ゲーム制作現場での DCCツール活用事例

リアルタイムシェーダーの周辺から

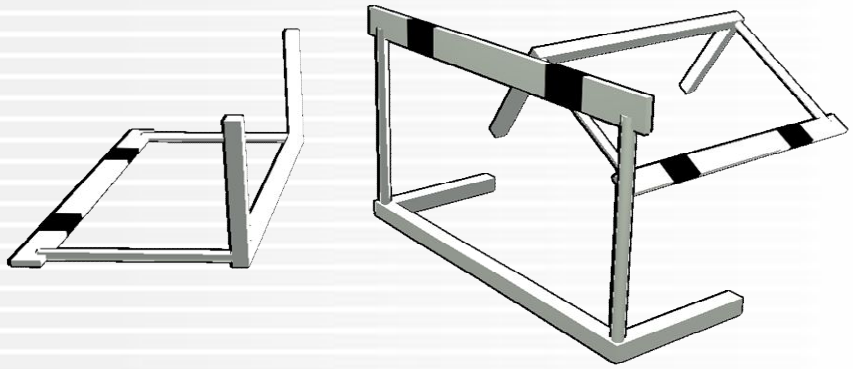
モデレーター
(株)ポリフォニー・デジタル
手島　孝人

アジェンダ

1. (株)セガ
   - 林　洋人、麓　一博

2. (株)ディンプス
   - 後藤　修一、前田　敦史

3. (株)スクウェア・エニックス
   - 太田 嘉彦、佐々木 隆典



標準シェーダースタイルにおける XSI

(株)セガ　CS R&D推進部
林　洋人
麓　一博

自己紹介

- CS R&D推進部
  - テクニカルサポートを行なう部署
  - 社内プロジェクトや外注プロジェクトにグラフィックス開発環境を提供
  - 林
    - 設計
    - （あまり）プログラムしないプログラマー
  - 麓
    - テクニカルアーティスト
    - （あまり）デザインしないデザイナー

シェーダーといっても…

- リアルタイムシェーダー周辺について話しやすくするために少し整理
  - シェーダーを使用した開発全体のポイント
  - シェーダーパイプラインのポイント（＝スタイル）

シェーダー開発のポイント（その１）

- シェーダーはマテリアルではない
  - 同じマテリアルでもシェーダーはゲームシーンで動的に変化する
  - 同一シェーダーで別パラメーター
    - ライティング／フォグ／スキニング／モーフィング
    - 動的シャドウマップ／動的環境マップ
- 実際に使用されるシェーダーは１マテリアルにつき１つではない

シェーダー開発のポイント（その２）

- サンプルに特化したシェーダーとゲーム内で使用するシェーダーは別物

- 表現に関する部分のみを指定、エクスポートする必要がある
  - DCCツールの関与すべき部分としない部分を区別

シェーダー開発のポイント（その３）

- 独立性の強い処理用シェーダーはワークフローをあまり考えなくてもよい
  - ポストエフェクト（グレア、被写界深度）
  - シャドウマップ
  - スキニング

シェーダーパイプライン

- 今回のテーマ
  - シェーダーを使うプロジェクトのアートワーク作業の流れ
  - XSI, Photoshopから実機まで
- シェーダーコードの開発やポストエフェクトは別の話題

