インハウスツール開発事例:ソニックの開発に使われているツール紹介

中川 展男
株式会社セガ　第二CS研究開発部 プログラマー

内容

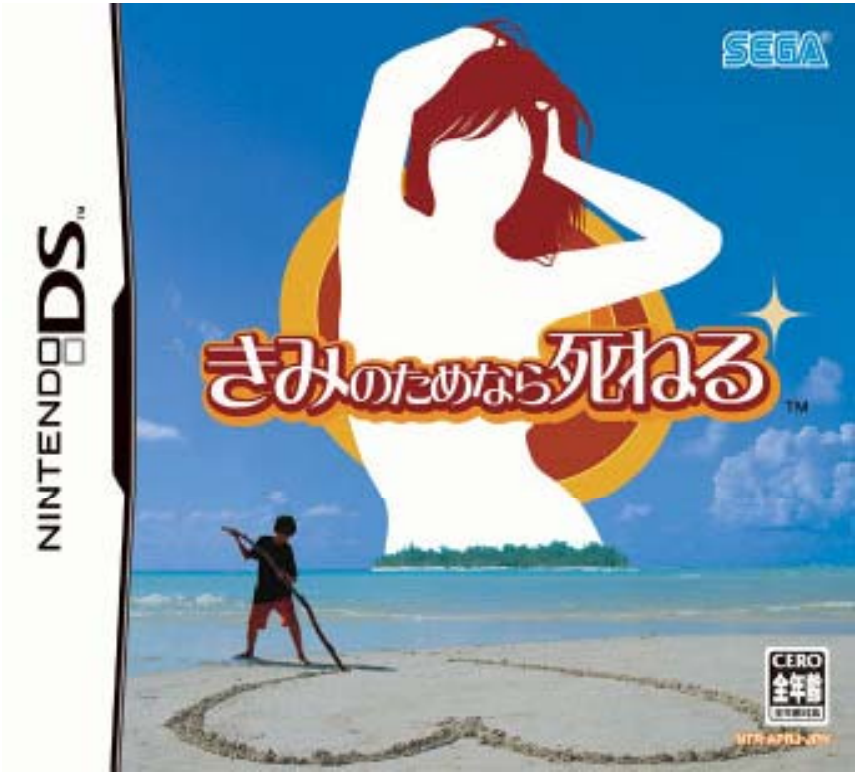
- プレビュー環境の紹介
- 学んだ事
- プレビュー環境の作り方(TIPS)
- 質疑応答

内容

- プレビュー環境の紹介
- 学んだ事
- プレビュー環境の作り方(TIPS)
- 質疑応答

はじめに

- ツール開発のうちプレビュー環境に絞ったお話をさせて頂きたいと思います。
  - ここでいうプレビュー環境とは、DCC ツールからエクスポートされたデータをゲームに組み込む前に確認できるようにしたものを指します。
- 過去のプロジェクトで作成したプレビュー環境の実例を示します。

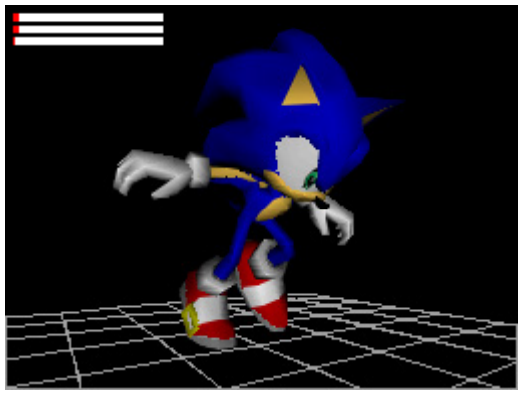


『きみのためなら死ねる』(NDSローンチ)での実例紹介

2004.11.16 北米　Feel the Magic: XY/XX
2004.12.02 日本　きみのためなら死ねる
2005.03.11 欧州 Project Rub

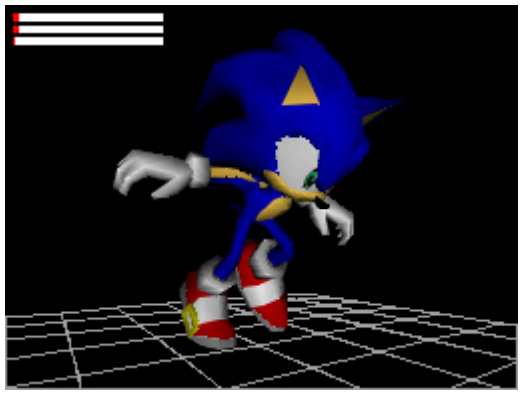
プレビュー環境紹介

『きみのためなら死ねる』プレビュー環境紹介



- DCC ツールからエクスポートしたデータを実機上で表示
- 実機のパッド操作でカメラを動かしながら確認できます。

『きみのためなら死ねる』プレビュー環境紹介



- 機能は最小限
  - モデル
  - モーション
    - テクスチャアニメーション
    - テクスチャパターンアニメーション
    - マテリアルアニメーション
    - ビジビリティアニメーション
  - CPU時間使用率
  - テクスチャ(イメージ、パレット)使用率

開発動機

開発動機

- 開発動機
  - NDSローンチ当時はプレビュー環境は未提供でした。
  - ハードウェアの機能の調査や表現力の確認を行う必要もありました。

開発方針

開発方針

- 開発方針
  - 必要最小限の機能を最短時間で作成
    - 開発機材を入手後2週間で開発
  - ゲーム本体を作る時間を多く取れるように

わかったこと

わかったこと

- 新ハードの表現力の確認を行うのにプレビュー環境を作成するのは効果的。
  - 3Dのソニックを作るのに十分なスペックがあることがわかった。
    - E3用ソニックのテクニカルデモ作成
      - その後『きみ死ね』開発スタート

わかったこと

- プロジェクトの初期に使い勝手の良いプレビュー環境があることはスムーズなプロジェクトスタートに役立つ

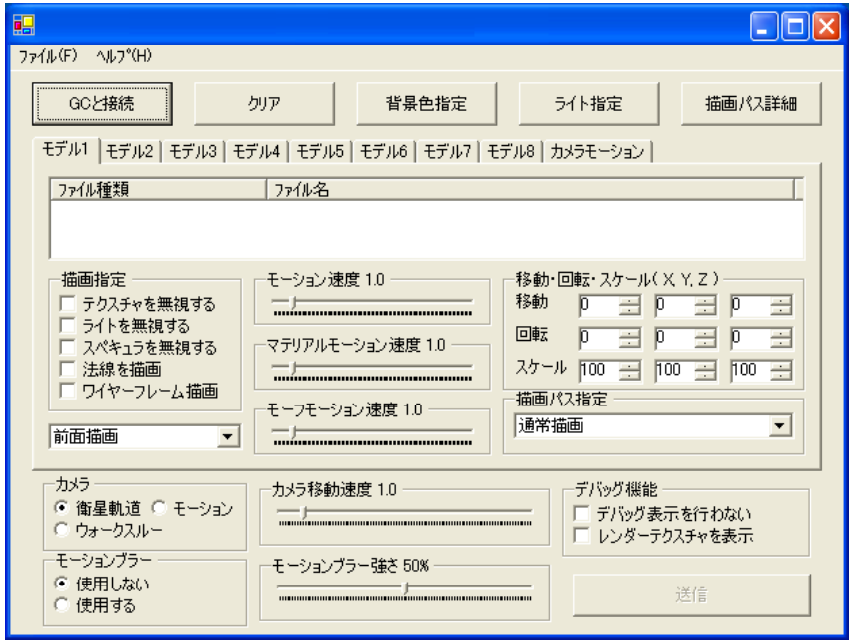
『ソニックと秘密のリング』(Wiiローンチ時期)での実例紹介

2007.02.20 北米 Sonic and the Secret Rings
2007.03.15 日本 ソニックと秘密のリング
2007.03.～ 欧州 Sonic and the Secret Rings

