# R43 9/28（金）16:40～18:00

# MELを使って作業を短縮しよう
## ～デザイナーによる効率化～

CEDEC2007
株式会社スクウェア・エニックス
井和田　洋一

# 概要



1. ゲーム「Crisis Core」における効率化の実例
2. 効率化のポイントと仕組み
3. 代表的なスクリプトの紹介
4. 質疑応答

# 講演者紹介

CEDEC 2007
CESA DEVELOPERS CONFERENCE

井和田　洋一

- ビジュアルワークス　VFXセクション所属

  「Crisis Core ～Final Fantasy VII～」では、デザイナーとして
  ムービーパートのエフェクトを制作。
  現在は、VFXのチームリーダーとして次期プロジェクトの
  エフェクトを制作中。

- これまでに参加した代表的な作品
  - スクウェア・エニックス
    「Final Fantasy IX & X」
  - その他
    - 映画 「アップルシード」、「ピンポン」
    - ゲーム 「ナムコ リッジレーサーR4」
    - CM 「ナショナル ななめドラム洗濯機」、「サンスター GUM」
    - アトラクション 「お台場ジョイポリス 全ワイルドシリーズ」など

# デザイナーの現状



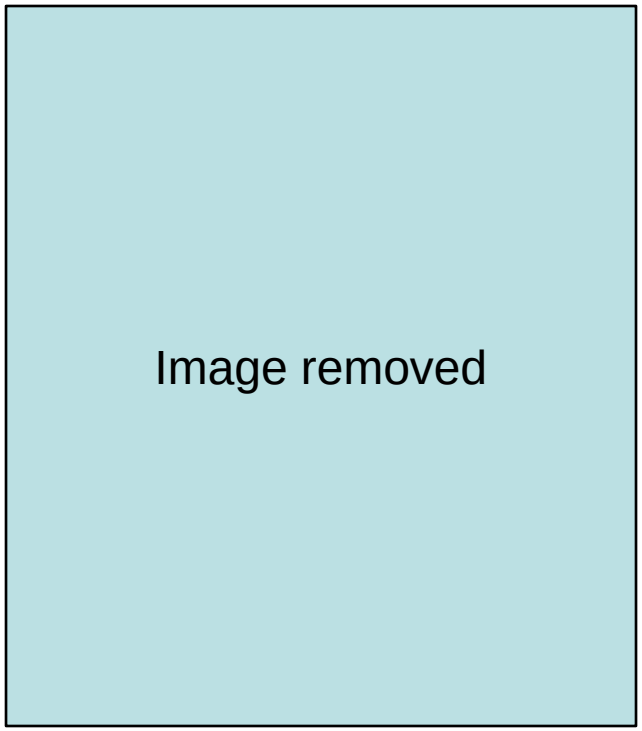
- 厳しいスケジュール
- スタッフ不足
- 要求される高いクオリティー
- 作業量の増大

# ワークフローを効率化して
# 生産性を向上させる

# 1. Crisis Core ～Final Fantasy VII～ における効率化の実例



発売日　:2007. 9.13
ジャンル:アクションRPG
プラットフォーム:PSP



# 宣伝映像

# 「Crisis Core」VFXワークフローの工程



① シーン構築

モデル、スケルトン、アニメーション、カメラを設定

② エフェクト制作

魔法、スパーク、雨、煙、爆発、崩壊

③ レンダリング

エフェクトをレイヤー毎にレンダリング

④ 合成

ぼかし、グロー、色調整など

# オペレーションを短縮したところ



① シーン構築
- VFXの作業シーンの作成

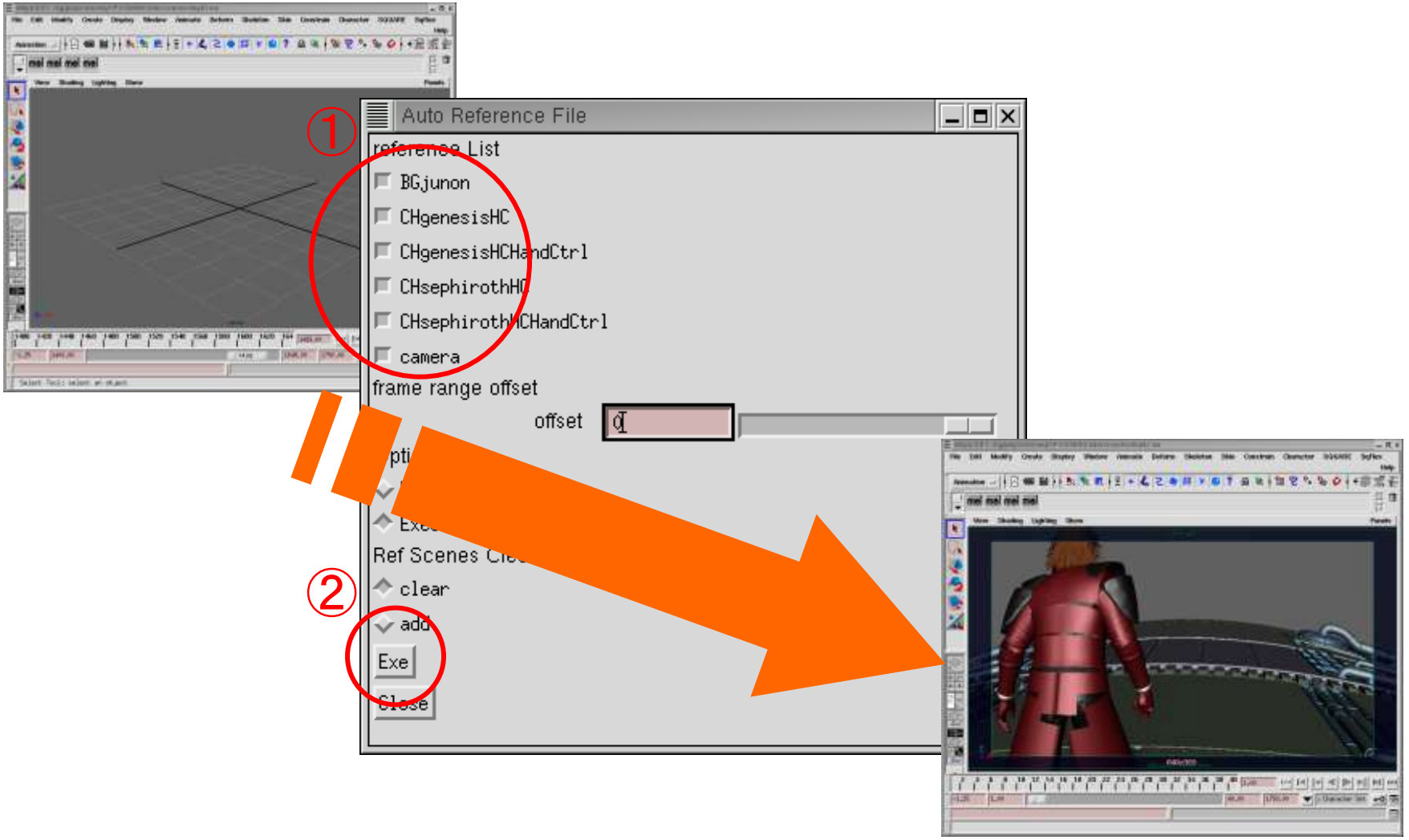
② エフェクト制作
- グリッド分割を作成
- 崩壊のセットアップを自動化

③ レンダリング
- パーティクルのレンダリング設定を簡略

④ 合成
- 素材の読み込み操作を短縮

# ①シーン構築





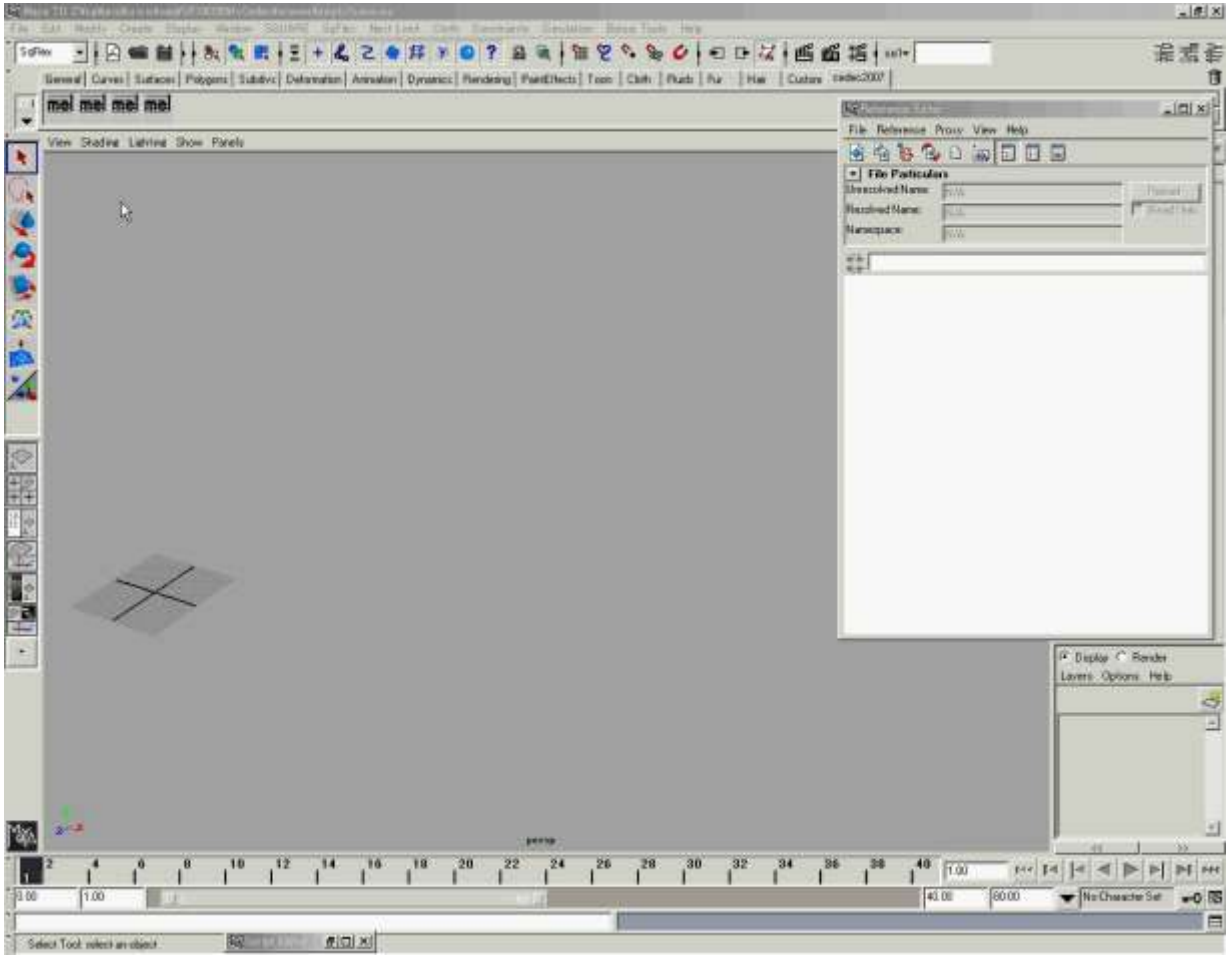
# ①シーン構築－２



- 作った理由
  - シーンを読み込む手間を省きたい
  - 間違わずに設定したい
- 短縮した操作
  - ① モデル、スケルトン、アニメーション、カメラの読み込み
  - ② カメラの切り替え
  - ③ フレームレンジの変更
  - ④ パーティクルの開始フレームの変更