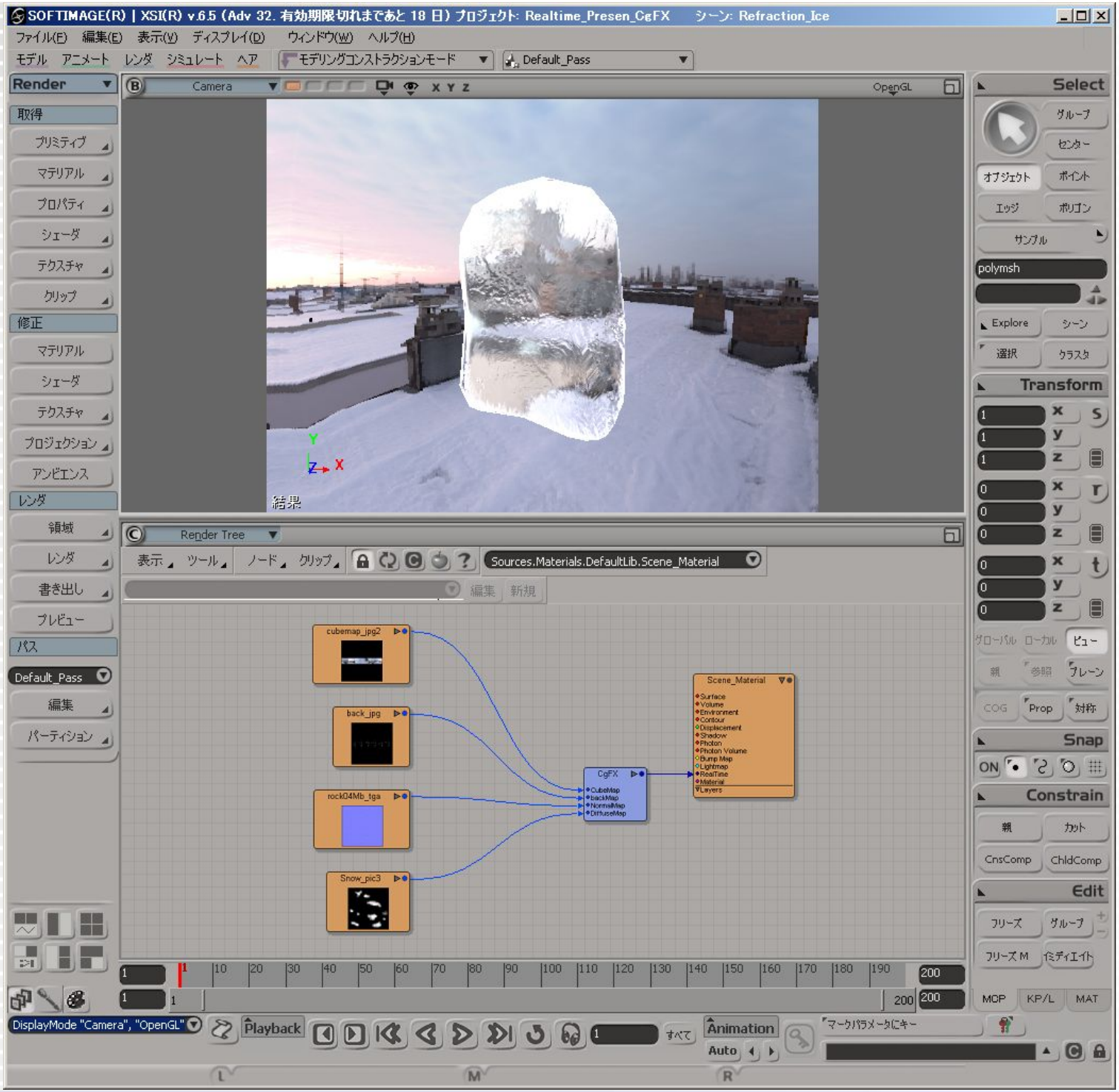
シェーダーパイプラインの種類

- 「シェーダーを使った開発」と言っても思い描いているものが違う…

- 議論のための言葉を定義してみる
  - リアルタイムシェーダーマテリアルスタイル
  - 選択スタイル
  - 外部スタイル
  - 標準シェーダースタイル

リアルタイムシェーダーマテリアルスタイル

- DCCツールのリアルタイムシェーダーマテリアルを使用する


SOFTIMAGE(R) | XSI(R) v.6.5 (Adv 32. 有効期限切れまであと 18 日) プロジェクト: Realtime_Presen_CgFX シーン: Refraction_Ice
ファイル(F) 編集(E) 表示(V) ディスプレイ(D) ウィンドウ(W) ヘルプ(H)
モデル アニメート レンダ シミュレート ヘア モデリングコンストラクションモード Default_Pass
Render
取得
プリミティブ
マテリアル
プロパティ
シェーダ
テクスチャ
クリップ
修正
マテリアル
シェーダ
テクスチャ
プロジェクション
アンビエンス
レンダ
領域
レンダ
書き出し
プレビュー
パス
Default_Pass
編集
パーティション
Camera
OpenGL
結果
Render Tree
表示 ツール ノード クリップ
Sources.Materials.DefaultLib.Scene_Material
編集 新規
cubemap_jpg2
back_jpg
rock04Mb_tga
Snow_pic3
CgFX
Scene_Material
Select
グループ
センター
オブジェクト
ポイント
エッジ
ポリゴン
サンプル
polymsh
Explore
シーン
選択
クラスタ
Transform
Snap
Constrain
親
カット
CnsComp
ChldComp
Edit
フリーズ
グループ
フリーズ M
イミディエイト
MCP
KP/L
MAT
DisplayMode "Camera", "OpenGL"
Playback
Animation
Auto

