

PhyreStation ユーザガイド

リリース 2.5.0

© 2009 Sony Computer Entertainment Inc.
All Rights Reserved.
SCE Confidential

目次

本ガイドの使い方	5
対象読者	5
表記規則	5
訂正情報	6
1 PhyreStation の概要	7
はじめに	7
主な概念	8
PhyreStation の実行	9
PhyreStation に関する情報	9
Help Viewer	10
2 ワークフロー	13
概要	13
ゲームテンプレート	13
ワークフローの計画	13
ワークフロー	14
3 ワークスペース	19
概要	19
ワークスペースオブジェクト	19
ワークスペースファイル	20
新しいワークスペースの作成	21
既存のワークスペースを開く	21
Workspace Explorer	22
ワークスペースへの外部ファイルのドロップ	25
ワークスペースの保存	25
ワークスペースを閉じる	26
4 データベース	27
概要	27
データベースファイル	27
データベースのブラウズ	28
データベースのアイコン	28
リンク先データベース	29
PhyreEngine™内部データベース	30
データベース処理の実行	30
データベースの処理	32
5 PhyreEngine™オブジェクト	38
概要	38
一時オブジェクト	38
オブジェクトのブラウズ	38
オブジェクト処理の実行	41
オブジェクトの処理	42

6 PhyreStation コマンド	51
概要	51
利用可能なコマンドの表示	51
コマンドの実行方法	51
Command Window	52
コマンドからのログ情報	54
7 スクリプト	55
概要	55
スクリプトファイル	55
スクリプトのブラウズ	56
GUI スクリプトオブジェクトの処理	56
スクリプトの実行	57
コマンドラインからのスクリプトの実行	58
追加の Lua データ型	61
8 PhyreStation のカスタマイズ	64
概要	64
新しい PhyreEngine™ユーティリティの作成	65
PhyreStation への新しい PhyreEngine™オブジェクト型の追加	65
PhyreStation への PhyreEngine™オブジェクトコマンドの追加	66
PhyreEngine™オブジェクトコマンドのリファレンス情報	73
9 カスタムグループ	74
概要	74
カスタムグループのブラウズ	74
カスタムグループの処理	76
10 DNet 通信	79
概要	79
DNet Communication ツールバー	79
DNet による接続	80
DNet 経由でのカメラオブジェクトの表示	80
DNet 経由でのリモートデータベースのロード	81
11 ビューア	85
概要	85
Render Viewer	85
Segment Set Viewer	89
Shader Instance Viewer/Shader Instances Viewer	89
Texture Viewer	90
Shader Program Viewer	91
Shader Group Viewer	91
12 Graph Editor	93
概要	93
共通の Graph Editor 動作	93
個別の Graph Editor 動作	99
13 Animation Editor	102

概要	102
Animation Editor を開く	102
Animation Editor のインターフェース	103
PhyreEngine™アニメーションの編集	105
14 Particle Editor	107
概要	107
Particle Editor	108
パーティクルシステムの作成	111
パーティクルシステムの編集	112
パーティクルシステムのエクスポート	112
15 GUI	114
概要	114
ツールバー	114
ステータスバー	115
ワークエリア	115
ウィンドウの履歴	116
ダイアログボックス	117
PhyreStation のリソースマネージャ	118
16 インストール/アーキテクチャ	119
インストール	119
ライセンス	119
アーキテクチャ	119
17 PhyreStation のログ/ユーザプリファレンス/エラー処理	121
Log ウィンドウ	121
ユーザプリファレンス	123
以前のバージョンのアプリケーション設定の使用	125
ローカリゼーション	125
エラー処理	126
PhyreStation サポート	126
付録 A : 演習	127
概要	127
演習 1 : ワークスペースとデータベースに関する基本的な演習	127
演習 2 : データベースのリンク	130
演習 3 : PhyreEngine™オブジェクトを表示、検索、および編集する	131
演習 4 : スクリプトファイルを作成し、ワークスペースと関連付ける	135
演習 5 : PhyreEngine™データベースを統合する	137
演習 6 : パーティクルシステムをセットアップする	140
用語集	143

本ガイドの使い方

本ガイドの目的は、PhyreStation の概要を紹介した後、PhyreStation を使って PhyreEngine™のデータ やオブジェクトを調べたり、スクリプトを記述/実行して繰り返し作業を自動化したり、ワークスペース を使ってアセットや作業をプロジェクト方式で統合したりする方法を説明することです。

PhyreStation を初めて使用する場合、前半の章の内容を理解していないと後半の章の内容が理解できない可能性があるため、本ガイドを最初から順にお読みになることをお勧めします。

PhyreStation にすでに習熟されている場合は、本ガイドをリファレンスとして使用してください。

対象読者

本ガイドは、アーティストやサポートスタッフをはじめ、テクニカルディベロッパやプログラマに至るまで、PhyreStation ソフトウェアを使用するあらゆるユーザの方を対象としています。

本ガイドは、読者が GUI ソフトウェアを使い慣れていることを前提にしているため、「ドラッグ」、「ドロップ」、「最小化」などの一般的な用語の説明は行いません。

表記規則

ハイパーリンク

ドキュメント内でのナビゲーションを簡単に行えるように、ハイパーリンクを使用しています。ハイパー リンクをクリックした位置に戻れるようにするには、Adobe® Acrobat® Reader の **Previous View/Next View** ボタンを有効にします (Adobe® Acrobat® Reader のメインメニューから **View > Toolbars > More Tools...** を選択します)。

ヒント

PhyreStation を最大限に活用するための GUI ショートカットやその他の有用なヒントについては、「ヒント」という見出しを付け、罫線で囲んで示します。次に例を示します。

ヒント : リンク先データベースのファイルが見つからない場合には、外部リンク先データベース エラーが発生する可能性があります。

注意

PhyreStation や「PhyreStation ユーザガイド」を最大限に活用するために役立つ追加アドバイスや具体的な関連情報については、「注意」という見出しを付け、罫線で囲んで示します。次に例を示します。

注意 : PhyreStation のソースコードはサードパーティに提供されていません。

本文

- 名前の付いたアプリケーションウィンドウや GUI 機能では、太字の「sans-serif」フォントを書式として使用します (例 : **Texture Viewer**、**Log** ウィンドウ、**Save** ボタン)。
- キーボードの機能やキーの名前では、太字の「serif」フォントを書式として使用します (例 : **Ctrl**、**Delete**、**F9**)。
- ファイル名、ソースコード、およびコマンドラインテキストでは、等幅フォントを書式として使用します。次に例を示します。

[PhyreStation_root_directory]/Preferences

- メニューのオプションは「>」記号で区切ります。たとえば、File > Exit は、File メニューの Exit オプションを表します。

グラフィック

本ガイドで使用されているスクリーンショットは、すべて Windows 版の PhyreStation のものです。使用的するデスクトップのスタイルや Windows のバージョンによっては、GUI が多少異なる可能性があります。

訂正情報

本ガイドに対する更新や修正（本ガイドの出版後に PhyreStation に追加された新機能も含む）については、[PhyreStation_root_directory]/Readme_j.txt を参照してください。

1 PhyreStationの概要

この章では、PhyreStation アプリケーションの概要を簡単に説明します。

はじめに

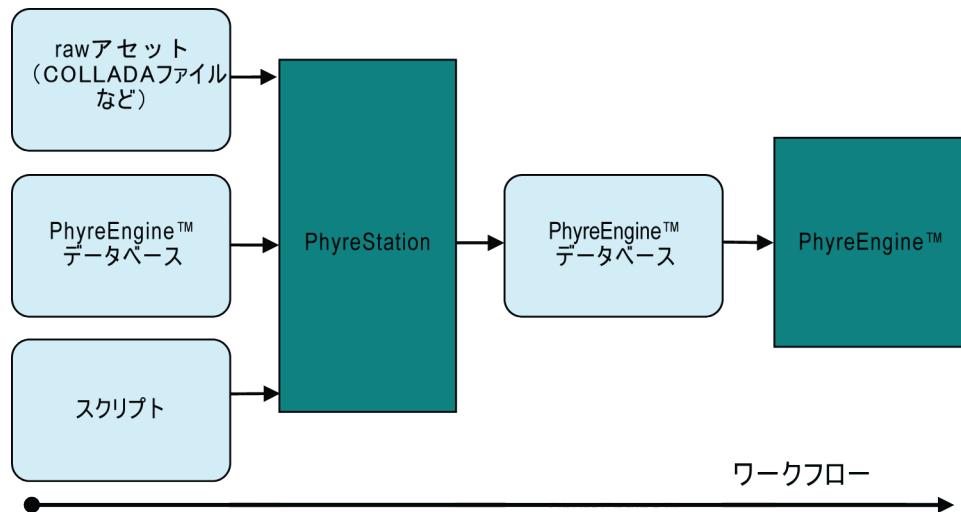
PhyreStation は、開発プロジェクトで使用するデジタルアートアセットを表示/編集/処理/チューニングできるようにするための GUI ツールです。PhyreStation に備わっている各種の機能を使えば、さまざまなものソース間の関係を調べたり、それらを操作したりすることができます。

PhyreStation でサポートされているデータベースのファイル形式は、PhyreEngine™ (.hier、.PSSG) と COLLADATM (.xml、.dae) です。

PhyreStation では、まずプロジェクトを作成し（「ワークスペース」と呼ばれます）、そこにデータベースを格納して、作業を自動化するためのスクリプトを関連付けます。

PhyreStation は、スタンドアロンツールとして動作させることも、バッチモード（GUI なし）で動作させることもできます。バッチモードの場合、より大きなバッチプロセスの一部として PhyreStation を実行し、生のアセットに対してスクリプトを実行させることができます。スクリプトは生のアセットを変換または圧縮することで、PhyreEngine™データベースやそれらの内部の PhyreEngine™オブジェクトを生成します。PhyreStation はこのようにして、すべてのゲームアセットを効率的にコンパクトな最終ソリューションに統合するという目的をユーザが達成できるように支援します。

図 1 PhyreStation のワークフロー



PhyreStation を使えば、ファイル内または実行中の PhyreEngine™インスタンス内に含まれている PhyreEngine™データベースの内容を簡単に調べることができます。また、個々の PhyreEngine™オブジェクトを調べることも可能です。

通常、プログラマは PhyreStation を使って PhyreEngine™データベースを監視/デバッグ/最適化したり、オブジェクトを管理したり、オブジェクトの属性を編集したりします。一部のオブジェクト属性は操作可能で、それには GUI を使用するか、付属の定義済みコマンドのいずれかを使用するか、あるいはプログラマが作成したカスタムコマンドを使用します。定義済みコマンドやカスタムコマンドの大部分は、スクリプト内から使用できます。継続的な繰り返しにより、データだけでなくワークフロー作成プロセスも微調整することができます。使用するスクリプティング言語は Lua です。

アーティストは PhyreStation を使って、アートパッケージから PhyreStation のワークスペース内にアートアセットをインポートし、適切なスクリプトを実行し、結果を PhyreStation 内またはターゲットプラ

ットフォーム上で表示することができます。また、試しに PhyreEngine™オブジェクトのパラメータを変更して、その影響をすぐにターゲットプラットフォーム上で確認することもできます。ターゲットプラットフォームにエクスポートし直す必要は一切ないので、時間の節約になります。

本ガイドでは、ユーザが PhyreStation を最大限に活用できるように、ワークフローと演習を用意しています。

主な概念

ここでは、PhyreStation の主な概念の概要を説明します。

ワークスペース

ワークスペースは PhyreStation の主なコンテナ要素で、一種のプロジェクトファイルです。ワークスペースには、PhyreEngine™データベース、スクリプト、およびカスタムグループ（オブジェクトを格納）のいずれかを含めることができます（これらは「ワークスペースオブジェクト」と呼ばれます）。これらのオブジェクトの大部分は、複数の異なるワークスペース同士で共有できます。

ワークスペース内では、任意の PhyreEngine™データベース内にある任意の PhyreEngine™オブジェクトを表示できます。オブジェクトの名前や型を指定して、ワークスペース内でオブジェクトの検索を行えます。詳細については、「3 ワークスペース」を参照してください。

データベース

PhyreEngine™データベース（以下、単に「データベース」と呼びます）は、PhyreEngine™オブジェクトまたはユーザデータの集合体です。データベースのオブジェクトがリソースを共有するように操作するには、他のデータベース内のリソースを参照させます。共有されたリソースを含むデータベースは、「リンク先データベース」と呼ばれます。詳細については、「4 データベース」を参照してください。

オブジェクト

PhyreEngine™オブジェクトは、PObject クラスインスタンスから派生したデータアセットです。

PhyreEngine™オブジェクトは、事前に定義された任意の PhyreEngine™オブジェクト型、またはユーザによって定義されたカスタム型のどちらであってもかまいません。カスタムオブジェクトを作成するには、「5 PhyreEngine™オブジェクト」を参照してください。

コマンド

PhyreStation コマンドには、次の 2 種類が存在しています。

- PhyreStation 内部コマンド：これらは PhyreStation の内部で定義されたコマンドであり、変更することはできません。通常、このタイプのコマンドは、ワークスペースレベルとデータベースレベルのどちらかで動作します。内部コマンドに含まれるのは、すべてのワークスペースコマンドと (OpenWorkspace () など)、多くのデータベースコマンドです (ResolveLinks () など)。これらのコマンドは、自動的に PhyreStation に登録されます。
- PhyreStationDLL コマンド：これらはオブジェクトレベルとデータベースレベルのどちらかで動作可能なコマンドです。PhyreStationDLL に含まれるコマンドを変更したり、新しいコマンドを作成したりできます。ユーザのカスタムコマンドやオブジェクトクラスのコードを 1 つにまとめるには、独自の PhyreEngine™ User Utility 拡張を作成します。さまざまな実行方法 (GUI コンテキストメニューからの実行、ドラッグ&ドロップ操作による実行、スクリプトからの実行) で PhyreStationDLL コマンドを利用できるようにするには、PhyreStation に登録する必要があります。

詳細については、「6 PhyreStation コマンド」を参照してください。

スクリプト

PhyreStation の主な目的の 1 つは、データ処理の自動化を支援することです。スクリプトファイルは単純な ASCII テキストファイルで、任意の有効な PhyreStation 登録コマンドと Lua コマンドを組み合わせて使用することができます。PhyreStation のスクリプトイントンタプリタは、Lua スクリプトエンジンに基づいています。詳細については、「7 スクリプト」を参照してください。

PhyreStationの実行

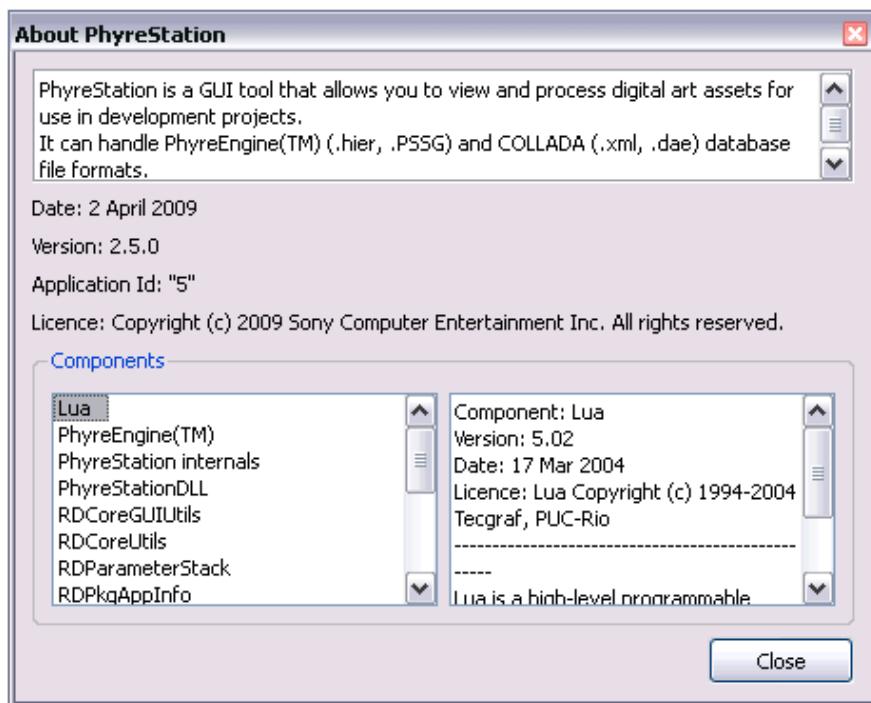
PhyreStation の適切なプラットフォームは、Microsoft Windows XP と Vista だけです。DOS コマンドプロンプトから PhyreStation を実行するには、次のようにします。

```
[PhyreStation_root_directory]> PhyreStation.exe
```

PhyreStationに関する情報

PhyreStation とそのコンポーネントに関するバージョン情報を取得するには、メインメニューから **Help > About PhyreStation...** を選択します。About PhyreStation ダイアログボックスが表示されます。

図 2 About PhyreStation ダイアログボックス



PhyreStation はパッケージ（コードモジュール）とコードライブラリから構成されており、About PhyreStation ダイアログの **Components** セクションにそれらの一覧が表示されます。PhyreEngine™または PhyreStationDLL のどちらかを選択した際に右側のグループボックスに表示される情報は特に役立ちます。右側のグループボックスには、日付、バージョン番号、および PhyreStation に現在登録されている PhyreEngine™オブジェクトの型と PhyreStationDLL コマンドが表示されます。新しいコマンドや PhyreEngine™オブジェクト型を追加すると、その情報が表示されます。

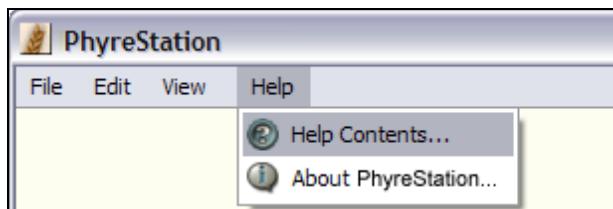
Help Viewer

Help Viewer ウィンドウに表示されるのは、すべての PhyreStation 内部コマンドと登録済み PhyreStationDLL コマンド、および PhyreStationDLL 経由で登録されたすべての PhyreEngine™オブジェクト型に関する情報です。これらのページに表示される情報は、PhyreStation が起動されるたびに更新されます。

Help Viewer の表示

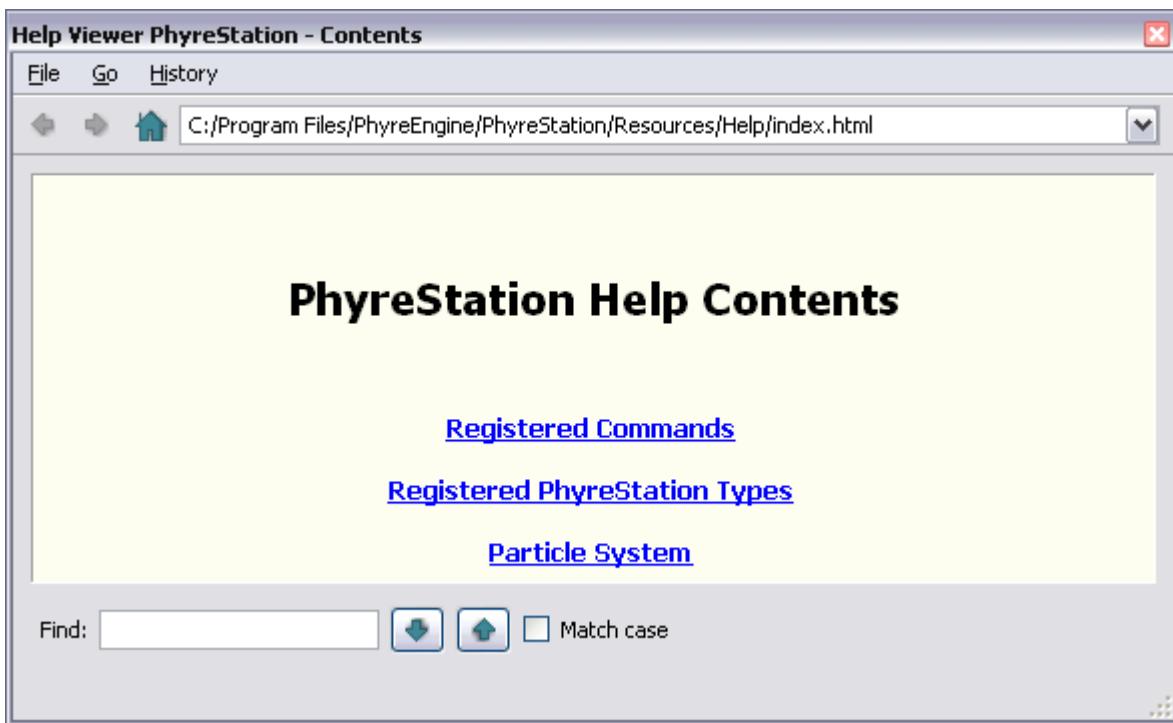
Help Viewer を開くには、メインウィンドウのメニューバーから **Help > Help Contents...** を選択します。

図 3 Help メニュー



Help Viewer の **Contents** ページが開きます。

図 4 Help Viewer の Contents ページ



登録済みコマンドをすべて表示したり、それらに関する情報を検索したりするには、**Registered Commands** をクリックします。すべての登録済み PhyreEngine™オブジェクト型を表示したり、それらに関する情報を検索したりするには、**Registered PhyreStation Types** をクリックします。

図 5 Help Viewer のコマンドページ



ナビゲーション

Help Viewer ページ間のナビゲーションを行うには、次のようにします。

- インデックスページに戻るには、**Home** ボタンをクリックするか、ウィンドウのメニューから **Go > Home** を選択します。
- 以前に表示したページ間を前後に移動するには、矢印ボタンを使用するか、**Go** メニューから項目を選択します。
- あるページをもう一度表示するには、ツールバーのドロップダウンリストからそのページを選択するか、ウィンドウのメニューから **History** を選択します。

検索

Help Viewer では、現在表示されているページで検索機能を使用できます。

- 現在のドキュメント (commands.html など) の内側で検索を行うには、**Find** フィールドにテキストを入力し、**Enter** を押します。
- ドキュメント内を前方に検索するには、**Enter** を押すか、下矢印ボタンをクリックします。
- ドキュメント内の現在位置から後方に検索するには、上矢印ボタンをクリックします。
- Match case** チェックボックスは、検索時に大文字と小文字を区別するかどうかを切り替えます。

Help Viewerへのページの追加

Help Viewer には追加カスタム情報ページを追加できますが、この機能は、ユーザがカスタムワークフローを処理する場合に役立ちます。

追加ページを追加するには、PhyreStation インストールの Resources/Help ディレクトリに外部ヘルプファイルを格納します。次回 PhyreStation が起動されてブラウザが有効になると、**Help Viewer** に追

加ヘルプページが自動的に表示されます。**Help Viewer** は、.txt と.html の拡張子を持つファイルの表示をサポートしています（ただし、そのコンテンツが有効な場合だけです）。

2 ワークフロー

この章では、ユーザの典型的なワークフローをいくつか取り上げ、各ワークフローに記述されているタスクの実行方法に関する詳細説明へのリンクを提供します。

概要

この章で説明するワークフローは、PhyreStation を（主にデータ管理用として）使用する可能性のあるプログラマやアーティストが実施する作業を記述したものとして定義されます。ワークフローは、設定した目標を達成するにはタスクをどのように実行すればよいかに関する概要説明です。また、ワークフローでは、あるタスクを実行するための PhyreStation の操作方法が概要レベルで説明されることもあります。この章で取り上げるワークフローのほかに、[付録A：演習](#)に含まれる一連の演習も参照してください。これらの演習では、ワークフローを完了するにはタスクをどのように実行すればよいかが、詳しく説明されています。

アーティストのワークフロー

PhyreStation の基本的な目的は、生のアートアセットを、プログラマによって指定された構造を持つ PhyreEngine™オブジェクトに変換するプロセスを自動化することです。PhyreStation は、アーティストのワークフローに制限を加えることなしに、アーティストがアートワークを必要な構造で提供するのを支援します。

アーティストは自らの作業習慣に従って、生のアートアセットを自由に作成することができます。PhyreStation を使用する際の唯一の条件は、アートパッケージが COLLADA™ファイル形式へのエクスポートを行えることだけです。

アートワークの外観はしばしば、レンダリングシステム（つまり、アーティストが使用するアートパッケージとプラットフォーム）に大幅に依存します。PhyreStation は PhyreEngine™を使用しています。したがって、PhyreStation を使えば、アーティストはアートアセットを一貫した方法で表示できるようになり、さまざまなプラットフォーム上のさまざまなパッケージの下でアセットを表示した場合の微妙な違いに悩まされることもなくなります。さらに PhyreStation は通常、複雑なシーンのレンダリングを、包括的なアートパッケージよりも効率的に行えます。

ゲームテンプレート

PhyreEngine™ゲームテンプレートは、特定ジャンルのゲームで PhyreEngine™を最大限に活用するための方法を示したサンプルです。また、これらは、PhyreEngine™を使ってソフトウェアを開発する際にセットアップされる典型的なワークフローを示したものでもあります。

ゲームテンプレートは<PhyreEngine>\Samples\Applications ディレクトリ内に収められています。各ゲームテンプレートのディレクトリ内には、そのゲームテンプレートの利点について説明したホワイトペーパーが存在しています。これらのホワイトペーパーでは、技術面での特定の解決策における PhyreEngine™の最適な活用方法が、テクニカルなプログラマの視点から説明されています。

ワークフローの計画

ユーザはおそらく、カスタマイズされた独自のワークフローを作成します。この章で取り上げたワークフローは、ユーザ独自のワークフローを定義する際の参考にしてください。ワークフローを計画する際に考慮すべき点は、次のとおりです。

- PhyreEngine™データをどのように構成するのがもっとも効率的かを判断します。
- PhyreEngine™オブジェクト間のあらゆる依存関係を検討します。
- 最終アプリケーションでの PhyreEngine™オブジェクトのロード/使用方法を決定します。
- ワークフロー内でアートチームがアートワークを使用する方法を確立します。
- 生のアセットを処理して最終アプリケーションで必要になる最適化されたアセットを生成する場合は、PhyreStation のワークスペースを作成します。
- 標準の処理シーケンス（コマンドシーケンス）を確立することで必要な手順がうまく機能することを確認した後、これらのコマンドから、プロセスを自動化する「スクリプト」をコーディングします。
- 必要に応じて、ユーザがすでに作成したか作成する予定の新しい PhyreEngine™オブジェクトを処理できるように、PhyreStation を拡張します。Help > About PhyreStation を選択し、現在登録されている PhyreEngine™ PObject 派生型の一覧を確認してください。
- 必要に応じて、ユーザのワークフローに固有の新しいコマンドを処理できるように PhyreStation を拡張します。これらのコマンドは、ユーザ独自のカスタム PhyreEngine™オブジェクトを処理するか、あるいは処理中に特定のタスクを実行します。Help > About PhyreStation を選択し、現在利用可能になっている一連のコマンドを確認してください。
- PhyreStation アプリケーションで動作する一連のスクリプトを作成して使用します。これらのスクリプトはプログラマのワークフローの自動化された部分を表しており、チームの他のメンバ（アーティストなど）に実行を依頼できます。

ワークフロー

このセクションに含まれるワークフローは、PhyreStation を使って実行可能なプロセスのいくつかについて、その概要を示したものです。それらを以下に示します。

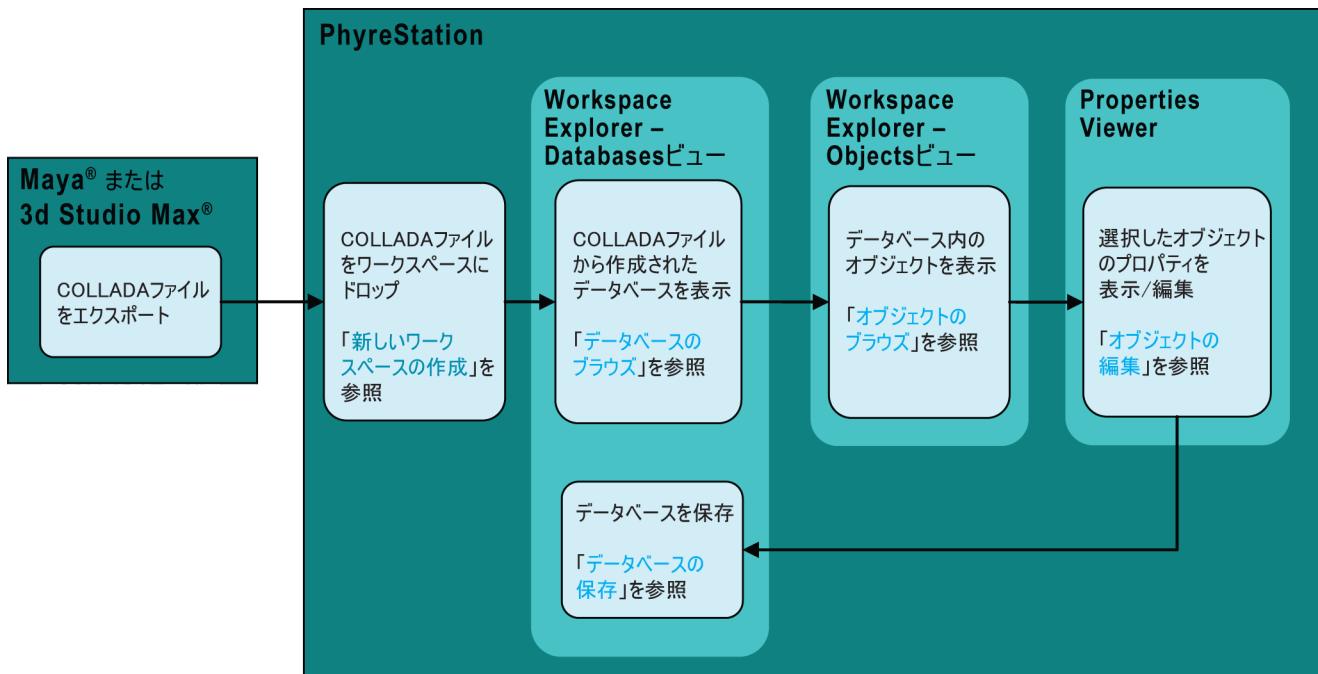
- 入門
- シーンの表示
- PhyreStation のカスタマイズ
- .cgfx から.pssg ファイルへの変換と、その後のデバッグ
- リモートデータベースのデバッグ
- 複数のデータベースに含まれるアセットの統合
- パーティクルシステムの作成

ヒント :ハイパーリンクのクリック後にまたワークフロー図に戻れるように、Adobe® Acrobat® Reader の Previous View/Next View ボタンを有効にします (Acrobat® Reader のメインメニューから View > Toolbars > More Tools... を選択します)。

ワークフロー：入門

このワークフローは、PhyreStation を起動し、COLLADA™ファイルの内容、PhyreEngine™オブジェクト、およびオブジェクトのプロパティを表示する方法を示したものです。

図 6 入門

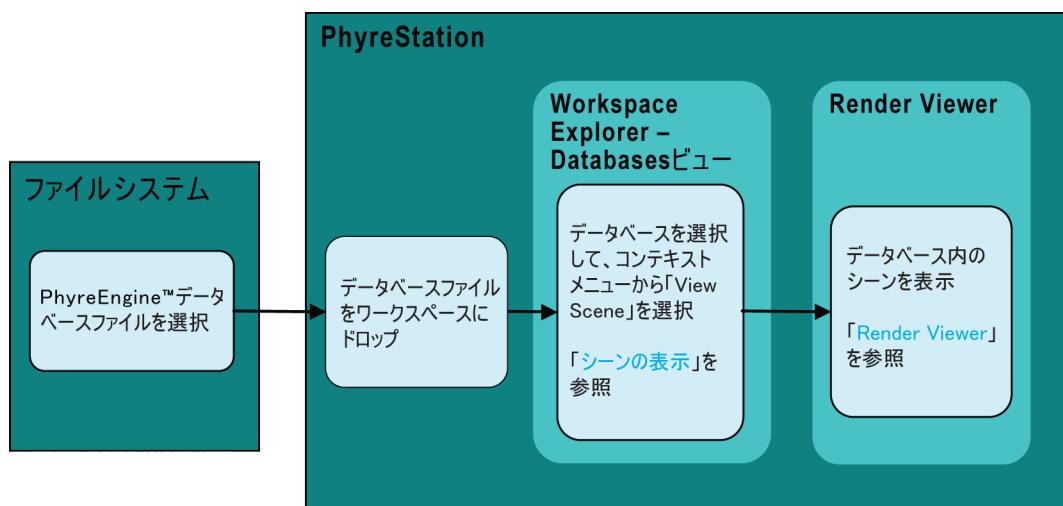


ワークフロー：シーンの表示

このワークフローは、データベースシーン内のオブジェクトを表示するための「ファーストラック」を記述したものです。PhyreStation が一時的なカメラオブジェクトと光源オブジェクトを作成した後、**Render Viewer** 内でシーンが自動的に表示されます。

シーン (PhyreEngine™ルートノード) がデータベース内に存在している必要があります。

図 7 データベース内のシーンの表示



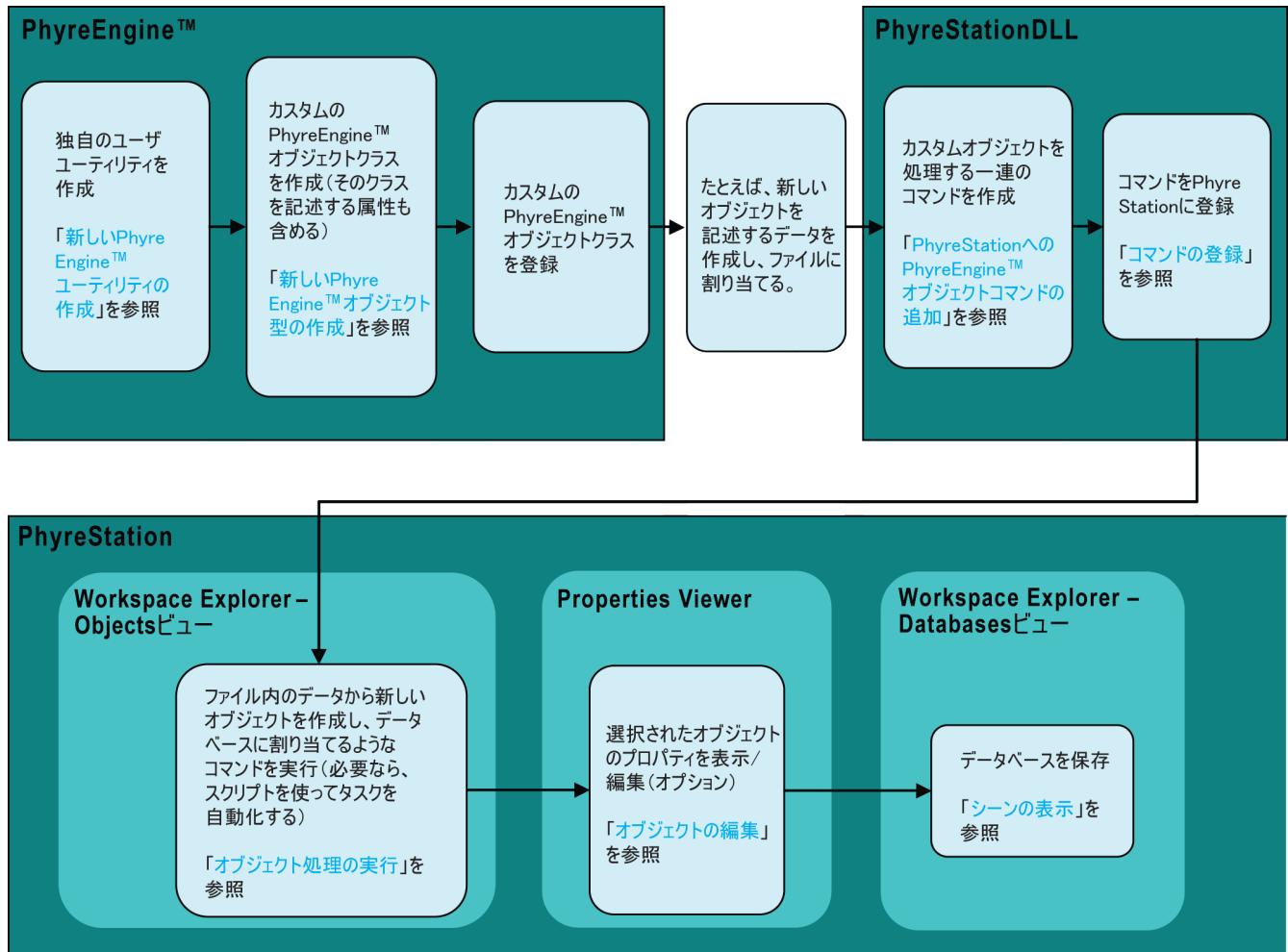
一時オブジェクトの詳細については、「一時オブジェクト」（5 PhyreEngine™オブジェクト）を参照してください。

ワークフロー : PhyreStation のカスタマイズ

このワークフローは、PhyreStation でユーザーのオブジェクトやコマンドを処理できるようにするための、典型的なカスタマイズ方法を記述したものです。

PhyreStation は、ユーザーのデータ管理要件を満たせるように拡張可能となっています。PhyreStation と PhyreEngine™を拡張することで、カスタム PhyreEngine™オブジェクトを処理するユーザーの独自コマンドを作成できます。完全な詳細については、「8 PhyreStation のカスタマイズ」を参照してください。

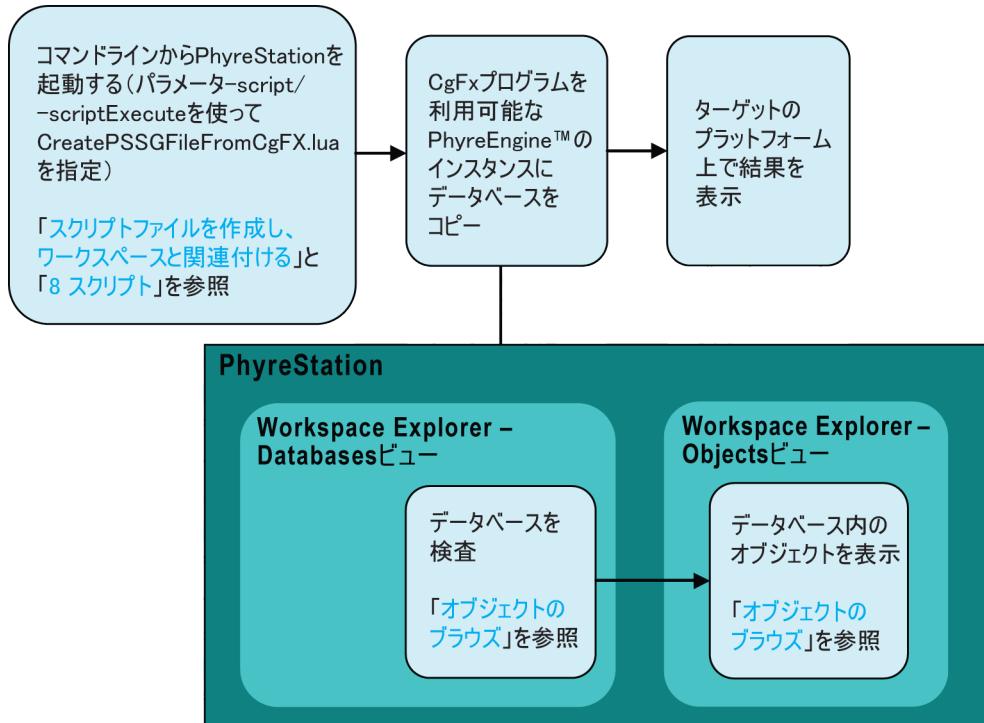
図 8 PhyreStation のカスタマイズ



ワークフロー : .cgfx から.pssg ファイルへの変換と、その後のデバッグ

PhyreEngine™の build ディレクトリに収められているスクリプト CreatePSSGFileFromCgFX.lua は、Sony PLAYSTATION®3、PC ターゲットのいずれかで PhyreEngine™が使用できる形式に、CgFX ファイルタイプを変換する方法を示したものです。

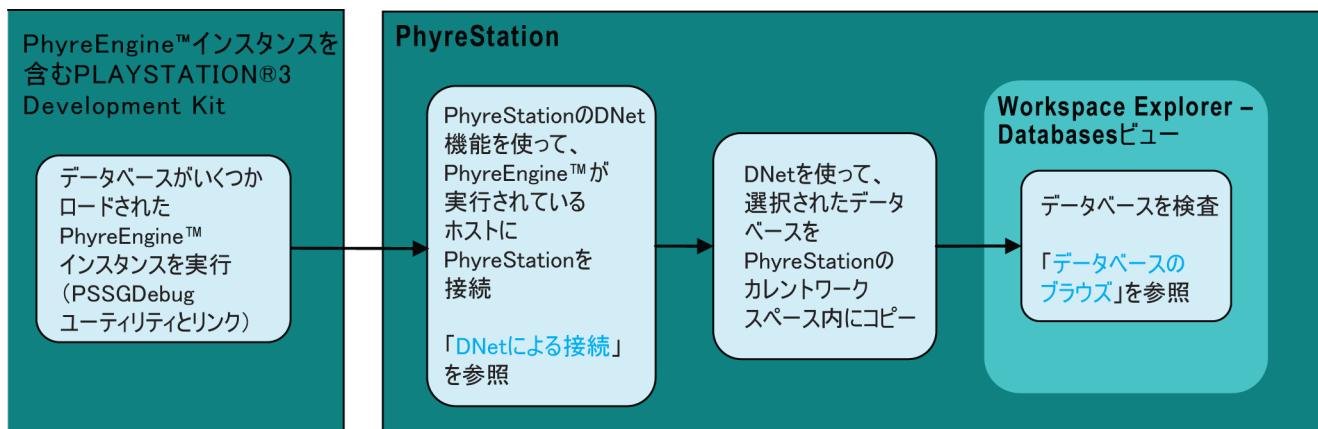
図 9 .cgfx ファイルのデバッグ



ワークフロー : リモートデータベースのデバッグ

このワークフローは、ターゲットプラットフォーム上で動作する PhyreEngine™インスタンスが現在使用している PhyreEngine™データベースを、PhyreStation を使って検査する方法を示したものです。

図 10 リモートデータベースのデバッグ



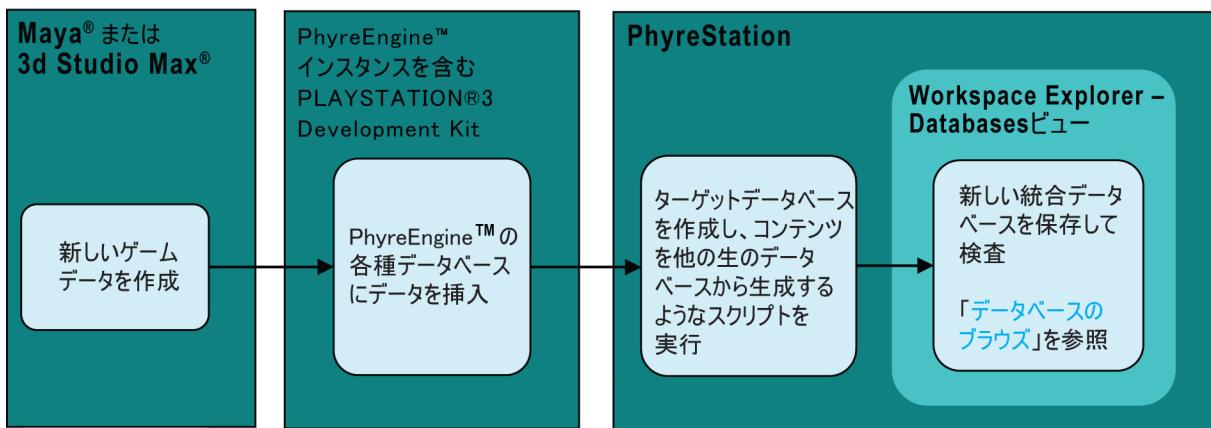
ワークフロー：複数のデータベースに含まれるアセットの統合

PhyreStation を使えば、ゲームの生データの量を減らすために、重複するデータや不要なデータを削除して 1 つの PhyreEngine™ データベースにデータを統合するという、長時間で繰り返し発生することの多い作業を自動化することができます。また、必要であれば、自動化を実現するスクリプトを使ってカスタムデータを挿入することもできます。

このワークフローは、一連の「生の」PhyreEngine™ データベースを受け取り、それらをターゲットプラットフォームに適した 1 つのデータベースに統合する方法を示したもので。結果のデータベースに含まれるのは、関連するアセットのみであり、重複するものはすべて削除されます。最後に、データベースの圧縮が行われます。

このワークフローを開始する前に、「ワークフローの計画」(2 ワークフロー) を参照してください。アセットを統合する方法の詳しい例については、「[演習 5 : PhyreEngine™ データベースを統合する](#)」(付録 A : 演習) を参照してください。

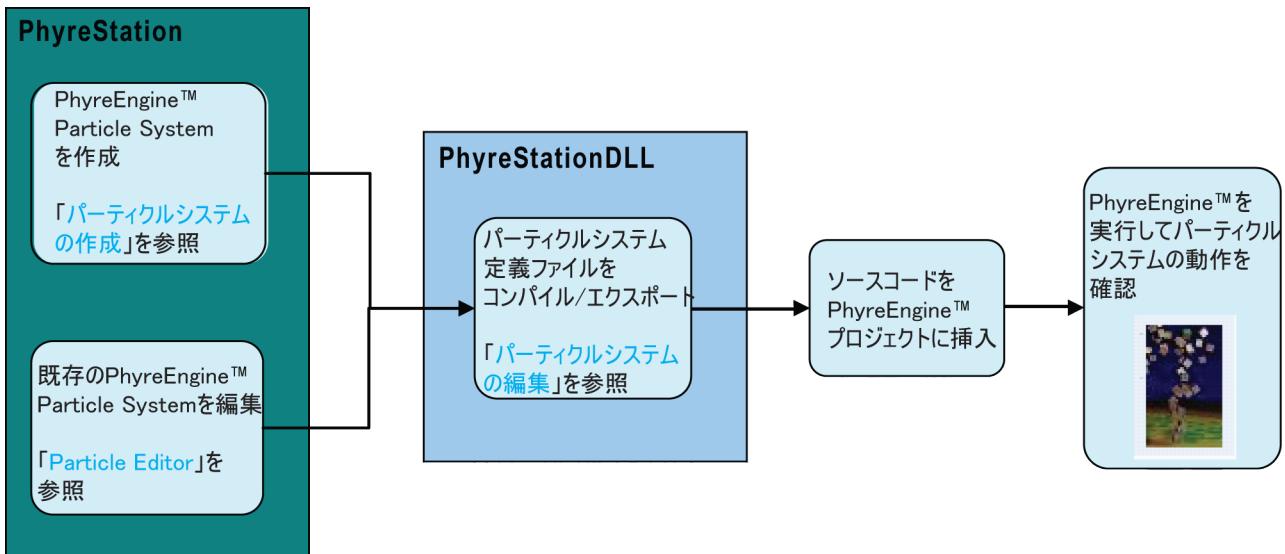
図 11 アセットの統合



ワークフロー：パーティクルシステムの作成

このワークフローは、PhyreEngine™ Particle System の作成方法を示したもので。サンプルパーティクルシステムをセットアップする方法の詳しい説明については、「[演習 6 : パーティクルシステムをセットアップする](#)」(付録 A : 演習) を参照してください。

図 12 パーティクルシステムの作成



3 ワークスペース

この章では、PhyreStation ワークスペースの概念を説明します。ワークスペースやワークスペースファイルの管理方法や、ワークスペースに含まれるオブジェクトへのアクセス方法について説明します。

概要

PhyreStation での作業環境は、ワークスペースとして定義されています。ワークスペースの概念は、他の GUI アプリケーションにおけるプロジェクトの概念に似ています。ワークスペースは、開発プロジェクトに含まれるすべての要素（「ワークスペースオブジェクト」と呼ばれる）のコンテナです。

ワークスペースファイルを開くと、そこにはおそらく 1 つ以上のデータベースが含まれています。さらに、ワークスペースには、いくつかのスクリプトやカスタムグループが関連付けられていることもあります。ただし、常にそうであるとは限りません。

ワークスペースのナビゲーションは、**Workspace Explorer** を使って行います。各 **Workspace Explorer** フィルタビューはそれぞれ異なる方法で、ワークスペース内の情報または特定のオブジェクトを表示します。他のデータベース内のリソースに依存するデータベースがワークスペース内に存在している場合、それらのデータベースは通常、解決する必要があります。PhyreStation はあるデータベースを解決する際に、そのデータベース（と必要に応じて他のデータベース）に含まれるすべてのオブジェクトを、

PhyreEngine™から収集します。この処理中に、PhyreStation はデータベース内のオブジェクトの「表現」を作成します。ワークスペース内でユーザが操作するのは、これらのオブジェクト表現です。

ワークスペースに含まれるデータベース内のオブジェクトがワークスペースに格納されたら、それらのオブジェクトまたはデータベースに対して作業を実行できます。たとえば、あるオブジェクトのコンテキストメニュー や **Command Window** からコマンドを実行できます。コマンドやコマンドを含むスクリプトが実行されると、ワークスペースが自動的に更新されます。

ワークスペースが開かれ、PhyreEngine™内部データベース以外のデータベースが少なくとも 1 つ存在している限り、GUI で PhyreEngine™オブジェクトの作成、編集、または削除を行えます。

1 つのワークスペースが数千個の PhyreEngine™オブジェクトを表す場合もあります。PhyreStation では、あるオブジェクト型を検索したり、名前の一部を入力してオブジェクトを検索したりすることができます。あるオブジェクトを選択した後、そのオブジェクトの属性や属性値を表示することができます。場合によっては、それらの値を編集できます。オブジェクト型によっては、他のウィンドウを開いてオブジェクトの詳細情報を表示させることもできます。

ワークスペースオブジェクト

ワークスペースオブジェクトは次のいずれかになります。

- データベース – PhyreEngine™オブジェクトのコンテナ。「4 データベース」を参照してください。
- PhyreEngine™オブジェクト – データベースに格納されるさまざまなオブジェクトのいずれか（シーケンサプログラム、光源ノードなど）。「5 PhyreEngine™オブジェクト」を参照してください。
- カスタムグループ – ワークスペースオブジェクトのユーザ定義グループ。「9 カスタムグループ」を参照してください。
- スクリプト – 「7 スクリプト」を参照してください。

注意 : PhyreStation は PhyreEngine™データベースをワークスペースオブジェクトとして扱いますが、PhyreEngine™データベースは実際には、PhyreEngine™オブジェクトが格納されたファイルです。PhyreEngine™の視点から見れば、PhyreEngine™データベースは PhyreEngine™オブジェクトではありません。