プレビュー環境紹介

『ソニックと秘密のリング』プレビュー環境紹介

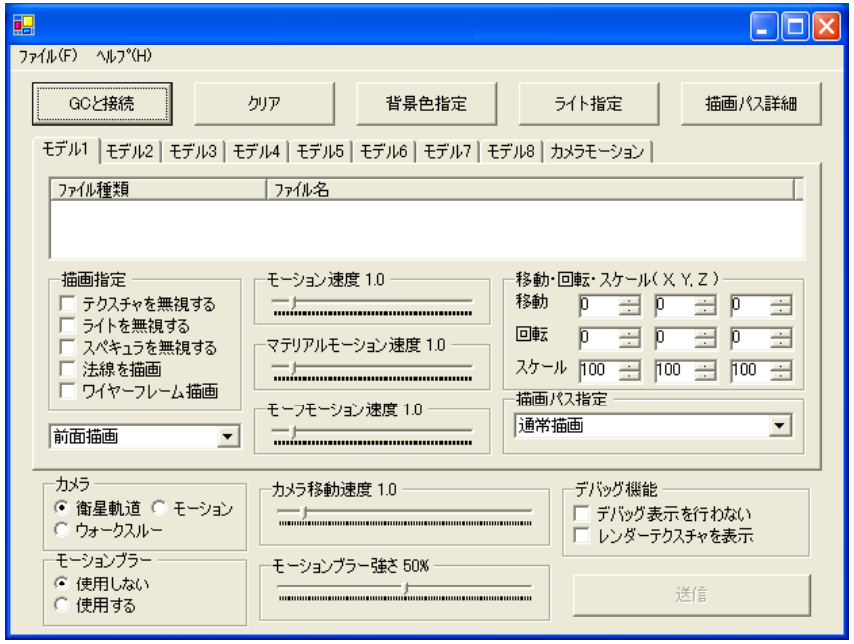
ホストPC側



Wii 実機側

『ソニックと秘密のリング』プレビュー環境紹介

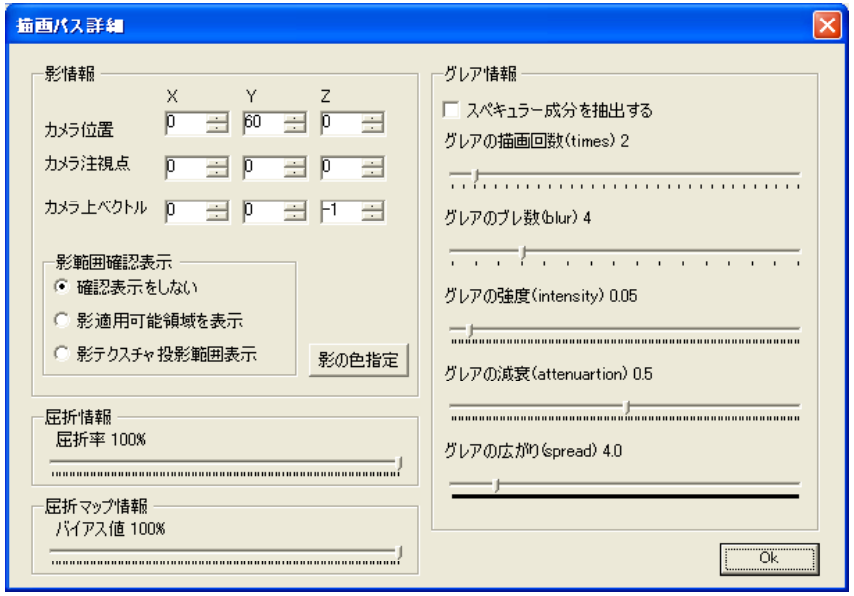
ホストPC側



- （ホストPC）→(ターゲットWii)と通信。
- PC側の GUI にファイルをドロップして「送信」で実機の画面が更新される。

『ソニックと秘密のリング』プレビュー環境紹介

ホストPC側



- マルチパス描画
  - 影描画のプロジェクション範囲確認
  - グレア詳細指定
  - 屈折周りの指定

『ソニックと秘密のリング』プレビュー環境紹介

Wii実機側

- 影描画用パス
- 屈折描画用パス
- モーションブラー用パス

『ソニックと秘密のリング』プレビュー環境紹介

Wii実機側

- 影描画用パス
- 屈折描画用パス
- モーションブラー用パス

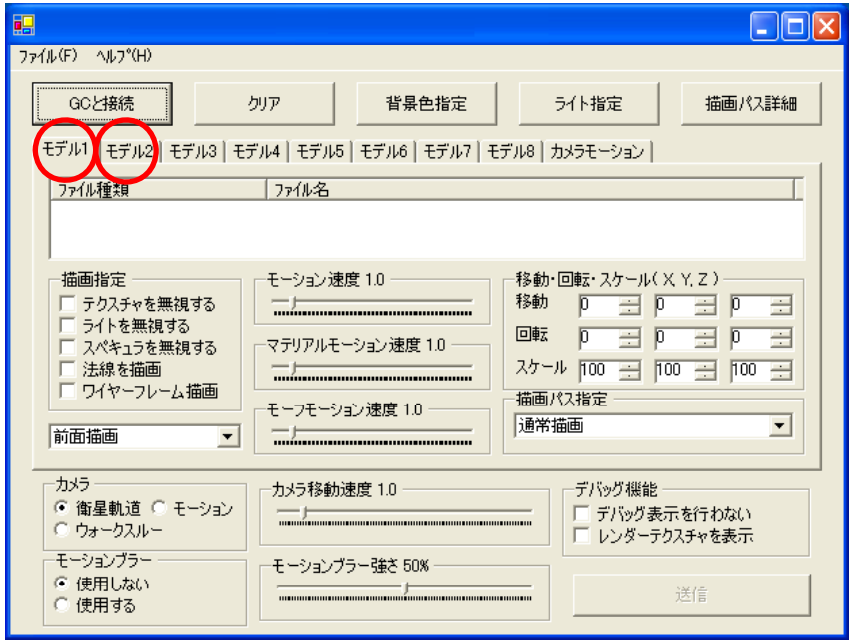
『ソニックと秘密のリング』プレビュー環境紹介

Wii実機側

- 影描画用パス
- 屈折描画用パス
- モーションブラー用パス

『ソニックと秘密のリング』プレビュー環境紹介

ホストPC側

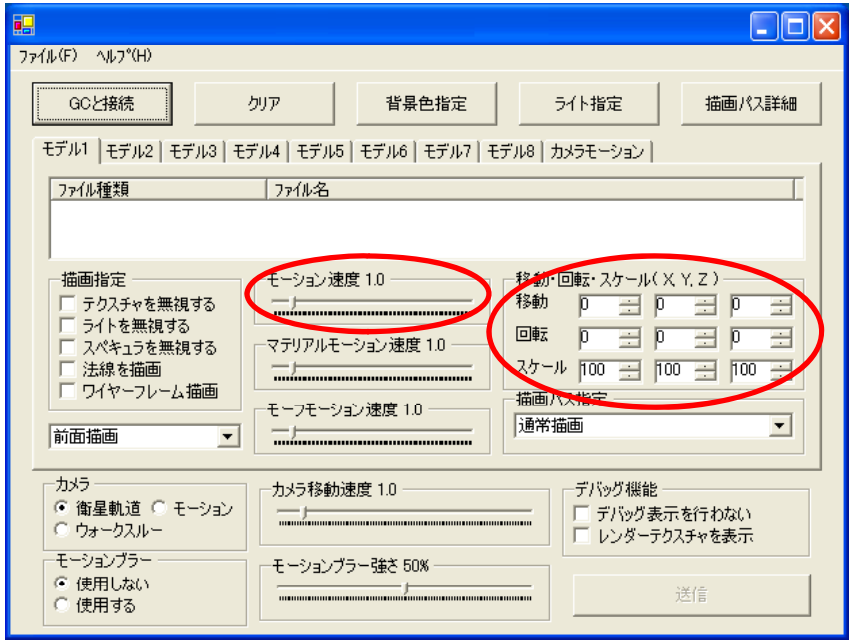


- 複数モデル表示
  - 地形
  - キャラクター

地形とキャラクターと一緒に見たいという要望。
タブでモデル切り替えするようなのは変…。

『ソニックと秘密のリング』プレビュー環境紹介

ホストPC側

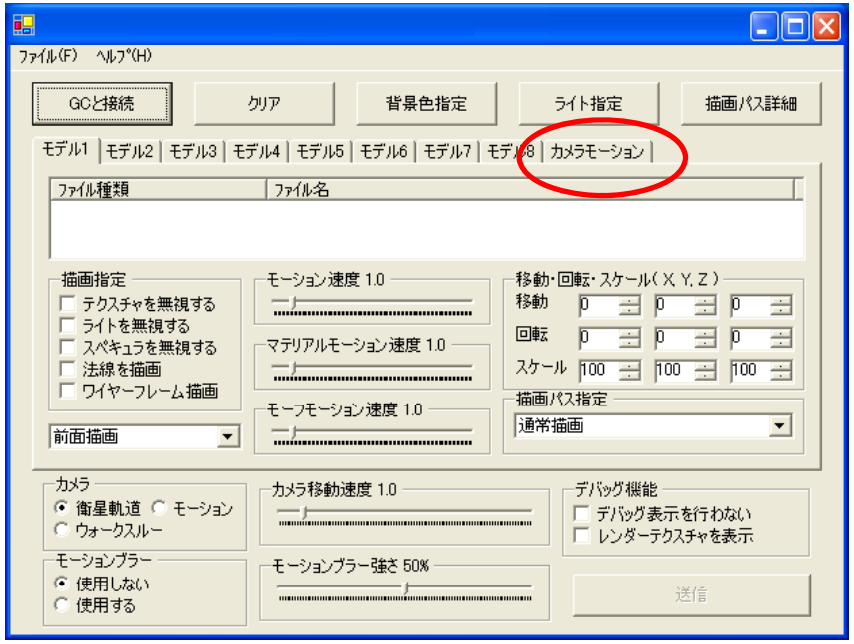


- 簡単な値の編集
  - モーション速度変更
  - TRS編集

モデルの基点位置の都合などで
地形に埋まる事がよくある。
最低限位置指定は必要。

『ソニックと秘密のリング』プレビュー環境紹介