リアルタイムシェーダーマテリアルスタイル

- DCCツールのプレビューに表示

- エフェクトファイル
- ツール規定のセマンティック

- DCCツールベンダさんのイメージ？

リアルタイムシェーダーマテリアルスタイルのメリット

- 全てのシェーダー表現がDCCツール内で確認可能
- すでに完成したツールとフォーマットが公式に存在する
- デザイナーが様々なシェーダーを試せる

リアルタイムシェーダーマテリアルスタイルのデメリット

- DCCツール上のプレビューが重い
  - 地形などでは実際にプレビュー困難になるレベル
- 設定されたシェーダはプレビュー専用
  - さらにゲーム内用に相当するシェーダーを別途開発する必要がある
- 意外に互換性がない
  - ツール、シェーダー言語ごとに独自の環境、セマンティック
- ポストエフェクトは難しい

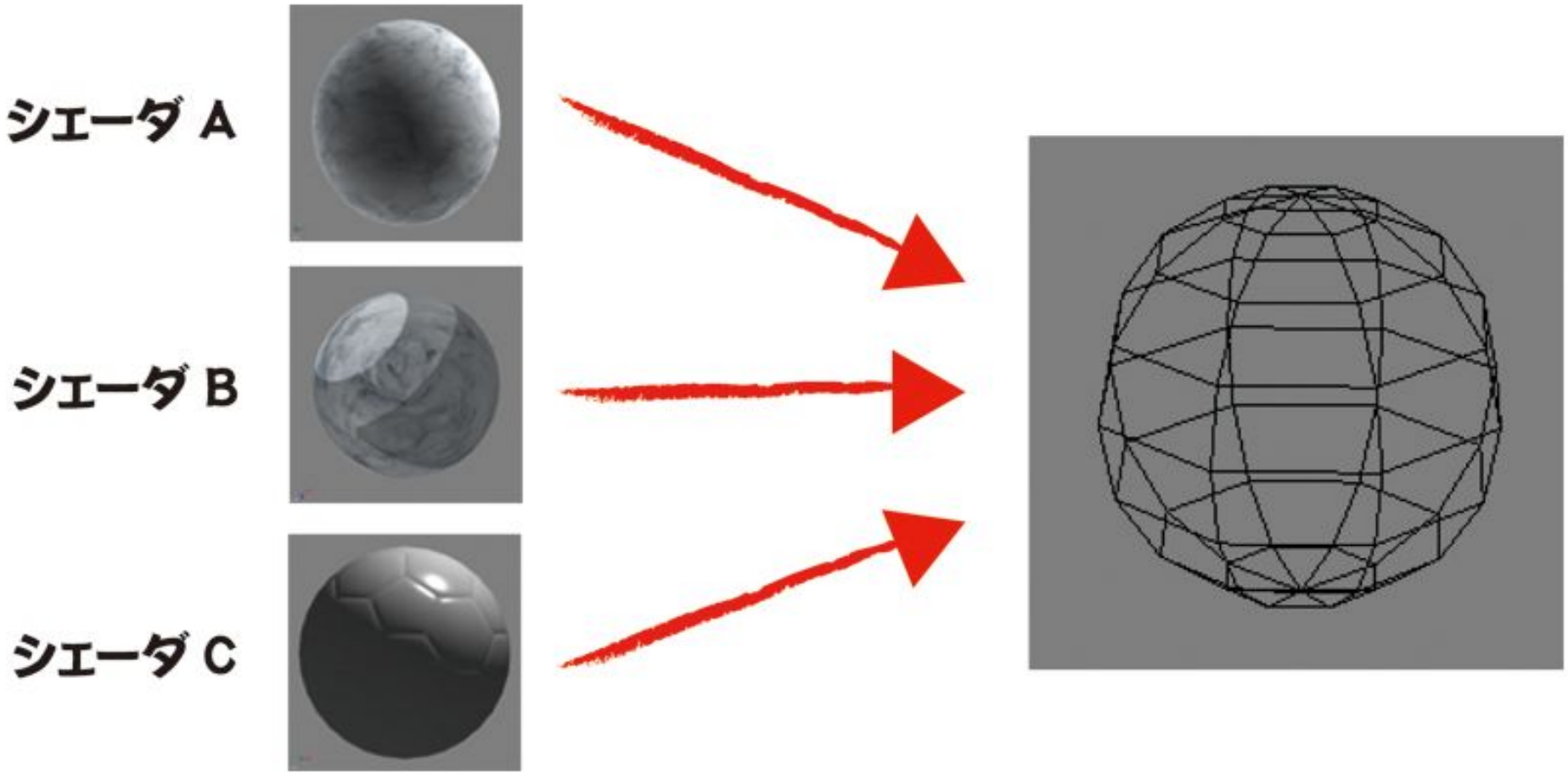
リアルタイムシェーダーマテリアルスタイルにおけるDCCツール

- デザイナーがシェーダーコード実験
- シェーダーパラメーター込みのプレビュー
  - 実機に近い表現を確認しながらモデリング
- シェーダーマテリアルからのエクスポーター
  - デザイナが自由にシェーダーをいじっていたら？
  - パラメーターはちゃんと出る？

選択スタイル

- プログラマーが予めシェーダー（群）を用意してリスト化しておく
- リストの中からデザイナーが選択する
  - DCCツール上ではマテリアル名か？
  - カスタムアトリビュートだと格好いい
  - Exceだと格好わるい
  - リアルタイムシェーダーマテリアルを選択することもできる