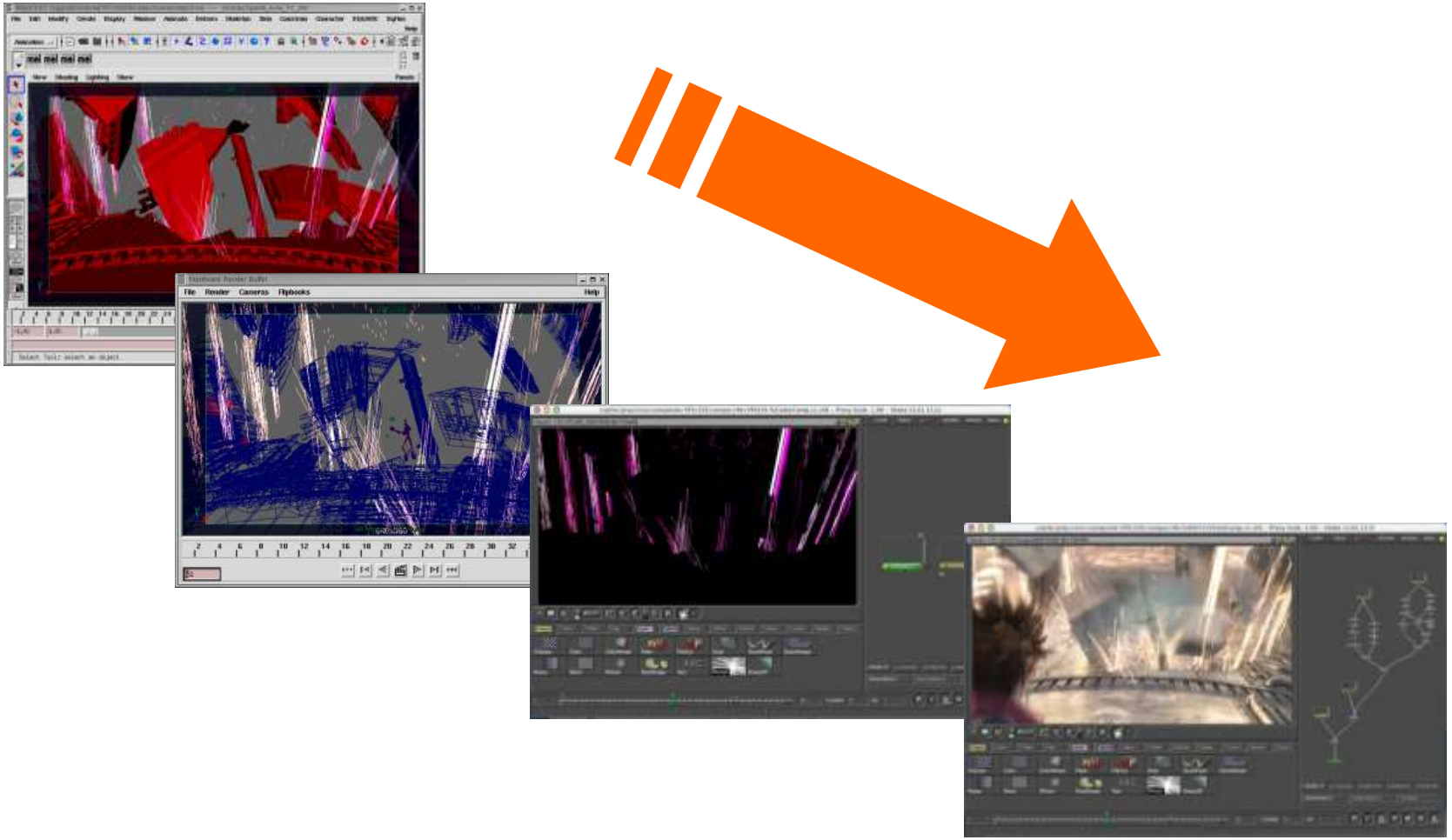
# 実演ムービー





# ③レンダリング～④合成

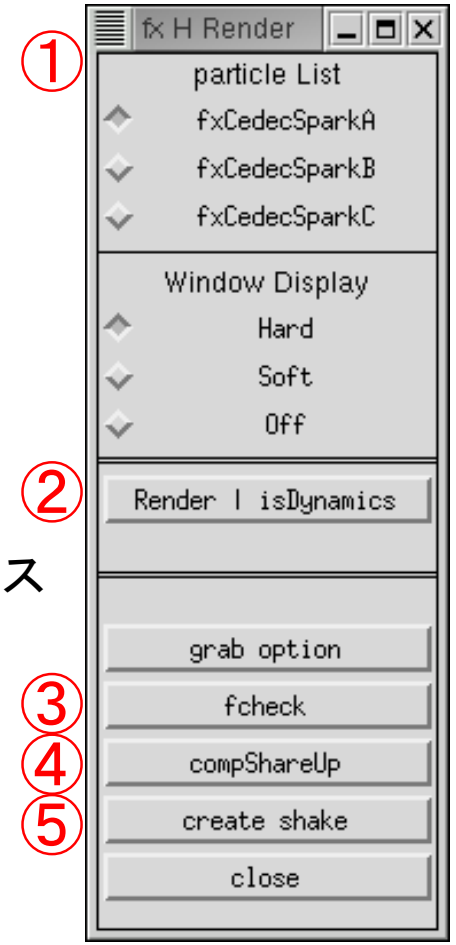




# ③レンダリング～④合成

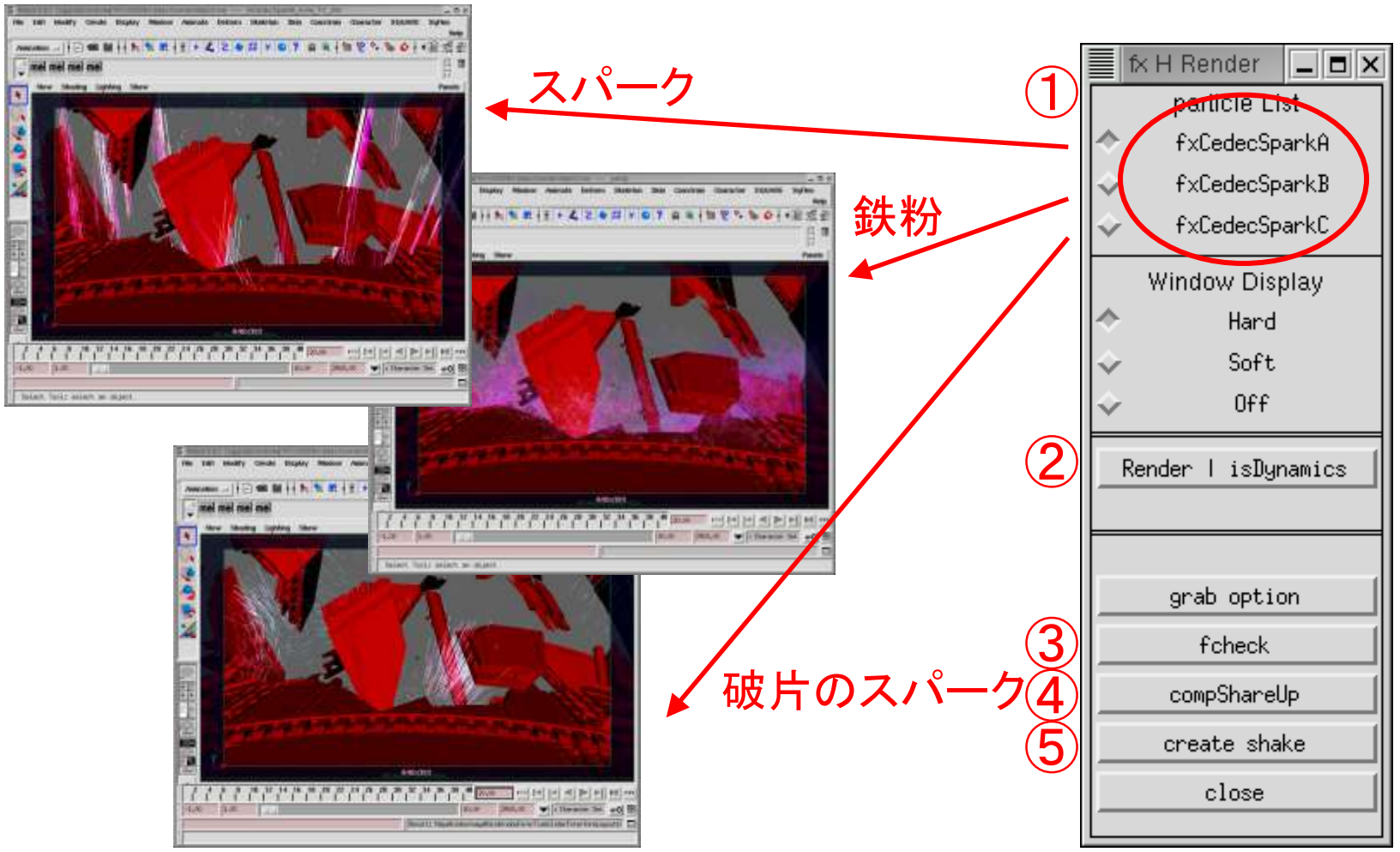


- 作った理由
  - レンダリング設定を複数保存したい
  - オプションを変更する手間を省きたい
  - 合成作業を直ぐに始められるようにしたい
- 短縮した操作
  - ① パーティクルの切り替え
  - ② レンダリングオプションの設定
    - ファイル名、フレームレンジ、モード、マルチパス
  - ③ プレビューの再生
  - ④ サーバーへの素材のコピー
  - ⑤ Shakeへの画像の読み込み