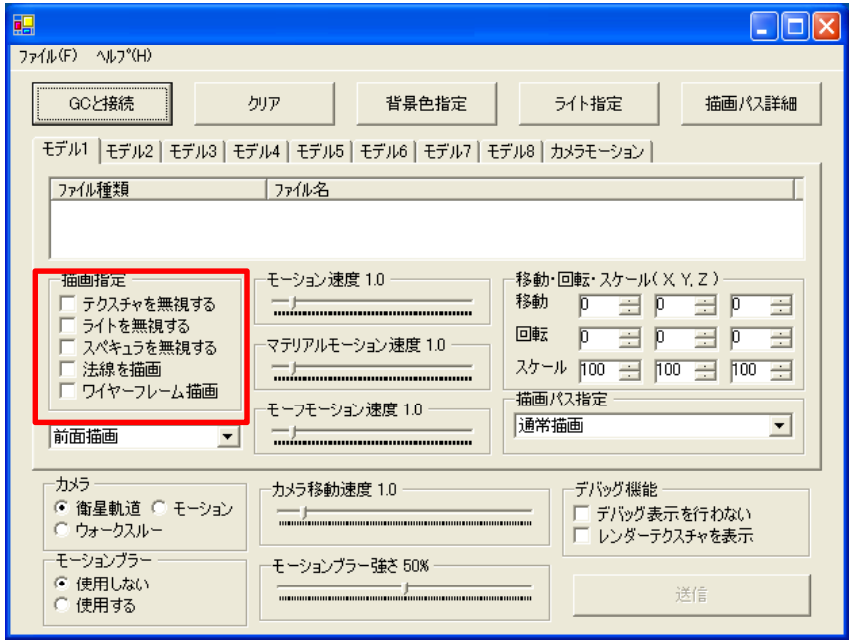
ホストPC側



- カメラモーション指定
  - モーションと同時に再生
  - 簡単なイベントシーン

『ソニックと秘密のリング』プレビュー環境紹介

ホストPC側



- 描画指定
  - 正しく表示されない時の調査
  - テクスチャ有無でのパフォーマンス確認
  - 法線向きとライティング具合の確認

開発動機

開発動機

- 開発動機
  - Wii での表現範囲の調査。
  - 描画機能の取捨選択など
    - 例えば影描画どうする？
      - テクスチャ投影 (セルフシャドウ無いけど)
      - シャドウマップ (エイリアシングの問題が)

開発方針

開発方針

- 開発方針
  - –デザイナーの表現テスト
  - –パフォーマンス確認
    - 仕様の確定を手助けできるように

わかったこと

わかったこと

- プレビュー環境とゲーム本体で見た目が異なる。
  - ビューワーを参考にせずにゲーム本体を作ったため、
    - サーフェースの解像度や投影方式の違いで影の見た目が変わっている
    - NDSの環境と比較すればマルチパス描画で自由度があがっているので注意が必要。

わかったこと

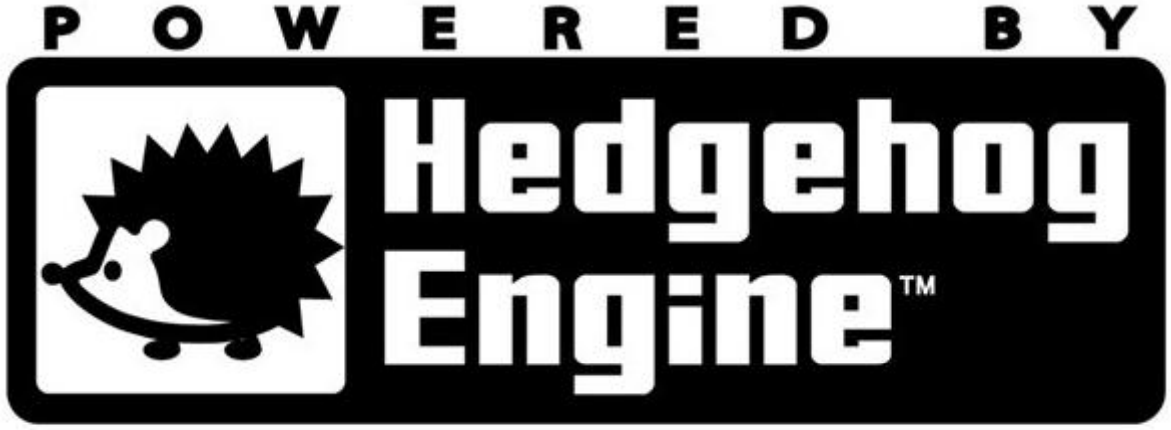
- ビューワーは遅くても許されるのですが実際のゲームでは描画速度やメモリ容量の都合などを考慮して影描画の解像度が決まる。
  - こういった変更が簡単にできるビューワーにすべきだった

『ソニックワールド アドベンチャー(PS3,360)』での実例紹介

2008.11.20～ 北米・欧州 Sonic Unleashed

2009.02.19　日本　　ソニック ワールドアドベンチャー

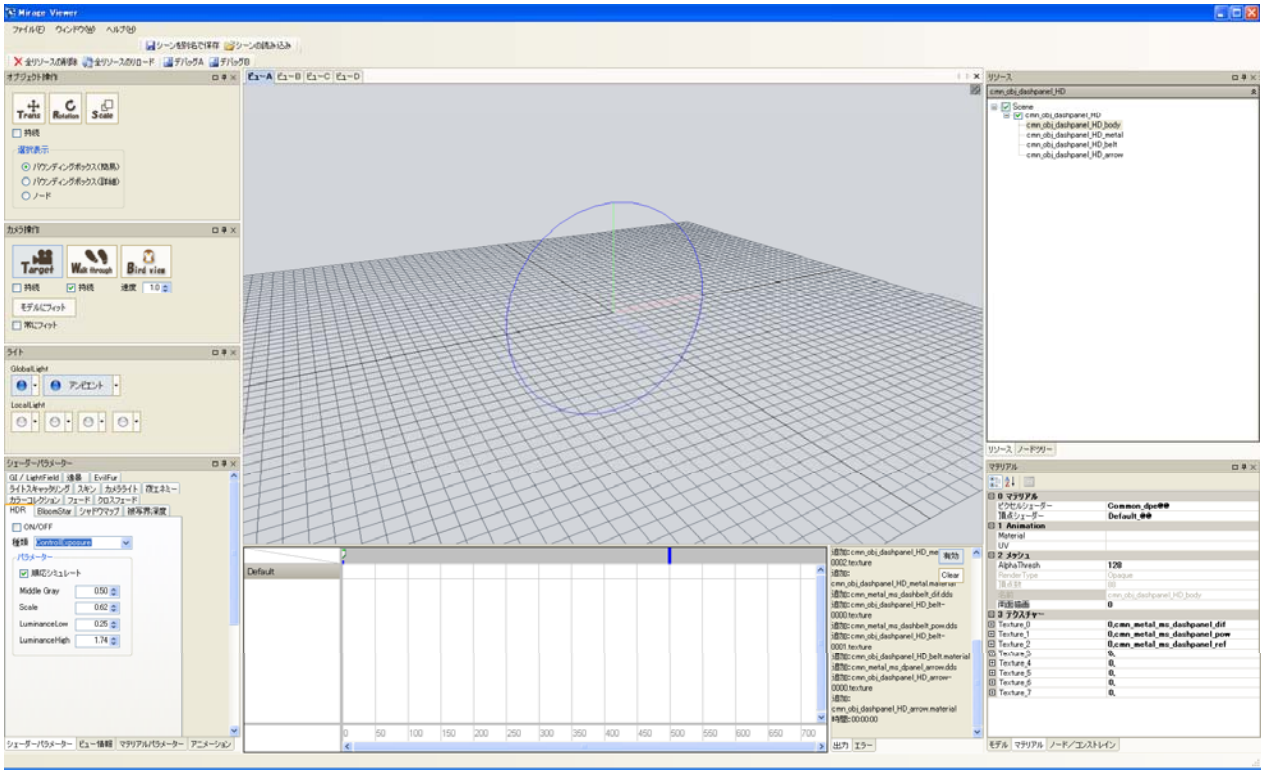
- PS3, 360, PC 対応のゲームエンジン Hedgehog Engine を作成。



