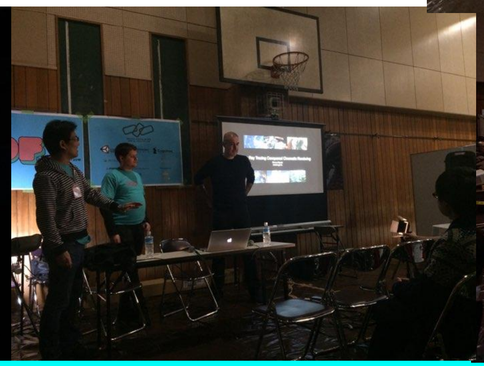
Tokyo Demo Festについて



- ６回目の開催
- 3331 ARTS Chiyoda
- 参加者132人
- 19のデモエントリー
- 19のグラフィックスエントリー
- 16のミュージックエントリー
- 5つのセミナー
- DJ/VJ イベント
- http://tokyodemofest.jp/2016/

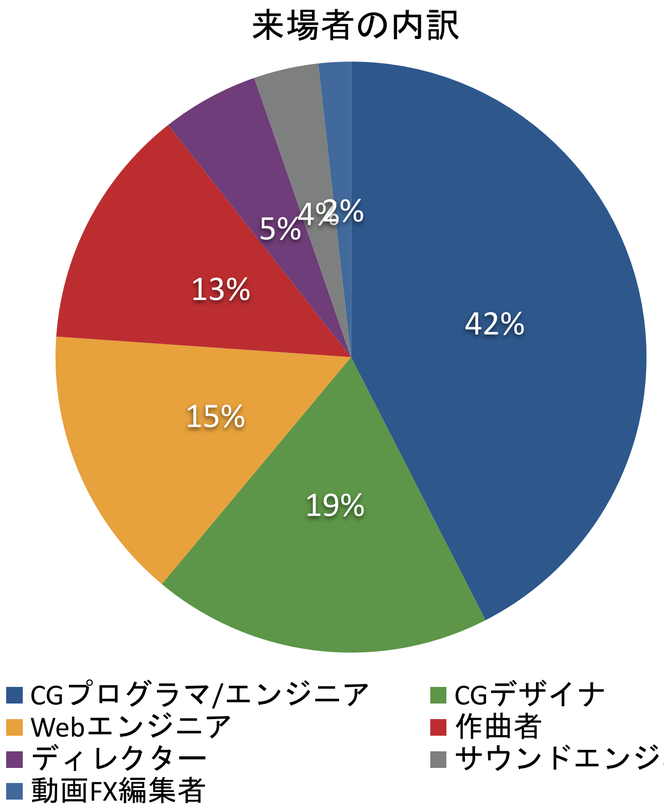
- Combined Demo compo
- Graphics compo
- Music compo
- GLSL Graphics compo

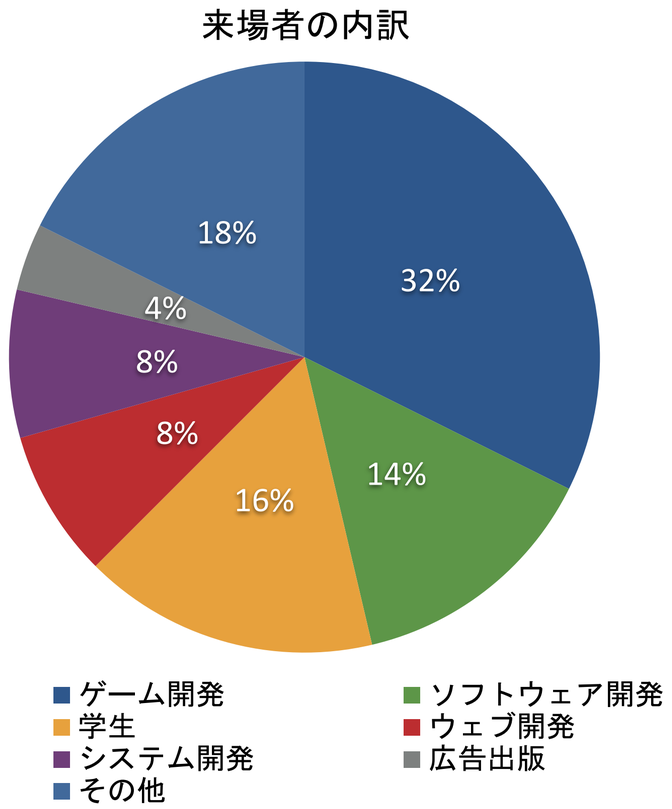




Tokyo Demo Festについて

- Tokyo Demo Fest 2016
- 参加者 合計132人
  （過去最大）
  - 一般: 83人
  - 女性: 15人
  - 学生: 22人
  - オーガナイザー: 12人





We are organizers!

FL1NE

mog

falken

zavie

kemas

i-saint

kioku

hole

gyabo

stephane

ototoi

tomohiro

hachisuka

Tokyo Demo Fest について

2nd Stage Boss by 0x4015, YET11 [2016] 4KB http://www.pouet.net/prod.php?which=66962

[LIVE] https://www.youtube.com/watch?v=zZ5HsBTYB4M

4k intro の技術

4k intro の技術

4k intro とは

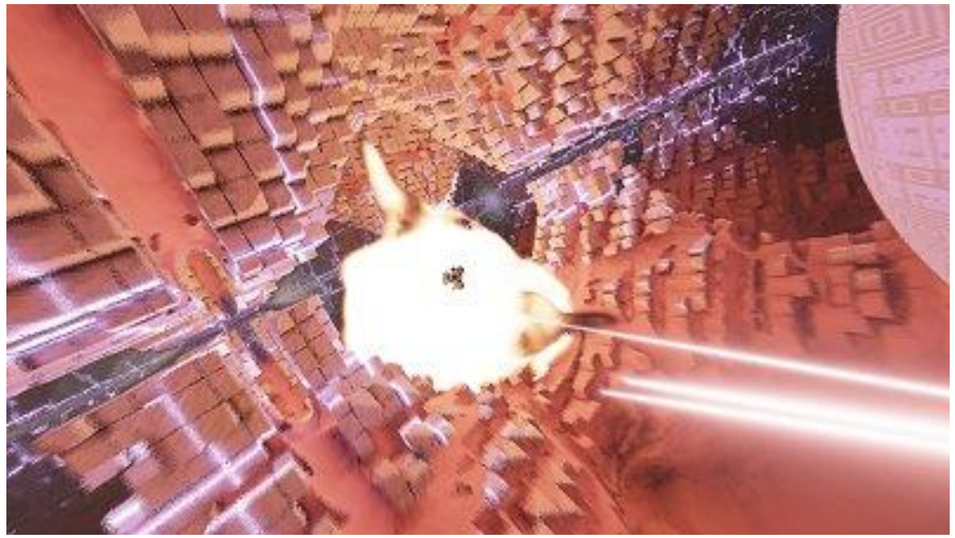
- 4KB の exe ファイルによる映像作品
- 外部ファイルへの入出力は禁止
- デモシーンの中でも花型なジャンル

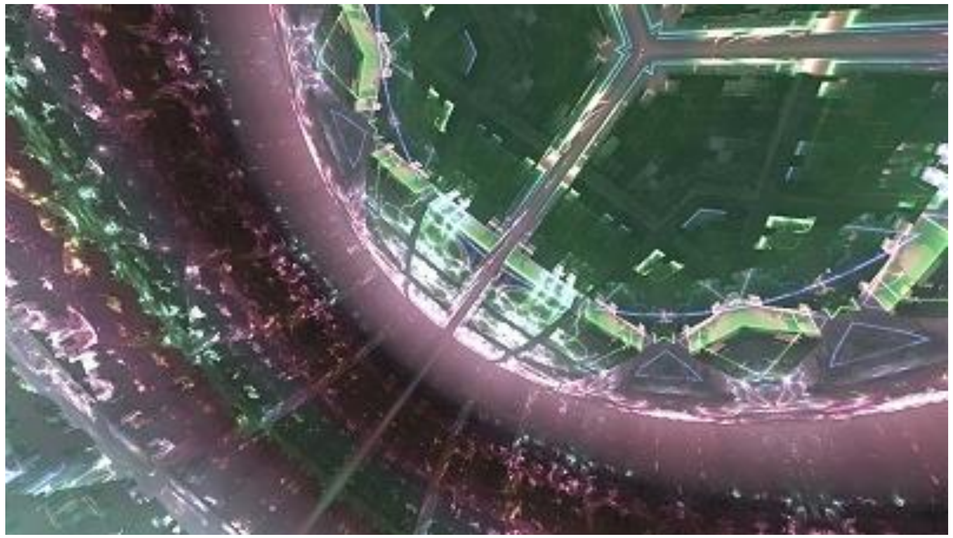
4k intro とは

(動画による紹介)

4k intro とは

- TokyoDemoFest 2016, 2015 の最優秀賞作品も 4k intro
  - 2nd stage BOSS: http://www.pouet.net/prod.php?which=66962
  - Optical Circuit: http://www.pouet.net/prod.php?which=65125





本講演の前提

- Windows
- x86 (32bit)
- OpenGL
- 近年の 4k intro でよく用いられる手法にフォーカス
  - distance function & raymarching

Agenda

- 小さい exe を作る
- レンダリング
- サウンド
- まとめ

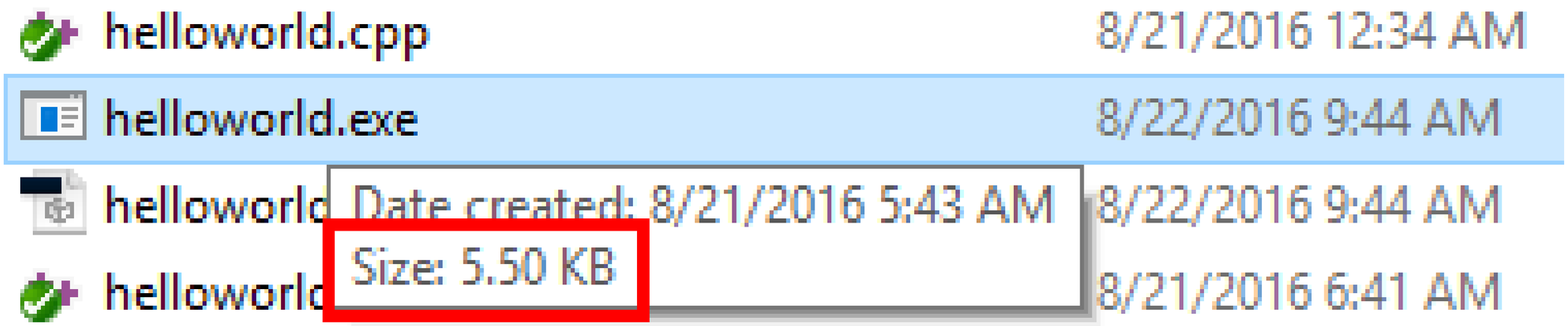
Agenda

- 小さい exe を作る
- レンダリング
- サウンド
- まとめ

小さい exe を作る

```
1  #include <cstdio>
2
3  int main(int argc, char *argv[])
4  {
5      printf("hello world!");
6  }
```

小さい exe を作る


helloworld.cpp
8/21/2016 12:34 AM
helloworld.exe
8/22/2016 9:44 AM
helloworld
Date created: 8/21/2016 5:43 AM
8/22/2016 9:44 AM
Size: 5.50 KB
helloworld
8/21/2016 6:41 AM

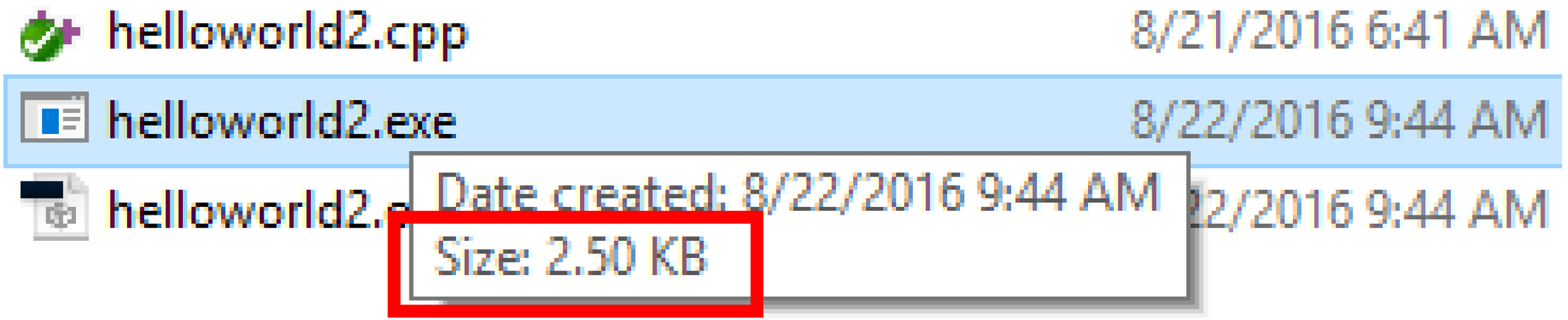
小さい exe を作る

- 普通にやっては 4KB 以下にはならない
- C Runtime のコードが含まれるのが原因
  - 使っていないつもりでも CRT のコードが一部含まれる
- リンカオプション /nodefaultlib で CRT を除外可能

小さい exe を作る


```
1  #pragma runtime_checks("", off)
2  #pragma comment(linker, "/nodefaultlib /subsystem:windows")
3
4  #include <windows.h>
5
6  void WINAPI WinMainCRTStartup()
7  {
8      MessageBoxA(nullptr, "hello world!", "hello", MB_OK);
9  }
```

小さい exe を作る


helloworld2.cpp
8/21/2016 6:41 AM
helloworld2.exe
8/22/2016 9:44 AM
helloworld2.
Date created: 8/22/2016 9:44 AM
Size: 2.50 KB
22/2016 9:44 AM

小さい exe を作る

- サイズは縮むが標準ライブラリは当然使えなくなる
  - printf(), malloc() 等
- 一部の C++ の機能も使えなくなる
  - new, global object のデストラクタ (= atexit()) 等
- WinAPI は使えるが、でかいバッファは静的に確保しておく方が有利
  - e.g.: static char buf[1024 * 1024 * 64];
- これらの要因により、intro はほぼ C で書くことになる

小さい exe を作る

- ４KB 以下にはなったが、このままではあまりに余裕がない
  - ウィンドウと OpenGL 初期化だけで約 4KB
- Packer というツールを使ってさらなる容量削減を行う

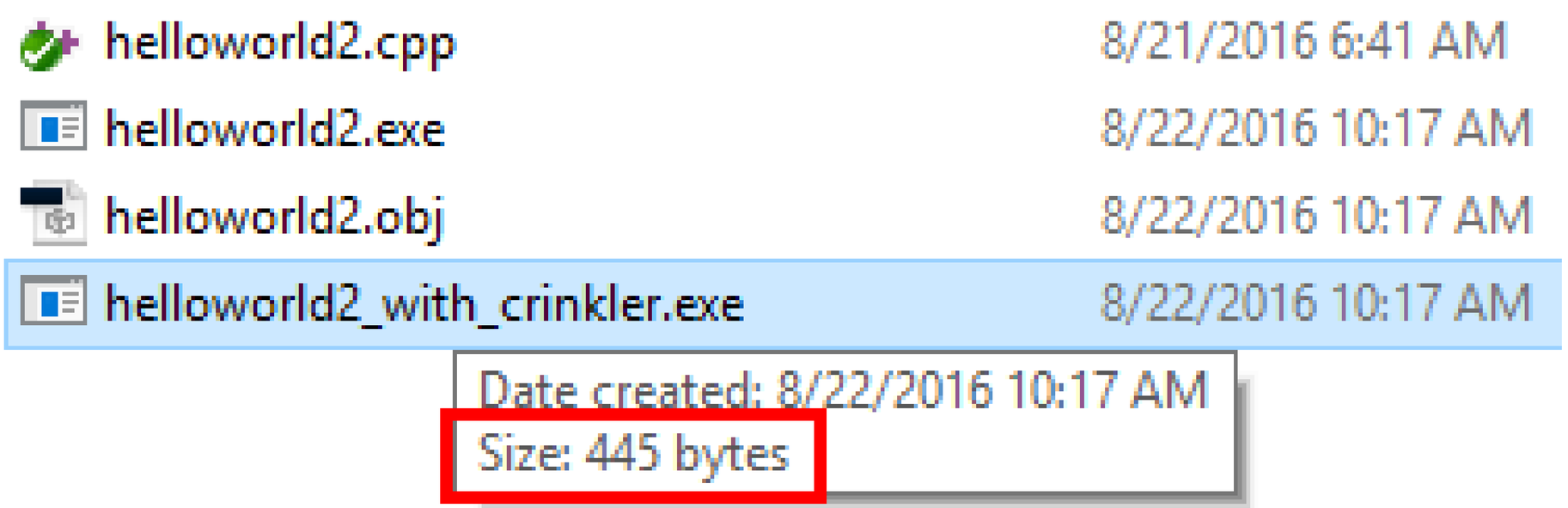
Packer

- 実行コードを圧縮した状態で格納し、プログラム起動時にそれを展開して実行するプログラム
- crinkler という packer が有名 http://crinkler.net/
  - kkrunchy, UPX などもある
- 今回は crinkler を中心に解説