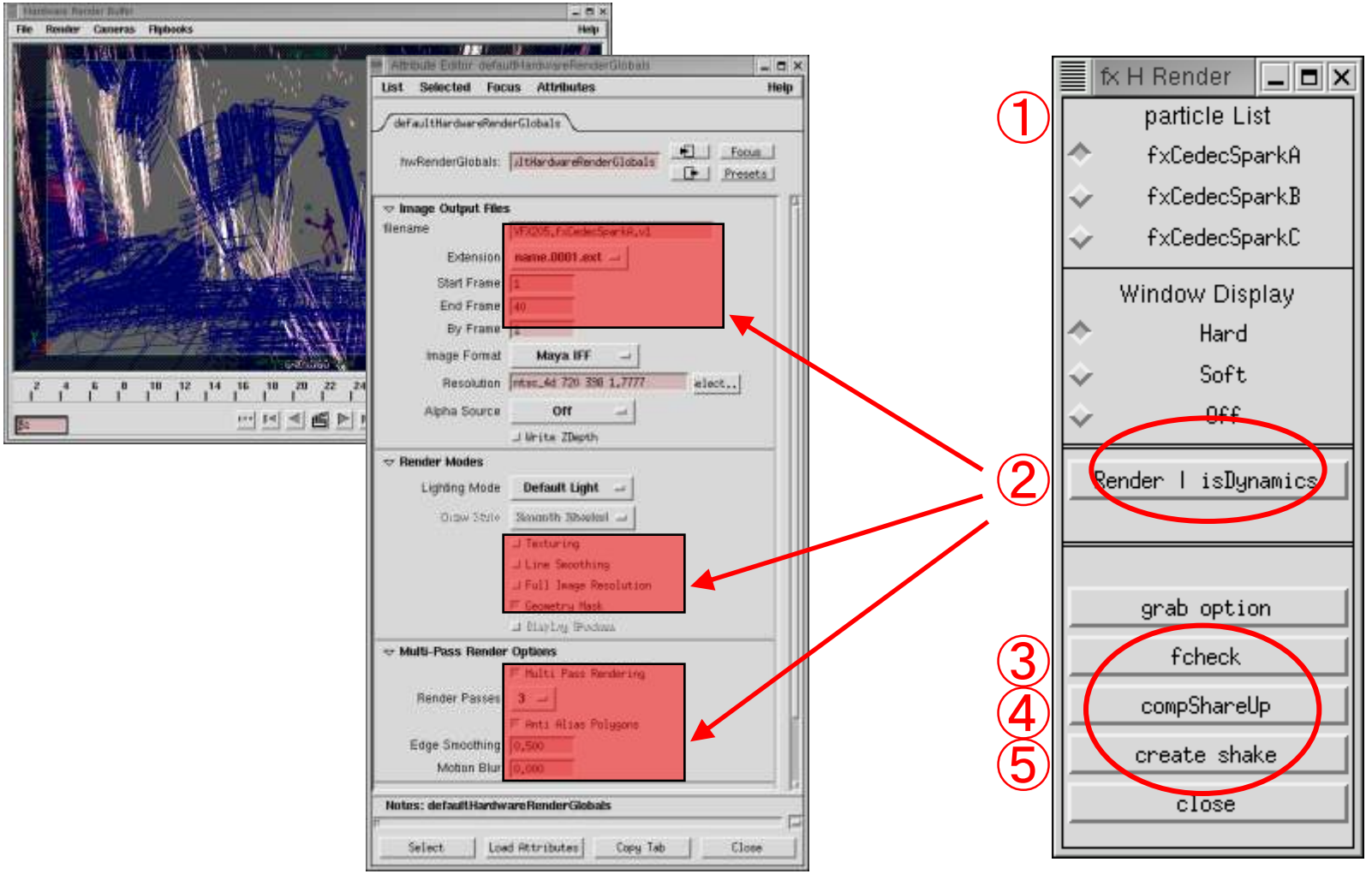
# ③レンダリング





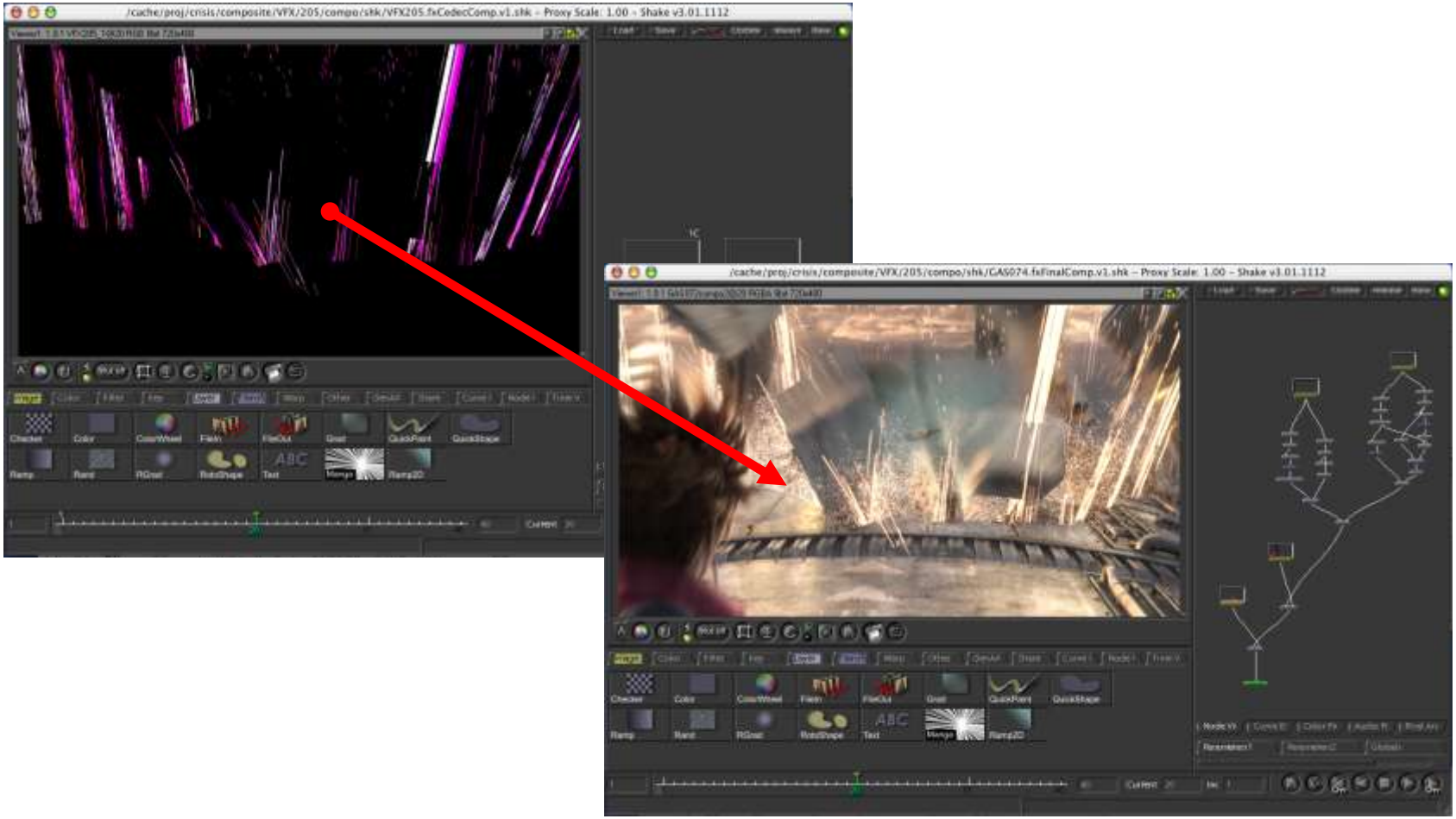
# ③レンダリング－２





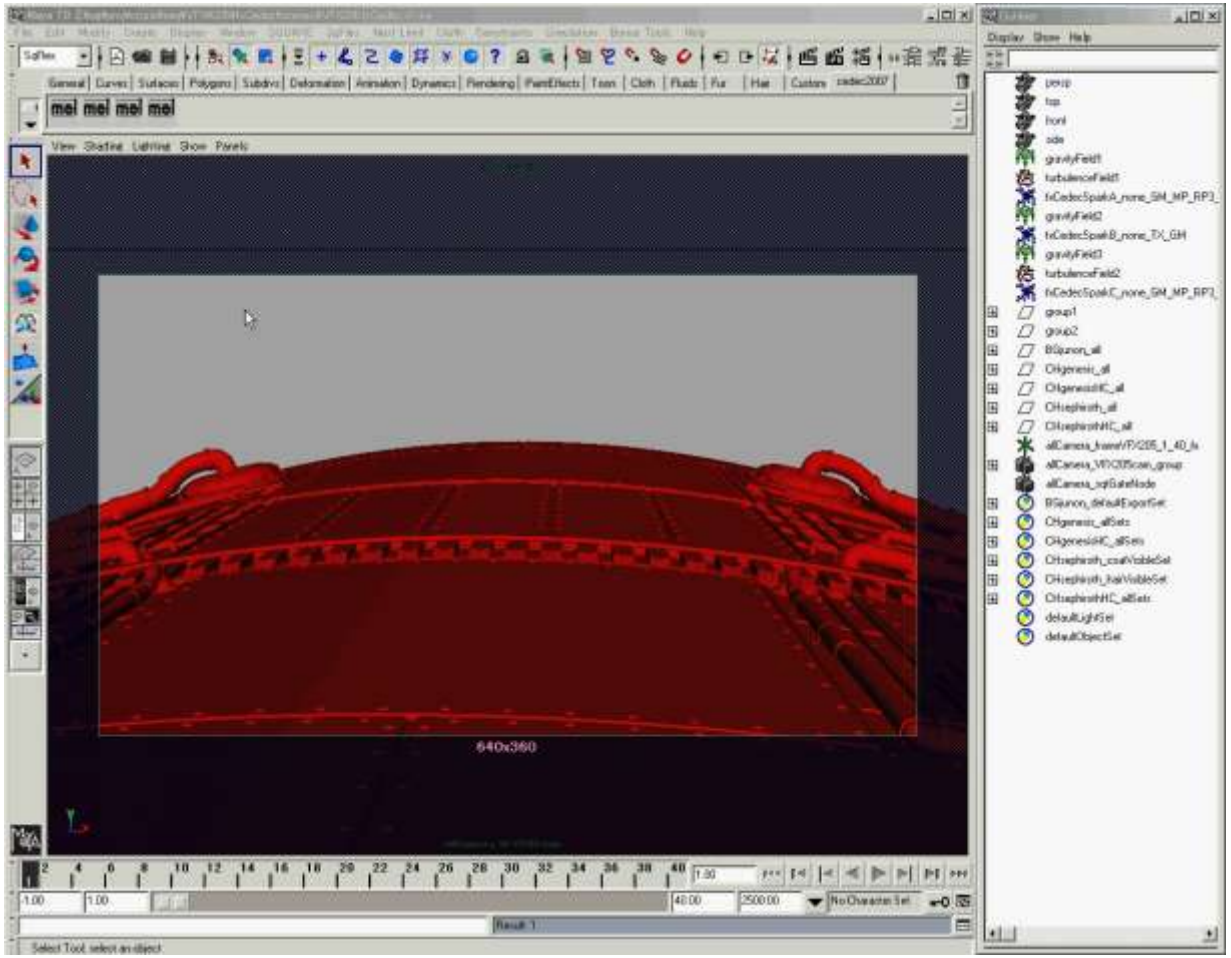
# ④合成





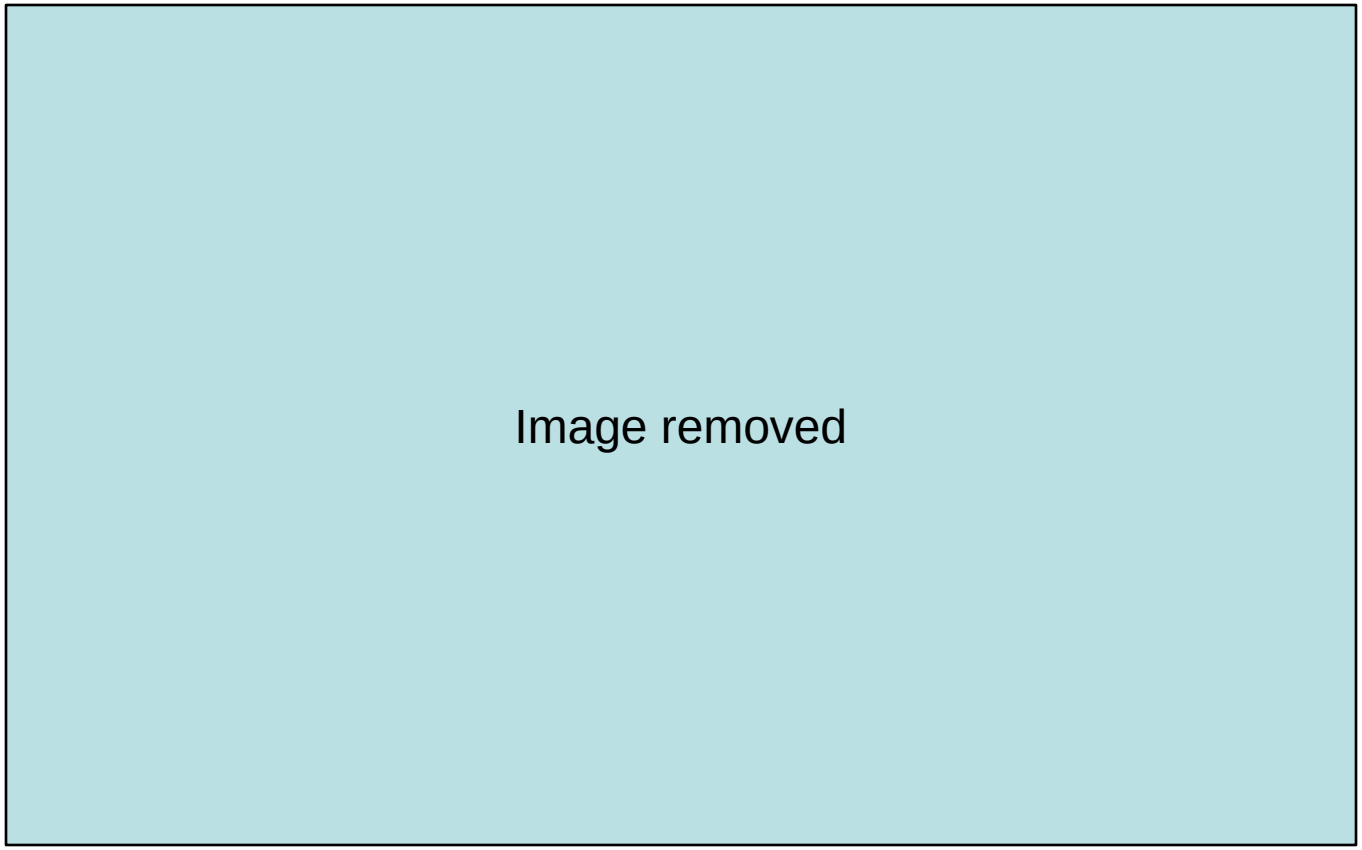
# 実演ムービー





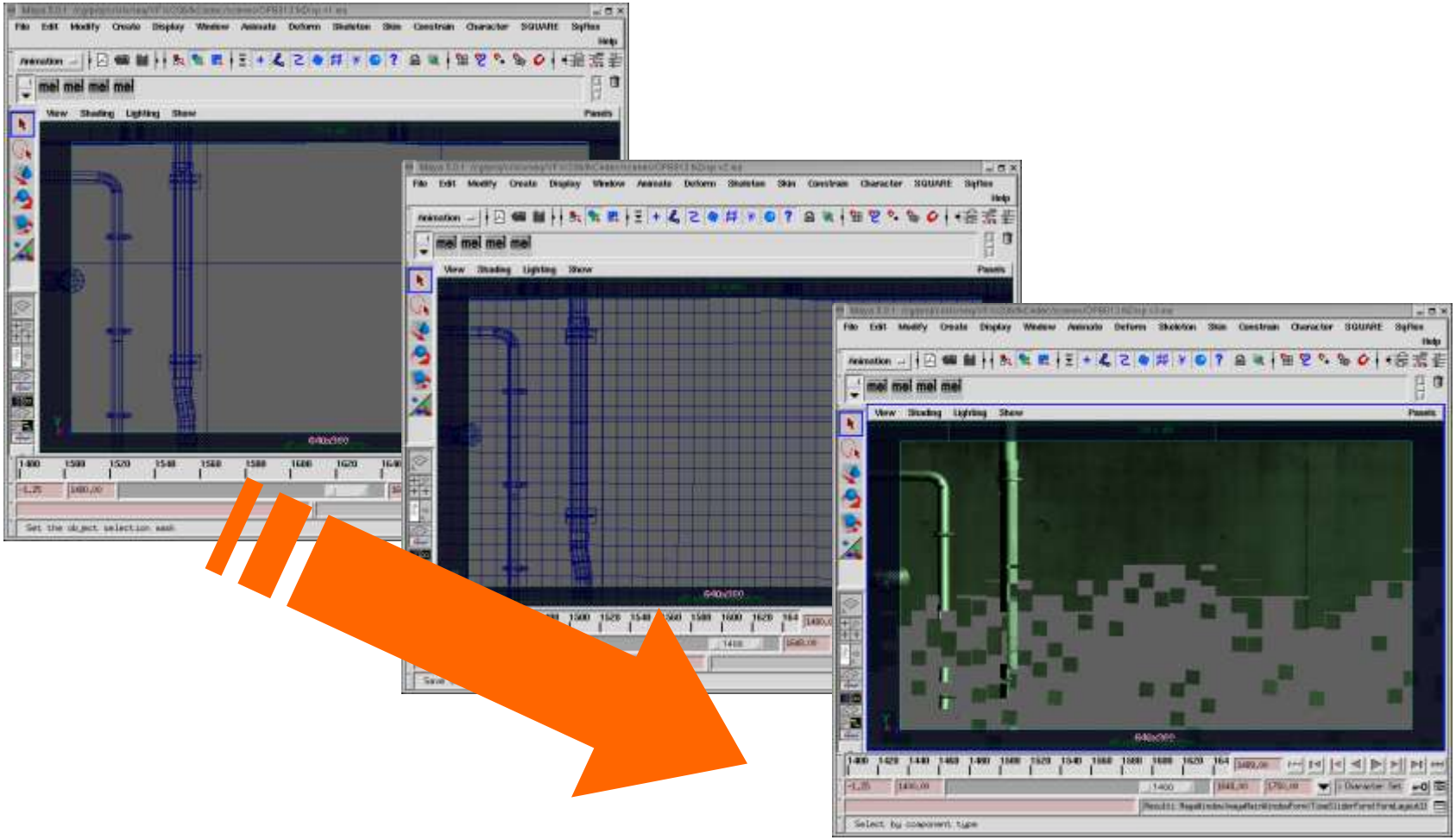