プレビュー環境紹介

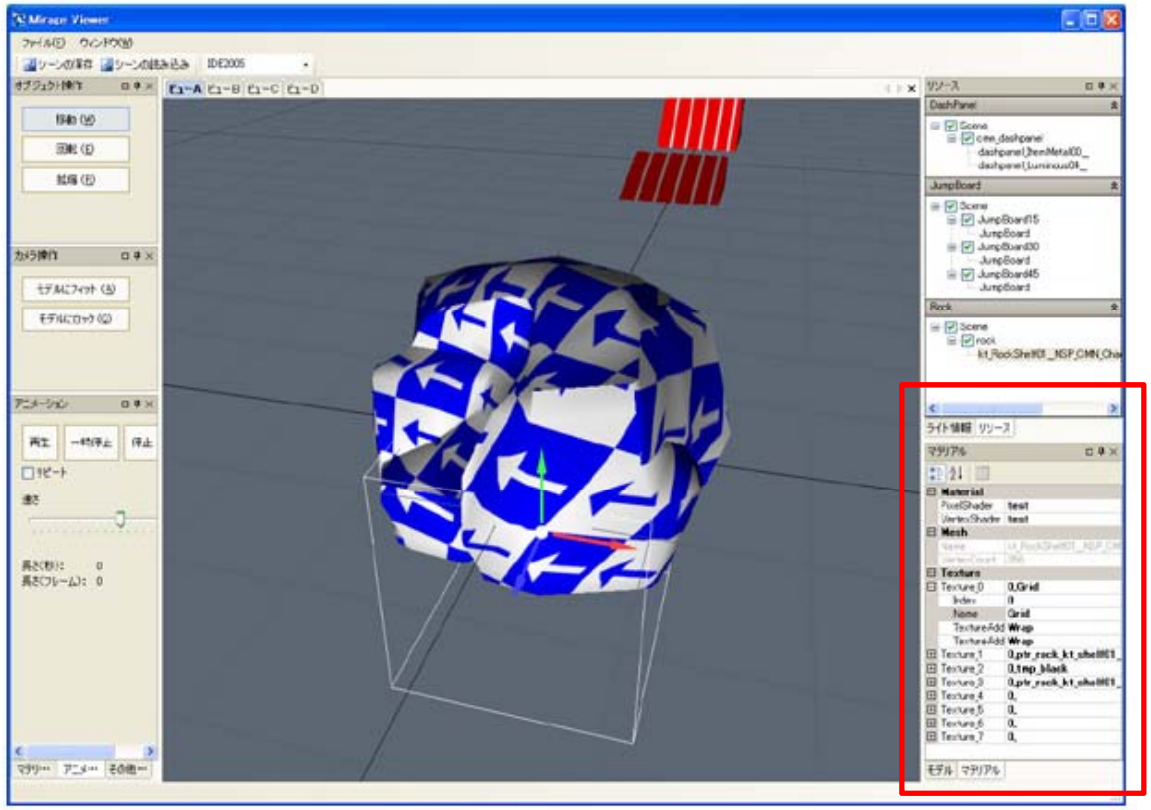
MirageViewer

MirageViewer 機能紹介



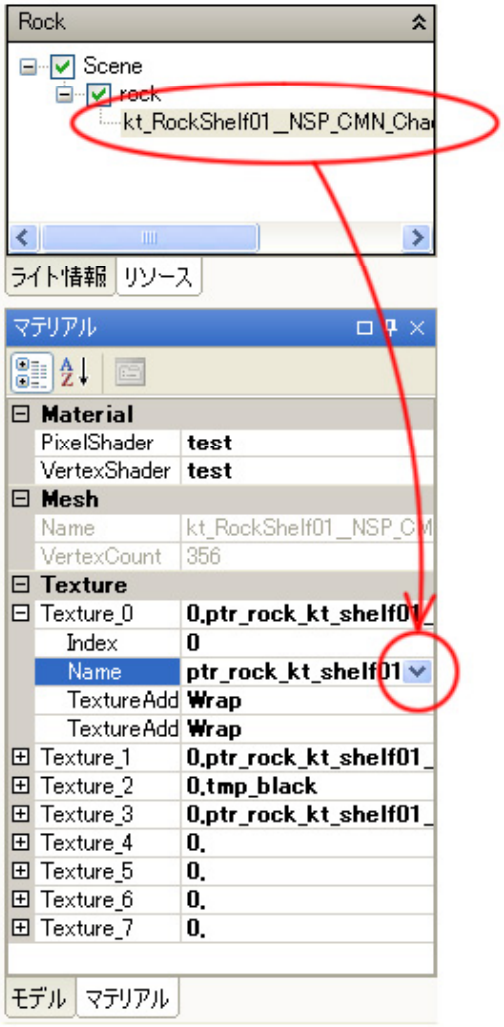
PC単体で完結するプレビュー環境。
PS3,360の実機不要ですし、ターゲットへのロードなども不要。