ワークスペースファイル

ワークスペースはワークスペースファイルとして格納されます。ワークスペースを保存すると、ワークスペースデータへの参照がワークスペースファイル内に保存されます。ワークスペースファイルは比較的小さな XML ファイルであり、PhyreStation で指定されたワークスペース名 (`MyWorkspace.xml` など) と同じ名前を持ちます。

次のディレクトリに、サンプルのワークスペースファイルが含まれています。

`[PhyreStation_root_directory]/PhyreStationWorkspaces`

ワークスペースファイルには、データベースファイルとスクリプトファイルの場所が相対パスとして記録されます。これらのパスは、ロード中に内部で絶対パスへと変換されます。

カレントワークスペース

PhyreStation では、一度に開けるワークスペースは 1 つだけです。これはカレントワークスペースと呼ばれます。カレントワークスペースの名前はアプリケーションのタイトルバーに表示されます。カレントワークスペースが変更後まだ保存されていない場合には、その名前の末尾にアスタリスクが表示されます。

図 13 変更されたワークスペース



ワークスペースが変更済みとしてマークされるのは、次のいずれかが発生した場合です。

- ワークスペースの名前が変更された。
- ワークスペースオブジェクト（データベース、スクリプト、カスタムグループのいずれか）が変更、追加、または削除された。
- PhyreEngine™オブジェクトが作成された、データベースに追加された、あるいはデータベースから削除された。
- PhyreEngine™オブジェクトの属性が変更された。

タイトルバーのアスタリスクは、すべての変更が保存されるまで消えません。ワークスペースを保存しても、ワークスペース内のデータベースが自動的に保存されるわけではありません。それらは別個に保存する必要があります。

注意：ワークスペースファイルの作成時に使用されたロケールがプラットフォーム上の地域設定と異なっていると、カレントワークスペースの名前に予期しない文字が含まれる可能性があります。

デフォルトでは、PhyreStation の起動時に `Untitled.xml` という名前の空のワークスペースが開かれます。この動作を変更するには、**Edit > User Preferences > PhyreStation** を選択し、**Blank workspace** プリファレンスを設定します。

ワークスペースデータ

ワークスペースファイルに含まれるデータには、次の 2 種類があります。

- PhyreEngine™データベース、スクリプト、カスタムグループなどのユーザ定義データ。
- ワークスペースデータ（ウィンドウやその位置など）およびワークスペースプリファレンスデータ。ユーザがワークスペースを再度開くと、PhyreStation はこのデータに基づいて、ワークスペースウィンドウの状態を、ワークスペースが前回閉じられた時点の状態に戻します。ワークスペースを開じると、そのワークスペースが変更済みとしてマークされているかどうかにかかわらず、必ずワークスペースデータがワークスペースファイルに保存されます。

ワークスペースの保存/復元方法を変更したり、デフォルトのワークスペースディレクトリを設定したりするには、**Edit > User Preferences > Workspaces** を選択し、対応するプリファレンスを設定します。また、ワークスペースにはバージョン情報も含まれています。PhyreStation はファイルを開こうと試みる際に、まずこの情報を読み取ります。あるバージョンの PhyreStationにおいて、ファイルのバージョン番号がサポートされていない場合には、ロード処理が異常終了し、警告メッセージが表示されます。

ヒント：ワークスペースファイルのバージョンが使用中の PhyreStation のバージョンよりも新しい場合には、新しいバージョンの PhyreStation にアップグレードします。ファイルが古い場合には、その古いワークスペースに関連付けられたオブジェクトをすべて洗い出し、それらのオブジェクトを含む新しいワークスペースを作成します。

新しいワークスペースの作成

新しいワークスペースを作成するには、次の手順に従います。

- (1) ツールバーのボタンをクリックして **Workspace Explorer** を開いた後、次のいずれかを行います。
 - メインメニューから **File > Workspaces > New** を選択します。
 - メインメニューから **File > Workspaces > Save As...** を選択し、デフォルトの Untitled.xml を別の名前で保存します。
 - **Workspace Explorer** ウィンドウの余白部分を右クリックし、コンテキストメニューから **New workspace...** を選択します。
- (2) ワークスペースの名前（「MyWorkspace」など）を入力し、保存します。

新しいワークスペースを作成したら、PhyreEngine™データベースを格納してスクリプトを関連付けることで、そのワークスペース内のオブジェクトの処理を自動化することができます。詳細については、「4 データベース」と「7 スクリプト」を参照してください。

新しいワークスペースを作成する際には、カレントワークスペースのセッションが自動的に閉じられます。

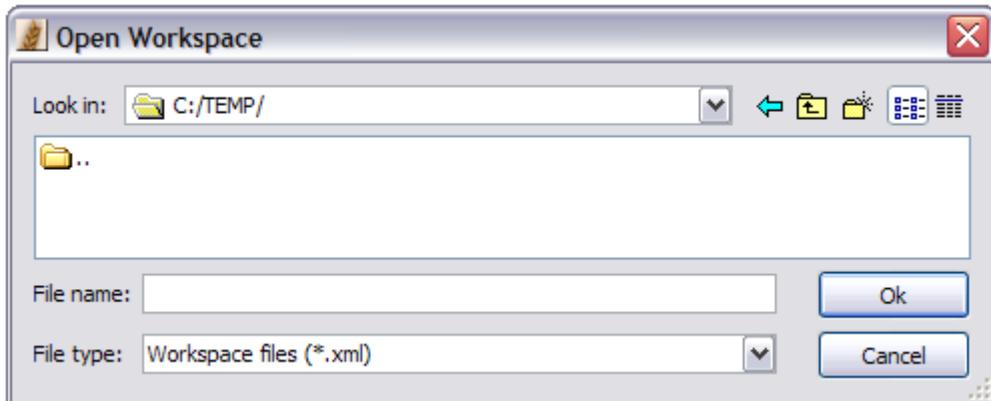
既存のワークスペースを開く

既存のワークスペースファイルを開くと、PhyreStation は自動的にそのワークスペースのデータベースをロードするとともに、そのワークスペースのスクリプトを使用可能な状態にします。あるワークスペースがすでに開かれている状態で別のワークスペースを開くと、最初のワークスペースが閉じられます。ワークスペースを開くには、次のいずれかを行います。

- メインメニューから **File > Workspaces > Open...** を選択します。
- **Workspace Explorer** ウィンドウの余白部分を右クリックした後、コンテキストメニューから **Open workspace...** を選択します。
- OS のファイルブラウザから PhyreStation のメインウィンドウに、ワークスペースファイルをドラッグします。

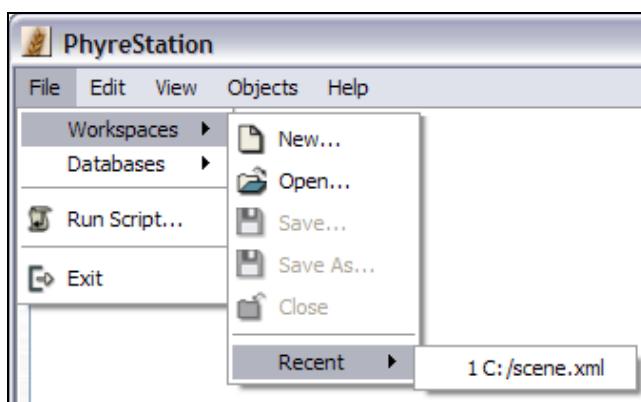
最初の 2 つの方法を使用した場合には、**Open Workspace** ダイアログボックスが表示されます。

図 14 Open Workspace ダイアログボックス



最近表示したワークスペースを再度開くには、メインメニューから **File > Workspaces > Recent** を選択します。

図 15 最近使用したワークスペースのロード



ヒント : このリストに表示する最近使用したワークスペースファイルの数を指定するには、**Edit > User Preferences > PhyreStation** を選択し、**Recent Workspaces** プリファレンスを設定します。

Workspace Explorer

Workspace Explorer を使えば、カレントワークスペース内のすべてのオブジェクトにアクセスできます。一度に複数の **Workspace Explorer** を開くことができます。**Workspace Explorer** を開くには、メインツールバーのボタンをクリックします。

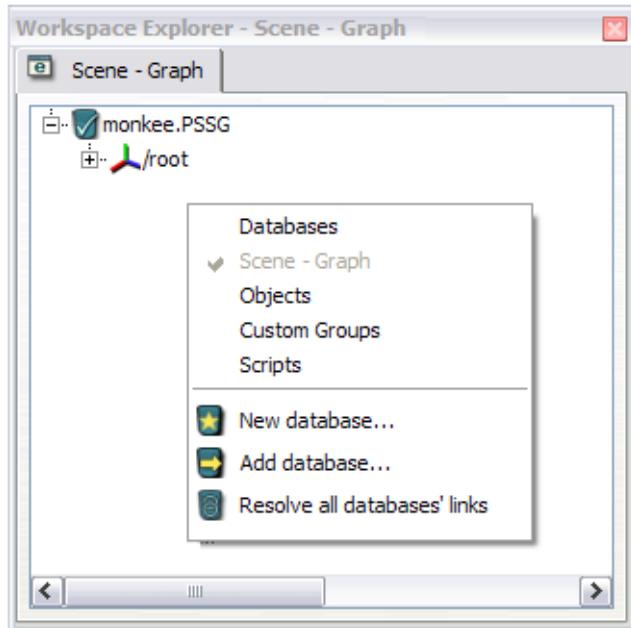
図 16 Workspace Explorer ツールバーボタン



Workspace Explorer のビュー

Workspace Explorer では、いくつかのフィルタビューが利用可能になっています。各フィルタビューには、特定タイプのワークスペースオブジェクト、またはオブジェクトのサブセットが表示されます。各フィルタビューは情報をある特定の方法で簡潔に表示し、一連の独自メニューとオプションを備えています。フィルタビューを選択するには、**Workspace Explorer** ビューの余白部分（項目が存在しない部分）で右クリックし、ビューのコンテキストメニューを表示させます。このメニューからフィルタビューを選択できます。

図 17 Workspace Explorer のコンテキストメニュー



注意 : Shift を押しながら一連のノードオブジェクト（折りたたまれたノードオブジェクトも含む）を選択すると、折りたたまれたノードの子ノードも選択されます。それらの子ノードは表示されていないにもかかわらず、子ノードに対してもすべてのコマンド（Delete など）が実行されます。これを避けるには、折りたたまれたノードをすべて Ctrl キーを使って展開し、子ノードの選択を解除してください。

注意 : PhyreStation では、PhyreStation が依存する PhyreEngine™ の安定性を損なう恐れのある処理を、ユーザが実行できるようになっています。PhyreStation は、ユーザがそれらの処理を実行する前に警告を発しますが、その実行を防ぐことはしません。ユーザがその実行を継続した場合、PhyreEngine™ と PhyreStation の両方の信頼性が低下する危険性があります。

シーディングラフビュー

Workspace Explorer - Scene-Graph ビューには、ワークスペース内の PhyreEngine™ PNode 派生オブジェクトのみが、データベース別にグループ化されて表示されます。

各シーディングラフビューは、PhyreEngine™ PNode 派生オブジェクト（カメラ、光源、関節など）の階層です。これらのオブジェクトを相互に接続したものが、グラフィカルシーンの物理的または論理的な構造になります。シーンをレンダリングする際には、特殊なノードオブジェクト型を派生させて特定の機能を実行することもできます（シーンに光源やジオメトリを追加するなど）。

データベースにルートノードが少なくとも1つ含まれている場合、親ノードを持つ子ノードはすべて、階層化されて表示されます。そうでない場合には、親を持つノードも持たないノードも、データベース項目の下でフラットなリストとして表示されます。

オブジェクトビュー

Workspace Explorer - Objects ビューには、ワークスペース内のすべてのリンク解決済みデータベースとオブジェクト収集済みデータベースに含まれる、すべての PhyreEngine™ オブジェクトが表示されます。このビューの詳細については、「オブジェクトのブラウズ」（5 PhyreEngine™ オブジェクト）を参照してください。

データベースビュー

Workspace Explorer - Databases ビューには、ワークスペース内の最上位の PhyreEngine™データベースと、リンク先データベースの第 1 階層が表示されます。このビューの詳細については、「データベースのブラウズ」(4 データベース) を参照してください。

カスタムグループビュー

Workspace Explorer - Custom Groups ビューには、ワークスペースオブジェクトのユーザ定義グループが表示されます。このビューの詳細については、「カスタムグループのブラウズ」(9 カスタムグループ) を参照してください。

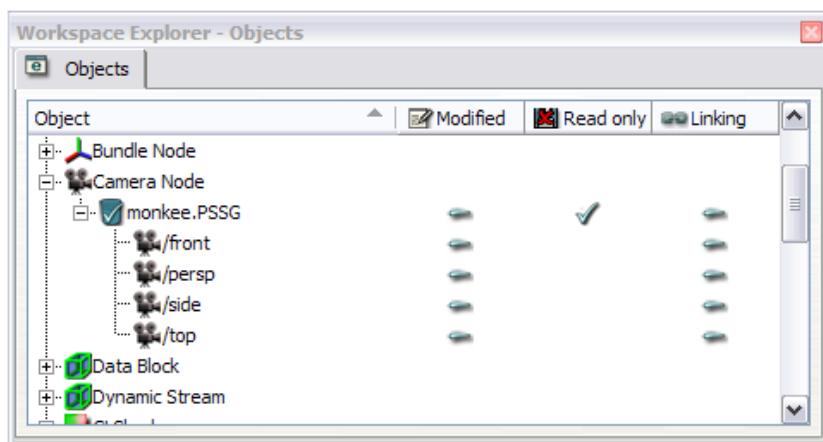
スクリプトビュー

Workspace Explorer - Scripts ビューには、カレントワークスペースに関連付けられたスクリプトファイルが表示されます。このビューの詳細については、「スクリプトのブラウズ」(7 スクリプト) を参照してください。

Workspace Explorer のフィルタプロパティ

Workspace Explorer では、ワークスペースオブジェクトの追加プロパティを表示できます。これらのプロパティを表示するには、Edit > User Preferences > Workspace を選択し、Enable workspace explorer filter properties チェックボックスを選択します。図 18 に、このオプションを有効にした状態の Workspace Explorer を示します。

図 18 Workspace Explorer のフィルタプロパティ



ヒント: フィルタプロパティのプリファレンスを有効にすると、ワークスペースオブジェクトに関する基本情報が表示されます。PhyreEngine™オブジェクトのすべての属性を表示するには、そのオブジェクトのコンテキストメニューから Open Properties window... を選択します。「オブジェクトの編集」(5 PhyreEngine™オブジェクト) を参照してください。

フィルタプロパティは次のとおりです。



Modified – データベースが前回保存された後にこのオブジェクトが変更されたかどうか。



Read only – この PhyreEngine™データベースファイルが読み取り専用ファイルかどうか。データベースオブジェクトにのみ適用されます。



Linking – この PhyreEngine™オブジェクトが別の PhyreEngine™オブジェクトから参照されているかどうか。このオブジェクトがデータベースの場合、このプロパティは、別のデータベースがこのデータベースにリンクされているかどうかを示します。

各列に表示される情報は次のとおりです。

-  このプロパティが true です。
-  このプロパティが false です。
-  特定の PhyreEngine™データベースファイルが見つかりません。それはおそらく存在していないか、現時点では利用不可能になっています。

その他の機能

- オブジェクト型の並び順を、アルファベットの降順または昇順に変更するには、プロパティ見出しの **Object** ラベルをクリックします。
- PhyreStation はフィルタプロパティを 5 分おきに自動更新します。プロパティを任意のタイミングで更新するには、**Workspace Explorer** のコンテキストメニューから **Refresh** を選択するか、F5 キーを押します。
- プロパティ見出しのタイトルを非表示にするには、コンテキストメニューから **Hide Text** を選択するか、F4 キーを押します。

ヒント : PhyreEngine™データベースを解決する前にフィルタプロパティを無効にすると、その処理速度が向上します。

ワークスペースへの外部ファイルのドロップ

外部ファイルを次のいずれかにドラッグすれば、PhyreStation にデータをロードすることができます。

- ワークスペース（空の作業領域）
- **Workspace Explorer** ビュー（オブジェクト、データベース、シーン）
- PhyreEngine™データベース
- PhyreEngine™オブジェクト

ファイルをドロップすると、PhyreStation によって一連の PhyreStationDLL ファイルハンドラが繰り返し呼び出され、ファイルがドロップされた場所に基づいてそのファイルが処理可能かどうかが判定されます。ファイルが処理可能な場合には処理が行われ、ドロップが完了します。適切なファイルハンドラが存在しない場合、PhyreStation はそのファイルをデータベースファイルとみなし、カレントワークスペースにデータベースとして追加します。そのファイルがデータベースファイルでない場合、PhyreStation はその項目を表現したバイナリオブジェクトを含むデータベースを作成します。

PhyreStation は現在、3 種類のファイルハンドラ（オーディオ、CgFx、テクスチャ画像）をサポートしています。

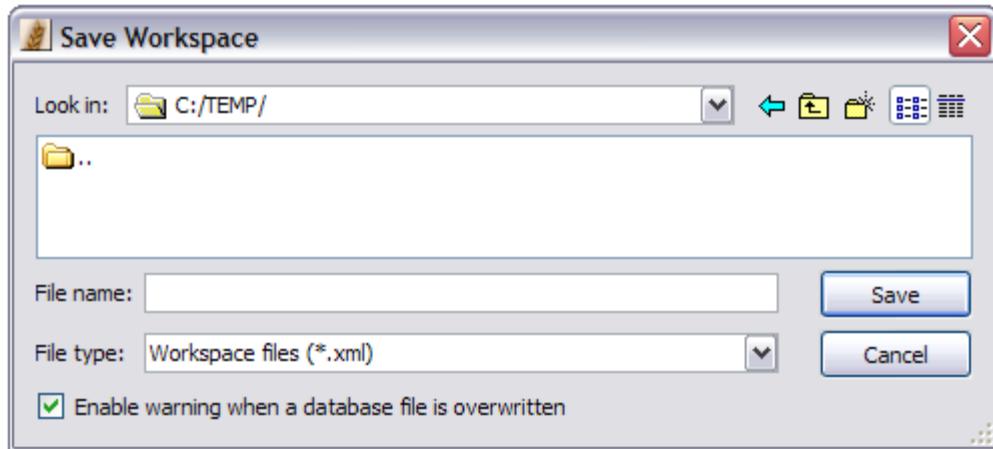
スクリプティングでは、この機能はスクリプトコマンド `AddExternalObject` によって提供されています。利用可能なコマンドの一覧については、「Help Viewer」（1 PhyreStation の概要）を参照してください。

ワークスペースの保存

書き込み可能なワークスペースファイルが変更されると、**File > Workspaces > Save** オプションが有効になります。カレントワークスペースがデフォルトワークスペース `Untitled.xml` である場合には、このオプションは利用できません。

ワークスペースファイルを任意のタイミングで保存するには、メインメニューから **File > Workspaces > Save As...** を選択します。Save Workspace ダイアログボックスが表示されます。

図 19 Save Workspace ダイアログボックス



ワークスペースの自動保存

ワークスペースを指定された間隔で自動的に保存するには、**Edit > User Preferences > Workspace** を選択し、**Auto save workspace** プリファレンスを設定します（デフォルトは「オフ」）。このプリファレンスは、ワークスペース内のデータベースの最新リスト、ワークスペースのプリファレンス、およびウィンドウレイアウト設定を保存するように、PhyreStation に指示します。PhyreStation は、変更された PhyreEngine™ データベースが存在していても、それらを一切保存しません。変更されたデータベースがワークスペース内に存在する場合、ワークスペースは保存されますが、変更済みとマークされたままになります（タイトルバーにアスタリスクが表示される）。

カレントワークスペースがデフォルトの Untitled.xml ワークスペースである場合、ワークスペースの名前が変更されるまで PhyreStation は自動保存を実行しません。

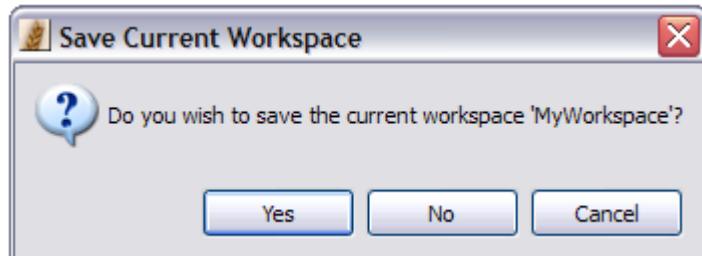
ワークスペースファイルの書き込みが禁止されている場合、PhyreStation は自動保存を実行しません。警告は一切表示されません。

注意： アプリケーションがスクリプトを実行している間は、自動保存機能が有効になっていても保存は実行されません。

ワークスペースを閉じる

保存されていない変更が含まれるワークスペースセッションを閉じようすると、ワークスペースの保存を促すメッセージボックスが表示されます。また、変更されたデータベースがワークスペース内に存在する場合にも、警告が表示されます。

図 20 Save Current Workspace メッセージボックス



4 データベース

この章では、PhyreEngine™データベースとデータベースファイルについて説明します。また、データベースに対して処理を実行する方法についても説明します。

概要

PhyreEngine™データベース（ここでは「データベース」と呼ぶ）は、PhyreEngine™オブジェクトの集合体です。

PhyreEngine™オブジェクトをデータベースに追加するには、まずデータベースを作成するか、既存のデータベースをワークスペース内にロードします。既存のデータベースは通常、どこかにある何らかの媒体上のファイルです。ただし、「ライブ」状態のPhyreEngine™インスタンスからデータベースをロードすることもできます。

既存のワークスペースを開くと、そのワークスペースに関連付けられたデータベースが自動的にロードされますが、必ずしも解決されるとは限りません（解決とは、オブジェクトを収集してワークスペースに格納することを意味する）。

カレントデータベース内のデータベースを表示/アクセスするには、**Workspace Explorer - Databases** ビューを開きます。

データベースファイル

データベースファイルには、PhyreEngine™オブジェクトデータと、オブジェクト間の関係についての情報が格納されます。PhyreStationでサポートされるデータベースファイル形式は、次の4つです。

表1 データベースファイル

ファイル形式	解説	アクション
PhyreEngine™ .PSSG ファイル	拡張子が.PSSG のファイルは、バイナリデータファイルです。これが PhyreStation のデフォルト形式です。	インポートまたはエクスポート
PhyreEngine™ .hier ファイル	拡張子が.hier のファイルは、判読可能な XML 版の PhyreEngine™ ファイルです。	インポートまたはエクスポート
PhyreEngine™ .gz ファイル	拡張子が.gz のファイルは、圧縮されたバイナリデータファイルです。	インポートまたはエクスポート
COLLADA™ ファイル (.xml または.dae)	COLLADA™は、アートアセットの表現に一般的に使用される標準ファイル形式です。	インポートのみ

PhyreStation では、表1に示したものに限らず、どのような名前/ファイル拡張子を持つファイルでも使用できます。

注意：PhyreEngine™データベースファイルに文字「#」を含めることはできません。PhyreStation では、文字「#」をオブジェクトの名前の一部として使用することはできません。

サードパーティのデータ

データベースファイルには、サードパーティソフトウェアで使用されるが PhyreEngine™エンジンには無意味なデータを含めることができます。こうしたデータは、PhyreEngine™エンジンによって識別され、バイナリオブジェクトの内側にラップされた後、PhyreStation では無視されます（ただし、バイナリオブジェクトを処理するコマンドが記述されている場合は除きます）。したがって、任意のタイプの.xml

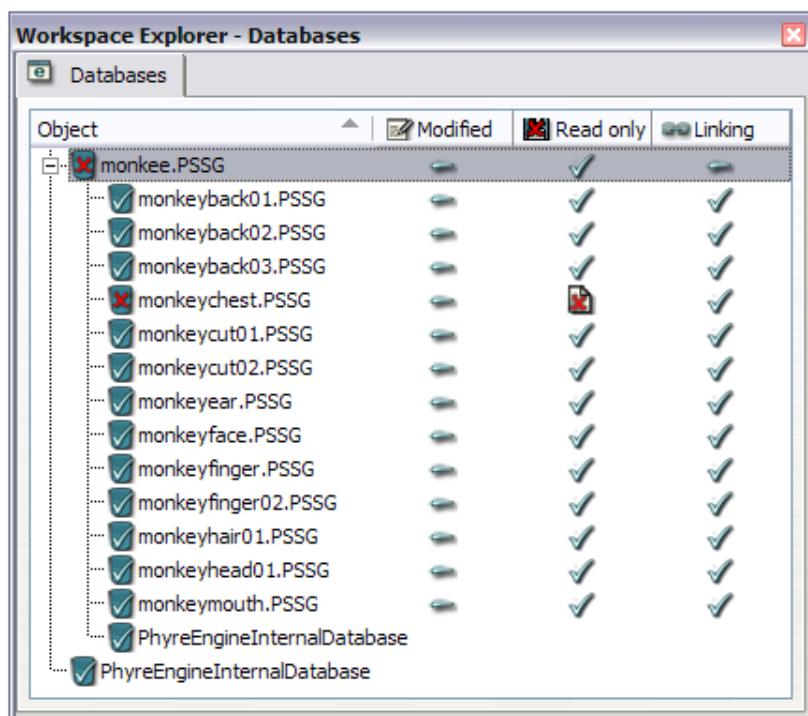
ファイルを PhyreStation にロードした場合に、**Workspace Explorer** に空のデータベースのようなものが表示される可能性があります。この動作は正常であり、バグではありません。すべての非 PhyreEngine™ データはデータベースファイル内にそのまま保持され、適切なサードパーティソフトウェアを使えばそれらのデータにアクセスすることができます。**Workspace Explorer - Objects** ビューでは、バイナリオブジェクトは表示できますが、そのオブジェクトが表現しているデータは表示できません。

データベースのブラウズ

カレントワークスペース内のデータベースを表示するには、次の手順に従います。

- (1) ツールバーボタンを使って **Workspace Explorer** を開きます（図 16 を参照）。
- (2) **Workspace Explorer** で右クリックし、コンテキストメニューから **Databases** を選択します。
Workspace Explorer - Databases ビューに、カレントワークスペースに含まれるすべての最上位データベースが表示されます。

図 21 Workspace Explorer – Databases ビュー



The screenshot shows the 'Workspace Explorer - Databases' window. The title bar says 'Workspace Explorer - Databases'. Below it is a toolbar with a 'Databases' button. The main area is a grid table with columns: Object, Modified, Read only, and Linking. The 'Object' column lists database names. The 'Modified' column has a checkmark icon. The 'Read only' column has a checkmark icon. The 'Linking' column has a checkmark icon. Some rows have a red 'X' icon next to them.

Object	Modified	Read only	Linking
monkeye.PSSG	✓	✓	✓
monkeyback01.PSSG	✓	✓	✓
monkeyback02.PSSG	✓	✓	✓
monkeyback03.PSSG	✓	✓	✓
monkeychest.PSSG	✗	✓	✓
monkeycut01.PSSG	✓	✓	✓
monkeycut02.PSSG	✓	✓	✓
monkeyear.PSSG	✓	✓	✓
monkeyface.PSSG	✓	✓	✓
monkeyfinger.PSSG	✓	✓	✓
monkeyfinger02.PSSG	✓	✓	✓
monkeyhair01.PSSG	✓	✓	✓
monkeyhead01.PSSG	✓	✓	✓
monkeymouth.PSSG	✓	✓	✓
PhyreEngineInternalDatabase	✓	✓	✓
PhyreEngineInternalDatabase	✓	✓	✓

データベースのアイコン

Workspace Explorer - Databases ビュー(図 21)に表示されるデータベースアイコンは、状態を示します。

- ロード済み  カレントワークスペースに追加されたすべての新規データベースのデフォルト状態
- リンク解決済み  データベースのすべての参照先要素（依存先も含む）の検証に成功しました。データベースのリンクを解決するには、そのデータベースのコンテキストメニューから **Resolve Links** を選択します。
- リンク未解決  あるデータベースが「リンク未解決」（壊れた状態）になるのは、そのデータベースの参照先要素または依存先のうち、正常に検証できないものが 1 つ以上存在している場合です。

最上位データベースまたはリンク先データベースのアイコンに赤色の×印が表示され

ている場合、それは、PhyreEngine™がそのデータベースの一部のリソースリンクの解決に失敗したことを示します。ある外部データベースがリソースエラーを示している場合、その最上位データベースも未解決としてマークされます。処理を継続する前にすべての解決エラー問題を修正すべきです。

ヒント：リンク先データベースファイルが見つからない場合には、リンク先データベースエラーが発生する可能性があります。

オブジェクト収集済み



選択されたデータベースに含まれるすべてのオブジェクトが収集され、ワークスペース内にロードされました。リンク先オブジェクト（またはリンク先データベース）は一切、ワークスペース内にロードされていません。

アンロード済み



データベースのアンロードを行うと、そのすべてのオブジェクトが PhyreEngine™とワークスペースの両方から削除されます。データベースのアンロードは、貴重なシステムリソースを利用可能にするのに役立ちます。

データベースをアンロードすると、リンク先データベースもすべて PhyreEngine™によってアンロードされ、可能な限りリンク先データベースの表現もワークスペースから削除されます。ただし、「他の」 ロード済みデータベースからリンク/ロードされたリンク先データベースはすべてメモリにロードされたままになります。

データベースをアンロードするには、そのデータベースのコンテキストメニューから **Unload** を選択します。

リンク先データベース

リンク先データベースとは、別の PhyreEngine™データベースから参照されている PhyreEngine™データベースのことです。これが **Workspace Explorer - Databases** ビュー内に表示されるのは、データベースが解決され、PhyreEngine™がリソース接続を決定した時です。

リンクが必要となるのは、あるデータベースに属するオブジェクトが他の 1 つ以上のデータベース内のオブジェクトに依存している場合です。リンク先データベースは、その参照元データベース（最上位データベース）の子項目として表示されます。リンク先データベースは、カレントワークスペースに最上位データベースとして追加することもできます。

リンク先データベースと PhyreEngine™

PhyreEngine™には、最上位データベースやリンク先データベースの概念はありません。すべてのデータベースは同じです。**Workspace Explorer - Databases** ビューには、他の PhyreStation 処理では意識することのない PhyreEngine™データベース間のリソースリンク関係が表示されます。

ヒント：あるデータベースが別のデータベースにリンクされているかどうかを判定する他の唯一の方法は、オブジェクトの **Properties** ビューを表示することです。オブジェクトが、他のデータベースのオブジェクトを依存先または依存元として参照している可能性があります。「オブジェクトの編集」(5 (PhyreEngine™オブジェクト) を参照してください。

Workspace Explorer - Databases ビューには、複雑である可能性のあるデータベース間の関係に含まれるリソースリンクの第 1 階層のみが表示されます。

PhyreEngine™データベースは、リンク先データベースとしてワークスペースに追加することはできません。最上位データベースとしてのみ追加できます。

PhyreEngine™は各データベースを最小単位として扱っており、データベースの部分読み込みはサポートしていません。PhyreEngine™では、未解決リンクを含むデータベースリソースや未解決リンクを含むデータベースに依存するデータベースリソースのレンダリングは行えません。

PhyreEngine™は内部的に、最上位データベースの解決に必要な情報を取得する際に必要となるデータベースの間にリンクを生成します。データベースの解決が完了すると、他の PhyreEngine™オブジェクトから参照されている PhyreEngine™オブジェクトは、GUI から容易に削除や移動を行えなくなります。

PhyreEngine™が何らかの作業を実行する必要が生じた際に、必要な項目がすべて利用可能であることが保証されます。

最上位データベースをワークスペースから削除したときに、それがたまたまリンク先データベースでもあった場合、PhyreEngine™はそのリンク先データベースを使用し続けます。ユーザがそのデータベースをワークスペースに追加しなかったとしても、PhyreEngine™はそのデータベースを使用します。

注意：

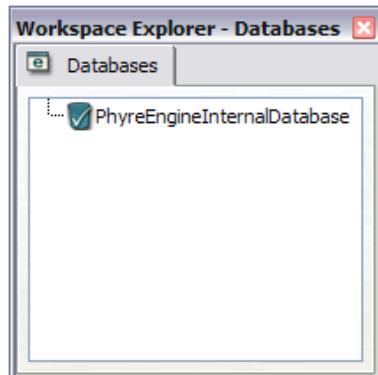
- リンク/ロードの状態は、ワークスペースファイルには保存されません。ワークスペースファイルからデータベースがロードされると、その状態は必ずロード済みか解決済みになります。
- データベースのリンク先データベースがさらに別のデータベースにリンクしているというように、最上位データベースは複雑な相互関係のリソース依存関係を含んでいる可能性がありますが、**Workspace Explorer - Databases** ビューには第 1 階層のデータベースだけが表示されます（図 21 を参照）。

PhyreEngine™内部データベース

PhyreEngine™内部データベースは、すべてのワークスペース内に常に存在する特殊なデータベースです。このデータベースは、ほとんどの開発プロジェクトで必要になる、使用頻度の高い PhyreEngine™オブジェクトを提供します。

新しいワークスペースを作成すると、PhyreEngine™内部データベースが自動的に追加されます。このデータベースは **Workspace Explorer - Databases** ビューで常に表示されています。

図 22 PhyreEngine™内部データベース



PhyreEngine™内部データベースを変更することも可能ですが、そこに事前に用意されたオリジナルのオブジェクトは一切、永久的に削除しないことをお勧めします。

データベース処理の実行

データベースの処理を実行するには、次の機能を使用します。

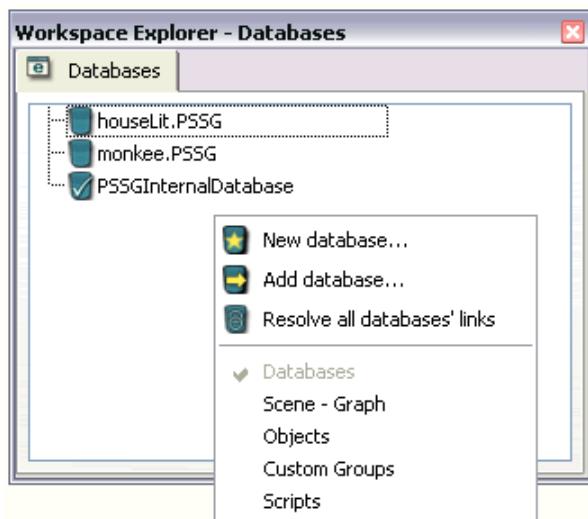
- 一般的なコンテキストメニュー。これは、**Workspace Explorer - Databases** ビューの余白部分を右クリックすると表示されます。
- 具体的なコンテキストメニュー。これは、**Workspace Explorer - Databases** ビューでデータベース名を右クリックすると表示されます。

- スクリプト - 「[Z スクリプト](#)」を参照してください。

一般的なコンテキストメニュー

一般的なデータベース処理を実行するには、**Workspace Explorer - Databases** ビューの余白部分を右クリックします。

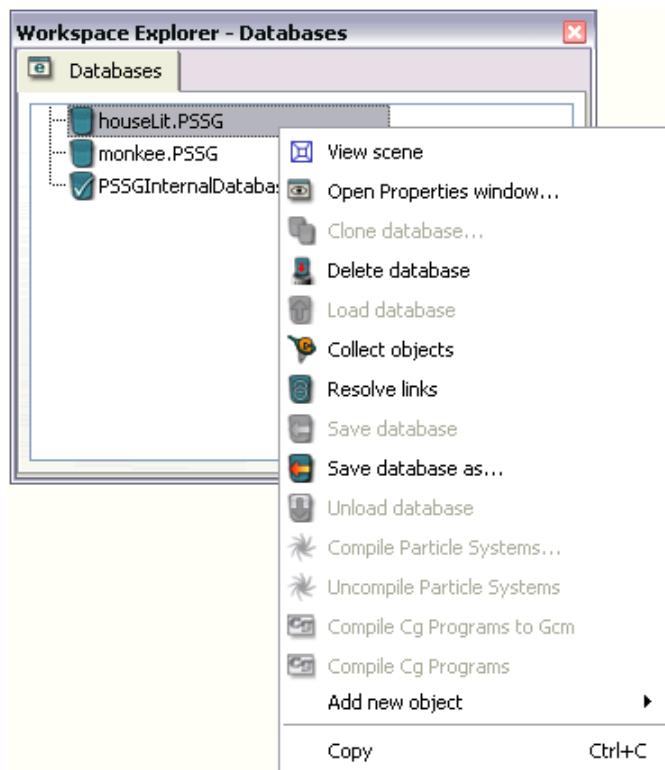
図 23 一般的なデータベースコンテキストメニュー



データベースコンテキストメニュー

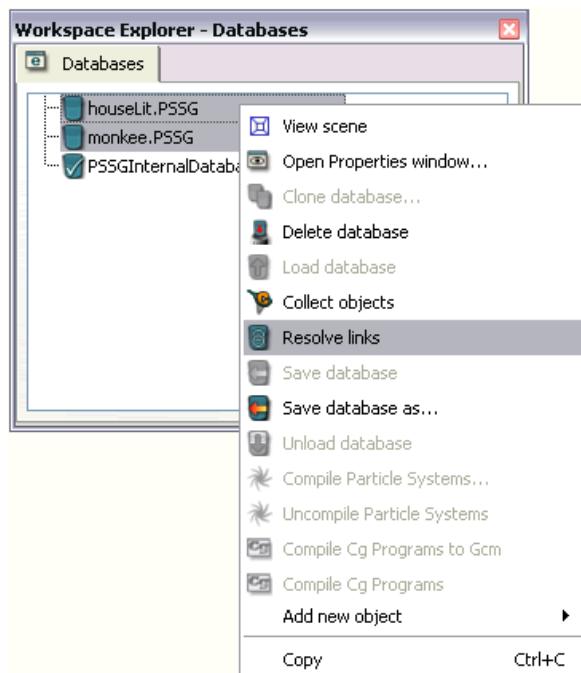
あるデータベースに対して処理を実行するには、最上位データベースを右クリックします。

図 24 具体的なデータベース処理



一度に複数のデータベースに対して処理を実行するには、必要なデータベースを選択した後、その選択領域を右クリックしてコンテキストメニューを表示します。たとえば、図 25 では、2 つのデータベースが単一ステップで選択/解決されています。

図 25 複数の PhyreEngine™ データベースの解決



データベースの処理

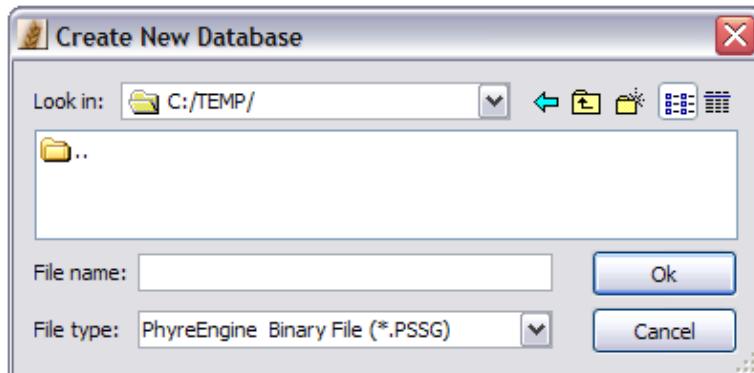
新しいデータベースの作成

新しいデータベースを作成するには、次のいずれかを行います。

- メインメニューから **File > Databases > New database...** を選択します。
- **Workspace Explorer - Databases** ビューの余白部分を右クリックし、コンテキストメニューから **New database...** を選択します。

Create New Database ダイアログボックスが表示されます。新しいデータベースの名前を入力します。

図 26 Create New Database ダイアログボックス



作成されるデータベースの種類は、次のようにファイル名の拡張子で決まります。

- *.PSSG を指定すると、バイナリの PhyreEngine™データベースが作成されます。
- *.hier を指定すると、テキストの PhyreEngine™データベースが作成されます。

たとえば、MyFile.hier は、テキストとして保存された PhyreEngine™形式のデータベースです。拡張子が入力されなかった場合には、デフォルトで*.hier が追加されます。

注意：拡張子では大文字と小文字が区別されません。

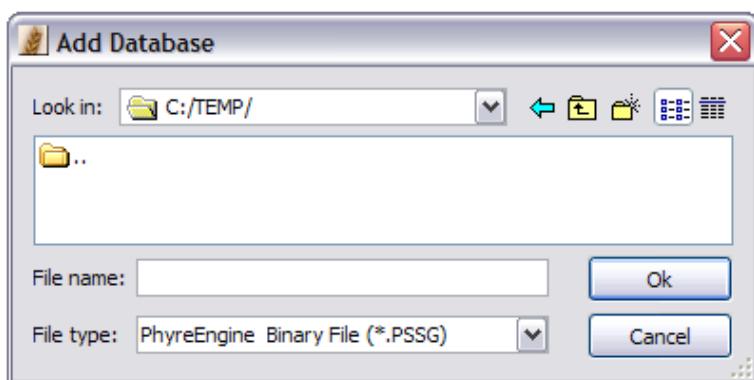
既存のデータベースの追加

既存のデータベースをカレントデータベースに追加するには、次のいずれかを行います。

- PhyreEngine™データベースファイルを、OS のファイルブラウザから PhyreStation のメインウィンドウまたは **Workspace Explorer - Databases** ビューにドラッグします。無効なファイルは警告なしに無視されます。
- **Workspace Explorer - Databases** ビューの余白部分を右クリックし、コンテキストメニューから **Add database...** を選択します。
- メインメニューから **File > Databases > Add database...** を選択します。

最後の 2 つの方法を使用した場合には、**Add Database** ダイアログボックスが表示されます。

図 27 Add Database ダイアログボックス



データベースのリンク解決

ワークスペースに追加されたばかりのデータベースは、その依存関係の解決を PhyreEngine™に依頼する必要があります。データベースのリンクを解決するには、データベースを右クリックし、コンテキストメニューから **Resolve Links** を選択します。

カレントワークスペースに含まれるすべてのロード済みデータベースのリンク解決を一度に行うには、次のいずれかを行います。

- **Workspace Explorer - Databases** ビューの余白部分を右クリックし、コンテキストメニューから **Resolve all databases' links** を選択します。
- メインメニューから **File > Databases > Resolve all databases' links** を選択します。

注意：他の PhyreEngine™データベースへのリンクを一切含まないデータベースは、自動的に解決されます。

ヒント：データベースのロード時にリンク解決を自動的に行うには、**Edit > User Preferences > Workspaces** を選択し、**Auto-resolve databases when loading a new workspace** プリファレンスを設定します。

リンク解決時には、参照先の要素が内部要素、外部要素（つまり、別のデータベース内に存在する要素）のいずれであるかにかかわらず、データベース内のすべての参照先要素に対して包括的な反復処理が行われます。

リンク解決アルゴリズムの実行手順は、次のとおりです。

- カレントデータベースから要求されたが他のデータベース内で見つかったリソースの有効性を確認します。
- すべての内部 PhyreEngine™オブジェクトのリンク解決が行われます。
- 別のオブジェクトに依存するオブジェクトが見つかるたびに、このアルゴリズムは、関連するすべてのリソースが存在しており、かつそれらが取得可能であるかどうかを確認します。
- このアルゴリズムは必要であれば、リンク先リソースが以前の処理によってまだロードされていない場合に、それらのリソースをメモリにロードします。

データベースオブジェクトの収集

選択されたデータベース内のオブジェクトのみをワークスペースに格納するには、そのデータベースを右クリックし、コンテキストメニューから **Collect Objects** を選択します。

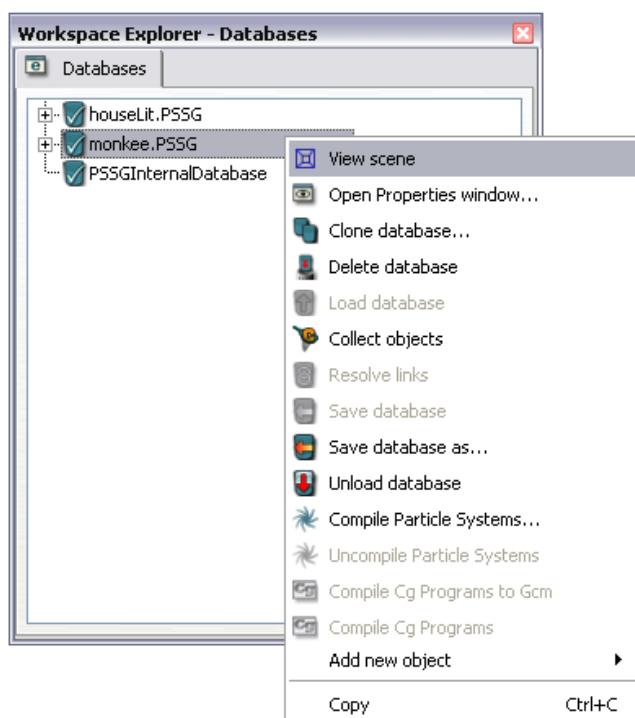
これは、選択されたデータベースとそのすべてのリンク先データベースからオブジェクトを収集する **Resolve Links** に対する代替コマンドです。**Collect Objects** は、データベースがロードさえされていれば、その解決状態にかかわらず使用可能です。

スクリプティングでは、この機能は `DBCollectObjects` スクリプトコマンドによって提供されています。

シーンの表示

データベース内のシーン全体を表示するには、データベースを右クリックし、コンテキストメニューから **View Scene** を選択します。

図 28 シーンの表示



注意：

- **View Scene** メニューのオプションが利用可能になるのは、データベースにシーン（PhyreEngine™ルートノード）が含まれる場合だけです。
- ユーザがモデルの「内側」にいると、シーンが表示されない可能性があります。シーンが表示されるまで、カメラを動かしてズームアウトしてください。

PhyreStation は、カメラオブジェクトや光源オブジェクトがデータベースにすでに含まれているかどうかにかかわらず、カメラノードと光源ノードを表す一時ワークスペース PhyreEngine™オブジェクトを作成します（「一時オブジェクト」（5 PhyreEngine™オブジェクト）を参照）。**Render View** が自動的に開かれ、そこにデータベース内のシーンが、一時的なカメラ/光源オブジェクトを使って表示されます。データベース内の他のオブジェクトは、何の影響も受けません。このコマンドを使用しても、データベースやワークスペースはダーティーとしてマークされません。

データベースの保存

変更された書き込み可能データベースを保存するには、データベースを右クリックし、コンテキストメニューから **Save database** を選択します。

新しいファイル名やファイル名でデータベースを保存するには、データベースを右クリックし、コンテキストメニューから **Save database as...** を選択します。**Save Database As** ダイアログボックスが表示されます。新しい名前でデータベースを保存すると、他のデータベースへのリンクが自動更新されます。

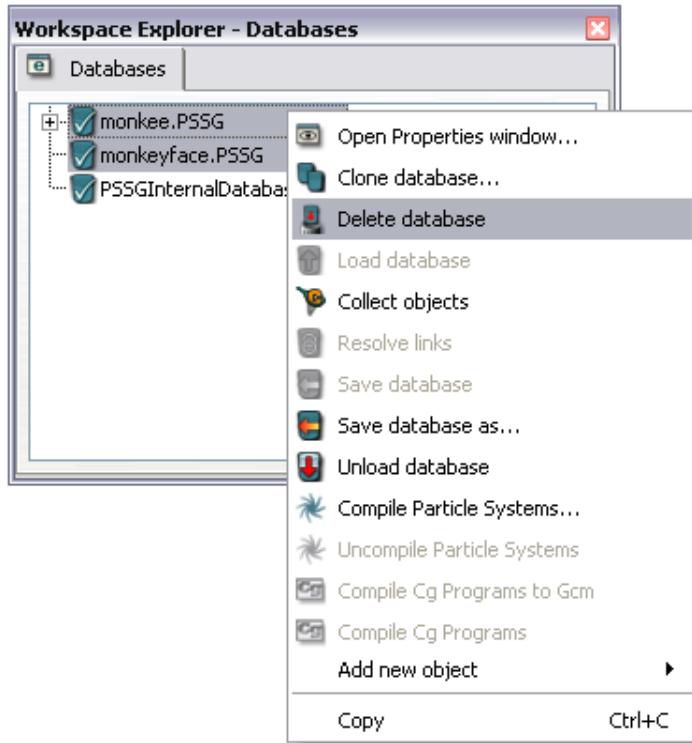
注意：

- PhyreStation は、COLLADA™ファイル形式へのデータのエクスポートをサポートしていません。
- PhyreEngine™の「データ保存」メカニズムの実装方法によって、データベースの名前が変更されないなくても、データベースファイルのサイズやその中身の構造が、ロード時のものとは異なる可能性があります。
- PhyreStation は一時ワークスペース PhyreEngine™オブジェクトを保存しません。

データベースの削除

1つ以上のデータベースを削除するには、データベースを選択してから右クリックし、コンテキストメニューから **Delete database** を選択します。

図 29 複数のデータベースの削除（リンク済み）



削除を確認するダイアログボックスが表示されます。

図 30 データベース削除確認ダイアログ



「has dependants」や「not saved」というメッセージが、削除対象として表示された各データベースの横に表示されます。

- 「Has dependants」は、このデータベースが 1 つ以上の他のデータベースから参照されている（あるいは他のデータベースにリンクされている）ことを意味します。データベースを削除すると、オブジェクトへの参照が切断されます。
- 「Not saved」は、データベースが前回保存されてから行ったデータベースへの変更が失われることを意味します。

注意：削除するデータベースにリンク解決済みデータベース（依存元）が関連付けられていると、その依存元もメモリから削除されます。依存元を持つデータベースを削除する際には注意が必要です。その場合、PhyreEngine™によってそのデータベースがアンロードされ、まだ必要なリソースへのリンクが破棄される可能性があるからで、PhyreEngine™が不安定になる可能性があります。

データベースのアンロード/リロード

他のデータベースから参照されていないデータベースは、一時的にアンロードできます。データベースのアンロードは、システムメモリの空き領域を増やしたい場合やデータベースを再度解決したい場合、あるいはデータベースが壊れている場合に役立ちます。データベースをアンロードするには、データベースを右クリックし、コンテキストメニューから **Unload database** を選択します。

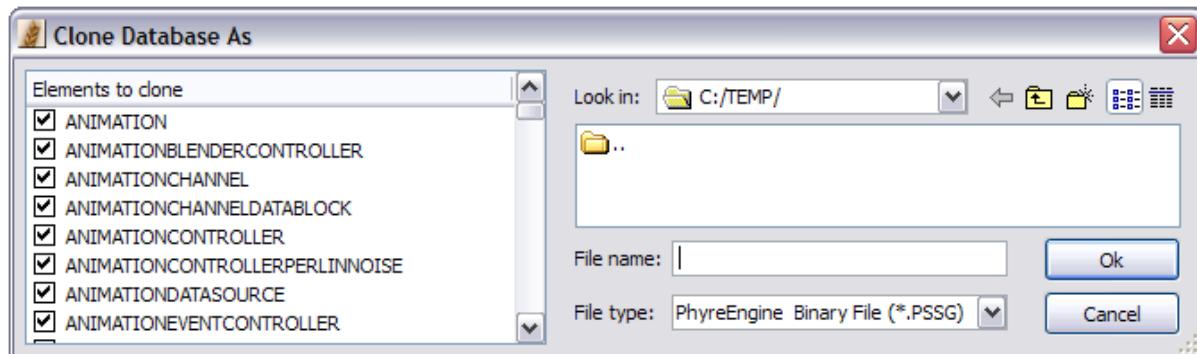
一時的にアンロードされたデータベースを再度ロードするには、データベースを右クリックし、コンテキストメニューから **Load database** を選択します。