# 完成映像





# ②エフェクト制作





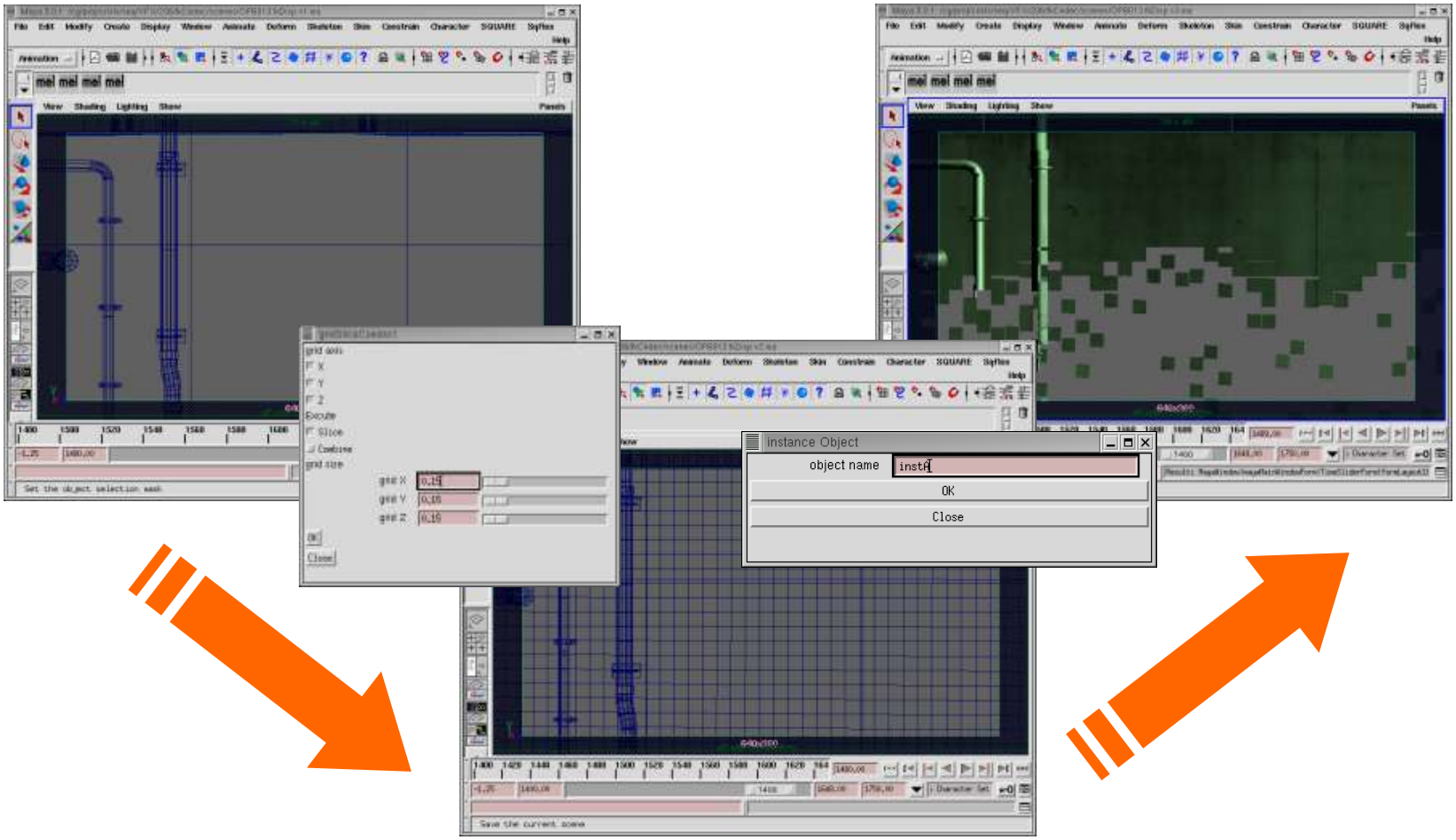
# ②エフェクト制作－2



- 作った理由
  - グリッド切断を可能にしたい
  - モデルの差し替えに素早く対応したい
- 短縮した操作
  - グリッド分割
    - ① モデルの切断
  - 崩壊のセットアップ
    - ② オブジェクト名の変更
    - ③ パーティクルを生成
    - ④ タイミング用のロケーターの作成
    - ⑤ フィールドの作成
    - ⑥ エクスプレッションの記述