MirageViewer 機能紹介



詳細なマテリアルの確認

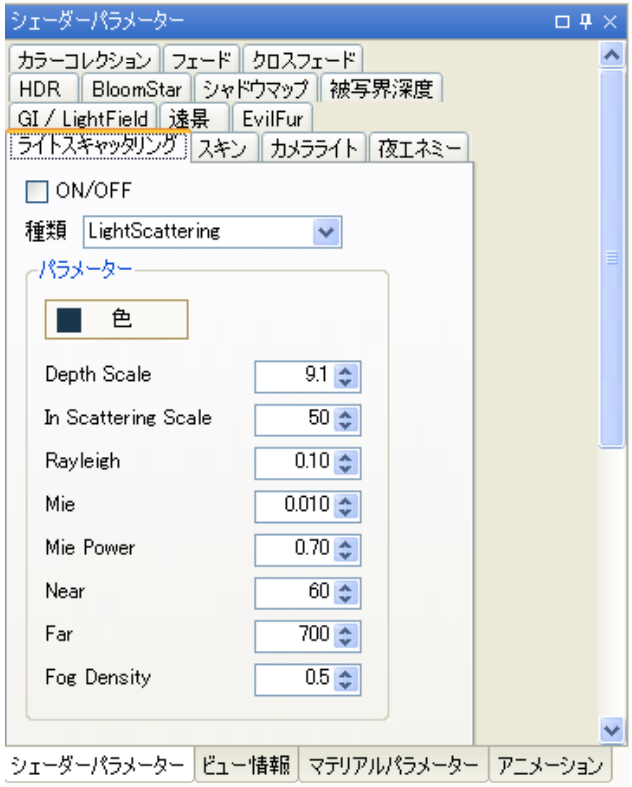
MirageViewer 機能紹介



テクスチャ、マテリアル、シェーダーを差し替えて確認できる。

マテリアルを編集して再度エクスポートする事なしに試行錯誤できるように

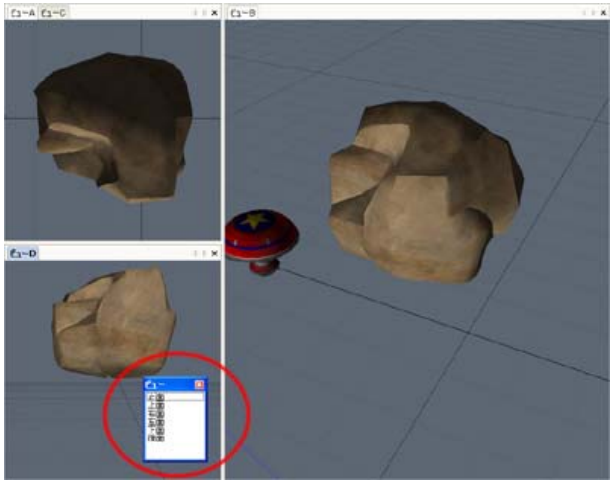
MirageViewer 機能紹介

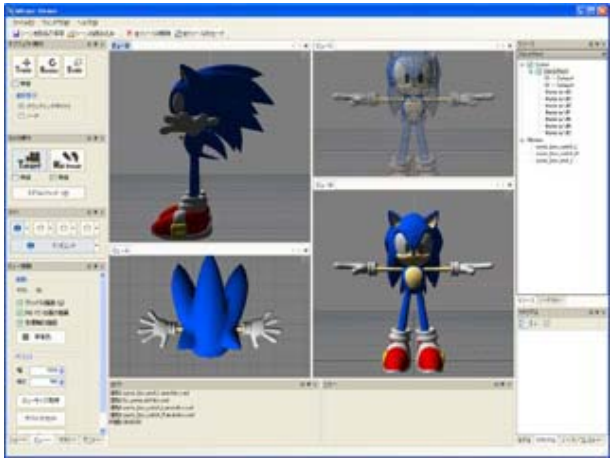


シェーダーに渡すパラメータの編集もできます。

見た目の調整を映像を確認しながら試行錯誤を行えるように

MirageViewer 機能紹介

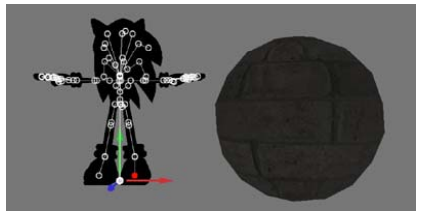


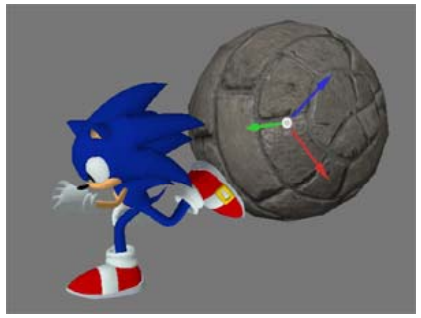


ビューを複数持てますので上から見たり横から見たりできます。

カメラ視点が異なる場合のアニメーションの違いを同時に確認できるように。

MirageViewer 機能紹介

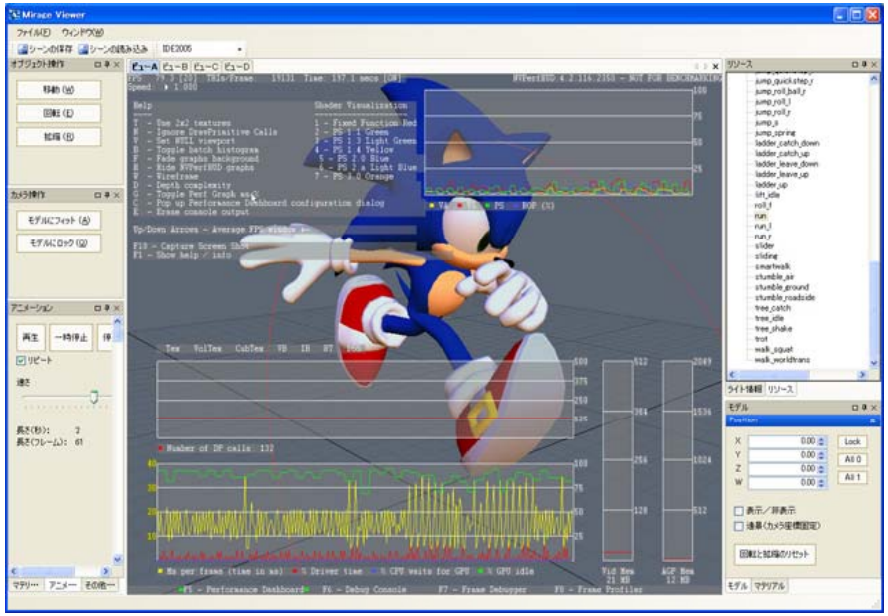




階層モデルの各ノードに別のモデルをコンストレインして確認できます。

この例ではソニックの足のノードに別のオブジェクトを割り当ててアニメーションさせて確認しています。

MirageViewer 機能紹介



詳細なパフォーマンスの確認が行えます。
(NVPerfHUD利用)

ただしPC上でのパフォーマンスになるのでPS3,360でのパフォーマンスとは別

描画順序やメッシュ描画時に適用されるテクスチャに問題がないかデバッグにも有用。

開発動機

開発動機

- 開発動機
  - インハウスのゲームエンジン作成
    - エンジン構成要素の一部として不可欠

開発方針

開発方針

- 開発方針
  - PC上でPS3、360と同じ絵素材の確認ができる。
    - これはプレビュー環境だけではなく、エンジン全体にわたって徹底されている。

わかったこと

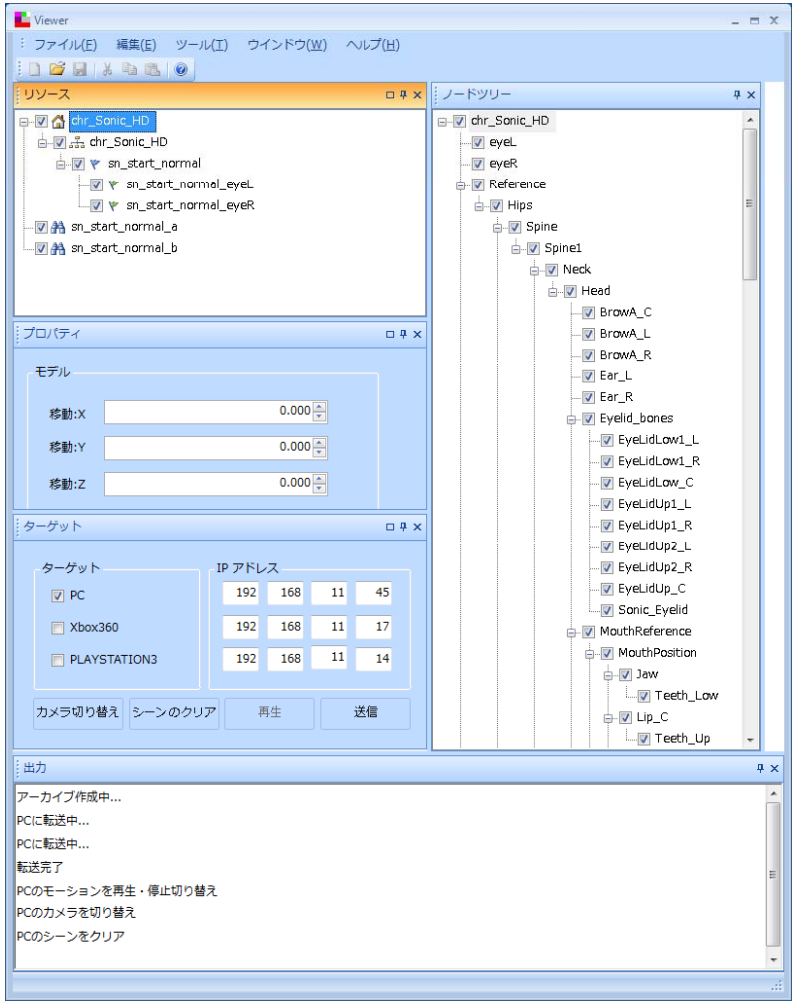
わかったこと

- 開発機材を必要としないビュー環境は、コスト面だけでなくSDK更新やファームのバージョン管理などの煩わしさもない。
- PCでのビューは実機転送不要で手軽。
- ただしPCでの確認は、最終的な色味やパフォーマンスを確認できない。

その後開発したプレビュー環境

その後開発したプレビュー環境紹介

ホストPC側



ターゲット実機側

その後開発したプレビュー環境紹介

- 大域照明(GI)の効果も確認でき、実際のゲーム画面と同等の状態でプレビューできます

開発動機

開発動機

- MirageViewer は、手軽に確認するには良い。しかし、Hedgehog Engine の特徴である高度なライト表現の確認ができなかった。
  - 実機でビューイングできる環境も追加し併用していきたい。
- 実機を使用することで最終的な描画結果を確認したい

わかったこと

わかったこと

- 開発実機でのビューは転送コストや開発実機用SDKの管理コストなどが大きくなる。
  - ちょっと確認したいだけの場合には大げさ
- 単機能のビューワーと最終確認用ビューワーどちらも必要。(メリットもデメリットもある)

まとめ

- これまでのプロジェクトでのプレビュー環境のご紹介を致しました。
- 機能の違いだけでなく実装方法も少しずつ異なっていました。

内容

- プレビュー環境の紹介
- 学んだ事
- プレビュー環境の作り方
- 質疑応答

はじめに

- 『失敗は成功の母』
  - 過去に経験した失敗例を示し、原因と解決方法をご紹介したいと思います。

はじめに

- プレビュー環境とゲーム本体で同じように描画できますか
- データ量の見積もりは担当者ごとに意識が異なる
- 見ただけで使い方がわかるツールとは

プレビュー環境とゲーム本体で同じように描画できますか

ビューワーとゲーム本体で同じように描画できますか

- 失敗例
  - –プレビュー環境とゲーム本体とで見た目の違いや利用できる機能に違いが出てきてしまった。
    - ライティング
    - シャドウ
    - 大域照明
    - …

ビューワーとゲーム本体で同じように描画できますか

- 原因
  - 描画フレームワークの汎用化、共通化が不十分。

ビューワーとゲーム本体で同じように描画できますか

- 対策方法
  - ライト、カメラ、ポストエフェクト、シェーダーなども含めて完全にゲーム側と同じ設定をプレビュー環境で用意。
  - マルチプラットフォームの場合、PCと差異を作らなければPCのみのビューも可能ですが、最終確認は各実機が理想。

データ量の見積もりは担当者ごとに意識が異なる

データ量の見積もりは担当者ごとに意識が異なる

- 失敗例
  - 想定していたよりも大量のデータを扱うことになり以下のような問題がおこる。
    - ロード時間が非常に長い。
    - ドラッグ&ドロップの回数が非常に多くて煩雑。