データベースの複製

データベースは複製可能ですが、それが行えるのは、データベースが正常に解決されている場合だけです。データベースを複製するには、次の手順に従います。

- (1) データベースを右クリックし、コンテキストメニューから **Clone database...** を選択します。
Clone Database As ダイアログボックスが表示されます。
- (2) 複製する PhyreEngine™オブジェクト型を選択します。

図 31 **Clone Database As** ダイアログボックス



デフォルトでは、すべての PhyreEngine™オブジェクト型が複製対象として選択されます。つまり、データベース全体が正確に複製されます。

データベースの複製処理では、必要に応じてファイルパスを入力できますが、PhyreEngine™データベースファイルは作成されません。このファイルパスは、**Save database** コマンドとスクリプト版のコマンド `SaveDatabase` によって使用されます。

Properties Viewerによるデータベースプロパティの表示

データベースのプロパティを表示するには、データベースを選択し、次のいずれかを行います。

- データベースを右クリックし、コンテキストメニューから **Open Properties window...** を選択します。すると、そのデータベースの静的な **Properties Viewer** が開かれます。
- メインメニューから **Show the Dynamic Property Viewer** ボタンをクリックします。動的な **Properties Viewer** には、現在選択されているオブジェクトのプロパティが表示されます。目的のデータベースを選択すると、そのプロパティが **Properties Viewer** 内に表示されます。

5 PhyreEngine™オブジェクト

概要

PhyreEngine™オブジェクトは、アートアセットまたはデータアセットを表現したものです。すべてのPhyreEngine™オブジェクトは PhyreEngine™クラス PObject から派生したものです。PhyreEngine™オブジェクトはワークスペースオブジェクトのサブセットです。

PhyreStation では、カレントデータベース内で PhyreEngine™オブジェクトのブラウズ、作成、編集、複製、移動、および削除が行えます。

サポートされているすべての PhyreEngine™オブジェクト型の一覧を表示するには、メインメニューから **Help > Help Contents** を選択した後、**Registered PhyreStation Types** をクリックします。利用可能なオブジェクト型の一覧を表示する別 の方法として、**Help > About PhyreStation** を選択し、PhyreStationDLL コンポーネントを選択する方法があります。

カスタムオブジェクト型を作成した後、それらの型のオブジェクトを操作するカスタムコマンドを作成することができます。PhyreStationDLL でカスタムオブジェクト型とカスタムコマンドを登録するまで、それらを使用することはできません。詳細については、「8 PhyreStation のカスタマイズ」を参照してください。

一時オブジェクト

PhyreStation は、ユーザが **View Scene** データベースコマンドを選択した場合などの特定の状況では、一時オブジェクトを作成してある機能を実行する場合があります。「シーンの表示」(4 データベース) を参照してください。

一時オブジェクトは PhyreEngine™オブジェクトです。これはデータベース内に存在し、ワークスペース内では他のすべてのオブジェクトとまったく同様に表現されます。ただし、一時 PhyreEngine™オブジェクトは、任意の Save、SaveAs、または Clone Database コマンドが実行される直前にデータベースから削除され、それらのコマンドの完了後に再度挿入されます。名前が _PhyreStationPSSGTmpObj で始まっているれば、一時オブジェクトであると判断できます。

注意：

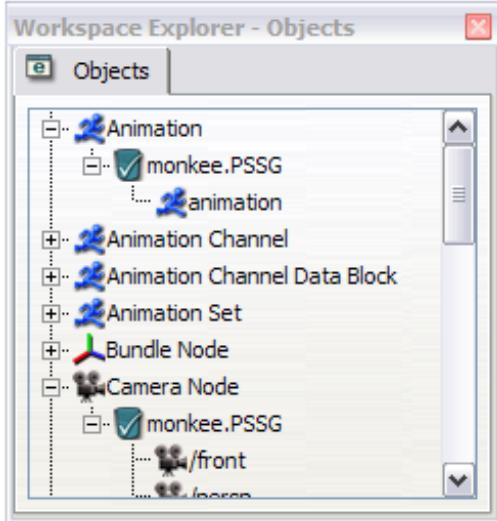
- 一時 PhyreEngine™オブジェクトはデータベースのオブジェクトカウントに加算されます。PhyreStationDLL の視点から見ると、それらは名前以外は、他の PhyreEngine™オブジェクトと区別が付きません。
- PhyreStationDLL 側からのコマンドまたはアクションが一時 PhyreEngine™オブジェクトを正しく処理しなければ、データベースが汚染される可能性があります。一時オブジェクトを削除するには、スクリプトコマンド RemoveUnusedObjects() か、一時オブジェクトを識別できるコードを使用します。一時 PhyreEngine™オブジェクトは、削除しても必要になればまた自動的に作成されます。
- PhyreStation の GUI 環境内からは、PhyreStationDLL コマンドを使って一時 PhyreEngine™オブジェクトを処理することはできません。

オブジェクトのブラウズ

PhyreEngine™オブジェクトにアクセスするためには、それらのオブジェクトがワークスペース内に存在している必要があります。つまり、1つ以上の PhyreEngine™オブジェクトを含むデータベースがワークスペース内に含まれており、そのデータベースが解決済み、収集済みのいずれかの状態になっている必要があります。

PhyreEngine™オブジェクトをブラウズするには、ツールバーの「Workspace Explorer」を開いた後、その **Workspace Explorer** で右クリックし、コンテキストメニューから **Objects** を選択します。
Workspace Explorer - Objects ビューには、ワークスペース内のすべてのリンク解消済み PhyreEngine™データベースに含まれるすべての PhyreEngine™オブジェクトが表示されます。

図 32 **Workspace Explorer - Objects** ビュー



デフォルトでは、オブジェクトはまず PhyreEngine™オブジェクト型ごとにグループ化され、次にデータベースごとにグループ化されます。オブジェクトが 1 つも含まれないカテゴリを含め、すべてのカテゴリを表示するには、ビューのコンテキストメニューから **Show Empty Categories** を選択します。オブジェクトがデータベースごとにグループ化された後で PhyreEngine™オブジェクト型ごとにグループ化されるようにビューを変更するには、ビューのコンテキストメニューから **View By Database** を選択します。

Workspace Explorer では、あるシングラフやカスタムグループの一部となっている PhyreEngine™オブジェクトを表示することもできますが、それには **Workspace Explorer** コンテキストメニューから **Scene-Graph**、**Custom Groups** のいずれかを選択します。

ビューア/エディタによるオブジェクトへのアクセス

1 つ前のセクションで説明したように、**Workspace Explorer - Objects** ビューを使えばすべての PhyreEngine™オブジェクトにアクセスできます。これに加え、一部の PhyreEngine™オブジェクト型に備わる特殊な属性を表示/編集するには、ビューアやエディタと呼ばれる専用の PhyreStation ウィンドウを使用します。

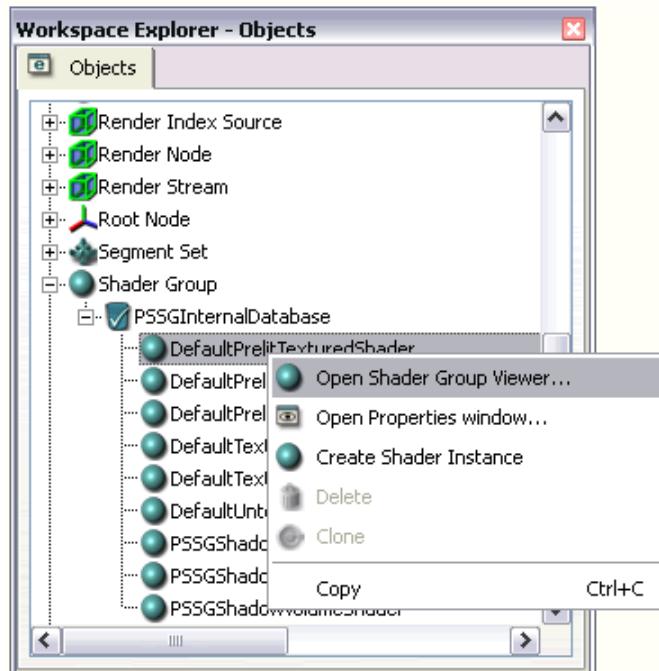
注意：

- **Properties Viewer** を使ってオブジェクトのプロパティを表示/編集することもできます。この章で後述する「オブジェクトの編集」を参照してください。
- 一部の PhyreEngine™オブジェクトについては、PhyreEngine™がある処理の実行時にそれらが必要だと判断するまで、ロックされない可能性があります。例としては、PhyreEngine™ Animation Set が挙げられます。ユーザはこのオブジェクトを削除できますが（非推薦）、削除できるのはアニメーションが再生されるまでです。
- 一部の PhyreEngine™オブジェクトは、それらのオブジェクトを作成しないと処理を継続できないと PhyreEngine™が判断するまで、存在しない可能性があります。

オブジェクト	ビューア/エディタ	参照先
カメラノード オブジェクト	Render Viewer	「Render Viewer」(11 ビューア) を参照
セグメントセット オブジェクト	Segment Set Viewer	「Segment Set Viewer」(11 ビューア) を参照
シェーダインスタンス オブジェクト	Shader Instance Viewer または Shader Instances Viewer	「Shader Instance Viewer/Shader Instances Viewer」(11 ビューア) を参照
テクスチャオブジェクト	Texture Viewer	「Texture Viewer」(11 ビューア) を参照
シェーダプログラム オブジェクト	Shader Program Viewer	「Shader Program Viewer」(11 ビューア) を参照
シェーダグループ オブジェクト	Shader Group Viewer	「Shader Group Viewer」(11 ビューア) を参照
アニメーションターゲット トレンダオブジェクト	Graph Editor	「Target Blender Editor」(12 Graph Editor) を参照
モディファイアネットワ ークオブジェクト	Graph Editor	「Modifier Network Editor」(12 Graph Editor) を参照
モディファイアネットワ ークインスタンスオブジ ェクト	Graph Editor	「Modifier Network Instance Editor」 (12 Graph Editor) を参照
パーティクルエミッタ ノードオブジェクト	Particle Editor	「14 Particle Editor」を参照

上記の特殊オブジェクトのいずれかに対する専用のビューアまたはエディタを表示するには、**Workspace Explorer - Objects** ビューでそのオブジェクトを右クリックし、コンテキストメニューから **Open <オブ
ジェクト型> Viewer** または **Open <オブジェクト型> Editor** を選択します。

図 33 オブジェクトビューアの表示



オブジェクト処理の実行

オブジェクト処理の実行は、選択されたオブジェクト（複数可）に対してコマンドを実行することを意味します。

- あるオブジェクトに対して処理を実行するには、**Workspace Explorer - Objects** ビューでそのオブジェクトを右クリックし、コンテキストメニューから処理を選択します。
- コンテキストメニュー上の処理は、PhyreStation の内部コマンド、サードパーティによって開発され、PhyreStationDLL で登録されたコマンドのいずれかになります。
- 処理は、複数のオブジェクトに対して実行できます。複数のオブジェクトは、1つのウィンドウから選択することも、複数のウィンドウから選択することもできます。スクリプトを実行することによっても、オブジェクトに対する処理を実行できます。
- ウィンドウ間でオブジェクトをドラッグ&ドロップできます。

ヒント：

- ドラッグ&ドロップ操作やコンテキストメニュー命令が、複数選択のすべてのオブジェクトに対して有効であるとは限りません。**Log** ウィンドウで処理が成功したか確認してください。同じ型のオブジェクトに対してオブジェクト固有のコマンドを呼び出すことをお勧めします。
- Shift** キーを押しながらツリー階層内の一連のオブジェクト（**Scene-Graph** ビュー内のノードなど）を選択すると、折りたたまれた（つまり非表示の）子ノードもすべて選択されます。選択されたオブジェクトに対して処理を実行すると、その非表示の子ノードに対しても処理が実行されます。これを避けるには、折りたたまれたすべてのノードを展開し、**Ctrl** キーで子ノードの選択を解除してください。
- Shift** キーによる選択中に、折りたたまれたツリー表示項目を選択した場合、その折りたたまれた項目の中に隠れたすべての子項目を選択するためにその下にある次項目を選択した後、その次項目の選択を解除します。そうしないと、ハイライト表示された項目だけが選択され、非表示の子項目が選択されません。

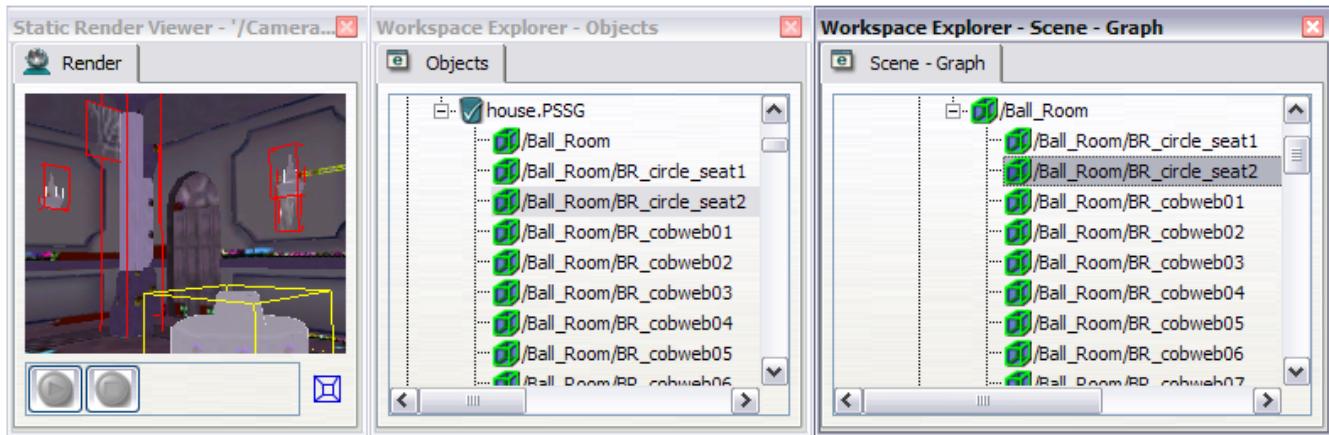
複数のウィンドウからの選択

あるウィンドウ内に表示されているオブジェクトを選択すると、そのオブジェクトを含む他の任意のウィンドウでも、たとえそのオブジェクトが現在可視でなくなっていても、そのオブジェクトが選択済みとして自動的にハイライト表示されます。

オブジェクトのハイライト表示は次のようになります。

- 青色：アクティブウィンドウ（フォーカスを持つウィンドウ）内の選択項目を示します。
- 灰色：他のウィンドウ（複数可）内の選択項目を示します。
- 淡青色：非表示の選択項目（複数可）を持つ親項目を示します。
- 黄色：**Render** ビュー内の選択項目（バウンディングボックス）を示します。

図 34 Render Viewer からの選択



注意：選択された項目が可視でない場合、その項目に最も近い可視の親が淡青色でハイライト表示されます。

ヒント：

- 多数の項目を選択する場合には、不要な **Workspace Explorer** をすべて閉じてください。そうすれば、項目の選択や選択解除が高速になります。PhyreStation が更新すべきビューの数が減るからです。
- Render** ビューで選択されたオブジェクトを表示するには、コンテキストメニューの **Bounding Boxes** または **Highlight Selection** オプションを有効にします。このビューにはハイライト表示の色は適用されません。

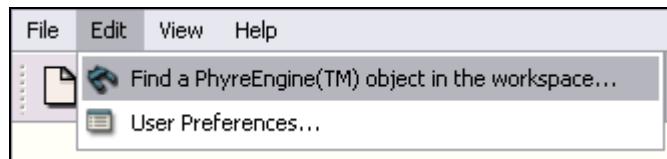
オブジェクトの処理

ワークスペース内のオブジェクトの検索

ワークスペースには、数千もの PhyreEngine™ オブジェクトが含まれている場合があります。Find ダイアログを使えば、カレントワークスペース内で検索を行い、特定の検索条件に一致する PhyreEngine™ オブジェクトの一覧を表示することができます。

Find ダイアログを表示するには、メインメニューから **Edit > Find a PhyreEngine(TM) object in the workspace...** を選択するか、メインツールバーの **Find** ボタン  をクリックします。

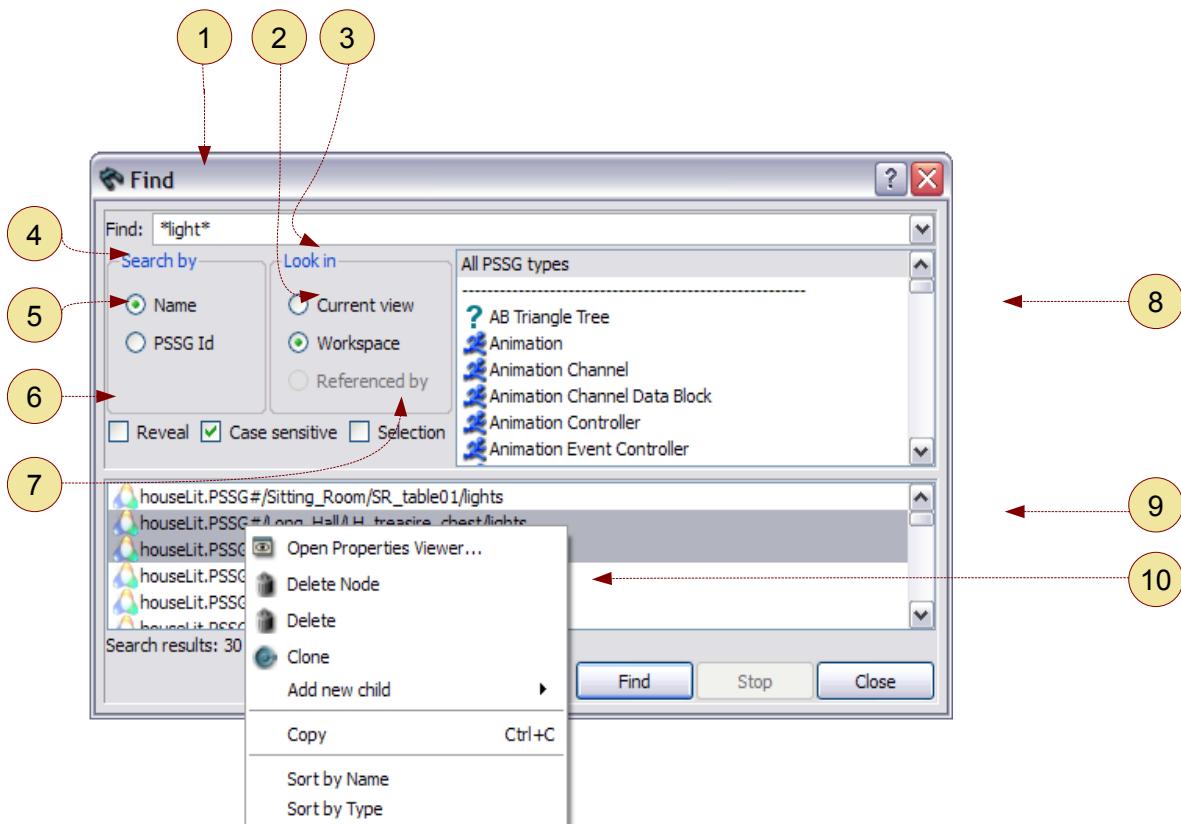
図 35 Find ダイアログの表示



Find ダイアログから次のいずれかを行えます。

- オブジェクトを表示し、**Workspace Explorer** ウィンドウ内で該当するオブジェクトを展開/配置し、ユーザから見えるようにします（項目 6）。
- コンテキストメニューを表示し、検索結果の全部または一部に対して処理を実行する（項目 10）。

図 36 Find ダイアログボックス



- (1) テキスト検索条件を入力するか、任意の **Workspace Explorer** から編集行に PhyreEngine™オブジェクトをドラッグ&ドロップします。あるいは編集行で右クリックし、メニューから **Paste from views...** を選択します。続いて、テキストを編集して似た名前のオブジェクトを検索できます。ワイルドカード文字を使用できます。
- (2) ワークスペース内のすべての PhyreEngine™オブジェクトを検索します。
- (3) フォーカスのあるカレントウィンドウ内のオブジェクトのみを検索します。
- (4) ワークスペースでの検索を、**Workspace Explorer** ウィンドウに表示される PhyreEngine™オブジェクト名に基づいて行います。これにはデータベース名は含まれません。
- (5) ワークスペースでの検索を、PhyreEngine™オブジェクトリンク ID (*databaseName#objectName*)に基づいて行います。
- (6) 検索で見つかったワークスペースオブジェクトを、開かれているすべての **Workspace Explorer** ウィンドウ内で表示します。
- (7) 検索結果一覧（項目 9）から選択された項目を、ワークスペース全体のオブジェクト選択に追加します。
- (8) 検索をさらに絞り込み、選択された PhyreEngine™オブジェクト型だけを検索するようにします。
- (9) 最新の検索条件（項目 1）で見つかった PhyreEngine™オブジェクトの一覧。
- (10) 検索結果の全部または一部に対するコマンドコンテキストメニュー。

オブジェクトの検索を行うには、次の手順に従います。

- (1) **Find** ボックス（項目 1）にテキスト検索条件を入力します。検索条件は、PhyreEngine™オブジェクト名でも、オブジェクト名と検索キー記号を組み合わせたものでもかまいません。

- (2) **Find** ボタンをクリックして検索を開始します。検索条件に一致するオブジェクトの一覧と、実際に見つかった項目の数が表示されます（項目 9）。表示された項目からさらに選択を絞り込むことができます。選択範囲で右クリックし、オブジェクトのコンテキストメニュー（項目 10）を表示します。

Find ダイアログは最新の検索条件テキストを履歴バッファに自動的に格納し、そのテキストを後で再利用できるようにします。

注意：コンテキストメニューには、選択されたすべての PhyreEngine™オブジェクト型に対するコマンドの一覧が表示されます。したがって、コンテキストメニューから選択したコマンドが一部の PhyreEngine™オブジェクトで失敗する可能性があります（そうしたオブジェクトにコマンドが対応していないため）。

高度な検索

結果のフィルタリングを行うより高度な方法として、表 2 の記号を使って式を作成する方法があります。

表 2 検索記号

*	これは、0 個以上の任意の文字に一致します。
?	これは、任意の単一文字に一致します。
[...]	角括弧内では、コンマで区切られた一連の文字を表現できます。

表 3 に、上記の記号を用いた式の例を、いくつか示します。ワークスペースに次の 3 つのオブジェクトが含まれているとします。Texture1（データベース DB1）、texture2 と _textureOld（データベース DB2）。

表 3 検索記号を用いた式

ラベルによる検索	
exture	ワークスペース内のすべてのテクスチャを返します。
?texture?	DB1#Texture1 と DB2#texture2 を返します。
*	ワークスペース内のすべてのオブジェクトを返します。
PhyreEngine™オブジェクト ID によるワークスペース検索	
*#[T,t]exture?	DB1#Texture1 と DB2#texture2 を返します。
DB?#*	2 つのデータベース内のすべてのオブジェクトを返します。
DB?#[T,t]exture[1-2]	DB1#Texture1 と DB2#texture2 を返します。

式の長さに制限はありません。余分な空白や連続した「*」記号はすべて無視されます。单一の「*」を指定すると、すべてのオブジェクトが取得されます。

オブジェクトの作成

PhyreStation では次のようにさまざまな方法で、新しい PhyreEngine™オブジェクトを作成できます。

- データベースに新しいオブジェクトを追加するには、データベースのコンテキストメニューから **Add New Object** を選択します。
- テクスチャファイルなどの外部オブジェクトファイルを PhyreStation 内にドラッグ & ドロップします。（この方法はすべてのオブジェクト型で使えるわけではありません。適切な PhyreStationDLL ファイルハンドラが存在している必要があります）。
- ノードに新しい子ノードを追加するには、**Workspace Explorer - Scene-Graph** ビューでノードのコンテキストメニューから **Add New Child** を選択します。
- パーティクルシステムを追加するには、選択されたデータベース内でパーティクルエミッタノードを作成した後、そのノードのコンテキストメニューから **Create Particle System** を選択します。

- NewObject コマンドを **Command Window** から実行するか、スクリプト内から実行します。

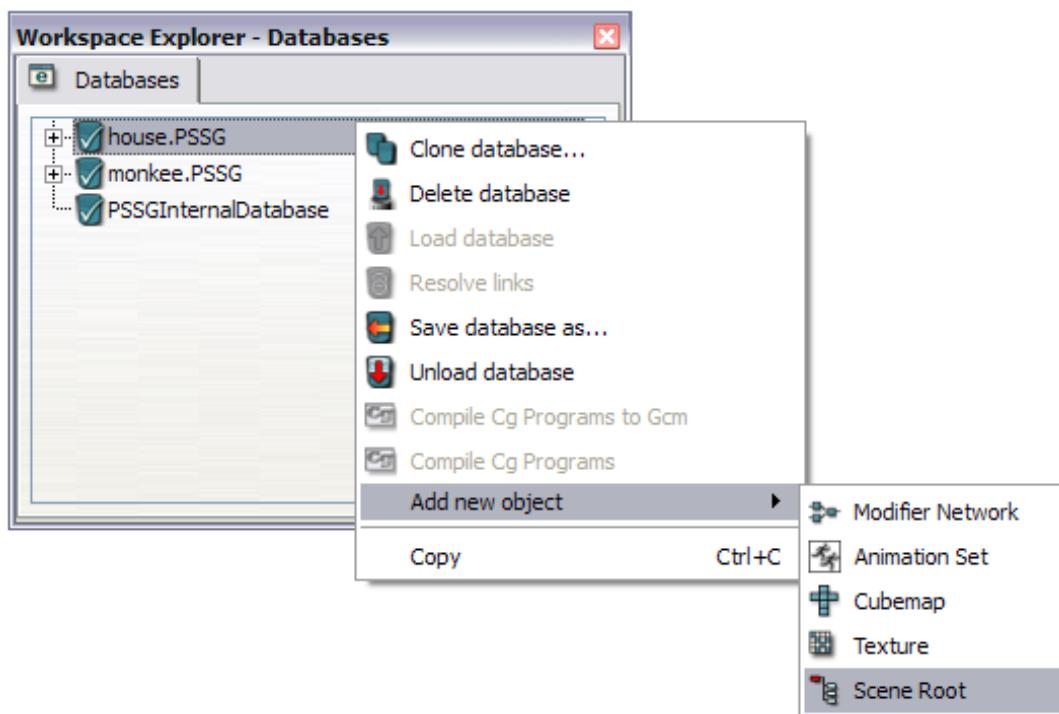
最初の 2 つの方法について、以下で説明します。

データベースへのオブジェクトの追加

新しい PhyreEngine™オブジェクトを作成し、それをデータベースに追加するには、次のいずれかを行います。

- 別のデータベース内のオブジェクトを選択し、それを目的のデータベースにドラッグした後、コンテキストメニューから **Clone** を選択します。「オブジェクトの複製」(5 PhyreEngine™オブジェクト) を参照してください。
- **Workspace Explorer** でデータベースを右クリックした後、コンテキストメニューから **Add new object** > [オブジェクト型] を選択します。

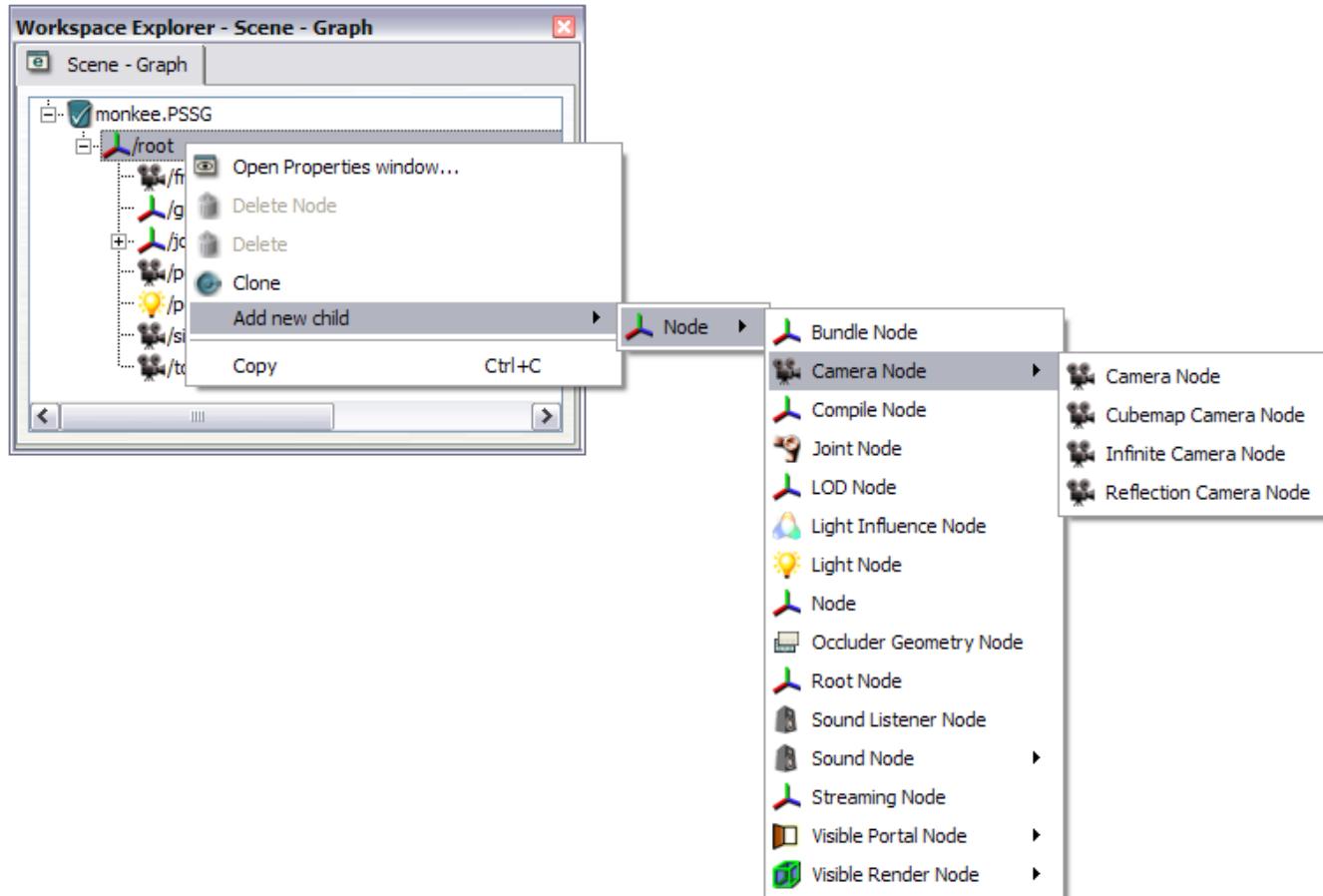
図 37 データベースへの新しい PhyreEngine™オブジェクトの追加



シーングラフへのオブジェクト追加

シーングラフに新しいノードオブジェクトを追加するには、**Workspace Explorer - Scene-Graph** ビューで親ノードのアイコンを右クリックした後、コンテキストメニューから **Add new child > Node > [ノードタイプ]** を選択します。

図 38 新しい PhyreEngine™ノードオブジェクトの追加

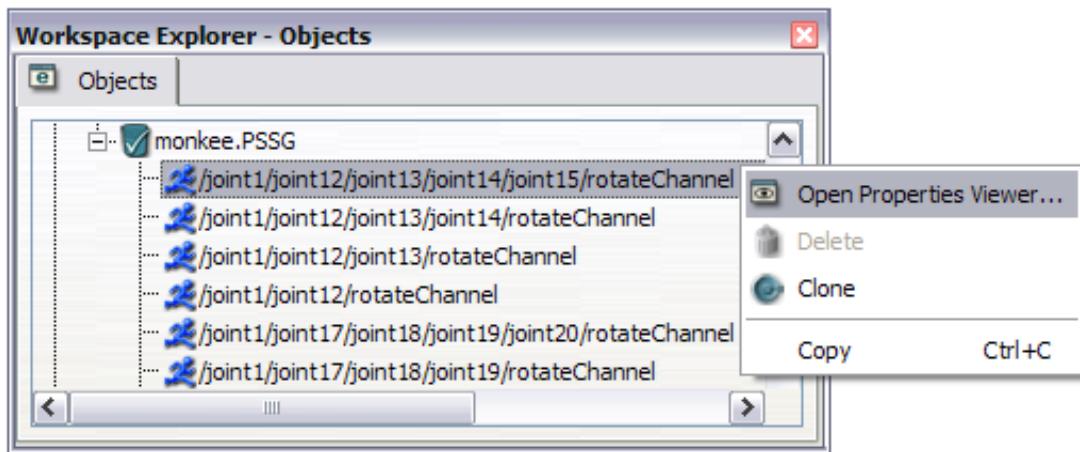


オブジェクトの編集

PhyreEngine™オブジェクトのプロパティを表示/編集するには、任意の **Workspace Explorer - Objects** フィルタビューまたは **Workspace Explorer - Scene-Graph** フィルタビューでオブジェクトを選択し、次のいずれかを行います。

- オブジェクトを右クリックし、コンテキストメニューから **Open Properties Viewer...** を選択します。すると、そのオブジェクトの静的な **Properties Viewer** が開かれます。
- メインツールバーの **Show the Dynamic Property Viewer** ボタンをクリックします。動的な **Properties Viewer** には、現在選択されているオブジェクトのプロパティが表示されます。目的のオブジェクトを選択すると、そのプロパティが **Properties Viewer** 内に表示されます。

図 39 Properties Viewer ウィンドウ（静的）のオープン

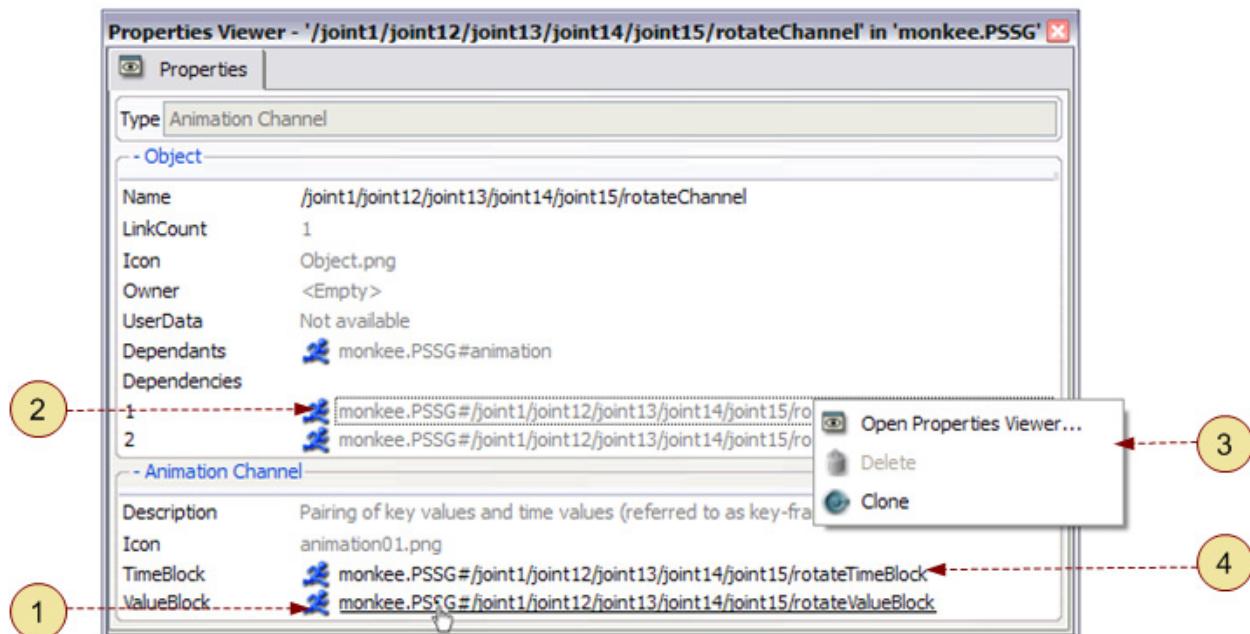


動的モードと静的モードについては、この章で後述する「動的モードと静的モード」を参照してください。PhyreEngine™オブジェクトのプロパティを編集する別 の方法として、オブジェクトを直接変更するか関連する他の何らかの変更を通じてオブジェクトを変更するスクリプトを実行する、という方法もあります。あるオブジェクトの1つ以上の属性が変更されると、そのオブジェクト、そのオブジェクトが属するデータベース、およびカレントワークスペースがデーターとしてマークされます。変更を確定するにはデータベースを保存する必要があります。

オブジェクトの Properties Viewer

Properties Viewer には、あるオブジェクトの属性の値が表示されます。一部の属性の値は、新しい値を入力するかオプションリストから選択することにより、直接編集することができます。

図 40 Properties Viewer ウィンドウ



- (1) ある PhyreEngine™オブジェクトにカーソルを移動すると、そのオブジェクトがハイパーリンクとして表示されます。そのリンクを1回クリックすると、**Properties Viewer** の表示が変更され、そのリンクで指定されたオブジェクトが表示されます。
- (2) テーブル内の灰色の項目は無効化されています。それらを編集/変更することはできません。

- (3) 子 PhyreEngine™オブジェクトを右クリックすると、コンテキストメニューが表示されます。
- (4) PhyreEngine™オブジェクト名。これは、このビューではセカンダリオブジェクトになります。

ヒント：不要な **Properties Viewer** ウィンドウはすべて閉じることをお勧めします。PhyreStation は、これらのウィンドウの表示を最新に保つためにリソースを消費するので、反応が遅くなる場合があります。

Properties Viewer に表示されるプロパティのいくつかは、他の PhyreEngine™オブジェクトへの参照である可能性があります（図 40 を参照）。**Properties Viewer** では、これらのセカンダリ PhyreEngine™オブジェクトに対してコマンドを選択して実行できます。

PhyreEngine™オブジェクトに対して表示されるプロパティは、現在の PhyreEngine™オブジェクトのプロパティのクラス/静的関数によって制御されます。これらのクラス/関数は、PhyreEngine™ユーティリティの Extra ソースコード内で PhyreStationDLL とともにコンパイルされます。これはカスタマイズ可能です。PhyreEngine™オブジェクトの属性にアクセスするには、`SetObjectAttribute` 関数と `GetObjectAttribute` 関数を使用します。属性フィールドが利用できないか、PhyreStationDLL でエラーが発生した場合、PhyreStation はパラメータの代わりに「Unhandled PhyreEngine Attribute type」と表示します。

動的モードと静的モード

Properties Viewer ウィンドウには「静的」モードと「動的」モードという、2つの動作モードがあります。静的モードでは、ユーザが後でワークスペース内の他のオブジェクトを選択した場合でも、ビューの対象は固定されたままです。動的モードのビューは、ユーザが選択した最新のオブジェクトを表示するように変更されます。

この2つのモードを区別できるように、タブの背景色が動作モードに応じて変化します。灰色またはニュートラルな背景色は、ウィンドウが「静的」モードで動作していることを示します。青色の背景色は、ウィンドウが「動的」モードで動作していることを示します。

「動的」モード版を表示するには、ツールバーの **Properties Viewer** ウィンドウアイコンをクリックします。「静的」モード版を表示するには、オブジェクトのコンテキストメニューから **Open Properties Viewer...** オプションを選択します。

オブジェクトプロパティの履歴ナビゲーション

Properties Viewer では、対象の **History Navigation ツールバー**（「ウィンドウの履歴」（15 GUI）を参照）を使用できます。これは、親子関係を持つオブジェクトや他のオブジェクトから参照されているオブジェクトのナビゲーションを行う際に役立つ可能性があります。履歴バッファの動作は、ビューのモードごとに異なります。

- 動的モードでは、ツールバーは異なる2つの履歴バッファ（リスト）を操作します。一方のリストには、ウィンドウのオープン後に選択されたすべてのオブジェクトが表示されます。他方のリストには、親子関係内を移動することでナビゲートされたオブジェクト（図 38 の項目 4）が記録されます。
- 静的モードでは、履歴機能は通常無効になっています。というのも、通常は1つのオブジェクトしか表示されないからです。ただし、静的ビューアにオブジェクトをドラッグ＆ドロップしてその対象を変更すると、履歴機能が有効になります。

オブジェクトの属性値の変更

オブジェクトの一部の属性は編集できる可能性があります。ユーザがあるオブジェクトの属性の値を編集しても、その新しい値がオブジェクトの文脈内で有効でないと PhyreEngine™が判断した場合には、

PhyreEngine™はその新しい値を設定せず、**Properties Viewer** の値は編集前の値に戻されます。処理のステータスマッセージが **Log** ウィンドウ内に表示される可能性があります。

図 40 の項目 4 は、**Properties Viewer** の対象となっている PhyreEngine™オブジェクトが参照する子 PhyreEngine™オブジェクトを表しています。この子オブジェクトの参照を変更するには、**Workspace Explorer** で別の PhyreEngine™オブジェクトを選択し、それをビュー内の子オブジェクトの上にドラッグします。PhyreEngine™によって変更が検証されます。

参照されている PhyreEngine™オブジェクトの更新

Properties Viewer (動的または静的) で PhyreEngine™オブジェクトのプロパティを変更すると、その変更されたプライマリオブジェクトに依存するセカンダリオブジェクトの状態にも影響が及ぼします。理想的には、プライマリオブジェクトが変更されるたびに、セカンダリオブジェクトのすべてのウィンドウも自動的に更新されるべきです。ところが、PhyreStation はデフォルトではこれを行いません。なぜなら、他のウィンドウを更新するとツールのパフォーマンスが低下するからです。この機能を有効にするには、メインメニューから **Edit > User Preferences...** を選択した後、**PhyreStation** タブを選択し、**PhyreEngine™ objects referenced by other PhyreEngine™ objects** プリファレンスを設定します。

注意 : PhyreStation がセカンダリオブジェクトのウィンドウを更新するために必要とする PhyreEngine™オブジェクト間の関係情報は、PhyreEngine™データベースのロード/解決時に収集されます。したがって、これらの処理の実行時間が長くなります。

データベースの統計情報の表示

Properties Viewer はオブジェクトの属性を反映しますが、それと同様に各データベースの状態も反映します。**Properties Viewer** で利用可能なデータベース属性（統計情報）は、現在の PhyreEngine™データベースのプロパティのクラス/静的関数によって決まります。これらのクラス/関数は、PhyreEngine™ユーティリティの Extra ソースコード内で PhyreStationDLL とともにコンパイルされます。これは、ユーザーの要件に応じてカスタマイズできます。

オブジェクトの複製

1 つ以上の PhyreEngine™オブジェクトを複製するには、**Workspace Explorer - Objects** ビューでオブジェクトを右クリックし、コンテキストメニューから **Clone** を選択します。選択したオブジェクトのコピーが作成され、そのコピーから新しいオブジェクトが作成されます。

ヒント : データベース内の全部または一部のオブジェクトを複製する方法については、「データベースの複製」(4 データベース) を参照してください。

オブジェクトの移動または削除

1 つ以上の PhyreEngine™オブジェクトを削除するには、次のいずれかを行います。

- 選択されたオブジェクトを右クリックし、コンテキストメニューから必要な削除アクションを選択します。別の PhyreEngine™オブジェクトから参照されているオブジェクトが 1 つ以上存在する場合、それらのオブジェクトは削除できない可能性があります。
- あるデータベースから別のデータベースにオブジェクトをドラッグします。すると、オブジェクトが移動元データベースから削除され、移動先データベースで再作成されます。

オブジェクトを移動するには、次のいずれかを行います。

- オブジェクトを移動先のデータベースにドロップし、コンテキストメニューから **Move** を選択します。別の PhyreEngine™オブジェクトから参照されているオブジェクトでは、**Move** 処理は利用できません。PhyreEngine™ PNode 型のオブジェクトは移動できません。

- オブジェクトを選択し、別のデータベースにドラッグします。
- ノードを選択し、新しい親ノードまたはデータベースにドラッグします。
- オブジェクトのディープクローンを実行した後、元のオブジェクトを削除します。

移動は、削除処理と追加処理を組み合わせたものです。オブジェクトを追加する際、その属性の中には、値が同じままのものもあれば、値が変わる（新しい親オブジェクトなど、他のオブジェクトへの再参照に変更される）ものもあります。このオブジェクトへのリンクが存在している場合、GUI から移動が行えない可能性があります。ただしその場合でも、スクリプトコマンドを使えば移動を実行できます。

ヒント : **Properties Viewer** を使って対象 PhyreEngine™オブジェクトの **Link Count** 属性を表示します。この属性がゼロでない場合、その PhyreEngine™オブジェクトは別のオブジェクトから参照されているため、GUI から削除できない可能性があります。その場合、スクリプトコマンド `DeleteObject()` を使えばオブジェクトを削除できます。

6 PhyreStationコマンド

この章では、各種 PhyreStation コマンドとそれらのさまざまな実行方法について説明します。

概要

PhyreStation で実行されるすべての主要な処理には、コマンドの実行が関係しています。PhyreStation コマンドの定義元は、PhyreStationDLL コンポーネントか、PhyreStation 自体の内部かのいずれかです。ほとんどの場合、コマンドは 1 つのタスクのみを実行します。しかしながら、コマンドから他のコマンドを呼び出したり、複数のオブジェクトに対してコマンドを実行したりすることもできます（ただし、後者は GUI から実行する場合のみ）。

このコマンドシステムは、ディベロッパが PhyreStation を新しいコマンドで拡張できるように設計されています。新しいコマンドは、PhyreEngine™ユーザユーティリティで開発するか、あるいは PhyreStationDLL 内に配置します（前者はユーザ独自の作業用、後者は個々のスタンダードアロンコマンド用）。

PhyreStation 内部コマンド

PhyreStation には少數の内部コマンドが用意されています。これにはすべてのワークスペースコマンド (OpenWorkspace() など) と多数のデータベースコマンド (ResolveLinks() など) が含まれます。PhyreStation のソースコードは提供されていないので、PhyreStation 内部コマンドは変更できません。

PhyreStationDLL コマンド

PhyreStation のコマンドの大部分は PhyreStationDLL 内で定義されたものであり、主に PhyreEngine™ オブジェクトに対する処理を行います。これらのコマンドを呼び出すには、GUI でオブジェクトを選択してからアクションを選択するか、あるいはスクリプト内からコマンドを発行します。典型的な例として、CloneObject() コマンドが挙げられます。

クライアントは PhyreStation を新しいコマンドでカスタマイズし、それによって PhyreStation のコマンドベースを拡張することができます。カスタムコマンドは、PhyreStation に登録しないと使用できません。詳細については、「8 PhyreStation のカスタマイズ」を参照してください。

利用可能なコマンドの表示

利用可能なすべてのコマンドとその構文を表示するには、メインメニューから **Help > Help Contents...** を選択した後、**Registered Commands** をクリックします。

詳細については、「Help Viewer」（1 PhyreStation の概要）を参照してください。

コマンドの実行方法

PhyreStation コマンドの実行方法を以下に示します。

- PhyreStationDLL に登録されたスクリプトコマンドを **Command Window** に入力します。後述の「Command Window」を参照してください。
- メインウィンドウのメニューバーからコマンドを選択します（例：**File > Databases > New database**）。
- メインウィンドウのツールバー上のボタン（**New Workspace** ボタンなど）をクリックします。

- **Workspace Explorer** 内のワークスペースオブジェクト（または余白部分）を右クリックした後、コンテキストメニューからコマンド（**Delete** など）を選択します。
- **Workspace Explorer** 内でオブジェクトをドラッグ&ドロップした後、ポップアップメニューからコマンド（**Clone** など）を選択します。
- スクリプトファイルを実行するために、**Workspace Explorer - Scripts** ビューでスクリプトオブジェクトを選択し、コンテキストメニューから **Run** を選択します。スクリプトの詳細については、「7 スクリプト」を参照してください。

コマンドの中には、上記の 1 つの方法でしか実行できないものもあれば、複数の方法で実行できるものもあります。たとえば、`NewWorkspace()` はスクリプトコマンドなので、**Command Window** から実行できるほか、メインメニュー やツールバーからも起動できます。

Command Window

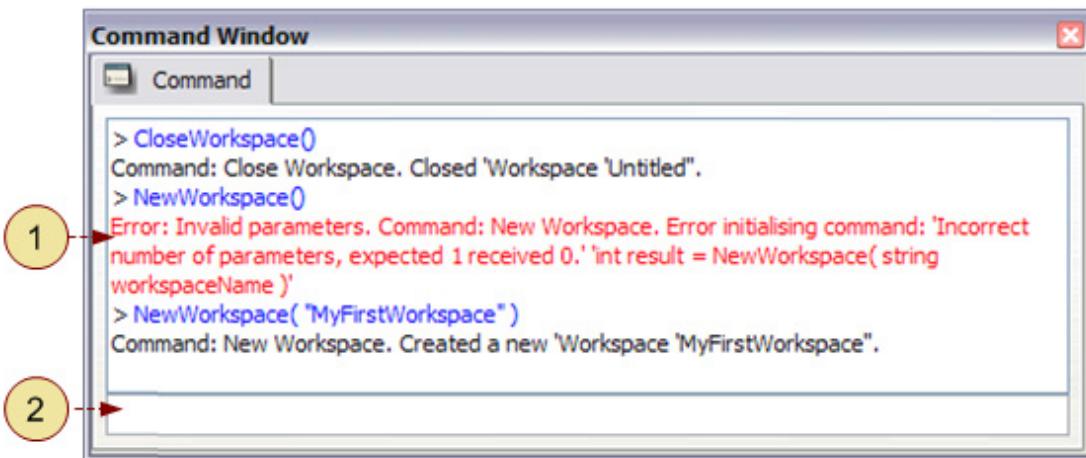
概要

Command Window は、スクリプトコマンドを入力/実行する際に使用します。これらのコマンドは、PhyreStationDLL 内でスクリプト可能コマンドとして明示的に登録されたものです。スクリプト可能コマンドの作成方法の詳細については、「PhyreStation への PhyreEngine™ オブジェクトコマンドの追加」（8 PhyreStation のカスタマイズ）を参照してください。一般的なスクリプティングについては、「7 スクリプト」を参照してください。

Command Window では、PhyreStation の GUI から利用できないコマンドや、ブロックされているコマンド（GUI のコンテキストメニュー上で無効化されているコマンド）を実行することができます。**Command Window** からコマンドを実行する場合、多機能で制限も少なくなりますが、一部の処理は特定の理由があるためブロックされています。したがって、PhyreEngine™ の安定性を損なわないように注意する必要があります。