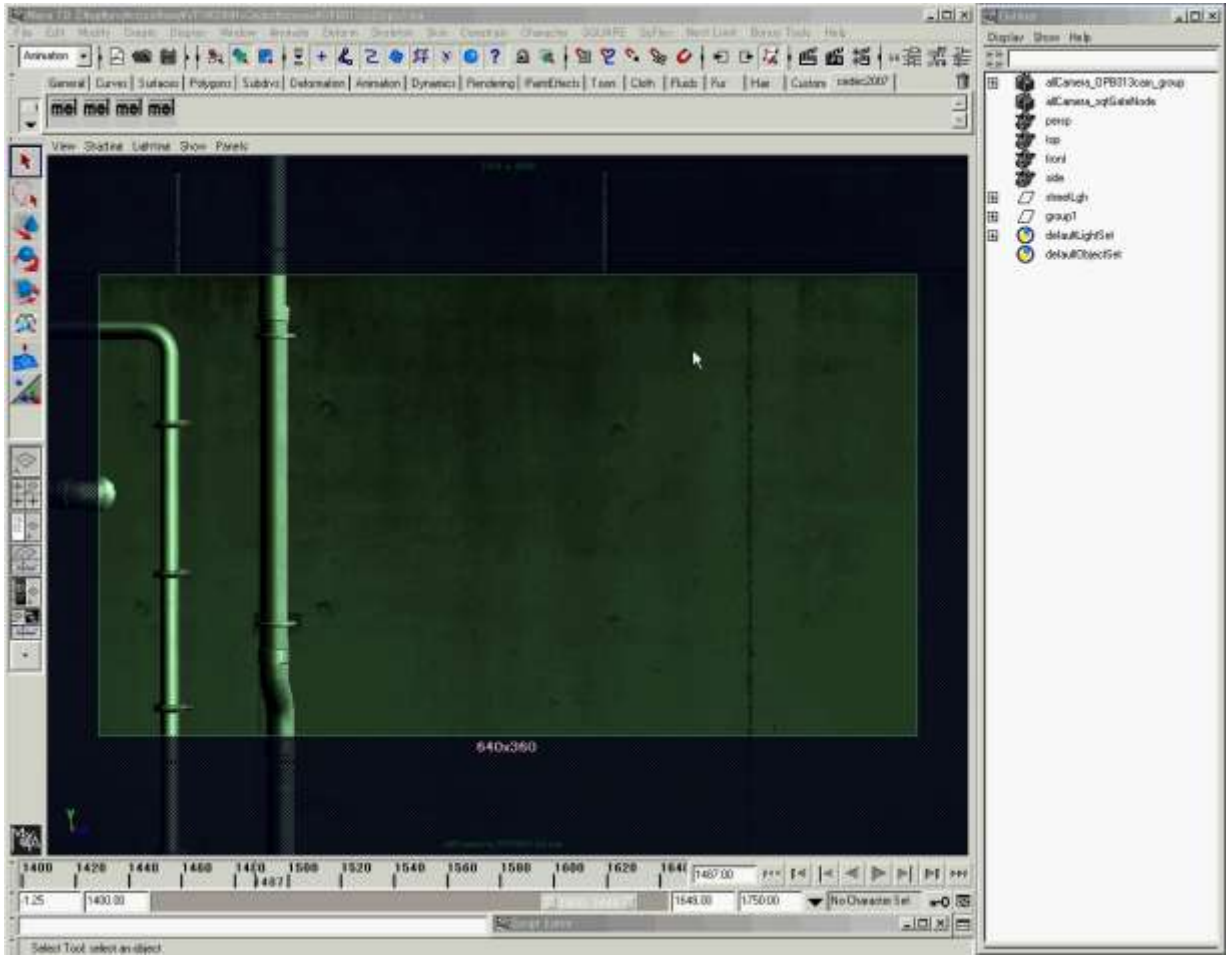
# ②エフェクト制作－３





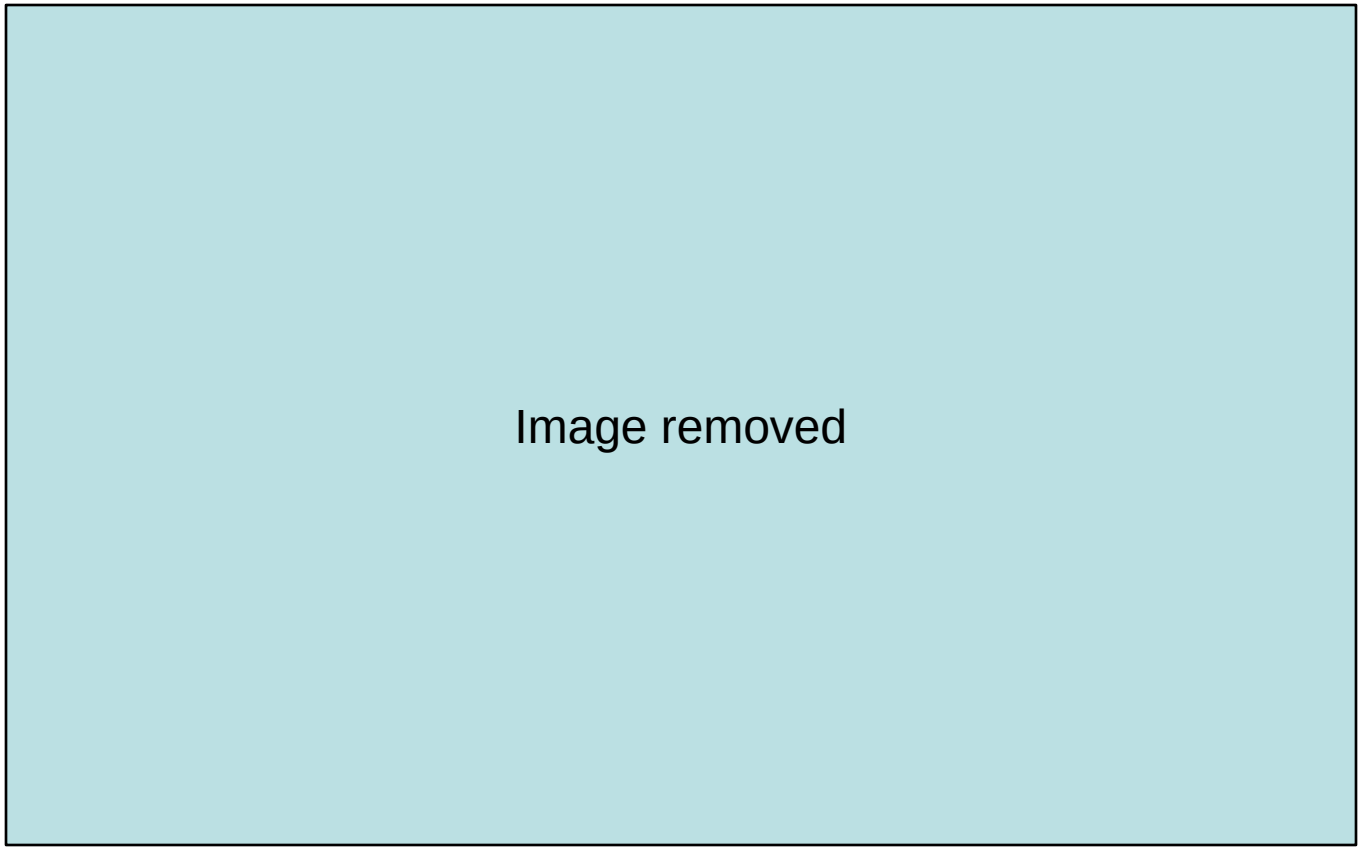
# 実演ムービー





# 完成映像

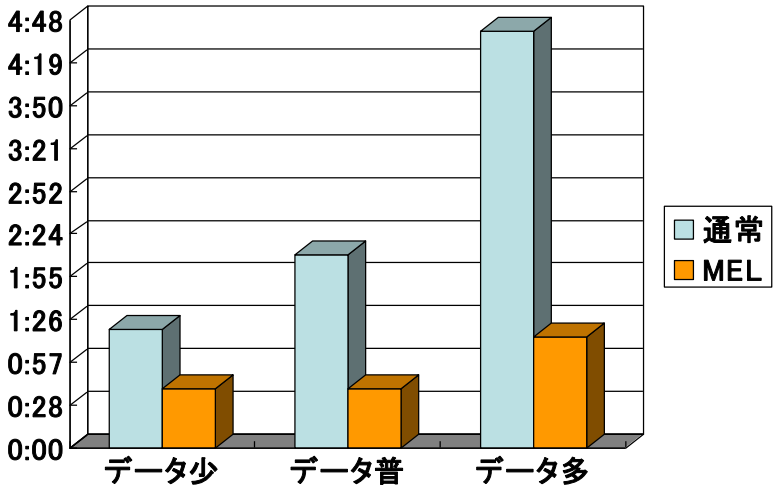




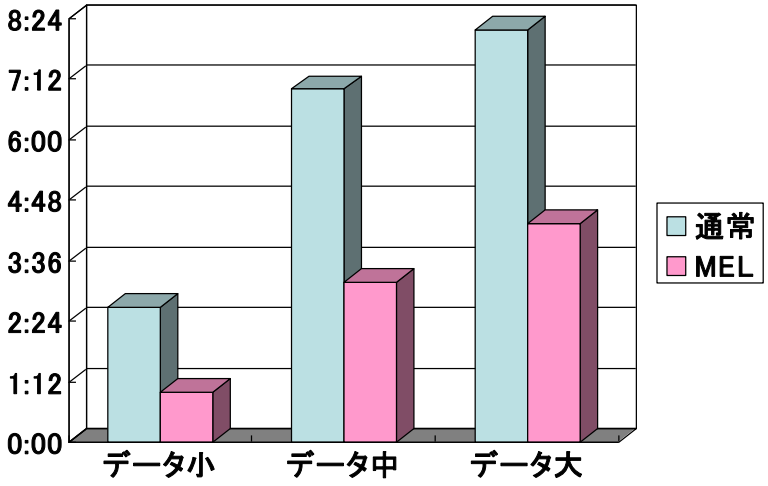
# 短縮時間



- 約60%（約32時間）の短縮
- オープニング（3分30秒）の場合
  - ショット数　60
  - VFX要素数　174




シーン構築




レンダリング

# 結果



オペレーション短縮で
時間は短縮できる

# 2. 効率化のポイントと仕組み



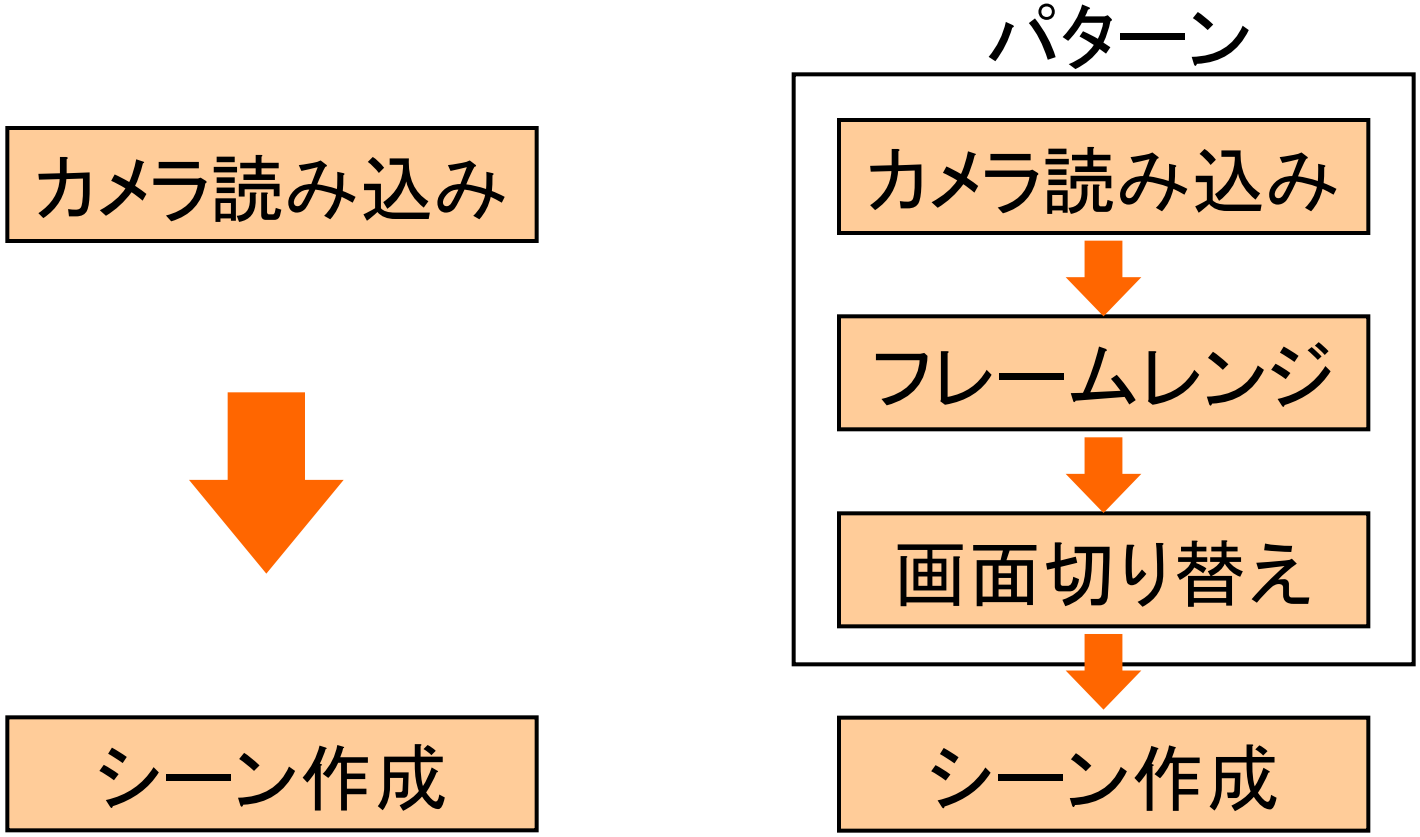
- オペレーション短縮って？

デザイナーのパターン化した操作を短縮する

# パターン化した操作



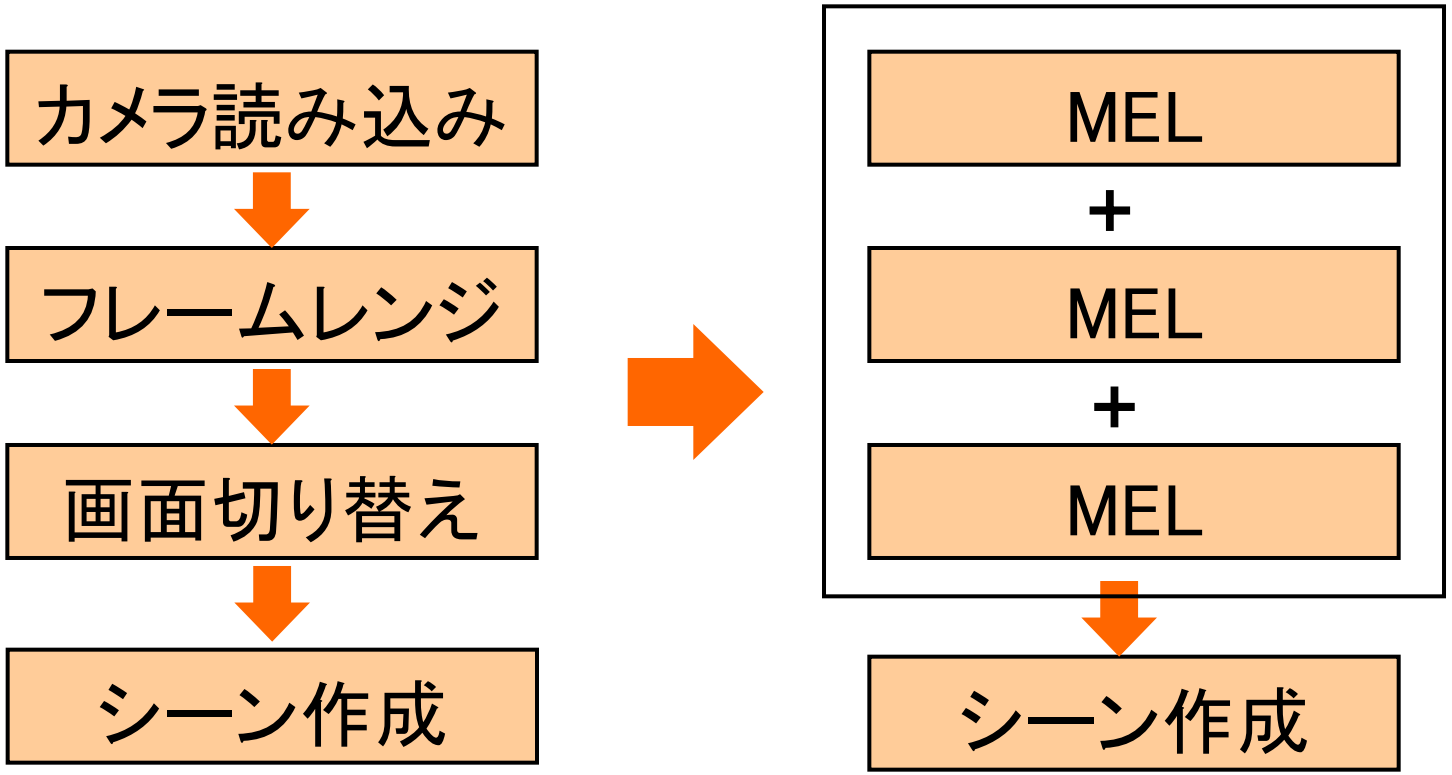
- カメラを読み込む場合



# スクリプトに置き換える



・ カメラを読み込む場合



# メリット



- 作業時間を短縮できる
- スケールメリットを活かせる
- 単調な作業を軽減する

# 特徴



- デザイナーが作りやすい
- 高度なプログラミングの知識は不要
- 短時間で作れる
- 使いながら調整し易い

# 効果的なところ

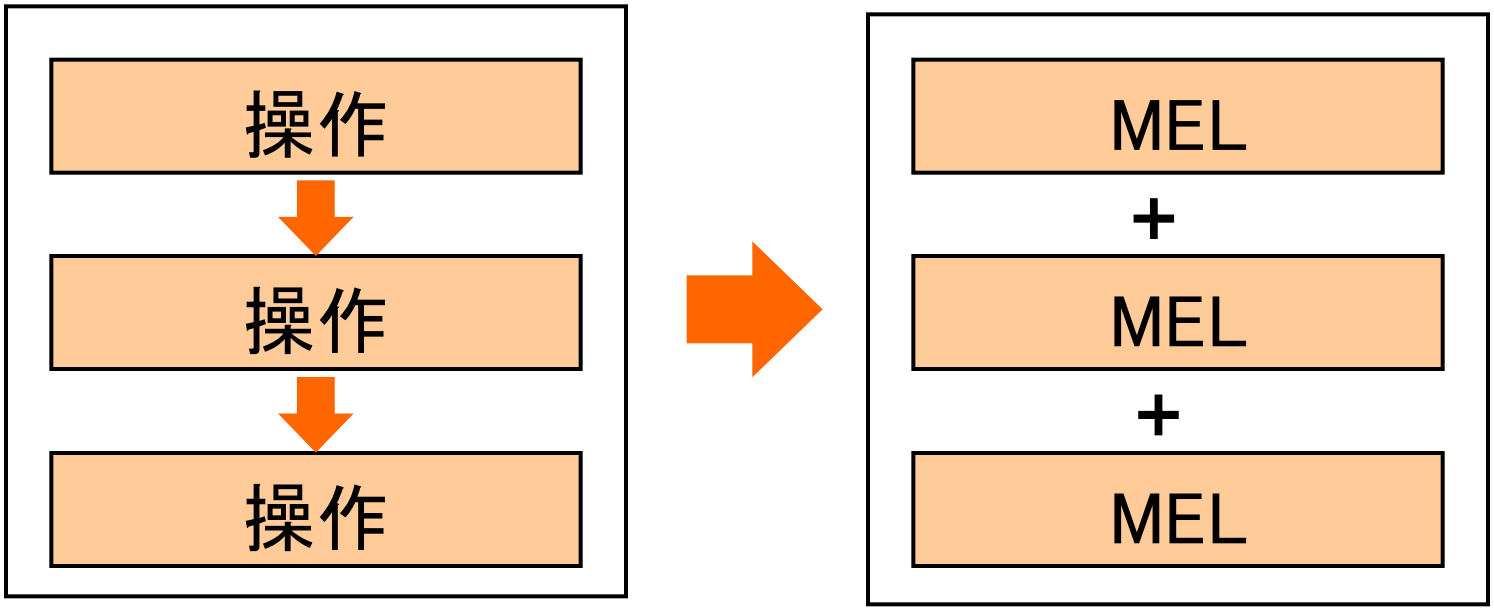


- 反復作業
  - ワークフロー
  - 修正箇所
- お決まりな設定
- 大量のオブジェクトを扱う時

# その仕組みは？



・ パターン化した操作をMELに置き換える



# 操作とMELは同じ

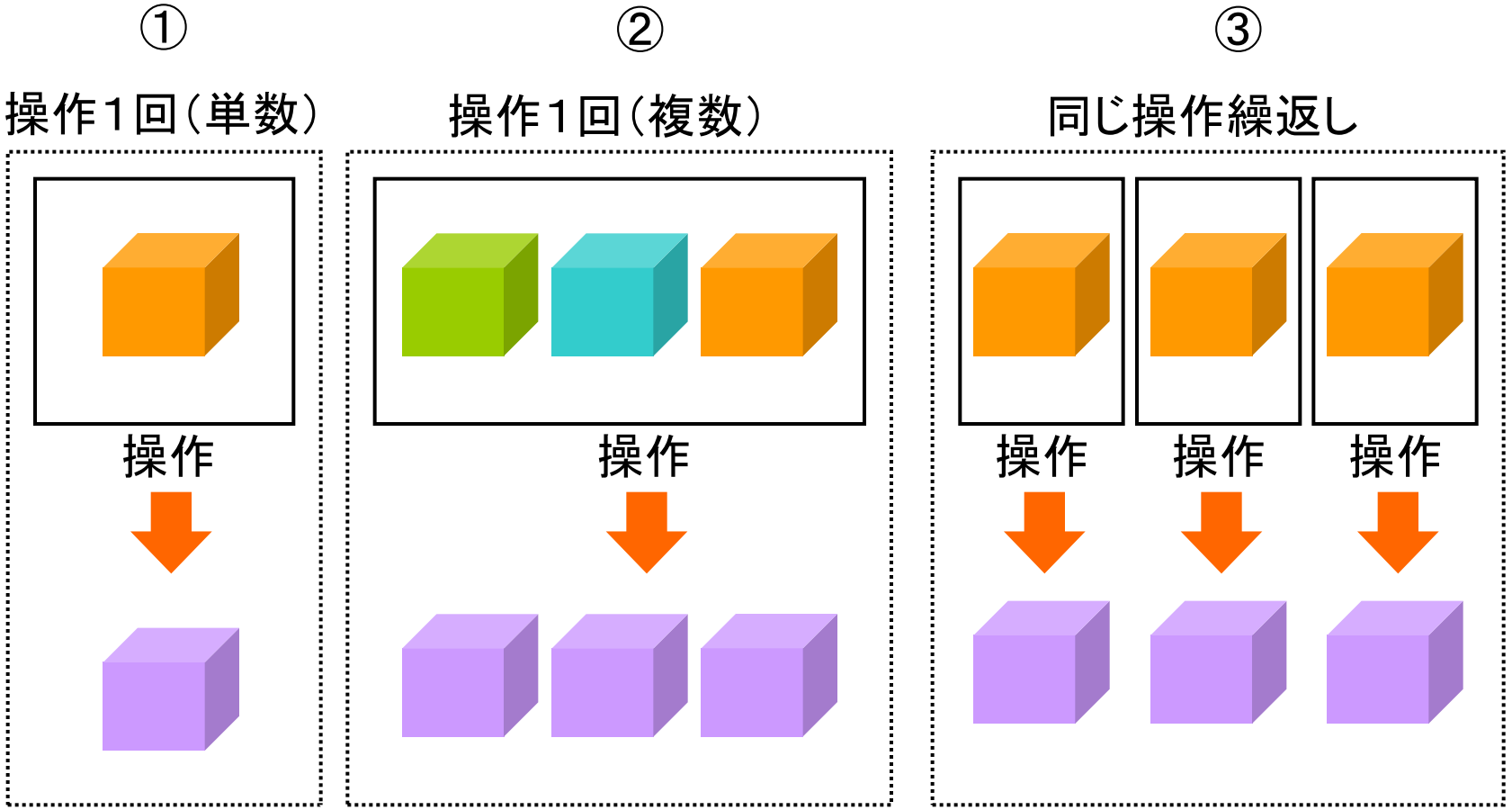


- MAYAを操作すると、その裏では対応するMELコマンドが実行されている
- メニューから操作できることはMELもできる

操作 = MEL

# 置き換えのパターン





# ① 操作1回(単数)



- 操作
  - 操作は1回
  - オブジェクトが1つ、又は選択が1箇所
    - メニューの操作など
- 形
  - 操作に対応するコマンドを実行する

# 色んな例



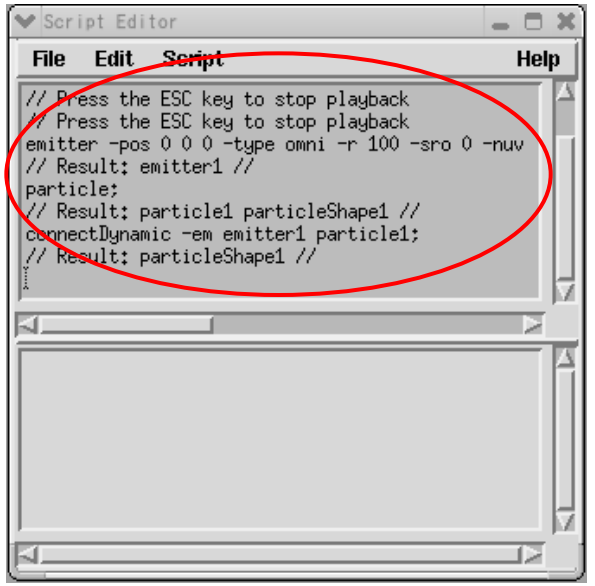
- フレームレンジの設定
  - コマンド → フレームレンジの設定　最小　最大