データ量の見積もりは担当者ごとに意識が異なる

- 原因
  - 実際のツール使用シーンでのデータ数量が正確に把握できていない。
  - 小さなテストデータでしか確認していないのでロード時間などの問題が把握できていない

データ量の見積もりは担当者ごとに意識が異なる

- 対策
  - 実データでの確認が重要。
  - 大量のデータを扱う際にツール側で最適な自動化の手段を考えて随時対応。

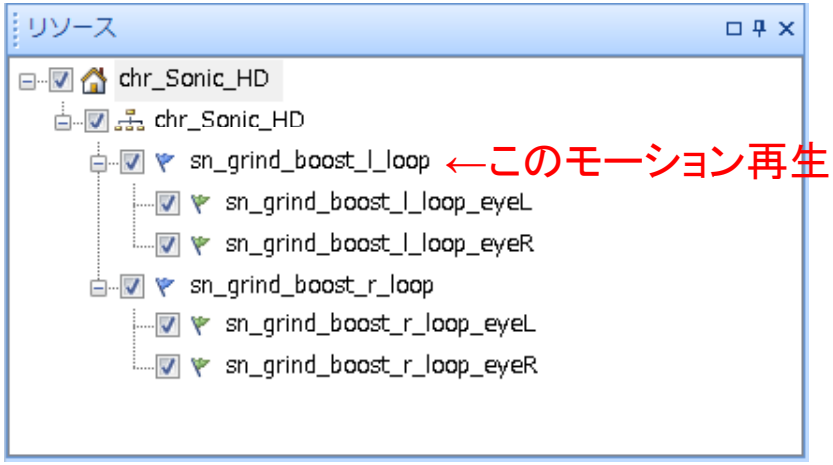
データ量の見積もりは担当者ごとに意識が異なる

- 大量のデータ処理の自動化の例

大量のデータ処理の自動化の例

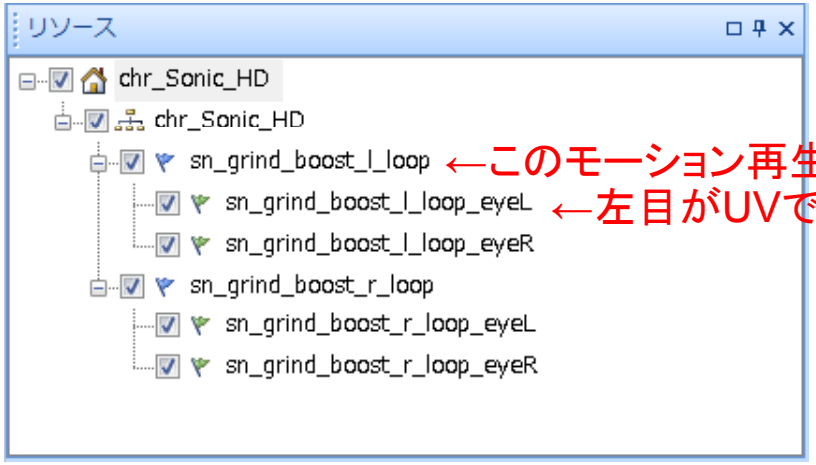
- アニメーション担当デザイナーからの要望
  - ソニックのアニメーション100種類以上を快適に確認できるように。
  - 通常のアニメーションに加え UV アニメーションとの連動機能も面倒なくできるように。

大量のデータ処理の自動化の例



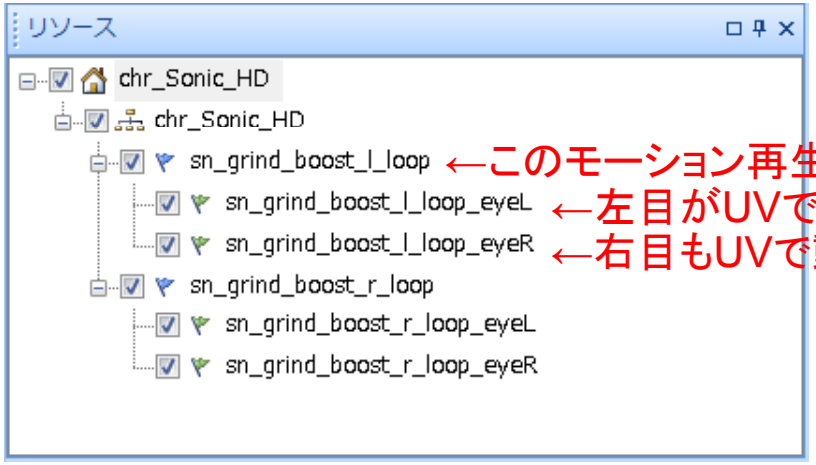
- ノードアニメーションの子供に、UVアニメーションをぶら下げることで連動して再生できるようにしました。

大量のデータ処理の自動化の例



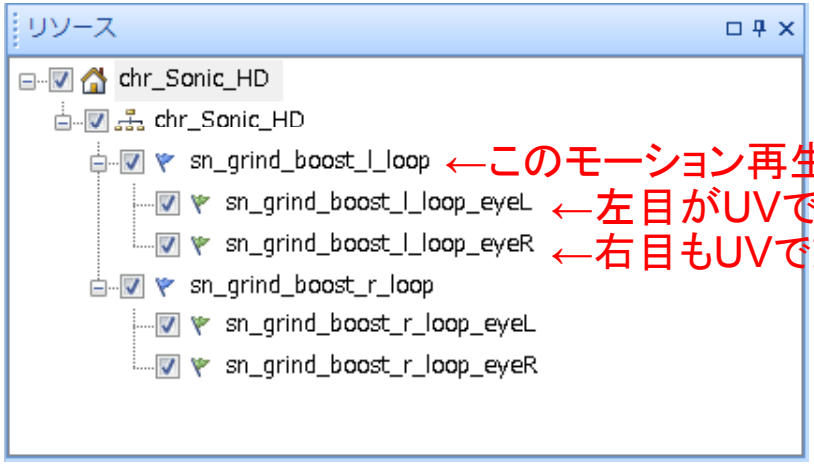
- ノードアニメーションの子供に、UVアニメーションをぶら下げることで連動して再生できるようにしました。

大量のデータ処理の自動化の例



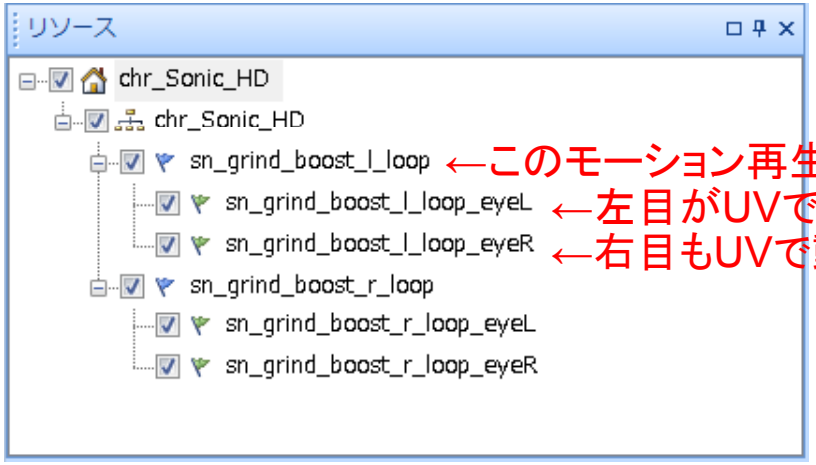
- ノードアニメーションの子供に、UVアニメーションをぶら下げることで連動して再生できるようにしました。

大量のデータ処理の自動化の例



- ノードアニメーションの子供に、UVアニメーションをぶら下げることで連動して再生できるようにしました。
  - ソニックの全アニメーションに目のUVアニメーションを追加できました。

大量のデータ処理の自動化の例



- ノードアニメーションの子供に、UVアニメーションをぶら下げることで連動して再生できるようにしました。
  - ソニックの全アニメーションに目のUVアニメーションを追加できました。
  - 命名規則に従っていればファイル丸ごとドロップで全部勝手に適切なアニメーションにぶら下がります。

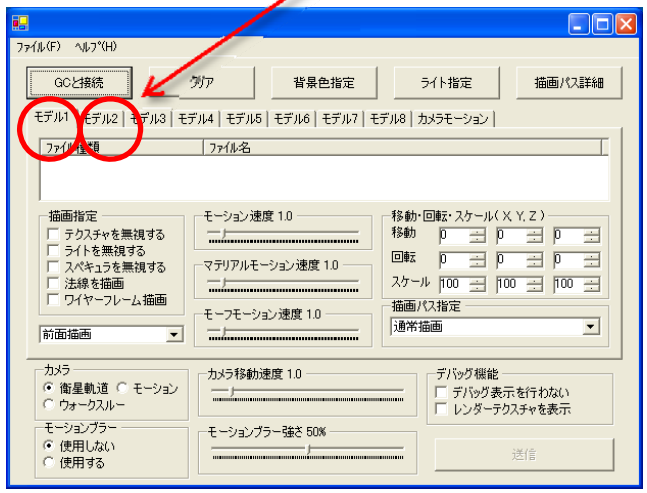
見ただけで使い方がわかるツールとは

見ただけで使い方がわかるツールとは

- 失敗例
  - –ツールの機能が多機能でどのように操作すればよいのかさっぱり分からない。
  - –ドキュメントは提供されているが膨大で学習コストが高いツールになってしまっている。

見ただけで使い方がわかるツールとは

- 原因
  - 操作方法が独自になってしまっている。
    - なぜ複数モデルをタブで扱うのか?