Crinkler

- リンカとして機能し、VisualStudio の link.exe とほぼ同様に使える
  - .obj や .lib を食わせて .exe を生成
- 生成された exe は pack されており、小さい
  - 近年の intro は大部分がシェーダコードで縮みやすい
  - 12KB -> 4KB くらい
- 現在 VisualStudio2015 とは相性が悪い？様子。VS 2010 で検証

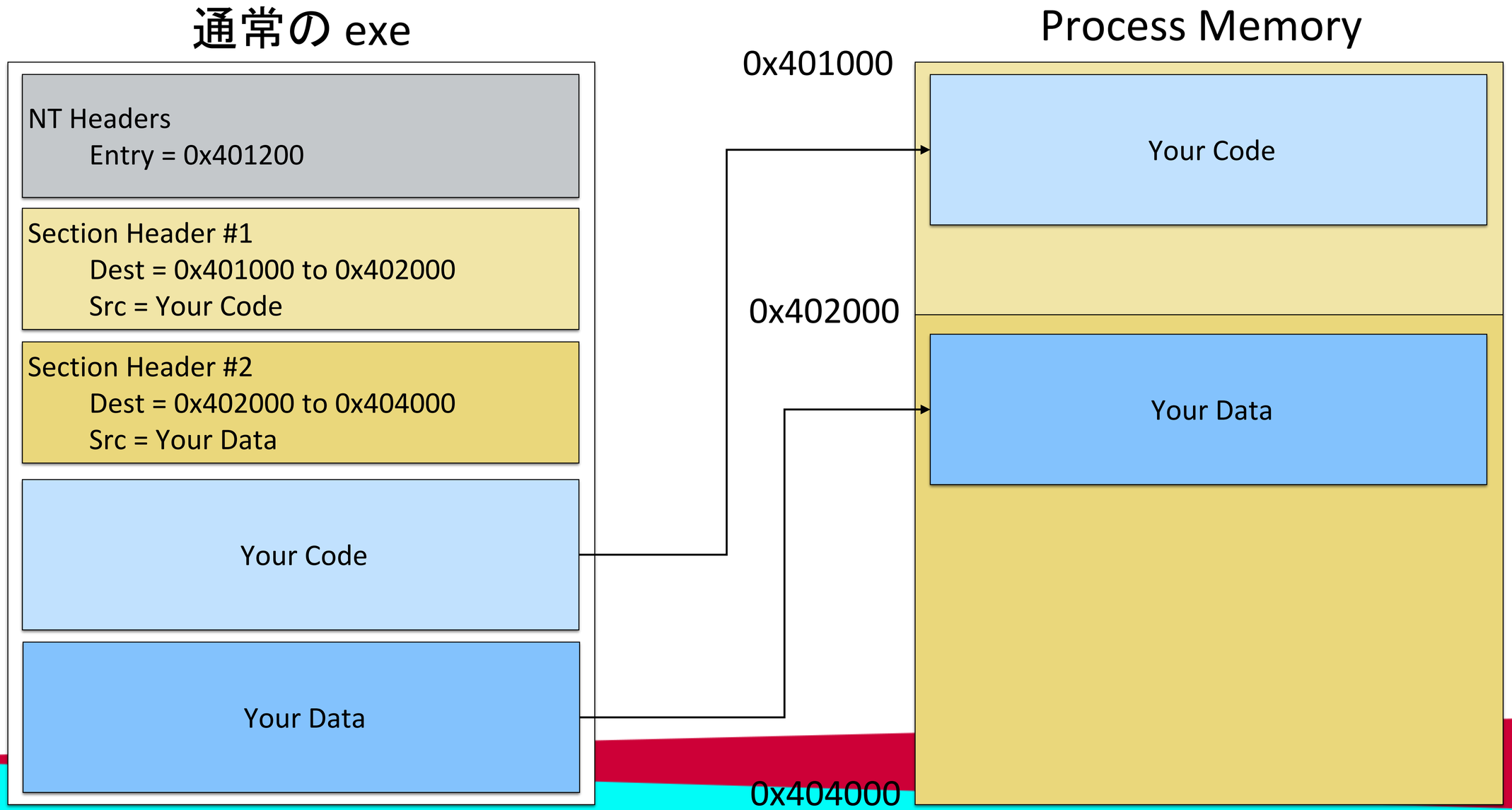
Crinkler


helloworld2.cpp 8/21/2016 6:41 AM
helloworld2.exe 8/22/2016 10:17 AM
helloworld2.obj 8/22/2016 10:17 AM
helloworld2_with_crinkler.exe 8/22/2016 10:17 AM
Date created: 8/22/2016 10:17 AM
Size: 445 bytes

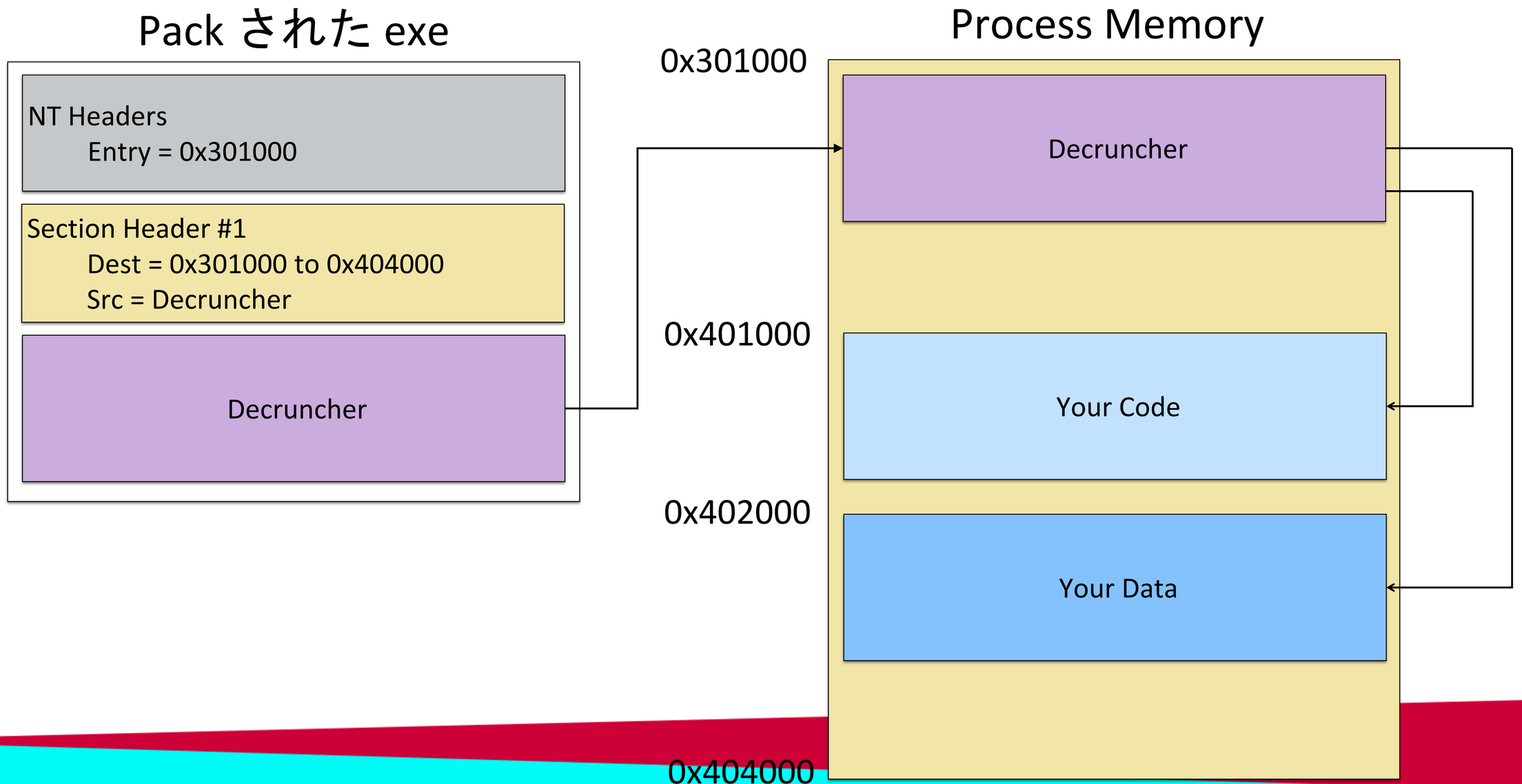
Crinkler の動作概要

- 実行コードおよびデータを圧縮した状態で保持
- それらを展開して jump するコードをエントリポイントとして持つ

Crinkler の動作概要



Crinkler の動作概要



Crinkler の動作概要

- 依存 dll および関数の解決
  - 通常は OS が自動的にやるが、packer はそれを自力で解決する
  - 文字列を hash 化して照合することにより容量削減
- exe ファイルのヘッダの未使用領域にコードを詰める
  - exe ファイルの内容はそのままメモリにマップされる
  - 未使用領域にコードを詰め込んでそこに制御を移す

小さい exe を作る

- ウィンドウを開いて OpenGL を初期化して空のシェーダで描画し、空のサウンドを鳴らすプログラムが約 1KB
- ここからどう内容を詰め込んでいくかが勝負

Agenda

- 小さい exe を作る
- **レンダリング**
- サウンド
- まとめ

レンダリング

- ポリゴンモデルの生成は複雑な処理でコード量も肥大化しがち
- 4k intro ではポリゴンを使わない手法がよく用いられる
- それが distance function & raymarching

Distance function & Raymarching

- 画面全体を覆う 4 角形を 1 枚出す
- あとは全てピクセルシェーダでがんばる

Distance function & Raymarching

- モデルはピクセルシェーダの中で数式で表現し、可視化する
- このモデルの数式が distance function、可視化のプロセスが raymarching

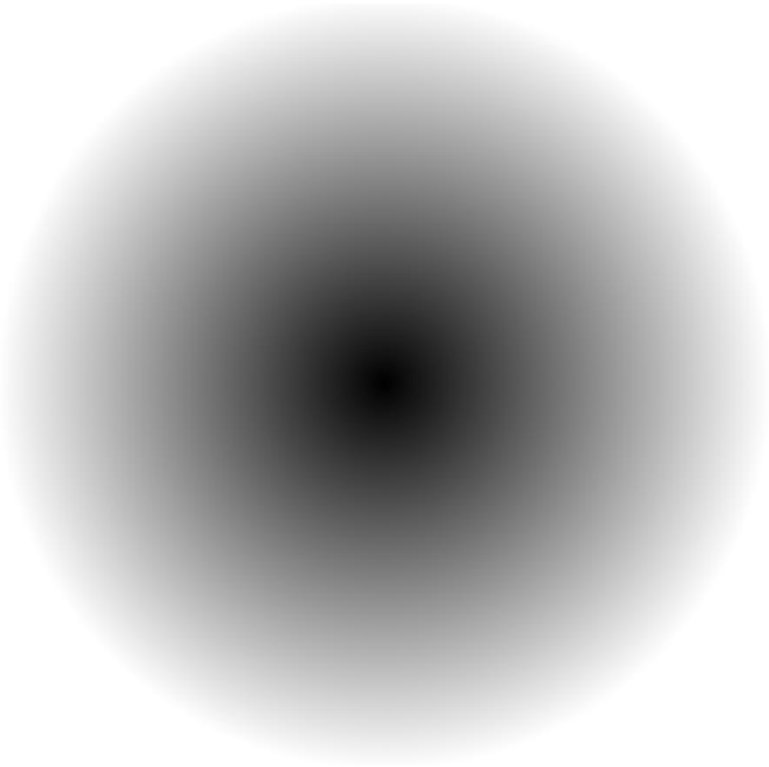
Distance function – 2D

- まずは 2D で考える
- 2D の場合 raymarching は不要

Distance function – 2D

- 最も単純な distance function
- 画面中心からの距離を可視化

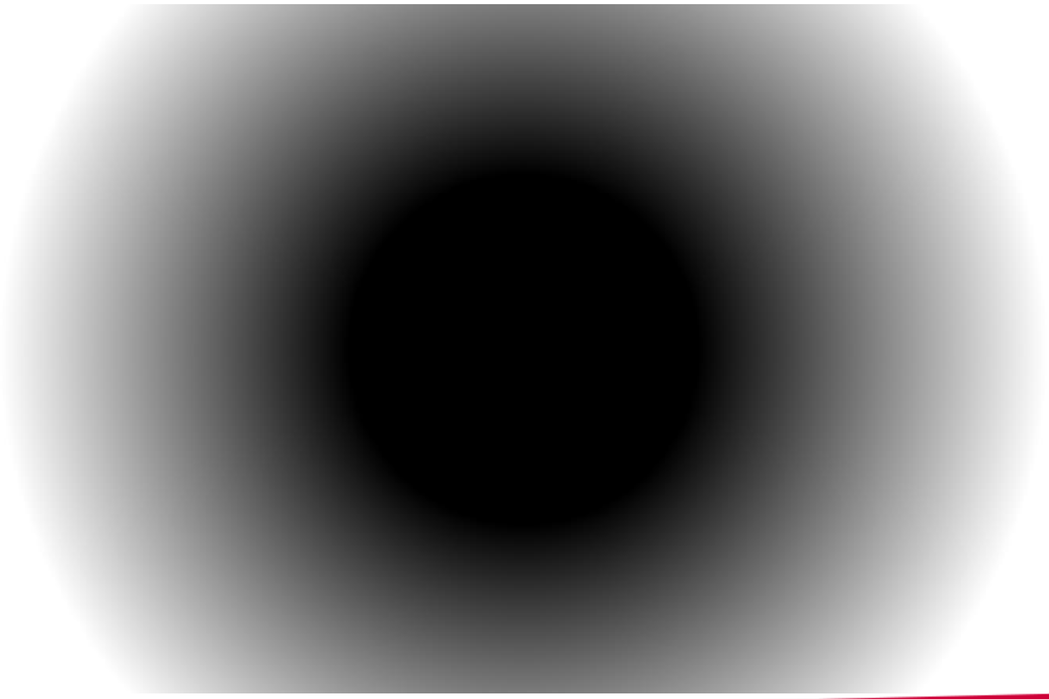
```
float distance = length(pos);
```



Distance function – 2D

- 円の distance function
  - 円の外周からの距離を可視化 (内側は負)

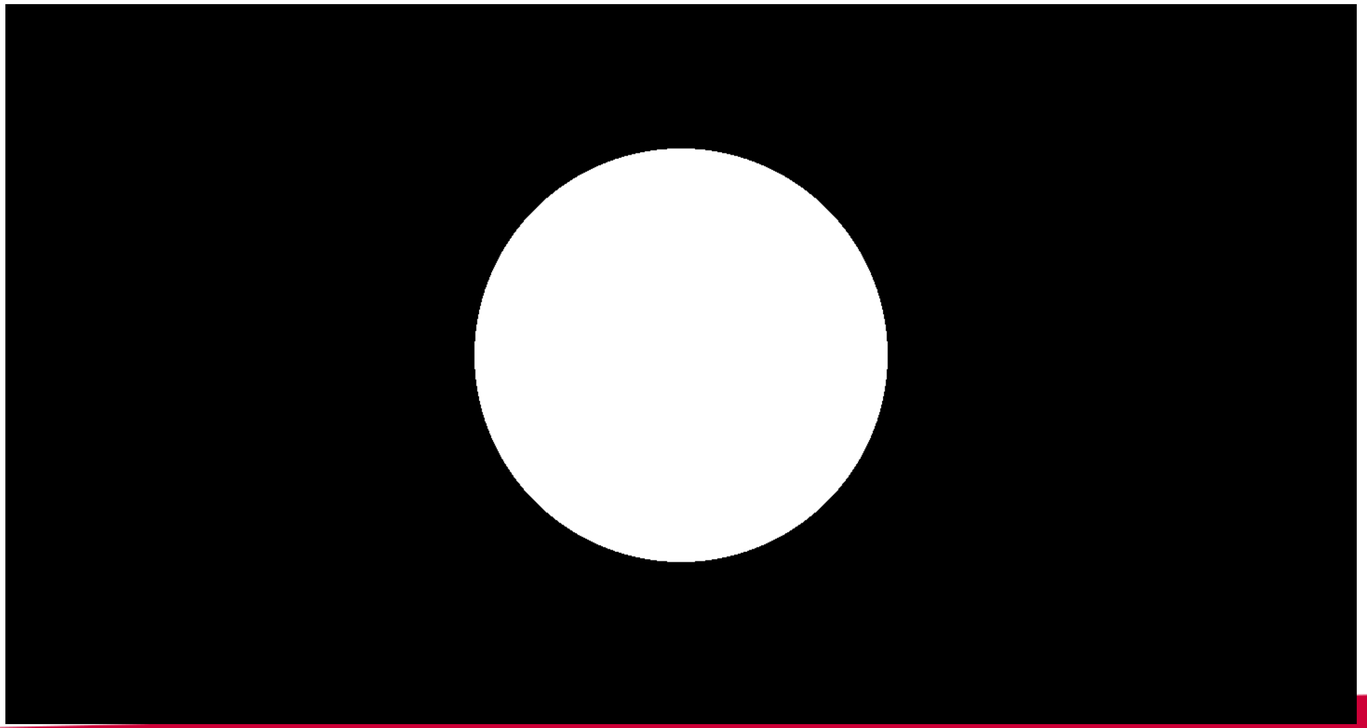
```
float circle(vec2 pos, float radius)
{
        return length(pos) - radius;
}
```