- カメラの切替
  - コマンド → ビューの設定　カメラ　パネル
- オプションの設定
  - コマンド → アトリビュート設定　アトリビュート名　値

# コマンドの調べ方



- Script editor のヒストリウインドウに表示される
- 表示されない場合は、Echo All Commandsをチェック

| | | |
|---|---|---|
| アトリビュートの設定 | → | setAttr |
| ポリゴンの切断 | → | polyCut |
| リファレンスの設定 | → | file |



# ②操作１回（複数）



- 操作
  - 操作は1回
  - 複数のオブジェクト、複数の情報
- 形
  - リストを使ってコマンドを実行する
    - リストは要素の集合

リスト←{要素１　要素２　要素3}
コマンド　リスト

# パーティクル生成の場合



操作

① 位置を取得

② パーティクルを生成

```
点リスト←{点１　点２　点３}
パーティクル生成　点リスト
```

# ③同じ操作繰返し



- 操作
  - 操作をするのは複数回
  - 複数のオブジェクト、複数の情報
  - ①の操作の連続
- 形
  - リストとfor文の組み合わせ
    - for文はリストの要素を順に取り出し、要素の数だけ処理を繰り返す

```
リスト←{要素１、要素２、要素3}
for(要素 in リスト){
        コマンド
}
```

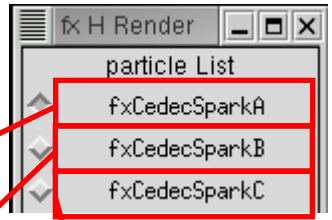
# レンダリング設定の場合



操作

① パーティクルを選ぶ

② isDynamicを1に




```
リスト←{パーティクル1、パーティクル2、パーティクル3}
for(パーティクル in リスト){
        アトリビュートの設定　パーティクル.isDynamic 1
}
```

# 3. 代表的なスクリプトの紹介

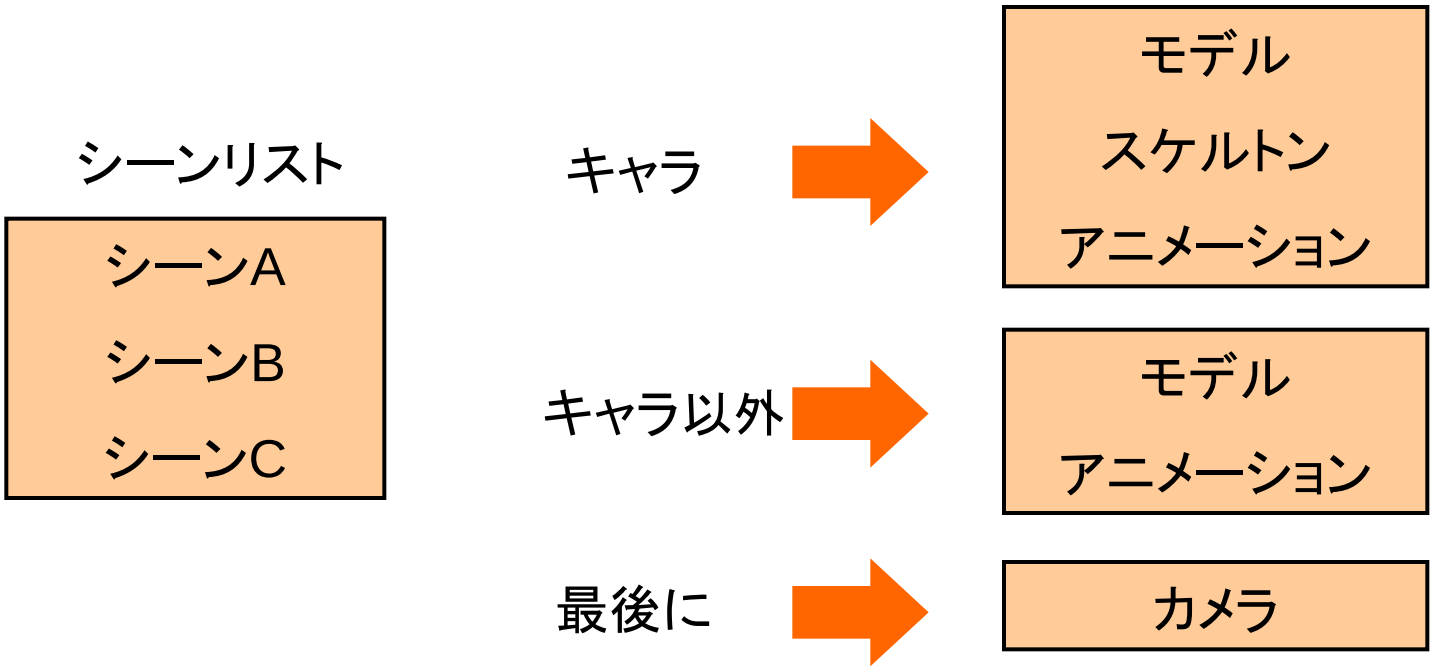


1. シーン構築
2. レンダリング設定
3. 崩壊セットアップ

# ①シーン構築



- アニメーションのシーンリストを基に全リファレンスシーンを設定する
- 最後にカメラを設定する



# 仕組み



- ファイル名からすべての情報を取得
  - 種類（キャラクタ、背景、スケルトンなど）
  - LOD（lowモデル、midモデル、highモデルなど）
  - ショット名
  - モデル名