選択スタイル（図）



選択スタイルのメリット

- ワークフローがシンプル
- 間違いが少ない
  - 遅くなりすぎたり使えない機能を使ったりしない
- ツールとシェーダー言語によらない

選択スタイルのデメリット

- プレビューとパラメーターの調整に手間がかかる
  - 選択を後から変更すると？
- 表現の幅が狭まる可能性がある
- シェーダーの種類が膨大になる場合がある

選択スタイルにおけるDCCツール

- リストの中からデザイナーが選択する

- 必要なパラメーターは正しく作成できているか？
- プレビューにはリアルタイムシェーダーマテリアルもいい

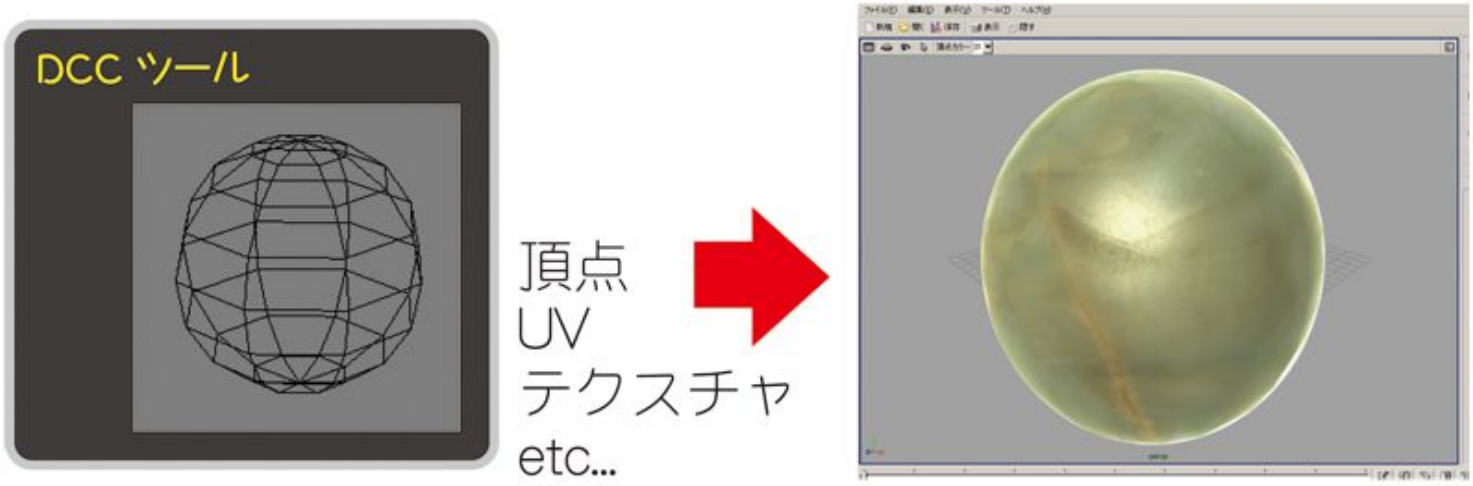
外部ツールスタイル

- DCCツールから離れ、専用エディターでシェーダーの割り当てとパラメーター設定を行なう

- 独立したシェーダーツールで設定
- ゲームエンジン内で設定

外部ツールスタイル

- ビューワー機能も持つ
- マテリアルだけでなく、物理やコリジョン、ゲーム要素に関わる設定も同時に行なえる



外部ツールスタイルのメリット

- ＤＣＣツールによらない
  - マルチツール対応
- 本番相当のプレビューが簡単

- ポストエフェクトに強い

外部ツールスタイルのメリット

- 独立ツールの場合
  - ツールと作成したデータの汎用化ができる

外部ツールスタイルのメリット

- ゲームエンジンの場合
  - プロジェクトに応じたワークフローを過不足なく設計できる
  - マテリアルを超えたシェーダ調整
  - ライト、シャドウ、グレア、被写界深度、その他ゲーム要素から影響を受けるものなど
  - マルチプラットフォーム

外部ツールスタイルのデメリット

- ＤＣＣツール上では確認できない
- ツール（エンジン）を作る（買う）のが大変

- 手戻りに弱い
  - モデルのマテリアル構成が変わったら？
  - 外部ツールでの編集が無駄にならない機構が必要

- そのエンジン、大丈夫？

外部ツールスタイルにおける DCCツール

- モデルとモーションを作る
- シンプルなエクスポーターがあれば十分

- 外部ツールへの追加情報
  - 独自アトリビュート
  - 上書きエクスポート

標準シェーダースタイル

- システムがシェーダー機能要素を決める
- 機能の組み合わせとパラメーターをデザイナーが決める

- プログラムが自動的にシェーダーを生成

- 標準でない表現は別途追加する

シェーダファイルイメージ

各パラメーターを ON/OFF

- [x] ライト１
- [ ] ライト２
- [ ] ライト３
- [x] シェーディング
- [x] スペキュラ
- [x] ディフューズマップ
- [x] スペキュラーマップ
- [ ] 環境マップテクスチャ
- [ ] 法線マップ
- [x] リムライティング
- ▼ ・・・等など

標準シェーダースタイルのメリット

- デザイナーはシェーダーの管理が不要
- ワークフローがシンプルで表現に集中できる
- ツールとシェーダー言語によらない
  - マルチプラットフォーム＆マルチツール
- 他のプロジェクトやプラットフォームににデータが流用できる
- 外注に出しやすい

標準シェーダースタイルのデメリット

- 新たな表現をしたい場合はプログラマが対応
- DCCツール内では確認できない表現がある
  - カスタマイズ性の確保が課題
- シェーダーの自動生成に時間がかかる場合がある
- ポストエフェクトは記述しにくい

標準シェーダースタイルにおけるDCCツール

- ＤＣＣツールにある機能を極力生かすことができる
- 従来と同じ、決められた範囲で作成した「普通のデータ」
- レンダリング可能
- マルチテクスチャ、法線マップ、スペキュラマップなどの作成
- プレビューにも対応してきた

ハイブリッドスタイル

- 組み合わせることによっていいところ取りができる
  - リアルタイムシェーダマテリアルスタイル＋選択スタイル
  - 標準シェーダスタイル＋外部ツールスタイル

DCCツール上のアトリビュート

- どこに設定するか？
  - 標準アトリビュート
  - リアルタイムシェーダアトリビュート
  - カスタムアトリビュート
  - 外部ツール

- スタイルごとのメリットデメリットに直結する

DCCツール上のアトリビュートの課題

- 一覧性
  - データデバッグ
- SDKの対応
- 基本パラメーターはもう大丈夫？
  - 接線／従法線編集
  - 任意段数の頂点カラーやテクスチャ座標
  - カスタムアトリビュートのエクスポート
  - 座標系／継承／ファンクションカーブなど