Distance function – 2D

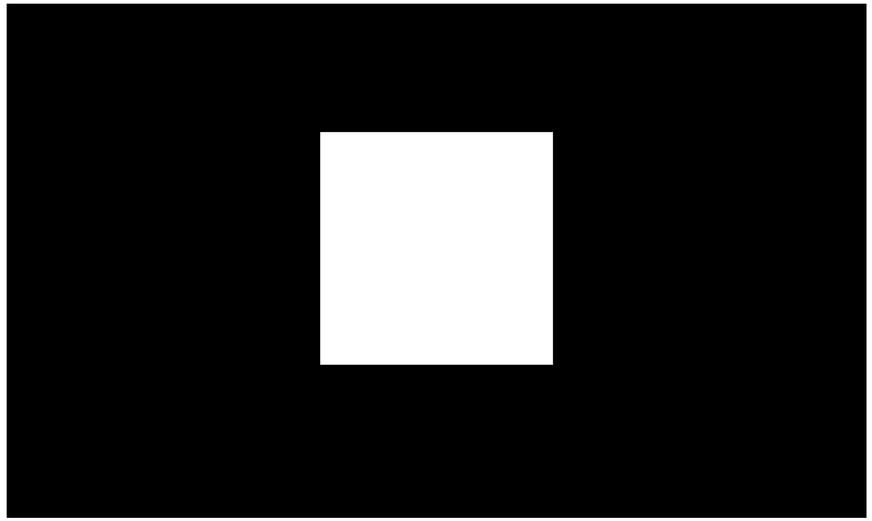
- ここから先はピクセルが図形の内側なら白、外側なら黒として表示

```
float circle(vec2 pos, float radius)
{
        return length(pos) - radius;
}
```

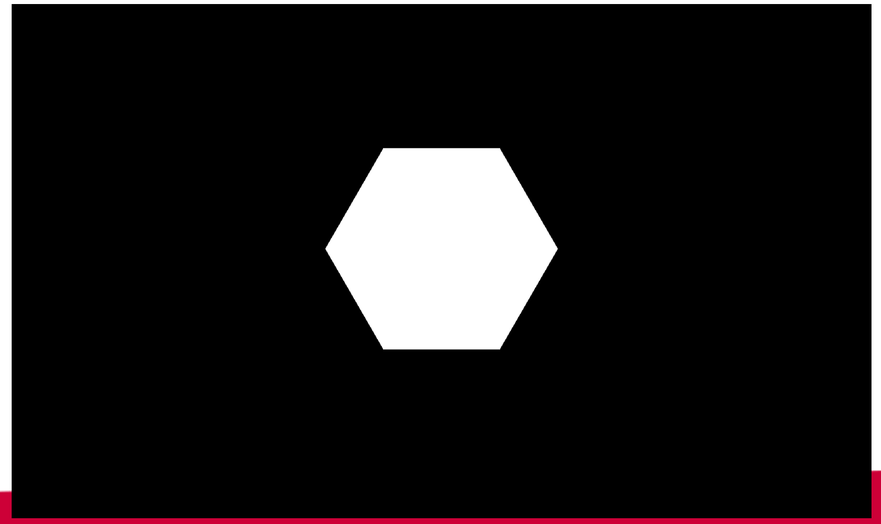


Distance function – 2D

```
float box( vec2 p, vec2 b )
{
    vec2 d = abs(p) - b;
    return min(max(d.x,d.y),0.0) + length(max(d,0.0));
}
```



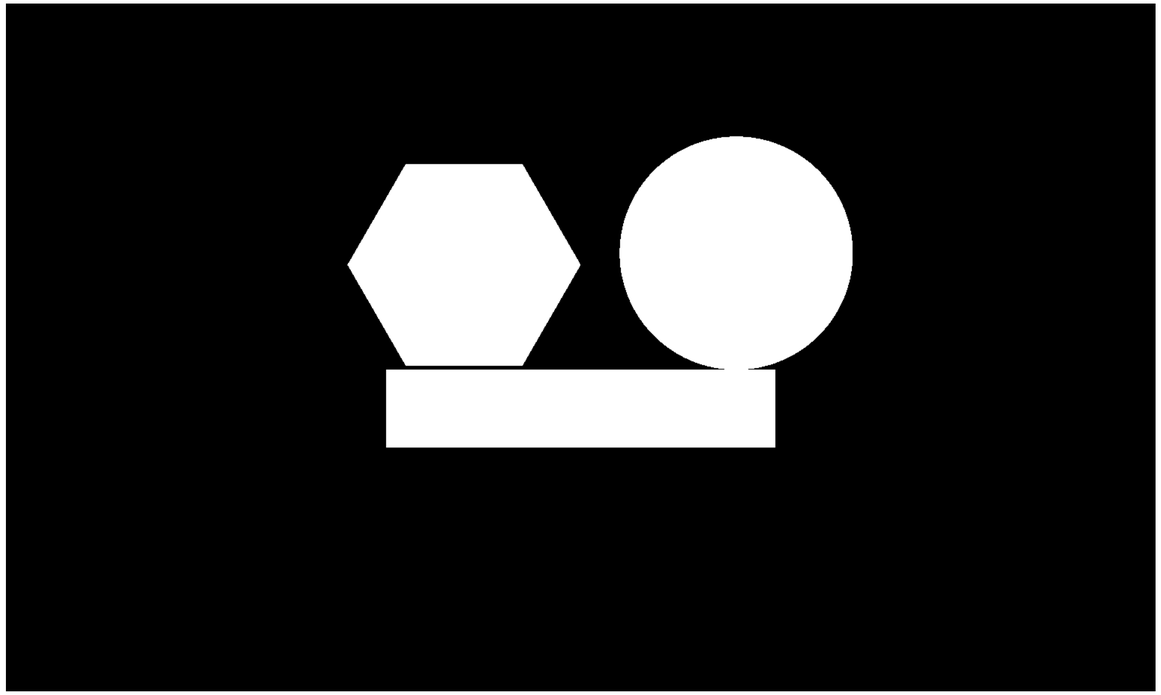
```
float hex( vec2 p, vec2 h )
{
    vec2 q = abs(p);
    return max(q.x-h.y,max(q.x+q.y*0.57735,q.y*1.1547)-h.x);
}
```



Distance function – 2D

- 図形の合成
  - 複数の図形との距離の min() をとる

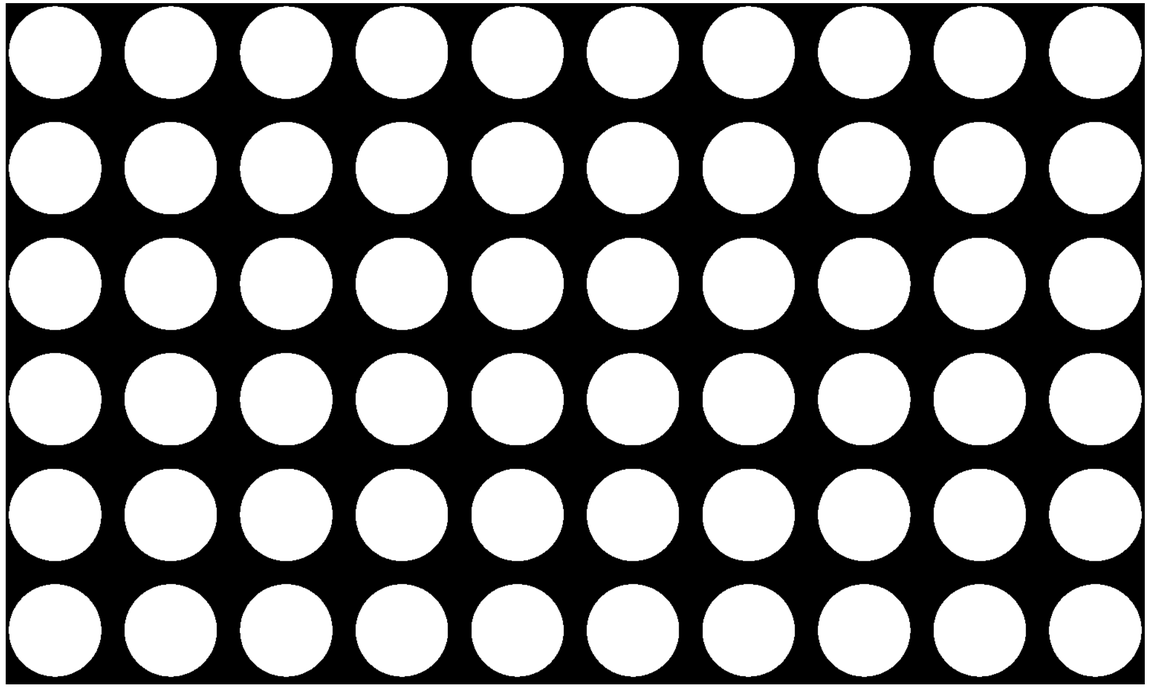
```
float d1 = circle(pos + vec2(-0.4, -0.2), 0.3);
float d2 = box(pos + vec2(0.0, 0.2), vec2(0.5, 0.1));
float d3 = hex(pos + vec2(0.3, -0.17), vec2(0.3, 0.3));
float distance = min(d1, min(d2, d3));
```



Distance function – 2D

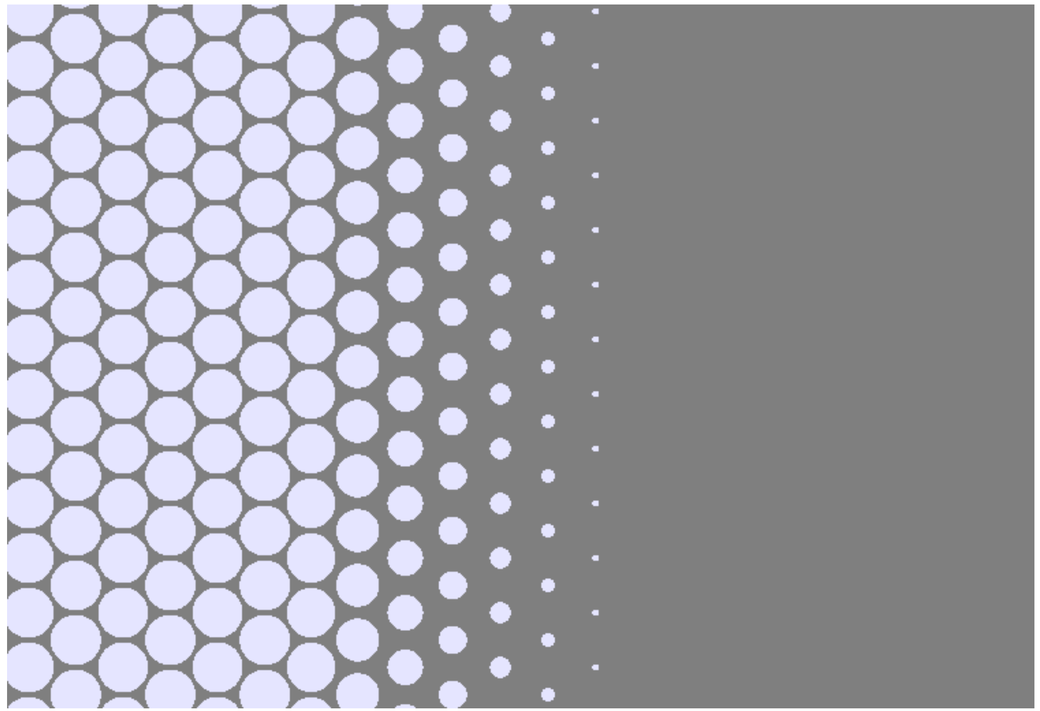
- 繰り返し
  - mod() で空間を折りたたむ

```
pos = mod(pos, vec2(0.3)) - 0.15;
float distance = circle(pos, 0.12);
```