シーン名 → ショット名＋種類＋LOD＋モデル名

- ディレクトリ構造とファイル名が対応

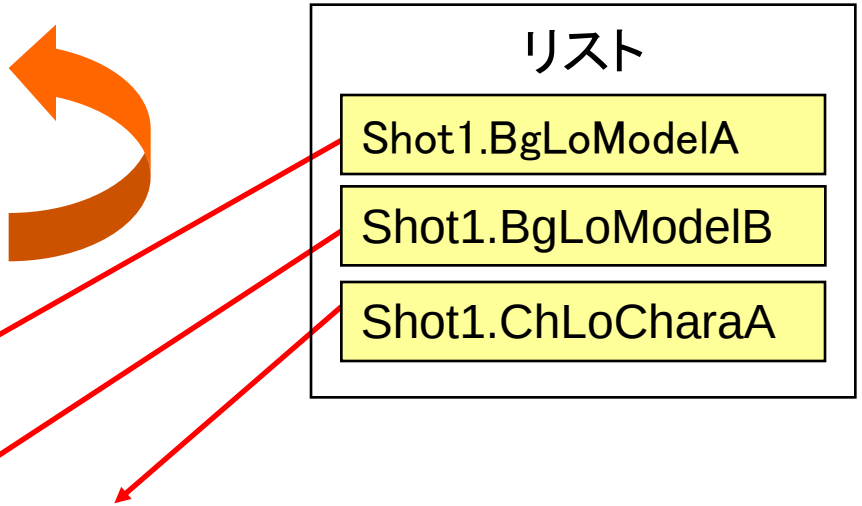
／ショット名／モデルタイプ／LOD／シーン名

# リファレンス設定を自動化



操作

① ファイルを選択
② スケルトンファイルの設定
③ モデルファイルの設定
④ アニメーションファイルの設定
⑤ カメラの設定

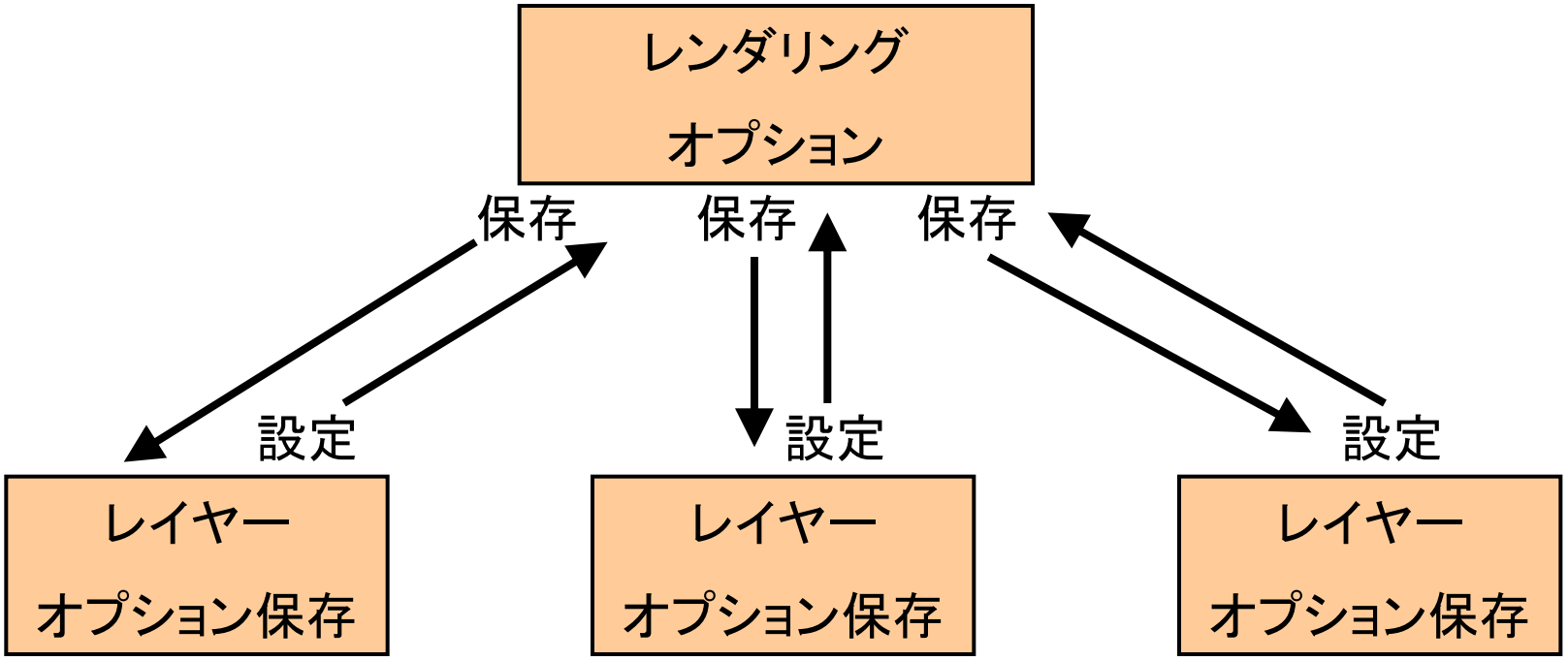



```
リスト＝{ファイル名１、ファイル名２、ファイル名３}
for(ファイル名 in リスト){
        キャラ場合 → スケルトンファイルの設定
        モデルファイルの設定
        アニメーションファイルの設定
}
カメラの設定
```

# ②レンダリング設定



- 複数のレンダリング設定を保存できる
- レンダリング設定を自動化



# 仕組み



- パーティクル名を書式化してオプションを記述
- オプションをコード化して管理

# 書式



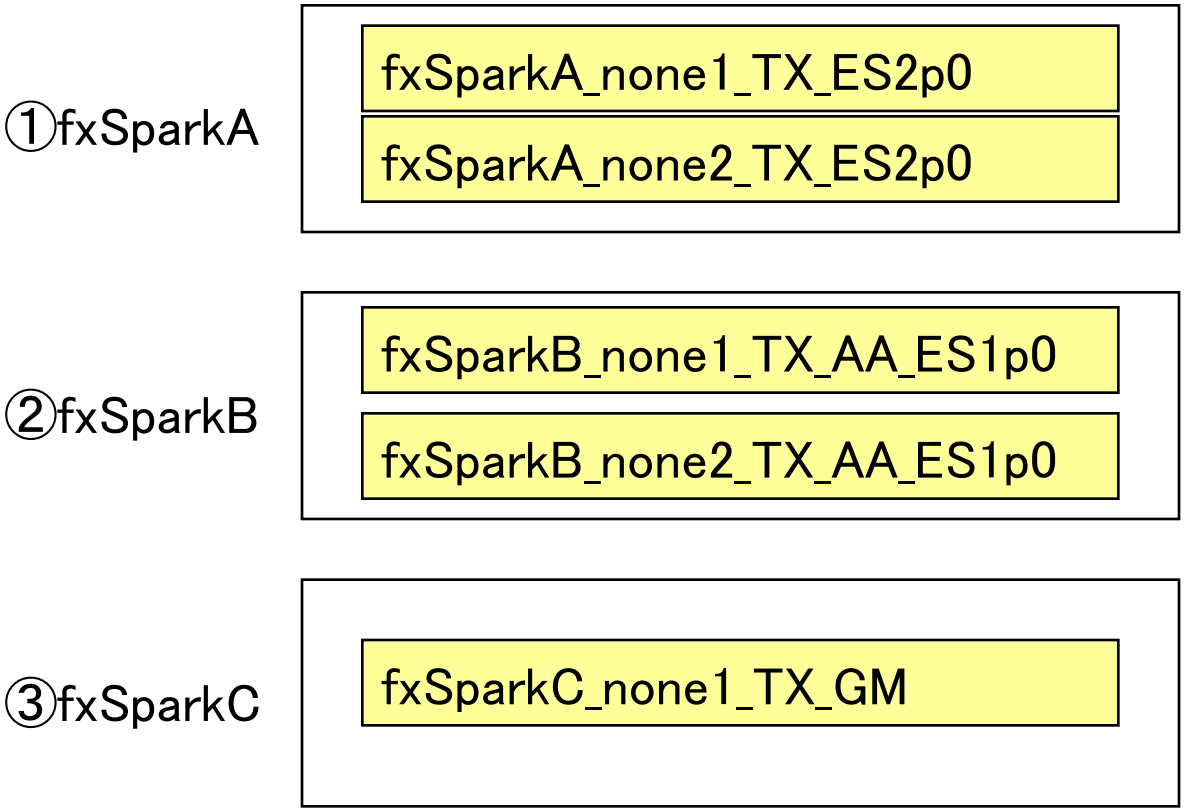
## fx[レイヤー]_[コメント]_[オプション]

- レイヤー
  - レイヤー名を記述
  - レンダリング時はファイル名
- コメント
  - コメントを記述
  - 1レイヤーに複数のパーティクルを利用する時は名前の重複を避けるために利用
- オプション
  - オプションコードを記述
  - 複数の時は、”_”を挟む