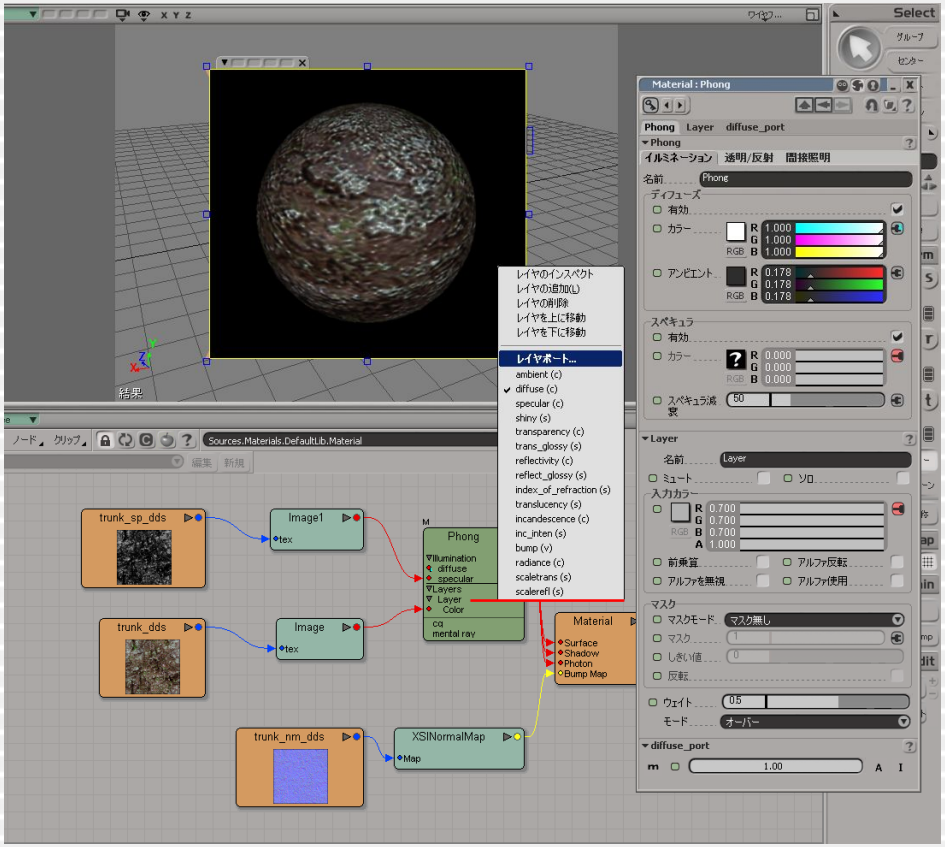
セガでの実例

- タイトルによって様々
- 標準シェーダー（NNグラフィックスシステム）
- ハイブリッド
  - 標準シェーダー＋外部ツール
  - リアルタイムシェーダマテリアル＋選択
  - カスタムアトリビュートによる選択スタイル