見ただけで使い方がわかるツールとは

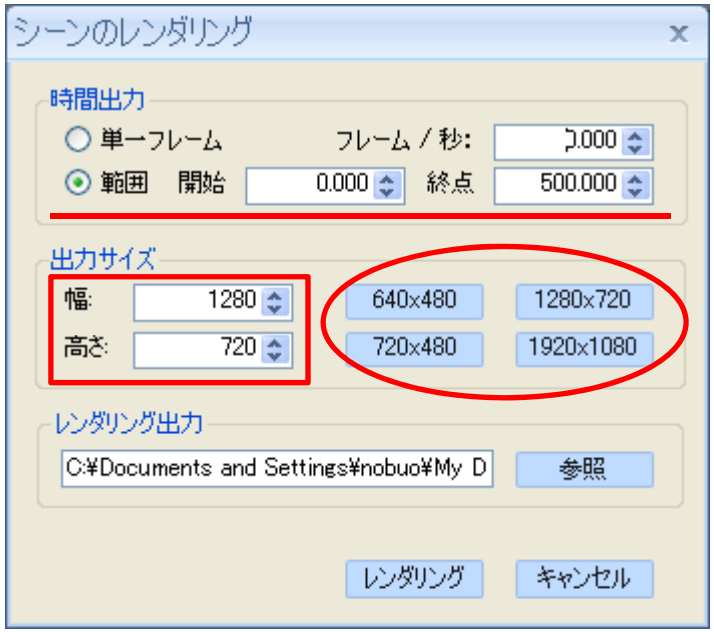
- 原因
  - 段階的に学べるチュートリアル形式のドキュメントの提供が行われていない。

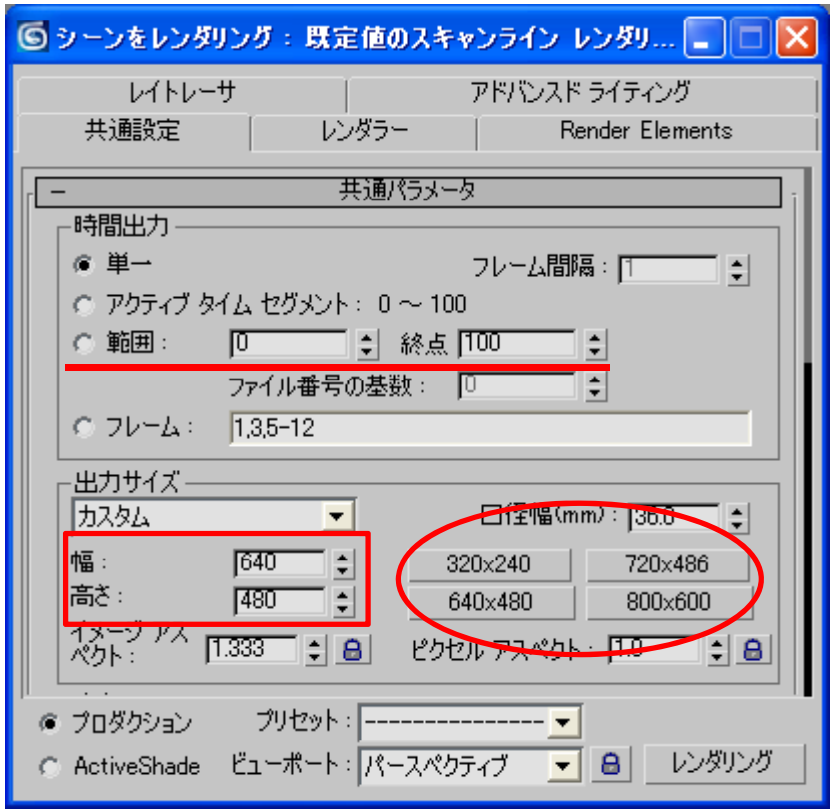
見ただけで使い方がわかるツールとは

- 対策
  - 既存のツールと操作方法を似せることで、既に学習済みの知識で使い始められるようなデザインに。

見ただけで使い方がわかるツールとは

- 既存のツールと操作方法を似せる例





見ただけで使い方がわかるツールとは

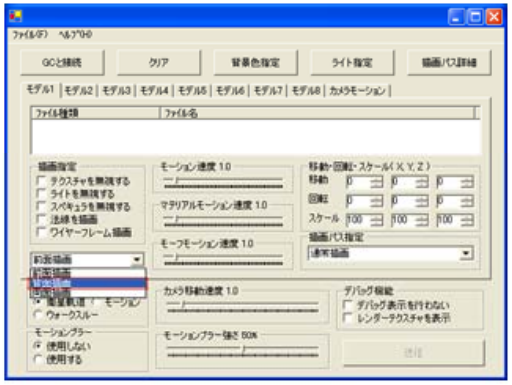
- 対策
  - 段階的に学習できるようにしましょう。
    - 『ソニックと秘密のリング』の Wiki より

チュートリアル

- チュートリアル:投影影の確認
- チュートリアル:投影影の調整
- チュートリアル:屈折
- チュートリアル:グレア
- チュートリアル:グレア(輝度抽出)
- チュートリアル:モーションブラー
- チュートリアル:屈折マップ
- チュートリアル:カメラモーション
- チュートリアル:背面描画

チュートリアル

・描画設定に「*背面描画*」という項目が追加になりました。今回はこちらを使った表現を試してみることにします。



まとめ

- 過去に経験した問題に関して原因と対策方法を考えました。
- プロジェクトが終了時に、皆で反省会を行い失敗例を皆でシェアすることで改善していけました。

内容

- プレビュー環境の紹介
- 学んだ事
- プレビュー環境の作り方(TIPS)
- 質疑応答

はじめに

- プレビュー環境を作る上で役立つ情報を記しておきます。

はじめに

- トピック紹介
  - DCCツール上でのプレビュー環境/独立したプレビュー環境/ゲーム本体でプレビューする環境
  - ネイティブのゲームエンジンと C# との連携
  - Undo 可能な設計
  - アイコン作成
  - ツールの配布に関して
  - UAC制御に関する問題

- DCCツール上でのプレビュー環境
- 独立したプレビュー環境
- ゲーム本体でプレビューする環境

DCCツール上、独立環境,ゲーム上でのプレビュー

- よく見かける3種類のプレビュー環境

プレビュー環境実現手段	ワークフローの上流、下流？
DCC ツール上でプレビュー	上流(エクスポート前)
独立したプレビュー環境	中流(エクスポート直後)
ゲーム本体プログラムでプレビューする環境	下流(ゲーム組み込み段階) プレビューとは呼べない？

・DCCツール上でのプレビュー環境

DCC ツール上で動かす

◎メリット	×デメリット
・シームレスなプレビュー ・ワークフローの最上流 (作業戻りが少ない)	・DCCツールだけではどうしても設定できない項目がある。 ・複数のDCCツールを混在して使用している際など同じ機能を提供するのが困難。

- 向いているのは
  - 顔のモデリングなどマテリアルのみの確認でいい場合

- 独立したプレビュー環境

独立したプレビュー環境

◎メリット	×デメリット
・オーバーヘッドが少ない。 ・ワークフローの中流	・ゲーム本体と見た目の違いや機能の違いが生まれ得るでしょう。

- 向いているのは
  - エクスポートした(ランタイム用に圧縮された)モーション確認など。手軽に確認したい場合。

- ゲーム本体でプレビューする環境

ゲーム本体でプレビューする環境

◎メリット	×デメリット
・見た目の違いは起こり得ない。	・オーバーヘッドが大きくなりがちでしょう。 ・ワークフローの最下流

- 向いているのは
  - 最終確認、背景とのなじみ具合や実際に操作した時に近い状態での確認

ネイティブのゲームエンジンと C# との連携

ネイティブのゲームエンジンと C# との連携

- C++/CLI でネイティブコードをラップ
  - 煩雑な作業
    - SWIG(=Simplified Wrapper and Interface Generator)で自動化?

