当たって砕けろッ！
プロシージャルオーディオ制作

自己紹介

サウンド部
サウンドデザイナー
廣瀬　裕貴

自己紹介

テクノロジー推進部
サウンドリサーチャー
シディーク・サジャード

自己紹介

サウンド部
サウンドプログラマ
谷山　輝

はじめに

- 鋭意制作、組み込み中の技術の発表です。

本日の内容

1. プロシージャルオーディオ開発記
   - 開発経緯
   - 開発工程
2. Granular Synthesisを用いた
   破壊音生成
3. まとめ

1.プロシージャルオーディオ開発記

従来のアプローチ

- 厳密に合わせるのでなく「変化」を作る。
  - いくつかの波形でシーケンスを組む。
  - ランダムなピッチ、ディレイなどを入れる。
  - 波形を増やして、ランダムに鳴らす。

1つ目の理由

- 厳密な予測はしてこなかった
- すべてのバリエーションに対応することは不可能

2つ目の理由

- メモリ容量
  - コンソール機のメモリ配分
    - PS3　256MB
    - Xbox360　512MB
  - サウンド最大使用メモリ
    - LRFFXIII 約16MB
    - FFXIV 約10MB

ハードウェアの進化

- メモリ容量
  - コンソール機のメモリ配分
    - PS3　256MB　⇒　PS4　8GB
    - Xbox360　512MB　⇒　XboxOne　8GB
  - サウンド最大使用メモリ
    - LRFFXIII　約16MB　⇒　16MB * 約15 = 240MB(?)
    - FFXIV 約10MB　⇒　10MB *約15 = 150MB(?)

※概算です。本当は丸々使えるわけではありません。

新しいアプローチへ

- 処理能力の向上！
- 使用メモリの増加！！

正確で豊かな表現
を目指そう！！

というわけで・・・

プロシージャルオーディオをつくろう！

はじまり

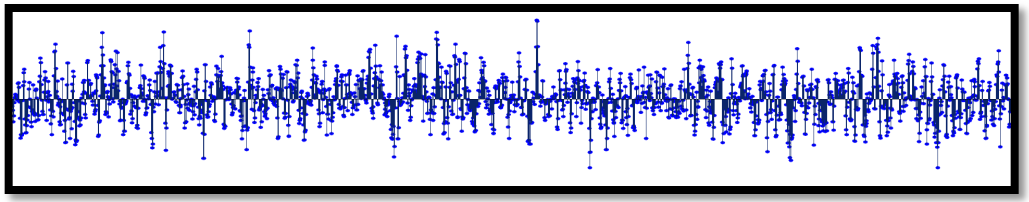
えぇから、やれやぁ～！

！？

！？

プロシージャルオーディオとは

- プロシージャル化
  - テクスチャ、地形、街並み、木、etc
  - オーディオ？



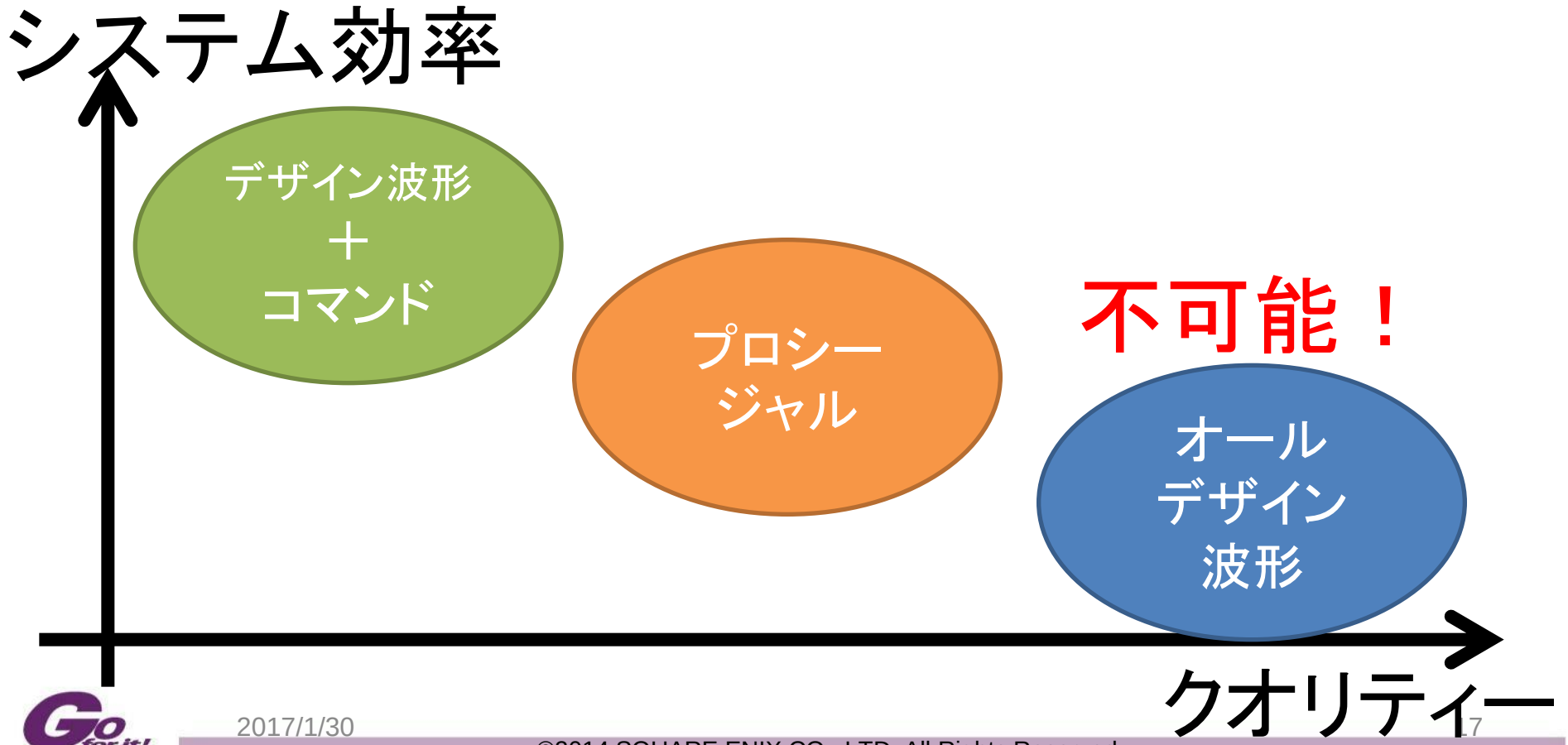
- 音声波形をアルゴリズムで作れるようにすること

収録・DAW ⇒ アルゴリズム

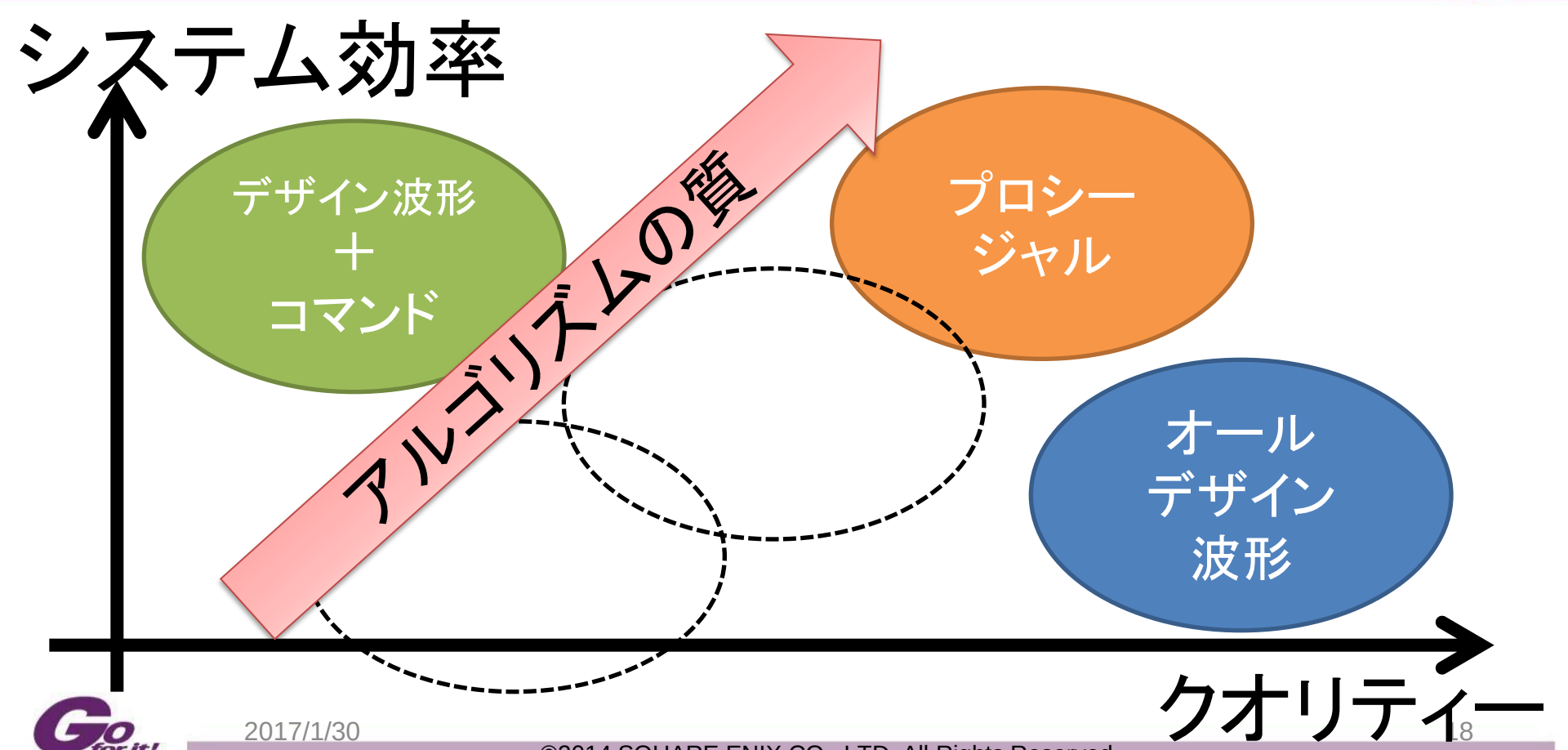
メリット

- パラメータの変更で多くの波形バリエーションを得られる
- ゲーム（物理など）との接続部分が実装できていれば、勝手に出来上がる
- システム（メモリ・CPU）効率・クオリティはアルゴリズム次第

プロシージャルオーディオ



プロシージャルオーディオ

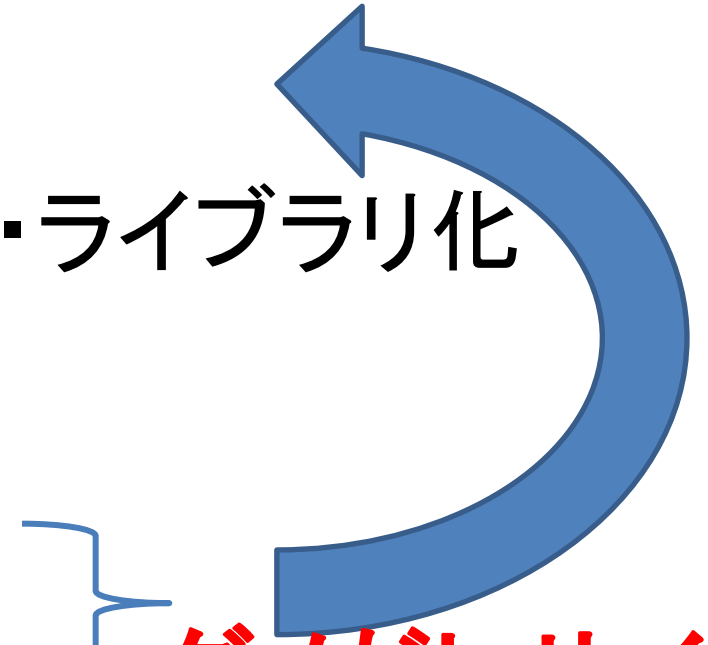


デメリット

- 設計が難しい
  - 「波形生成アルゴリズム」を作れる人があまりいない
  - 効率とクオリティのバランス
- 開発に時間が掛かる
  - トライ＆エラー型
    - 聞いてみないと評価できない
    - 実際にゲームと接続しないと評価ができない

開発工程

1. 目標とする音をデザイナーが選定・制作
2. 音響分析
3. アルゴリズム設計・ライブラリ化
4. ツールの制作
5. クオリティチェック
6. ゲームに実装