NNのシェーダーワークフローの方針

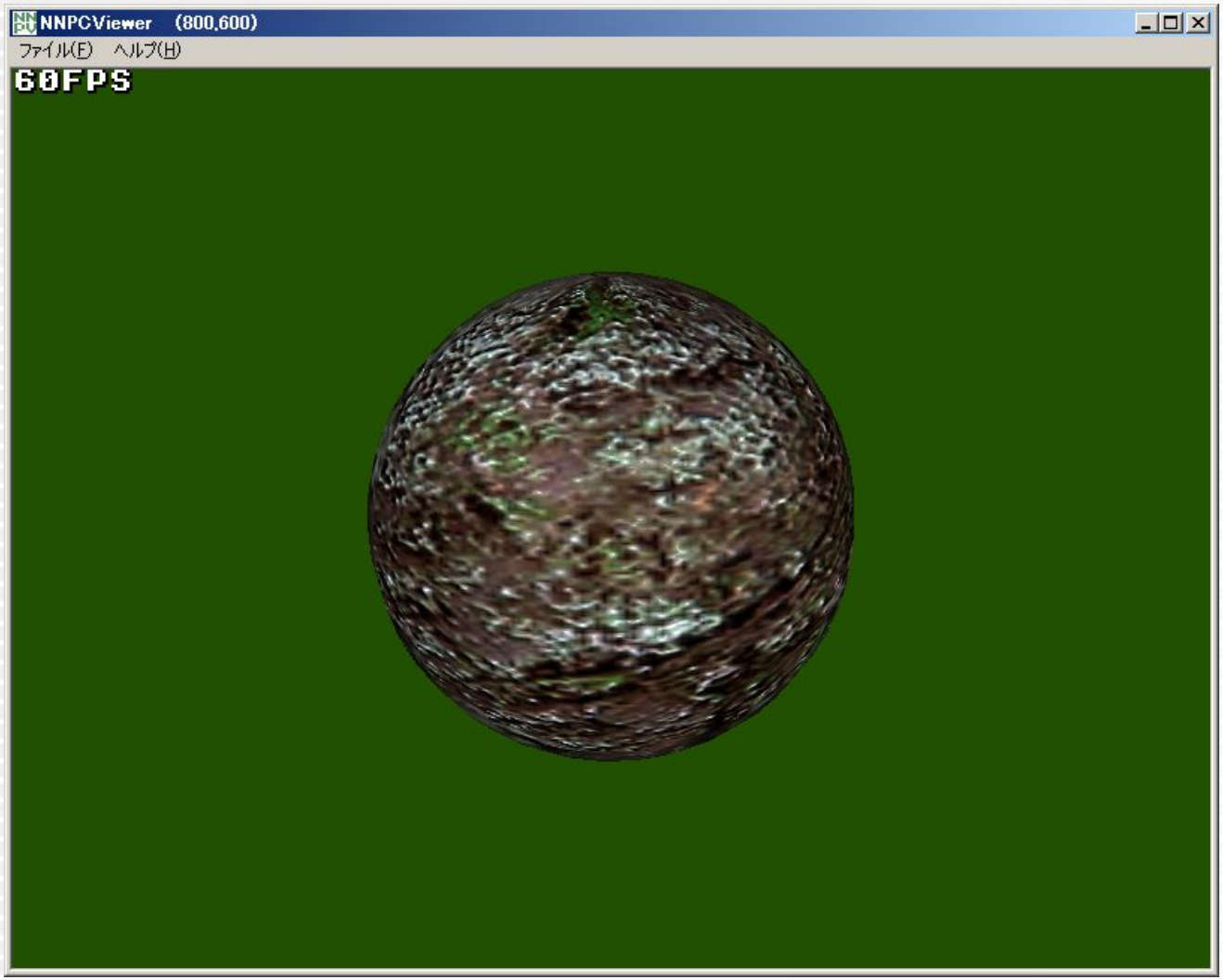
- 標準シェーダースタイルの例
- 可能な限り、DCCツール側のマテリアル設定を利用する。
- 可能な限り、DCCツールのマテリアル設定に基づいたレンダリング画像とイメージをあわせる