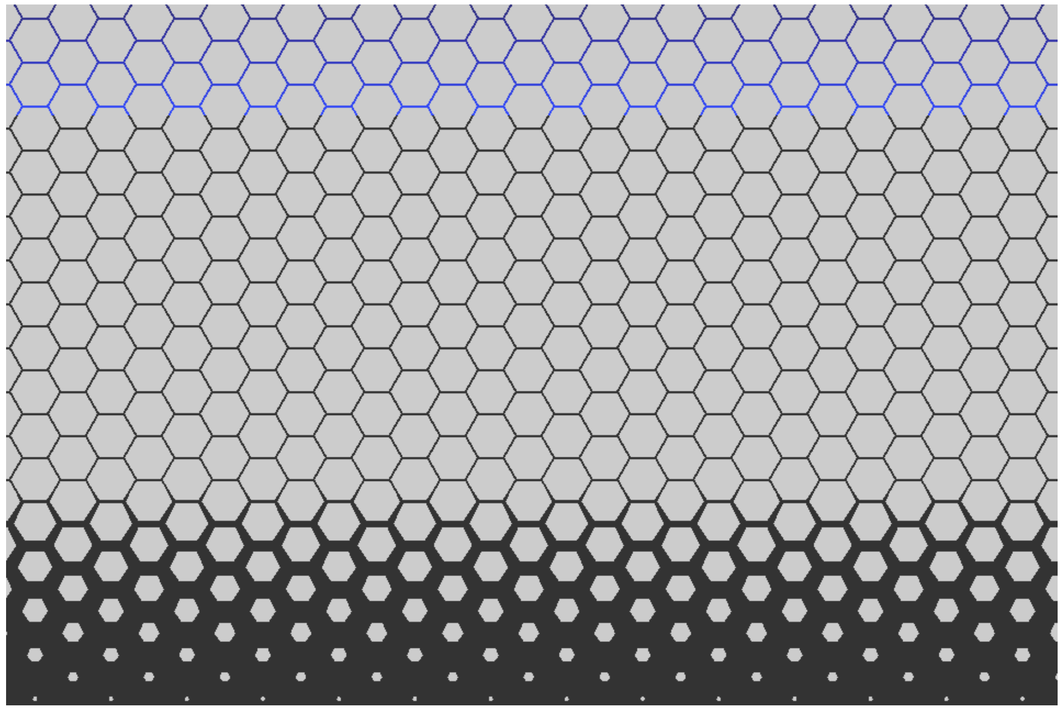


Distance function – 2D

- アニメーション
  - 時間に応じてパラメータ変更



http://glslsandbox.com/e#34782.0



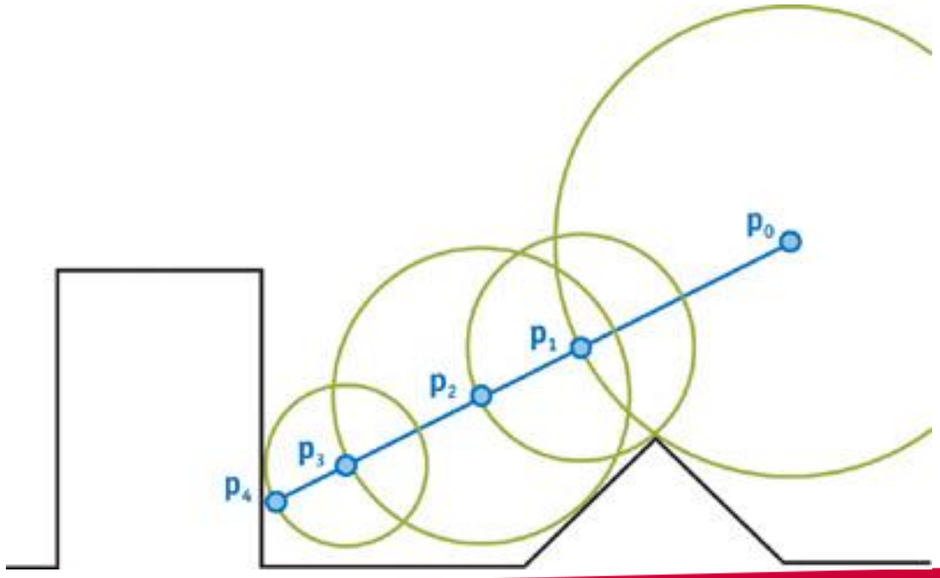
http://glslsandbox.com/e#34809.0

Distance function – 2D

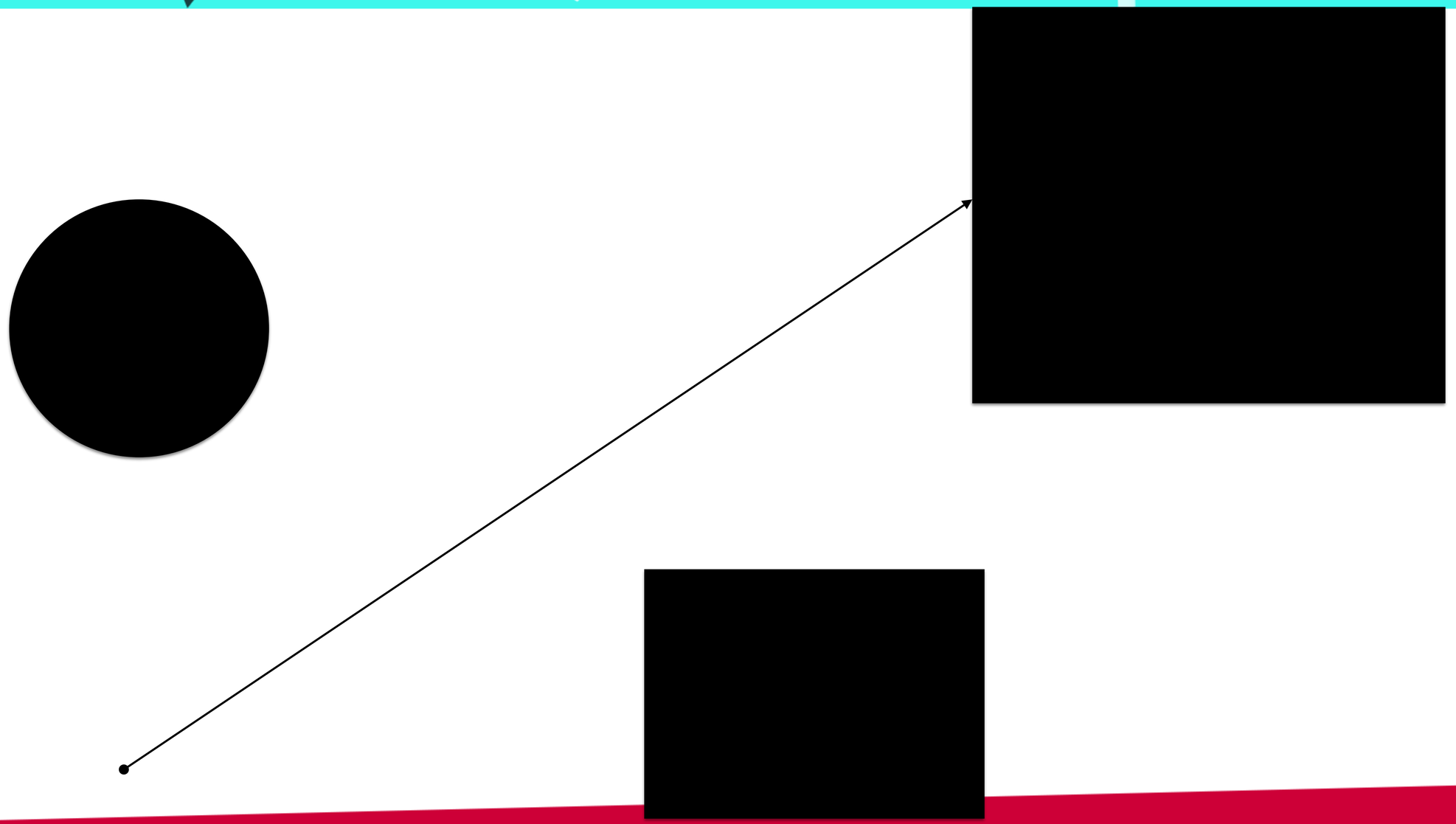
- このように単純な図形 (式) の組み合わせでシーンを構成していく
- 3D でも考え方は同じ
  - 可視化のプロセスが変わるだけ

Distance function – 3D

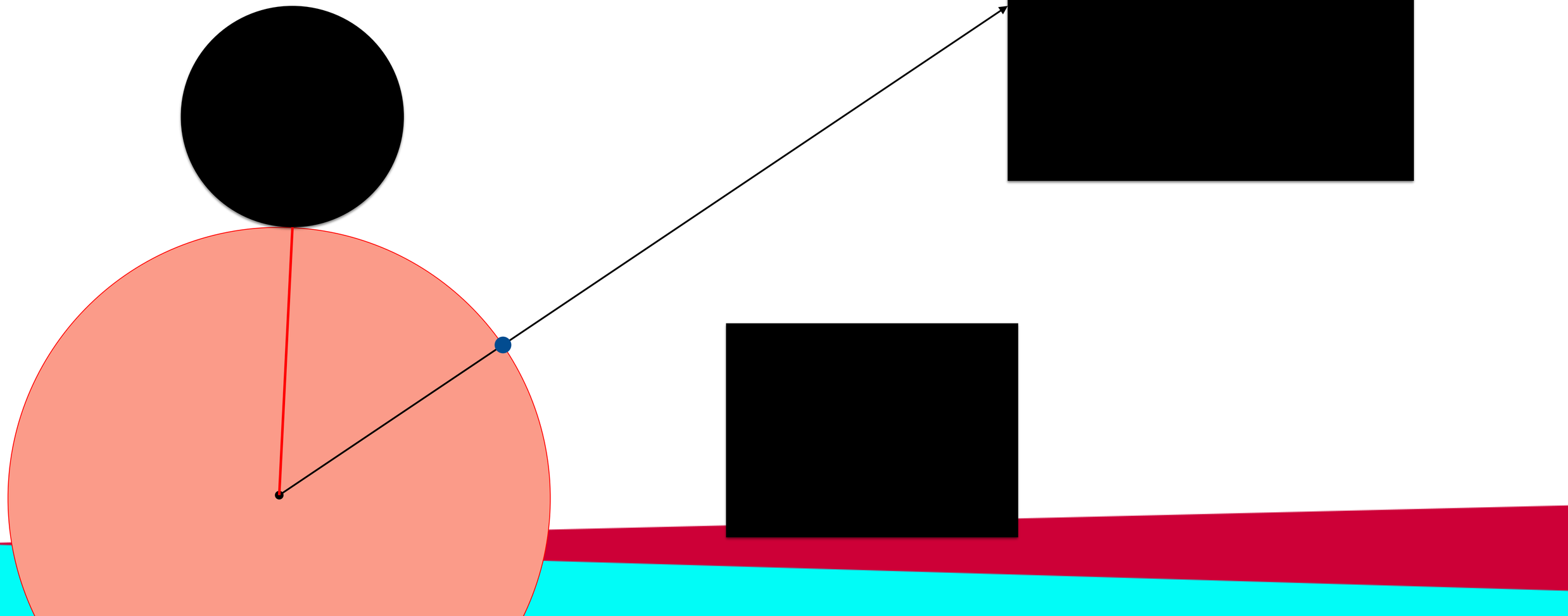
- カメラからレイを飛ばして distance function との交差点を求める
- 3D だと一発で求めるのが困難
- distance function を何度も計算することで漸近的に求める
  - このプロセスが raymarching