Command Window を開くには、メインメニューから **View > Command Window** を選択します。

図 41 Command Window



このウィンドウには 2 つの領域、「履歴ペイン」（項目 1）と「入力フィールド」（項目 2）が含まれます。

履歴ペイン

入力フィールドにコマンドを入力すると、そのコマンドが上の履歴ペインに表示されます。履歴ペインのバッファには、現在の PhyreStation セッションが始まってから入力されたすべてのスクリプトコマンド

(無効なコマンドや失敗したコマンドも含む) が格納されています。入力したテキストは青色で表示されます。エラー/警告メッセージはすべて赤色で表示されます。履歴ペイン内のテキストは、選択してコピーすることができます。

入力フィールド

入力フィールドでは大文字と小文字が区別されます。このフィールドでは、Lua コマンドまたはスクリプト登録コマンドを手動で入力できます。

コマンドの構文が認識/理解された場合、ユーザがキーボードの **Return** または **Enter** を押すとすぐにコマンドが実行されます。入力したテキストは必ず、青色で表示されます。構文エラーが発生すると、必要な構文の説明が履歴ペインに表示されます。

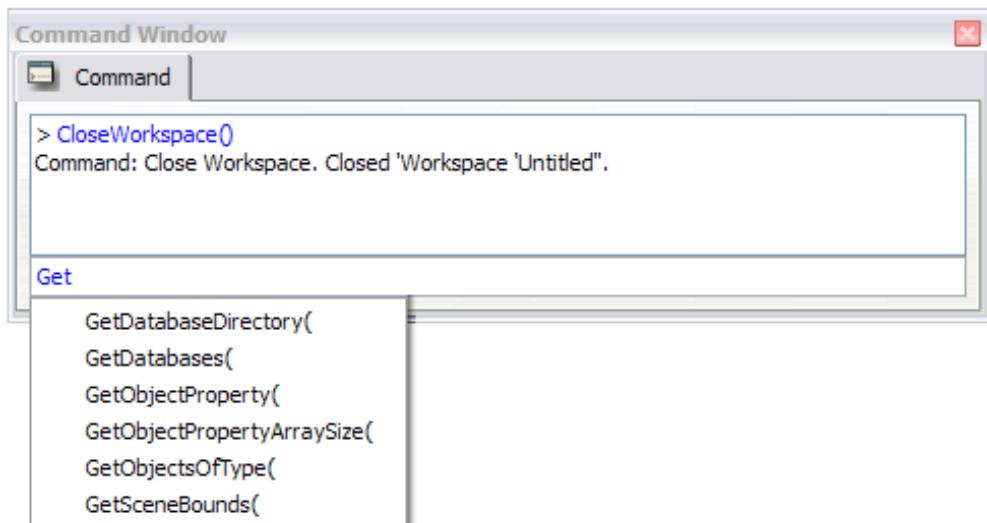
履歴バッファ

入力フィールドは履歴バッファを備えています。以前のエントリを表示するには、**上矢印キー**を押します。すべてのエントリを日付の古い順に表示するには、**下矢印キー**を押します。

オートコンプリータ

入力フィールドはオートコンプリータを備えています。1つ以上のコマンド名に一致するテキストを入力すると、それらのコマンド名がドロップダウンメニューに表示されます。このメニューからコマンドを選択するには、カーソルキーを使用します。

図 42 入力フィールドのオートコンプリータ



入力フィールドが空の状態で右クリックすれば、すべてのスクリプト登録コマンドのポップアップリストを表示できます。このリストからコマンドを選択すると、そのコマンドが自動的に入力フィールドにコピーされます。コマンドの編集が完了したら、通常と同じく **Return** または **Enter** キーを押します。

ヒント :

- コマンドインタプリタでは、フォワードスラッシュ ('/') とバックスラッシュ ('\') ペアの両方をファイルパスの区切り文字として使用できます。たとえば、C:/Docs/PhyreEngine と C:\\Docs\\\\PhyreEngine は等価です。
- 何らかの OS ファイルブラウザ (Windows Explorer など) から入力フィールドにスクリプトファイルをドラッグすることができます。PhyreStation はそのスクリプトの実行を試みます。スクリプトファイルの実行は、入力フィールドで Lua コマンド `dofile` を使って行うこともできます。

注意：Tab キーが **Command Window** のコンテキスト内でこここの説明に従って期待どおりに動作する（キー ボードのフォーカスが別のウィンドウやウィジェットに移動しないようにする）ためには、**Command Window** の任意の部分にカーソルを置きます。

PhyreEngine™オブジェクト ID

コマンドの中には、PhyreEngine™オブジェクト ID をパラメータの 1 つとして要求するものがあります。

PhyreEngine™オブジェクト ID は、次の 2 つの形式のいずれかに従う文字列です。

- database.extension#object
- database.extension

ID には次のルールが適用されます。

- オブジェクト名を定義する必要があります。つまり、database.extension#は無効です。
- データベースオブジェクトを扱っている場合には、データベース名の後に PhyreEngine™ファイル拡張子 (.hier または.pssg) を指定する必要があります。
- ID では大文字と小文字が区別されます。

コマンドからのログ情報

コマンドは実行時にステータスマッセージを生成できますが、このメッセージは PhyreStation によって **Log** ウィンドウかログファイルに転送されます。

通常、メッセージには、ワークスペース名、PhyreEngine™オブジェクト名、および実行された作業のタイプに関する情報が含まれます。パフォーマンスが問題にならない限り、コマンドやスクリプトが成功したかどうかをユーザが監視できるよう、すべてのコマンドにステータス情報を報告させることをお勧めします。詳細については、「Log ウィンドウ」(17 PhyreStation のログ、ユーザプリファレンス、およびエラー処理) を参照してください。

7 スクリプト

この章では、PhyreStation のスクリプティングメカニズムについて説明します。スクリプトコマンドの実行方法やスクリプトファイルの管理方法を説明します。

概要

PhyreStation の主な目的の 1 つは、データの処理を自動化するスクリプトを実行することです。スクリプトを使えば、**Command Window** で個々のコマンドを入力しなくてすみます。OS のコマンドラインプロンプトからアプリケーション実行文の一部として PhyreStation スクリプトパラメータを渡す際にスクリプトファイルを指定することもできますし、PhyreStation の GUI からスクリプトを実行することもできます。

PhyreStation には、スクリプトコマンドを自動化するためのコマンドインタプリタ（スクリプタ）が組み込まれています。このコマンドインタプリタは Lua スクリプティングエンジンを基に開発されたもので、いくつかの追加データ型やユーザ定義コマンドを扱えるように拡張されています。一般的な概要については、関連する Lua ドキュメントを参照してください。<http://www.Lua.org> も参照してください。

スクリプトコマンドは、PhyreStationDLL 内でスクリプト可能コマンドとして具体的に登録することで、PhyreStation や Lua から認識/処理できるようにする必要があります。コマンドの詳細については、「6 PhyreStation コマンド」を参照してください。

PhyreStation での処理用として登録されたコマンド（スクリプト可能なコマンドも含む）の一覧を表示するには、メインメニューから **Help > Help Contents...** を選択します。

スクリプトファイル

PhyreStation のスクリプトファイルは、通常のテキストエディタで作成可能な、ASCII 形式のテキストファイルです。スクリプトファイルには、スクリプトコマンドと Lua 命令の任意の有効な組み合わせを含めることができます。

ワークスペースへのスクリプトの関連付け

スクリプトファイルはワークスペースファイルとは別個に存在します。しかしながら、スクリプトをワークスペースに追加すれば、スクリプトファイルをワークスペースに関連付けることができます。スクリプトファイルへの参照が、相対ファイルパスを使ってワークスペースファイルに保存されます。1 つのスクリプトファイルを複数のワークスペース（ワークスペースファイル）に関連付けることができます。ワークスペースに関連付けられたスクリプトファイルは、ワークスペース内でスクリプトオブジェクトとして表現されます。

スクリプトファイルをワークスペースに関連付けるには、次のいずれかを行います。

- OS のファイルブラウザを使ってスクリプトファイルを選択し、それを **Workspace Explorer - Scripts** ビューにドロップします。
- **Workspace Explorer - Scripts** ビューのコンテキストメニューから **Add Script** を選択します。

空のスクリプトファイルをワークスペースに関連付けるには、ビューのコンテキストメニューから **Create Script** を選択します。デフォルトのエディタが指定されていなかった場合、PhyreStation はユーザに対し、スクリプト Lua テキストファイルに関連付けるエディタをブラウズして選択するよう要求します。その後、ユーザが選択したエディタが自動的に開かれ、空のスクリプトファイルを編集できる状態になります。

注意 : Windows の場合、スクリプトファイルとワークスペースファイルが同じドライブを共有している必要があります。

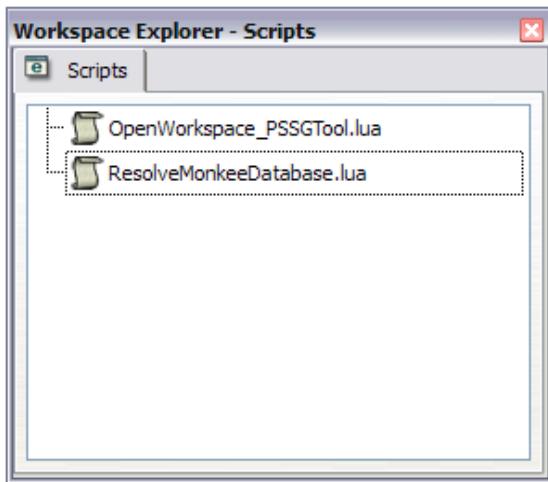
スクリプトエディタの変更

デフォルトのエディタを変更するには、メインメニューから **Edit > User Preferences > PhyreStation** を選択し、**Default script editor** プリファレンスを設定します。

スクリプトのブラウズ

スクリプトオブジェクトを表示するには **Workspace Explorer - Scripts** ビューを使用します。ワークスペースに関連付けられたスクリプトファイルが見つからない場合、そのスクリプトオブジェクトはグレー表示になります。

図 43 Workspace Explorer – Scripts ビュー



次のディレクトリ内に、サンプルのスクリプトファイルがいくつか含まれています。

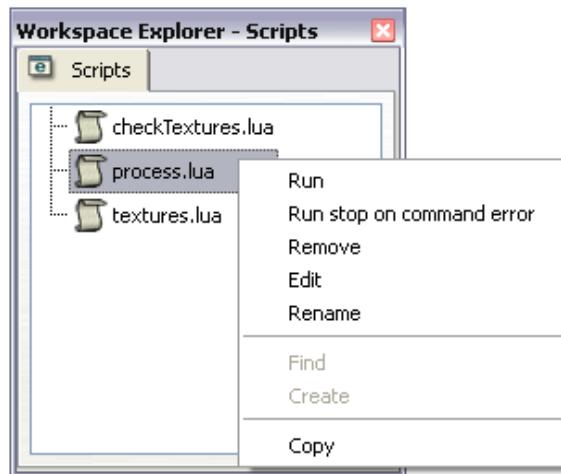
[PhyreStation_root_directory]/PhyreStationScripts

GUIスクリプトオブジェクトの処理

Workspace Explorer - Scripts ビューでスクリプトオブジェクトを右クリックすれば、次のオプションを含むコンテキストメニューを表示できます。

- **Run** – スクリプトの末尾に達するか Lua エラーが発生するまで、スクリプトを実行します。
- **Run stop on command error** – Lua エラーまたはコマンドエラーが発生するかスクリプトの末尾に達するまで、スクリプトを実行します。
- **Remove** – ワークスペースからスクリプトオブジェクトを削除します（関連付けを解除する）。
- **Edit** – ユーザが選択したエディタにスクリプトを送り、スクリプトを表示または変更できるようにします。
- **Rename** – スクリプトファイルの名前を変更します。
- **Find** – 行方不明のスクリプトファイルを見つけ出し、それをワークスペースに再度関連付けます。
- **Create** – 行方不明のスクリプトファイルを作成し直します。
- **Copy** – スクリプトファイルの名前を OS のクリップボードにコピーします。

図 44 Workspace Explorer – Scripts ビュー



スクリプトの実行

PhyreStation は、その起動方法に応じて次の 3 つのモードのいずれかで動作します。

- GUI モード : PhyreStation のアイコンをクリックするか、あるいは OS のコマンドラインからパラメータを何も指定せずに実行します。
- スクリプト GUI モード : OS のコマンドラインから追加のパラメータ (パラメータ -scriptQuit="false" を含む) を指定して実行します。PhyreStation はスクリプトを実行した後、GUI アプリケーションとして動作を続けます。コマンドラインで指定されたスクリプトからワークスペースを指定できます。
- バッチモード : OS のコマンドラインから実行します。PhyreStation はスクリプトの実行後、すぐに終了します。

GUI からスクリプトファイルをロード/実行するには、次のいずれかを行います。

- メインメニューから **File > Run script from file...** を選択します。ワークスペースが開いていると、このオプションは無効化されます。
- メインツールバーの **Run script from file** ボタン をクリックします。ワークスペースが開いていると、このボタンは無効化されます。
- **Command** ウィンドウの入力フィールドから、Lua の `dofile()` コマンドを使ってスクリプトを実行します。
- OS のコマンドラインプロンプトから PhyreStation を実行した場合。
- 別のスクリプトファイルから、Lua の `dofile()` コマンドを使ってスクリプトをロード/実行します。
- OS のファイルブラウザからスクリプトファイルをドラッグし、それを **Script Manager** ダイアログ ボックスまたは **Command Window** の入力フィールドにドロップします。

PhyreStation が GUI モードでスクリプトを実行している場合、PhyreStation は GUI を無効にし、スクリプトの実行が完了するまで GUI を更新しません。PhyreStation はスクリプトを実行している間、すべてのコマンドステータスマッセージを **Log** ウィンドウ、オプションのスクリプトファイルのいずれかに転送します。

コマンドラインからのスクリプトの実行

概要

PhyreStation は OS のコマンドラインから実行することができますが、その際、コマンドラインパラメータを指定してもしなくともかまいません。

- PhyreStation 実行可能ファイルのコマンドラインエントリの一部としてパラメータを 1 つも指定しなかつた場合、デスクトップアイコンからアプリケーションを起動した場合と同じく、GUI モードでアプリケーションが起動されます。
- パラメータの設定で、PhyreStation の起動方法を制御することができます。それらのパラメータには、コマンドラインからのスクリプト実行を可能にするスクリプトタイプのパラメータが含まれます。スクリプト固有パラメータの名前は必ず、「script」という単語で始まります。すべてのパラメータで大文字と小文字が区別されます。一部のスクリプトパラメータを指定すると、PhyreStation が最小モードで起動されますが、その場合、スプラッシュ画面は表示されず、OS のタスクバー上でもアプリケーションが実行中であることがわかりません。

PhyreStation を OS のコマンドラインから実行した場合、コマンドプロンプトが即座にリターンします。

アプリケーションインスタンス ID

アプリケーションインスタンス ID は、現在実行されている PhyreStation アプリケーションのインスタンスを識別します。PhyreStation を GUI モードで起動する場合、この ID は事前に決定されます。このインスタンス ID を確認するには、メインメニューから **Help > About PhyreStation** を選択します。ところが、PhyreStation をコマンドラインから起動する場合には、-appId パラメータを使ってこの ID を指定できます。このパラメータを使えば、新しいタスクを実行する際のターゲットとして、特定のアプリケーションインスタンスを指定できます。次に例を示します。

```
-appId="OvernightJob05"
```

コマンドラインを使えば、複数のアプリケーションインスタンスを同時に実行し、たった 1 つの OS コマンドラインウィンドウを使ってさまざまなジョブを特定の PhyreStation インスタンスに送る、といったことが可能になります。

パラメータ

-help

利用可能なすべてのコマンドラインオプションとその指定手順を確認するには、-help パラメータオプションを使用します。

-script

PhyreStation の起動時に Lua スクリプトファイルを自動的に実行するには、OS コマンドラインに-script オプションを追加します。次に例を示します。

```
-script="C:/user/idave/projects/setup.lua"
```

これにより、指定されたファイルディレクトリから Lua スクリプトファイルがロード/実行されます。

- パラメータworkspace をこのパラメータ-script と組み合わせて使用した場合、このパラメータに指定する必要があるのは、指定されたワークスペース内に格納された Lua スクリプトファイルの名前だけになります。パラメータworkspace には、既存の PhyreStation ワークスペースファイルの名前とディレクトリパスを指定します。次に例を示します。

```
PhyreStation -workspace="C:/user/idave/projects/AWorkspace.xml"
-script="AScript.lua"
```

- このワークスペースでスクリプトが見つからない場合には、PhyreStation は同じ名前を持つファイルからのスクリプトのロード/実行を試みます。
- 環境変数をファイルパスの一部として使用できます。次に例を示します。

```
PhyreStation -workspace="$ (MY_ENV_VAR) /AWorkspace.xml"
-script="AScript.lua"
```

```
PhyreStation -workspace="%MY_ENV_VAR%/AWorkspace.xml"
-script="AScript.lua"
```

- パラメータ `-workspace` は、必ず `-script` パラメータと組み合わせて使用する必要があるほか、すでに実行中のアプリケーションのインスタンスに対して使用することはできません。
- パラメータ `-workspace` は指定されていないがプリファレンス **Load workspace on startup** は設定されている場合、スクリプトの実行前に空のデフォルトワークスペースが作成されます。スクリプト内で `NewWorkspace` または `OpenWorkspace` コマンドが実行されると、そのデフォルトワークスペースがオーバーライドされます。

注意：

- ワークスペースファイル名、スクリプト名のいずれかを指定する場合には、ファイル拡張子（`.lua` など）を含める必要があります。そうしないとパラメータが拒否されます。
- コマンドの中には、PhyreStation 内でワークスペースセッションが開かれていないと動作しないものもあります。

-scriptQuit

スクリプトファイルまたはコマンドの実行後にシャットダウンすべきかどうかを PhyreStation に指示するには、`-scriptQuit` パラメータを使用します。次に例を示します。

```
-scriptQuit=true
```

デフォルトでは、このオプションは `false` に設定され、スクリプトファイルのロード/実行後も PhyreStation の実行が継続されます。

このオプションが `true` に設定されると、PhyreStation はバッチモードで動作します。アプリケーションは最小モードで実行され、その結果、次のような効果が得られます。

- アプリケーションは GUI システムをまったく起動/更新しません。したがって、アプリケーションの実行速度が向上します。
- アプリケーションを実行しても、OS のコマンドラインウィンドウ内にアプリケーションメッセージやコマンドメッセージがまったく表示されません。メッセージをテキストファイルにエクスポートするには、`-scriptLog` パラメータを使用します。
- アプリケーションインスタンスは、起動されるものの画面上には表示されず、タスクバー上で実行中として表示されることもありません。
- `-scriptQuit` パラメータは、既存の実行中のアプリケーションインスタンスに対して指定した場合には無視されます。

-workspace

`-script` パラメータを参照してください。

-scriptExecute

`-scriptExecute` パラメータを`-script` パラメータと組み合わせて使用すれば、コマンドラインから追加 Lua スクリプトを渡すことができます。最初に追加スクリプトが実行され、その後で`-script` パラメータに指定されたスクリプトが実行されます。

このため、外部から既存のスクリプトにパラメータを渡すことが可能となります。次に例を示します。

```
-scriptExecute="MyDbName=\"a.PSSG\"; MyDbDir=\"C:/user/idave/databases\""
-script="C:/user/idave/projects/readfiles.lua"
```

ファイルから読み込まれた Lua スクリプトにこれらの変数が含まれ、使用されていた場合、コマンドラインから入力された値がそのスクリプトの実行時に使用されます。

Lua コマンドや PhyreStationDLL 登録スクリプトコマンドも実行できます。次に例を示します。

```
-scriptExecute="dofile(\"C:/user/idave/projects/setup.lua\")"
-scriptExecute="AddDatabase(\"C:/user/idave\", \"monkee.pssg\")"
```

`-script` パラメータと`-scriptExecute` パラメータをいっしょに使用する必要はありません。

注意 : PhyreStation でワークスペースセッションが開かれていないと動作しないコマンドもあるので注意してください。

-scriptLog

指定されたファイルにすべてのメッセージを出力するようにアプリケーションに指示するには、`-scriptLog` パラメータを使用します。通常は **Log** ウィンドウ内に表示されるメッセージが、代わりにファイル内に格納されます。生成されるファイルはテキストファイルです。次に例を示します。

```
-scriptLog="BatchJob01Results.txt"
```

ファイルがすでに存在している場合、そのファイルは同名の新しいファイルで上書きされます。

このパラメータは、`-scriptQuit` パラメータもいっしょに指定しないと無視されます。

注意 : `scriptLog` で報告されるのはエラーだけです。**Log** ウィンドウの内容がそのまま反映されるわけではありません。なぜなら、そうすることで、スクリプトの実行速度を最高に保てるからです。スクリプトをまずアプリケーションから実行してみて、スクリプトが正常に動作するか確認することをお勧めします

-scriptStopCmdError

このパラメータが次のように使われたとします。

```
-scriptStopCmdError=true
```

このスクリプトの実行は、失敗（「completed task ok」に等しくないリターン結果）を返すコマンドが初めて検出された時点で停止します。このパラメータのデフォルト値は `false` です。

注意 : Lua エラーが発生すると、このパラメータの値が何であってもスクリプティングシステムが停止します。

PhyreStation 実行のリターン結果

PhyreStation は、アプリケーションの実行後に整数値を返します。表 4 にリターン結果の一覧を示します。

表 4 PhyreStation の実行結果

ステータスコード	解説
0	アプリケーションが正常にタスクの実行を完了しました
1	PhyreStation が OS のウィンドウシステムをサポートしません
2	PhyreStation の初期化中にエラーが発生しました
3	PhyreStation でターミナルコードエラーが発生しました
-1	コマンドラインスクリプトエラーが発生しました
-2	PhyreStation の内部用

追加のLuaデータ型

PhyreStation の Lua 実装は、PhyreEngine™で重要ないくつかの追加データ型を扱えるように拡張されています。それらの新しい型を作成/操作するためのコマンドが、新たに追加されています。

ベクトル

`vector` には、4 つの数値から成るベクトルが格納されます。ベクトルを作成するには関数 `vector(a,b,c,d)` を使用します。この関数は新しいベクトルを返します。

`a`、`b`、`c`、および `d` は、ベクトルの各要素の初期値を設定するオプションパラメータです。値を指定しなかった要素では値ゼロが使用されます。次に例を示します。

```
v1 = vector()
v2 = vector( 0, 1, 0, 3 )
```

ベクトルの要素にアクセスするには、配列インデックスを使用するか、名前を指定します。`v[0]` は `v.x` と、`v[1]` は `v.y` と、`v[2]` は `v.z` と、そして `v[3]` は `v.w` と、それぞれ等価になります。次に例を示します。

```
v3[0] = v1.y // v3[0] = v1[1] と等価
```

ベクトルに対して出力や連結などの文字列操作を行うと、そのベクトルが文字列に自動変換されます。次に例を示します。

```
v3 = vector( 1, 2, 3, 4 )
print( v1 )
```

ベクトルの演算

以下では、`v1` と `v2` はベクトル、`m1` は行列、`n1` は数値を表します。

表 5 ベクトルの演算

ベクトルの加算	次に例を示します。 <code>r = v1 + v2</code>	<code>v1</code> と <code>v2</code> の和を新しいベクトルとして返します。
ベクトルの減算	次に例を示します。 <code>r = v1 - v2</code>	<code>v2</code> を <code>v1</code> から引いた結果を、新しいベクトルとして返します。
ベクトルと行列の積	次に例を示します。 <code>r = v1 * m1</code>	<code>v1</code> を <code>m1</code> に適用した結果を、新しいベクトルとして返します。
ベクトル同士の積	次に例を示します。 <code>r = v1 * v2</code>	各フィールドが <code>v1</code> と <code>v2</code> の対応するフィールドの積になっているような新しいベクトルを返します。

ベクトルと数値の積	次に例を示します。 <code>r = v1 * n1</code> これは次の式と等価です。 <code>r = n1 * v1</code>	v1 を n1 の値でスケーリングしたものを、新しいベクトルとして返します。
ベクトル同士のクロス積	次に例を示します。 <code>r = v1 ^ v2</code>	v1 と v2 のクロス積を新しいベクトルとして返します。ただし、どちらのベクトルも vector-3 として扱われ、結果の 4 番目の要素は 0 に設定されます。
ベクトルの正規化	次に例を示します。 <code>r = v1:normalize()</code>	v1 を正規化したものを、新しいベクトルとして返します。
ベクトルの長さ	次に例を示します。 <code>r = v1:length()</code>	v1 の長さを表す数値を返します。

行列

行列には、数値を要素を持つ 4×4 行列が格納されます。

行列を作成するには関数 `matrix()` を使用します。この関数は新しい行列を返します。新しい行列は単位行列に設定されます。次に例を示します。

```
m1 = matrix()
```

行列の要素にアクセスするには配列インデックスを使用します。m[0] は行列の 1 行目の最初の要素、m[1] は行列の 1 行目の 2 番目の要素、m[4] は行列の 2 行目の最初の要素、といった具合になります。次に例を示します。

```
m1[0] = 2.0
m2[5] = m2[5] * 3.0
```

行列に対して出力や連結などの文字列操作を行うと、その行列が文字列に自動変換されます。次に例を示します。

```
m1 = matrix()
print( m1 )
```

クオータニオン

クオータニオンには、姿勢として使用される 4 つの数値が格納されます。

クオータニオンを作成するには関数 `quaternion(a, b, c, d)` を使用します。この関数は新しいクオータニオンを返します。a、b、c、および d は、クオータニオンの各要素の初期値を設定するオプションパラメータです。値を指定しなかった要素では値ゼロが使用されます。次に例を示します。

```
q1 = quaternion()
q2 = vector( 3, 2, 1, 1 )
```

クオータニオンの要素にアクセスするには、配列インデックスを使用するか、名前を指定します。q[0] は q.x と、q[1] は q.y と、q[2] は q.z と、そして q[3] は q.w と、それぞれ等価になります。次に例を示します。

```
q3[0] = q1.z // q3[0] = q1[2]と等価  
q2.z = q2[1] * q2[0]
```

クオータニオンに対して出力や連結などの文字列操作を行うと、そのクオータニオンが文字列に自動変換されます。次に例を示します。

```
q1 = quaternion( 0, 0, 0, 1 )  
print( q1 )
```

8 PhyreStationのカスタマイズ

この章では、PhyreStationDLL のカスタマイズに関する詳細情報を提供します。

概要

PhyreStation を拡張すれば、新しい型の PhyreEngine™オブジェクトに対応させることができます。また、それらのカスタムオブジェクト型を PhyreStation で操作できるように、PhyreStationDLL に新しいコマンドを追加して PhyreStation の内部コマンドセットを拡張することもできます。PhyreEngine™オブジェクトコマンドとは、PhyreEngine™オブジェクトのみに関連付けられた（登録された）コマンドのことです。

PhyreStationDLL のソースコードは次の 2 つの部分に分けられます。

コア機能

ソースコードのコア機能部分は変更すべきではありません。PhyreStationDLL のこの部分には、ビジタークラスや処理インターフェースなど、PhyreStation が正しく動作するために必要となる基本的なクラス/インターフェースが含まれています。また、PhyreStation のコマンドシステムを拡張するための基底クラスも、ここに含まれています。

注意：プログラマは、PhyreStation がすでに依存しているそれらの PhyreStationDLL インタフェース / ビジタークラスを変更しないように注意する必要があります。

拡張ユーティリティ

PhyreStationDLL の拡張ユーティリティは、PhyreStationDLL のコア機能を使ってシステムを拡張する新規コードで構成されています。これらのライブラリは、PhyreStationDLL のビルドプロジェクトにリンクされます。

プログラマは、拡張ユーティリティーアーキテクチャを使って特定タスク専用の各種コードを管理しつつ、PhyreStationDLL のコア機能の整合性を維持すべきです。

PhyreStationDLL をカスタマイズするには次の手順に従います。

- (1) 新しい PhyreEngine™ユーティリティを作成します。
- (2) 新しい PhyreEngine™オブジェクト型を作成します。
- (3) 新しい PhyreEngine™オブジェクト型を登録します。
- (4) PhyreStationDLL をコンパイルすると、PhyreEngine™ユーティリティもコンパイル/リンクされます。
- (5) 新しい PhyreEngine™オブジェクトコマンドを作成します。
- (6) 新しい PhyreEngine™オブジェクトコマンドを登録します。
- (7) オプション：新しいコマンドを PhyreStation 内（コンテキストメニュー上など）で利用できるようにします。必要に応じて新しいバインディングクラスを作成します。
- (8) PhyreStation DLL を再度コンパイルします。
- (9) PhyreStation から新しいコマンドを実行します。

新しいPhyreEngine™ユーティリティの作成

新しいクラスと PhyreEngine™ Core とを分離できるように、作成する新規 PhyreEngine™オブジェクト用の PhyreEngine™ユーティリティクラスを新たに実装することをお勧めします。

新しい PhyreEngine™ユーティリティを作成する方法については、「PhyreEngine™プログラミングガイド」の「PhyreEngine™ユーティリティ」の章を参照してください。PhyreEngine™ユーティリティの作成方法の詳細については、PhyreEngine™ Utilities ディレクトリ内の `readme.txt` ファイルを参照してください。

PhyreStationへの新しいPhyreEngine™オブジェクト型の追加

ここでは、新しい PhyreEngine™オブジェクト型を作成/登録して PhyreStation を拡張する方法について説明します。

概要

PhyreEngine™の強力な機能の 1 つは、ユーザの要件に適応させることができることですが、それは固有の属性セットを備えたユーザ独自のオブジェクト型を実装することで実現されます。新しいオブジェクト型の作成とそれらの PhyreEngine™への登録が完了すると、PhyreStation は自動的にそれらのオブジェクト型を認識し、(それらがデータベース内に存在する場合には) それらを表示できるようになります。新しいオブジェクトに割り当てられた属性はそのオブジェクトの **Properties Viewer** 内で可視となり、検査可能な状態となります。

PhyreEngine™には広範なデフォルトオブジェクト型が用意されています。現在登録されているオブジェクト型のセットを確認するには、PhyreStation を実行し、**Help > About PhyreStation** を選択した後、左側のグループボックスで PhyreStationDLL コンポーネントを選択します。右側のグループボックスに、現在登録されているすべてのコマンドとオブジェクト型の一覧が表示されます。

付属のオブジェクト型

新しいオブジェクトは常に付属の PhyreEngine™オブジェクト型クラスから派生させる（したがって付属の型は変更しない）ことをお勧めします。そうでないと、次の PhyreEngine™リリースをインストールする際に、それまでに行った変更がすべて失われてしまいます（ユーザ独自のオブジェクト型クラスをオリジナルファイルに追加した場合も含む）。

カスタムオブジェクト型

新しいオブジェクト型クラスのコードは必ず、PhyreEngine™ Utilities または PhyreStationDLLUser ファイルを使って追加することをお勧めします。そうすれば、付属の PhyreEngine™コードやメインの PhyreStationDLL コードからユーザのコードを分離することができます。

新しいオブジェクトを処理する独自コマンドを作成する方法については、「PhyreStation への PhyreEngine™オブジェクトコマンドの追加」(8 PhyreStation のカスタマイズ) を参照してください。

新しい PhyreEngine™オブジェクト型の作成

新しく作成するオブジェクトは必ず、PhyreEngine™の基底オブジェクト型クラスから派生させる必要があります。PhyreEngine™に現在登録されているすべてのオブジェクト型を PhyreStation で表示するには、次のようにします。**Workspace Explorer - Objects** ビューを開き、ビューのコンテキストメニューから **Show Empty Categories** を選択します。現在利用可能なすべての型がビューに表示されます。

ユーザが作成したオブジェクトが PhyreStation 内に表示されるためには、ワークスペースにロードされた PhyreEngine™データベース内にそのオブジェクトが存在しているか、あるいはユーザが作成した型の

新しいオブジェクトを作成してワークスペース内のデータベースに追加するようなコマンドを実行する必要があります。ワークスペース内にオブジェクトが存在していて、それがユーザのオブジェクトカテゴリの下に表示されている場合には、そのオブジェクトを選択したり、その **Properties Viewer** でその属性を確認したり、そのオブジェクトに変更を加えたり、データベースを保存したりできます。

新しい PhyreEngine™オブジェクトを作成する際に役立つ情報を、以下に示します。

- ・「PhyreEngine™プログラミングガイド」の「PhyreEngine™オブジェクト」、「新しい PhyreEngine™オブジェクト型の追加」というタイトルの章を参照してください。
- ・[PhyreStation_root_directory]/Samples/Intermediate/newObject 内の PhyreEngine™コード例「newObject」を参照してください。この例に含まれるコードをユーザ独自の PhyreEngine™ユーティリティ内で使用します。
- ・PhyreEngine™の PSSG::PSSGInit() のコードを参照してください。

コンパイル、リンク、およびデバッグ

- (1) 新しい PhyreEngine™オブジェクトを作成するためのコードが完成したら、PhyreStationDLL をコンパイルします。すると、PhyreEngine™ユーティリティもコンパイル/リンクされます。
- (2) オプション：新しいコマンドを実行/登録するためのコードが完成したら、PhyreStationDLL を再度コンパイルします。そうすれば、PhyreStation からコマンドを使用できる状態になります。
- (3) 新しい PhyreStationDLL ファイルを PhyreStation のディレクトリにコピーします。

注意：Microsoft Visual Studio の場合、PhyreStationDLL は Release タイプのビルドになります。PhyreStationDLL ライブラリのデバッグを行うには、PhyreStationDLL プロジェクト設定経由で、デバッグを有効にし、かつコンパイラの最適化をすべて無効にする必要があります。

オブジェクトのアイコン

新しいオブジェクト型を作成する際の、省略可能であるが推薦される作業は、そのオブジェクト型のアイコンを作成することです。このアイコンは 16x16 PNG 画像です。アイコンを割り当てるには、作成したアイコンをディレクトリ [PhyreStation_root_directory]/Resources/Icons 内に格納します。ユーザの新しいオブジェクト型に対する PhyreEngine™アクセサ関数が呼び出され、そのアイコンがファイル名で参照されると、PhyreStation ビュー内のその新しいオブジェクト型の横にそのアイコンが表示されます。

PhyreStationへのPhyreEngine™オブジェクトコマンドの追加

ここでは、ユーザの要件に応じて PhyreStation のコマンドセットを拡張する方法について説明します。

概要

PhyreStationDLL（この DLL にはソースコードが用意されている）を拡張すれば、ユーザのコマンドを組み込んだり既存のコマンドを変更したりできます。

コマンドとは、PhyreStation コマンドパターン API でラップされた関数のことです。この関数は、ユーザが必要とする任意の処理を実行できます。たとえば、ファイルを開いて中身を解析するコマンドや、PhyreEngine™データベースまたはオブジェクトを処理するコマンドを作成することができます。

コマンドは、PhyreStation の GUI から実行する（通常は特定の PhyreEngine™オブジェクト型にバインドし、コンテキストメニューから実行する）ことも、Lua スクリプト内から実行することもできます。ただし、スクリプト内で動作するように登録されたコマンドのすべてを PhyreStation の GUI にも登録する必要があるとは限りませんし、またその逆も成り立ちます。

PhyreStationDLL のソースコードは次の 3 つの部分に分けられます。

コア機能

ソースコードのコア機能部分は変更すべきではありません。PhyreStationDLL のこの部分には、ビジタークラスや処理インターフェースなど、PhyreStation が正しく動作するために必要となる基本的なクラス/インターフェースが含まれています。

注意：プログラマは、PhyreStation が依存するそれらの PhyreStationDLL インタフェース/ビジタークラスを変更しないように注意する必要があります。

付属コマンド

PhyreStation に付属する大部分のコマンドも、PhyreStationDLL に含まれています。付属コマンドを確認するには、PhyreStation の **About PhyreStation** ダイアログを参照してください。この DLL に含まれるコマンドの一部は PhyreStation の機能コマンド（名前付きコマンド）ですが、これらは変更すべきではありません。

About PhyreStation ダイアログに表示されるコマンドは、必要であれば変更可能ですが、変更時には注意が必要です。付属コマンドは変更可能ですが、その変更内容のすべてを記録に残しておかない限り、次のリリース時に変更結果が上書きされてしまいます。同様に、ユーザ独自のコマンドを既存のコマンドといっしょにここに配置することも可能ですが、推奨できません。

拡張ユーティリティ

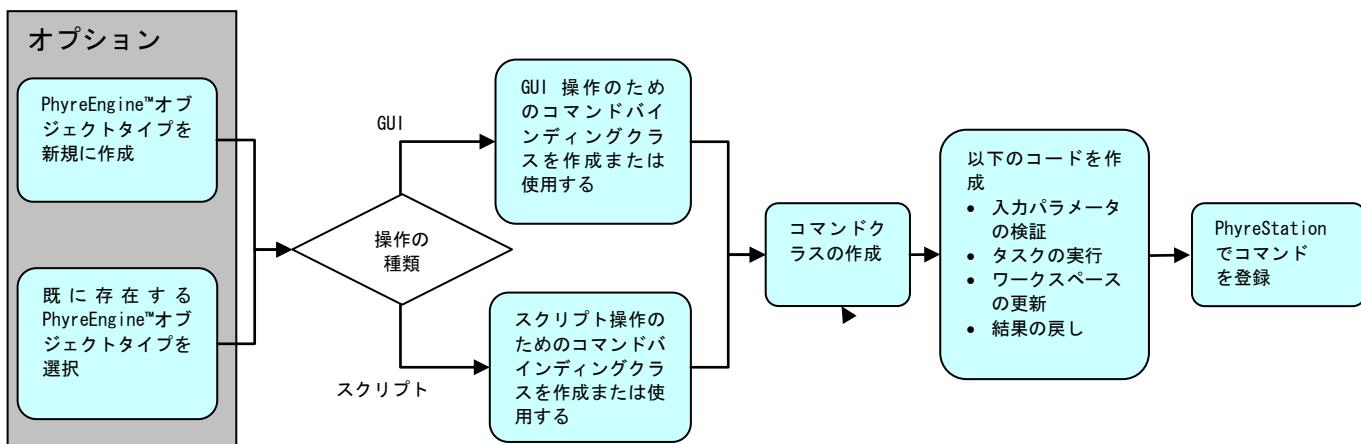
PhyreStation を拡張する最良の方法は、ユーザのコードを PhyreStationDLL ユーザファイル内に配置することです。ユーザの要件によっては、PhyreEngine™のユーティリティを使用してもかまいません。これらのライブラリは、PhyreStationDLL のビルドプロジェクトにリンクされます。

拡張ユーティリティアーキテクチャを使って特定タスク専用の各種コードを管理しつつ、

PhyreStationDLL のコア機能の整合性を維持することをお勧めします。

ユーザ独自のオブジェクトとそれを操作する新しいコマンドの両方を作成する場合には、この章の最初のほうの「PhyreStation への新しい PhyreEngine™オブジェクト型の追加」を参照し、まずユーザのオブジェクトを作成してください。

図 45 新しいコマンドの作成



コマンドが利用可能かどうかの制御 - コマンドバインディングクラス

ここで説明する内容が必要になるのは、PhyreStation の GUI からコマンドを実行する場合（コンテキストメニューから実行する場合やマウスのドラッグ & ドロップ操作の一環として実行する場合など）だけです。（コマンドをスクリプト内で使用するために登録する場合にもバインディングが使用されますが、そ

のタイプのバインディングでは、コマンドが利用可能かどうかの制御は行えません。) GUI コマンドバインディングによる制御内容は、次のとおりです。

- コンテキストメニューにコマンドの名前を表示すべきかどうかを指示します。
- コマンドを関連付けるべき PhyreEngine™オブジェクト型が存在する場合に、そのオブジェクト型を指示します。
- コンテキストメニューにコマンドが表示される場合に、そのコンテキストメニュー内でコマンドを有効にするかどうかを指示します。

GUI は、ユーザとのインタラクションの一環としてすべての登録済みコマンドのバインディングオブジェクトにクエリを送り、オプションまたはアクションをユーザに提供できるようにします。上記の情報は、バインディングが提供しなければならない答えの種類を表しています。

指定された条件またはルールが満たされると、GUI はそのバインディングクラスに、コマンドクラスのインスタンスを作成するように指示します。PhyreStation がこのコマンドクラスの `Execute()` 関数を呼び出すと、このコマンドクラスは選択されたワークスペースオブジェクトに対して必要なタスクを実行します。

最後に、バインディングオブジェクトはコマンドオブジェクトの作成を試みた際に、処理が成功したかどうかを示すステータスを PhyreStation に通知するために、その成功ステータスマップ変数 `m_status` とステータスマッセージ変数 `m_statusMsg` に値を設定します。

PhyreStationDLL には、ユーザが目的に応じて使用したりその派生クラスを作成したりできるコンテキストメニュー基底クラス `PDllContextCmdBindingBase` が用意されています。同様に、すべてのドラッグ&ドロップ操作を使用する場合には、`PDllDragCmdBindingBase` クラスを使用したりその拡張クラスを作成したりすることができます。基底バインディングクラスインターフェースに指定されている一連の関数を以下に示します。これらの関数は、ユーザのバインディングクラスかその親バインディングクラスで実装する必要があります。

- `IsCmdValid()`
- `IsCmdEnabled()`
- `CreateCmd()`
- `IsCmdValidForSrc()` (ドラッグ&ドロップバインディングの場合のみ)

PhyreStationDLL のコマンドバインディングクラスは、各ルールで必要とされる条件を満たしたり、コマンド作成時にコマンドに渡すための追加情報を取得したりするためにユーザに質問を行いますが、その質問方法には制限があります。PhyreStation のメッセージダイアログシステムを利用する目的で、「はい/いいえ」形式の基本的なダイアログインターフェースが PhyreStationDLL インタフェース `PhyreStationDLLInterfaceAskUser` によって提供されます。

また、これらのコマンドバインディング基底クラスから派生したテンプレートクラスも用意されています。それらのテンプレートクラスは、条件満足時にバインディングによって作成されるコマンドの種類に基づいています。

PhyreStationDLL コマンドバインディングクラスの例を、以下に示します。

- `PDllDragCmdBindingLinksAndInternalDB`
ドラッグコマンドを定義します。ただし、項目が他のオブジェクトへのリンクを含んでいるか内部データベースである場合には、コマンドを実行しません。
- `PDllContextCmdBindingLinksResolved`
データベースが正常に解決できる場合専用のコンテキストメニュー命令を定義します。
- `PDllContextCmdBindingDisableForPhyreEngineInternalDatabase`

PhyreEngine™内部データベースに属さない任意のオブジェクト用のコンテキストメニュー命令を定義します。

- PDllContextNodeCmdBinding

PNode から派生したオブジェクト用のコンテキストメニュー命令を定義します。このコマンドを使用できる一連の PNode オブジェクトが、バインディングクラスのコンストラクタに渡されます。

コマンドの登録プロセスの一環として、バインディングクラスのインスタンスが PhyreStation に渡されます。バインディングクラスのどのインスタンスも静的であり、かつ DLL 内のある時点での参照されるようにすることをお勧めします。というのも、コンパイラがバインディングクラスに対してデッドストップを行う可能性があるからです。その場合、コマンド登録に関する問題が発生する可能性があります。

コマンドクラス

PhyreStation と通信を行うコマンドはすべて、PhyreStationDLL に含まれる基底コマンドクラス PDLLCmd から派生したもので、このことは、GUI で使用するコマンド、スクリプト内から使用するコマンドの別を問わず、ユーザが作成するどのコマンドにも当てはまります。PhyreStationDLL には、PDLLCmd 基底クラスから派生したさまざまな基底クラスが含まれています。それらの派生クラス (SaveDatabaseCompressed() など) は、処理対象の PhyreEngine™オブジェクト型や実行するタスクの種類をより絞り込んだものになっています。

それらの各種基底クラスの中には、テンプレートタイプのクラスもいくつか含まれています。ここでもやはり、ユーザはそれらを使用しても、独自のものを作成してもかまいません。

コマンドクラスのさらなる分類基準としては、コマンドが単一のオブジェクトのみを処理するのか、あるいは複数のオブジェクトを処理できるのか（つまり、ユーザが GUI で多数のオブジェクトを選択した場合に対応できるのか）が挙げられます。複数のオブジェクトを処理するコマンドは、スクリプトコマンドとして使用するには適していません。

注意：PDLLCmd クラスは BBaseCmd クラスから派生しています。BBaseCmd は変更しないでください。これは、PhyreStation の動作に必須のものだからです。

コマンド ID

クラスメンバ変数 m_CmdId には、このコマンドを他のすべてのコマンド ID から区別するための一意のテキスト文字列が格納されます。これは PhyreStation によって、個々のコマンドを追跡する目的で使用されます。これは通常、ユーザに表示されません。

コマンド名

クラスメンバ変数 m_CmdName には、コマンドの名前を表すテキスト文字列（「Database」など。通常は処理対象となるオブジェクトの型名が使用される）が格納されます。このテキストはユーザに表示されます。これがコマンドの名詞になります。

コマンドコンテキスト

クラスメンバ変数 m_CmdContext には、コマンドのアクションを表すテキスト文字列が格納されます。これは通常、何を行うかを記述します（例：「Save Compressed」）。このテキストはユーザに表示されます。これがコマンドの動詞になります。PhyreStation は、コマンド名とコマンドコンテキストを連結してユーザ用の記述（「Save Compressed Database」など）を生成します。この記述は PhyreStation の Log レポートに表示されるほか、GUI のコンテキストメニューの項目として表示される可能性もあります。したがって、この記述は短くなければなりません。

コマンドの説明

すべてのコマンドは、自身に関する情報を格納できます。この追加情報は、ユーザによるコマンドの理解や使用を容易にする目的で、PhyreStation によって表示される可能性があります。この情報は次のカテゴリに分割されます。

- 構文 – コマンドのフォーマットを説明するテキスト。
- 構文の説明 – 入出力パラメータの型に関するテキスト情報。
- 説明 – コマンドの機能とコマンドが使用される文脈について、ユーザに説明するテキスト。

この情報はメタデータとして格納されます。PhyreStation は、コマンド API 関数 GetMetaData(...) 経由でこの情報にアクセスします。

PhyreStation はコマンド情報を使って次のことを行います。

- PhyreStation の **Help Viewer Registered Commands** ページに含まれるオンラインヘルプページの生成。
「Help Viewer」(1 PhyreStation の概要) を参照してください。
- ツールヒントの生成。
- コマンドの構文情報の提供。

コマンドの引数

多くのコマンドは、実行時に追加のデータや情報を与えてやらないと、タスクを実行できません。また、タスク完了時に PhyreStation やスクリプトに情報を返さなければならぬこともあります。

コマンドの入力データは、コマンドのパラメータ（括弧の内側に指定された情報）を介してコマンドに送られます。コマンドの出力データは、左辺が存在する場合はそこに値を代入することで返されます。

PhyreStation コマンドは複数のデータ値を返すことができます。コマンドは、バリアント値 FIFO スタックを使って入力データ、リターンデータの両方を渡します。

あるコマンドが特定のオブジェクトで使用される場合の最初の入力パラメータは、常に PhyreEngine™ オブジェクト ID またはリンク名 (*database#name*) になります。これは、スクリプト内からコマンドを使用する際に容易に確認できます。他の値はコマンドごとに異なります。コマンドのパラメータの検証は、そのコマンドクラスのコンストラクタ、Execute() 関数のいずれかで行えます。無効なパラメータが見つかった場合には、タスクのどの部分も実行されることなく、Execute() 関数経由でコマンドステータスコード PE_CMDRESULT_CMDOKBUTBADPARAMSIN が返されます。

コマンドのリターンスタックの最初の値は、そのコマンドの PhyreEngine™ エラーコードでなければなりません。コマンドの Execute() 関数内で PDLLCmd::m_result にエラーコードが代入された後、

PhyreStation による PDLLCmd::GetReturnsStack() の呼び出し時に自動的に、その値がリターン結果スタック上に格納されます。PhyreStation はこの結果コードに基づいて、実行したばかりのコマンドの成功ステータスを評価できます。コマンドがスクリプト内で使用された場合には、スクリプトに結果コードが返されます。他の後続のリターン値は、コマンドごとに異なります。

BRredoUndoParamStackBase は、パラメータ渡しに使用されるバリアントタイプのスタッククラスです。このスタックにデータ（コマンドパラメータデータ）を追加するには、その AddParameter() 関数を使用します。関数 GetParameters() は、コマンド実行中のリターンスタック準備時に格納された値を取得する目的で、PhyreStation によって使用されます。

コマンドの Execute() 関数

コマンドは、Execute() 関数のスコープ内でタスクを実行します。この関数は PhyreStation から呼び出された時点で、タスクの実行に必要なすべてのデータや情報を持っているべきです。コマンドが

PhyreStation の GUI で実行される場合、そのコマンドはユーザに追加情報の入力を要求すべきではありません。必要な情報はすべて、コマンドのバインディングクラスを使って要求すべきです。

コマンドステータスマッセージ

コマンドステータスマッセージの生成は必須ではありませんが、コマンドのタスクの成功/失敗に関するフィードバックをユーザに返せるので、そうすることをお勧めします。たとえば、タスクが成功した場合、ユーザはおそらく、コマンドが使用したオブジェクトや新しいオブジェクトの配置場所を知りたいはずです。タスクが失敗した場合、コマンドステータスマッセージを使ってアドバイスを与えることができます。返されたメッセージはすべて、PhyreStation のログファイルにも送られます。

1つのコマンドから送り返すことのできるメッセージは1つだけですが、その際、コマンドの関数 `SetLogReport()`、`SetCmdErrorMsg()` のいずれかを使用します。

注意：コマンドのタスクの成功/失敗を報告するテキストメッセージは常に、UTF8 形式でフォーマットされます。「コマンドのローカリゼーションサポート」(17 PhyreStation のログ、ユーザプリファレンス、およびエラー処理) を参照してください。

コマンドの成功ステータスコード

コマンドは、PhyreEngine™エラーコードを（コマンドのリターンstack 経由で）PhyreStation に渡します。このエラーコードはスクリプト、GUI のいずれかに渡されます。

さらに、コマンドはコマンドステータスコードも返します。PhyreStation はこのコマンドステータスコードに基づいて、自身の内部状態を更新したり、適切なアクションを実行したり、ユーザ向けのレポートを作成したりします。コマンドステータスコードは、コマンドの `Execute()` 関数のスコープを離れる際に返す必要があります。