ダメだしサイクル

開発工程

1. 目標とする音をデザイナーが選定・制作
   - 自由に。
   - 表現したいものをできるだけ折り込む。

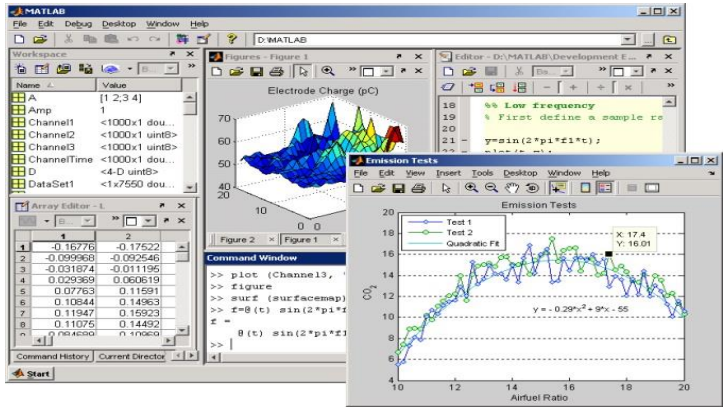
開発工程

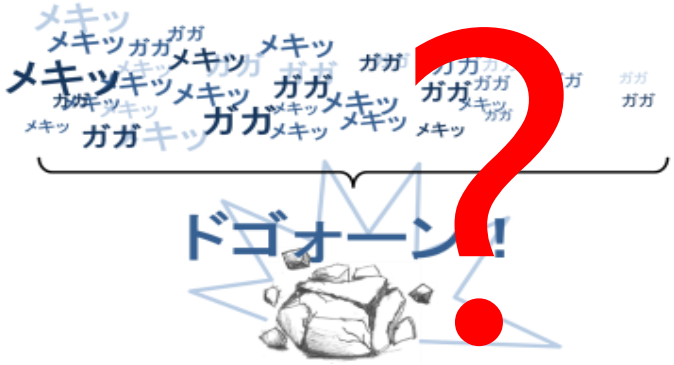
2.　音響分析

– 音の原因、構成要素を明らかにする

「この音は、～が原因で発生している。」

Matlab

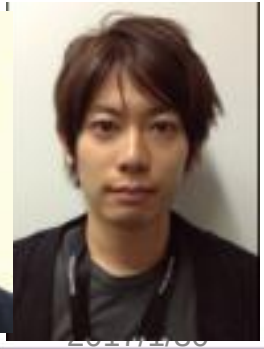




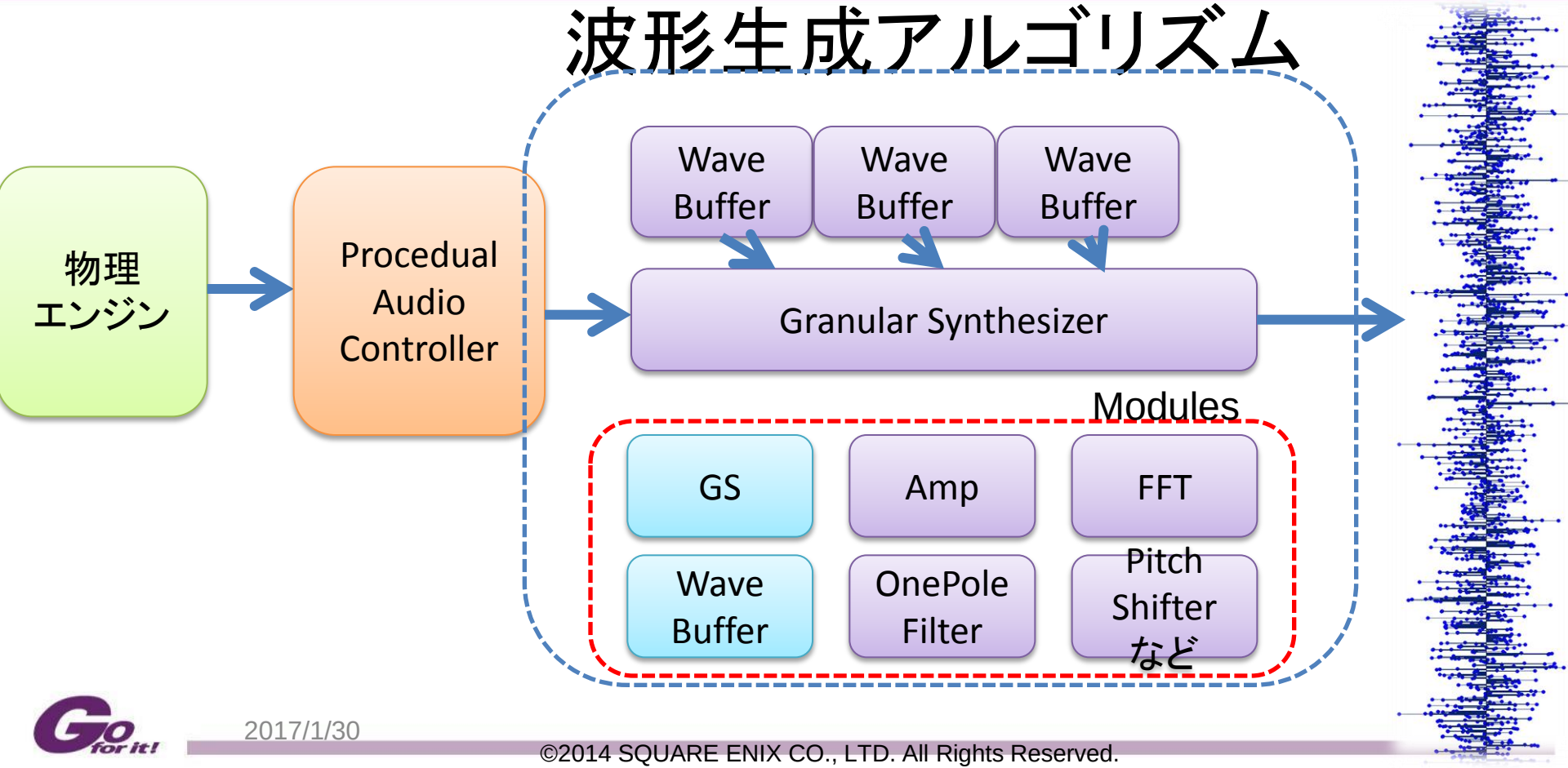
開発工程

3.　アルゴリズム設計・ライブラリ化

- 構成要素をモジュール化
  - 基本的にありもので構成
  - 無いモジュールは制作する
  - 波形デザインだけでなく物理との接続も考える

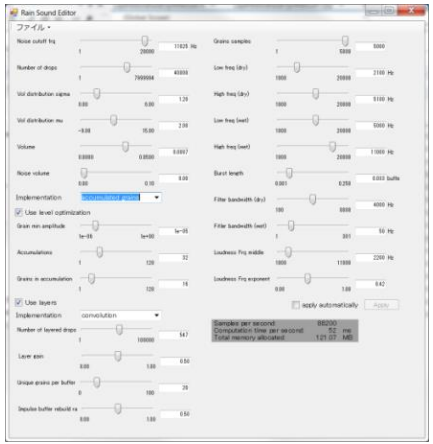


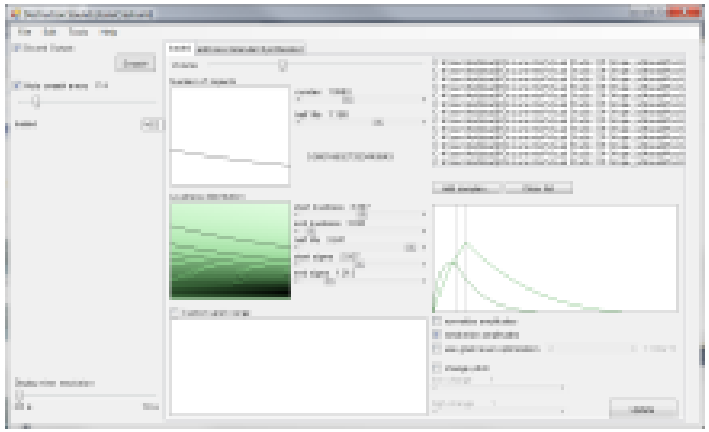
アルゴリズム設計例



4. ツールの制作

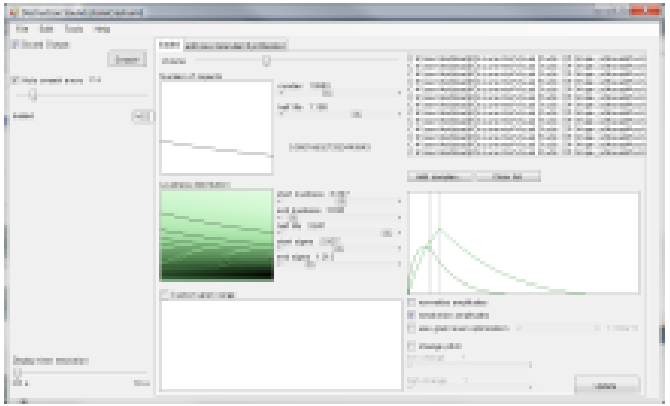
- 早く聞かせること
- トライ＆エラーすることを念頭に
  - あらゆるパラメータを編集できるように





開発工程

5. クオリティチェック
   - 出力音の検査
     - なんかダサい、ノイズ？
   - パフォーマンス検査
     - 処理負荷、メモリ使用量
   - パラメータ検査
     - プリセット化
     - 不必要パラメータ削除



ツールの変化：初期

Rain Sound Editor

ファイル

Noise cutoff frq

1 11025 Hz

Number of drops

1 40000

Vol distribution sigma

0.00 1.20

Vol distribution mu

-9.00 15.00 2.00

Volume

0.0000

Noise volume

0.00

Implementation accumulated gra

1 5000

1000 20000 2100 Hz

High freq (dry)

1000 20000 5100 Hz

Low freq (wet)

1000 20000 5000 Hz

High freq (wet)

1000 20000 11000 Hz

Burst length

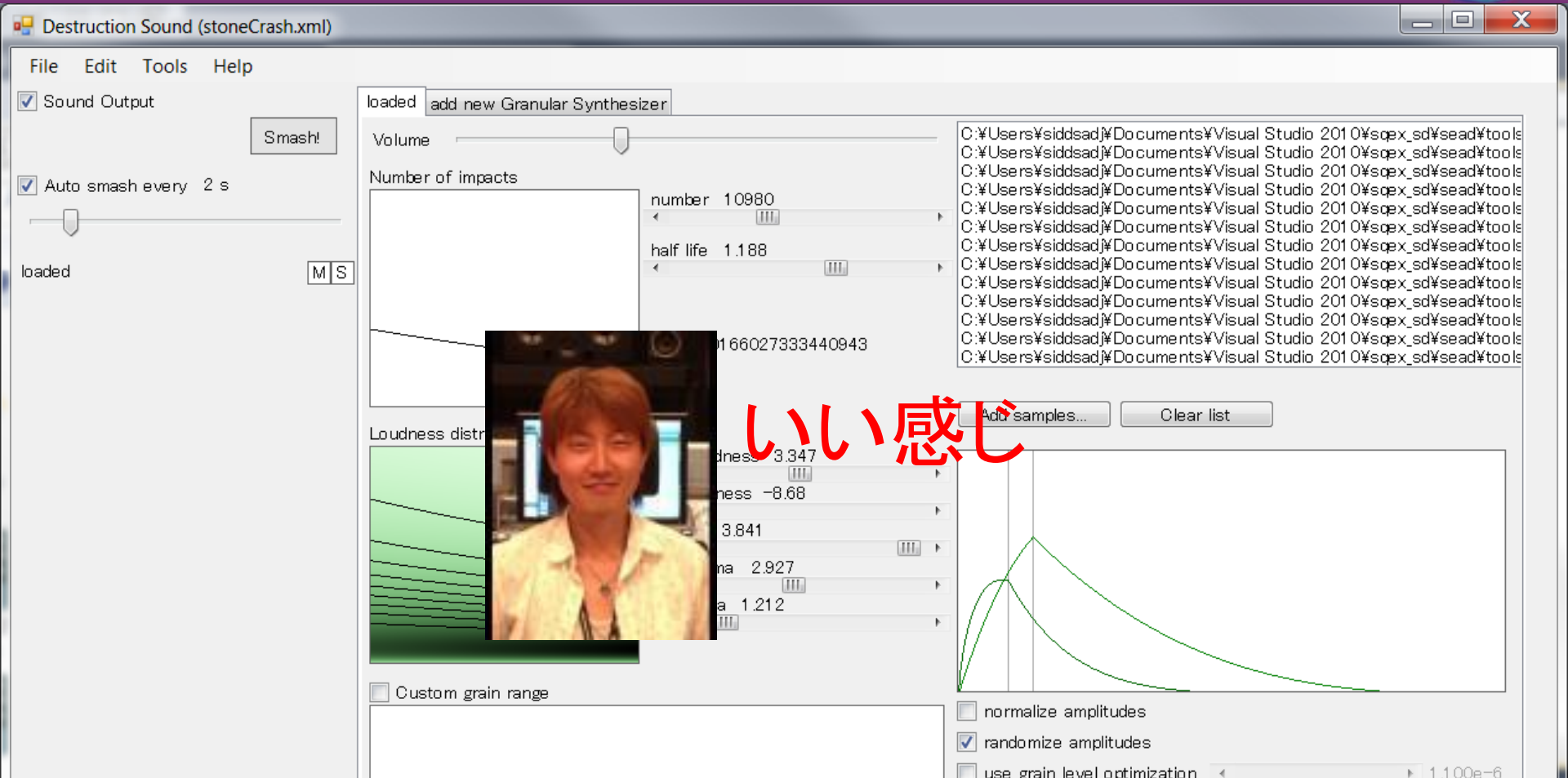
0.001 0.250 0.033 buffe

Filter bandwidth (dry)