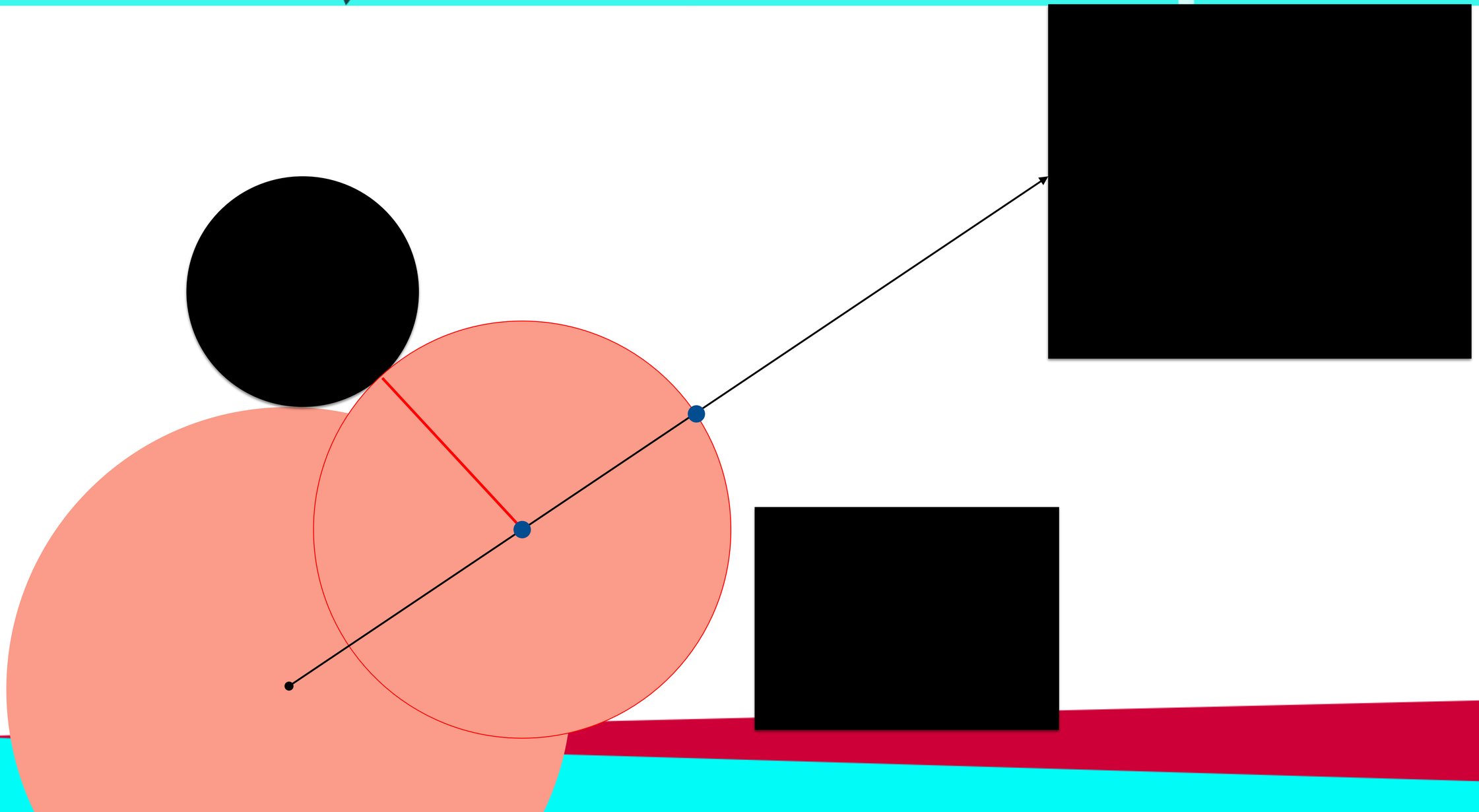
Raymarching



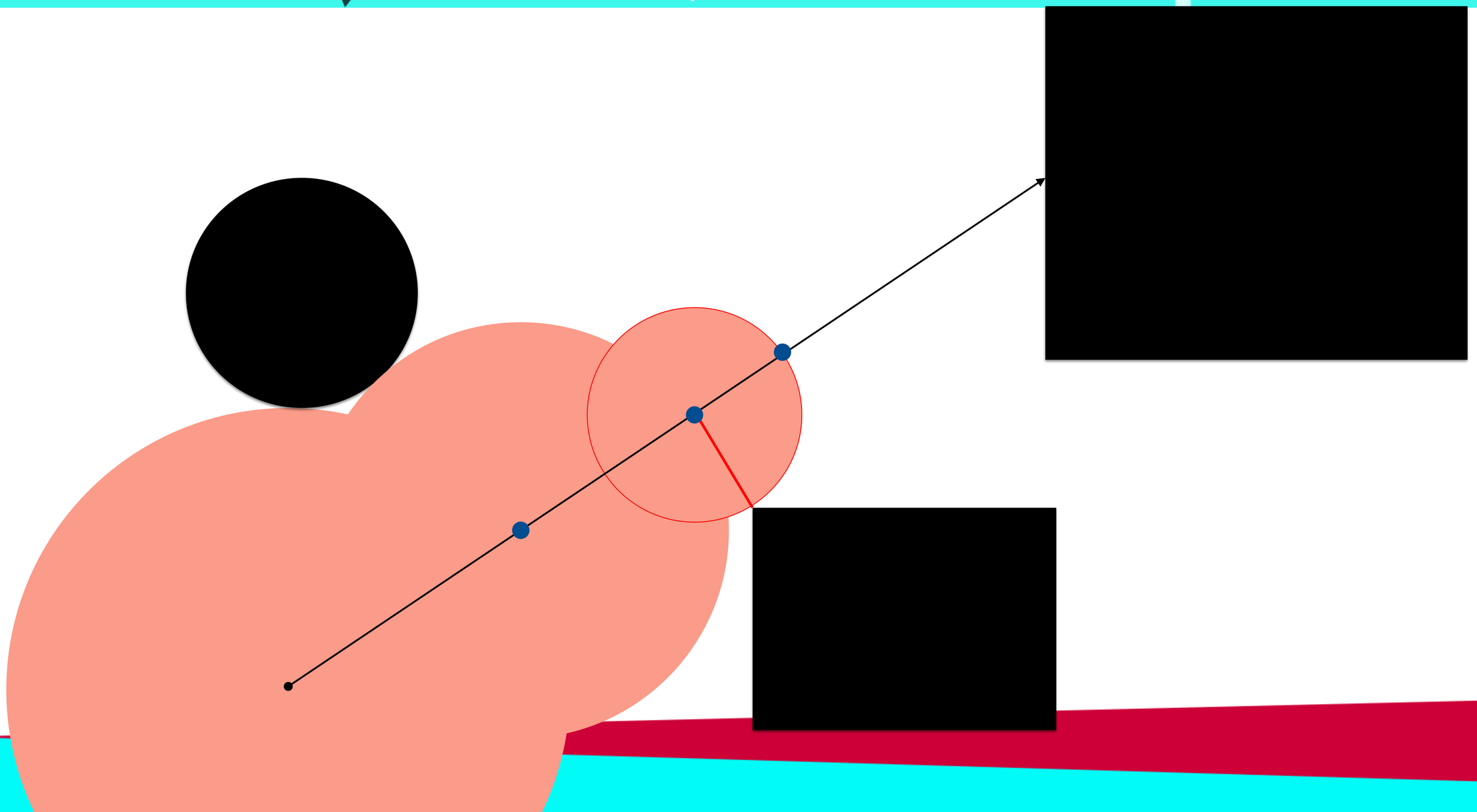
Raymarching



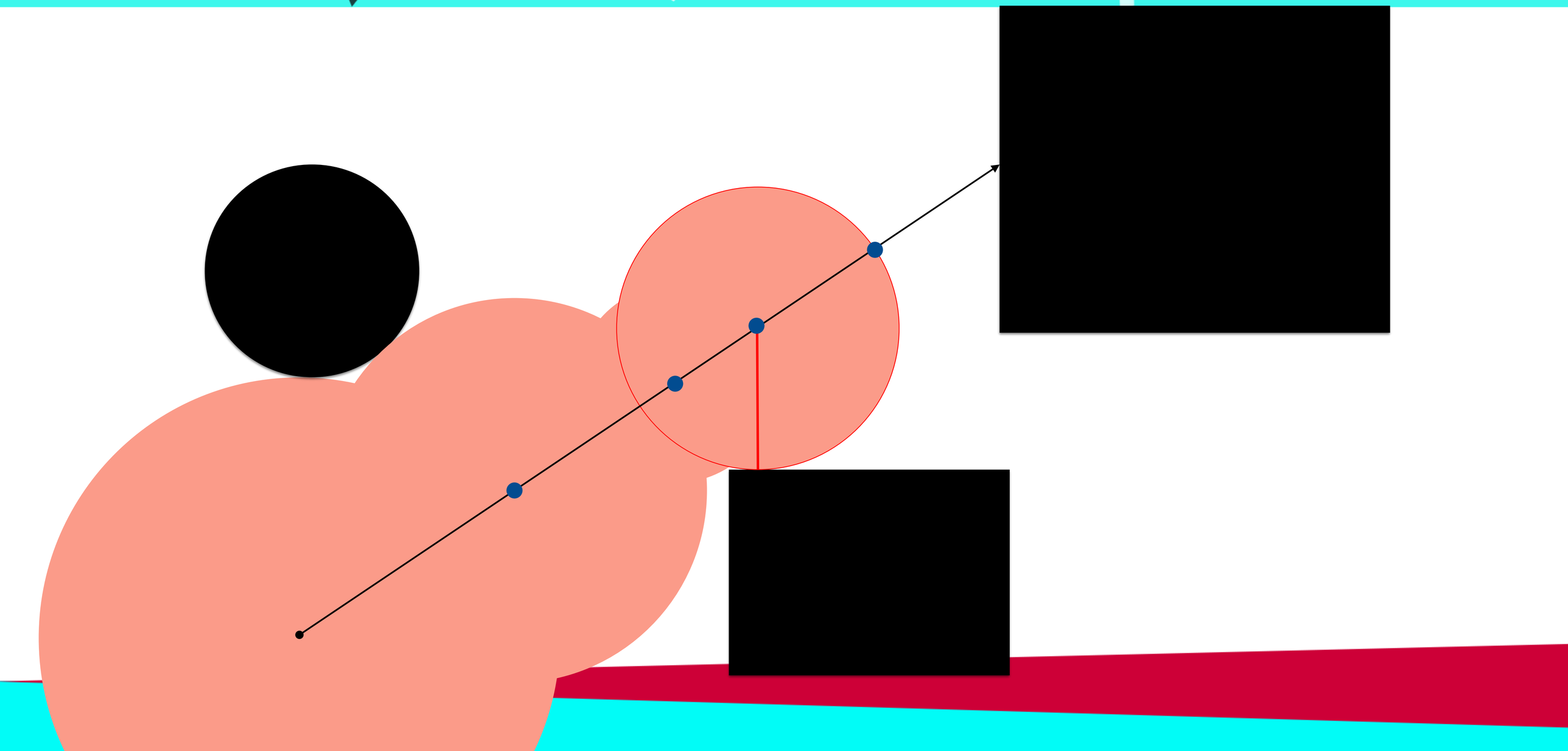
Raymarching



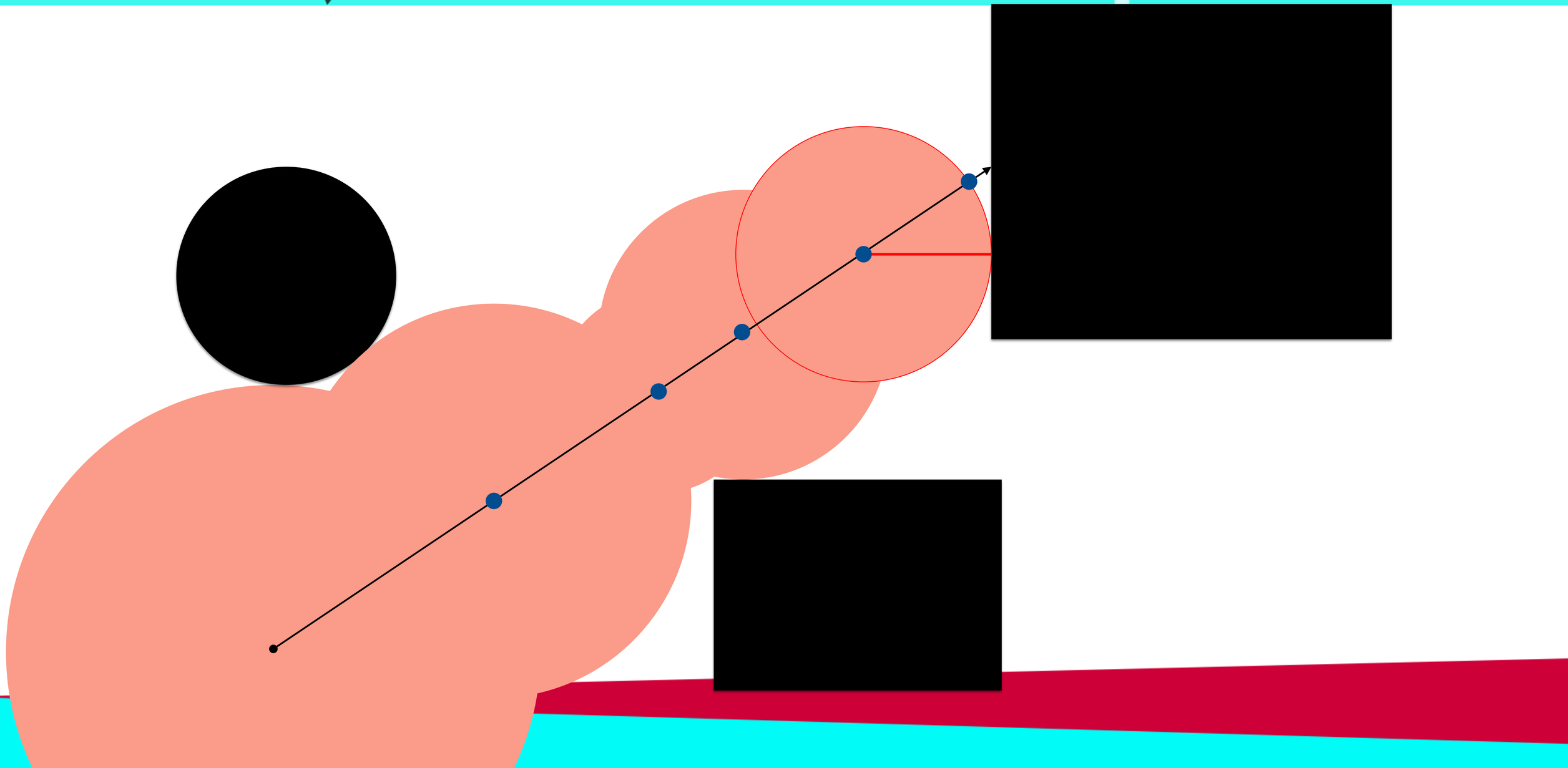
Raymarching



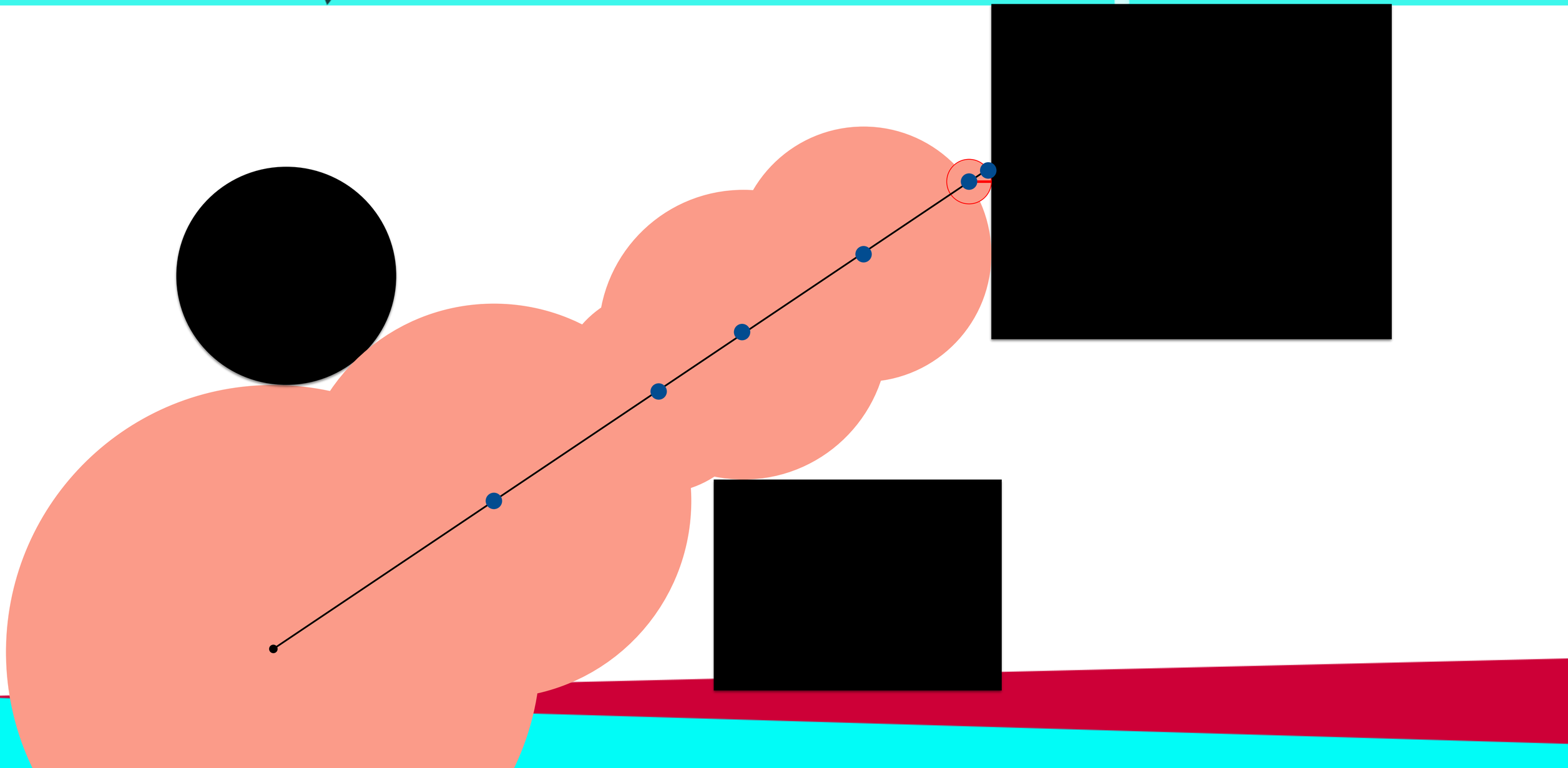
Raymarching



Raymarching



Raymarching



Raymarching

- 十分に距離が小さくなったら計算を打ち切る
  - もしくは上限マーチ回数を超えたら打ち切る
  - 完全に正確な交差点は求まらないが、必要十分
- この計算を全てのピクセルに対して行う
- distance function の複雑さ * マーチ回数 に応じて負荷が増大
- 特徴的なレイの進み方から Sphere Tracing と呼ぶことも

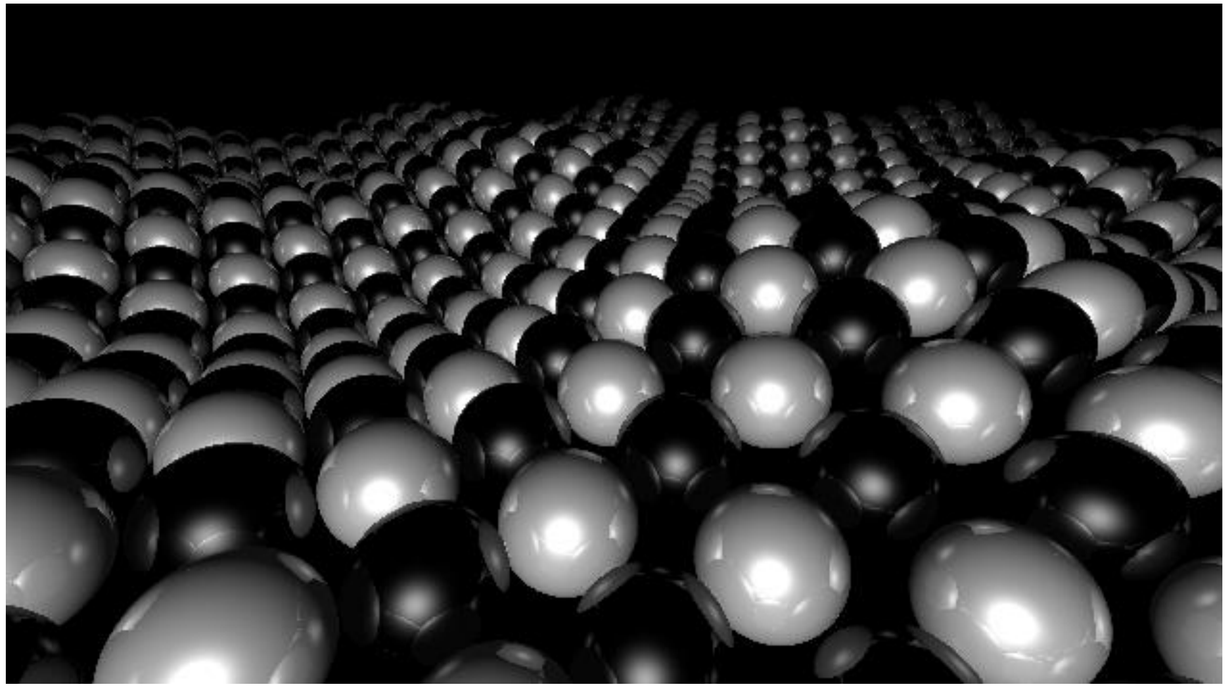
Raymarching

- 法線を求めることも可能
- 交差点からx,y,z軸それぞれ前後に少しずらした地点との距離を計算
- 勾配から法線を推測

```
vec3 guess_normal(vec3 p)
{
    const float d = 0.001;
    return normalize( vec3(
        distance_function(p+vec3(  d,0.0,0.0)) - distance_function(p+vec3( -d,0.0,0.0)),
        distance_function(p+vec3(0.0,  d,0.0)) - distance_function(p+vec3(0.0, -d,0.0)),
        distance_function(p+vec3(0.0,0.0,  d)) - distance_function(p+vec3(0.0,0.0, -d)) ));
}
```

Raymarching

- 法線が求まるのでライティングも可能
- レイ→法線 の反射方向へ再度レイマーチで反射も表現可能



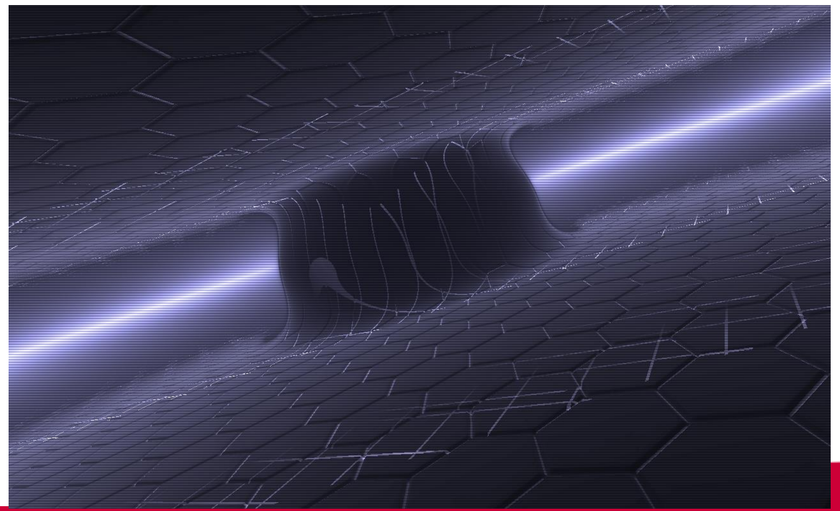
Raymarching

- ポリゴンでは難しいモデルの合成が簡単に実現できる

```
float subtract( float d1, float d2 )
{
    return max(-d1,d2);
}
```

```
float and( float d1, float d2 )
{
    return max(d1,d2);
}
```

```
float soft_min(float d1, float d2, float r)
{
    float e = max(r - abs(d1 - d2), 0.0);
    return min(d1, d2) - e*e*0.25 / r;
}
```