表 6 に、利用可能なコマンドステータスリターンコードの一覧を示します。

表 6 コマンドステータスリターンコード

ステータスコード	解説	使用するフィードバック関数
<code>PE_CMDRESULT_COMPLETEDTASKOK</code>	コマンドのタスクが正常に終了しました。	<code>SetLogReport()</code>
<code>PE_CMDRESULT_USERCANCEL</code>	コマンドがユーザによってキャンセルまたは中断されました。	<code>SetLogReport()</code>
<code>PE_CMDRESULT_CMDOKBUTTASKFAILED</code>	何らかの外部要因によってタスクが失敗しました（読み取り専用ファイルに書き込もうとした場合など）。	<code>SetCmdErrorMsg()</code>
<code>PE_CMDRESULT_CMDOKBUTBADPARAMSIN</code>	1つ以上のコマンドパラメータが不足しているか、型が間違っているか、あるいはその他の原因で無効です。	<code>SetCmdErrorMsg()</code>
<code>PE_CMDRESULT_CMDFAILED</code>	コマンドが異常終了しました。原因是、内部の整合性チェックコードでのエラーです。	<code>SetCmdErrorMsg()</code>
<code>PE_CMDRESULT_UNDOFNNOTIMPLEMENTED</code>	<code>IsUnExecuteable()</code> が <code>true</code> を返す場合に、コマンドの <code>UnExecute()</code> メンバ関数から返されます。	<code>SetCmdErrorMsg()</code>

リターンコード `PE_CMDRESULT_CMDFAILED` は、突発的な障害によってコマンドが失敗した場合（通常は、例外のスロー/キャッチが発生した場合）にも、PhyreStation によって使用されます。

コマンドの UnExecute()関数

コマンドクラスには `UnExecute()` 関数が用意されています。この関数は、コマンドの `Execute()` 関数と逆の機能を提供します。

このリリースの PhyreStation は、コマンドの `Execute()` 関数の Undo 機能を積極的にはサポートしません。代わりにコマンドの `IsUnExecuteable()` を実装し、`false` を返すようにしてください。

`UnExecute()` 関数は実行されません。

ワークスペースの更新

PhyreStation がバッチモードでスクリプトを実行する（起動してスクリプトを実行した後、すぐに終了する）のでない限り、コマンドはワークスペースを更新できるよう、自身が作成、変更、または削除したすべてのオブジェクトの情報をワークスペースに伝える必要があります。コマンドの `Execute()` 関数のスコープ内で、コマンドは自分が処理したオブジェクトの情報を PhyreStation に伝えます。コマンドの実行が完了すると、PhyreStation は提供されたオブジェクトのリストに基づいてワークスペースを更新します。スクリプト内でコマンドを実行する場合には、スクリプトの実行終了時にワークスペースの更新が行われます。

コマンドが保留中のすべての変更情報をワークスペースに伝えるには、コマンドの以下の基底クラス関数の 1つまたは複数を呼び出す必要があります。

- `void PDLLCmd::addObject(...)`
- `void PDLLCmd::refreshObject(...)`
- `void PDDLCmd::deleteObject(...)`

これらの関数は何回でも呼び出すことができるほか、同じオブジェクトに対して繰り返し呼び出すこともできます。以前に更新されたオブジェクトに対して再度関数が呼び出された場合、以前の関数の結果はその新しい関数の結果でオーバーライドされます。ただし、削除関数の後でリフレッシュ関数を呼び出しても、何の意味もありません。

コマンドによる変更処理が完了したら、最後に、コマンドの基底クラス関数 `commandCompleted(...)` を呼び出す必要があります。この関数は保留中の変更を PhyreStation にフラッシュします。すると、PhyreStation はワークスペースを更新します。PhyreStationDLL 内の既存のコマンドクラスからユーザのコマンドクラスを派生させた場合には、この関数がすでに呼び出されている可能性があります。

注意 : PhyreStation によるワークスペースの更新は、上記 3 つの関数のいずれかを使って指定されたオブジェクトに対してのみ行われます。指定されたオブジェクトの変更に伴って変更されたオブジェクトは自動更新されません。したがって、それらも手動で指定する必要があります。この手法が採用されているのは、ワークスペースの内容をリフレッシュする際のパフォーマンスを最適化するためです。

コマンドの GUI アイコン

PhyreStation の GUI でコマンドを使用する場合には、そのコマンドを表現するアイコンを用意することをお勧めします。アイコンは、16×16 ピクセルの PNG タイプ (.png) の画像です。このアイコンは、PhyreStationDLL 内でのコマンド登録プロセスの一部として指定されます。

コマンドのアイコンは、PhyreStation のリソースディレクトリ Icons 内に格納します。新しいアイコンを使用するには、PhyreStationDLL() 内でそのアイコンを PhyreStation に登録する必要があります。

コマンドの登録

PhyreStation はその初期化またはスタートアッププロセス中に、使用可能なすべての追加コマンドの情報を知らされる必要があります。登録のタイプによって、コマンドを使用できる場所が決まります。登録のタイプは次のとおりです。

- コンテキスト – GUI のコンテキストメニュー
- スクリプト – Lua スクリプトまたは **Command Window** での実行
- ドラッグ – GUI オブジェクトの選択/ ドラッグ & ドロップ操作
- 名前付き – PhyreStation の内部専用

登録タイプごとに適切なコマンドバイディングクラス（既存のバイディング、ユーザが作成したバイディングのいずれか）を使用します。

大部分のコマンドは、PhyreStationDLL の関数 `PPSSGDll::registerDllCommands()` で登録されます。関数 `PPSSGDll::registerDllCommands()` で採用されているのと同じアプローチを使用することをお勧めしますが、ユーザコードは `PPSSGDll::registerDllUser()` 関数内に配置してください。そうすれば、ユーザコードと付属コードを分離できます。

特定のオブジェクト型に関連付けられたコマンドを作成したが、その型のオブジェクトがワークスペース内にまだ存在していない場合、登録が成功したかどうかをチェックするには次のようにします。

PhyreStation の **About PhyreStation** ダイアログを開き、左側のグループボックスで PhyreStationDLL コンポーネントを選択します。右側のグループボックスに、現在登録されているすべてのコマンドとオブジェクト型の一覧が表示されます。

PhyreEngine™オブジェクトコマンドのリファレンス情報

ここでは、PhyreStation のユーザ独自コマンドを作成する場合に役立つ可能性のある追加情報を提供します。

コマンドの作成

PhyreStationDLL のソースコード (`PhyreStationDLLObjectCmds.h` の `PObjectGetAttributeCmd` クラスなど) を参照してください。

コンパイル、リンク、およびデバッグ

- (1) オプション：新しい PhyreEngine™オブジェクトを作成するためのコードが完成したら、PhyreStationDLL をコンパイルします。すると、PhyreEngine™ユーティリティもコンパイル/リンクされます。
- (2) 新しいコマンドを実行/登録するためのコードが完成したら、PhyreStationDLL を再度コンパイルします。そうすれば、PhyreStation からコマンドを使用できる状態になります。
- (3) 新しい PhyreStationDLL ファイルを PhyreStation のディレクトリにコピーします。

注意：Microsoft Visual Studio の場合、PhyreStationDLL は Release タイプのビルドになります。PhyreStationDLL ライブラリのデバッグを行うには、PhyreStationDLL プロジェクト設定経由で、デバッグを有効にし、コンパイラの最適化をすべて無効にする必要があります。

9 カスタムグループ

この章では、PhyreStation でのカスタムグループの使用方法について説明します。

概要

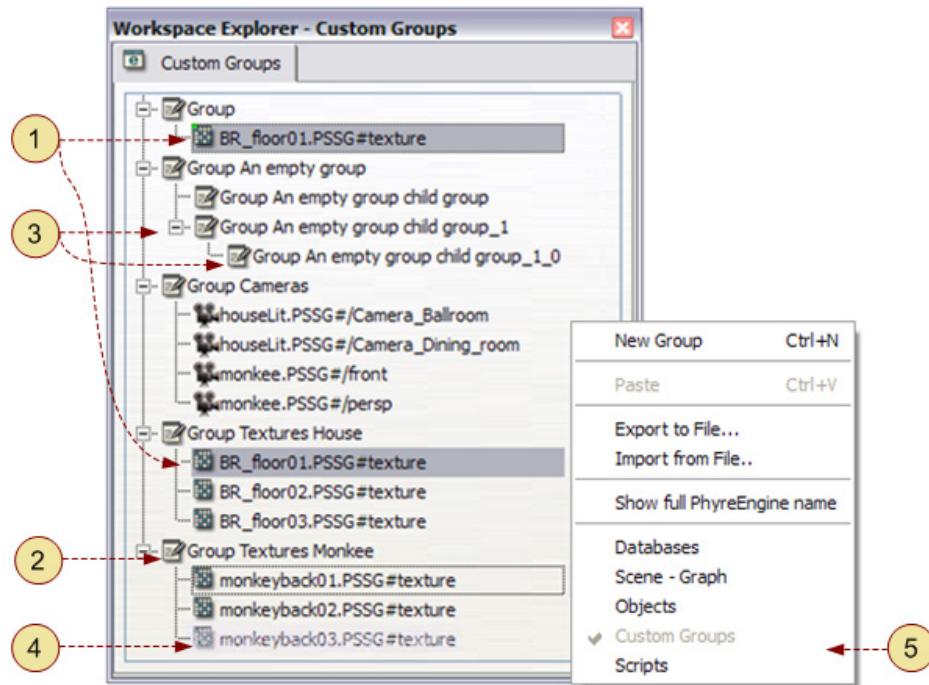
カスタムグループとは、ワークスペースオブジェクトをまとめるユーザ定義可能グループのことです。異なるタイプのオブジェクト（データベース、PhyreEngine™オブジェクト、およびスクリプト）を1つのグループに割り当て、そのグループに名前を付けることができます。これにより、ワークスペース内のオブジェクトの特定のサブセットを操作するたびに、ワークスペース内を繰り返しナビゲートしてそれらのオブジェクトを見つける必要がなくなります。関心のあるオブジェクトをグループ化しておけば、それらのオブジェクトの管理も容易になります。

- 1つのワークスペースに含めることのできるグループの数には制限はありません。
- あるグループを別のグループの子にすることができます。
- 1つのワークスペースオブジェクトを複数のグループ内で表現できます。
- あるワークスペースオブジェクトをグループに割り当てると、そのオブジェクトそのものではなくその「表現」が、グループに格納されます。グループ内のワークスペースオブジェクトの表現は、実際のオブジェクトと同じように見えます。この表現に対する（コマンドによる）作業の実行は、まるでそのオブジェクトが固有の **Workspace Explorer** ビュー内に存在するかのように行えます。
- **Workspace Explorer - Custom Group** ビュー内ではグループに含まれるオブジェクトをコピー、移動、または削除することができますが、そのようにしても、「実際」のオブジェクトには何の変更も加えられません。
- グループは、ワークスペースからエクスポートして別のワークスペースにインポートできます。
- グループはカレントワークスペースの一部であり、適切なプリファレンスを設定することでワークスペース内に保存できます。
- カスタムグループのプリファレンスは、**User Preferences** ダイアログで利用可能になっています。

カスタムグループのブラウズ

カレントワークスペース内のカスタムグループを表示するには、**Workspace Explorer** ウィンドウを開き、ビューのコンテキストメニューから **Custom Groups** を選択します。**Workspace Explorer - Custom Groups** ビューが開きます。

図 46 Workspace Explorer – Custom Groups ビュー



- (1) 表現オブジェクトがハイライト表示されるのは、ユーザが実際のオブジェクトや重複する表現オブジェクトを選択した場合です。緑色の点は、最初に選択されたオブジェクトを指しています。
- (2) ユーザ定義グループ。
- (3) ユーザ定義の子グループ。
- (4) ゴースト表現オブジェクト（実際のオブジェクトがカレントワークスペースに存在しない）。
- (5) ビューのコンテキストメニュー。

複数のオブジェクトを選択できるほか、**Workspace Explorer - Custom Groups** ビューで選択されたオブジェクトを、複数のビューから構成されるより大きなオブジェクト選択セットの一部にすることもできます（項目 1）。

Workspace Explorer - Custom Groups ビューには、オブジェクトの隣に基本的なオブジェクトプロパティを表示できます。これらのプロパティを表示するには、**Edit > User Preferences > Workspace** を選択し、**Enable workspace explorer filter properties** プリファレンスを設定します。詳細については、「3 ワークスペース」の「Workspace Explorer のフィルタプロパティ」を参照してください。

グループ内での PhyreEngine™オブジェクトの識別

カスタムグループに割り当てられるワークスペースオブジェクトの大部分は、PhyreEngine™オブジェクトです。1つの PhyreEngine™オブジェクトを複数のグループ内で表現することができます。表現オブジェクトは、実際のオブジェクトと同じ名前（この場合は PhyreEngine™オブジェクト名）を使用します。このため、同じ名前の実際のオブジェクトが異なる PhyreEngine™データベース内に重複して存在する場合は特に、オブジェクトやその所属先 PhyreEngine™データベースの識別が困難になる可能性があります。**Workspace Explorer - Custom Groups** ビューには、PhyreEngine™オブジェクトの完全な PhyreEngine™オブジェクトリンク ID (`databaseName#objectName` の形式) を表示できます。このオプションを有効にするには、**Workspace Explorer - Custom Groups** ビューのコンテキストメニューから **Show full PhyreEngine™ name** を選択します。

カスタムグループの処理

すべてのカスタムグループ処理は、**Workspace Explorer - Custom Groups** ビューのコンテキストメニュー、グループ内のオブジェクトのコンテキストメニューのいずれかから選択することで実行できますが、そのどちらになるかは、実行するコマンドによります。

グループの作成

カスタムグループを作成するには、次のいずれかを行います。

- **Workspace Explorer - Custom Groups** ビューで右クリックし、コンテキストメニューから **New Group** を選択します。
- キーボード入力 Ctrl-N を使用します。**Workspace Explorer - Custom Groups** ビューにフォーカスがなければなりません。
- 別の **Workspace Explorer** ビューから **Workspace Explorer - Custom Groups** ビューにオブジェクトをドラッグします。

上記アクションのいずれかを実行すると、新しいグループがデフォルト名で表示されます。グループの名前を変更するには、そのラベルをクリックするか、グループのコンテキストメニューから **Rename Group** を選択します。

ワークスペースへのグループの保存

デフォルトでは、カスタムグループはワークスペースに保存されません。カスタムグループを保存し、ワークスペースの次回オープン時にそれらを復元するには、メインメニューから **Edit > User Preferences > Workspace** を選択し、**Remember and restore the custom groups** プリファレンスを選択します。

グループへのオブジェクトの追加

カスタムグループにオブジェクトを追加するには、次のいずれかを行います。

- 別の **Workspace Explorer** ビューからグループ内にオブジェクトをドラッグします。
- 別のグループからこのグループ内に表現オブジェクトをカット&ペースト（コピー&ペースト）します。

追加しようとしているオブジェクトと同じ名前のオブジェクトが、すでにグループ内に含まれている場合、処理が終了します。

グループへのグループの追加

あるグループを別のグループの子として追加するには、次のいずれかを行います。

- グループを右クリックし、コンテキストメニューから **Add Group** を選択します。
- 子グループを親グループにドラッグします。
- グループをカットまたはコピーし、その親グループを選択し、そこに子をペーストします。

グループからの子グループの切り離し

子グループを親グループから切り離すには、子グループのコンテキストメニューから **Detach Branch** を選択します。すると、子グループ（とそのすべてのオブジェクト）が移動して最上位グループになります。グループとすべてのオブジェクトの名前は変更されません。これは、グループをドラッグ&ドロップするための余白がビュー内に残っていない場合に便利です。

グループ間でのオブジェクトのコピー/移動

Workspace Explorer - Custom Groups ビュー内のオブジェクトは複数のグループ内に複製することができますが、その複製された表現は依然として、ワークスペース内の唯一のオブジェクトを表しています。同じオブジェクトの2つの表現を1つのグループに含めることはできません。

同じオブジェクトの別の表現を作成するには、次のいずれかを行います。

- 表現オブジェクトを別のグループにドラッグし、コンテキストメニューから **Copy** を選択します。
- 別の **Workspace Explorer** ビューから同じオブジェクトをドラッグし、それをターゲットグループにドロップします。

表現オブジェクトをグループ間で移動するには、表現オブジェクトのあるグループから別のグループにドラッグします。

グループの名前変更

グループの名前を変更するには、次のいずれかを行います。

- その名前をクリックし、名前を編集します。
- グループのコンテキストメニューから **Rename Group** を選択します。

グループに子グループが少なくとも1つ含まれていると、コンテキストメニューの **Rename Branch** オプションが有効になります。このオプションを使えば、選択されたグループとその子グループの名前を変更できます。

グループの削除

グループを削除するには、次のいずれかを行います。

- グループを選択し、**Delete** キーを押します。
- グループを右クリックし、コンテキストメニューから **Cut** を選択します。
- Remember and restore custom groups** プリファレンス設定のチェックを外し、ワークスペースを閉じます。カスタムグループがワークスペースに保存されません。

グループを削除すると、その子項目も削除されます。実際のオブジェクトはワークスペースから削除されません。他のグループに含まれる、実際のオブジェクトの重複する表現オブジェクトは、削除されません。

グループからのオブジェクトの削除

オブジェクトをグループから削除するには、次のいずれかを行います。

- オブジェクトを選択し、**Delete** キーを押します。
- オブジェクトを右クリックし、そのコンテキストメニューから **Cut** を選択します。

オブジェクトをグループからカットしても、実際のオブジェクトはワークスペースから削除されません。ところが、コンテキストメニューから **Delete** を選択すると、実際のオブジェクトが削除されます。

ワークスペースからのオブジェクトの削除

あるオブジェクトをワークスペースから削除した場合に、そのオブジェクトの表現がカスタムグループ内に存在していれば、次のいずれかの処理が行われます。

- すべての表現オブジェクトが「ゴースト」(つまり、グレー表示)になります。実際のオブジェクトがカレントワークスペースに戻されると、各ゴースト表現も「実質的」な表現に戻されます。ゴースト表現に対してコマンドを実行することはできません。

- すべての表現オブジェクトが **Workspace Explorer - Custom Groups** ビューから削除されます。

この動作を定義するには、メインメニューから **Edit > User Preferences > Workspace** を選択し、**Create ghost item(s) when a workspace object is deleted** プリファレンスを設定します。

グループのインポートとエクスポート

カレントワークスペース内のすべてのカスタムグループをファイルにエクスポートし、それを後で別のワークスペースセッション内にインポートすることができます。グループをエクスポートするには、

Workspace Explorer - Custom Groups ビューのコンテキストメニューから **Export to File...** を選択します。

グループファイルをインポートするには、ビューのコンテキストメニューから **Import from File...** を選択します。 **Custom Groups** ファイルをインポートすると、ワークスペース内のすべてのグループが、そのファイル内のグループで置き換えられます。

10 DNet通信

この章では、PhyreStation の DNet 通信メカニズムについて説明します。ここでは、ネットワーク接続経由でリモートの PhyreEngine™派生アプリケーションに接続し、アプリケーションに対して問い合わせや監視を行ったり、アプリケーションに PhyreEngine™データを送信したりする方法について説明します。

概要

PhyreStation は DNet 通信を使って次のことを行えます。

- PhyreEngine™派生アプリケーション (PhyreEngine™のサンプルプログラム COLLADA Viewer など) に接続し、そのアプリケーションで表示すべき PhyreEngine™カメラシーン情報を渡します。
- Sony PLAYSTATION®3 上の PhyreEngine™アクティブセッションにリモート接続し、PhyreStation のワークスペースへの PhyreEngine™データベースのインポートを、まるでファイルからロードされたかのように行います。

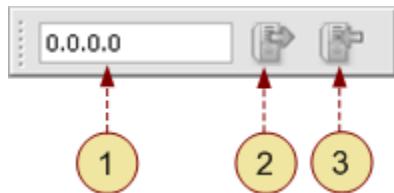
この章では、PhyreEngine™ COLLADA Viewer が、DNet 通信に使用されるアプリケーションであることを前提にしています。それを行えるようにコンパイルされた PhyreEngine™ COLLADA Viewer であれば、Microsoft Windows プラットフォーム、Sony PLAYSTATION®3 Reference Tool のどちらにおいても正常に動作します。

DNet Communicationツールバー

DNet Communication ツールバーは、ネットワーク接続を介して PhyreEngine™派生アプリケーションに接続する場合に使用します。このツールバーには、ホスト IP アドレス用のテキストボックス、**Connect** ボタン、および **Load Remote Database** ボタンが含まれています。

DNet Communication ツールバーを表示するには、メインメニューから **View > Toolbars > DNet** を選択します。

図 47 DNet Communication ツールバー



1. 編集ボックス。ここで指定する情報は次のとおりです。
 - ホスト名 (「localhost」など)
 - IP アドレス
2. Connect ボタン。リモート接続を確立したり、接続を切断したりします
 - 接続確立ボタンアイコン
 - 接続切断アイコン
3. Load Remote Database ボタン

DNetによる接続

DNet 通信を使って接続を確立するには、次の手順に従います。

- (1) ホスト名または有効な IP アドレスを入力すると、接続ボタンが有効になります。
- (2) 接続ボタンをクリックします。接続が確立されると、接続ボタンが接続切断ボタンに変わります。PhyreStation が接続を確立するまで、おそらく数秒かかります。
- (3) リモート接続が不要になったら、接続ボタンを再度クリックします。

注意 : DNet 通信が動作するには、コンパイル時に PhyreEngine™ PUtilityDebug ライブラリがリンクされた PhyreEngine™ セッションが存在している必要があります。

ヒント :

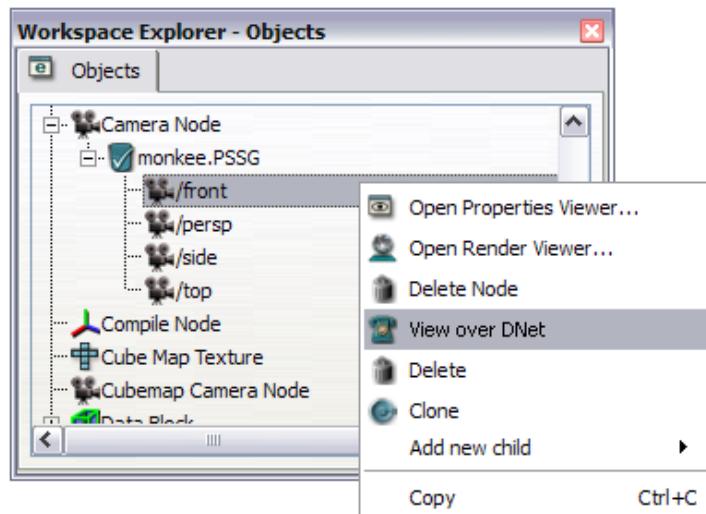
- PhyreStation を使って PhyreEngine™ 派生アプリケーションへの接続を試みる前に、そのアプリケーションが実行されている必要があります。
- Microsoft Windows OS プラットフォーム上では、PhyreEngine™ 派生アプリケーションはデフォルトローカルポートを持っている必要があります。アプリケーションが PhyreStation の前に実行された場合、アプリケーションはデフォルトポートを取得します。たとえば、Microsoft Windows でコンパイルされたバージョンの PhyreEngine™ COLLADA Viewer の場合もそうなります。
- Sony PLAYSTATION®3 をホストとして使用し、そこで SN ProDG Target Manager ソフトウェアを使用している場合には、そのログ出力内で、現在の TTY 接続 IP アドレスを報告するテキストを探します。この IP アドレスを使ってホストへの接続を行います。

DNet経由でのカメラオブジェクトの表示

選択されたカメラオブジェクトの投影シーンを DNet 経由で表示するには、次の手順に従います。

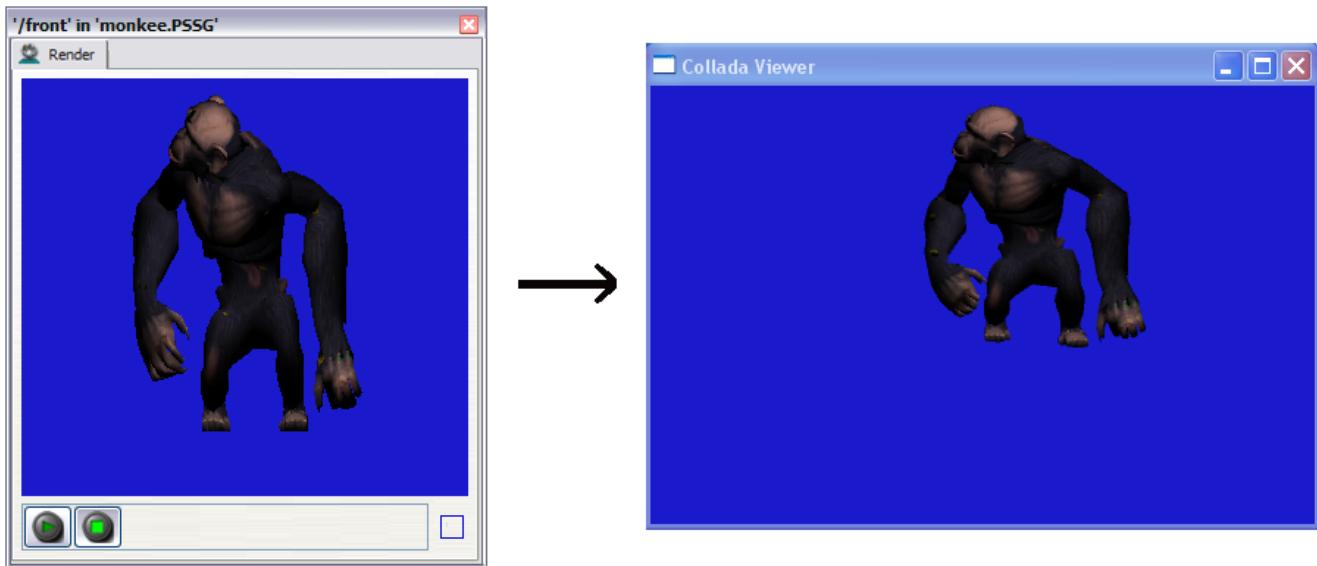
- 前述の説明に従って接続を確立します。ホストへの有効な接続が確立されると、PhyreEngine™ カメラオブジェクトのコンテキストメニューで、**View Over DNet** オプションが有効になります。
- **View Over DNet** を選択することで、必要なデータをホストに転送し、シーンを表示します。PhyreEngine™ COLLADA Viewer の場合、データの受信後、その選択されたカメラに対して PhyreStation **Render Viewer** で表示されていたシーンが表示されます。

図 48 View Over DNet コマンド



PhyreStation の **Render Viewer** で選択/表示されたのと同じカメラ経由でのシーンのビューが、PhyreEngine™ COLLADA Viewer に反映されている様子を、図 49 に示します。

図 49 View Over DNet コマンドによる転送



DNet経由でのリモートデータベースのロード

リモートデータベースをロードするには、次の手順に従います。

- 前述の説明に従って接続を確立します。接続が確立されると、リモートデータベースロードボタンが有効になります。

図 50 Load Remote Database ボタン



- リモートデータベースロードボタンをクリックします。ホスト上で見つかった PhyreEngine™ データベースの一覧が表示されます。
- 必要なデータベースを選択し、それをカレントワークスペース内にロードします。赤色で表示されたデータベースは選択できません。その名前のデータベースがカレントワークスペース内にすでに存在しています。

インポートされたデータベースは、ホストから選択されたデータベースと同じであるか、ほぼ同じです。ホストからデータベースをインポートする際に生じる制限のいくつかを、以下に示します。

- 同じ名前のデータベースがワークスペース内にすでに存在する場合は、データベースをインポートできません。
- リモート接続経由でのみ利用可能なリソースを必要とするデータベースがホストから切断されている場合、そのデータベースは解決できません。そのデータベースは、**Workspace Explorer - Databases** ビューで壊れたデータベースとして表示されます。
- メモリの使用量を最小限に抑えるため、ホストアプリケーションがデータを破棄した可能性があります。このデータは、ワークスペース内や PhyreEngine™ からは利用できません。場合によってはこのことが、一部のシーンのレンダリング方法に影響を与える可能性もあります。

- PhyreStation の **Render Viewer** を使ってシーンをレンダリングすると、そのシーンがホスト上のシーンと異なっている可能性もあります。その原因としては、ホストが実行時にデータを動的に作成していることが考えられます。そのようなデータは次のいずれかになります。

- テクスチャ
- シャドウ
- テキスト

静的なデータのみがホストから転送されます。

- シェーダも、一部のシーンのレンダリング方法の違いの原因となり得ます。シェーダはプラットフォームに固有である可能性があるため、PhyreStation が動作しているプラットフォームとは互換性がない場合があります。

PhyreEngine™はリモートからワークスペース内にロードされたデータベースを解決する際に、まずローカルですべてのリンクデータベースを検索し、その後でリモート接続を検索します。

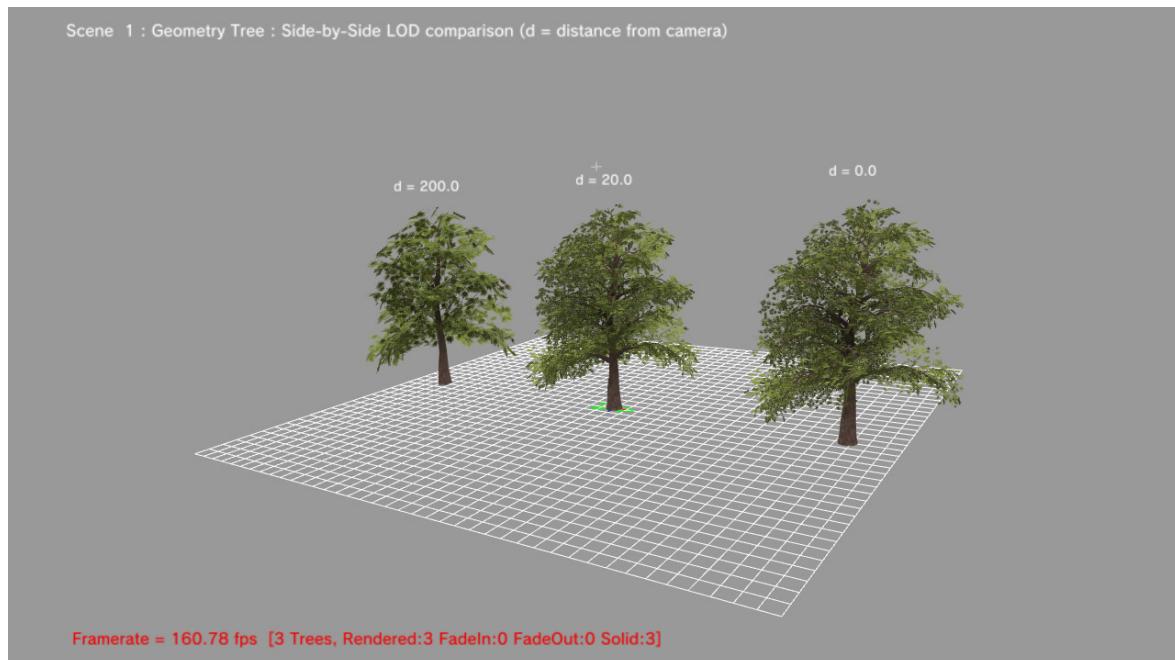
例：リモートデータベースの使用

この例では、DNet を使ってリモートデータベースをロードする方法を示します。

PhyreStation は Sony PLAYSTATION®3 Reference Tool に接続されており、そこで PhyreEngine™のサンプル FoliageRendering ([PhyreEngine]\Samples\Advances\FoliageRendering) が実行されています。

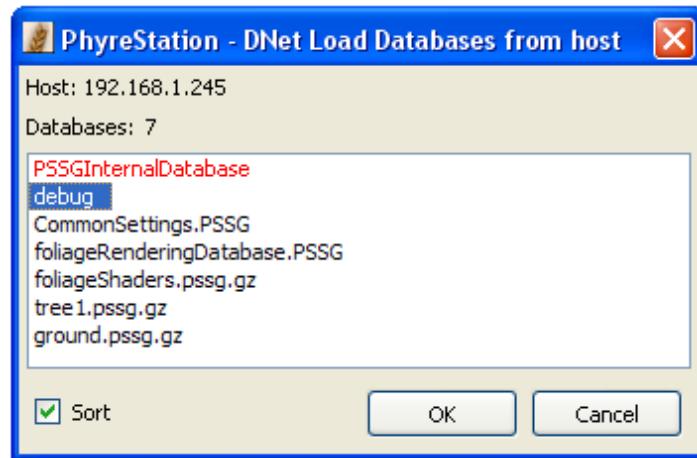
図 51 に、ホスト上で実行されている FoliageRendering サンプルを示します。

図 51 ホスト上で実行されている PhyreEngine™サンプル



Load Remote Database ボタンをクリックすると、ホスト上に現在存在しているデータベースの一覧が表示されたダイアログが開きます。これらのデータベースの一部を PhyreStation にコピーできます。

図 52 利用可能なりモートデータベース



ユーザの要件に応じて、データベースの全部または一部（ただし、赤色で表示されていないもの）を、リストから選択できます。**Workspace Explorer - Databases** ビューに、ワークスペース内のロード済みデータベースが表示されます。

図 53 Workspace Explorer 内のロード済みデータベース

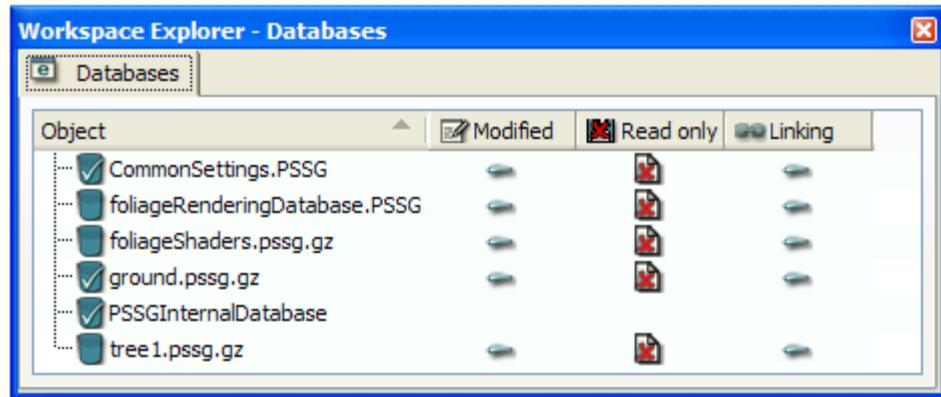
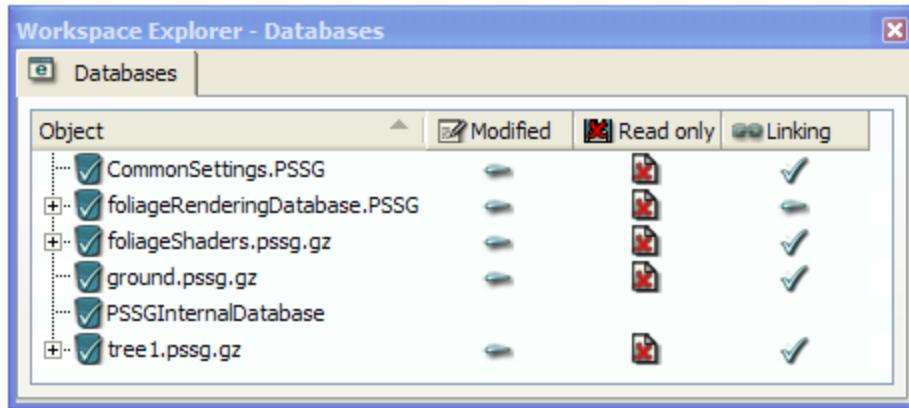


図 53 で、次の点に注意してください。

- 各データベースに **File does not exist** アイコンが表示されています。これは、対応するファイルがローカルに存在していないからです。
- []** アイコンにより、一部のデータベースを解決する必要があることがわかります。解決する必要のあるデータベースを選択し、コンテキストメニューから **Resolve** を選択します。すると、PhyreEngine™が任意のシーンをレンダリングする際に必要とする PhyreEngine™オブジェクトが、ワークスペースに格納されます。

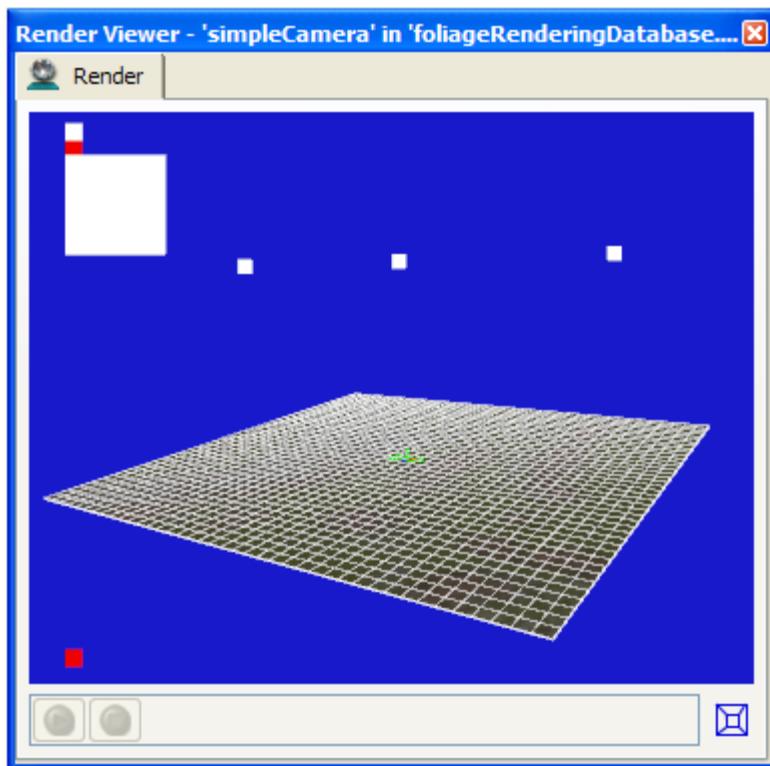
この例では、ワークスペース内のすべてのデータベースを解決します（図 54 を参照）。

図 54 解決済みのデータベース



適切なカメラオブジェクトを選択するか、データベースのコンテキストメニューから **View Scene...** を選択すると、図 55 に示すようなシーンが **Render Viewer** に表示されます。このシーンは、ホスト上で表示されているシーン（図 51 を参照）に一致します。

図 55 Render Viewer に表示されたシーン



ただし、レンダリングされたシーンの細部が、ホスト上で表示されているシーンと非常に異なることに注意してください。これは、ツリー、ツリーのシャドウ、およびテキスト表示はすべて、ホスト上でリアルタイムで動的に作成されているからです。これらの要素を表すオブジェクトは静的に存在していないため、PhyreStation に転送することはできません。PhyreStation にデータベースをコピーする際には常にそうであるとは限りません。それはユーザの環境や要件、およびエンジン/データの構築方法によります。PhyreStation にコピーされたデータベースについては、視覚的な側面以外にもさまざまな側面を確認できます。これは、ユーザの作業やワークフローの結果を評価するうえで非常に役立つ可能性があります。

11 ビューア

概要

PhyreStation で利用可能なビューアは、次のとおりです。

- Render Viewer – カメラノードオブジェクトを表示します
- Segment Set Viewer
- Shader Instances Viewer
- Texture Viewer
- Shader Program Viewer
- Shader Group Viewer

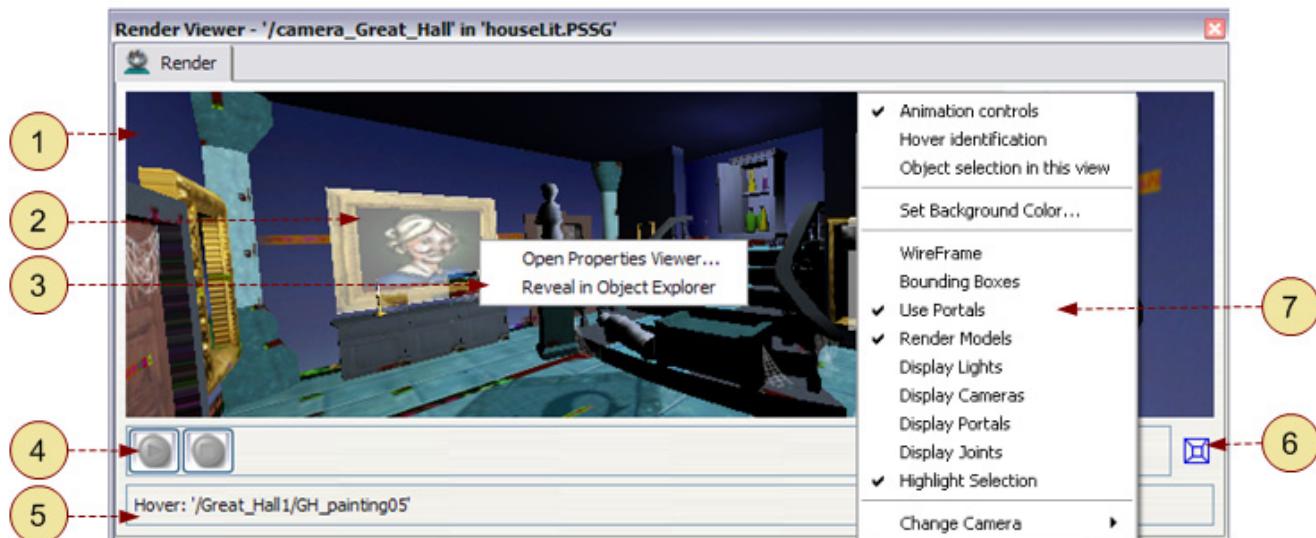
Render Viewer

Render Viewer は現在の PhyreEngine™シーンを、特定の PhyreEngine™カメラノードオブジェクトの視点から表示します。カメラが間違った場所に配置されていたり間違った方向を向いていたりすると、何も表示されない可能性があります。カメラの向きや位置を操作するには、キーボードコントロールを使用するか、カメラオブジェクトの属性値を変更します。

ヒント：シーン全体を表示したり、シーン内の特定のノードを表示したりする場合は、**Database > View Scene**、**Node > View Scene Node** のいずれかの機能を使用します。これらのオプションを選択しても、データベース内のオブジェクトの状態は一切変更されません。

カメラノードオブジェクトを **Render Viewer** で表示するには、カメラオブジェクトのコンテキストメニューから **Open Render Viewer...** を選択します。

図 56 Render Viewer



1. 選択された PhyreEngine™カメラノードオブジェクトの視点から PhyreEngine™によってレンダリングされたシーン。

2. 選択された PhyreEngine™オブジェクトがハイライト表示で描画されています（ハイライト表示はオプション）。
3. 選択されたシーンオブジェクトのコンテキストメニュー。
4. アニメーションコントロールボタン（表示はオプション）。
5. マウスカーソルが置かれている PhyreEngine™シーンオブジェクトの名前（表示はオプション表示）。
6. シーンのレンダリングに使用されているカメラのタイプ（表示はオプション）。
7. ビューのコンテキストメニュー（ビューの余白部分でマウスを右クリック。必要に応じて **Ctrl** キーを使用）。

Render Viewer のコンテキストメニュー

Render Viewer のコンテキストメニュー（項目 7）に含まれる表示オプションは、次のとおりです。

コンテキストメニューのオプション（PhyreStation 内部）

- **Animation controls** – アニメーションコントロールの表示/非表示を切り替えます。
- **Hover identification** – Hover インジケータの表示/非表示を切り替えます。Hover インジケータには、マウスのホーバー先となっている PhyreEngine™オブジェクトの名前が表示されます。
- **Object selection in this view** – マウスの左ボタンのクリックでシーンオブジェクトを選択する機能のオン/オフを切り替えます。これにより、他のウィンドウのように新たに PhyreEngine™オブジェクトを選択することなく、シーンをナビゲートすることが可能になります。
- **Set Background Color...** – カラー選択ダイアログを表示します。
- **Change Camera** – シーンの別の PhyreEngine™カメラビューに変更します。（このオプションを選択するとビューの対象が変わること）。

注意：大きなシーン上でマウスを動かす場合の Hover インジケータの更新処理には、非常に時間がかかる可能性があります。これは非常に負荷の高いタスクであるため、Hover インジケータが不要な場合は表示しないでください。

コンテキストメニューのオプション（PhyreStationDLL で定義）

次の各オプションは、シーンの各レイヤのレンダリングを制御します。

- **WireFrame** – ワイヤフレーム表示とサーフェス表示の切り替えを行います。
- **Bounding Boxes** – 有効化された場合、すべての PhyreEngine™ノードオブジェクト型に対して赤色のバウンディングボックスを描画します。黄色のバウンディングボックスは、オブジェクトが選択の一部であることを示します。
- **Use Portals** – 可視テストにポータルを使用するかどうかを切り替えます。
- **Render Models** – シーンの描画のオン/オフを切り替えます。
- **Display Lights** – シーン内の光源の表示/非表示を切り替えます。
- **Display Cameras** – シーン内の他のカメラの表示/非表示を切り替えます。
- **Display Portals** – ポータルの表示/非表示を切り替えます。
- **Display Joints** – 関節の表示/非表示を切り替えます。
- **Highlight Selection** – 有効化された場合、PhyreEngine™オブジェクトが選択の一部になっていることがわかるように、シーンオブジェクトが露出過度であるように再描画されます。

注意 :

- **Render Viewer** ビューの PhyreStationDLL コンテキストメニュー オプションの中に、現在レンダリングされているシーンに適さないものがあつても、PhyreStation はそのオプションを無効化しません。その場合でもコマンドは実行可能ですが、処理は失敗する可能性があります。
- **Render Viewer** ビューの PhyreStationDLL コンテキストメニュー オプションはカスタマイズ可能であるため、本ガイドで説明しているオプションとは異なる可能性があります。
- 利用可能な PhyreStationDLL 定義オプションは、上記のオプションと異なる可能性があります。

シーンオブジェクトのコンテキストメニュー

Render Viewer でシーンオブジェクトを選択できるようにするには、まず **Render Viewer** コンテキストメニューの **Object selection in this view** オプションを有効にする必要があります。**Render Viewer** でオブジェクトを選択すると、他のすべてのワークスペース ウィンドウでもそれらのオブジェクトが必要に応じてハイライト表示されます。

選択されたオブジェクトを右クリックすれば、次のコンテキストメニュー オプションを表示できます。

- **Show Properties Viewer...** は、選択されたオブジェクトの静的な **Properties Viewer** ウィンドウを表示します。静的/動的な **Properties Viewers** については、「動的モードと静的モード (5 PhyreEngine™ オブジェクト)」を参照してください。
- **Reveal in a Workspace Explorer** は、選択されたシーンオブジェクト（複数可）を適切な **Workspace Explorer** ビューで表示します。

アニメーションコントロール

アニメーションデータの制御は、アニメーションコントロール ボタンを使って行えます。



アニメーション
を再生します



アニメーション
を停止します

これらのボタンが有効になるのは、カメラと同じ PhyreEngine™ データベース内で、次の PhyreEngine™ オブジェクトが利用可能な場合だけです。1つのアニメーション セット、1つのアニメーション、1つのアニメーション チャンネル、および少なくとも 2 つのアニメーション チャンネル データ ブロック（時間ブロックと値ブロック）。

注意 :

- ある **Render Viewer** ウィンドウの再生状態を変更した場合に、その変更が適用されるのは、他の **Render Viewer** ウィンドウのうち、同じ対象を表示しているものだけです。たとえば、ある **Render Viewer** で **Stop animation** ボタンをクリックすると、それと同じデータベース内のシーンを表示している **Render Viewer** ウィンドウでしかアニメーションが停止されません。
- アニメーションを再生するとデータベースの状態が変化します。データベースに新たに追加されるオブジェクトもあれば、値が変更されるオブジェクトもあります。

カメラタイプインジケータ

選択された PhyreEngine™ カメラオブジェクトは、シーンに対して透視投影、平行投影のいずれかを行えます。これにより、カメラの移動に使われるナビゲーションコントロールが決まります。**Render Viewer** ウィンドウの下のほうの右隅に表示されたカメラタイプインジケータは、現在のカメラタイプを示します。

- 平行投影タイプのカメラ
- 透視投影タイプのカメラ

ナビゲーションコントロール

シーンのカメラビューを変更するには、**Alt** キーを押しながら、表 7 の説明に従ってマウスのボタンを使用します。

表 7 Render Viewer のコントロール

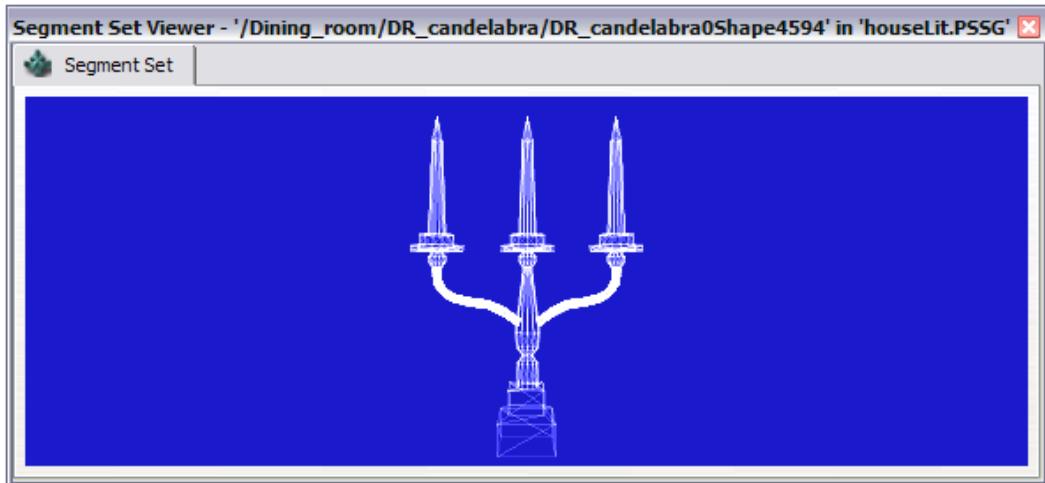
マウスボタンの 押下	カメラ	マウスを動かしたときの効果
左	<input type="checkbox"/>	ビューポートをズームして画像を拡大します。
	<input checked="" type="checkbox"/>	カメラをスポット上で回転させます（つまり見回す）。 シーン内の 1 つまたは複数のオブジェクトが選択されている場合には、 そのオブジェクトの周り、または複合オブジェクトの中心の周りをカメ ラが回転します。
中央	<input type="checkbox"/>	シーンに対してカメラを垂直または水平方向に平行移動させます。
	<input checked="" type="checkbox"/>	シーンに対してカメラを垂直または水平方向に平行移動させます。
右	<input type="checkbox"/>	カメラを前後に平行移動させます。 平行投影で表示している場合には、おそらく何の効果も持たないように 見えますが、 Properties Viewer では移動が確認できます。
	<input checked="" type="checkbox"/>	カメラを前後に平行移動させます。

注意：ナビゲーションコントロールを使ってカメラを操作するとそのカメラオブジェクトの状態が変わるために、データベースの状態も変わります。一時 PhyreEngine™ ワークスペースカメラオブジェクトは例外です。

Segment Set Viewer

セグメントセットオブジェクトを **Segment Set Viewer** に表示するには、オブジェクトのコンテキストメニューから **Open Segment Set Viewer** を選択します。