わからん！！

わかるよ～

ツールの変化：後期


Destruction Sound (stoneCrash.xml)
File
Edit
Tools
Help
Sound Output
Smash!
Auto smash every 2 s
loaded
M
S
loaded
add new Granular Synthesizer
Volume
Number of impacts
number 10980
half life 1.188
C:¥Users¥siddsadj¥Documents¥Visual Studio 2010¥sqex_sd¥sead¥tools
Add samples...
Clear list
Loudness distr
3.347
−8.68
3.841
2.927
1.212
Custom grain range
normalize amplitudes
randomize amplitudes
いい感じ

6. ゲームに実装

- 実際の物理エンジンと接続
  - ここで初めて気づく問題も多い。
    - エンジンの情報が足りない！
    - 特殊な表現がある！

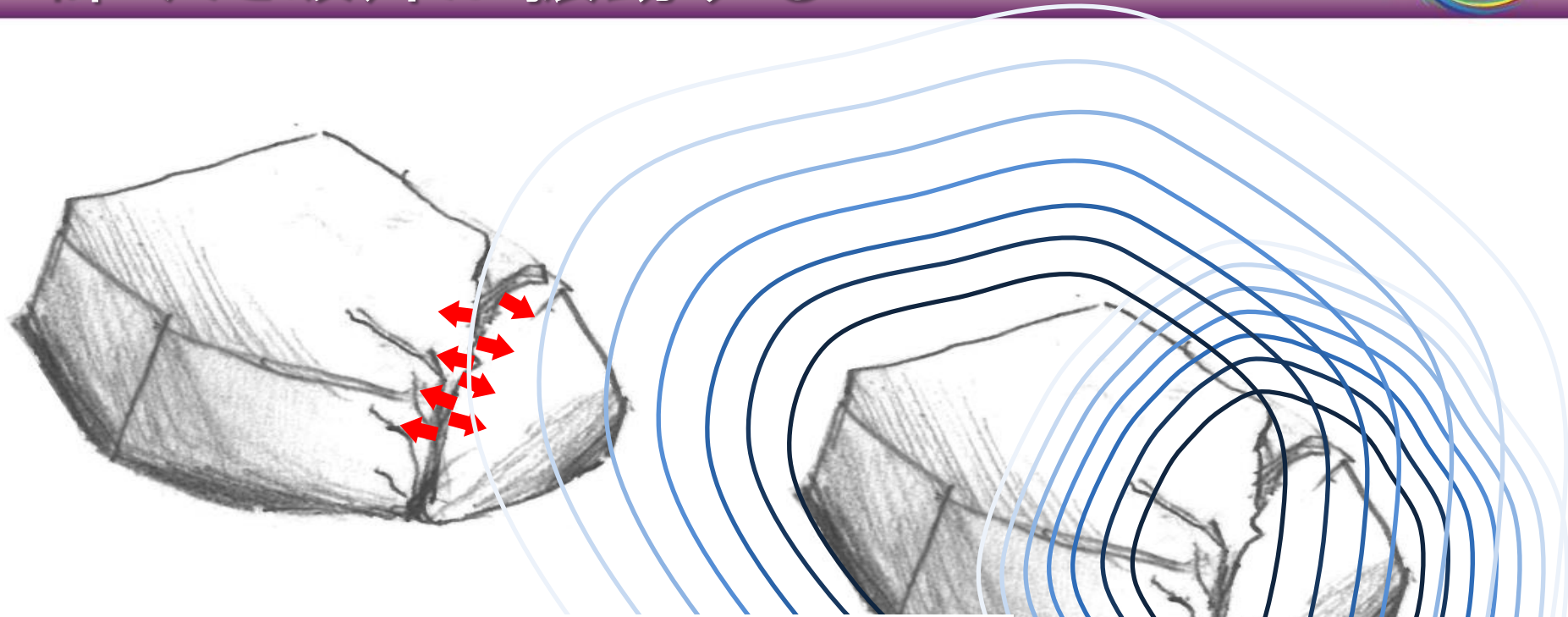
開発期間目安

- 初期開発（1⇒6）に大体2か月くらい
  1. 目標とする音をデザイナーが選定・制作　2，3日
  2. 音響分析　1～2週間
  3. アルゴリズム設計・ライブラリ化　2～3週間
  4. ツールの制作　1～2週間
  5. クオリティチェック　2，3日
  6. ゲームに実装　1週間
- だめだしサイクル（2⇒5、6）に1～2週間。

今回は、

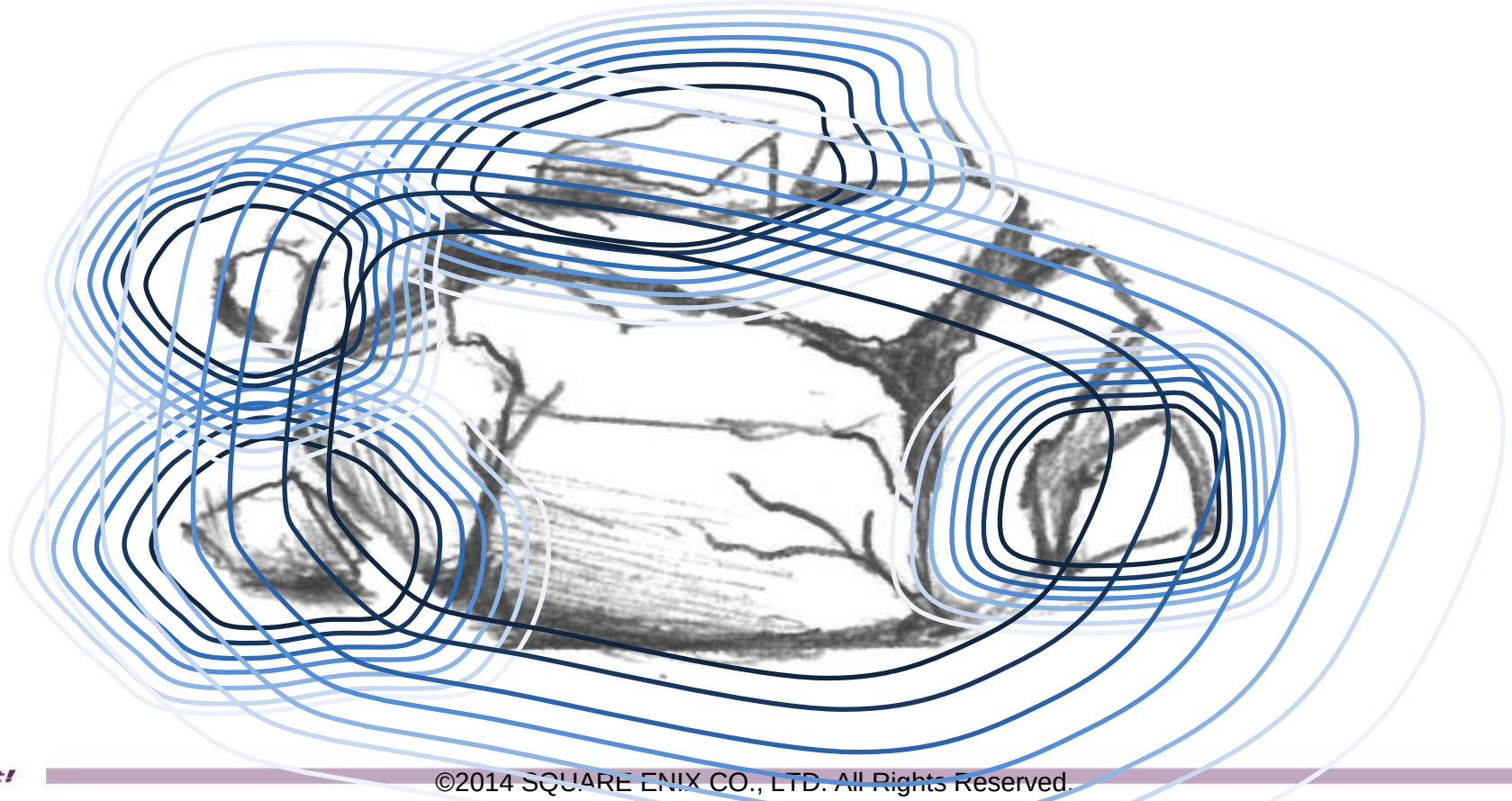
2. Granular Synthesisを用いた破壊音生成

砕けた破片が振動する

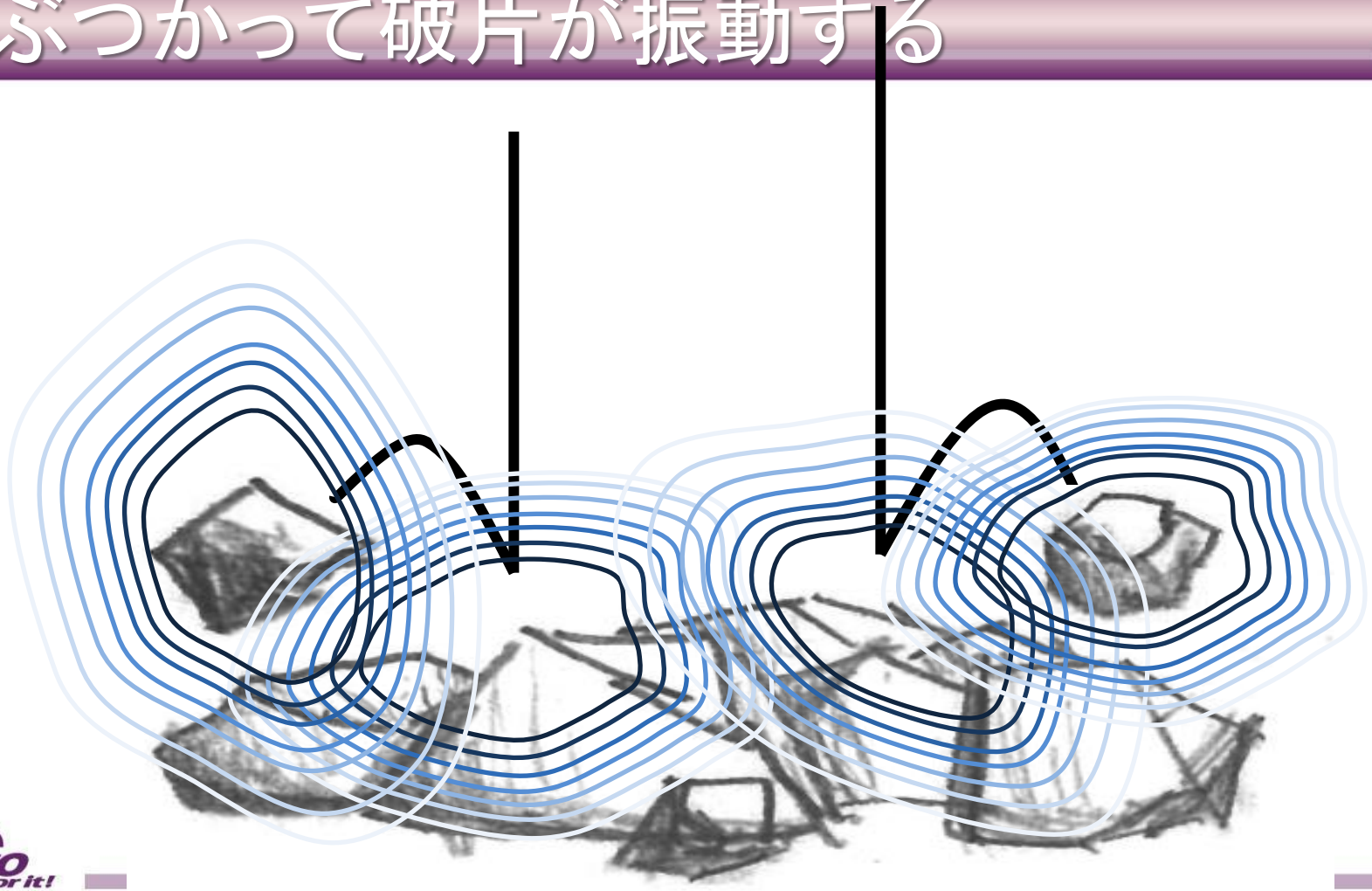


物が砕ける物理について参考
Zheng, James: Rigid-body Fracture Sound with Precomputed Soundbanks. Siggraph 2010.