DCCツール(XSI)上でのマテリアルの設定

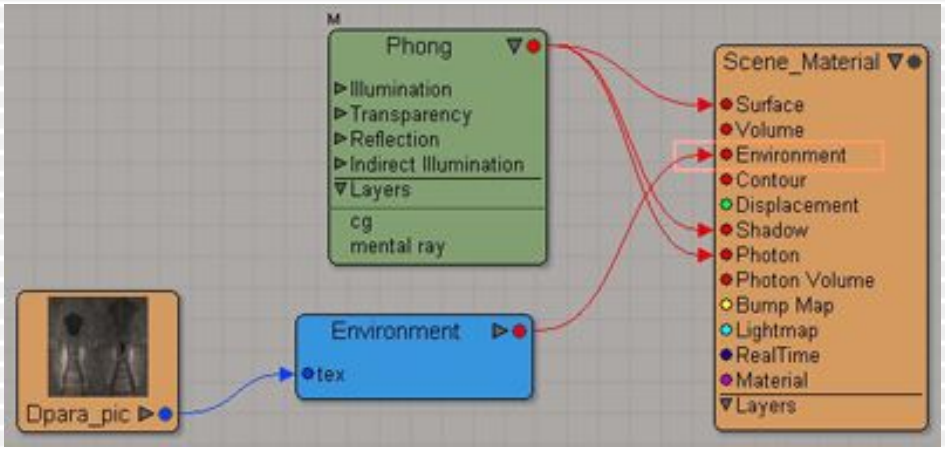


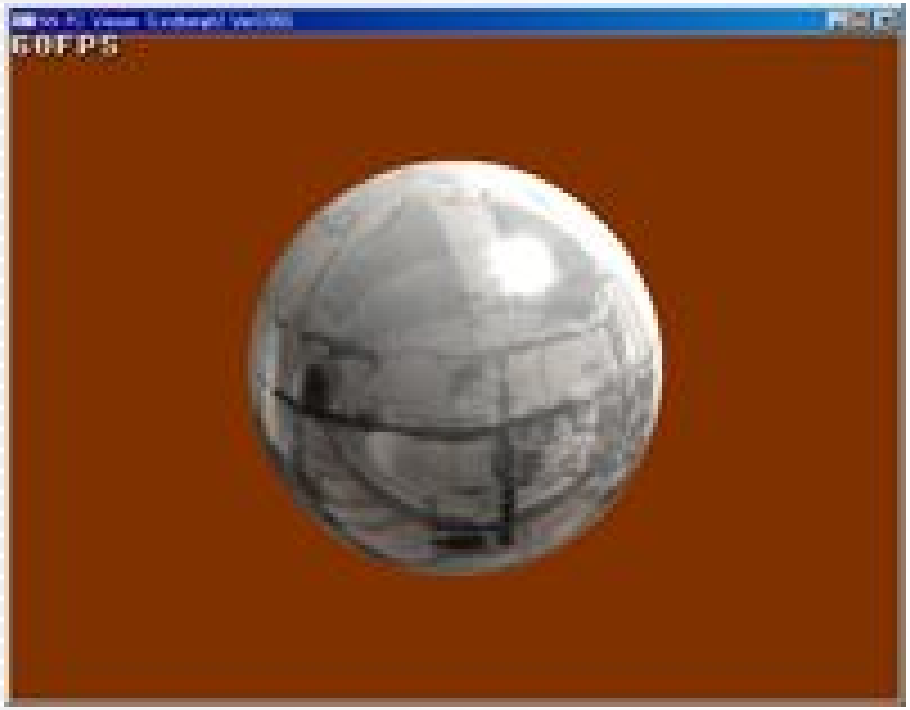
- ディフューズテクスチャ
- ディフューズカラー
- スペキュラーカラーテクスチャ
- 法線マップ