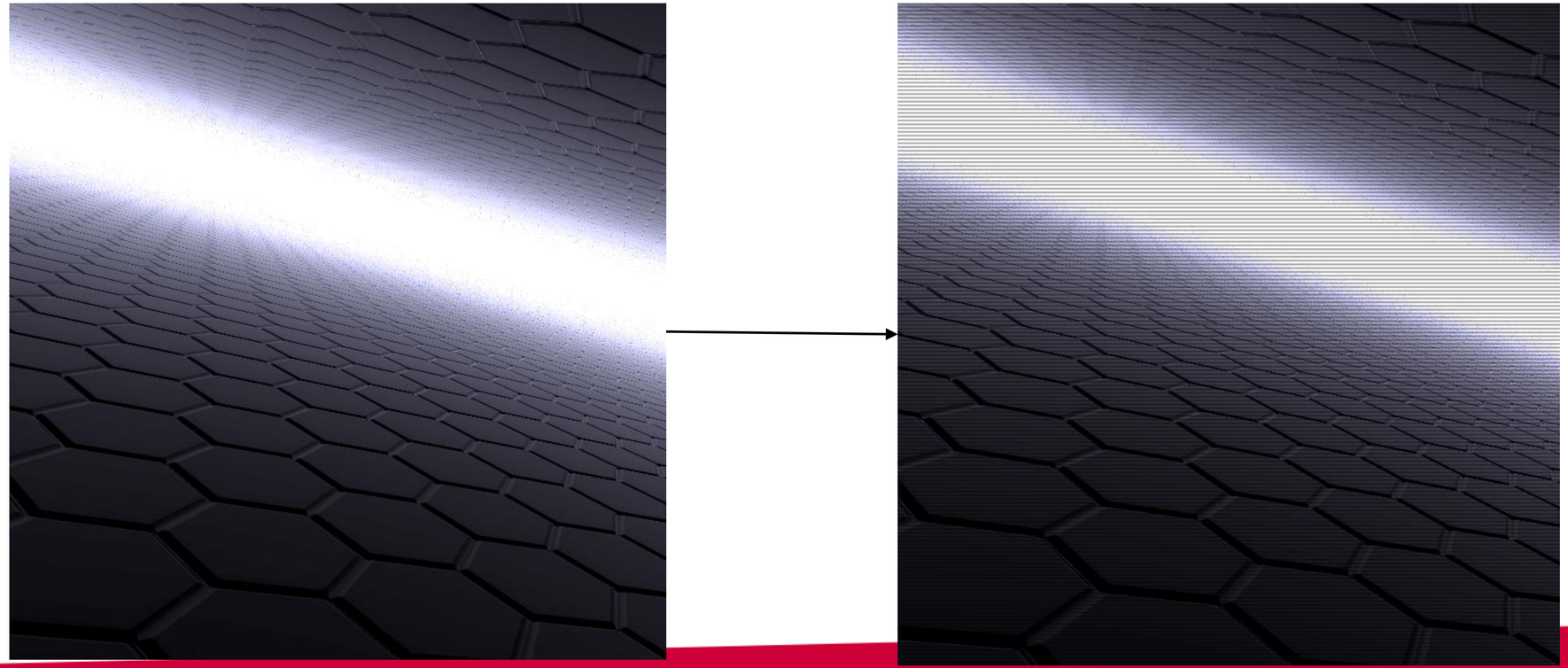
http://glslsandbox.com/e#27204

Effects

- 容量の都合でエフェクト類も大抵 1 pass でできるものに限られる
  - ブルームなどは厳しい

Effects

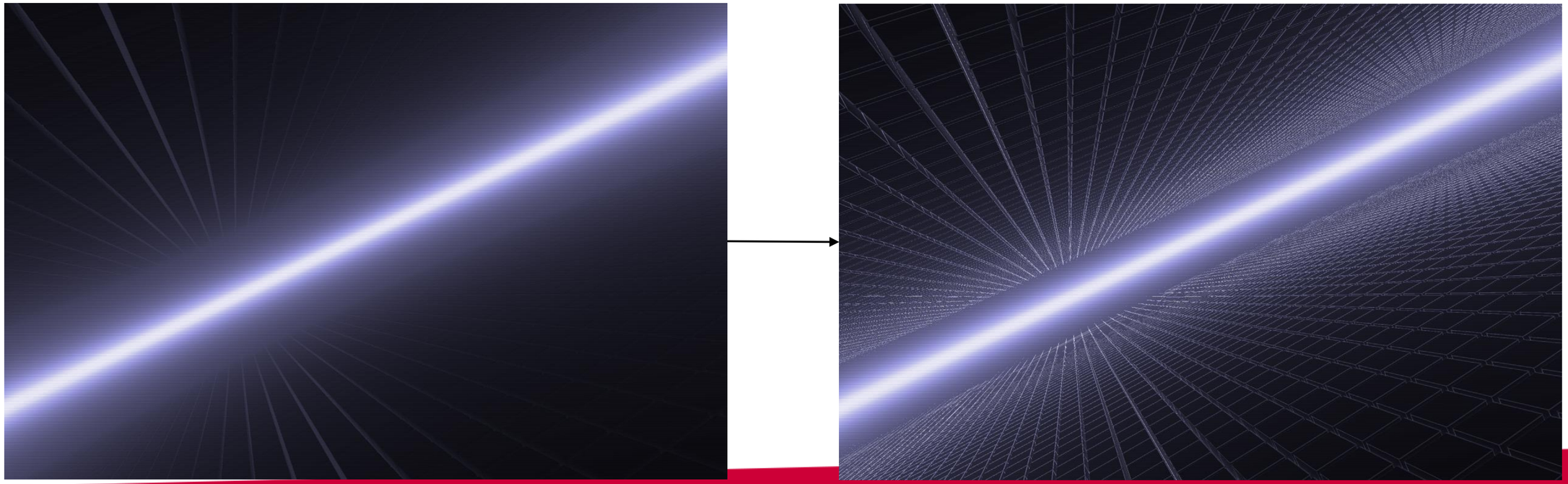
- 走査線
  - 縦数ピクセル毎に色を暗くする



http://glslsandbox.com/e#34783.1

Effects

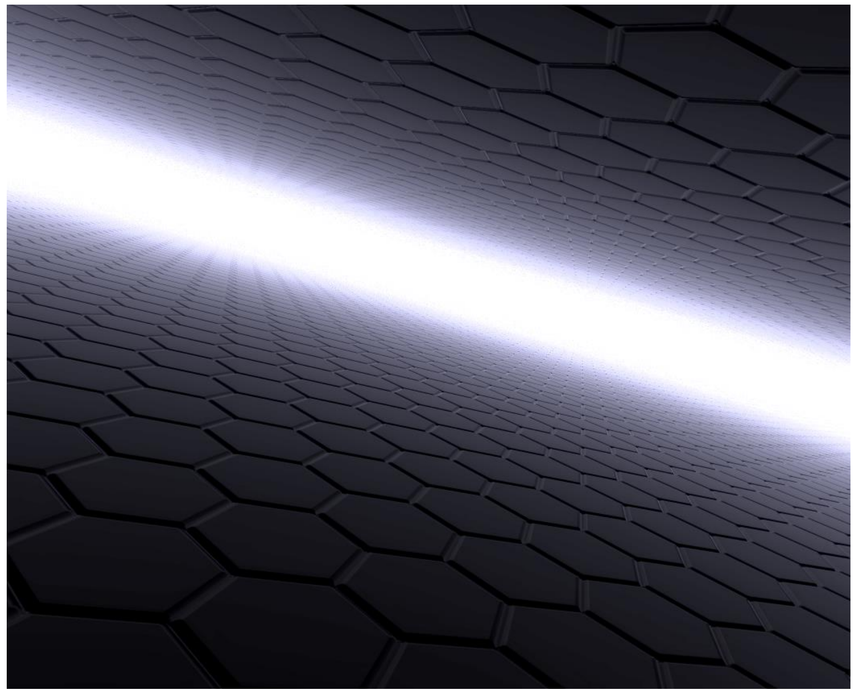
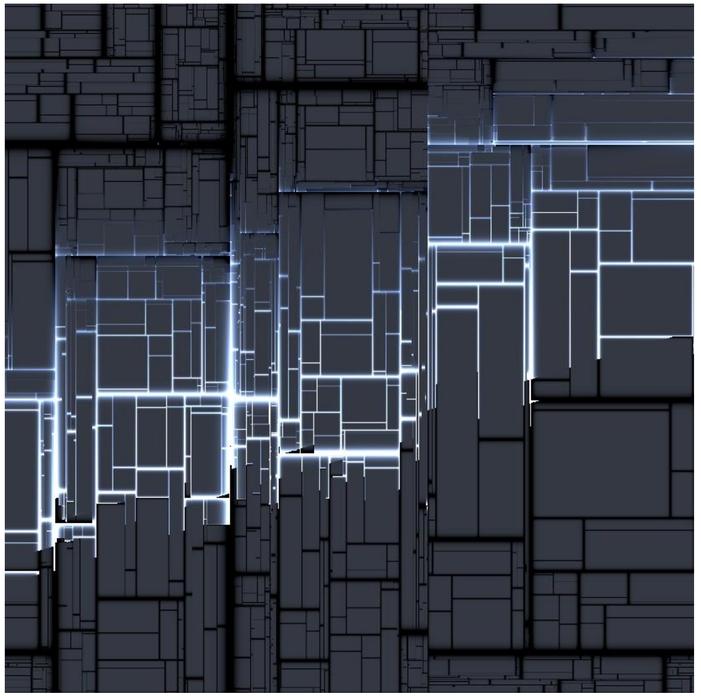
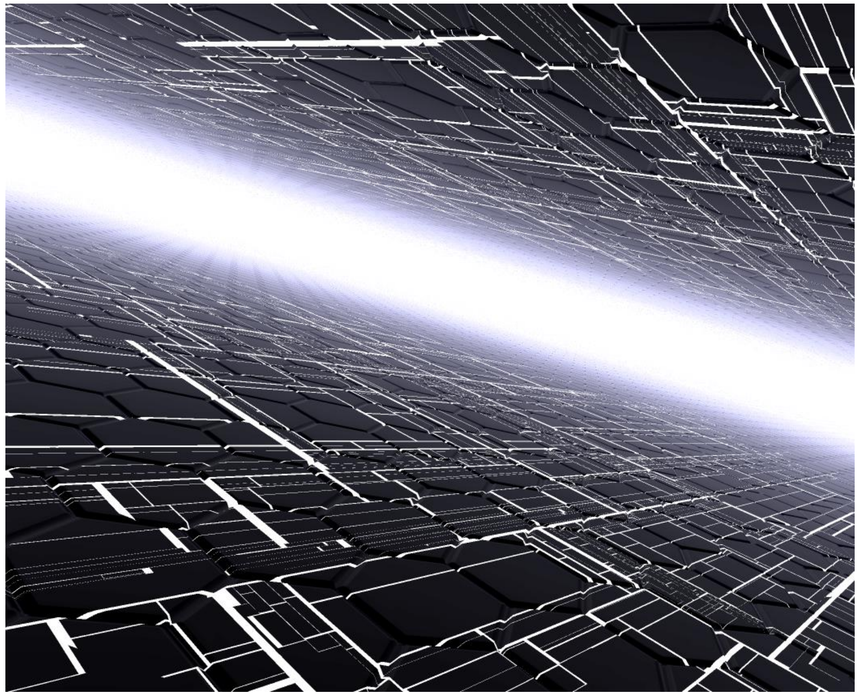
- エッジを強調
  - 周囲の法線を算出してある程度以上向きが違ったら色を変える
  - Raymarching だとピクセルの周辺の法線も求められるため 1 pass で可能



http://glslsandbox.com/e#34785.0

Effects

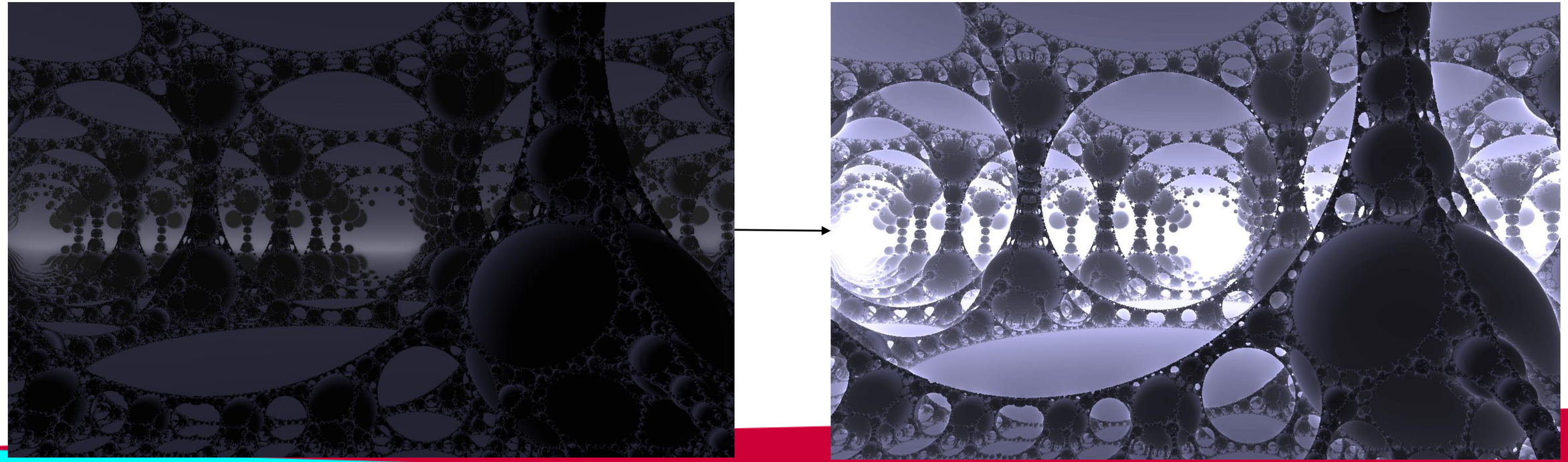
- テクスチャ投影
  - レイ到達点の位置、法線などを元に模様の式を計算、結果を合成


+

=


http://glslsandbox.com/e#21290.5 http://glslsandbox.com/e#34783.1

Effects

- オーラエフェクト
  - Raymarching のマーチ回数に応じて適当に明るくする
  - 輪郭の外側はマーチ回数が多くなるためオーラのように見える



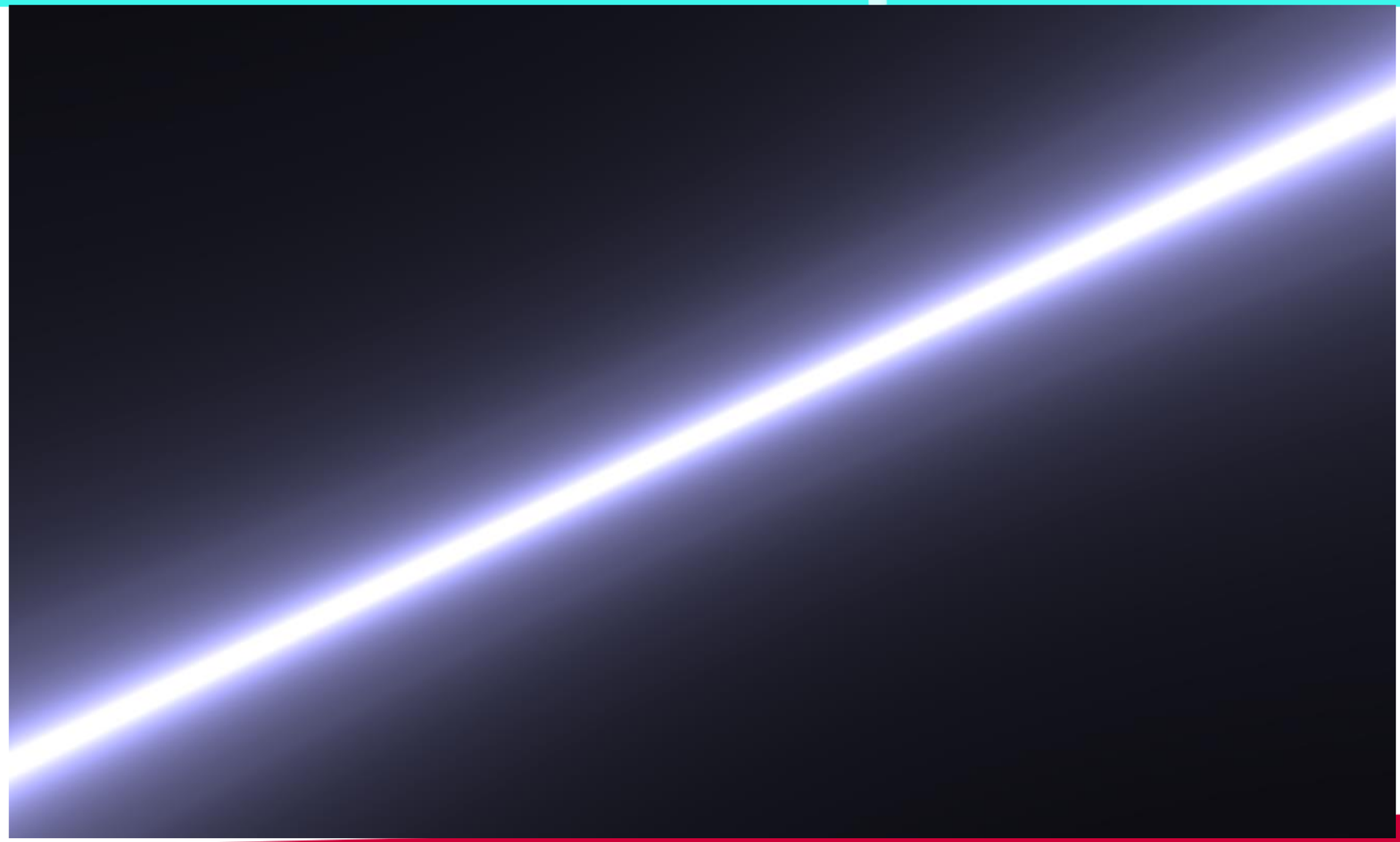
http://glslsandbox.com/e#23658

アニメーション

- 時間パラメータさえあれば必要十分
  - 時間に応じて図形を動かしたりパラメータを変えるだけ
- 時間からシーンの全てが決まるようにする
  - 音楽との動機を保つため
  - 処理落ちが日常的に発生しうるため、フレーム数や前フレームの結果に依存するような処理は望ましくない