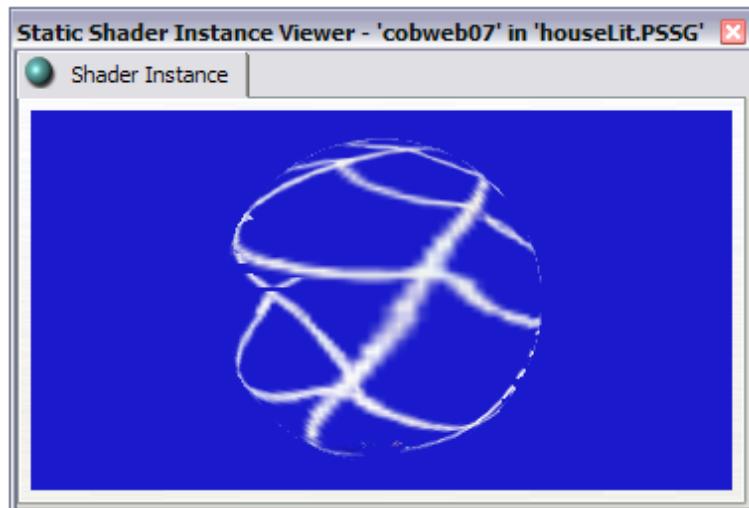
図 57 Segment Set Viewer



Shader Instance Viewer/Shader Instances Viewer

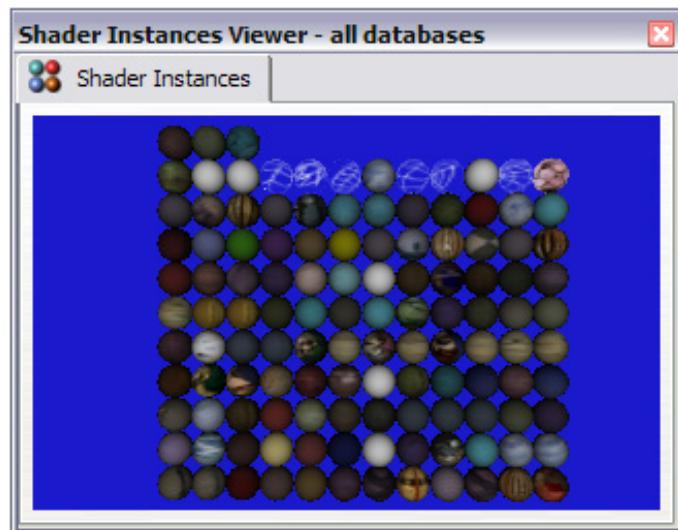
単一のシェーダインスタンスオブジェクトを **Shader Instance Viewer** に表示するには、オブジェクトのコンテキストメニューから **Open Shader Instance Viewer** を選択します。

図 58 Shader Instance Viewer – 単一のオブジェクト



ワークスペース内のすべてのデータベースに含まれるすべてのシェーダインスタンスオブジェクトを表示するには、任意のシェーダインスタンスオブジェクトのコンテキストメニューから **Open Shader Instances Viewer** を選択します。このビューアは、シェーダインスタンス同士の比較を行う際に役立ちます。

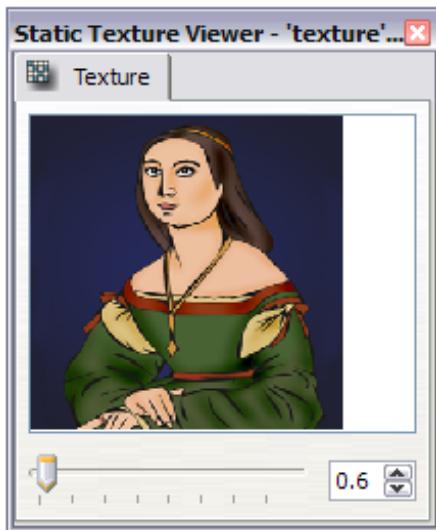
図 59 Shader Instances Viewer – すべてのオブジェクト



Texture Viewer

テクスチャオブジェクトを **Texture Viewer** に表示するには、オブジェクトのコンテキストメニューから **Open Texture Viewer** を選択します。

図 60 Texture Viewer



ズームコントロール

Texture Viewer の画像のズームイン/ズームアウトを行うには、次のいずれかを行います。

- ズームスライダコントロールを使用します。
- スピンコントロールボックスでズーム係数を編集します。ズームの初期値は、ビューアのウィンドウ内に画像がぴったりと収まる値に設定されます。
- **Ctrl** キーを押してマウスのホイールを回転させます。

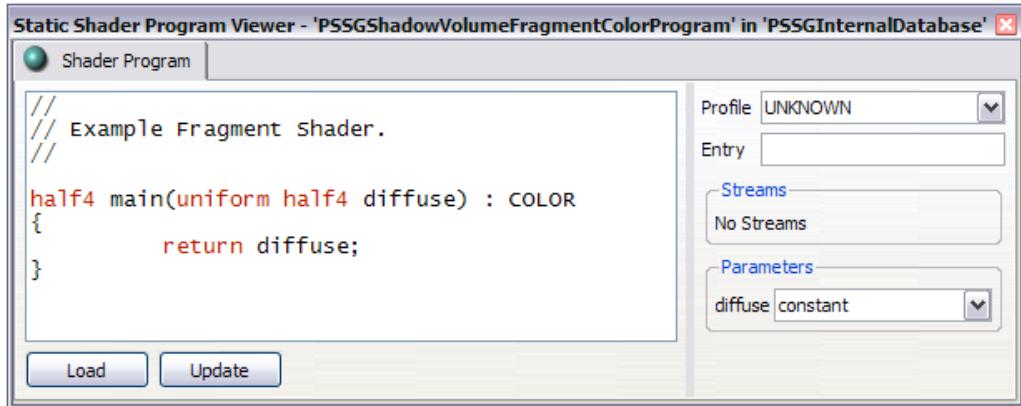
Texture Viewer の倍率範囲は、画像原寸の 0.1~16 倍です。倍率値 1.0 の場合、画像は原寸で表示されます。

Shader Program Viewer

シェーダプログラムオブジェクトのプログラム/パラメータ情報を表示するには、オブジェクトのコンテキストメニューから **Open Shader Program Viewer...** を選択します。

このオブジェクトで別のプログラムを選択するには、**Load** ボタンをクリックします。シェーダプログラムのパラメータに加えた変更をすべてコミットするには、**Update** ボタンをクリックします。

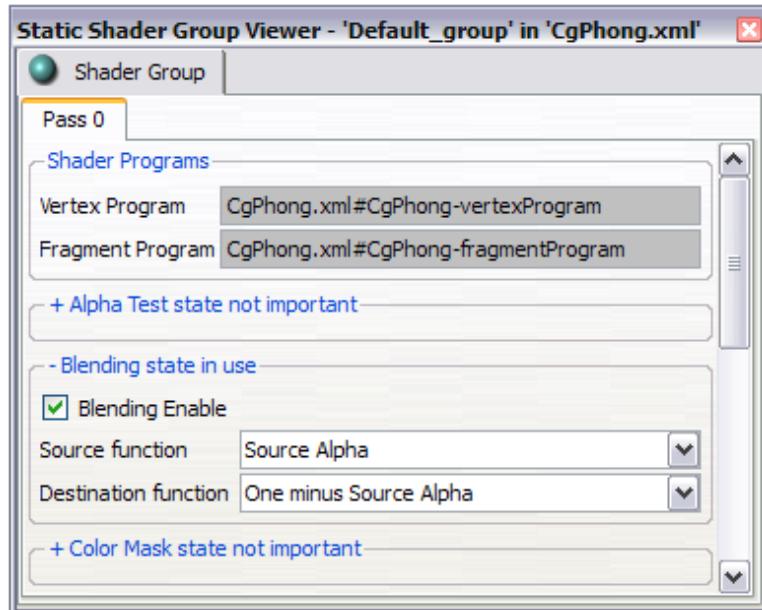
図 61 Shader Program Viewer



Shader Group Viewer

シェーダグループオブジェクトの情報を **Shader Group Viewer** に表示するには、オブジェクトのコンテキストメニューから **Open Shader Group Viewer...** を選択します。

図 62 Shader Group Viewer



Shader Group Viewer でパラメータに加えた変更はすべて、すぐに有効になります。**Render Viewer** で対象を表示すれば、その変更の効果を確認できます。

いずれかのグループボックス、たとえば **Alpha Test state** グループボックスを折りたたむと、その処理が「in use (使用中)」から「not important (重要でない)」に変更され、PhyreEngine™はその処理をパスに含めないように指示されます。グループボックスを展開すると、処理がパスに含まれられます。この設

定は、処理の **Enable** チェックボックスで示される有効/無効状態とは区別されます。たとえば、ある処理をパスに含めるが有効にはしない、といったことも可能です。

12 Graph Editor

概要

PhyreStation には、次の 3 つのグラフ（またはネットワーク）エディタが存在しています。

- ターゲットブレンダエディタ – アニメーションターゲットブレンダオブジェクト
- モディファイアネットワークエディタ
- モディファイアネットワークインスタンスエディタ

これらの Graph Editor を使って、ユーザは PhyreEngine™オブジェクトとプロセス間の関係を表示/編集します。すべての Graph Editor は共通の動作をいくつか共有していますが、相違点もいくつか存在しています。

- グラフの共通動作は PhyreStation によって提供されており、変更不可能です。
- 個々の Graph Editor に固有の動作は、PhyreStationDLL に用意されたクラスインターフェースによって制御されます。これらの動作は変更可能ですが。使用する Graph Editor の種類によっては、このインターフェースによって個別動作が指定される可能性があります（有効または無効な接続を示すために接続線の色を変更するなど）。

共通のGraph Editor動作

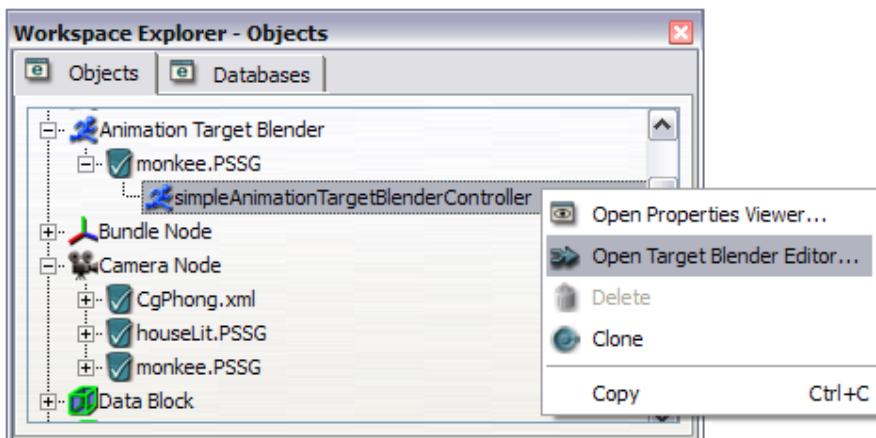
ここでは、**Target Blender Editor** を例として取り上げながら、Graph Editor のインターフェースや基本機能について説明します。ここで説明するインターフェースや機能は、すべての Graph Editor に共通したものです。

Graph Editor を開く

Target Blender Editor を開くには、次の手順に従います。

- (1) **Workspace Explorer Objects** ビューを開きます。
- (2) アニメーションターゲットブレンダオブジェクトを右クリックし、コンテキストメニューから **Open Target Blender Editor...** を選択します。**Target Blender Editor** が開きます。

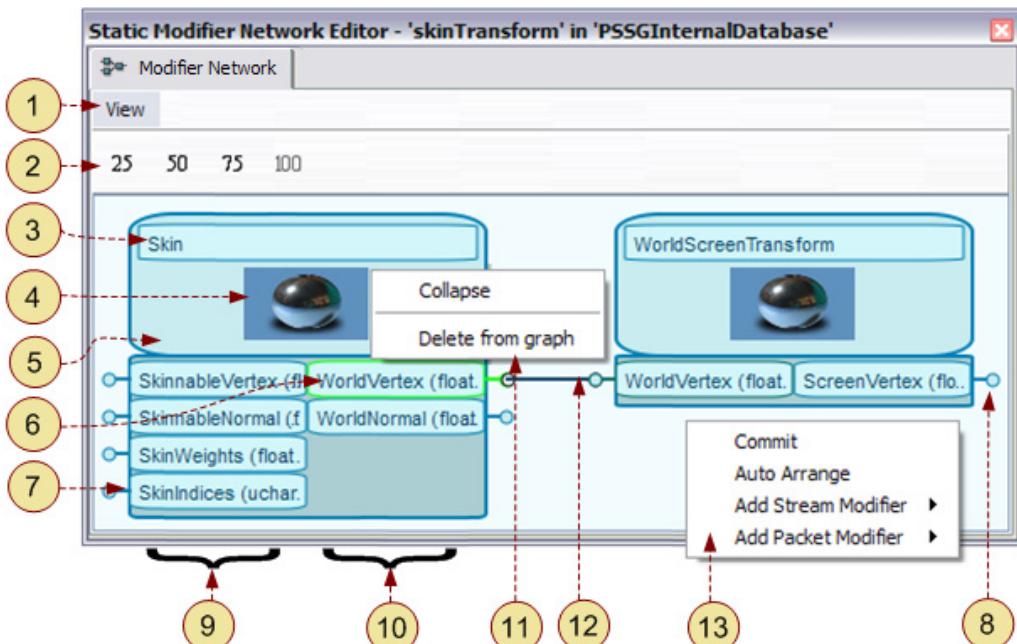
図 63 Target Blender Editor を開く



- (3) 関連する PhyreEngine™データベース（複数可）に変更を適用するには、エディタのコンテキストメニューから Commit を選択します。

Graph Editor のインターフェース

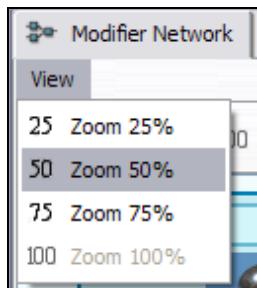
図 64 Graph Editor の機能



1. メニューバー
2. ツールバー
3. グラフノードのラベル：通常は PhyreEngine™オブジェクトまたはプロセスの名前
4. グラフノードのアイコン：PhyreEngine™オブジェクトまたはプロセスを表す画像
5. グラフノード
6. ノードコネクタ（出力）。緑色でハイライト表示されていますが、これは、このコネクタが接続済みであることを示します
7. ノードコネクタ（入力）。未接続状態です
8. ノードコネクタ（出力）へのハンドル
9. 一連の入力ノードコネクタ
10. 一連の出力ノードコネクタ
11. グラフノードのコンテキストメニュー
12. 2つのノードコネクタ間でデータフローが生じる可能性があることを示す線
13. 背景のコンテキストメニュー

メニューバー

Graph Editor には単一のメニュー (**View** メニュー) が用意されています。このメニューを使って背景領域の現在のズームを設定します。

図 65 Graph Editor の View メニュー

ヒント: Alt キーを押しながらマウスを水平方向に動かしても、ズームを起動できます。

コンテキストメニュー

コンテキストメニューを表示するには、背景領域かグラフノードで右クリックします。

- 背景コンテキストメニュー（図 64 の項目 13）には、グラフ全体に影響を与える処理の一覧が表示されます。
- ノードコンテキストメニュー（図 64 の項目 11）には、グラフノードに固有のオプションの一覧が表示されます。複数のノードに対して 1 つのコンテキストメニュー命令を選択した場合、そのコマンドの有効なターゲットでないノードはすべて無視されます。

コンテキストメニューの処理は PhyreStation、PhyreStationDLL のいずれかによって登録されます。

PhyreStationDLL プログラマは、PhyreStationDLL の情報を変更、拡張、公開、または隠蔽することができます。

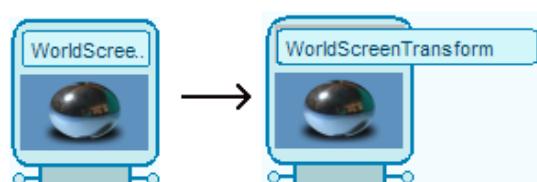
グラフノード

概要

エディタ内では、オブジェクトやプロセスはグラフノードとして表されます。ノードはノードコネクタ経由で互いにリンクすることができます。これらのコネクタは入力と出力に分類されます。入力はデータがノード内に入り込むことを意味し、出力はデータがノードから出て行くことを意味します。

Graph Editor では決まりとして、データフローを左から右に示すことになっています。入力ノードコネクタはグラフノードの左側に、出力ノードコネクタは右側に、それぞれ表示されます。

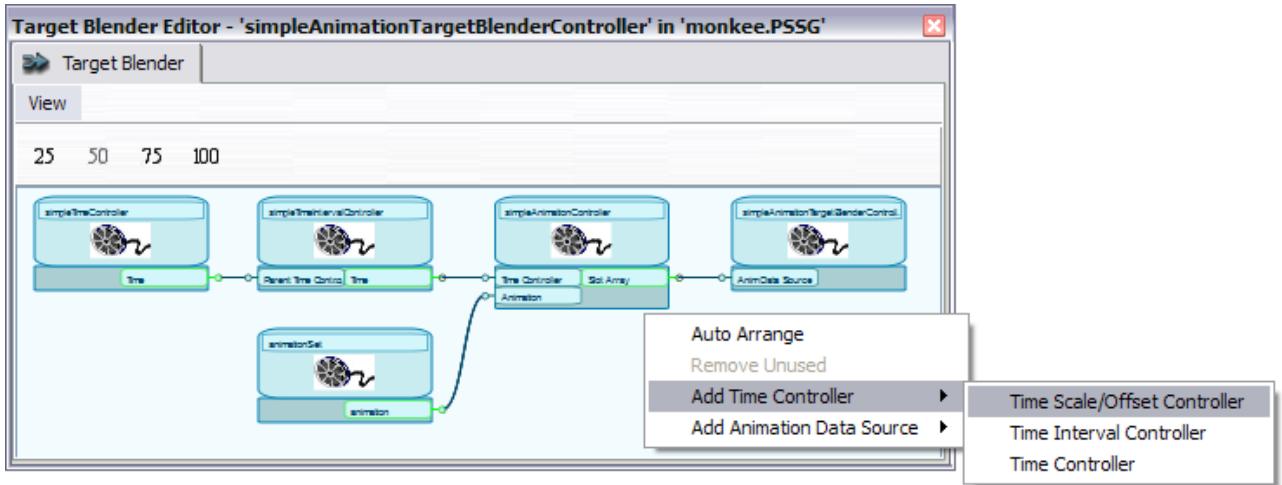
ノードやコネクタのラベルは、ノードの幅に合わせて切り詰められます。その全体を表示するには、ノードまたはコネクタのラベルの上でマウスを停止させます。

図 66 ノードラベルの表示

ノードの追加

グラフノードを追加するには、背景領域で右クリックし、コンテキストメニューから必要なタイプを選択します。

図 67 新しいグラフノードの追加



ノードの移動

グラフノードを移動するには、ノードを目的の位置にドラッグします。ドラッグ中は、ノードの接続線がハイライト表示されます。

データフローが左から右になるようにノードをグリッド形式で整列させるには、エディタのコンテキストメニューから **Auto Arrange** を選択します。接続のないノードは背景領域の下部に配置されます。

ノードの削除

グラフノードを削除するには、コンテキストメニューから **Delete** を選択します。**Delete** オプションは、依存関係を持つノードでは無効化されます。

必須でないすべての未接続ノードをグラフから削除するには、コンテキストメニューから **Remove Unused** オプションを選択します。

ノードのサイズ変更

グラフノードをサイズ変更すれば、背景領域により多くのスペースを作ることができます。グラフノードのサイズ変更は、4つの折りたたみ状態を通じて行われます。次の折りたたみ状態に遷移するには、ノードのコンテキストメニューから **Collapse** を選択します。

図 68 グラフノードの折りたたみ状態の遷移

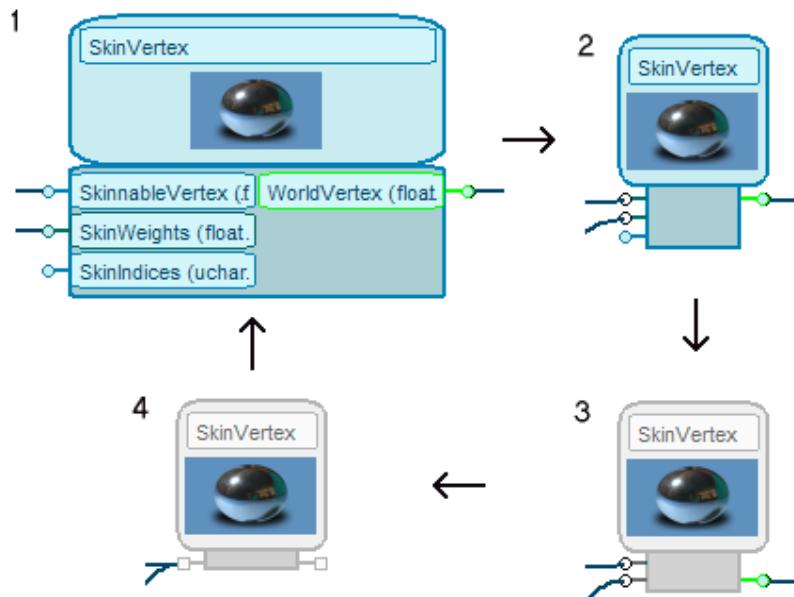


表 8 では、これらの折りたたみ状態について詳しく説明します。

表 8 グラフノードの折りたたみ状態

状態	解説
1	グラフノードがフルサイズで表示され、コネクタもその接続状態にかかわらず、すべて表示されます。
2	ノードを表示する際にすべてのコネクタが表示されますが、コネクタのラベルは表示されません。
3	ノードを表示する際に、接続されているコネクタのみが表示されます。この状態ではノードを編集できません。
4	ノードを表示する際に、すべての入力コネクタが結合され、すべての出力コネクタが結合されます。この状態ではノードを編集できません。

ノードの検索

Workspace Explorer - Objects ビューで特定のノードを検索するには、ノードのコンテキストメニューから **Find in Workspace Explorer** を選択します。

ノードの接続

グラフノードには、入力側と出力側で任意の数のコネクタを持たせることができます。あるノードの各入力コネクタは、他のノードの出力コネクタからの接続しか受け入れることができません。同様に、あるノードの各出力コネクタは、他のノードの入力コネクタにしか接続できません。

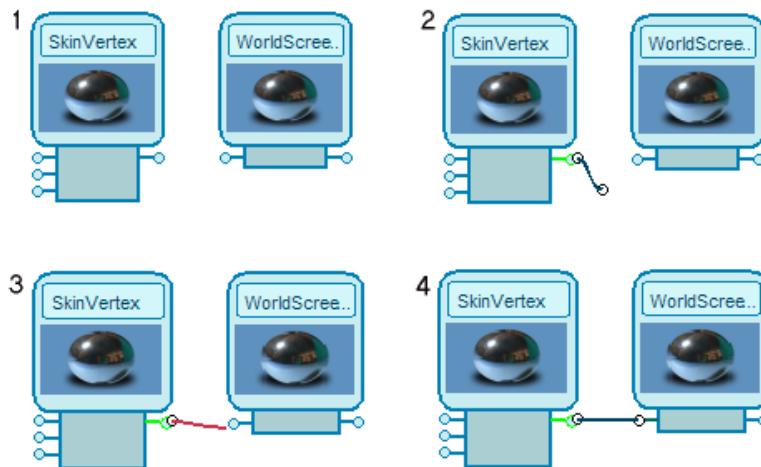
接続を行うには次の手順を実行します（図 69 を参照）。

- (1) マウスの左ボタンで、あるグラフノードのソースコネクタをクリックします。（これは入力コネクタ、出力コネクタのどちらでもかまわない。）
- (2) 別のノードのターゲットコネクタにマウスをドラッグします。（これは入力コネクタ、出力コネクタのどちらでもかまわない。）PhyreStationDLL 実装への問い合わせが行われ、接続の有効性がチェックされます。接続が有効な場合は接続線が赤色に変わり、そうでない場合は黒色のままになります。

(3) マウスのボタンを離して接続を作成します。

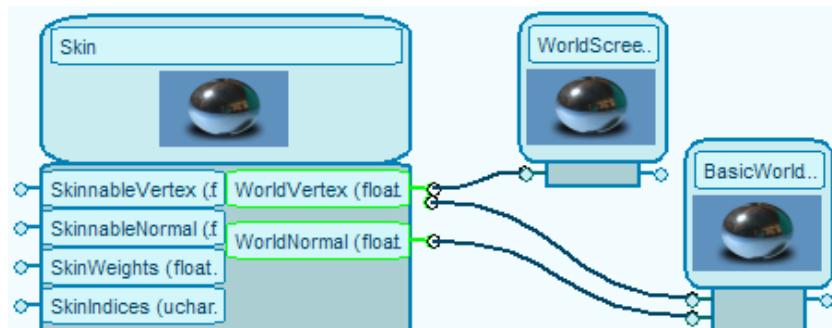
注意 : Graph Editor の種類によっては、**Commit** コマンドが発行されるまで、新しい接続が PhyreEngine™データベースに保存されない可能性があります。

図 69 Graph Editor のノードの接続



PhyreStationDLL 実装でサポートされていれば、1 つのノードコネクタを複数のノードに同時に接続することもできます。その一例を図 70 に示します。

図 70 複数のノードとの接続



接続の削除

接続を削除するには、2 つのノードコネクタのいずれかをクリックし、リンク線をドラッグして引き離し、それを余白部分にドロップします。マウスの左ボタンを押し下げた後、少し間をおいてからドラッグを実行する必要があります。そうすれば、ノードをワークエリアのあちこちに移動する際にリンクを間違って削除してしまう危険性が少なくなります。

個別のGraph Editor動作

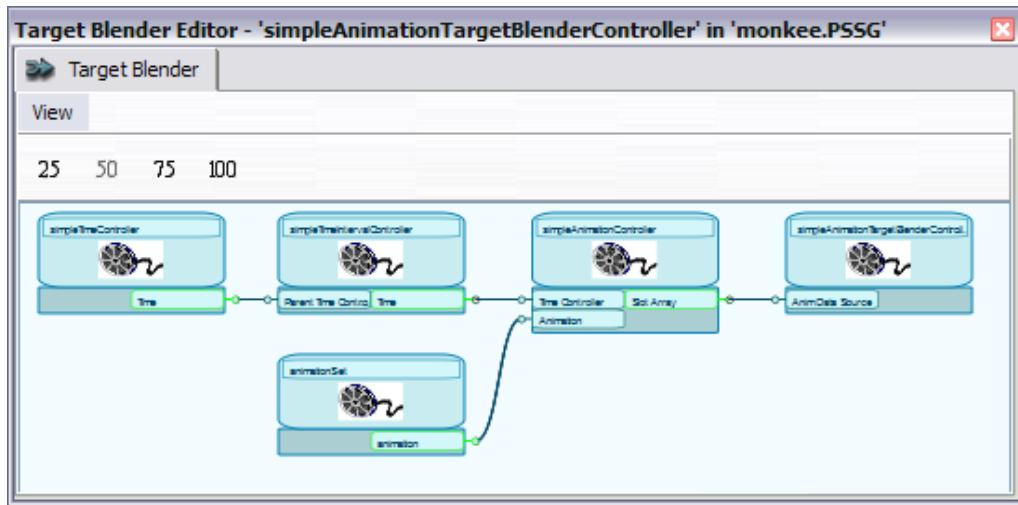
Target Blender Editor

Target Blender Editor は、アニメーションターゲットブレンダ PhyreEngine™オブジェクトの表示/編集に使用されます。

注意 : アニメーションターゲットブレンダは、関連するアニメーションが（同じデータベース内で）再生されるまで、**Workspace Explorer - Objects** ビュー内に表示されません。

アニメーションを再生するには、対応するカメラオブジェクトの **Render Viewer** ウィンドウを開き、**Play** ボタンをクリックします。

図 71 Target Blender Editor



Target Blender Editor 内のグラフノードは、アニメーションターゲットブレンダと一緒にタイムコントローラ/アニメーションデータソースを表します。

新しいタイムコントローラやアニメーションデータソースをグラフに追加するには、背景領域のコンテキストメニューから **Add Time Controller/Add Animation Data Source** を選択します。

注意 : **Target Blender Editor** を使って行った変更はすべて、対応する PhyreEngine™データベースに即座に適用されます。

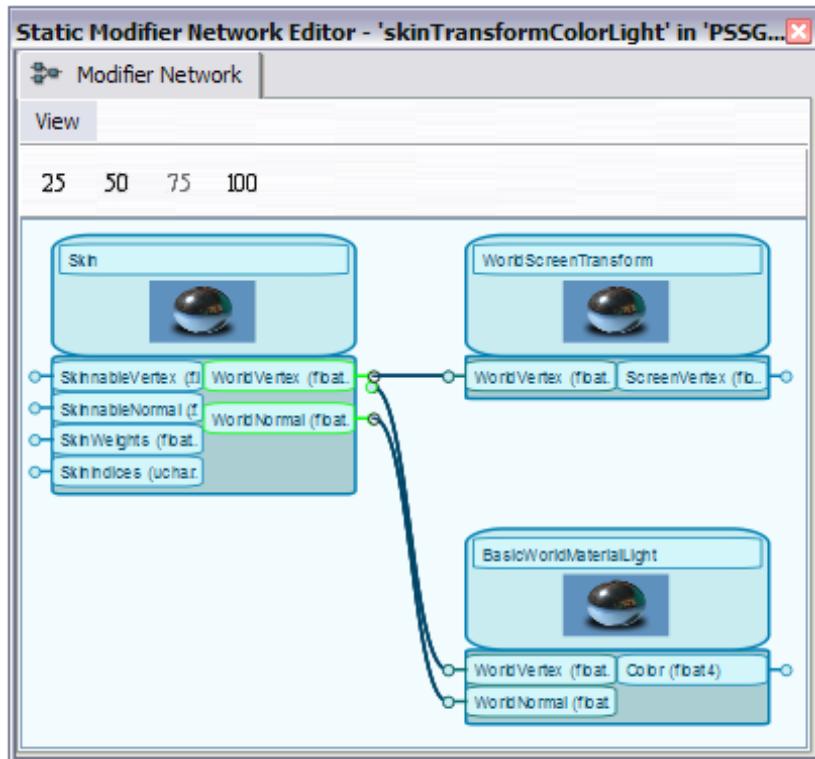
アニメーションタイムコントローラ/データソースの詳細については、「PhyreEngine™プログラミングガイド」を参照してください。

Modifier Network Editor

Modifier Network Editor は、モディファイアネットワーク PhyreEngine™オブジェクトの表示/編集に使用されます。

注意：モディファイアは PhyreEngine™オブジェクト型ではなくプロセスであるため、**Workspace Explorer - Objects** フィルタビューには表示されません。モディファイアは、**Modifier Network Editor** を使わないと編集できません。

図 72 Modifier Network Editor



Modifier Network Editor のグラフノードはモディファイアを表します。

新しいストリームモディファイアやパケットモディファイアをグラフに追加するには、エディタのコンテキストメニューから **Add Stream Modifier/Add Packet Modifier** を選択します。

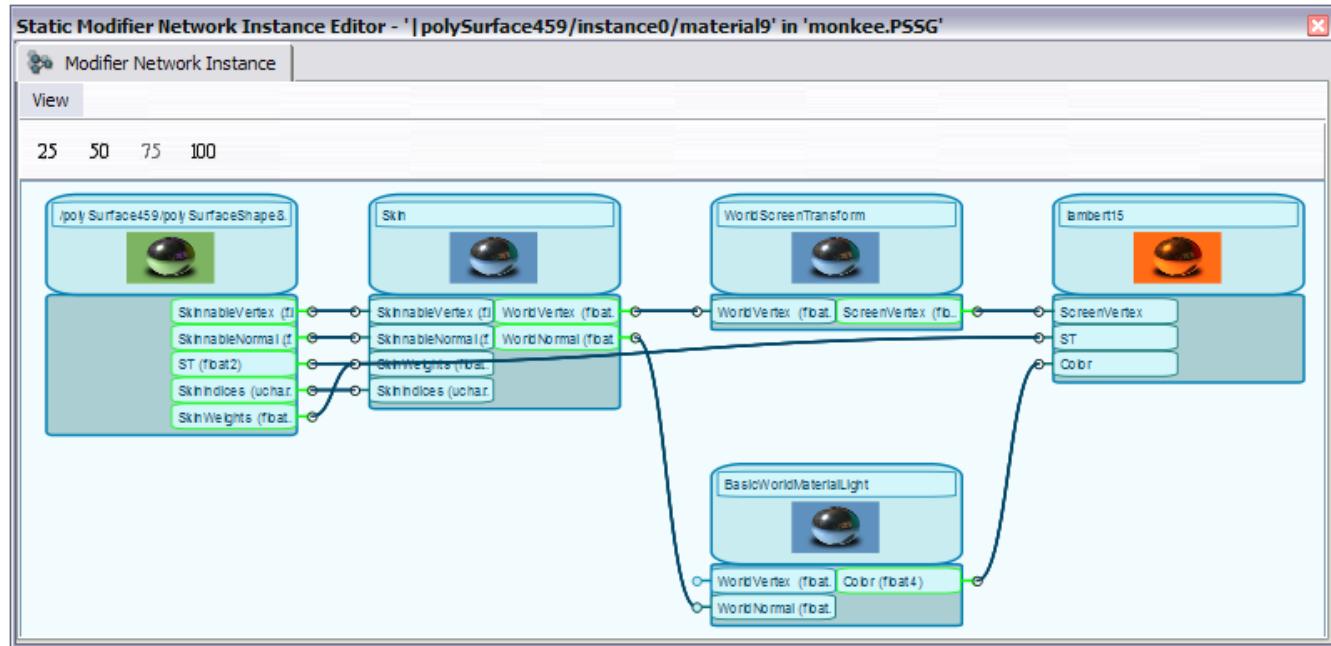
注意：Modifier Network Editor を使って行われた変更はすべて、Commit コンテキストメニューの「オプション」が選択されるまで適用されません。対応するモディファイアネットワークインスタンスオブジェクトは、それまで何の影響も受けません。

モディファイアの詳細については、「PhyreEngine™プログラミングガイド」を参照してください。

Modifier Network Instance Editor

Modifier Network Instance Editor は、モディファイアネットワークインスタンス PhyreEngine™オブジェクトの表示/編集に使用されます。

図 73 Modifier Network Instance Editor



Modifier Network Instance Editor 内のグラフノードはレンダーデータソースを表しており、緑色の画像を持ちます。(モディファイアは青色、シェーダインスタンスはオレンジ色です。)

新しいストリームモディファイアやパケットモディファイアをグラフに追加するには、背景領域のコンテキストメニューから **Add Stream Modifier/Add Packet Modifier** を選択します。

グラフ内で表現されているレンダーデータソースオブジェクトまたはシェーダインスタンスオブジェクトがこのエディタの外側で変更されるたびに、その変更されたばかりのオブジェクトのプロパティが対応するノードにも反映されるように自動更新されます。これは、モディファイアを表すノードには当てはまりません。なぜなら、モディファイアに対応する PhyreEngine™オブジェクトは存在しないからです。

注意 : **Modifier Network Instance Editor** 内で行われた変更はすべて、**Commit** コンテキストメニュー オプションが選択されるまで適用されません。

モディファイアの詳細については、「PhyreEngine™プログラミングガイド」を参照してください。

13 Animation Editor

この章では、PhyreStation の波形エディタ – **Animation Editor**について説明します。

概要

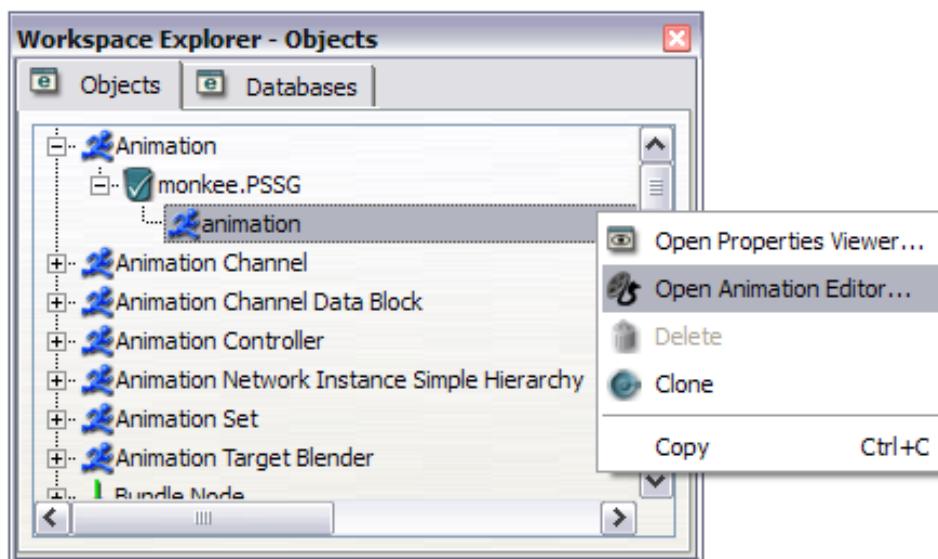
Animation Editor は、PhyreEngine™アニメーションオブジェクト型に関連付けられた波形エディタです。これを使えば、アニメーションのアニメーションチャンネルに含まれるアニメーションデータを表示/編集できます。

Animation Editorを開く

アニメーションエディタを開くには、次の手順に従います。

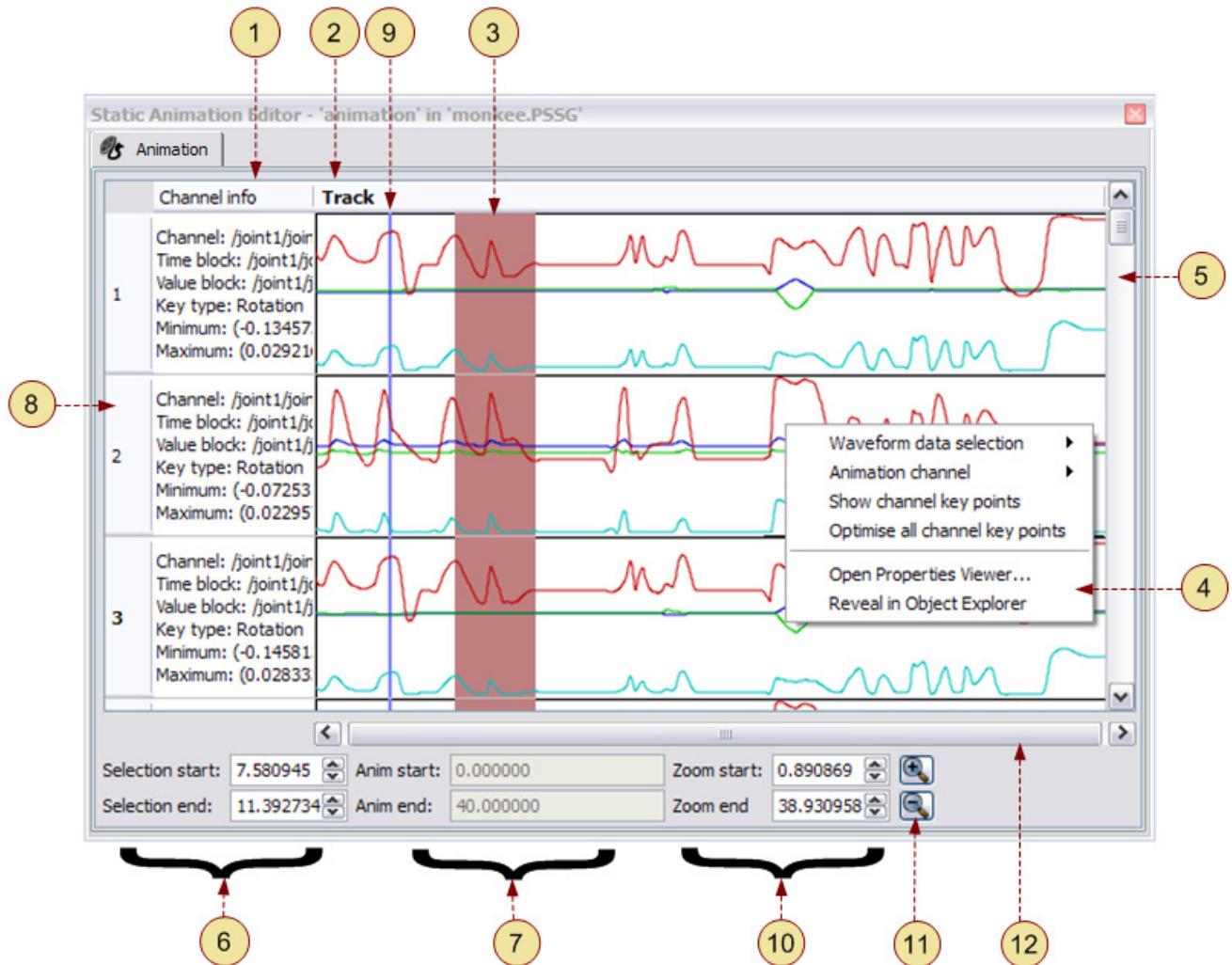
- (1) **Workspace Explorer - Objects** ビューを開きます。
- (2) PhyreEngine™アニメーションオブジェクトを右クリックし、コンテキストメニューから **Open Animation Editor...** を選択します。

図 74 Animation Editorを開く



Animation Editorのインターフェース

図 75 Animation Editor



1. アニメーションチャンネル情報（読み取り専用）。
2. アニメーションチャンネルの波形データ。横軸が時間を表します。
3. 選択バンド（ピンク色で表示された領域）。
4. Animation Editor のコンテキストメニュー。
5. アニメーションチャンネルの行をスクロールするための垂直テーブルスクロールバー。
6. 選択バンドの開始/終了位置の微調整。
7. アニメーションの開始/終了情報（読み取り専用）。
8. 選択されたテーブル行が黒色のボーダーでハイライト表示されます。
9. タイムライン。
10. エディタのズームの開始/終了情報。
11. エディタのズームイン/ズームアウトボタン。
12. すべてのアニメーションチャンネル情報をスクロールするためのスクロールバー。

アニメーションチャンネル情報

Channel info 列（図 75 の項目 1）には、アニメーション波形データ（項目 2）に関する情報が含まれます。この列には、PhyreEngine™ アニメーションチャンネルの名前、チャンネルの値ブロックと時間ブロックの名前、および格納されているアニメーションデータのタイプが表示されます。

アニメーション波形データ

Track 列（項目 2）では、アニメーションの波形データがグラフィカルに表現されます。回転の場合、クオータニオンの 4 要素がそれぞれ表示されます。平行移動の場合、3 つのデカルト座標が表示されます。

タイムラインの位置情報

Track 列（項目 2）で行の任意の部分をクリックすると、タイムライン（項目 9）が表示されます。波形データ内での時間位置は、選択スピンドル（項目 6）に表示されます。

波形データの選択

選択バンド（項目 3）を使えば、データの選択された部分を操作できます。データバンドを選択するには、マウスの左ボタンをクリックし、波形データの必要な部分をマウスでドラッグします。

コンテキストメニュー

コンテキストメニューを表示するには、背景で右クリックします（項目 4）。コンテキストメニューには、現在選択されているテーブル行（アニメーションチャンネル）または波形データの選択部分に対する処理の一覧が表示されます。

コンテキストメニューの処理は PhyreStation、PhyreStationDLL のいずれかによって登録されます。

PhyreStationDLL メニュー オプションについては、必要に応じて変更、拡張、公開、または隠蔽を行うことができます。詳細については、「6 PhyreStation コマンド」を参照してください。

選択スピンドル

選択スピンドル（項目 6）を使えば、現在選択されている領域の伸縮を行えます。また、表示されている数を直接編集することもできます。

波形の開始/終了ボックス

Anim Start ボックスと **Anim End** ボックス（項目 7）には、アニメーションの全体の長さ（時間）に関する読み取り専用情報が表示されます。

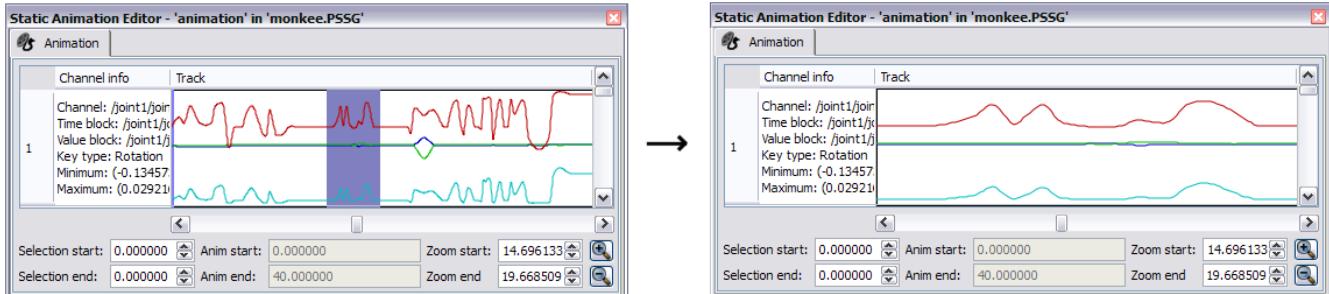
波形ズームコントロール

波形データを拡大表示すれば、データをより詳しく確認することができます。ズームスピンドル（項目 10）には、現在表示されているデータ領域の開始/終了位置が表示されます。

- ズームの開始/終了位置を調整するには、値を直接編集するか、スピンドルコントロールを使用します。
- ズームイン/ズームアウトをすばやく行うには、ズームボタン（項目 11）を使用します。ズームが完了すると、波形データの非表示部分へのアクセスを可能にするスクロールバー（項目 12）が有効になります。
- 波形データの特定部分にズームインするには、波形データ行をマウスの中央ボタンでクリックし、マウスをドラッグします。ズーム選択バンドが表示されます（図 76 を参照）。マウスの中央ボタンを離すと、ズームの手順が完了します。

- ビューをリセットしてすべての波形データを表示するには、波形データ領域でマウスの中央ボタンをクリックします。

図 76 波形データのズーム



PhyreEngine™ アニメーションの編集

このエディタを使えば、アニメーション波形データに対して多数の処理を実行できます。これらの処理の大部分は、波形データの特定部分を右クリックした後、コンテキストメニューからコマンドを選択することによって実行できます。その他のコマンドは、波形データの全体に対して処理を行います。

注意：このエディタでアニメーションに適用された変更はすべて、PhyreEngine™ Animation または PhyreEngine™ Animation Channel、あるいはその両方に即座に適用されます。それによりさらに、関連する PhyreEngine™ データベースも変更されます。

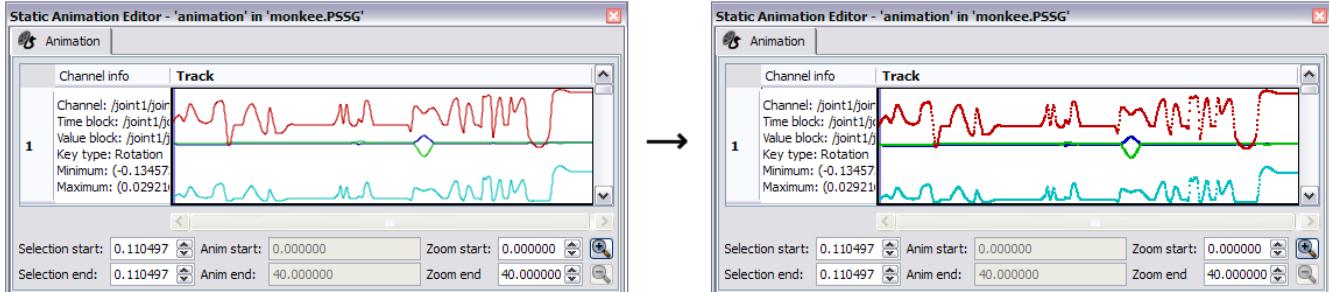
すべてのチャンネルキーポイントを最適化する

Optimise all channel key points コンテキストメニュー オプションは、編集中のアニメーションのすべてのアニメーションチャンネルのデータに対して処理を行います。
また、マウスをドラッグして波形データの特定部分を（複数のアニメーションチャンネルにわたって）選択することによっても、アニメーションキーフレームの数を減らすことができます。
この処理は、OptimizeAnimation、OptimizeAnimationChannel のいずれかのスクリプトコマンドを使ってスクリプト内で実行することもできます。

チャンネルキーポイントを表示する

Show channel key points オプションは、波形の表現を、キーフレームデータを含む連続線（デフォルト）から特定のキーフレームデータを表す離散ポイントへと変更します。

図 77 波形データの表現



選択部分以外をアニメーションから削除する

Waveform data selection > Crop animation to selection オプションは、アニメーション波形データの量を減らすために、選択された領域の外側にあるすべての波形データを、アニメーションのすべてのチャンネルから削除します。この処理を実行すると、アニメーションの長さが短くなります。
この処理は、CropAnimation スクリプトコマンドを使ってスクリプト内で実行することもできます。

選択部分をアニメーションから削除する

Waveform data selection > Delete selection from animation オプションを使えば、アニメーション波形データの選択された領域を、アニメーションのすべてのチャンネルから削除できます。この処理を実行すると、アニメーションの長さが短くなります。
このコマンドは、RemoveAnimationInterval スクリプトコマンドを使ってスクリプト内で実行することもできます。

選択部分から新しいアニメーションを作成する

Waveform data selection > Create a new animation from selection オプションを使えば、アニメーションの各チャンネルから選択されたアニメーション波形データの領域に基づいて、新しいPhyreEngine™アニメーションオブジェクトを作成することができます。
このコマンドは、CreateNewAnimationFromInterval スクリプトコマンドを使ってスクリプト内で実行することもできます。

チャンネルをアニメーションから削除する

Animation channel > Delete channel from the animation オプションを使えば、あるアニメーションチャンネルテーブル行（図 75 の項目 8）をアニメーションから削除できます。
このコマンドは、RemoveAnimationChannel スクリプトコマンドを使ってスクリプト内で実行することもできます。

注意：PhyreEngine™アニメーションや他の PhyreEngine™オブジェクト（複数可）からまだ参照されている PhyreEngine™アニメーションチャンネルオブジェクトを削除すると、PhyreEngine™が不安定になる可能性があります。結果に自信がもてない限り、このような操作は試みないでください。

14 Particle Editor

この章では、PhyreStation の **Particle Editor**、パーティクルシステムの処理（要素やアニメーションの追加/削除など）、および PhyreEngine™で使用可能なパーティクルシステムをエクスポート/コンパイルするプロセスについて説明します。

概要

PhyreStation の **Particle Editor** を使えば、PhyreEngine™ Particle System を作成および編集できます。満足のいくパーティクルシステムが定義できたら、そのパーティクルシステムを定義ファイルにエクスポートします。定義ファイルは解析され、ソースコードが生成されます。このソースコードはコードプロジェクトとともにコンパイルされ、PhyreEngine™の実行時にパーティクルシステムを生成するために使用されます。