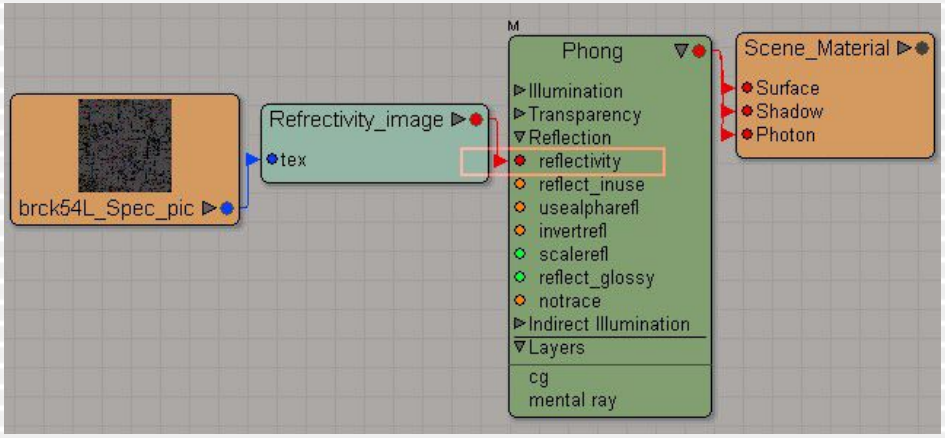
PCビューワーのシェーダープレビュー



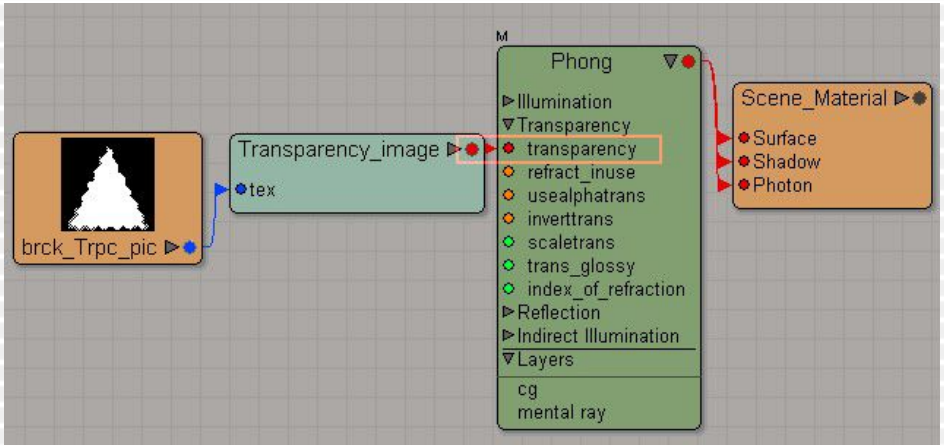
テクスチャ設定のその他　1







テクスチャ設定のその他　2



- 環境（Environment）マップ
- 環境マスクマップ
- 透明度（Transparency）マップ

等など

NN標準シェーダのカスタマイズ

- 実際のゲームにはカスタマイズは必須

- 標準シェーダーテンプレートのカスタマイズ
  - ユーザーデータ（マテリアル）
  - ユーザーマップ（テクスチャ）

  - ユーザープロファイル（シェーダー定数）
  - ユーザーユニフォーム（シェーダーパラメーター）
  - ユーザーフラグ（描画プログラム）