ネイティブのゲームエンジンと C# との連携

- SWIGで自動化?
  - C# の場合例外なども自動生成
  - (私の場合は)煩雑さが減らない
  - 機械的な変換なので読みやすくない。
  - 開発初期段階から SWIGで C++から C# 用のコードをすべて生成する方針で開始しておけば問題なさそう。

ネイティブのゲームエンジンと C# との連携

- ライブラリの多くはネイティブのみ提供
  - ラップするには利用するだけよりも深い理解が必要。
  - ツールとゲームとで必要なデータ構造が異なる

ツール	ゲーム
最適化前のメッシュ情報	高速に描画できるメッシュ情報

- ツール用とゲーム用の両方の視点を持った設計になっていると嬉しい。
- .lib(static link) と .dll(dynamic link) の両方が提供されていると嬉しい。
  >ミドルウェア会社さん

Undo 可能な設計

Undo 可能な設計

- 使い勝手に大きく影響。
  - ただし実装は大変。(Undo stack)
  - あらゆる機能を Undo 可能なデータ構造にしておきたい。
    - 後からの機能追加では対応できない

Undo 可能な設計

- 実装方法
  - –親子付きの Stack 構造
  - –オープンソースの実装なら Qt(キュート)の QUndoCommand あたりが参考になる。

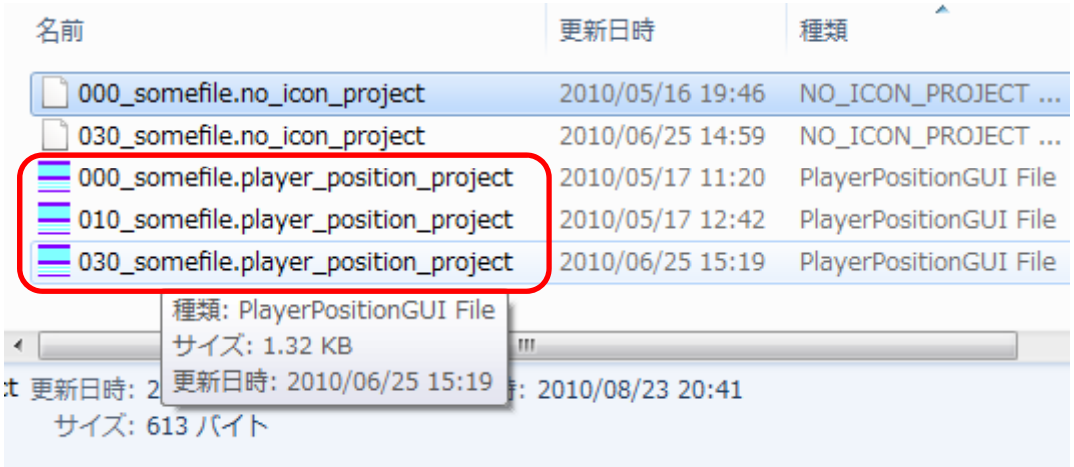
    http://doc.trolltech.com/4.7-snapshot/qundocommand.html

アイコン作成

アイコン作成

- ツールにアイコンを設定しましょう。
  - 視認性、拡張子との関連付けなどそれなりに意味がある。

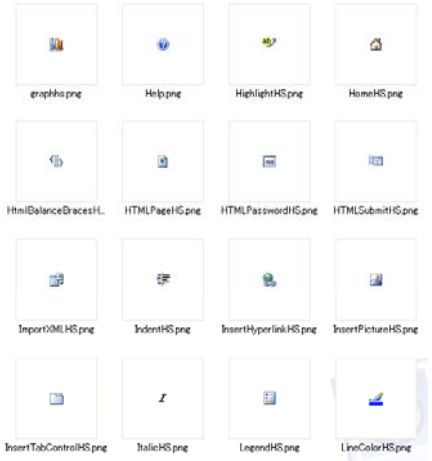
あるツールのプロジェクトファイルの例ですが、アイコンが設定されていることでダブルクリックでツールが起動しそうに見えます。

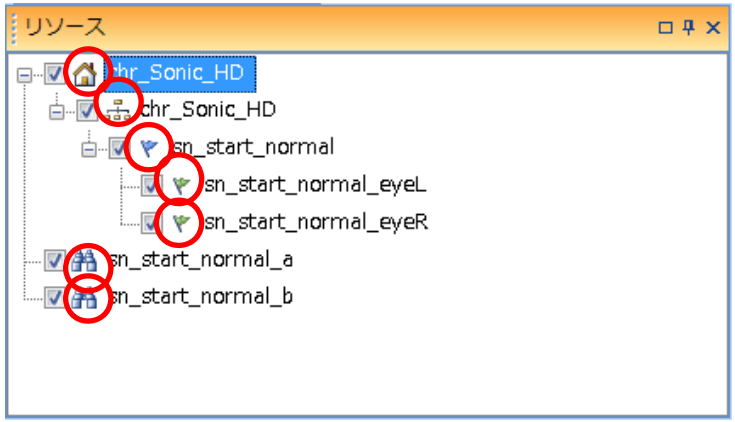


アイコン作成

- ツール内部で使用するアイコンは VisualStudio 付属のものがお勧め。

C:¥Program Files¥Microsoft Visual Studio 9.0¥Common7¥VS2008ImageLibrary¥1041¥VS2008ImageLibrary.zip





ツールの配布に関して

ツールの配布に関して

- 配布方法をどうするか
  - –subversion で配布
  - –ClickOnce 機能の利用
  - –独自配布ツールの作成

ツールの配布に関して

- ClickOnce
  - –アプリケーションの自動更新
  - –依存コンポーネントの更新も行える
  - – 署名はインハウスではセキュアすぎ
  - –インストーラー的なダイアログが困難

ツールの配布に関して

- 独自開発ツール
  - –アプリケーションの更新通知
  - –インストーラ呼び出し形式
    - ClickOnce で制御できなかったもの

UAC制御に関する問題

UAC制御に関する問題

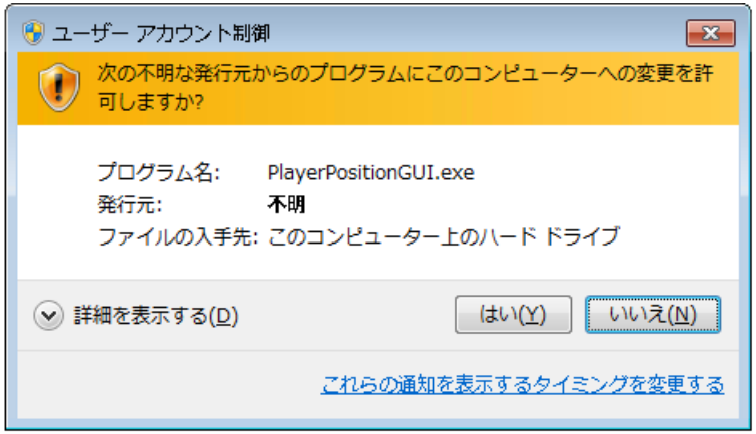
- **UAC**:**U**ser **A**ccount **C**ontrol
  - –WindowsVista 以降で導入されているセキュリティ基盤

UAC制御に関する問題

- WindowsVista 以降では UAC に起因するトラブルもありました。
- "C:¥Program Files" 以下へのアクセスや、拡張子の関連付けを行うためのレジストリ書き換えに UAC の昇格が必要になる

UAC制御に関する問題

- 対応方法
  - 管理者権限がない場合には昇格画面を表示して管理者としてアプリを再起動する



UAC制御に関する問題

- IsAdmin() の帰り値が false の場合は runas 指定で自分自身を再起動する形で実装。

```
class UACUtility
{
    public static bool IsAdmin(){
        System.Security.Principal.WindowsIdentity  usrId = System.Security.Principal.WindowsIdentity.GetCurrent();
        System.Security.Principal.WindowsPrincipal p     = new System.Security.Principal.WindowsPrincipal( usrId );
        return p.IsInRole( @"BUILTIN¥Administrators" );
    }

    public static void RestartApplication(String in_WorkingDir){
        System.Diagnostics.ProcessStartInfo startInfo = new System.Diagnostics.ProcessStartInfo();
        startInfo.UseShellExecute  = true;
        startInfo.WorkingDirectory = in_WorkingDir;
        startInfo.FileName = System.Windows.Forms.Application.ExecutablePath;
        startInfo.Verb = "runas";
        System.Diagnostics.Process p = System.Diagnostics.Process.Start( startInfo );
        System.Windows.Forms.Application.Exit();
    }
}
```

まとめ

- ツール開発に関して役立つ情報をご紹介致しました。

内容

- プレビュー環境の紹介
- 学んだ事
- プレビュー環境の作り方(TIPS)
- 質疑応答

何か質問は御座いますか？

Nakagawa_Nobuo@sega.co.jp