作例

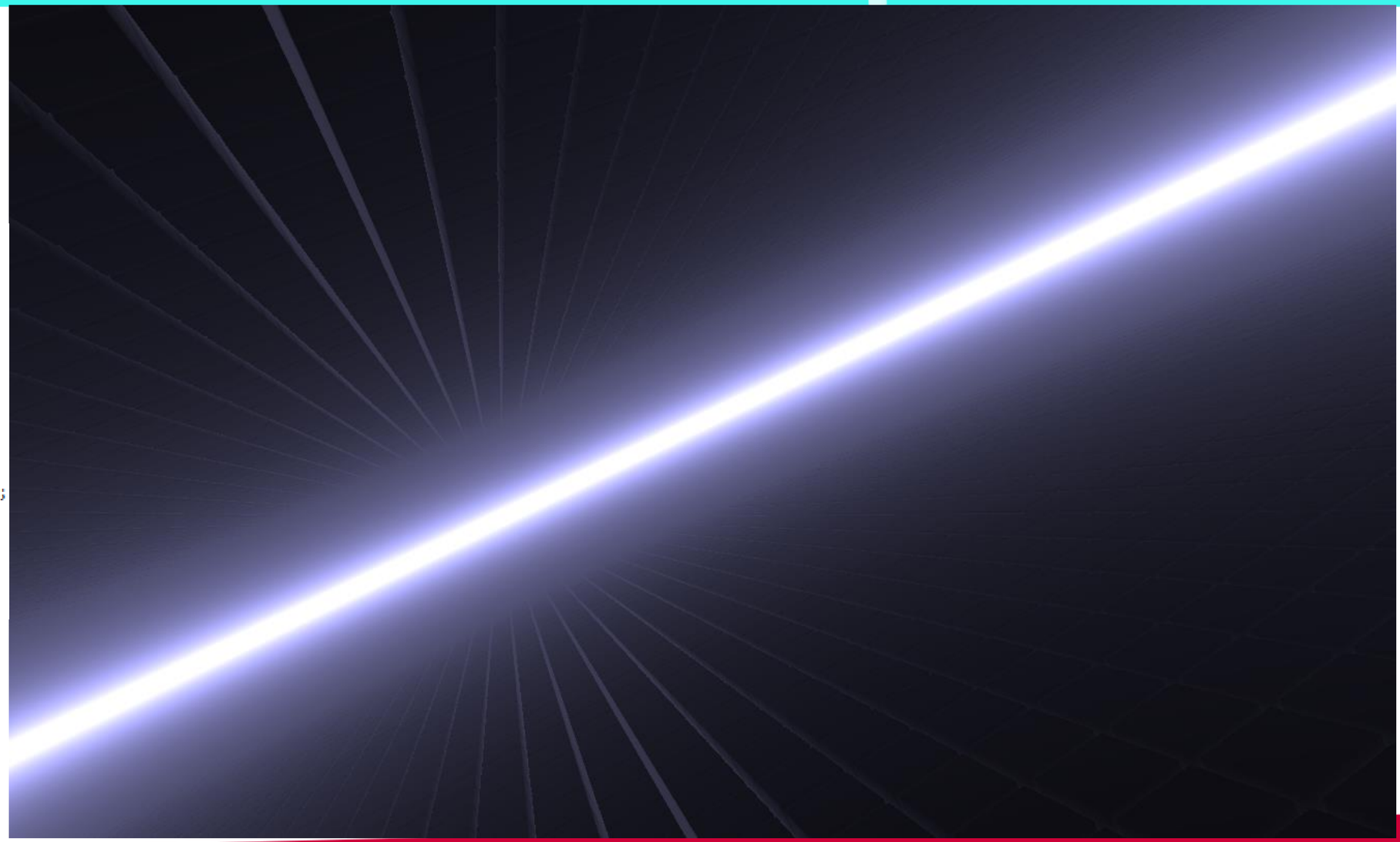
```
float map(vec3 p)
{
    float h = 1.8;

    p = -abs(p);
    float d1 = p.y + h;
    return d1;
}
```



http://glslsandbox.com/e#10544.1

作例

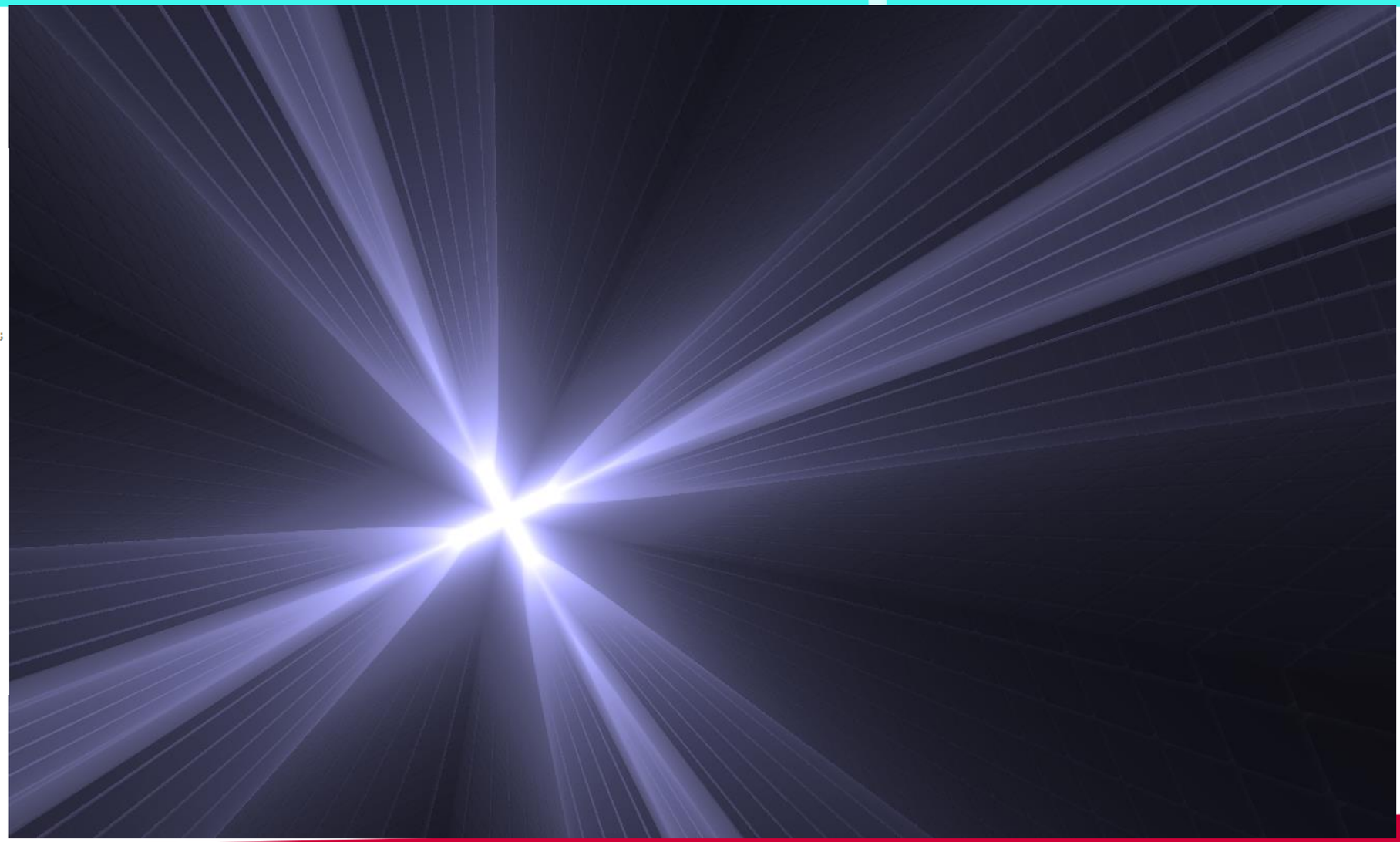
```
float map(vec3 p)
{
    float h = 1.8;
    float grid = 0.4;
    float grid_half = grid*0.5;
    float cube = 0.175;

    p = -abs(p);
    float d1 = p.y + h;
    vec2 p1 = mod(p.xz, vec2(grid)) - vec2(grid_half);
    float c1 = sdBox(p1,vec2(cube));

    return max(c1,d1);
}
```



http://glslsandbox.com/e#10544.1

作例

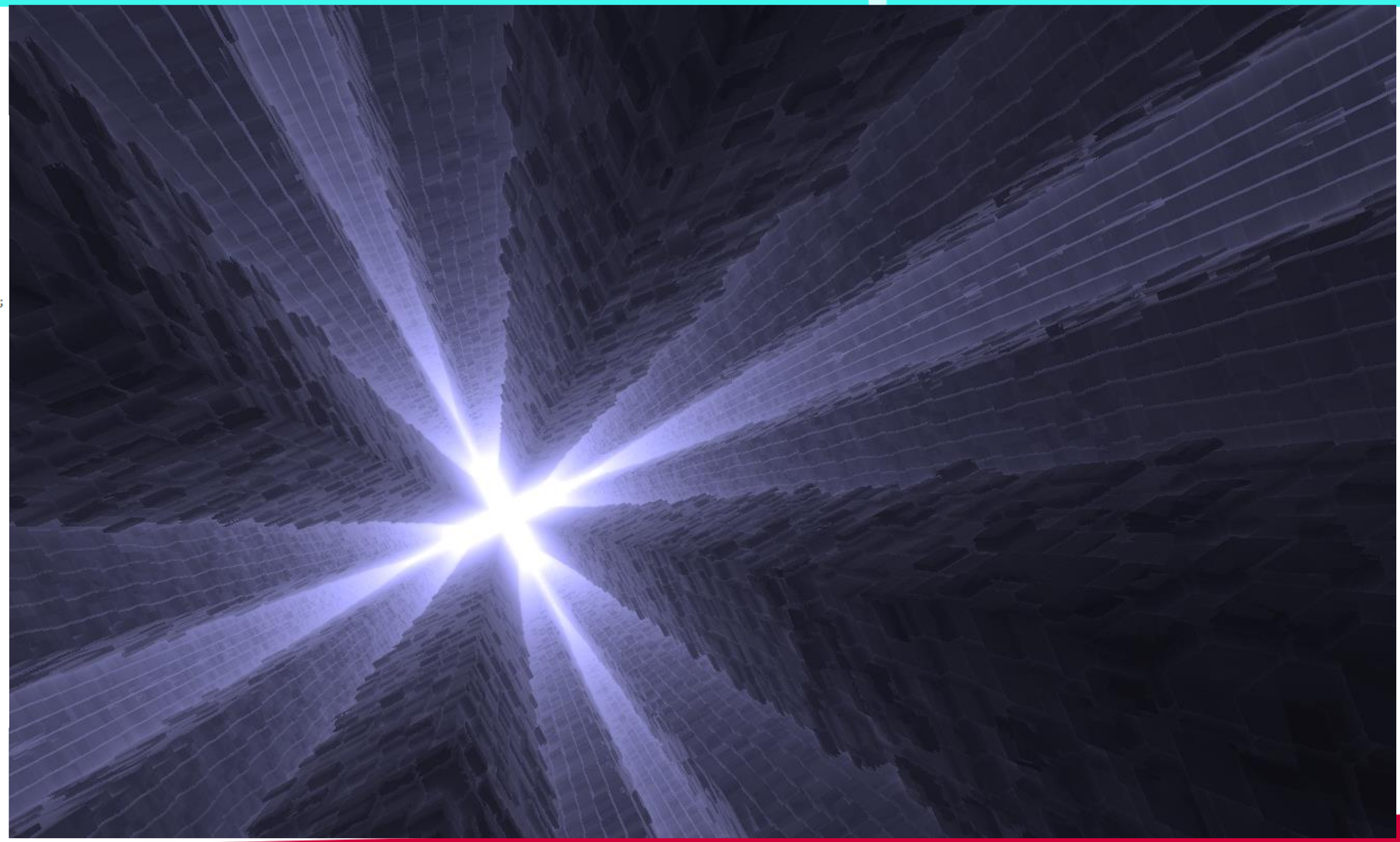
```
float map(vec3 p)
{
    float h = 1.8;
    float rh = 0.5;
    float grid = 0.4;
    float grid_half = grid*0.5;
    float cube = 0.175;
    vec3 orig = p;

    vec3 g1 = vec3(ceil((orig.x)/grid), ceil((orig.y)/grid), ceil((orig.z)/grid));

    p = -abs(p);
    vec3 di = ceil(p/4.8);
    p.y += di.x*1.0;
    p.x += di.y*1.2;
    p.xy = mod(p.xy, -4.8);

    float d1 = p.y + h;
    float d2 = p.x + h;

    vec2 p1 = mod(p.xz, vec2(grid)) - vec2(grid_half);
    float c1 = sdBox(p1,vec2(cube));

    vec2 p2 = mod(p.yz, vec2(grid)) - vec2(grid_half);
    float c2 = sdBox(p2,vec2(cube));

    return max(max(c1,d1), max(c2,d2));
}
```



http://glslsandbox.com/e#10544.2

作例

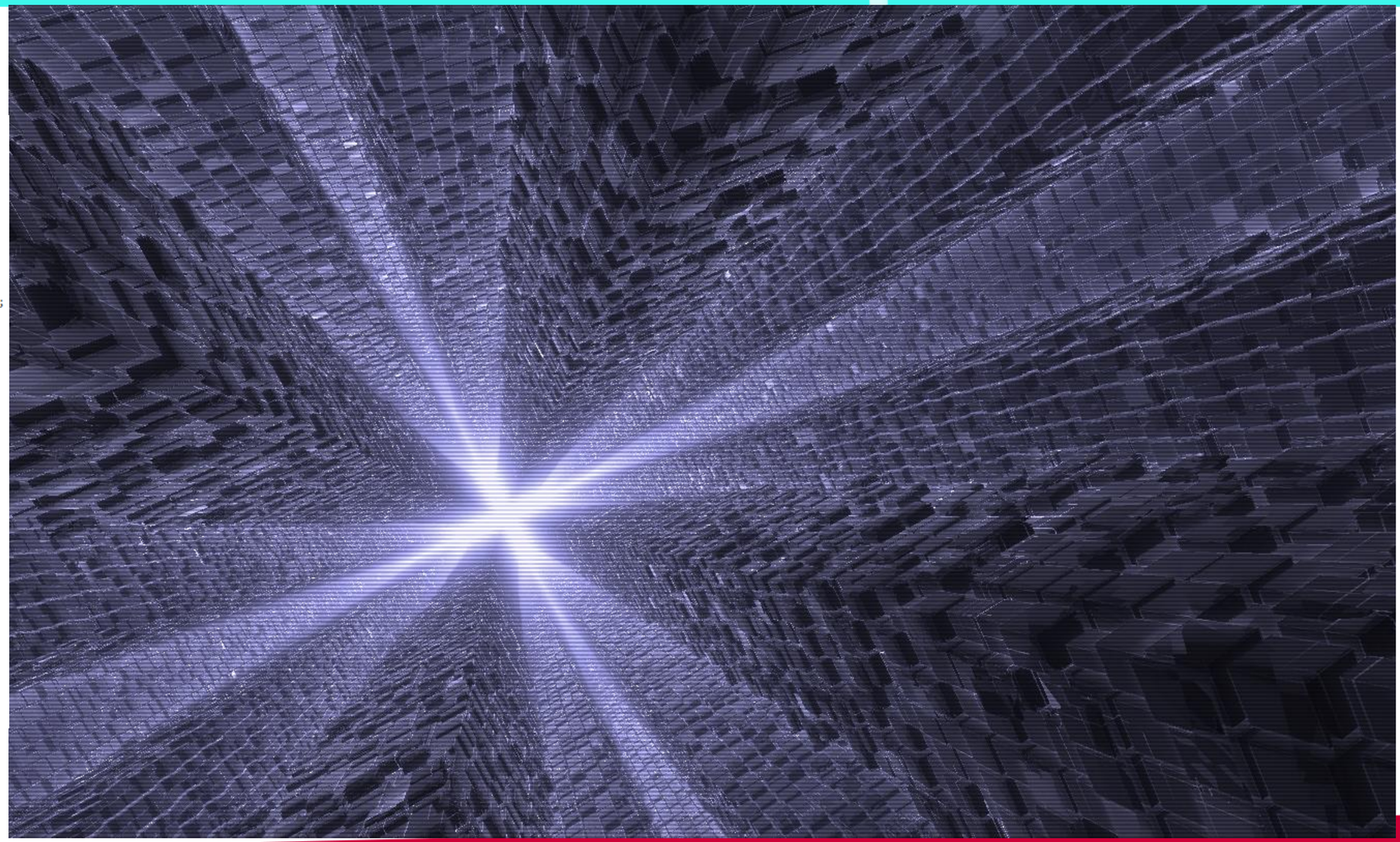
```
float map(vec3 p)
{
    float h = 1.8;
    float rh = 0.5;
    float grid = 0.4;
    float grid_half = grid*0.5;
    float cube = 0.175;
    vec3 orig = p;

    vec3 g1 = vec3(ceil((orig.x)/grid), ceil((orig.y)/grid), ceil((orig.z)/grid));
    vec3 rxz =  nrand3(g1.xz);
    vec3 ryz =  nrand3(g1.yz);

    p = -abs(p);
    vec3 di = ceil(p/4.8);
    p.y += di.x*1.0;
    p.x += di.y*1.2;
    p.xy = mod(p.xy, -4.8);

    vec2 gap = vec2(rxz.x*rh, ryz.y*rh);
    float d1 = p.y + h + gap.x;
    float d2 = p.x + h + gap.y;

    vec2 p1 = mod(p.xz, vec2(grid)) - vec2(grid_half);
    float c1 = sdBox(p1,vec2(cube));

    vec2 p2 = mod(p.yz, vec2(grid)) - vec2(grid_half);
    float c2 = sdBox(p2,vec2(cube));

    return max(max(c1,d1), max(c2,d2));
}
```



http://glslsandbox.com/e#10544.3

作例

```
float map(vec3 p)
{
    float h = 1.8;
    float rh = 0.5;
    float grid = 0.4;
    float grid_half = grid*0.5;
    float cube = 0.175;
    vec3 orig = p;

    vec3 g1 = vec3(ceil((orig.x)/grid), ceil((orig.y)/grid), ceil((orig.z)/grid));
    vec3 rxz =  nrand3(g1.xz);
    vec3 ryz =  nrand3(g1.yz);

    p = -abs(p);
    vec3 di = ceil(p/4.8);
    p.y += di.x*1.0;
    p.x += di.y*1.2;
    p.xy = mod(p.xy, -4.8);

    vec2 gap = vec2(rxz.x*rh, ryz.y*rh);
    float d1 = p.y + h + gap.x;
    float d2 = p.x + h + gap.y;

    vec2 p1 = mod(p.xz, vec2(grid)) - vec2(grid_half);
    float c1 = sdBox(p1,vec2(cube));

    vec2 p2 = mod(p.yz, vec2(grid)) - vec2(grid_half);
    float c2 = sdBox(p2,vec2(cube));

    return max(max(c1,d1), max(c2,d2));
}
```