砕けた破片が振動する

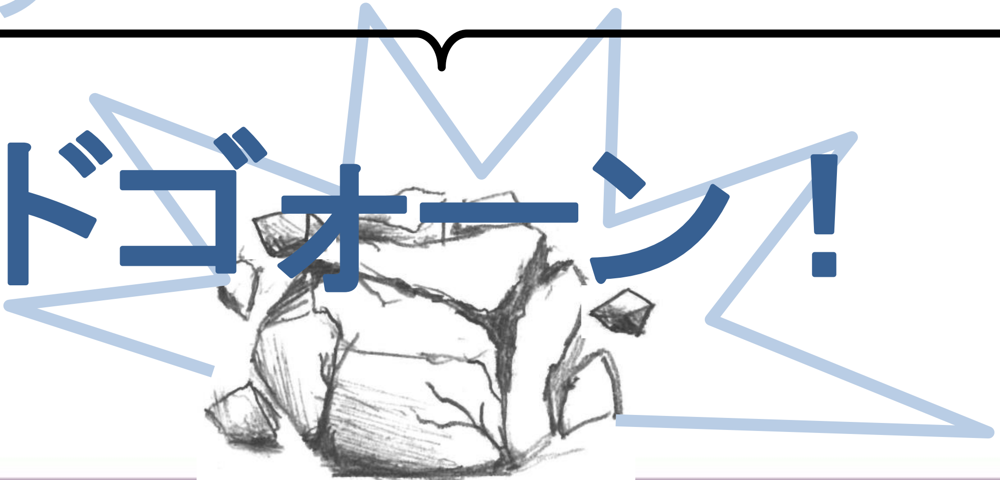


ぶつかって破片が振動する

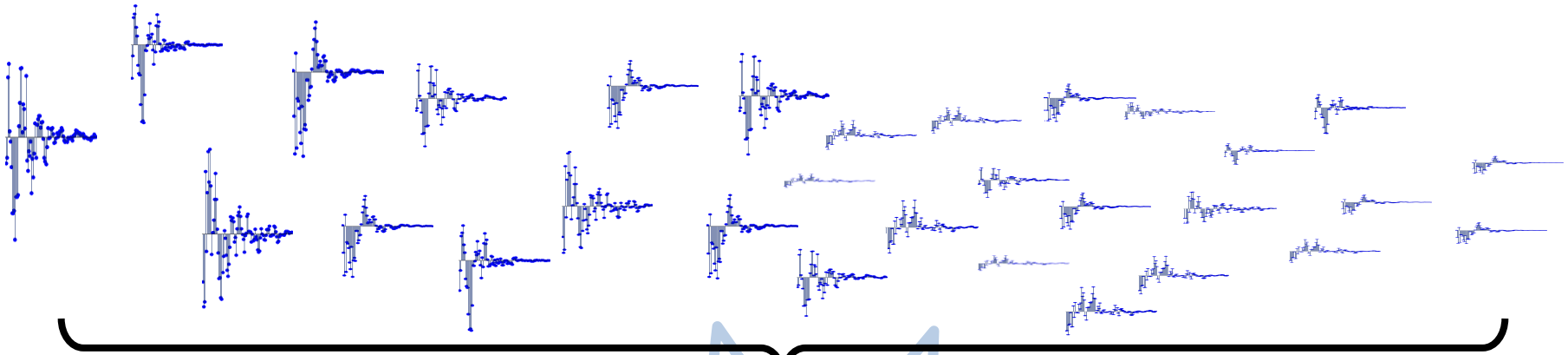


数多くの音の結果

メキッ
メキッガガガガ
メキッ
メキッ メキッ ガガ ガガ ガガ ガガ
メキッ メキッ ガガ ガガ ガガ メキッ ガガ
メキッ メキッ メキッ ガガ ガガ メキッ ガガ ガガ ガガ ガガ
ガガ メキッ ガガ メキッ ガガ ガガ ガガ ガガ ガガ
メキッ メキッ ガガ メキッ メキッ メキッ
ガガ メキッ ガガ メキッ メキッ