# 使用例



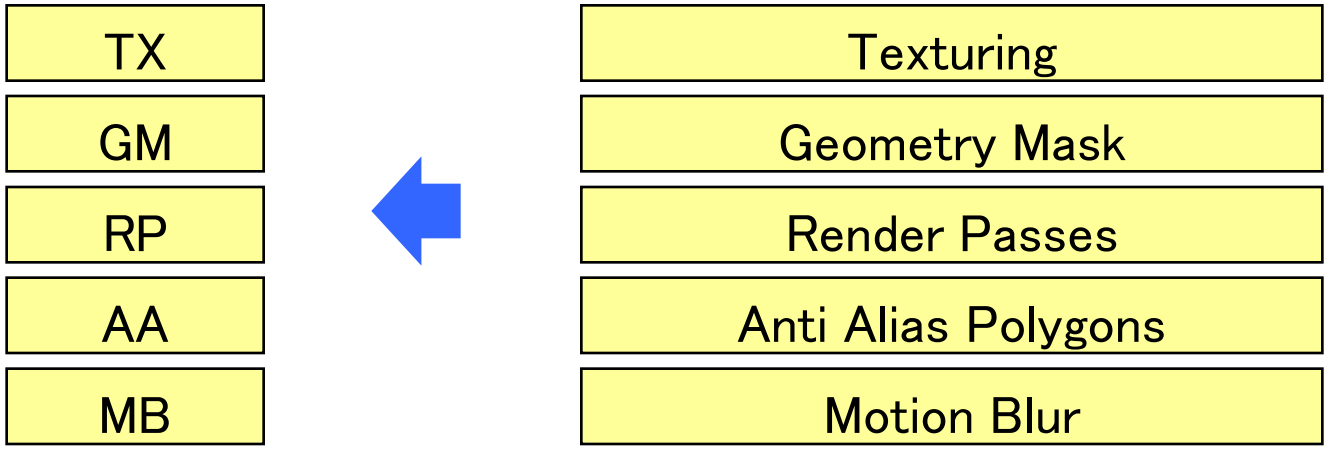
- 複数のパーティクルも可能



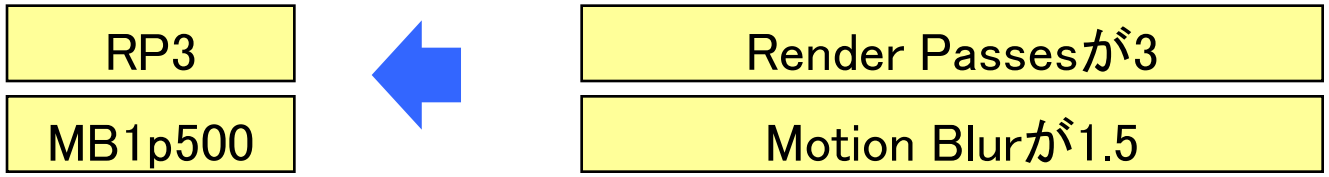
# オプションコード



- ２文字で表現



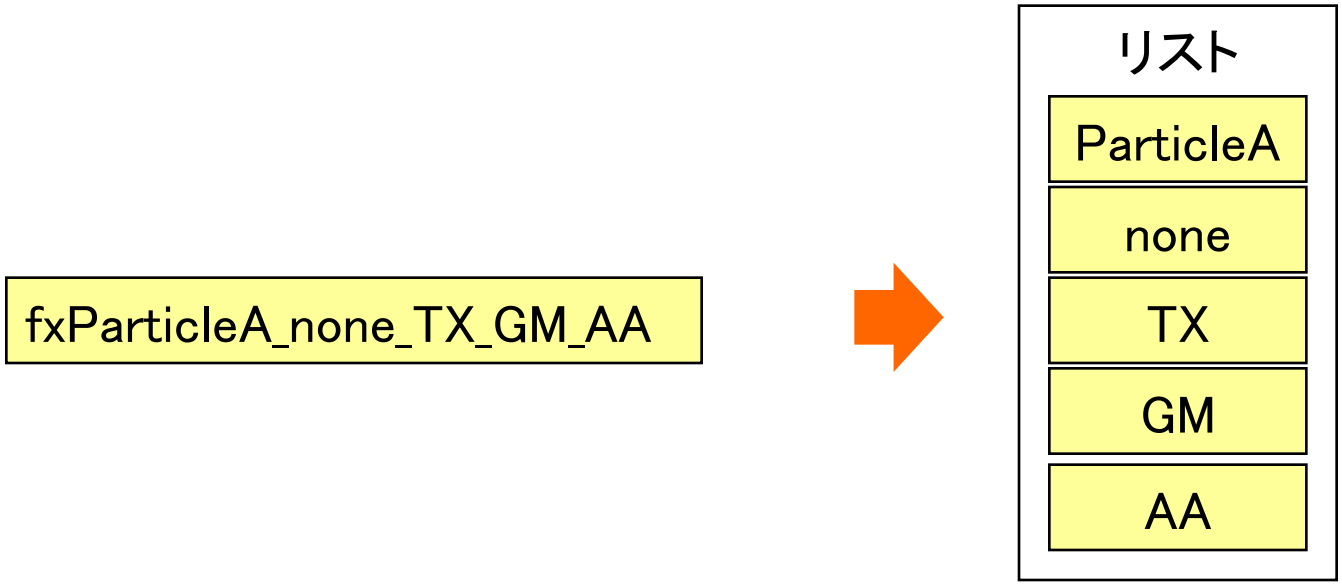

- 数値はコードに続けて記述
- 小数点は”p”で記述（”.”は使えない）



# オプションコードの解析



- 文字列を分解するtokenizeコマンドを利用
- パーティクル名からリストに変換



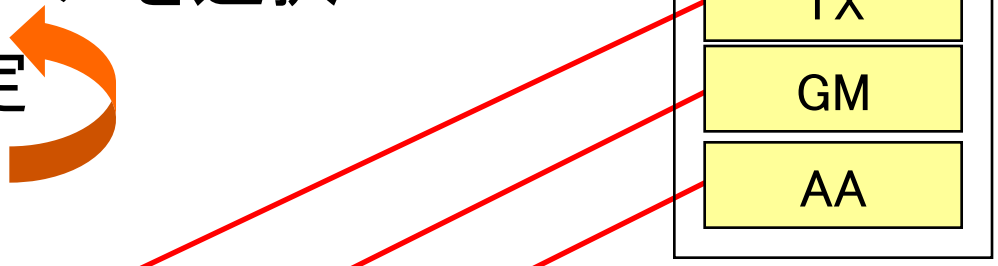
# レンダリング設定を自動化



操作

① レンダーグローバルを選択

② オプションを設定

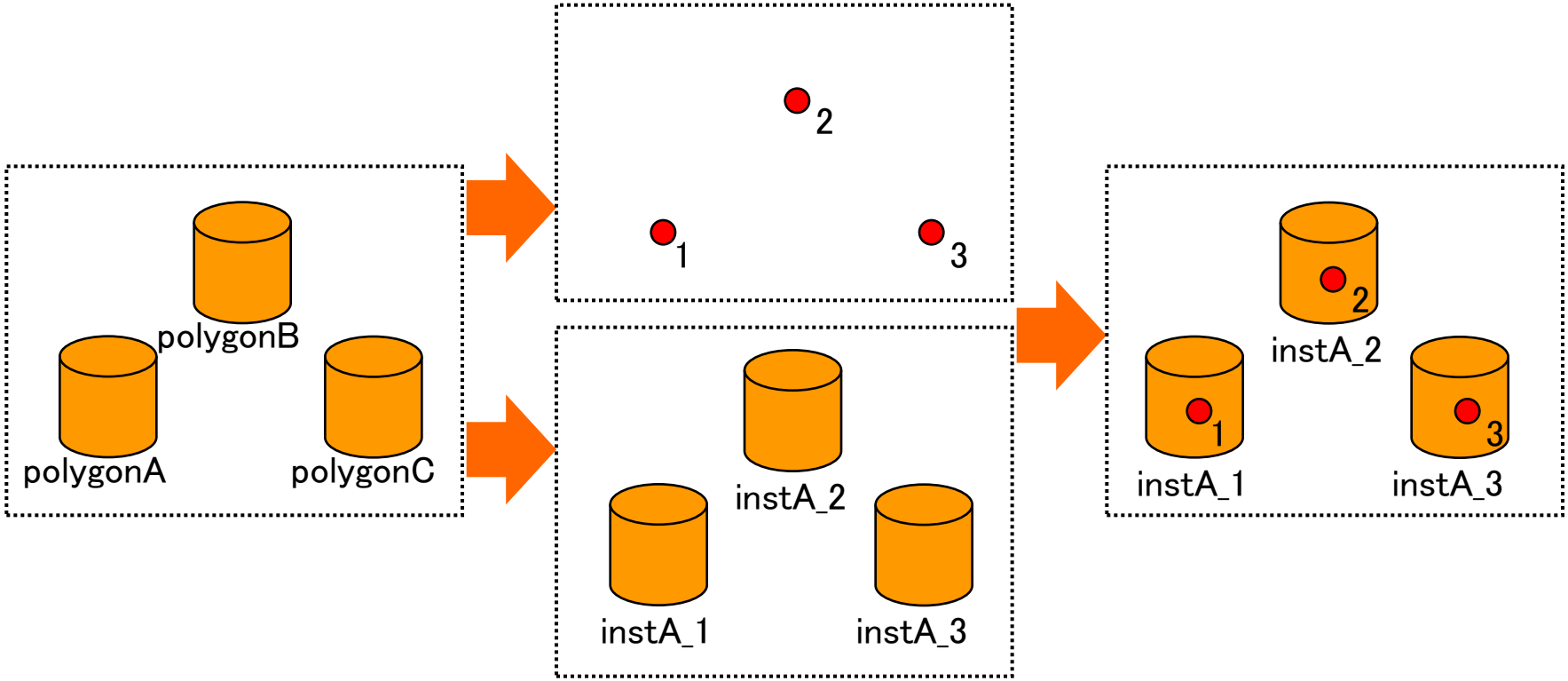



```
リスト＝{コード1、コード2、コード3}
for(コード in リスト){
        TXの場合 → Texturingを設定
        GMの場合→ Geometry Maskを設定
        AAの場合 → Anti Alias Polygonsを設定
}
```

# ③崩壊セットアップ



- エクスプレッションからコマンドを実行してオブジェクトを移動
- パーティクルIDと対になる名前でオブジェクトを管理



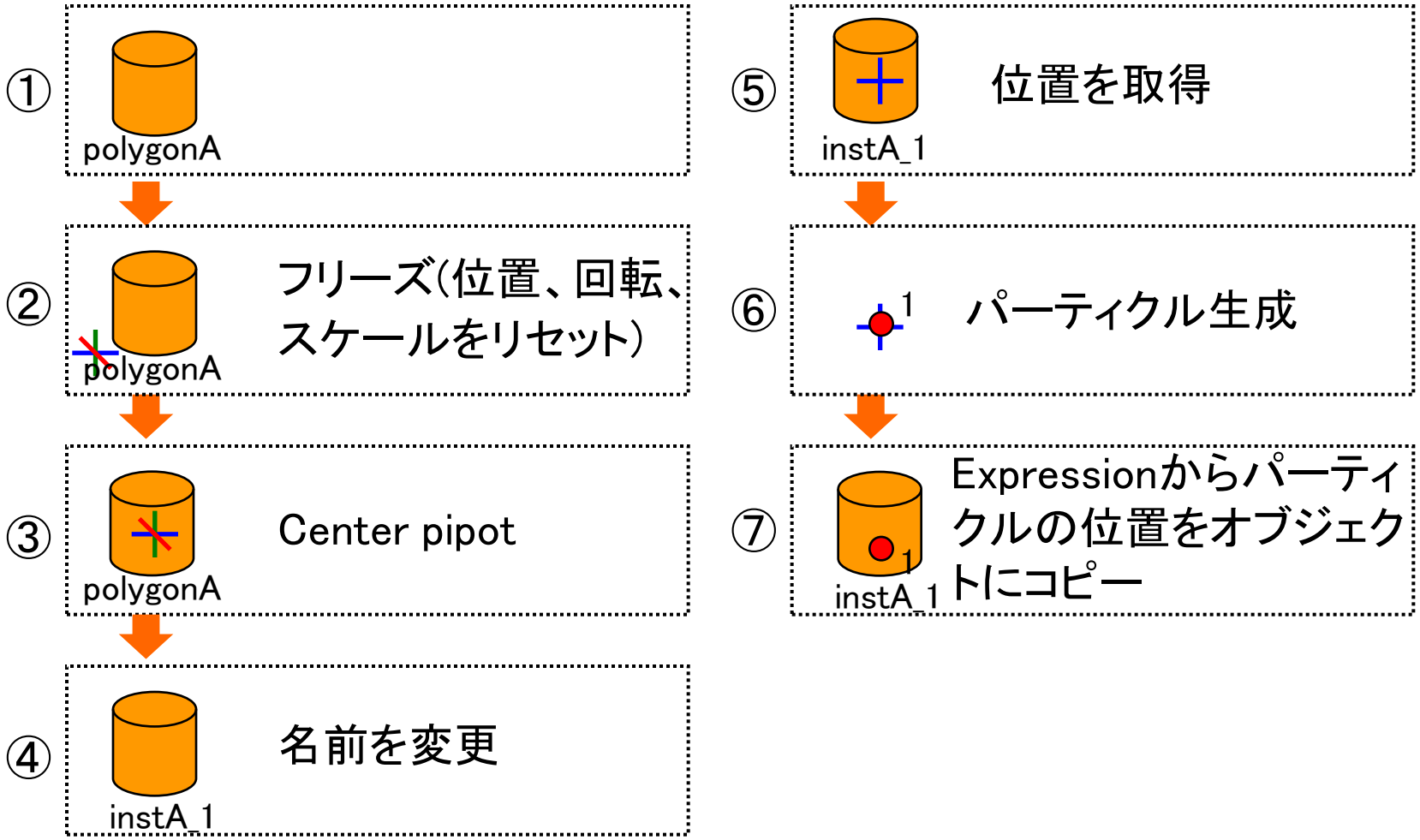
# なぜインスタンスじゃないのか



- インスタンスよりも非常に高速
  - 操作画面を再描画する時に計算されない
  - 余分な計算をしない
- 大量なオブジェクトを扱える
  - インスタンスは2000個が限界
  - xformコマンドは5万個でも大丈夫
    - setAttr ＞ move,rotate,scale ＞ xform

# セットアップ手順



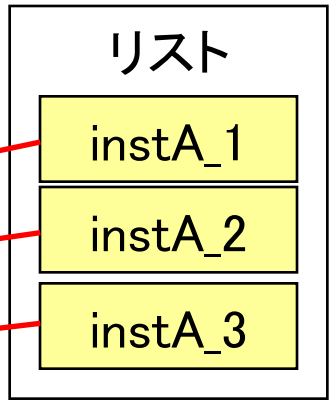


# パーティクル生成の場合



操作

① オブジェクトを選択
② フリーズ
③ 軸を中心に
④ リネーム
⑤ 位置を取得
⑥ 点を打って
⑦ パーティクルを生成
⑧ エクスプレッションの追加



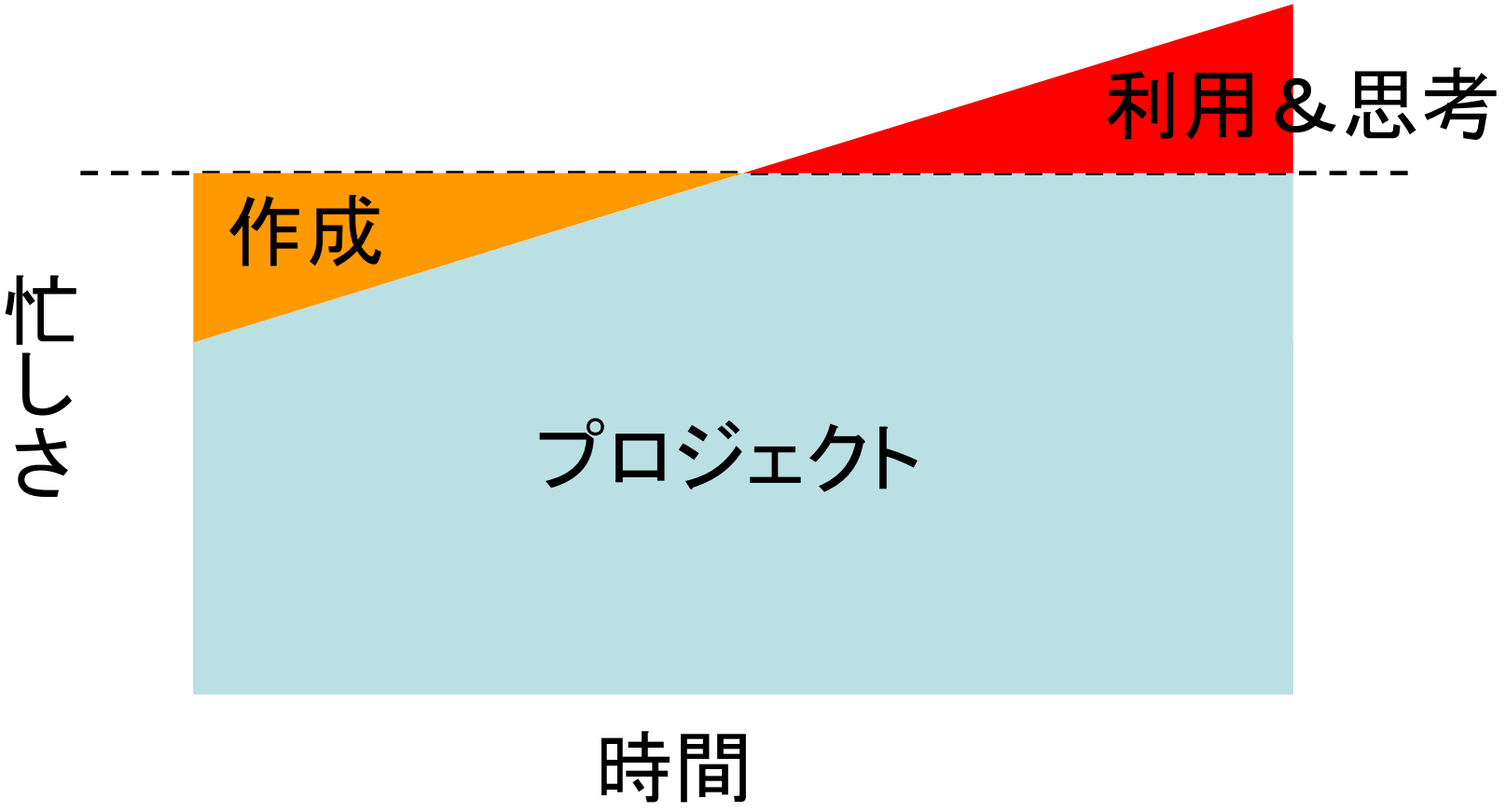

```
リスト←{オブジェクト1、オブジェクト2、オブジェクト3}
for( オブジェクト in リスト) {
        フリーズ   オブジェクト
        軸を中心   オブジェクト
        リネーム   オブジェクト
        点リスト(追加)←位置を取得   オブジェクト
}
パーティクル生成     点リスト
エクスプレッションの追加
```

# スクリプト作成のタイミング



# スクリプト作成のタイミング





# 効率化を図ってセンスを磨こう

# 4. 質疑応答

CEDEC 2007
CESA DEVELOPERS CONFERENCE