PhyreStation では、PhyreEngine™ Particle System は次の要素を使って表現されます。

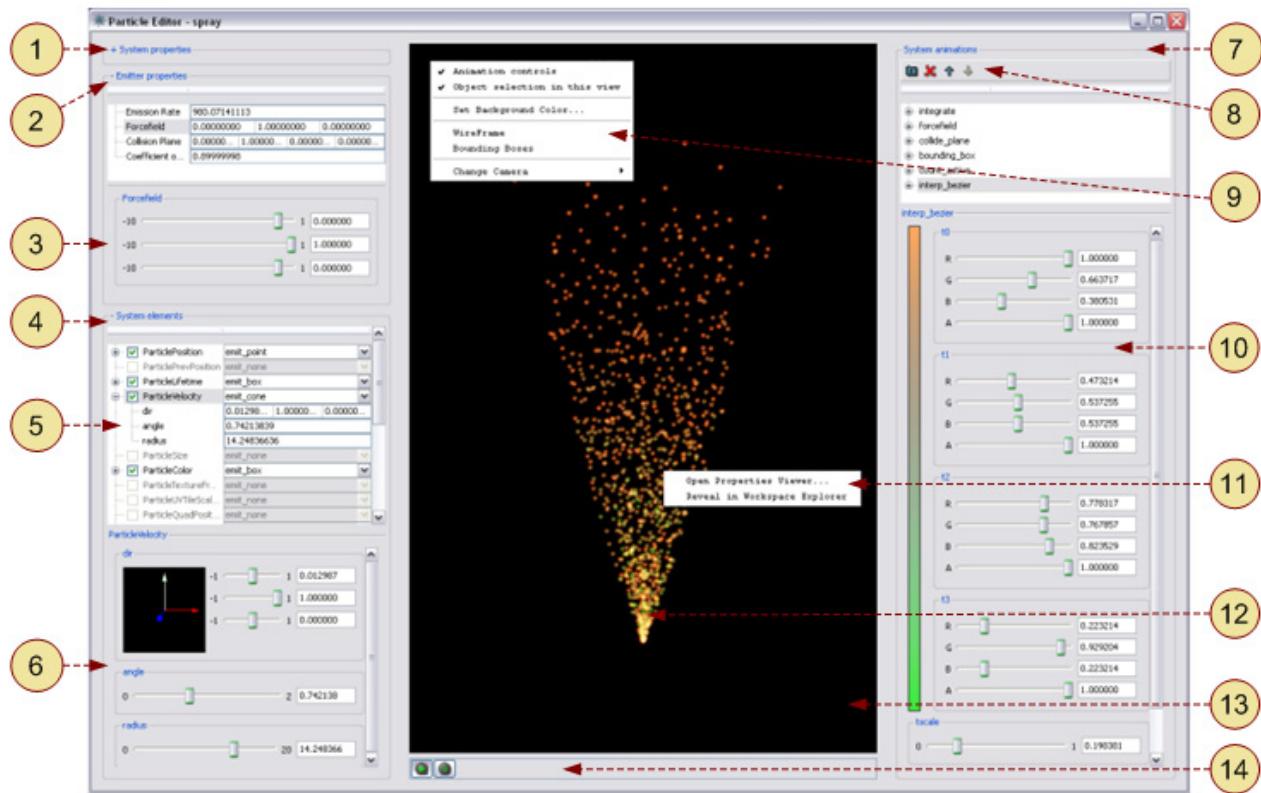
- テクスチャ、PhyreEngine™ Render Stream、および PhyreEngine™ Shader Group。「ビューア/エディタによるオブジェクトへのアクセス」(5 PhyreEngine™オブジェクト) を参照してください。
- PhyreEngine™ Particle Modifier Network。「Modifier Network Editor」(12 Graph Editor) を参照してください。
- PhyreEngine™ Particle Modifier Network Instance。「Modifier Network Instance Editor」(12 Graph Editor) を参照してください。
- PhyreEngine™ Particle Modifier Behavior – 定義ファイルによって表される。

パーティクルシステムの作成方法を説明する概要レベルのワークフローについては、「ワークフロー：パーティクルシステムの作成」(2 ワークフロー) を参照してください。サンプルパーティクルシステムの作成方法を説明した詳しい演習については、「[演習 6：パーティクルシステムをセットアップする](#)」([付録 A : 演習](#)) を参照してください。

Particle Editor

パーティクルシステムを作成/編集するには、**Particle Editor** を使用します。

図 78 Particle Editor



1. パーティクルシステムのプロパティ (折りたたまれた状態)
2. PhyreEngine™ Particle Modifier Behavior オブジェクトのプロパティ
3. パーティクルモディファイア動作オブジェクトのプロパティに対する拡張編集コントロール
4. PhyreEngine™ Particle System の動作要素
5. 動作要素の属性
6. 動作要素の属性に対する拡張編集コントロール
7. PhyreEngine™ Particle System の動作アニメーション
8. パーティクルシステムの動作アニメーションツールバー：追加、削除、上に移動、下に移動
9. レンダービューの余白部分のコンテキストメニュー
10. 動作アニメーションの属性に対する拡張編集コントロール
11. PhyreEngine™ Particle Emitter node オブジェクトのコンテキストメニュー
12. PhyreEngine™ Particle Emitter node オブジェクト
13. 編集中のパーティクルシステムを表示するレンダービュー
14. パーティクルシステムのアニメーションコントロール

システムプロパティ

Particle Editor の **System properties** 領域には、すべてのシステムプロパティが表示されます。

- PhyreEngine™オブジェクトリンクを含むフィールドを右クリックすると、そのオブジェクトのコンテキストメニューにアクセスできます。これらのフィールドを変更するには、既存の PhyreEngine™ オブジェクトまたはファイルをフィールド内にドラッグします。
- ユーザの変更が有効である場合には、**Update system** ボタンが有効になります。このボタンをクリックすると変更が適用されます。

システム要素

Particle Editor の **System elements** 領域には、現在サポートされているすべての動作要素とその属性を含むツリービューが表示されます。パーティクルシステムに関連付けられたパーティクルモディファイア動作オブジェクトに応じて、ツリービューの内容が変わる可能性があります。

- チェックの付いていない項目は無効化されていますが、編集は可能です。動作要素を有効化すると、その要素がパーティクルシステムに追加され、使用されます。要素を無効化すると、その要素がパーティクルシステムから削除されます。
- 各要素の最上位項目には、その要素の RenderDataType と、サポートされているエミッションタイプから利用可能なオプションが表示されます（「PhyreEngine™ Core ライブラリリファレンス」を参照）。各動作要素の属性に対するアクセスや編集を行うには、ツリービューのサブ項目を使用します。ある要素のエミッションタイプが変更されるたびに、パーティクルシステムの全体が即座に更新されます。
- 属性の中には、許容される最小値/最大値を持つものもあります。エミッションタイプを変更したことで新しい属性が範囲外になったと判定された場合、その新しい値が制限されて有効な範囲に保たれます。
- ある要素の最上位項目を選択すると、その要素の属性に対する拡張編集コントロールが表示されます。編集機能は属性ごとに異なります。方向やベクトルタイプの値を表す属性では小さな方向ビューが表示され、そこにスライダ値の 3 次元表現（図 78 の項目 6）が表示されます。

Particle Editor の **System elements** 領域には、要素のソートを実行するためのコントロールが用意されています。

- Sort key** ドロップダウンリストボックスには、ソートキーとして選択できる有効な要素の一覧が表示されます。ある要素を有効にするには、それを要素ツリービュー内で選択する（アクティブにする）必要があります。
- Sort type** ドロップダウンリストボックスには、利用可能なソート方法の一覧が表示されます。
- Sort all** チェックボックスは、すべてのアクティブな要素をパーティクルシステムのソートリストに追加したり、ソートリストから削除したりするためのメカニズムを提供します。ソートリストに対する追加や削除を個々の要素ごとに行うには、要素ツリービューの **Sort** 列のチェックボックスを使用します。

システムアニメーション

Particle Editor の **System animations** 領域には、パーティクルシステムに含まれる現在の動作アニメーションのプロパティが表示されます。

- アニメーションはツリービュー内で、上から下に並べられます。上に近いほうの動作アニメーションは、下に近いほうの動作アニメーションがパーティクルシステムを更新する方法に影響を与えます。

- **System animation** ツールバーを使えば、アニメーションの順番を変更したり、動作アニメーションの追加や削除を行ったりできます。
- ドロップダウンリストに表示された属性は、その動作アニメーションで現在有効になっている RenderDataType 属性を表します。
- アニメーションの最上位項目を選択すると、そのアニメーションの非 RenderDataType 属性に対する拡張編集コントロール（図 78 の項目 10）も表示されます。

システム要素/システムアニメーションの編集

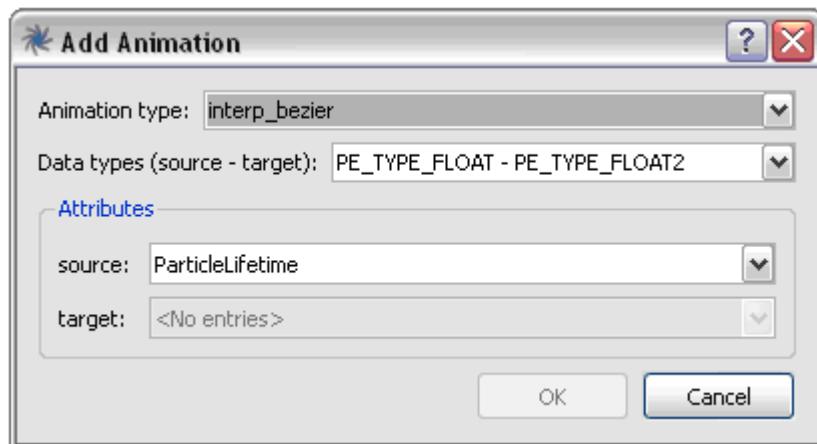
動作要素をパーティクルシステムの動作に追加したり、動作から削除したりするには、要素のチェックボックスをそれぞれオン、オフにします。すると、**System elements** 領域の RenderDataType 項目がその影響を受けます。

さらに、動作要素の追加/削除の影響は、パーティクルシステム内の関連するすべての動作アニメーションにも及ぶ可能性があります。そうした状況になるのは、動作要素のデータ型が、いずれかの動作アニメーションの RenderDataType 属性のデータ型に対応している場合です。

- ある動作要素を追加すると、その動作要素の RenderDataType が、互換性のある動作アニメーションの RenderDataType 属性のリストに追加されます。
- ある動作要素を削除すると、その要素の RenderDataType が、互換性のあるすべての動作アニメーションの RenderDataType 属性のリストから削除されます。RenderDataType を削除すると属性が無効な RenderDataType を持つようになる場合には、ユーザは無効化されようとしているすべての動作アニメーションの情報を知らされ、処理を継続するかキャンセルするかの選択を迫られます。処理を継続した場合、その動作要素と無効化された動作アニメーションが、パーティクルシステムから削除されます。

利用可能な動作要素と動作アニメーションとの関係は、新しい動作アニメーションを追加する際にも重要なとなります。図 79 に示す動作アニメーションは追加できません。なぜなら、選択されたターゲットデータ型をサポートする RenderDataType を含む動作要素が、パーティクルシステムの動作に 1 つも含まれていないからです。

図 79 Add Animation ダイアログ



このダイアログでは interp_bezier 動作アニメーションが指定されていますが、その 2 つの RenderDataType 属性のデータ型はそれぞれ PE_TYPE_FLOAT と PE_TYPE_FLOAT2 に設定されています。このパーティクルシステムの動作要素には、PE_TYPE_FLOAT のデータ型を持つ ParticleLifetime RenderDataType が含まれています。その反映として、**source** ドロップダウンリストでこの項目が利用可能になっています。PE_TYPE_FLOAT2 をサポートする RenderDataType を

含む動作要素は、現時点では1つも存在しません。このため **target** ドロップダウンリストは空になっており、ユーザはこの設定では動作アニメーションを追加できません。

パーティクルシステムの作成

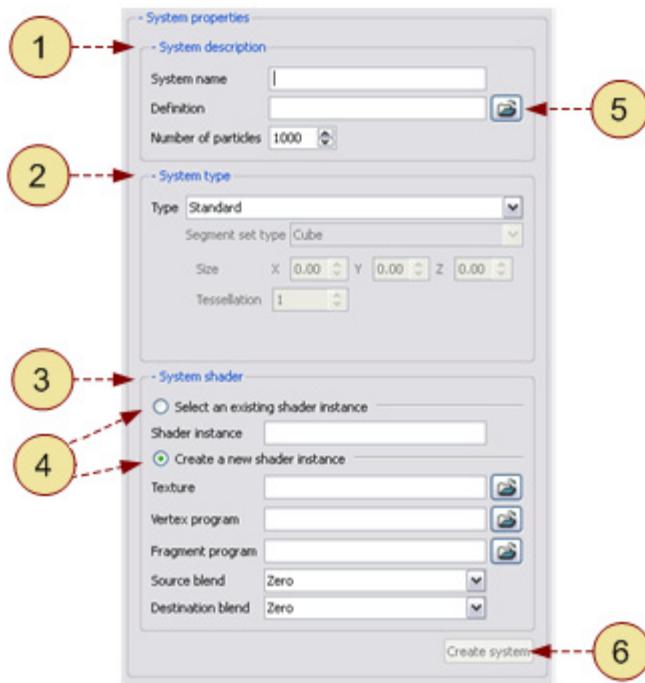
Particle Editor を使ってパーティクルシステムを作成できるためには、PhyreEngine™ Particle Emitter Node オブジェクトまたは PhyreEngine™ Visible Particle Emitter Node オブジェクトがワークスペース内に存在している必要があります（「PhyreEngine™ Core ライブラリリファレンス」を参照）。

注意：サンプルパーティクルシステムの作成方法を説明した詳しい演習については、「[演習6：パーティクルシステムをセットアップする](#)」（[付録A：演習](#)）を参照してください。

パーティクルシステムを作成するには、次の手順に従います。

- (1) **Workspace Explorer Scene Graph** ビューを開き、パーティクルシステムの追加先となるデータベースを見つけます。
- (2) ルートノードオブジェクトを右クリックし、次のいずれかを行います。
 - 可視パーティクルエミッターノードオブジェクトを追加するには、コンテキストメニューから **Add new child > Node > Visible Render Node > Visible Particle Emitter node** を選択します。
 - パーティクルエミッターノードオブジェクトを追加するには、コンテキストメニューから **Add new child > Node > Visible Render Node > Render Node > Particle Emitter node** を選択します。
- (3) 新しいノードオブジェクトを右クリックし、コンテキストメニューから **Create Particle System...** を選択します。**Particle Editor** が開きます。

図 80 Particle Editor – System properties グループボックス



1. パーティクルシステムの記述プロパティ
2. パーティクルシステムのタイププロパティ
3. パーティクルシステムのシェーダインスタンスプロパティ

4. シェーダインスタンスマードを指定するラジオボタン
5. パーティクルシステムの定義フィールド
6. **Create system** ボタンまたは **Update system** ボタン（モードに応じて）

(4) **System properties** グループボックスで新しいパーティクルシステムのプロパティを定義します。

隣に参照ボタンが表示されているフィールドでは、ファイル、既存の PhyreEngine™オブジェクト、のいずれかを指定できます。ファイルを指定するには、ファイルブラウザからファイルをフィールド内にドラッグします。PhyreEngine™オブジェクトを指定するには、アプリケーション内の他の場所でオブジェクトを選択し、それをフィールド内にドラッグします。

Definition フィールド（図 80 の項目 5）以外のフィールドはすべて必須です。定義ファイルまたは定義オブジェクトを指定した場合には、PhyreStation によってそのファイルまたはオブジェクトが読み込まれ、そこに定義された動作要素とアニメーション動作が表示されます。

Definition を指定しなかった場合には、**Particle Editor** にデフォルトの動作が表示され、新しいパーティクルモディファイア動作オブジェクトが作成されます。

(5) 一連の有効なパーティクルシステムプロパティを入力すると、**Create system** ボタンが有効になります。**Create system** ボタンをクリックすると、システムが作成され、利用可能なシステムプロパティが **Particle Editor** に設定されます。システムの **Definition** が指定された場合、システムの要素とアニメーションが表示されます。**Definition** が指定されなかった場合、これらの領域は空になります。

新しいパーティクルシステムを作成すると、PhyreEngine™ Particle Modifier Network Instance オブジェクトも作成されます。

パーティクルシステムの編集

パーティクルシステムを編集するには、次の手順に従います。

- (1) **Workspace Explorer - Objects** ビューを開き、目的の PhyreEngine™ Particle Modifier Network Instance オブジェクトを見つけます。
- (2) パーティクルモディファイアネットワークインスタンスオブジェクトを右クリックし、コンテキストメニューから **Edit Particle System...** を選択します。**Particle Editor** のウィンドウが表示されます。
- (3) パーティクルシステムのプロパティの編集方法の詳細については、「Particle Editor」(14 Particle Editor) を参照してください。

パーティクルシステムのエクスポート

パーティクルシステムの編集中、動作要素と動作アニメーション、およびそれらの属性がワールスペースに対して継続的にコミットされます。PhyreEngine™ Particle System 定義ファイルのエクスポートは任意のタイミングで行えますが、それにはパーティクルシステムをコンパイルします。コンパイル処理は定義ファイルを作成し、それをコンパイルして PhyreEngine™プロジェクトに挿入可能なソースコードを作成します。

パーティクルシステムをコンパイルするには、次のいずれかを行います。

- **Workspace Explorer Database** ビューを開き、PhyreEngine™データベースオブジェクトのコンテキストメニューから **Compile Particle Systems** を選択します。
- `DBCompileParticleSystems` コマンドをパラメータとともに、**Command Window** に入力します。

- コマンド DBCompileParticleSystems を使用する Lua スクリプトを実行します。「7 スクリプト」を参照してください。

15 GUI

この章では、PhyreStation ユーザインターフェースの重要な機能について説明します。

概要

PhyreStation のメインウィンドウには、互いに重なり合わない次の 3 つの領域が含まれています。

- ツールバー領域
- ワークエリア
- ステータスバー

注意 : PhyreStation を Microsoft プラットフォーム上で使用する場合、選択されたデスクトップのスタイルに応じて GUI が変わる可能性があります。

ツールバー

ツールバー領域には一連のツールバーが含まれます。

図 81 PhyreStation のツールバー



File ツールバー



File ツールバーを使えば、ワークスペースファイルを作成、オープン、および保存することができます。



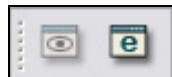
スクリプトボタン を使えば、スクリプトファイルを検索して実行できます。ワークスペースが開いていると、このボタンは無効化されます。

Objects ツールバー



Objects ツールバーには 1 つのボタン (Find ダイアログボタン) が表示されます。Find ダイアログでは、ワークスペース内で PhyreEngine™ オブジェクトの検索を行えます。

Workspace ツールバー



Workspace ツールバーには 2 つのボタンが含まれます。ボタンは、動的な Properties Viewer の表示/非表示を切り替えるためのトグルボタンで。



ボタンは別の Workspace Explorer ウィンドウを表示します。

Trace ツールバー



Trace ツールバーには 2 つのトグルボタンが含まれます。ボタンは Command Window を表示します。ボタンは Log ウィンドウを表示します。

DNet Communication ツールバー

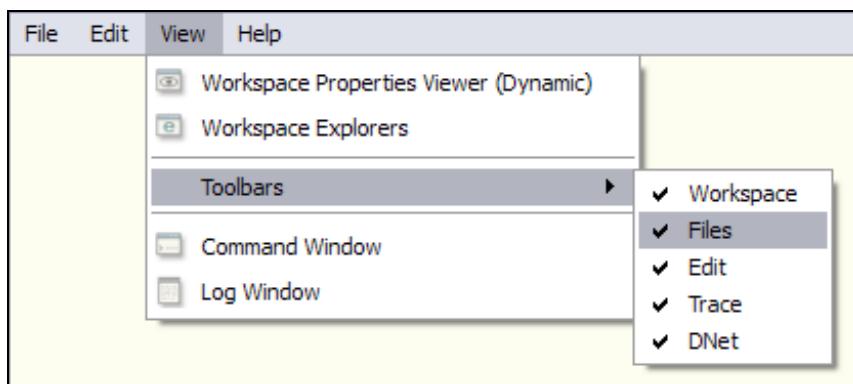


DNet Communication ツールバーの詳細については、「10 DNet 通信」を参照してください。

このツールバーが有効になるのは、ワークスペースが開いている場合だけです。

表示されるツールバーを定義するには、メインメニューから **View > Toolbars** を選択します。

図 82 ツールバーの選択メニュー



ツールバーを再配置したり、フローティング状態にして独自のウィンドウを備えるようにするには、ツールバーハンドルをドラッグします。

ステータスバー

メインウィンドウのステータスバーに表示される情報は、次のとおりです。

- アプリケーションの一般的なステータスマッセージ。
- コマンドのステータスマッセージ（コマンドまたはスクリプトの実行中に表示される）。これらのメッセージは Log ウィンドウにも送られます。詳細については、「Log ウィンドウ」(17 PhyreStation のログ、ユーザプリファレンス、およびエラー処理) を参照してください。

ワークエリア

ワークエリアとは、メインアプリケーションウィンドウ内で、他のアプリケーションウィンドウ（またはフレームウィンドウ）を表示可能な領域のことです。

フレームウィンドウを移動すれば、ワークエリアの内側、デスクトップ上の任意の場所のいずれかでウィンドウをフローティング状態にすることができます。

ウィンドウのドッキング

スペースが空いていれば、ワークエリアの上下左右にフレームウィンドウをドッキングさせることができます。フレームウィンドウは、別のフレームウィンドウにドッキングさせることもできます（その場合はタブページが作成される）。フローティング状態のフレームウィンドウをメインウィンドウにドッキングさせることができるのは、そのウィンドウが利用可能なスペースよりも小さい場合だけです。

ドッキング状態のフレームウィンドウをフリー状態にするには、ウィンドウのキャッシュバーをドラッグします。フレームウィンドウの状態（フローティング状態、ドッキング状態のいずれか）を切り替えるには、キャッシュバーをダブルクリックします。

フレームウィンドウをワークエリアの境界線上またはその近くに、ドッキング状態にならないように配置するには、**Ctrl** キーを押しながらウィンドウをドラッグします。

タブページの移動

タブページをフレームウィンドウから引き離すには、ページのタブ部分をドラッグします。その後、タブページを再配置するには次のようにします。

- ワークエリアの内側、外側のいずれかに、タブページをドロップします。タブページが独自のフレームウィンドウを伴って表示されます。
- 同じアプリケーションインスタンス内の任意のフレームウィンドウに、タブページをドロップします。フレームウィンドウ内でのタブページの位置は、フローティング状態の緑色の矢印で示されます。

現在のアプリケーション/ワークスペースのプリファレンス設定によっては、アプリケーションはワークスペースの再オープン時に、ウィンドウを以前の位置やサイズ、内容に戻すことができます（これは常に可能とは限らない。PhyreEngine™データベースの内容による）。

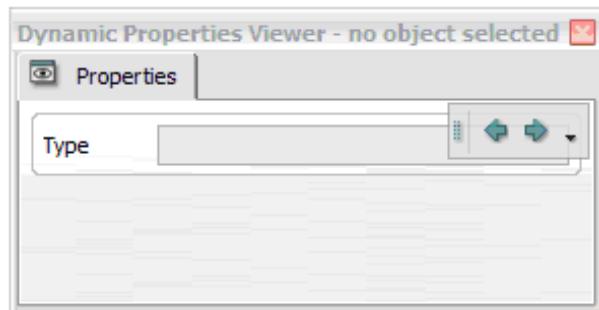
ウィンドウの履歴

PhyreStation のほとんどのウィンドウやビューでは、あるオブジェクトまたは対象の **History Navigation** ツールバーを使用できます。このツールバーを使えば、ウィンドウのオープン後にウィンドウ内に表示されたすべてのオブジェクトを、日付順に表示できます。

オブジェクトの **Properties Viewer**（「オブジェクトの編集」（5 PhyreEngine™オブジェクト）を参照）を除くすべてのウィンドウ/ビューでは、履歴ツールバーは次のように動作します。履歴ツールバーが初めて表示された時点では、ツールバーは無効化されています。これは、ユーザが（適切な型の）別のオブジェクトをビューにドラッグするまで、無効化されたままになります。1つ以上の他のオブジェクトがビューにドロップされると、そのビューに以前表示されたオブジェクトにナビゲートできるようになります。

History Navigation ツールバーを表示するには、ビューのタブのコンテキストメニューから **Show History** を選択します。

図 83 History Navigation ツールバー



History Navigation ツールバーは次のように使用します。

- ウィンドウに表示されたオブジェクト間を移動するには、左矢印と右矢印をクリックします。移動するにつれて、各オブジェクトとそのプロパティがビューア内に表示されます。
- いずれかの矢印の上でマウスの左ボタンを押し続けると、すべての先行するオブジェクト（左矢印）または後続のオブジェクト（右矢印）のリストが表示されます。リストからオブジェクトを選択すれば、そのプロパティを再びビューア内に表示できます。
- ウィンドウのオープン後にウィンドウ内に表示されたすべてのオブジェクトのリストを表示するには、ツールバーの右側にあるドロップダウン矢印をクリックします。それらのオブジェクトの並び順は、表示された順番になります。リストからオブジェクトを選択すれば、そのプロパティを再びビューア内に表示できます。
- 履歴バッファをクリアするには、履歴リストから **Clear list** を選択します。

注意：

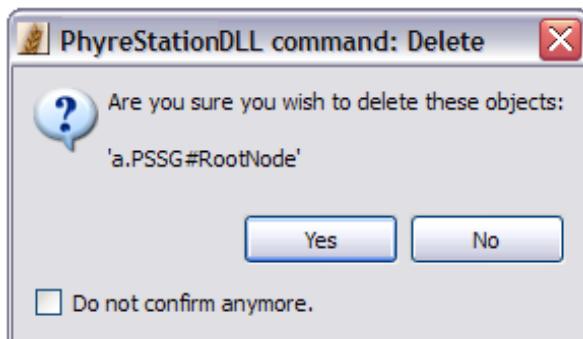
- あるオブジェクトをワークスペースから削除しても、そのオブジェクトがビューの現在の対象になっているのでない限り、そのオブジェクトは履歴バッファからは削除されません。もう存在していないオブジェクトを履歴バッファリストから選択した場合には、ビューの更新は行われず、単にそのオブジェクトがバッファから削除されます。
- 一部のビューでは、**History Navigation ツールバー**が「覆い隠される」、あるいは見えなくなる場合があります。3D シーンを描画するビューでは特にそうなります。これを解決するには、ツールバーが覆い隠されるビュー部分にツールバーを移動するか、あるいはツールバーが最後に表示されていた領域をクリックします。ツールバーはまだ動作しています。

ダイアログボックス

PhyreStation のほとんどのダイアログボックスについては、改めて説明するまでもありません。しかしながら、以下のダイアログは少し変わった動作を示します。

削除確認ダイアログボックス

図 84 削除確認ダイアログボックス



このダイアログボックスが表示されないようにするには、**Do not confirm anymore** チェックボックスにチェックを入れます。

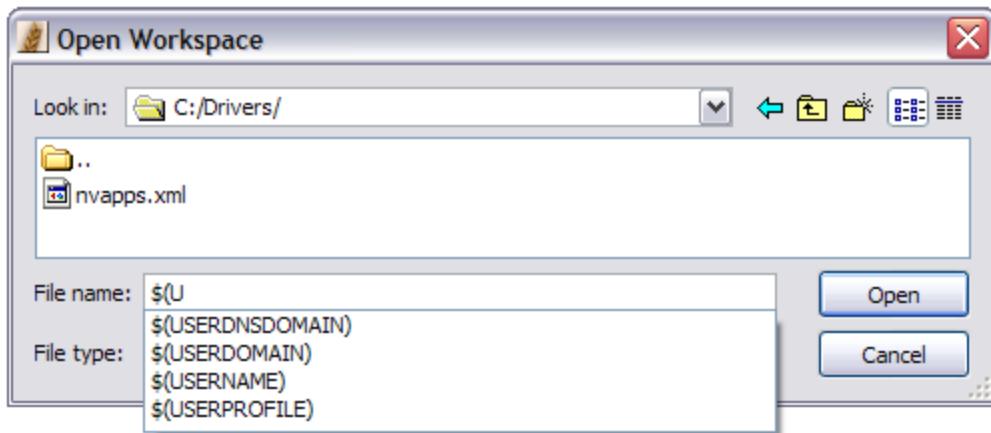
このダイアログボックスを再び有効化するには、**Edit > User Preferences** を選択し、**Prompt for Deletion MessageBox** プリファレンスを設定します。

Shift キーと **Delete** キーと同時に押すと、ダイアログは表示されません。ただし、削除に他の意味が含まれている場合はその限りではありません。たとえば、オブジェクトがワークスペース内の他のオブジェクトから参照されているような場合です。

File ダイアログボックス

PhyreStation ではファイルをロードまたは保存する際に、ファイルダイアログを使ってディレクトリを探します。これらのダイアログにはファイルオートコンプリータが含まれています。「\$(U」のように環境変数の先頭の文字列を入力したり、「%USER%」のようにワイルドカードを使用したりすることができます。条件に一致する選択可能な環境変数のすべてが、オートコンプリータによって一覧表示されます。

図 85 ファイルコンプリータ



PhyreStationのリソースマネージャ

PhyreStation も他の大部分の GUI アプリケーションと同じく、アイコンや画像を表示する必要があります。これらのリソースが見つからない場合、アプリケーションはその見つからないリソースをデフォルトの項目で置き換えます。見つからない GUI アイコンの位置に、疑問符付きの灰色の四角形が表示されます。PhyreStation のリソースは [PhyreStation_root_directory]/Resources ディレクトリに格納されます。

16 インストール/アーキテクチャ

インストール

インストールの詳細については、PhyreStation のインストールルートディレクトリに格納されたファイル `readme_j.txt` を参照してください。このファイルにはバージョン情報、前提条件となるソフトウェアの詳細情報、インストールされるファイルの一覧、サポート情報、および本ガイドに含まれていないインストール情報が記載されています。

ライセンス

Nokia Qt ソフトウェアのライセンスを別途入手しなくても、PhyreStation に付属する Qt 実行時ライブラリをインストールして使用することができます。

追加のライセンス情報については、`[PhyreStation_root_directory]/Readme_j.txt` を参照してください。

アーキテクチャ

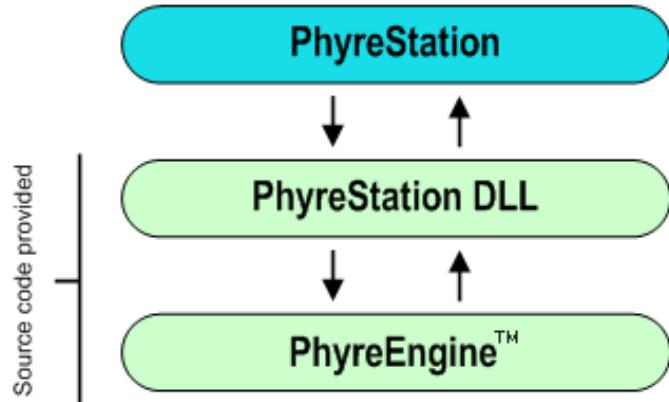
PhyreStation は、一連のコンパイル済みバイナリアプリケーションファイル（実行可能ファイルや多数の実行時ライブラリなど）として提供されています。完全なファイル一覧については、`[PhyreStation_root_directory]/Readme_j.txt` を参照してください。

注意：PhyreStation のソースコードはサードパーティに提供されていません。

外部コンポーネント

PhyreStation は多数の外部コンポーネントに依存しています。もっとも重要なコンポーネントは、PhyreStationDLL と PhyreEngine™ コアライブラリです。これらのコンポーネント間の関係を、図 86 に示します。

図 86 アプリケーションのコンポーネント



PhyreStationDLL

PhyreEngine™ API と PhyreStation の GUI 領域との通信は、「PhyreStationDLL」と呼ばれる中間コンポーネントを介して行われます。PhyreStationDLL には、PhyreStation のコマンドセットを拡張したり、カスタム PhyreEngine™オブジェクトを処理したりする機能も備わっています。

サードパーティが基盤となる PhyreEngine™ソースコードを変更/拡張するのと似た方法で、PhyreStationDLL ソースコードを変更して PhyreStation を拡張できます。「8 PhyreStation のカスタマイズ」を参照してください。

PhyreEngine™

PhyreEngine™グラフィックスシステムの詳細については、「PhyreEngine™プログラミングガイド」および「PhyreEngine™ Core ライブラリ リファレンス」を参照してください。

バージョン情報

PhyreStation とそのコンポーネントに関するバージョン情報を取得するには、メインメニューから **Help > About PhyreStation...** を選択します。「PhyreStation に関する情報」(1 PhyreStation の概要) を参照してください。

アプリケーションインスタンス ID

About PhyreStation ダイアログに表示されるアプリケーションインスタンス ID は、実行されている特定のアプリケーションインスタンスを識別します。インスタンス ID は通常、アプリケーション起動時にアプリケーションによって生成されます。しかしながら、PhyreStation をコマンドラインから起動する場合には、ユーザがこの ID を指定できます。コマンドラインパラメータでこの ID を使用すれば、新しいタスクの実行ターゲットとして特定のアプリケーションインスタンスを指定できます。

17 PhyreStationのログ/ユーザプリファレンス/エラー処理

この章では、PhyreStation のロギングシステム、ユーザプリファレンス、エラー処理、およびローカライズについて説明します。

Log ウィンドウ

概要

PhyreStation は、すべてのユーザ処理とフィードバックメッセージを Log ウィンドウにロギングします。この情報は、コマンド実行の進捗を監視したり一般的なトラブルシューティングを行ったりする際に役立ちます。

PhyreStation が深刻なシステムエラーによって終了すると、Log ウィンドウのコピーがファイル [PhyreStation_root_directory]/PhyreStation.htm に保存されます。このファイルは、問題の原因を診断する際に役立つ可能性があるほか、エラー報告時に PhyreStation サポートから提供を求められる可能性もあります。

Log ウィンドウの表示

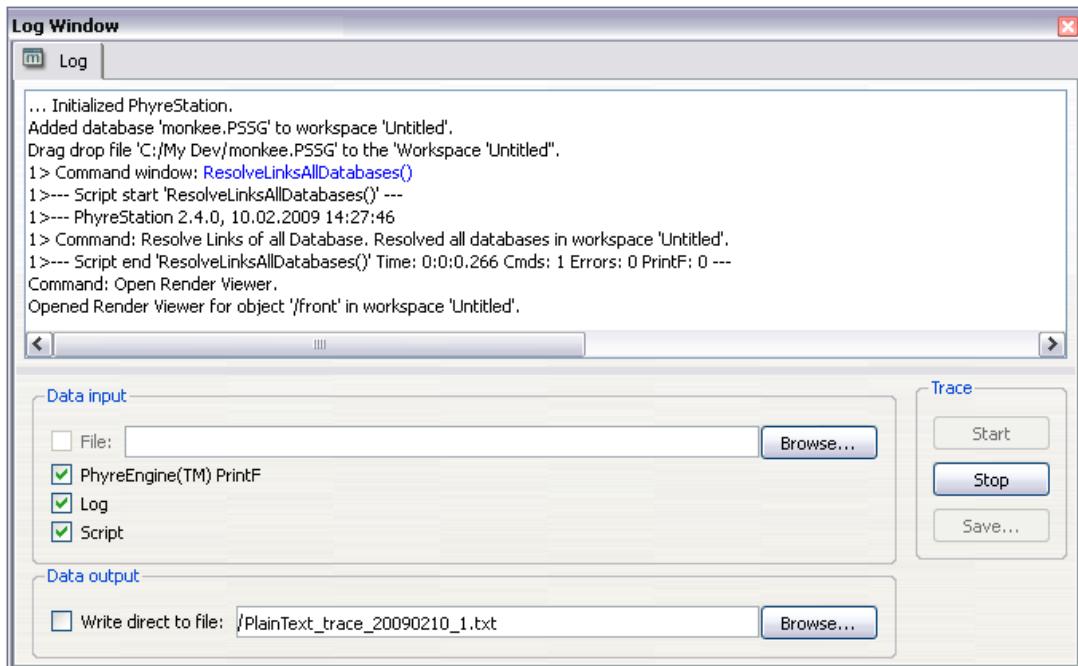
Log ウィンドウを開くには、メインメニューから **View > Log Window...** を選択するか、Log ウィンドウツールバー ボタンをクリックします。

図 87 Log ウィンドウツールバー ボタン



Log ウィンドウには、テキスト領域とオプションのコントロールグループボックスが含まれます。コントロールグループボックスの表示/非表示を切り替えるには、Log タブを右クリックし、コンテキストメニューから対応するオプションを選択します。

図 88 Log ウィンドウ



Log ウィンドウ内のテキストは、選択してコピーすることができます。

Log ウィンドウのコントロールグループボックス

Log ウィンドウのコントロールを使えば、ログプロセスの開始/停止、ログ情報の保存、および入力ソースの選択による表示テキストのフィルタリングを行えます。

Trace

- 監視プロセスと情報の表示を開始または再開するには、**Start** ボタンをクリックします。
- 監視プロセスを停止するには、**Stop** ボタンをクリックします。
- 現在表示されている情報をファイルに保存するには、**Save...** ボタンをクリックします。

注意：出力ストリームファイルに情報を保存できるのは、**Log** ウィンドウが停止されている時だけです。

Data Input

ログプロセスへの入力データのソースを定義するには、**Data Input** グループボックスのチェックボックスを使用します。

- 指定されたファイルの内容をリアルタイムで表示するには、**File** フィールドにファイル名を入力し、対応するチェックボックスを選択します。
- PhyreEngine™ PSSG_PRINTF** メッセージを表示するには、**PSSG** チェックボックスを選択します。
- アプリケーションやコマンドのメッセージを表示するには、**Log** チェックボックスを選択します。
- スクリプト実行メッセージを表示するには、**Script** チェックボックスを選択します。

デフォルトでは、**File** チェックボックス以外のすべてのチェックボックスが選択されています。

Data Output

監視プロセス中に**Log** ウィンドウに表示された情報をファイルに直接書き込むには、**Write direct to file** チェックボックスを選択します。アプリケーションと同じフォルダ内にデータ出力ファイルが作成されます。作成日付とインクリメントインデックスを含むファイル名が、自動的に生成されます。デフォルトでは、**Write direct to file** チェックボックスは選択されていません。

注意： このデータ出力ファイルは、**Save...** ボタンで指定されるファイルと同じものではありません。

データ出力ファイルは、クラッシュが発生していて、最新のトレース情報を**Log** ウィンドウから取得できないような場合に役立ちます。データ出力処理は PhyreStation.htm とは異なります。というのも、この処理では、アプリケーションの存続期間中のアクティビティがロギングされるほか、**PSSG** または **Script** チェックボックスが選択されていても、ログ情報のみが記録される（スクリプト情報や PhyreEngine™ 情報は記録されない）からです。

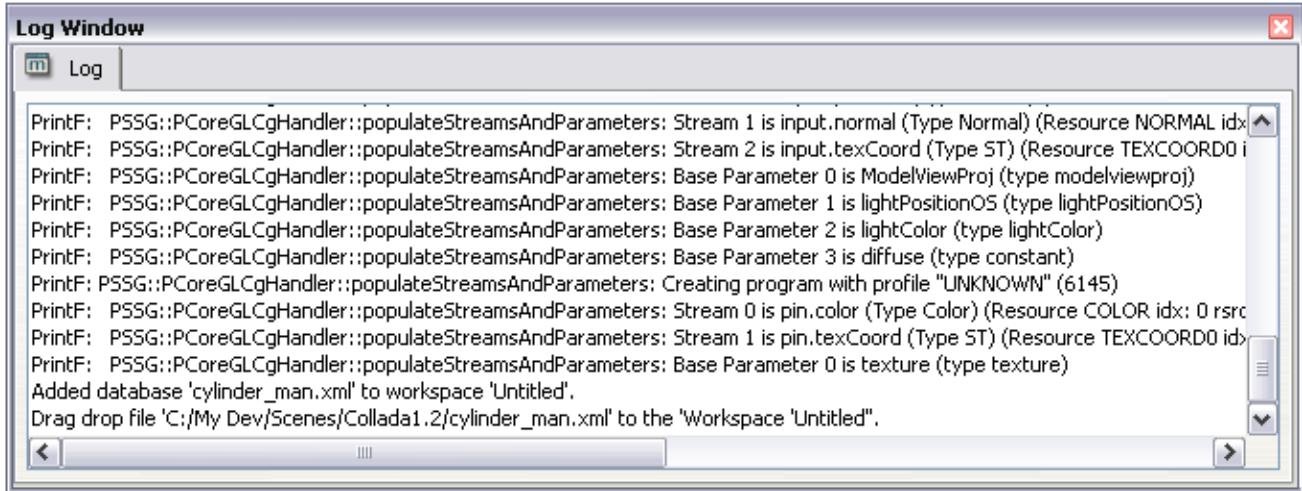
PhyreEngine™ ユーザ処理からのフィードバック

Log ウィンドウは、重要なユーザ処理のすべてをロギングするとともに、PhyreStation コマンドメッセージ/フィードバックシステムを使用するあらゆるコマンドから送信されてきたフィードバックメッセージを表示します。ただし、速度や効率上の理由により、これが常に実用的であるとは限りません。

PhyreEngine™を使用するディベロッパは PhyreEngine™の **PSSG_PRINTF** を使用することで、PhyreEngine™処理に関するより詳しい情報を入手できます。PhyreEngine™ **PSSG_PRINTF** メッセージを表示するには、**PhyreEngine™ PrintF** チェックボックスを有効にします。これは特に、PhyreEngine™

が COLLADATMファイルをロードしている場合などに役立ちます。PhyreEngineTMは、ファイルのパース時に検出された不整合や非互換をすべて報告します。メッセージの一例を図 89 に示します。

図 89 Log ウィンドウの PSSGPrintF



PSSG_PRINTF の機能は PhyreEngineTMによって提供されており、PhyreEngineTMまたは PhyreStationDLL コード内の任意の場所で利用できます。PSSG_PRINTF 出力を stdout にリダイレクトするには、PhyreStationDLL.h に定義されている

`PPhyreEngineDll::unregisterPhyreEnginePrintfCallback()`、`PhyreEngine.h` に定義されている `PhyreEngine::PhyreEngineResetPrintfCallback()` のいずれかを呼び出します。そうした場合、PSSG_PRINTF 出力が Log ウィンドウから永久に削除されます。

注意：PhyreEngineTMの PSSG_PRINTF 入力方式を介してユーザに送信されたメッセージを、PhyreStation から操作することはできません。

ユーザプリファレンス

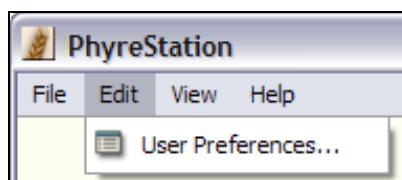
PhyreStation が処理するプリファレンスは、次の 3 つのカテゴリに分けられます。

- ワークスペースの保存/復元方法を制御するプリファレンス。「ワークスペースデータ」(3 ワークスペース) を参照してください。
- アプリケーション全体のプリファレンス設定。これは `prefs.xml` ファイルに格納されます。
- 現在ロードされているワークスペースセッションに基づくユーザ設定。これが有効化されると、アプリケーションプリファレンス設定がワークスペース設定でオーバーライドされます。

PhyreStation を初めて実行した際に、プリファレンスファイル `prefs.xml` が作成されます。

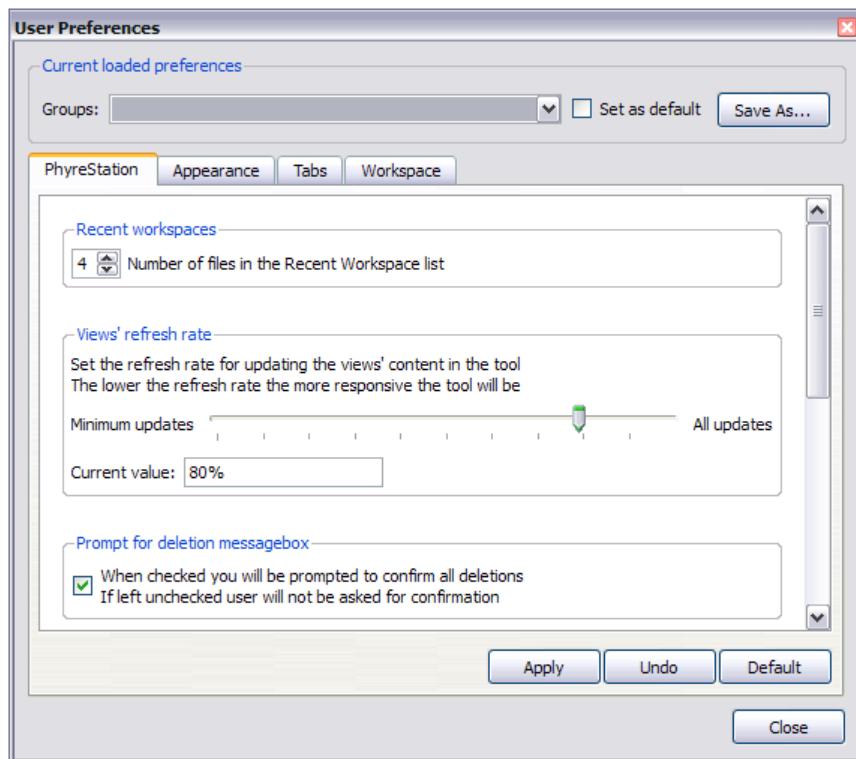
PhyreStation のユーザプリファレンスを編集、保存、およびロードするには、**User Preferences** ダイアログボックスを使用します。このダイアログを開くには、メインメニューから **Edit > User Preferences...** を選択します。

図 90 Edit メニュー



User Preferences ダイアログは、**Current loaded preferences** グループボックスとタブページ領域から構成されています。

図 91 User Preferences ダイアログボックス



ユーザプリファレンスタブページ

タブページには、PhyreStation アプリケーションとその各種コンポーネントに対するすべてのユーザプリファレンスが表示されます。

Default ボタンは、現在のタブに含まれるプリファレンスをデフォルト値にリセットします。各プリファレンスには、他のすべての PhyreStation ユーザプリファレンスに対して「安全」なデフォルト値が用意されています。

1 つ以上のプリファレンスに無効な値を適用しようとすると、**User Preferences** ダイアログの関連プリファレンスのグループボックス内に、赤色のフィードバックメッセージが表示されます。

ユーザの設定が、有効であるが推奨できない場合もあります。その場合、関連プリファレンスのグループボックス内に青色のフィードバックメッセージが表示されます。

注意 :

- 1 つのタブでデフォルトのプリファレンスにリセットしても、それらのプリファレンスが他の変更されたプリファレンスと互換性があるとは限りません。競合が発生した場合、それを警告するフィードバックメッセージが表示されます。
- 各タブはそれぞれ独自の Undo/Redo 履歴を備えていますが、異なるプリファレンスファイル間での切り替えを行うと、履歴がリセットされます。

Current Loaded Preferences グループボックス

ワークスペースはデフォルトで、ユーザのプリファレンスの状態を格納できます（機能が有効化されている場合）。各ワークスペースは、アプリケーション設定を指定/オーバーライドできます。

Current Loaded Preferences グループボックスを使えば、単一のワークスペースに対して複数のプリファレンス設定セット（グループと呼ばれる）を管理できます。これは、1つのワークスペースでタスクごとに異なる PhyreStation 設定が必要になる場合に役立ちます。ユーザは任意のタイミングで、**Groups** ドロップダウンリストから異なるプリファレンスグループを指定できます。

この機能を有効にするには、**Workspace** タブページで **Remember and restore the workspace preference settings** チェックボックスをオンに設定します。ワークスペースまたはグループに対する現在のプリファレンスは、ワークスペースのクローズ時にワークスペースファイルに保存されます。

新しいグループを作成するには、プリファレンス設定を必要に応じて変更し、それを別名で保存します。ワークスペースの現在のグループセットにその新しいグループが追加されます。

Set as default チェックボックスを使えば、ワークスペースが次回開かれる時にデフォルトでロードされるグループを指定できます。

注意： グループが有効化されていない場合、つまり **Remember and restore the workspace preference settings** チェックボックスが設定されていない場合、ワークスペースに指定されたグループはすべてクリアされます。

以前のバージョンのアプリケーション設定の使用

PhyreStation を初めて起動した際に、以前のバージョンの設定を変換して使用するかどうかを尋ねるダイアログが表示される場合があります。このダイアログが表示されるのは、PhyreStation が現行バージョン用の設定を検出できず、他の古い設定を検索している場合だけです。使用すべきバージョンをユーザが選択すると、PhyreStation はその設定の使用を試みます。古い設定を正確に変換して使用することが常に可能であるとは限りません。したがって、ある程度の妥協を強いられる可能性があります。

ローカリゼーション

現在のところ、PhyreStation で利用可能なバージョンは英語だけです。表示されるテキストはすべて英語です。これには、**Log** ウィンドウに表示されるメッセージ、メインウィンドウのメニューバーのラベル、ダイアログボックスに表示されるテキストなどが含まれます。

ただし、英語以外の言語/地域設定を使用しているシステム上でも、PhyreStation を使用することは可能です。たとえば、データベースのロード/保存時に日本語のファイルパスを使用したり、PhyreEngine™ オブジェクトに日本語の名前を付けたりすることができます。

コマンドのローカリゼーションサポート

コマンドのローカリゼーションサポートは、UTF8 形式の文字列を PhyreStationDLL 経由で通信することにより、可能となります。これは、PhyreStationDLL 内のすべての文字列を（ASCII などではなく）UTF8 形式にエンコードすることで実現されます。

（UTF8 は Unicode タイプのエンコーディング形式であり、マークアップ情報を含むマルチバイトシーケンスを扱います。これは、多言語文字などの非 ASCII 文字を格納するうえで効率的な方法です。UTF8 は ASCII のスーパーセットであるため、たとえば、ASCII を要求するレガシーアプリケーションでも、UTF8 で保存されたテキストを読み取ることができます。）

PhyreStation は、PhyreStationDLL から受信した各文字列（つまり、const char 配列と std::string オブジェクト）が UTF8 でエンコードされているものと仮定したうえで、それを表示に適した形式に変換し

ます。ユーザ独自のプライベートな文字列や内部文字列は、好みの方法で自由にエンコードしてもかまいませんが、PhyreStation 内で表示または編集するための文字列は、UTF8 でエンコードした後で PhyreStationDLL に渡す必要があります。文字列が常に正しく表示されるようにしたければ、UTF8 のみを使って作業を行うことをお勧めします。

UTF8 は ASCII に対して後方互換であるため、すでに ASCII で開発している西ヨーロッパや米国のユーザは何もする必要がありません。ASCII 文字列は PhyreStation で正しく表示されます。