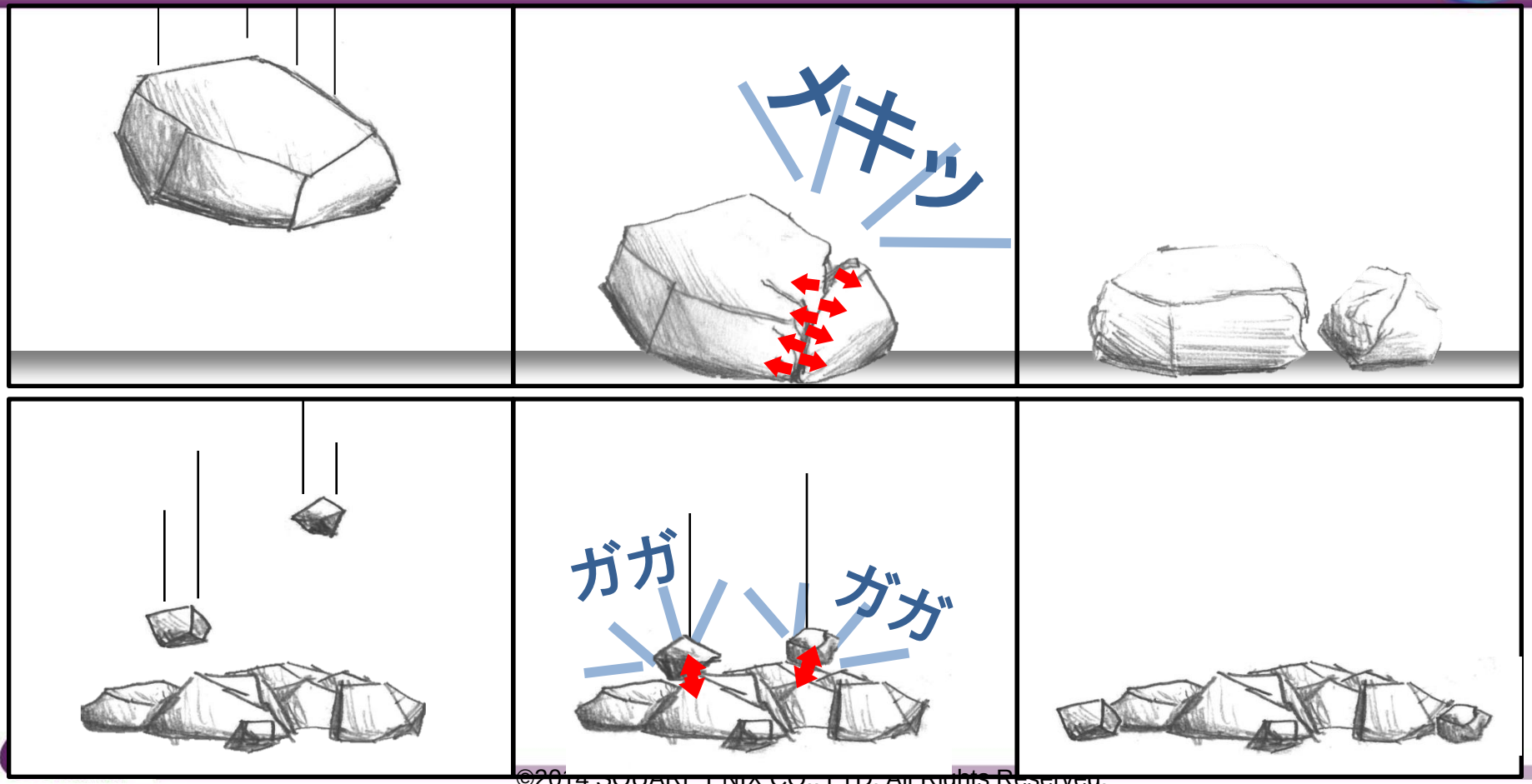
数多くの音の結果



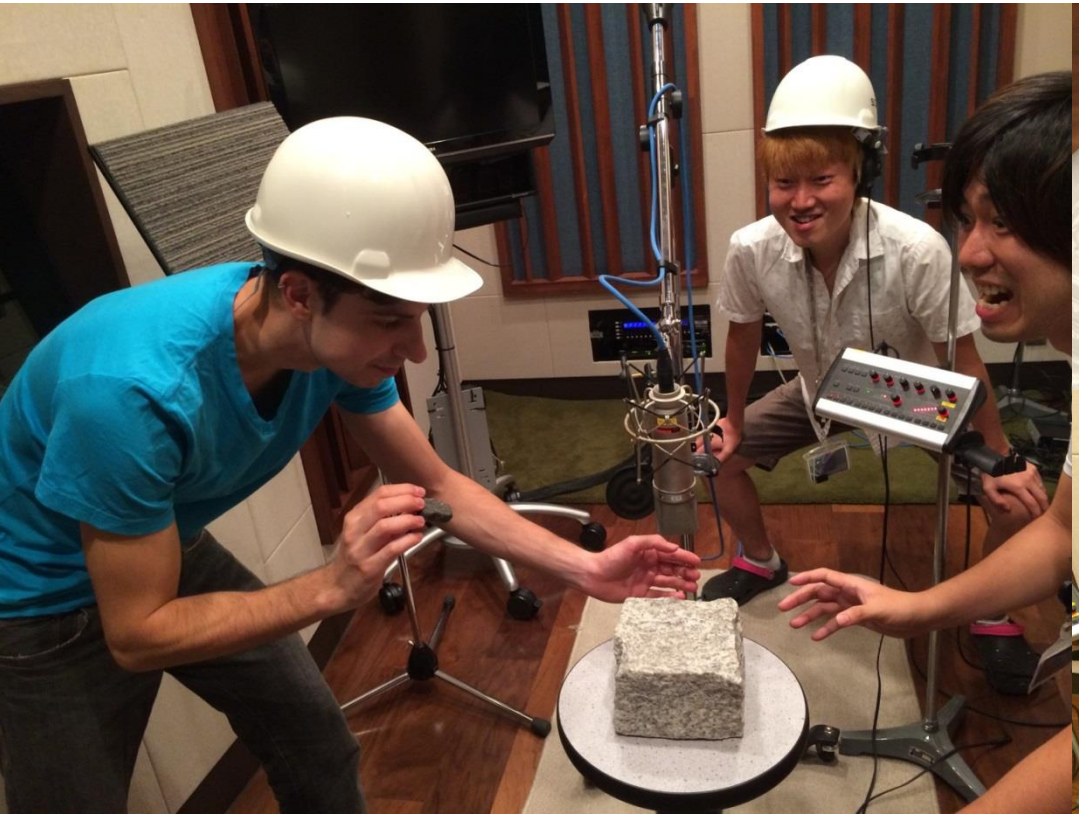
ミキシング

ドゴオーン！

砕ける音とぶつかる音

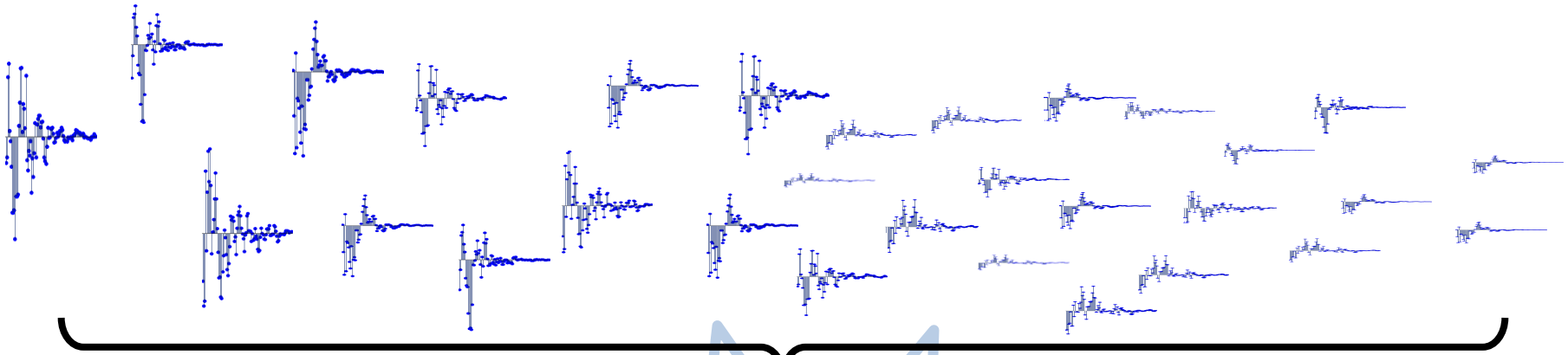


音の収録



SquareEnix社内収録スタジオにて。

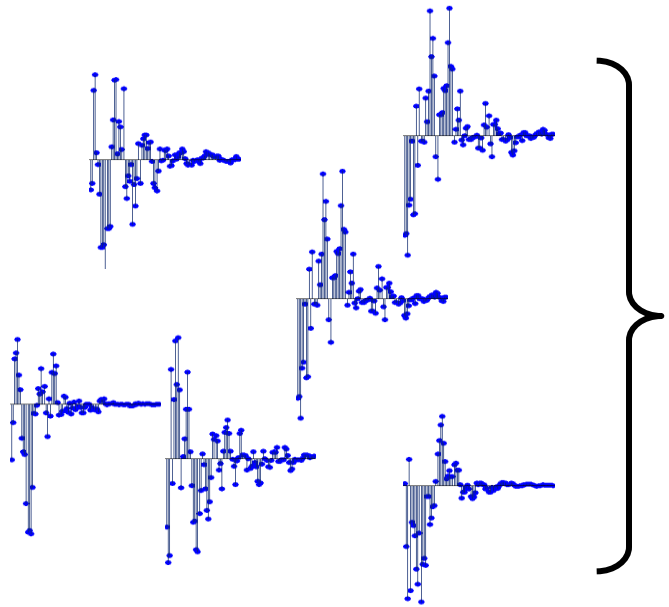
数多くの音の結果

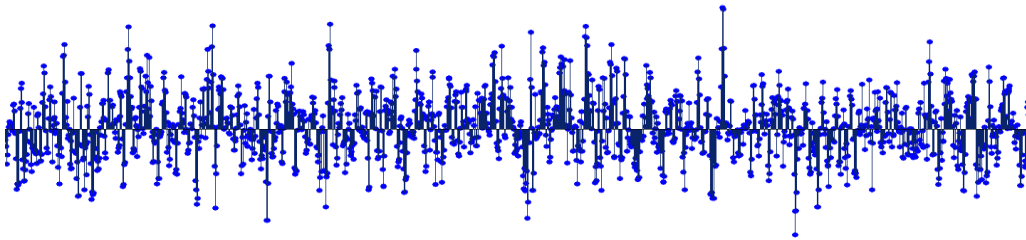


ミキシング

ドゴオーン！

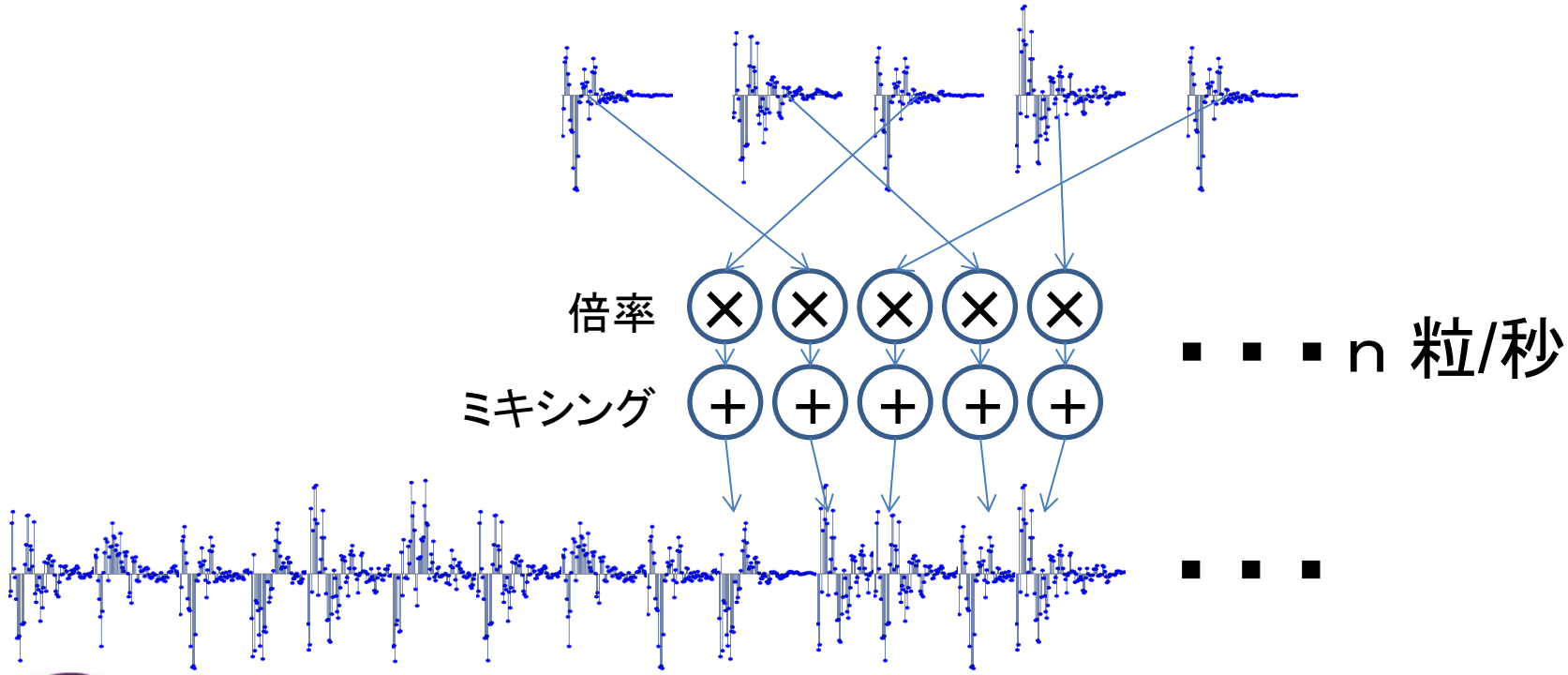
Granular Synthesis（GS）の基礎



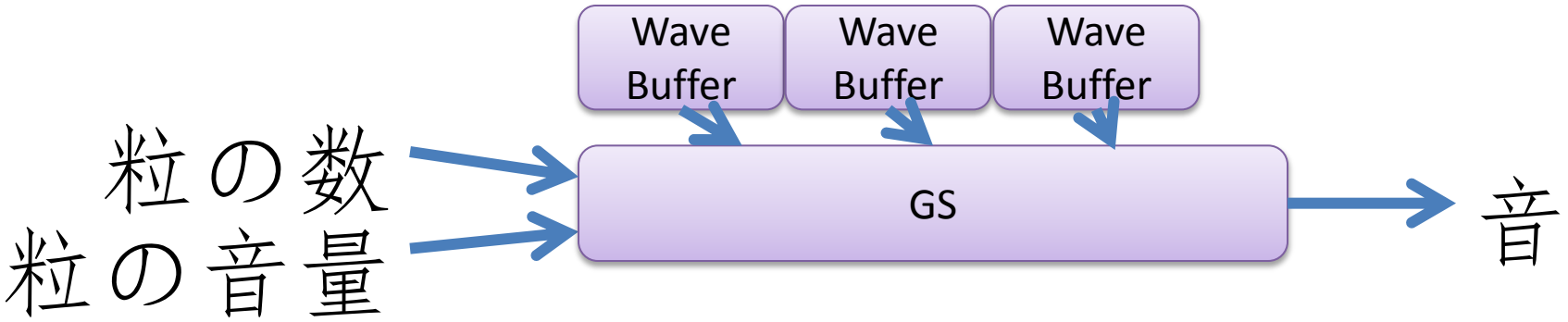


単一の粒を　　組み合わせて　　複雑な音にする。

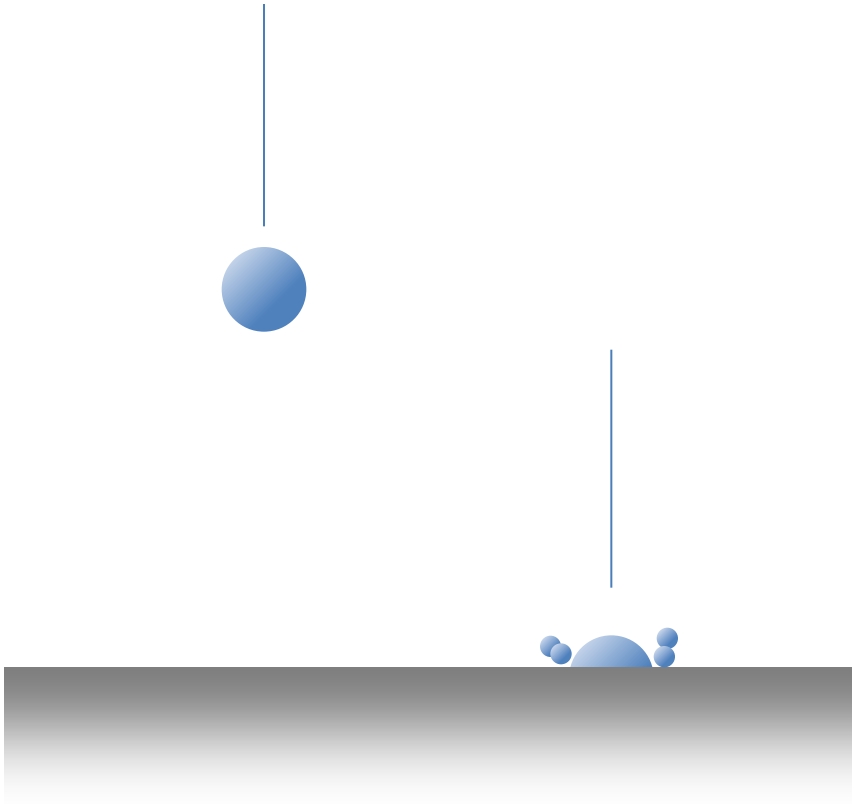