エフェクト追加

http://glslsandbox.com/e#10544

作例

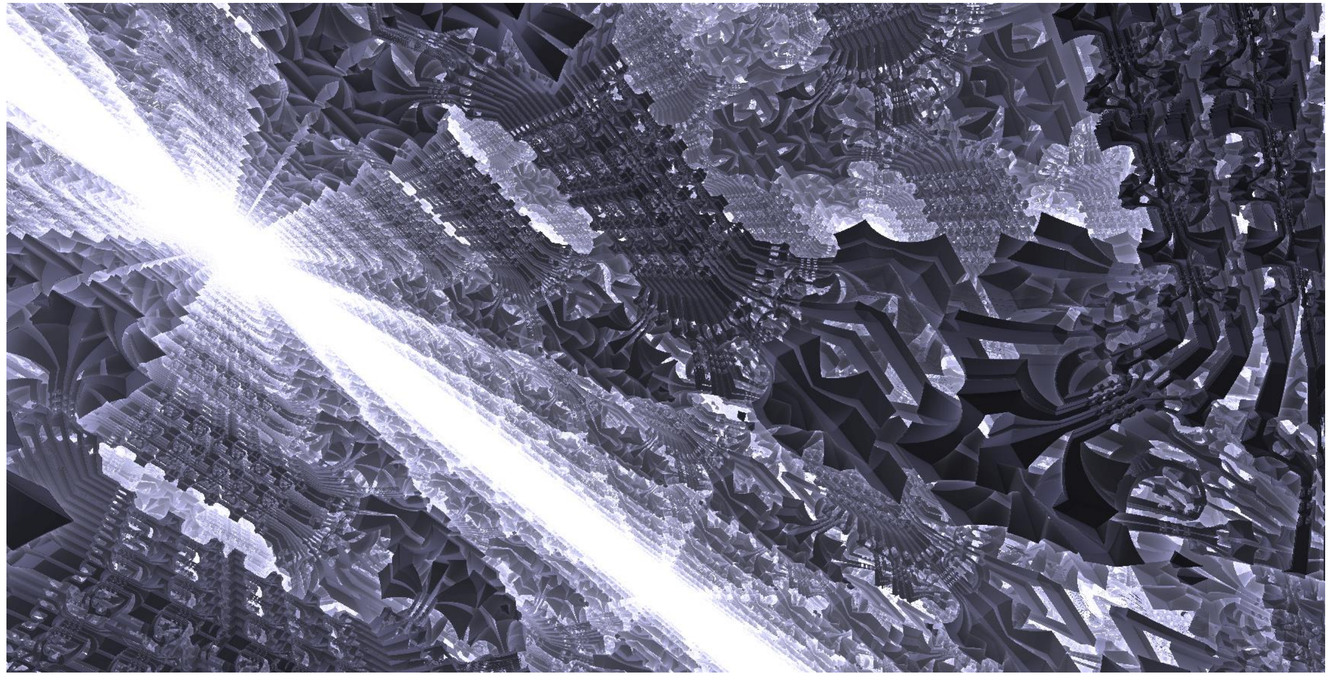
- 必ずしも複雑で難しい distance function を書く必要はない
- 単純な distance function でも見せ方次第で綺麗になる

Fractal

- distance function & raymarching における非常に強力なツール
- 短いコードでとても複雑で美しい図形を表現可能
- ただしとても重く、結果の予測も困難

Fractal

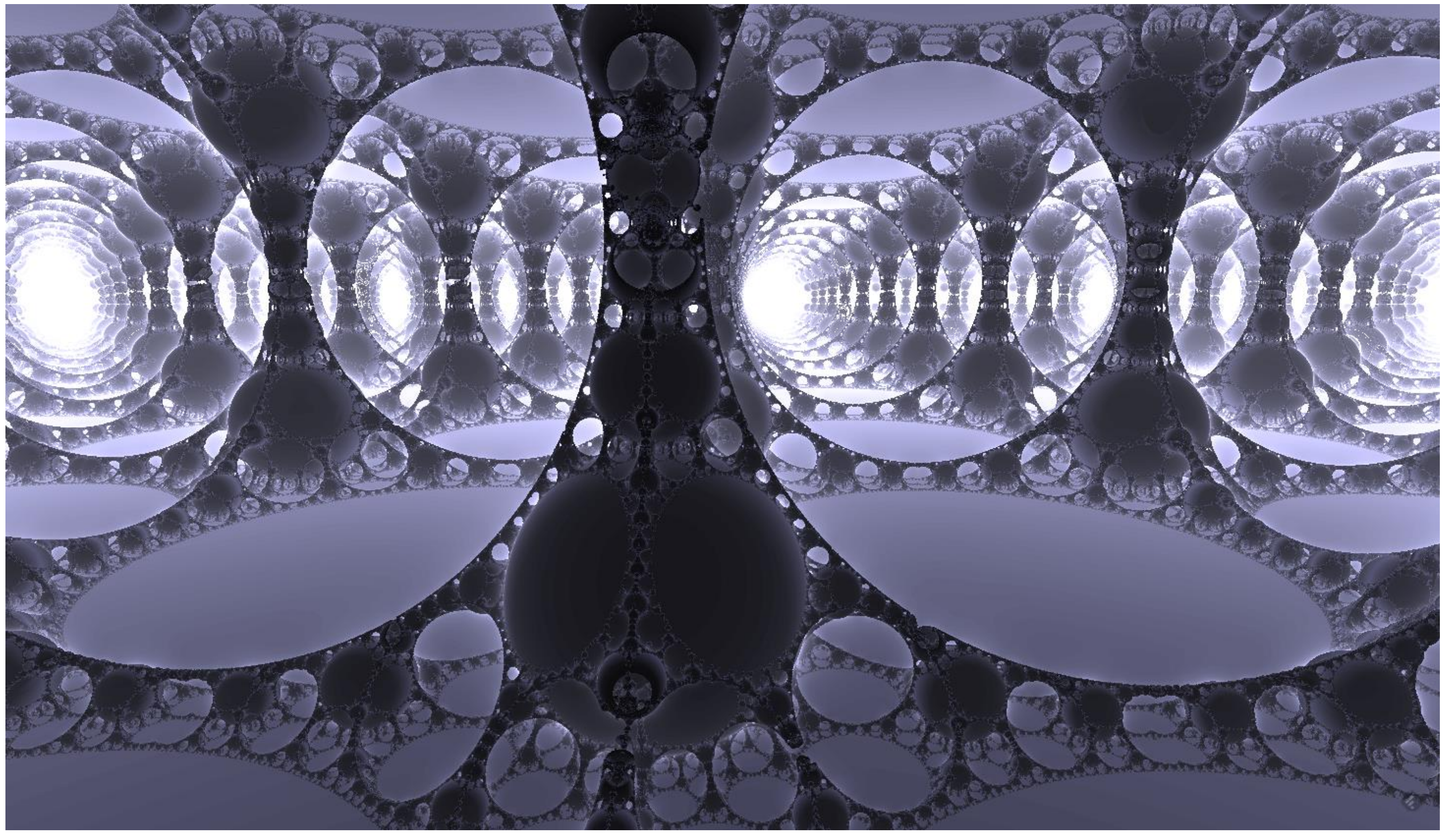
```
float tglad_formula(vec3 z0)
{
    z0 = mod(z0, 2.);

    float mr=0.25, mxr=1.0;
    vec4 scale=vec4(-3.12,-3.12,-3.12,3.12), p0=vec4(0.0,1.59,-1.0,0.0);
    vec4 z = vec4(z0,1.0);
    for (int n = 0; n < 3; n++) {
        z.xyz=clamp(z.xyz, -0.94, 0.94)*2.0-z.xyz;
        z*=scale/clamp(dot(z.xyz,z.xyz),mr,mxr)*1.;
        z+=p0;
    }
    float dS=(length(max(abs(z.xyz)-vec3(1.2,49.0,1.4),0.0))-0.06)/z.w;
    return dS;
}
```



http://glslsandbox.com/e#34784.1

Fractal

```
float pseudo_kleinian(vec3 p)
{
    const vec3 CSize = vec3(0.92436,0.90756,0.92436);
    const float Size = 1.0;
    const vec3 C = vec3(0.0,0.0,0.0);
    float DEfactor=1.;
    const vec3 Offset = vec3(0.0,0.0,0.0);
    vec3 ap=p+1.;
    for(int i=0;i<10 ;i++){
        ap=p;
        p=2.*clamp(p, -CSize, CSize)-p;
        float r2 = dot(p,p);
        float k = max(Size/r2,1.);
        p *= k;
        DEfactor *= k;
        p += C;
    }
    float r = abs(0.5*abs(p.z-Offset.z)/DEfactor);
    return r;
}
```



http://glslsandbox.com/e#34784.0

シェーダ圧縮

- シェーダコードサイズは exe サイズに直結
- 変数 / 関数名を 1 文字にする、改行スペースを省く、などの最適化が有効
  - Shader Minifier というツールがこれをやってくれる
  - http://www.ctrl-alt-test.fr/?page_id=7
- マクロにまとめるのも有効

Agenda

- 小さい exe を作る
- レンダリング
- **サウンド**
- まとめ

サウンド

- 波形データを生成し、waveOut 系関数に渡すことで音を鳴らす
  - waveOutOpen(), waveOutPrepareHeader(), waveOutWrite()
- 波形データをどう生成するかが問題
  - プロシージャル的なアプローチが必須となる

サウンド

- 基本は数式による生成
  - sin() 関数などを駆使して波形データを生成
- 大きく 2 通りのアプローチがある
  - 全て数式で生成
  - シンセサイザを用意

数式による生成

- 文字通り完全に数式だけで曲を生成
- シェーダ化できるためよく縮む
- 容量的に有利だが自由度は低い
- Shadertoy の例:
  - https://www.shadertoy.com/view/ldXXDj
  - https://www.shadertoy.com/view/ldfSW2
  - ピクセルシェーダで波形データを生成、テクスチャに書き込み、その内容を CPU 側に返している

シンセサイザ

- 楽譜データだけ事前に exe に持たせておく
- 音のパーツを実行時に数式で生成
- 楽譜と音パーツを組み合わせ、合成して曲を生成
- 自由度は高いが容量は大きくなる
- Elevated がこの方式でソースも公開されている
  - http://www.pouet.net/prod.php?which=52938

シンセサイザ

- 偉大なる先人たちの成果物
  - 4k lang: http://4klang.untergrund.net/
  - V2 Synthesizer System: http://www.pouet.net/prod.php?which=15073
- VSTi プラグインとして機能し、DAW で作曲が可能
- 結果は .obj ファイルとして出力
  - 波形データを生成する関数が含まれている
- お手軽だが容量はやや大きくなりがち

Agenda

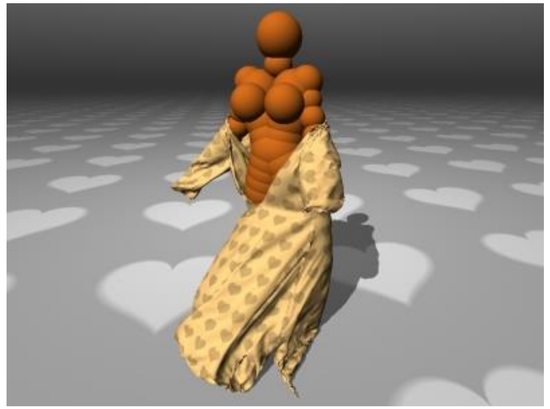
- 小さい exe を作る
- レンダリング
- サウンド
- **まとめ**

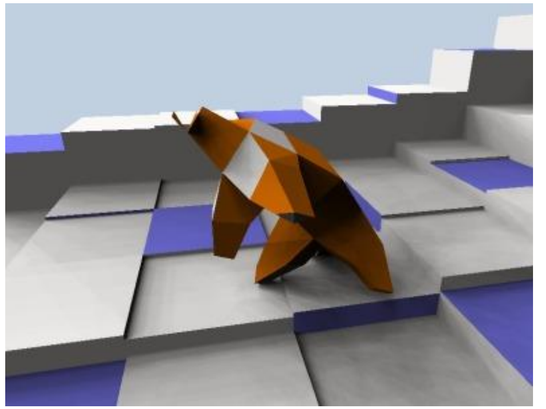
まとめ

- 4k intro を作るには
- C でプログラムを書いて
- 画面全体を覆う 4 角形を一枚出して
- ピクセルシェーダで distance function を raymarch して絵を出す
- サウンドは臨機応変にがんばる
- 最後に crinkler でリンクして圧縮

補足

- とはいえ、レイマーチだけが 4k intro ではない
- 敢えてポリゴン (非レイマーチ) で頑張っている作品
- 剛体物理、クロスシム、GA、レイトレなどで勝負する特殊な例も
- アイデア勝負も十分通用する





で？4KBにして何かの役にたつの？

利益

- ローカルのメディアの容量は爆発的に増えたが、コンテンツをダウンロードすることを考えると容量は少ない方がよい。
  - 特にモバイルデバイスでは通信量は課題
- 高解像度のクオリティが要求されるコンテンツではテクスチャデータなどデータとして持つのは現実的でない。
- 多くのバリエーションパターン、またはインタラクティブにデータの変化を必要とする場合

4KBの技術

=

プロシージャル技術として有用

まとめ

- デモパーティは楽しいです。
- デモシーナーは様々なところで活躍しています。
- デモシーンの技術は様々なところで役に立ちます。
- Tokyo Demo Fest をよろしくお願いします。

Tokyo Demo Fest 2017
開催決定！
2017年2月18-19日(予定)

みなさまのデモ作品お待ちしております！

Q/A

ありがとうございました

Reference

- 4k intro サンプル
  - https://github.com/i-saint/CEDEC2016_4kintro
- kioku さんによる 4k intro 解説
  - http://kioku.sys-k.net/tips/
- Packer
  - Executable Packerの構造と解釈 http://tokyodemofest.jp/2011/seminar/2011/301-siep.pdf
  - Crinkler 詳説 http://xoofx.com/blog/2010/12/28/crinkler-secrets-4k-intro-executable/
- distance function & raymarching
  - http://iquilezles.org/www/articles/distfunctions/distfunctions.htm
  - http://iquilezles.org/www/
  - http://blog.hvidtfeldts.net/index.php/2011/06/distance-estimated-3d-fractals-part-i/
  - http://i-saint.hatenablog.com/entry/2015/08/24/225254