Granular Synthesis（GS）の実装

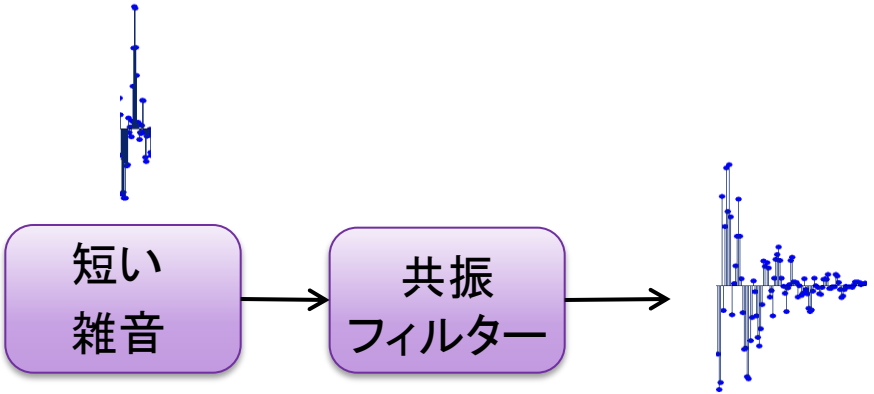


GSの制御



雨の音





滴の音の作り方

粒の数（雨の音）

1 粒/秒 

10 粒/秒 

100 粒/秒 

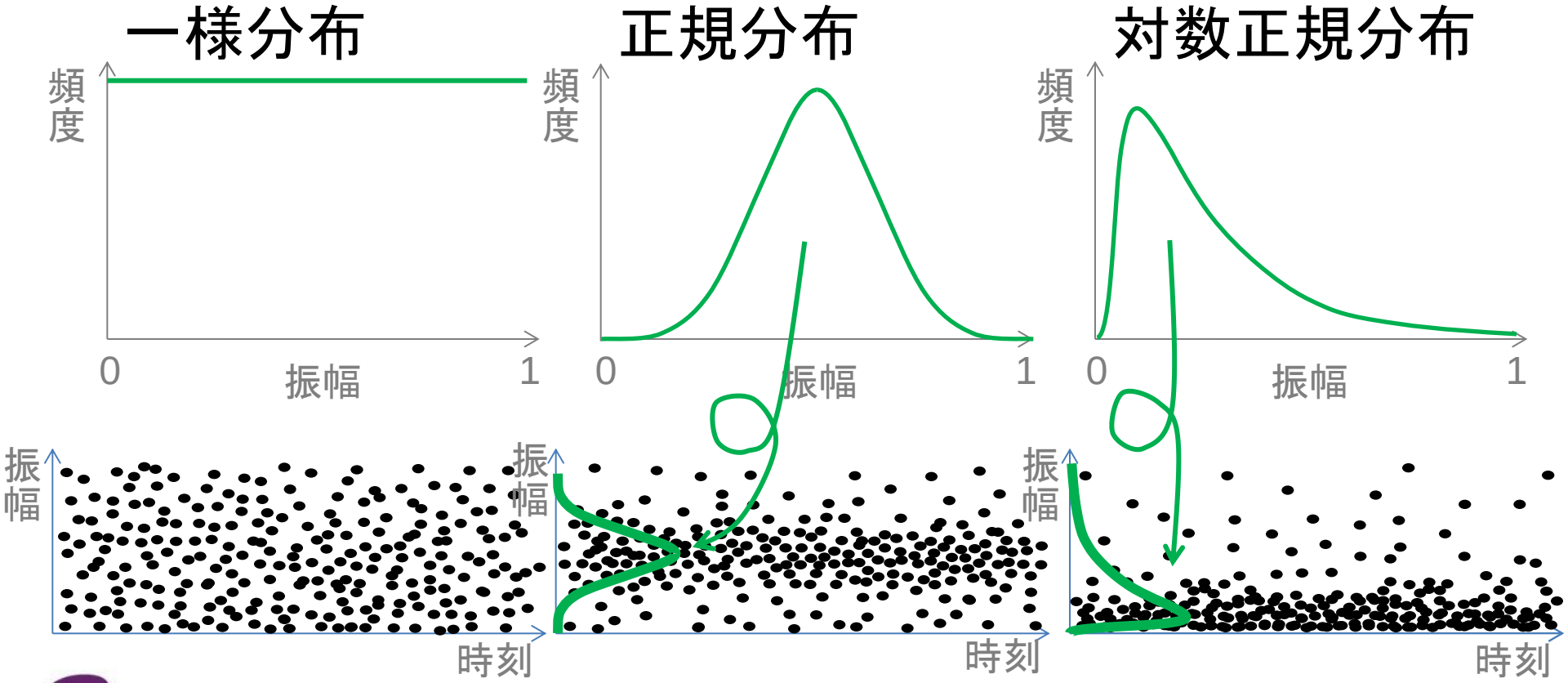
1000 粒/秒 

10000 粒/秒 

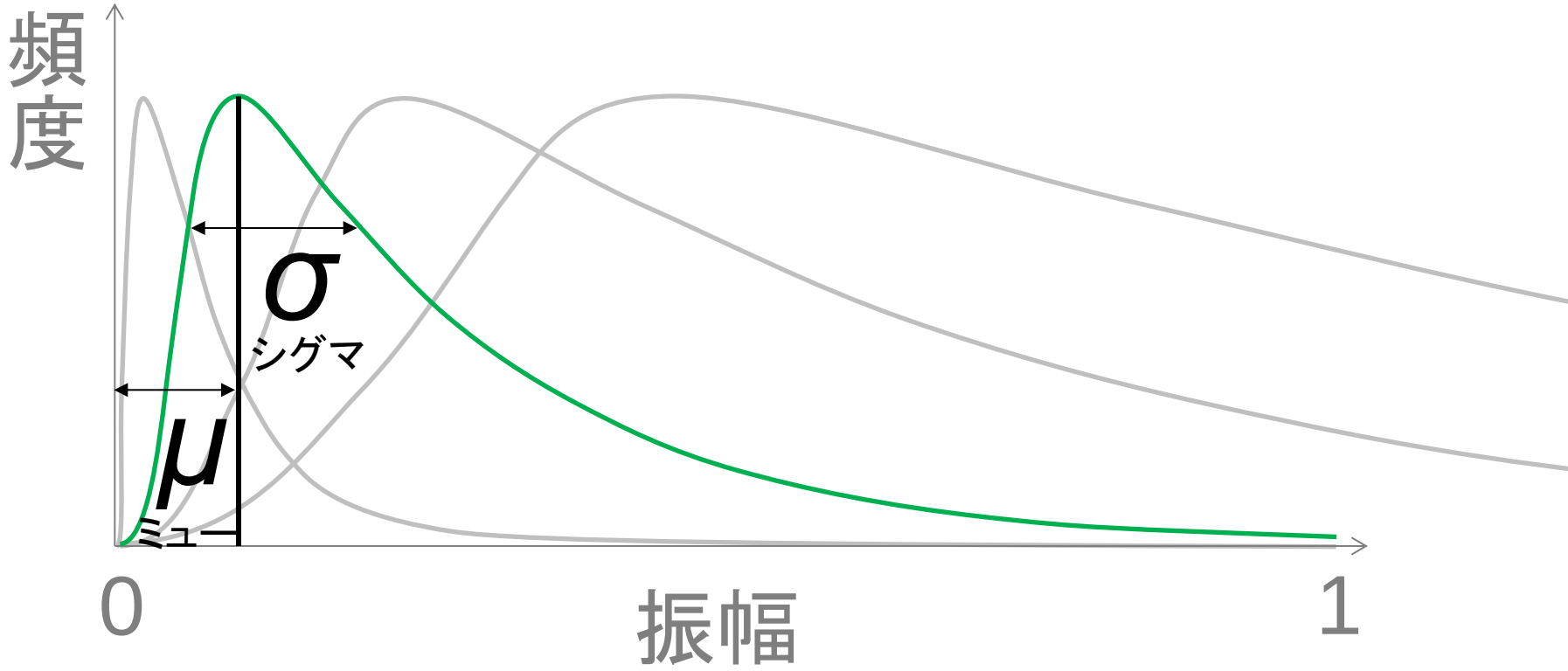
100000 粒/秒 

1000000 粒/秒 

いろいろな確率分布（確率密度関数）

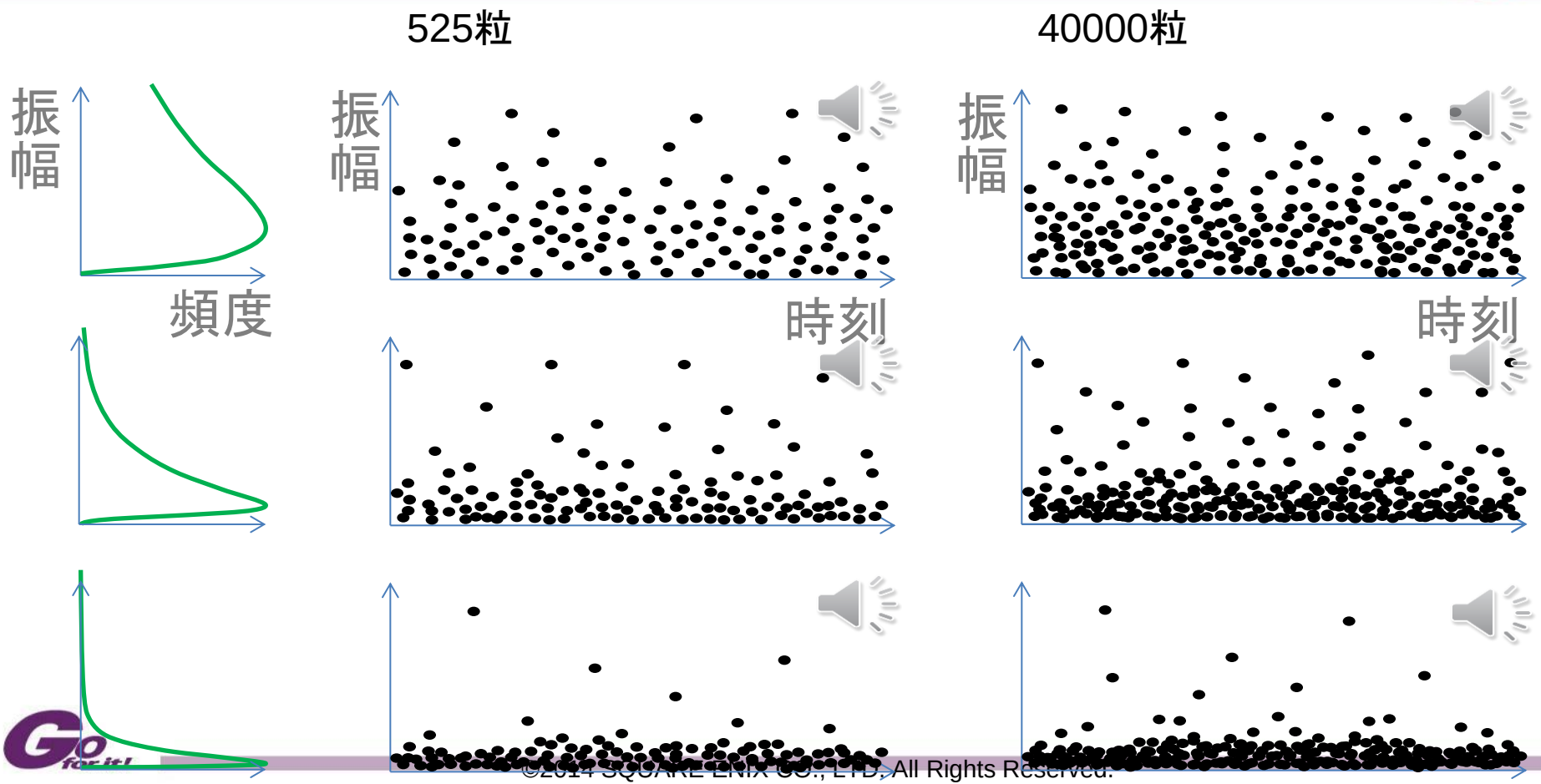


対数正規分布

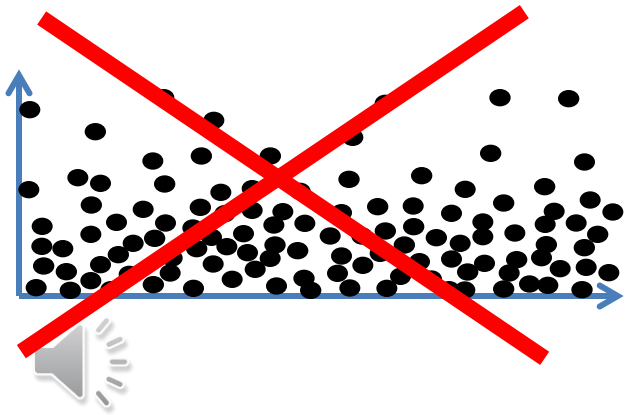


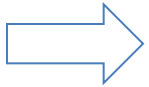
※This graph does not show the actual values of σ and μ.

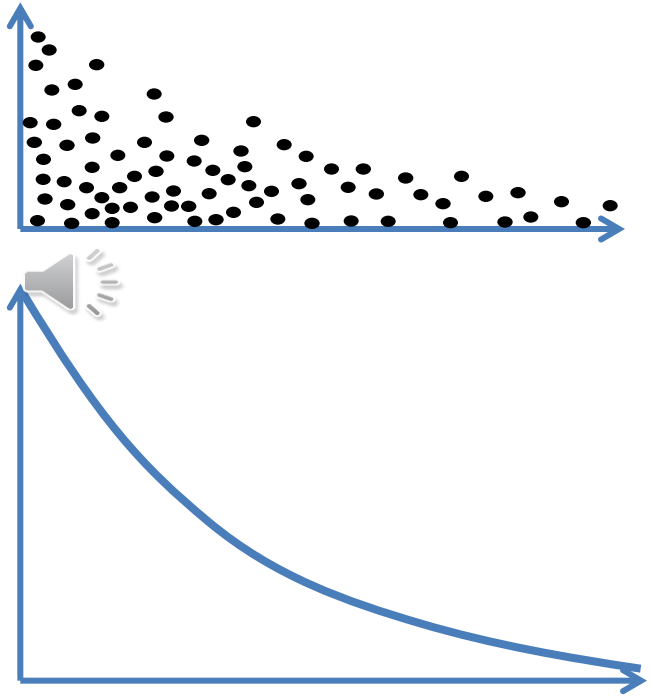
色々なパラメーターを使った例



砕ける岩の音

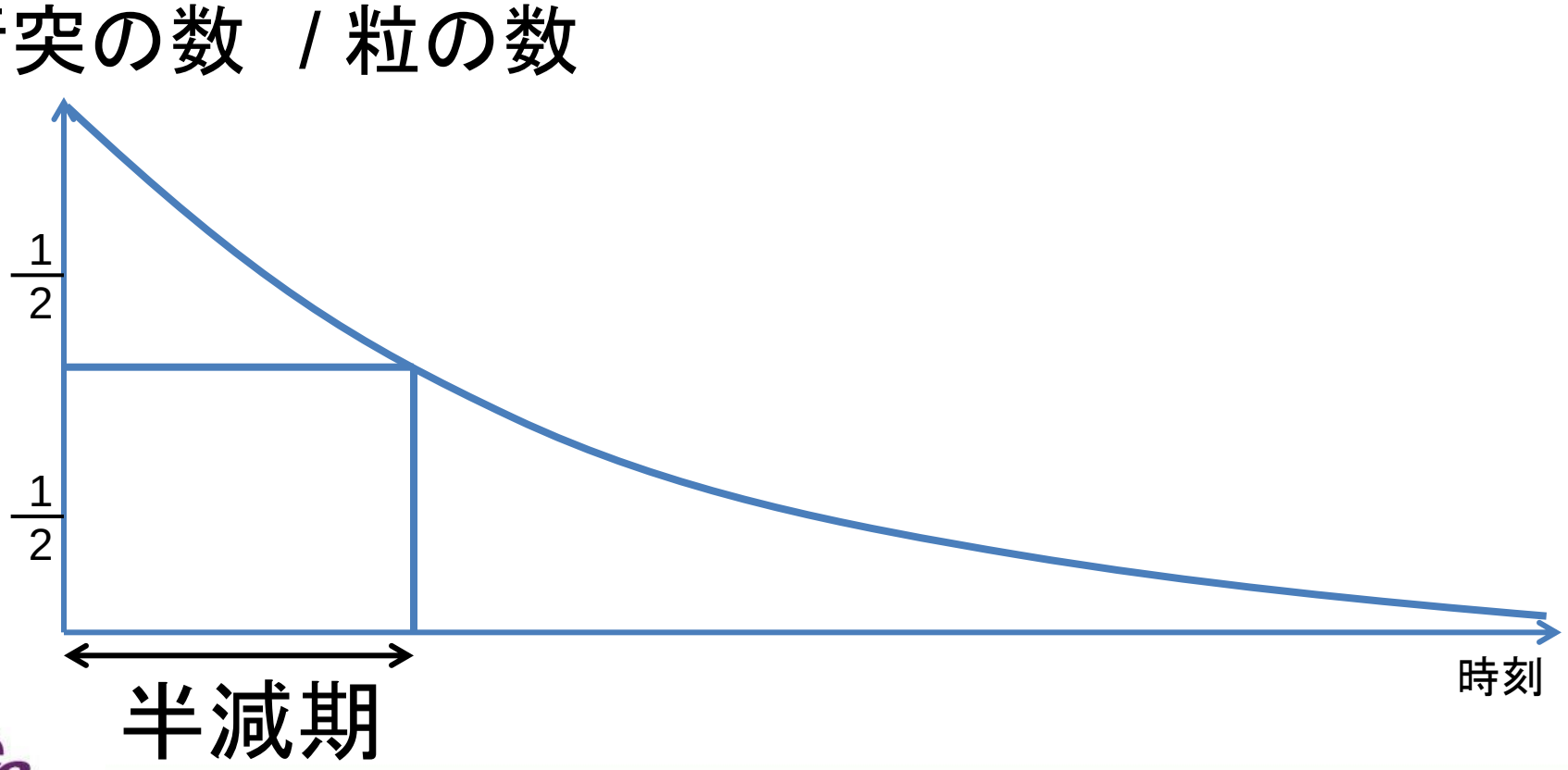




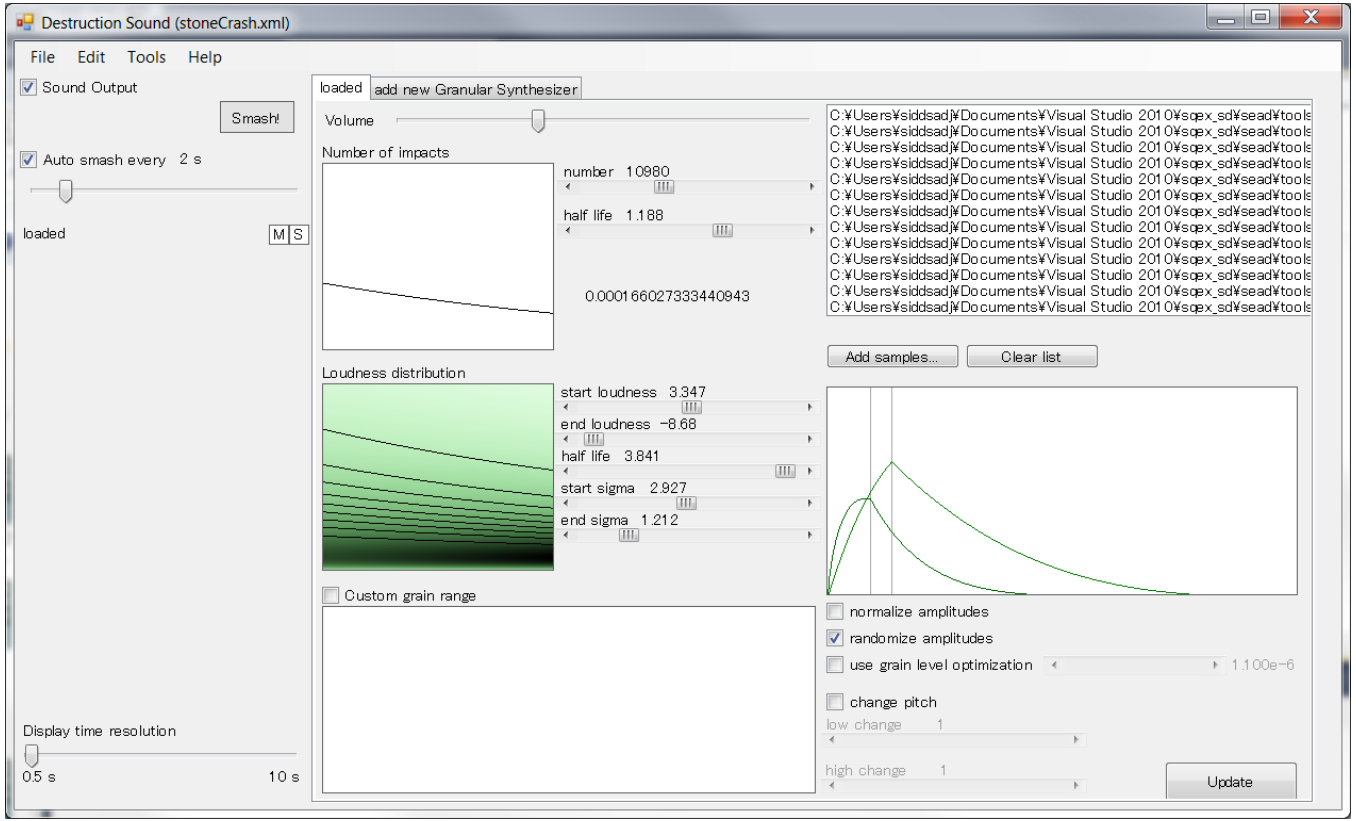


時間で変化する
パラメーターが必要

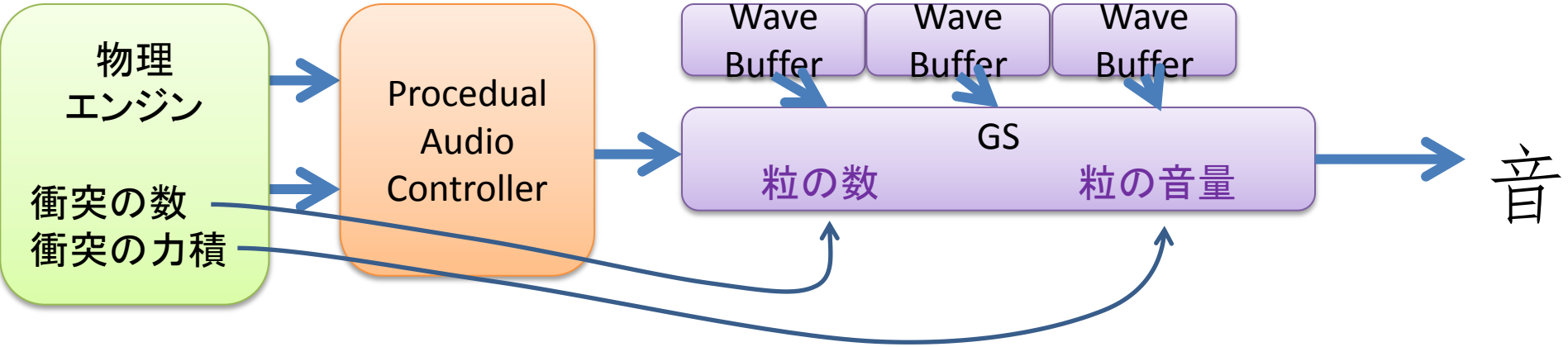
衝突数の時間変化曲線



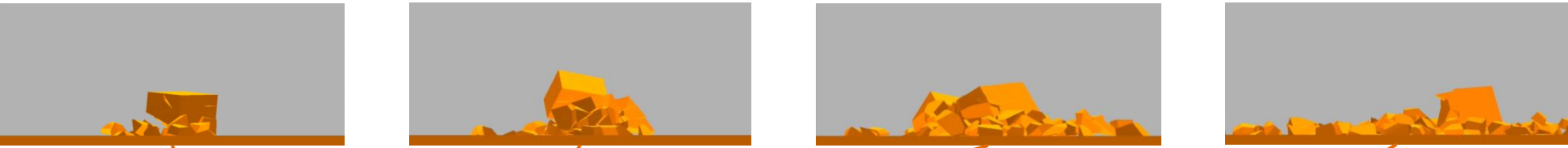
岩の破壊音ツール

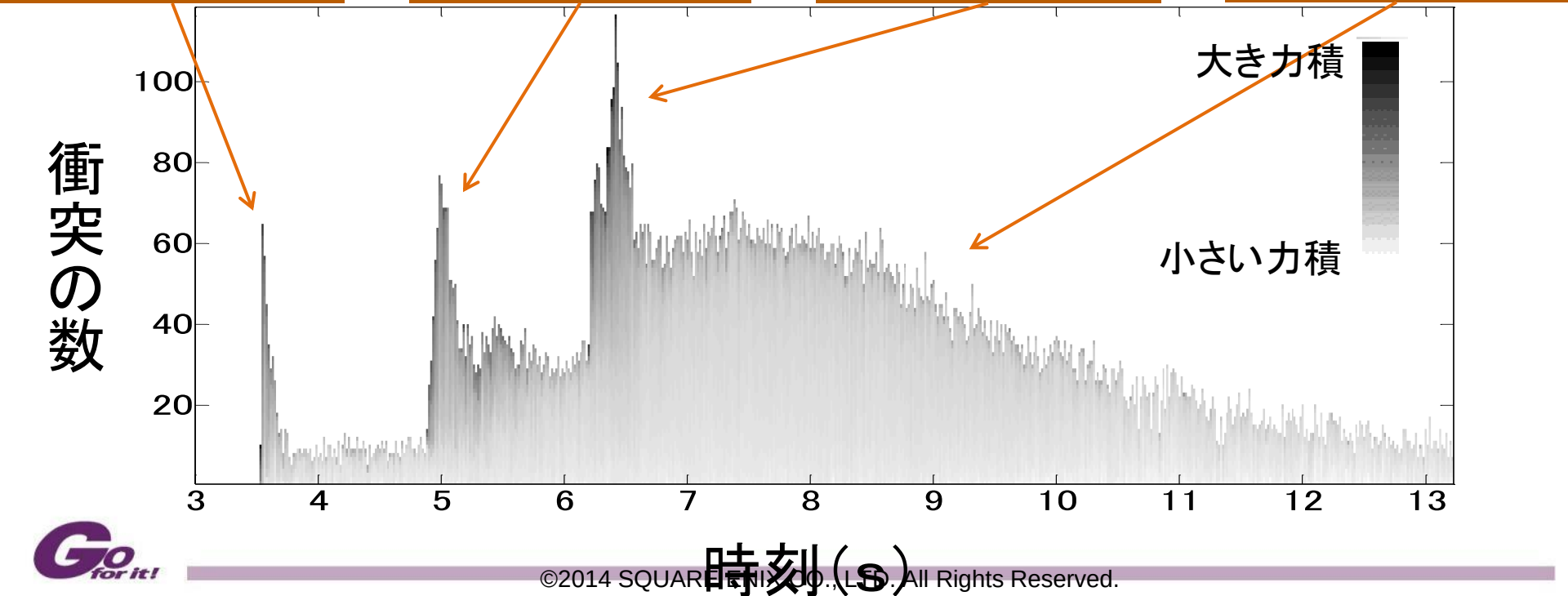


GSの制御

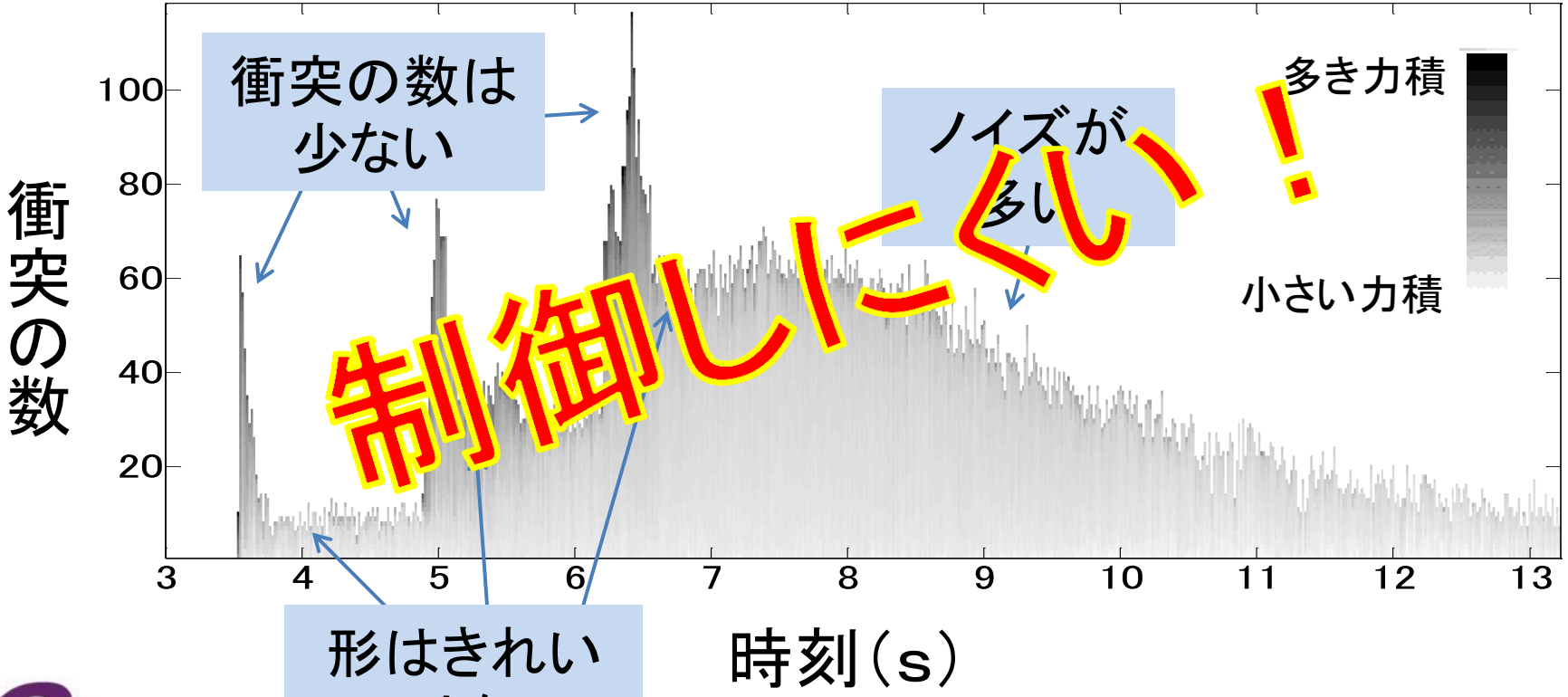


物理エンジンから得たデータ

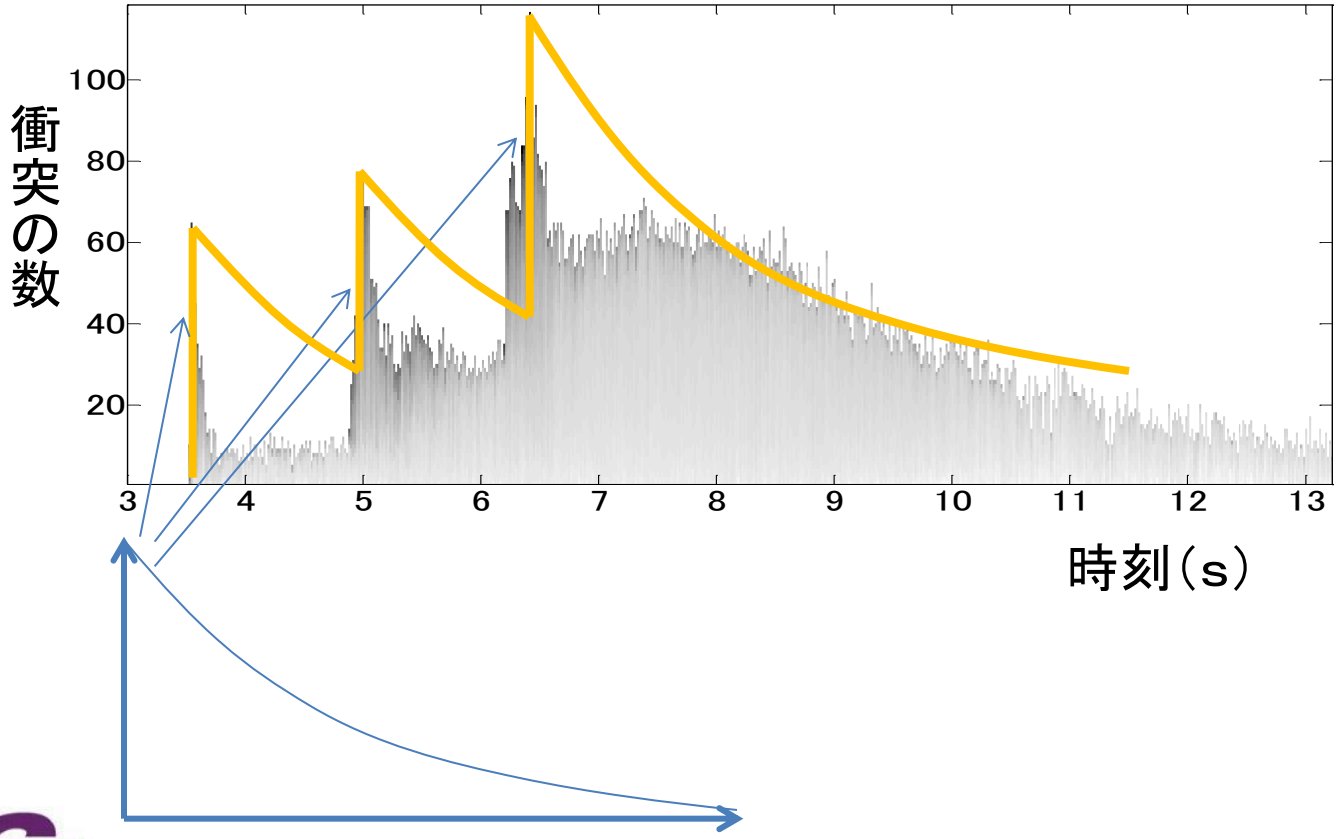




物理エンジンから得たデータの問題

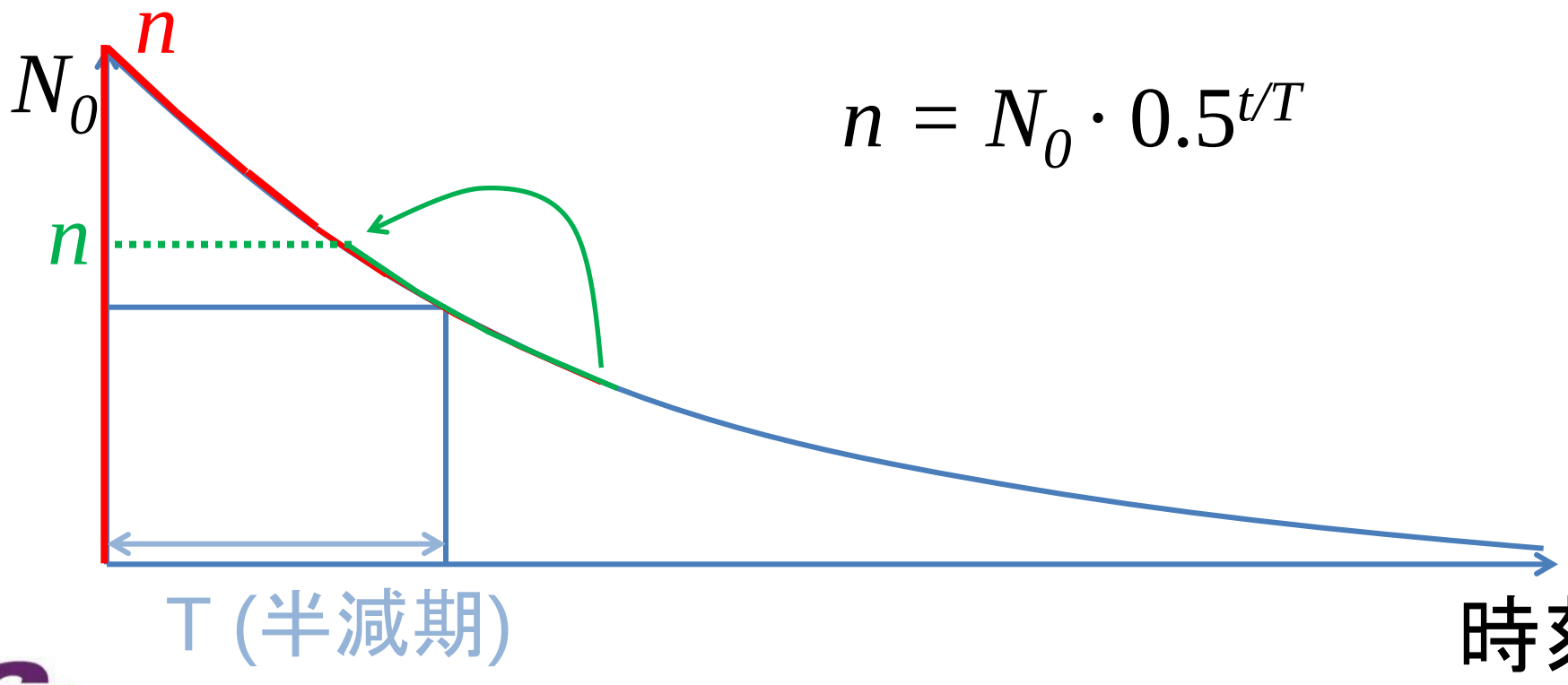


衝突数の時間変化曲線使用



衝突数の時間変化曲線減衰

衝突の数　/ 粒の数



$$n = N_0 \cdot 0.5^{t/T}$$

時間変化曲線を使う良い点・悪い点

良いところ

- 制御しやすい
- 分かりやすい
- ツールの設定を直接使える
- 頑健

悪いところ

- 半減期の曲線は抽象
- 時間変化はいつも同じ

使用メモリと処理負荷

未最適化

記憶	約 0.5 MB　(全粒)
処理負荷	最低 4 ms – 最高 60 ms / 1秒

(44100サンプル)

(Intel® Core™ i7-2600 CPU @ 3.40 GHz)

3. まとめ

まとめ

- 岩の破壊音のプロシージャルオーディオ
  - 波形合成アルゴリズム
    「Granular Synthesis」
  - 物理エンジンとの接続
    直接　⇒　×
    「対数正規分布で粒の音量をコントロール」
    「半減期を持つ時間変化曲線で衝突数をコントロール」

今後の課題

- さまざまな音の種類に対応
- 空間表現
- 各種物理エンジンへの対応

展望

なんでも作れる
　プロシージャルオーディオSDKへ！

9月3日(水) 16:30～17:30

俺らはこうした！FINAL FANTASY XIVのBGサウンド構築
～次世代開発への橋渡し～