非 ASCII (日本語文字など) のユーザは、自身の文字列を UTF8 に変換した後で、PhyreStationDLL 経由で通信を行う必要があります。これは次のことを意味します。

- ハードコードされた文字列を含むソースコードファイルは、UTF8 エンコーディングで保存する必要があります (MS .net では、ソースファイルの保存は **Save File As...** ダイアログボックスから行えます。Save ボタンの右側にあるドロップダウンメニューをクリックして **Save with Encoding...** を選択した後、**Unicode (UTF-8 without signature) - Codepage 65001** オプションを選択します)。
- PhyreStation のワークスペース XML ファイルは、サポートされているエンコーディングを使って保存する必要があります。PhyreStation は自動的に、これらのファイルを正しく保存します。テキストエディタは、UTF8 エンコーディングでの保存をサポートするもの (MS .net のテキストエディタなど) を使用する必要があります。
- PhyreStation のデータベース HIER ファイルは、サポートされているエンコーディングで保存する必要があります。PhyreStation は自動的に、これらのファイルを正しく保存します。テキストエディタは、UTF8 エンコーディングでの保存をサポートするもの (MS .net のテキストエディタなど) を使用する必要があります。

エラー処理

ソフトウェアエラーは次の 2 種類に分類できます。

- ユーザエラーは、ユーザが無効なデータを入力したり無効な処理を実行したりした場合（読み取り専用ファイルに書き込みを行おうとした場合や無効なスクリプトコマンドを入力した場合など）に発生します。PhyreStation ではユーザエラーは一般に、**Log** ウィンドウ内の適切なメッセージか警告メッセージボックス、あるいはその両方によって処理されます。場合によっては、**PhyreStation Error** ダイアログボックスが表示されることもあります。
- システムエラーは通常、より深刻な問題であり、バグの検出やシステムリソースの不足など、基盤となるコードやシステムに問題が生じていることを示しています。システムエラーは常に、**PhyreStation Error** ダイアログボックスに表示されます。

すべてのエラーは、[PhyreStation_root_directory]/PhyreStation.htm ファイルにも転送されます。

PhyreStationサポート

PhyreStation サポートの連絡先の情報については、

[PhyreStation_root_directory]/Readme_j.txt を参照してください。

付録A：演習

この付録には、PhyreStation の各種機能の使用方法を学ぶ際に役立つ演習が含まれています。

概要

以下の各演習では、おそらく PhyreStation におけるあらゆる作業で使用するであろう、非常に基本的な処理について解説しています。これらの演習は、先頭から順番に行うことをお勧めします。というのも、いくつかの演習は、それまでの演習の続きになっているからです。

- 演習 1：ワークスペースとデータベースに関する基本的な演習：新しいワークスペースの作成、データベースの作成、既存データベースの追加、およびデータベースリンクの解決を行います。
- 演習 2：データベースのリンク：データベースリンクの切断、壊れたデータベースのアンロード/ロード、再確立、およびリンクの解決を行います。
- 演習 3：PhyreEngine™オブジェクトを表示、検索、および編集する：オブジェクトの演習。オブジェクトの表示、検索、および編集をさまざまな方法で行います。
- 演習 4：スクリプトファイルを作成し、ワークスペースと関連付ける：スクリプトの演習。スクリプトを作成し、それをワークスペースに関連付け、そのスクリプトをさまざまな方法で実行します。
- 演習 5：PhyreEngine™データベースを統合する
- 演習 6：パーティクルシステムをセットアップする：パーティクルシステムを作成します。

演習 1：ワークスペースとデータベースに関する基本的な演習

この演習では、PhyreStation のワークスペースを紹介します。

ステップ	手順	追加情報
1. PhyreStation を実行します。	OS のファイルエクスプローラを使って PhyreStation.exe ファイルを見つけ、それをダブルクリックします。しばらくすると PhyreStation が起動し、デフォルトのレイアウトが表示されます。	現時点でカレントワークスペースが開いている場合はそれを閉じます。これはおそらく Blank Workspace ですが、これが作成されるのは、PhyreStation の対応するプリファレンスが有効になっている場合です。このプリファレンスを検索するには、PhyreStation のメインメニューから Edit > User Preferences... を選択し、 PhyreStation タブをクリックします。 「カレントワークスペース」(3 ワークスペース) を参照してください。

ステップ	手順	追加情報
2. 新しいワークスペースを作成します。	メインメニューから File > Workspace > New... を選択した後、新しいワークスペースの名前と場所を入力します。ワークスペースが保存されると、指定された場所にファイルが作成されます。	ワークスペースはプロジェクトファイルに似ています。ワークスペース内では、オブジェクト (PhyreEngine™データベースや PhyreEngine™オブジェクトなど) の追加や作成を行えます。スクリプトもワークスペースに関連付けることができます。一度開けるワークスペースは、1つだけです。 「3 ワークスペース」を参照してください。
3. PhyreEngine™データベースを作成します。	メインメニューから File > Database > New database... を選択し、新しいデータベースの名前と場所を入力します。データベースが保存されると、指定された場所にファイルが作成されます。	PhyreEngine™には.pssg と.heir という、2種類のデータベースがあります。.pssg はバイナリタイプのファイルで、.heir は XML テキストファイルです。 「4 データベース」を参照してください。
4. Workspace Explorer – Databases ビューを開きます。	PhyreStation のツールバーの  アイコンをクリックして Workspace Explorer ビューを開きます。ビューの余白部分で右クリックしてコンテキストメニューを表示し、 Databases を選択します。 作成したばかりの新しいデータベースが、 Workspace Explorer – Databases ビューに最上位データベースとして表示されます。	Workspace Explorer ではカレントワークスペース内をナビゲートし、オブジェクトを選択できます。ワークスペースには次のように、いくつかのフィルタがあります。 <ol style="list-style-type: none"> 1. Databases 2. Scene – Graph 3. Objects 4. Custom Groups 5. Scripts あるフィルタを選択すると、ワークスペース内のどのオブジェクトを表示するかと、それらのオブジェクトの表示方法が決まります。 「 Workspace Explorer 」(3 ワークスペース) を参照してください。
5. Workspace Explorer フィルタプロパティを表示します。	フィルタビューのプロパティを表示するには、PhyreStation のメインメニューから Edit > User Preferences... を選択し、 Workspace タブをクリックし、 Enable workspace explorer filter properties チェックボックスを選択します。	Workspace Explorer フィルタのプロパティには、あるオブジェクトが別のオブジェクトから使用 (リンクまたは参照) されているかどうかなど、オブジェクトに関する追加情報が表示されます。 「 Workspace Explorer のフィルタプロパティ」(3 ワークスペース) を参照してください。

ステップ	手順	追加情報
6. PhyreEngine™データベースを追加します。	<p>PhyreEngine™ monkee.pssg データベースファイルを見つけ、それをワークスペースにドラッグします。</p> <p>monkee.pssg データベースは通常、Monkee ディレクトリの下の PhyreEngine™ Scenes ディレクトリ内に格納されています。</p>	<p>カレントワークスペースにデータベースを追加するには、次のいずれかの方法を使用します。</p> <ul style="list-style-type: none"> • File メニュー • データベースフィルタビューの余白部分のコンテキストメニュー • PhyreStation の外側からデータベースファイルをドラッグ&ドロップする <p>COLLADA™ファイル (.dae または .xml) もカレントワークスペースに追加できます。PhyreEngine™はこのファイルを.pssg タイプのバイナリファイルに変換します。</p> <p>「既存のデータベースの追加」(4 データベース) を参照してください。</p>
7. PhyreEngine™データベースを解決します。	<p>Workspace Explorer – Databases ビューで、最上位データベースの monkee.pssg を選択します。右クリックしてコンテキストメニューを表示した後、コマンド Resolve Links を選択します。</p> <p>monkee.pssg のアイコンが、リンク解決の成功を示すチェックマーク (✓) に変わります。最上位データベースのすぐ下に、必要な追加リソースを収集するために必要であると PhyreEngine™が判断したリンク先データベースが表示されます。</p>	<p>選択された項目（複数可）のコンテキストメニューには、その項目に対して実行可能な一連のコマンド（アクション）が表示されます。ユーザは PhyreStationDLL でカスタムコマンドを定義できます。</p> <p>「PhyreStationへの新しい PhyreEngine™オブジェクト型の追加」(8 PhyreStation のカスタマイズ) を参照してください。</p> <p>Resolve Links コマンドは、シーンのレンダリングに必要なすべてのオブジェクトを検索するように、PhyreEngine™に指示します。さらにこのコマンドは、どの PhyreEngine™オブジェクトをワークスペースに追加すべきかを PhyreStation に指示します。</p> <p>注意：リンクデータベースに対して作業を実行することはできません。</p> <p>「データベースのリンク解決」(4 データベース) を参照してください。</p>

ステップ	手順	追加情報
8. Log ウィンドウ。	<p>Log ウィンドウには、このツールや最近実行されたコマンドのステータスが表示されます。Log ウィンドウがまだ表示されていない場合には、ツールバーの  アイコンをクリックします。</p>	<p>Log ウィンドウに表示されるテキストは、一般的なアプリケーションアクティビティ、実行されたコマンド、および特殊な PhyreEngine™ PSSGPrintF メッセージから収集されます。</p> <p>Log ウィンドウに表示されたテキストは、PhyreStation.html ファイルにも出力されます。</p> <p>「Log ウィンドウ」(17 PhyreStation のログ、ユーザプリファレンス、およびエラー処理) を参照してください。</p>
9. 演習はこれで終了です。演習 2 に進んでください。		

演習 2：データベースのリンク

この演習では、データベースの処理方法を紹介します。

PhyreEngine™には、データやリソースをデータベースに格納するという機能以外に、一連のデータベース間でリソースを共有するという重要な機能があります。これは、ある PhyreEngine™オブジェクトが別のデータベース内の必要なオブジェクトを参照した際に、自動的に起動されます。PhyreEngine™はデータベースの解決時に、これらの参照が有効かどうかをチェックします。一致するものが見つからない「リンク」が PhyreEngine™によって 1 つ以上検出されると、データベースの解決が失敗したことになります。PhyreStation では、最上位データベースから第 1 階層のリンク先データベースへの参照を表示できます。

Workspace Explorer - Databases ビューでは、リソースリンクの処理が失敗すると、データベースに  アイコンが表示されます。

ステップ	手順	追加情報
1. 演習 1を行います。		
2. データベースのリンクを切断します。	<p>PhyreEngine™の Scenes\Monkee ディレクトリを探します。</p> <p>monkeychest.pssg を選択し、その名前を一時的に変更します。</p>	<p>データベースファイルの名前が変更されたため、PhyreEngine™はデータベースの次回解決時に、monkeychest.pssg という名前のファイルを見つけることができません。</p>

ステップ	手順	追加情報
3. 壊れたリソースリンクを含むデータベースを表示します。	<p>Workspace Explorer – Databases ビューで monkee.pssg を選択します。</p> <ol style="list-style-type: none"> そのコンテキストメニューから Unload を選択します。 コンテキストメニューから Load を選択します。 コンテキストメニューから Resolve Links を選択します。 <p>monkeychest.pssg のアイコンが アイコンに変わったことに注意してください。</p>	<p>PhyreEngine™は、monkee.pssg で必要なリソースを解放し（リンク先データベース内のリソースも解放し）、それらのリソースを取得し直すように指示されました。ところが、1つのデータベースの名前が変更されたため、この時点で全部は見つからなくなります。</p> <p>「データベースのアンロード/リロード」と「データベースのリンク解決」（4データベース）を参照してください。</p>
4. 壊れたリソースリンクを含まないデータベースを表示します。	<p>変更されたデータベース名を元の名前 (monkeychest.pssg) に戻します。ステップ 3 を繰り返します。</p> <p>すると、データベースのアイコンが アイコンに戻るはずです。</p>	
5. 演習はこれで終了です。		

演習 3 : PhyreEngine™オブジェクトを表示、検索、および編集する

この演習では、PhyreStation ワークスペース内での PhyreEngine™オブジェクトの検索方法や、PhyreEngine™オブジェクトの属性の編集方法について説明します。

ステップ	手順	追加情報
1. 演習 1を行います。		
2. Workspace Explorer – Objects ビューを開きます。	<p>PhyreStation のツールバーの アイコンをクリックして Workspace Explorer を開きます。ビューの余白部分で右クリックしてコンテキストメニューを表示し、Objects を選択します。</p>	<p>Workspace Explorer – Objects ビューには、正常に解決されたすべてのデータベースに属するすべての PhyreEngine™ オブジェクトが表示されます。</p> <p>各カテゴリには、オブジェクトの一覧とそれらが所属するデータベースが隠されています。</p> <p>「オブジェクトのブラウズ」（5 PhyreEngine™オブジェクト）を参照してください。</p>
3. オブジェクトの表示方法を変更します。	<p>ビューの余白部分で右クリックしてコンテキストメニューを表示し、View By Database を選択します。</p>	<p>ビュー内に余白部分がない場合は、任意のカテゴリを選択します。Ctrl キーを押したまま同じカテゴリを再度選択した後、右クリックすると、ビューのコンテキストメニューが表示されます。</p>

ステップ	手順	追加情報
4. Workspace Explorer の Scene-Graph ビューを開きます。	PhyreStation のツールバーの  アイコンをクリックして別の Workspace Explorer を開きます。ビューの余白部分で右クリックしてコンテキストメニューを表示し、 Scene-Graph を選択します。	Workspace Explorer – Scene-Graph ビューには、ノードオブジェクト型から派生したオブジェクトのみが表示されます。このビューには、 Objects ビューに表示されるオブジェクトのサブセットが表示されます。 ルートノードの子になっているすべてのノードが、PhyreEngine™シーン内にレンダリングされます。
5. Render Viewerを開きます。	一部のデータベースにはカメラノードが含まれている可能性があります。データベース <code>monkee.PSSG</code> を展開してシーンのルートノード/ <code>root</code> を表示します。ルートノードを展開してノード階層を表示します。カメラノード <code>/front</code> を選択します。これが、表示の対象（プライマリオブジェクト）になります。そのコンテキストメニューから Open Render Viewer... を選択します。 余白部分（青色の背景）のコンテキストメニューから、 Object selection in this view 、 Bounding Boxes 、および Highlight Selection を選択します。	Render Viewer には、選択されたカメラの視点（ここでは <code>/front</code> カメラのシーン内での位置と向き）に基づいて、現在のシーディングラフが表示されます。 カメラを回転/移動しないと意味のある表示が得られないことがあります。Alt キーとマウスボタンを使いながらマウスを動かします。 猿のバウンディング矩形領域でクリックしないでください。 「Render Viewer」(11 ビューア) を参照してください。
6. Properties Viewerを開きます。	<code>/front</code> オブジェクトを選択し、コンテキストメニューから Open Properties window... を選択します。	Properties Viewer にはオブジェクトの属性が表示されます。一部の属性の値は、GUI から直接変更できます。黒色のフィールドは編集可能なフィールドです。属性の値を変更すると、その変更はすぐに反映されます。 「オブジェクトの編集」(5 PhyreEngine™オブジェクト) を参照してください。

ステップ	手順	追加情報
7. オブジェクトの属性値を変更します。	<p>ツールバーの  アイコンをクリックして Command Window を開きます。コマンド行編集フィールドで次のように入力します。 SetObjectProperty("monkee.pssg#/front", "FarPlane", 0, 101.0) Return キーを押します。 Render Viewer が更新されて変更が反映されます。</p>	<p>一部のコマンドは、コンテキストメニューから選択する方法と、対応するスクリプトコマンドを入力する方法の両方に対応しています。 SetObjectProperty() コマンドは、スクリプトコマンドとしてのみ利用できます。</p> <p>「コマンドの実行方法」(6 PhyreStation コマンド) を参照してください。</p> <p>オブジェクトの属性を変更する際に Log ウィンドウを見れば、Command Window に入力したのと同じコマンドが実行されるのに気づくはずです。</p> <p>「Log ウィンドウ」(17 PhyreStation のログ、ユーザプリファレンス、およびエラー処理) を参照してください。</p> <p>このツールの Help Viewer には、すべてのコマンドとその構文、および解説情報が含まれています。</p> <p>「Help Viewer」(1 PhyreStation の概要) を参照してください。</p>
8. Find ダイアログを使ってノードオブジェクトを検索します。	<p>PhyreStation のツールバーの  アイコンをクリックして Find ダイアログを開きます。このダイアログで、Workspace と Selection を選択します。 Find 編集ボックスに *poly* と入力し、Return キーを押します。</p> <p>monkee.PSSG#/polySurface459 スキンノードオブジェクトが見つかるはずです。</p>	<p>PhyreStation はワークスペース内のすべてのオブジェクトを検索し、完全にまたは部分的に一致するものを見つけます。</p> <p>「ワークスペース内でのオブジェクトの検索」(5 PhyreEngine™オブジェクト) を参照してください。</p>
9. ビュー内でオブジェクトを選択します。	<p>Find ダイアログで monkee.PSSG#/polySurface459 を選択します。他のビューでも同じオブジェクトがハイライト表示されることに注意してください。 Scene-Graph ビューで /joint1 を選択した後、Objects monkeychest.pssg ビューで polySurface459 オブジェクトを選択します。すべてのビューで選択内容が変わります。</p>	<p>ビューをスクロールしたりビュー内の項目を展開したりしないと、選択したオブジェクトが表示されない可能性があります。</p> <p>ビュー内の薄青色の選択項目は、その項目の子が非表示になっていることを示します。その項目を展開すると、灰色の選択項目が表示されます。</p> <p>「複数のウィンドウからの選択」(5 PhyreEngine™オブジェクト) を参照してください。</p>

ステップ	手順	追加情報
10. オブジェクトをドラッグ&ドロップします。	<p>/joint34 で終わる名前を持つノードを見つけます。それを Scene-Graph ビューで選択します。マウスの左ボタンを押しながらそのノードをルートノード/root のところまでドラッグした後、マウスのボタンを離します。ポップアップメニューから Reparent Node を選択します。</p> <p>ドラッグ&ドロップは複数のビュー間で行えます。</p>	<p>ドラッグ&ドロップ操作の効果を確認するには、Render Viewer のアニメーション再生ボタン をクリックします。アニメーションを停止するには ボタンをクリックします。アニメーションデータはすべての PhyreEngine™データベースに含まれるわけではないため、これらのボタンはグレー表示になる可能性があります。</p>
11. 新しいオブジェクトを作成します。	<p>Workspace Explorer – Databases ビューで monkee.PSSG を右クリックし、コンテキストメニューから Add new object > Node > Light Node を選択します。</p> <p>既存の光源オブジェクト /pointLight1 に対して Properties View を開きます。その平行移動ベクトル値を新しい光源オブジェクトにコピーします。値を少し変更し、シーン内の別の位置に光源が配置されるようにします。新しい光源オブジェクトの親を /root ノードに変更します。</p> <p>Render Viewer で Display Light を選択し、新しい光源ノードが表示されるようにします。</p>	<p>Scene-Graph ビューで、monkee.PSSG データベースの下に新しいオブジェクトが表示されます。</p>
12. オブジェクトを削除します。	<p>カメラオブジェクト /front を選択し、コンテキストメニューから Delete を選択します。削除の確認プロンプトに答えます。オブジェクトがワークスペースから削除され、データベースからも PhyreEngine™によって削除されます。この時点でのワークスペースとデータベースが「ダーティー」としてマークされます（タイトルバーのアスタリスクと、ビューのプロパティ (Modified の下) のチェックマークで示される）。</p>	<p>リンク先オブジェクト（ビューのプロパティを参照）は、GUI を使って削除できません。ただし、DeleteObject() スクリプトコマンドを使えばそれらも削除できます。これを行う際には注意が必要です。というのも、これを行うと PhyreEngine™が不安定になる可能性があるからです。</p>
13. ワークスペースを閉じます。	<p>PhyreStation のメニューバーから File > Workspaces > Close を選択します。</p>	<p>PhyreStation はワークスペースを閉じる前に、その時点で保存する必要のあるすべてのデータベースの一覧を含む確認ダイアログを表示します。それらのデータベースを保存しなかった場合、その変更内容が失われます。</p>

ステップ	手順	追加情報
14. 演習はこれで終了です。		

演習 4 : スクリプトファイルを作成し、ワークスペースと関連付ける

この演習では、PhyreStation スクリプトの処理方法を紹介します。スクリプトは、ファイル拡張子.lua を持つテキストファイルです。スクリプティングに使用する言語は Lua です。PhyreStation に登録されているコマンドのほとんどが、スクリプト内で使用できます（つまり、スクリプト内で動作するように登録されている）。これらのコマンドの大部分は、PhyreStationDLL 内に存在しています。

この演習では、PhyreStation のさまざまな動作モード（スクリプト実行時に何も表示しないモード（GUIなし - バッチファイルで使用）、スクリプトを実行してから PhyreStation を起動するモード、

PhyreStation を起動してからスクリプトを実行するモード）を使ってスクリプトを実行する方法を示します。

ステップ	手順	追加情報
1. 演習 1 と 3 を行います。		
2. スクリプトを作成します。	Microsoft Notepad など、任意のテキストエディタプログラムを開きます。 次のように入力します。 「 <code>print(PSSGVersion())</code> 」。このテキストファイルを <code>MyScript</code> という名前で保存します。必要に応じてファイル拡張子.lua で.txt を置き換えます。	<code>print()</code> 文はネイティブの Lua コマンドです。コマンド <code>PSSGVersion()</code> は、PhyreStationDLL で定義されたコマンドです。さらにこのコマンドは、スクリプト内から実行できるように PhyreStationDLL 内で登録されています。
3. PhyreStation を実行します。	PhyreStation を実行するには、そのアイコンをクリックするか、OS のコマンドラインから（パラメータを一切指定せずに）実行します。 カレントワークスペース（デフォルトの「空の」ワークスペースなど）が開いていない場合は、新しいワークスペースを作成します。	
4. Workspace Explorer の Scene-Graph ビューを開きます。	PhyreStation のツールバーの  アイコンをクリックして Workspace Explorer を開きます。ビューの余白部分で右クリックしてコンテキストメニューを表示した後、 Scripts (Scripts ビュー) を選択します。	Workspace Explorer – Scripts ビューには、カレントワークスペースに関連付けられたすべてのスクリプトが一覧表示されます。スクリプトは多くのワークスペース間で共有される可能性があるため、どこか中央の場所にスクリプトファイルを格納することをお勧めします。あるスクリプトから別のスクリプトを呼び出すこともできます。

ステップ	手順	追加情報
5. スクリプトをワークスペースに関連付けます。	作成したばかりのスクリプトファイルを見つけ、それをスクリプトビューにドラッグします。すると、スクリプトがカレントワークスペースに関連付けられ、ワークスペースから削除されるまでその状態が保たれます。	ワークスペースの保存時に、ワークスペースに対するスクリプトの相対パスが、ワークスペースファイル内に格納されます。 「ワークスペースへのスクリプトの関連付け」(7 スクリプト) を参照してください。
6. Scripts ビューからスクリプトを実行します。	Log ウィンドウを開きます。Scripts ビューでスクリプトファイル MyScript を選択し、そのコンテキストメニューから Run を選択します。スクリプトから Log ウィンドウへのレポート内で、PhyreStation が使用している PhyreEngine™ のバージョンが出力されているのを確認できるはずです。	Command Window も開いている場合には、実行されたスクリプトの結果が表示されます。 スクリプト内で実行されるコマンドの大部分は、成功、失敗のいずれかのメッセージを報告します。コマンド PSSGVersion() では、メッセージは不要なので発行されません。 「スクリプトの実行」(7 スクリプト) を参照してください。
7. ワークスペースとスクリプトの関連付けを保存します。	メニューバーから File > Workspaces > Save As... を選択します。ファイル名として MyWorkspace.xml と入力し、Save をクリックします。	
8. Command Window からスクリプトを実行します。	作成したばかりのスクリプトファイルを見つけ、それを Command Window の編集ボックスにドラッグします。ドロップが完了すると、スクリプトファイルが自動的に実行されます。	このステップで説明した方法を使えば、ワークスペースに関連付けられていないスクリプトを実行できます。
9. OS のコマンドラインウィンドウから PhyreStation とスクリプトを実行します。	OS のコマンドラインを開きます。次のように入力します。 「PhyreStation.exe -script=<path>\MyScript.lua -scriptLog=<path>\MyScriptLog.txt -scriptQuit="true"」 。そして Return を押します。PhyreStation はスクリプトの実行後、すぐに終了します。スクリプトログファイル内に実行のレポートが格納されています。	PhyreStation をバッチモードで実行すると、GUI は一切表示されません。 PhyreStation がシステムのパスに含まれている必要があります。そうでない場合は、実行可能ファイルへのパスを明示的に入力する必要があります。 バッチモードは、ワークフロー内の他のバッチ処理ジョブに混ざって一連の PhyreStation タスクを自動化する必要がある場合に役立ちます。

ステップ	手順	追加情報
10. OS のコマンドラインウインドウから PhyreStation とスクリプトを実行しますが、その際、スクリプトの完了後も PhyreStation が実行を継続できるようにします（スクリプト GUI モード）。	OS のコマンドラインでステップ 9 のテキストを入力しますが、最後のパラメータを -scriptQuit="false" に変更し、Return を押します。 PhyreStation は上記と同じように実行されますが、今度は終了せず、まるで PhyreStation をアイコンから起動したかのように起動されます。	コマンドラインからのスクリプト実行は、PhyreStation の起動時にタスク（データベースのロードなど）を実行する必要がある場合に役立ちます。 このモードではスクリプトログファイルは生成されない点に注意してください。 「コマンドラインからのスクリプトの実行」(7 スクリプト) を参照してください。
11. ワークスペースファイル内に含まれるスクリプトを、OS のコマンドラインから実行します。	OS のコマンドラインで次のように入力します。「 PhyreStation.exe -workspace=<path>\MyWorkspace.xml -script="MyScript.lua" -scriptQuit="false" 」。そして Return を押します。 ステップ 9、10 と同じスクリプトが実行されます。	スクリプト情報の一部は、PhyreStation のログファイル PhyreStation.html に報告されます。
12. 演習はこれで終了です。		

演習 5 : PhyreEngine™データベースを統合する

この演習では、PhyreEngine™を処理する際にほとんどのディベロッパが通常実行する基本的なワークフローの例を示します。その典型的な状況とは、一連の「生の」PhyreEngine™データベースが存在しており、それらを統合して、ターゲットプラットフォームに適した 1 つのデータベースを作成する必要がある、というものです。結果のデータベースに含まれるのは、関連するアセットのみであり、重複するオブジェクトはすべて削除されます。また、最終的なデータベースには圧縮も施されます。

この演習を行う前に、これまでの演習がすべて完了しているものと仮定しています。

ステップ	手順	追加情報
1. PhyreStation を実行します。		
2. 処理対象の「生の」データベースを含む新しいワークスペースを作成します。	PhyreEngine™のシーンサンプル Monkee を見つけます。データベース monkee.pssg をカレントワークスペースに追加します。このデータベースを選択し、そのコンテキストメニューから Resolve Links を選択します。	

ステップ	手順	追加情報
3. 新しいスクリプトを作成し、それをワークスペースに追加します。	新しいワークスペースファイルと同じディレクトリ内に新しいPhyreStationスクリプトファイルを作成し、 process.lua という名前を付けます。新しいスクリプトファイルを Workspace Explorer - Scripts ビューにドラッグすることで、そのスクリプトをワークスペースに追加します。	この演習の末尾にある「サンプルスクリプト – process.lua」のスクリプトコードを参照してください。 なお、このスクリプト内のコマンドのいくつかは、ユーザがこの演習の結果を確認できるようにPhyreStationを更新するためだけに含められています。
4. ワークスペースを保存します。	カレントワークスペースを MyWorkspace.xml として保存します。	
5. PhyreStationを終了します。		
6. ワークスペースファイルに関連付けられたスクリプトを、OSのコマンドラインから実行します。	OSのコマンドラインで次のように入力します。 PhyreStation.exe -workspace= "<path>\MyWorkspace.xml" script= "process.lua" -scriptQuit="true" -scriptLog="scriptLog.txt"	
7. スクリプト process を自動化バッチファイルに組み込みます。	スクリプトの実行をより大きなジョブやワークフロー内に組み込む必要性に迫られることが、よくあります。PhyreEngine™ Game Templatesはその典型例です。たとえば、 UnstoppableSpeedSmash サンプルの ArtWork ディレクトリには、バッチファイル updatePSSGFiles.bat が含まれています。そのバッチファイルはさらに3つのバッチファイルを実行しますが、それらのバッチはそれぞれ異なるスクリプトを使ってPhyreStationを実行することで、そのサンプルで必要になるアセットの処理に必要なワークフローを実行します。	PhyreEngine™ Game Templatesは、特定ジャンルのゲームでPhyreEngine™を最大限に活用するための方法を示したサンプルです。また、これらは、PhyreEngine™を使ってソフトウェアを開発する場合の典型的なワークフローを示したものでもあります。
8. 演習はこれで終了です。		

サンプルスクリプト – process.lua

```
-- 既存の monkee.PSSG から新しい monkee.PSSG データベースを作成しますが、
-- シーン内でのレンダリングに最小限必要なアセットを含めます。
-- これには、猿をレンダリングし、表示し、照明を当てるために必要なアセットが含まれますが、
-- アニメーションデータは一切含まれません。

-- 

srcDb = "monkee.PSSG"
destDb = "monkeeDst.PSSG"
PSSG = os.getenv("SCE_PSSG")
```

```

outDir = "C:\\\\<Your path>"

--
-- カレントワークスペース内のデータベース monkee.PSSG を解決します
--

ResolveLinks( srcDb )

--

-- PhyreEngine(TM) がロードしたすべてのデータベースを取得します
-- PhyreEngine(TM) 内部データベースとソースデータベースは削除します
--

print( "Acquire databases..." )
local dbList = { GetDatabases() }
local dbListTextures = dbList
for i,db in dbList
do
    if db == "PSSGInternalDatabase" or db == srcDb
    then
        dbListTextures[ i ] = nil
    end
end

----

-- 新しいデータベースを作成します
--

print( "Create new database..." )
NewDatabase( outDir, destDb )

----

-- 必要なオブジェクトを再帰的にコピーした後、
-- 不要なオブジェクトを削除します
--

print( "Copying and deleting objects..." )
CloneObjectDeep( srcDb .. "#/root", destDb )
DeleteObject( destDb .. "#/top" )
DeleteObject( destDb .. "#/side" )
DeleteObject( destDb .. "#/front" )
local roots = { GetObjectsOfType( destDb, "ROOTNODE" ) }
for i,r in roots
do
    RemoveUnusedLeafNodes( r )
end
ResolveLinks( destDb ) -- PhyreStation 用。新しいオブジェクトを更新します
--

-- 新しいデータベースにすべての外部テクスチャをコピーします
--

local newTextureList = {}
for i,a in dbListTextures
do
    local textureList = { GetObjectsOfType( a, "TEXTURE" ) }
    for j,b in textureList
    do
        local result,newTexture = CloneObject( b, destDb )
        print( "Copying " .. b .. ". New name is " .. newTexture )
        table.insert( newTextureList, j, newTexture )
    end
end

--

-- シェーダインスタンスを更新し、内部テクスチャを使用するようにします

```

```

--- (テクスチャの割り当て方が間違っていますが、このようなことは、サンプル以外では通用しません)
---
local shaderList = { GetObjectsOfType( destDb, "SHADERINSTANCE" ) }
for i,a in shaderList
do
    local newTexture = newTextureList[ i ]
    print( "Update attribute for " .. a .. " texture " .. newTexture )
    SetObjectProperty( a, "Parameters", 0, newTexture )
end

---
--- 新しいデータベースを保存します
---
SaveDatabaseCompressed( destDb )

```

演習 6：パーティクルシステムをセットアップする

この演習では、PhyreEngine™のインスタンス内に挿入して実行する準備の整った PhyreEngine™ Particle System をセットアップする方法を示します。

この演習を始める前に、これまでの演習がすべて完了しているものと仮定しています。

ステップ	手順	追加情報
1. PhyreStation を実行します。		
2. パーティクルデータベースを作成します。	カレントワークスペースまたは新しいワークスペースを使って新しいデータベースを作成し、それに particles.pssg という名前を付けてます。	
3. 可視パーティクルエミッタノードを作成します。	particles.pssg データベースを選択し、ルートノードを作成します。 Scene-Graph ビューでルートノードを選択し、新しい Visible Particle Emitter Node オブジェクトを追加します。	Visible Particle Emitter Node オブジェクトは Visible Render Node の下に表示されます。代わりに Particle Emitter Node を使用することもできます。
4. パーティクルシステムを作成します。	Visible Particle Emitter Node オブジェクトを選択し、コンテキストメニューから Create Particle System... を選択します。すると、Particle Editor ウィンドウが開きます。	

ステップ	手順	追加情報
5. 必要なプロパティを編集します。	<p>パーティクルシステムを有効化するために必要なプロパティを編集します。</p> <p>System properties > System description グループボックスで、次のフィールドを編集します。</p> <p>System name: colorWheelSystem と入力します</p> <p>Definition: ファイル colorWheel.xml を見つけます</p> <p>System Type: Packetized を選択します</p> <p>System properties > System shader グループボックスで、次のフィールドを編集します。</p> <ul style="list-style-type: none"> • Texture: ファイル colorWheel.bmp を見つけます • Vertex program: ファイル quadParticleRenderVert.cg を見つけます • Fragment program: ファイル particleRenderFrag.cg を見つけます • Source blend: Source Alpha を選択します • Destination blend: One を選択します <p>上記が正しく入力されると、Create System ボタンが有効になります。このボタンをクリックします。Particle Editor に新しいパーティクルシステムが表示されます。</p>	<p>ここで編集しなかったプロパティの値はデフォルト値になりますが、それらはそのままにしておいてかまいません。</p> <p>この演習で必要になるファイルは、PhyreEngine™ の Advanced サンプル particleUberModifier に含まれています。</p> <p>このステップが完了したら、続いて、エディタウィンドウ内で有効化されている他のプロパティを編集することができます。編集するたびに、その変更がすぐに反映されます。</p>
6. パーティクルシステムをコンパイルします。	<p>particles.pssg データベースを右クリックし、そのコンテキストメニューから Compile Particle Systems を選択します。このコマンドは、個別の定義ファイルと、このパーティクルシステム用に最適化されたパーティクルモディファイアを含む必要なソースファイルを生成します。</p>	<p>このステップにより、新しいパーティクルシステムが準備されます。</p> <p>パーティクルシステムのターゲットプラットフォームによっては、パーティクルシステムのコンパイルを行う前に、Compile Cg Programs コマンドの 1つをデータベースに対して実行しなければならない可能性があります。</p>
7. パーティクルシステムを保存します。	<p>particles.pssg データベースを選択し、コンテキストメニューから SaveDatabase を選択します。</p>	

ステップ	手順	追加情報
8. このパーティクルデータベースを PhyreEngine™で使用します。	新しいパーティクルシステムデータベースは利用可能な状態になっています。パーティクルシステムデータベースを見つけ、それを、PhyreEngine™がパーティクルシステムを使用する際にアクセスする場所にコピーします。	さらに、Compile Particle Systems コマンドによって生成された個別のソースファイルをプロジェクトに含め、コンパイルする必要があります。最適化されたパーティクルモディファイアが生成されたら、そのモディファイアを PhyreEngine™に登録する必要があります。
9. 演習はこれで終了です。		

用語集

この用語集では、PhyreStation に固有であるか、あるいは一部の読者にはなじみが薄いと思われる、技術的な頭字語やその他の用語について説明します。

用語	定義
アクション	ユーザによって実行されるコマンド。通常はコンテキストメニューから実行されます。「コマンド」を参照してください。
Animation Editor	「波形エディタ」の特殊バージョン。
API	アプリケーションプログラミングインターフェース (Application Programming Interface)。
アプリケーションインスタンス ID	同一プラットフォーム上で実行されているすべての PhyreStation インスタンスを一意に識別する数値。これを使えば、作業実行時のターゲットとして特定のインスタンスを指定できます。
アプリケーションプリファレンス	PhyreStation のアプリケーション全体の設定に対する「ユーザプリファレンス」。「ワークスペースプリファレンス」を参照してください。
アートアセット	アートデータの項目または項目の集まり。
オートコンプリータ	行編集ボックスに入力された情報に基づいて、可能性のある完全な選択肢を表示するポップアップメニュー。リスト内のいずれかの項目を選択すると、その項目が、まるでユーザ自身が入力したかのように入力されます。
バッチコマンド	内部的に 1 つ以上のセカンダリコマンドを生成する「コマンド」、あるいは GUI で複数のオブジェクトが選択された状態で実行される「コマンド」。
バッチモード	「コマンドラインインタフェース」からスクリプトを指定してバッチモードで PhyreStation インスタンスを実行した場合の、PhyreStation の実行モード。アプリケーションのメインウィンドウは表示されません。この動作モードは、ある PhyreStation コマンドラインパラメータによって指定されます。
動作アニメーション	パーティクルシステムのアニメーション動作。
壊れたデータベース	正常に解決されなかった「PhyreEngine™データベース」。
COLLADA™	アートアセット用のファイル形式。
コマンド	主要な PhyreStation 処理には必ず、コマンドの実行が関係しています。コマンドの定義元は、「PhyreStationDLL」コンポーネント、PhyreStation 自体のいずれかです。PhyreStation はコマンドをさまざまな方法で呼び出します。
コマンドラインインタフェース	コマンドラインインタフェース (CLI) またはコマンドラインウィンドウは、システムコマンドを実行するための OS ウィンドウです。
コマンドラインインターフェリタ	実行するスクリプトコマンドまたは Lua コマンドを入力可能な、 Command Window の行編集ボックス。
コマンドラインパラメータ	「コマンドラインインタフェース」から PhyreStation を使用する場合に指定可能な追加パラメータ。
Command Window	Lua コマンドまたはスクリプト対応コマンドの実行に使用される PhyreStation ウィンドウ。
カスタムグループ	ユーザ定義のオブジェクトセット。ラベルと、カレントワークスペースに含まれるオブジェクトのサブセットが割り当てられます。
Custom Groups フィルタ	Workspace Explorer フィルタ ビューの 1 つ。カレントワークスペースのユーザ定義「カスタムグループ」が表示されます。
コネクタ	「ノードコネクタ」を参照してください。
カレントワークスペース	PhyreStation に現在表示されているワークスペース。
データベース	「PhyreEngine™データベース」を参照してください。
Databases フィルタ	Workspace Explorer フィルタ ビューの 1 つ。カレントワークスペース内に含まれる「PhyreEngine™データベース」が表示されます。
DNet	PhyreEngine™ Debug ユーティリティライブラリと通信プロトコル。

用語	定義
DNet Communication ツールバー	ホストとのネットワーク通信に使用される PhyreStation アプリケーションツールバー。
フィルタビュー	Workspace Explorer ウィンドウでは、ワークスペース内のオブジェクトのさまざまなビューを表示できます。
Find ダイアログ	PhyreStation のダイアログの 1 つ。これを使えば、ワークスペース内のオブジェクトを検索できます。
フレームウィンドウ	1 つ以上の「タブ」 ウィンドウを含むウィンドウ。これは「フローティング状態」 にすることも、「ワークエリア」 にドッキングさせることもできます。
ゴースト	「グループゴーストオブジェクト」 を参照してください。
グラフ	PhyreStation の「ワークエリア」 の 3 つの Graph Editor (Target Blender Editor 、 Modifier Network Editor 、および Modifier Network Instance Editor) のいずれかを指します。「ネットワーク」とも呼ばれます。
グラフノード	「グラフ」 内の図形ノード。互換性のあるグラフノードは、「ノードコネクタ」 経由で互いに接続できます。
グループ	単一のユーザ定義ラベル。これにはオブジェクト表現を割り当てても、割り当てなくてもかまいません。「カスタムグループ」 を参照してください。
グループゴーストオブジェクト	現在ワークスペース内に存在していないオブジェクトを表現したオブジェクト。
グループオブジェクト	ワークスペースに含まれる実際のオブジェクトを表現したオブジェクト。グループオブジェクトは「Custom Groups フィルタ」 ビューに表示されます。
GUI	グラフィカルユーザインターフェース (Graphical User Interface)。
GUI モード	「コマンドラインインターフェース」 からパラメータを指定せずに PhyreStation インスタンスを実行した場合や PhyreStation のアイコンをクリックした場合の、PhyreStation の実行モード。アプリケーションのメイン ウィンドウが表示されます。
Help ウィンドウ	PhyreStation のオンラインヘルプ ウィンドウ。
ハイライト表示	1 つ以上のウィンドウ内で「選択」 され、かつ可視状態になっている、ワークスペース内のオブジェクトを指します。ワークスペースのオブジェクト選択の一部として表示されているオブジェクトを指す場合もあります。
内部データベース	「PhyreEngine™ 内部データベース」 を参照してください。
リンク	異なる「PhyreEngine™ データベース」 に含まれる 2 つの「PhyreEngine™ オブジェクト」 間における接続または参照。
リンク先データベース	別の「PhyreEngine™ データベース」 から参照されている「PhyreEngine™ データベース」。「Databases フィルタ」 ビューに表示されます。
最上位データベース	カレント「ワークスペース」 に追加された「PhyreEngine™ データベース」。「Databases フィルタ」 ビューで表示可能です。
リンク解決	「PhyreEngine™ データベース」 のすべての参照先要素（すべての依存先も含む）の検証が成功すると、そのデータベースは「リンク解決済み」 であると言われます。 データベースのリンクが何らかの理由で失敗した場合、そのデータベースは「リンク未解決」となります。
Log ウィンドウ	PhyreStation のウィンドウの 1 つ。重要なユーザ処理や「コマンド」 からのフィードバックメッセージをロギングします。
Lua	高レベルのプログラミング言語。「スクリプトコマンド」 の自動化機能を提供する目的で、PhyreStation に組み込まれています。
メインウィンドウ	PhyreStation のメインアプリケーション ウィンドウ。
メニューバー	メインウィンドウのメニューバーは、「タイトルバー」 の下側に表示された水平方向に広がる領域であり、タスク実行用のメニューを含んでいます。
Modifier Network Editor	「PhyreEngine™ モディファイアネットワーク」 オブジェクトに関連付けられたグラフ/ネットワーク用エディタ。

用語	定義
Modifier Network Instance Editor	「PhyreEngine™モディファイアネットワークインスタンス」オブジェクトに関連付けられたグラフ/ネットワーク用エディタ。
ネットワーク	「グラフ」を参照してください。
ノード	「グラフノード」、「PhyreEngine™ノードオブジェクト」のいずれかを参照してください。
ノードコネクタ	他の「グラフノード」への接続に使用される、グラフノード上のインターフェース。
オブジェクト	ワークスペース内のオブジェクト。通常は「PhyreEngine™オブジェクト」の表現。
Objects フィルタ	Workspace Explorer フィルタビューの 1 つ。現在ロードされているワークスペース内に含まれるすべての解決済みデータベースのすべての「PhyreEngine™オブジェクト」が表示されます。
OS	オペレーティングシステム (Operating System)。
Particle Editor	PhyreEngine™ Particle System の作成または編集に使用される PhyreStation ウィンドウ。
パーティクルシステム	PhyreEngine™ Particle System。他のパーティクルストリームに挿入された特定のパーティクルストリームに基づいて新データの更新または計算を行うプロセスの集合体。
プリファレンス	「ユーザプリファレンス」を参照してください。
プライマリオブジェクト	「タブ」 ウィンドウまたはエディタ（機能による）の「対象」オブジェクト。対象のプロパティを変更すると、「セカンダリオブジェクトウィンドウ」の「対象」も変更される可能性があります。
Properties Viewer ウィンドウ	ワークスペース内のオブジェクトの属性を表示する PhyreStation ウィンドウ。一部の属性は編集可能な場合があります。
PhyreEngine™	高レベルのグラフィックスレンダリングシステム。シーニングラフなどのコンポーネントが含まれています。
PhyreEngine™データベース	「PhyreEngine™オブジェクト」の格納された集まり。
PhyreEngine™内部データベース	すべてのワークスペース内に常に存在する「PhyreEngine™データベース」。すべての開発プロジェクトで通常必要となる必須「PhyreEngine™オブジェクト」とデフォルトオブジェクトが含まれています。
PhyreEngine™ノードオブジェクト	「PhyreEngine™オブジェクト」の型。ノードオブジェクトを階層的に相互接続すれば、グラフィカルシーンの物理構造や論理構造を表現することができます。シーンをレンダリングする際には、特殊なノード型を派生させて特定の機能を実行させることができます（シーンに光源やジオメトリを追加するなど）。「シーニングラフ」を参照してください。
PhyreEngine™オブジェクト	「アートアセット」を表現したもの。PhyreEngine™の基本的な構築ブロックです。「PhyreEngine™データベース」には複数のオブジェクトをいっしょに格納できます。
PhyreEngine™オブジェクトリンク ID	PhyreEngine™では、「PhyreEngine™データベース」内のすべての PhyreEngine™オブジェクトに一意の名前が割り当てられます。PhyreEngine™リンク ID は次のように、「PhyreEngine™ データベース」の名前に連結されます。 <i>databaseName#objectName</i> 。
PhyreEngine™オブジェクト型	「PhyreEngine™オブジェクト」のカテゴリ。
PhyreStation	「PhyreEngine™オブジェクト」と「PhyreEngine™データベース」を自動化機能を用いて管理するための GUI ツール。
PhyreStationDLL	PhyreStation のコンポーネントの 1 つ。プログラマはこれを介して、「PhyreEngine™オブジェクト」コマンドやカスタム「PhyreEngine™オブジェクト」を追加または変更できます。

用語	定義
PhyreStation エラー	PhyreStation の実行中にアプリケーションコード内で発生したエラー。PhyreStation のエラーは、非致命的 (PhyreStation は回復可能) と致命的 (PhyreStation は適切な終了を試みる) という、2つのカテゴリに分けられます。
Qt	「Trolltech Qt」はクロスプラットフォームな GUI ツールキットライブラリであり、PhyreStation に組み込まれています。
RenderDataType	「Particle System の動作要素」を指定する際に必要となる、結果タイプに付けられた PhyreEngine™名。
Render Viewer	Render Viewer は、レンダリングされたシーンを、このウィンドウの対象であるカメラノードオブジェクトの視点から表示するウィンドウです。
解決済み	「リンク解決」を参照してください。
シーングラフ	「PhyreEngine™オブジェクトノード型」とジオメトリから成る階層的なセンブリであり、グラフィカルシーンの物理構造や論理構造を表します。「Scene-Graph フィルタ」を参照してください。
Scene-Graph フィルタ	Workspace Explorer フィルタビューの1つ。カレントワークスペース内のすべての「PhyreEngine™シーングラフ」が表示されます。「シーングラフ」を参照してください。
スクリプト	「スクリプトファイル」を参照してください。
スクリプトコマンド	「PhyreStationDLL」経由で PhyreStation スクリプト処理用として登録されたコマンド。PhyreStation の自動化機能を提供します。スクリプトコマンドは、 Command Window から直接実行することもできますし、「スクリプトファイル」内に組み込むこともできます。
スクリプトファイル	一連の Lua コマンドまたは PhyreStation 「スクリプトコマンド」を含むテキストファイル。
Scripts フィルタ	Workspace Explorer フィルタビューの1つ。現在の「ワークスペース」のセッションに関連付けられたすべてのスクリプトが表示されます。
セカンダリオブジェクトウィンドウ/セカンダリビュー	ワークスペース内のどこか別の場所にある別のオブジェクト（「プライマリ」オブジェクト）に依存するオブジェクトを「対象」に持つ「タブ」ウィンドウ（機能による）。「プライマリ」オブジェクトの属性が変更されると、「プライマリ」オブジェクトとの関係により、「セカンダリ」オブジェクトにも変更が反映されることがあります。
Segment Set Viewer	「PhyreEngine™セグメントセット」オブジェクトをレンダリングする PhyreStation ウィンドウ。
Shader Group Viewer	単一の「PhyreEngine™ Shader Group」オブジェクトに関する情報を表示する PhyreStation ウィンドウ。シェーダプロセスのパラメータや属性の変更機能（限定版）を一部提供します。
Shader Instance Viewer	単一の「PhyreEngine™シェーダインスタンス」オブジェクトを表示するための PhyreStation ウィンドウ。
Shader Instances Viewer	カレントワークスペース内のすべてのロード済みかつ解決済みの「PhyreEngine™データベース」に含まれる、すべての「PhyreEngine™シェーダインスタンス」オブジェクトを表示するための PhyreStation ウィンドウ。
Shader Program Viewer	「PhyreEngine™ Shader Program」オブジェクトとそのプログラムを表示するための PhyreStation ウィンドウ。シェーダプロセスシェーダプログラムのパラメータや属性の変更機能を一部提供します。
スクリプト GUI モード	「コマンドラインインターフェース」からスクリプトを指定して PhyreStation インスタンスを実行した場合の、PhyreStation の実行モード。スクリプトの終了後にアプリケーションのメインウィンドウが表示されます。この動作モードは、ある PhyreStation コマンドラインパラメータによって指定されます。
ステータスバー	「メインウィンドウ」の最下部に表示された水平方向に広がる領域であり、状態情報やその他のフィードバックが表示されます。

用語	定義
静的モード	ユーザが別のワークスペースオブジェクトを選択しても「対象」が変更されないPhyreStation「タブ」ウィンドウ。
対象	一部のウィンドウ（ Render Viewer ウィンドウなど）では、特定のワークスペースオブジェクト（通常は「PhyreEngine™ オブジェクト」）が対象として扱われます。ウィンドウには、その対象または関連オブジェクトに関する情報が表示されます。対象は通常、「フィルタビュー」からユーザによって選択されます。
システムアニメーション	パーティクルシステムのアニメーション動作項目。
システム要素	パーティクルシステムの動作要素。
Target Blender Editor	「PhyreEngine™ アニメーションターゲットブレンダ」オブジェクトに関連付けられたPhyreStation グラフ/ネットワーク用エディタ。
Texture Viewer	「PhyreEngine™ テクスチャ」オブジェクトの画像を表示するPhyreStation ウィンドウ。
タイトルバー	メインウィンドウのタイトルバーは、「メインウィンドウ」の最上部に表示された水平方向に広がる領域であり、そこには通常、カレントワークスペースの名前が表示されます。
ツールバー	メインウィンドウのツールバーは、「メニューバー」の下側に表示された水平方向に広がる領域であり、一般的なタスクを実行するためのツールバーボタンから構成されます。
ユーザプリファレンス	PhyreStation の一部の機能に対する設定。ユーザプリファレンスはユーザによる変更が可能であり、PhyreStation のセッション終了後も持続されます。
Untitled ワークスペース	空のワークスペース。PhyreStation の起動時に自動的に開かれます（対応する「ユーザプリファレンス」が有効になっている場合）。
ビューフィルタ	「Workspace Explorer」で、利用可能なフィルタのリストから選択されます。これにより、特定の情報の表示方法が決まります。
波形エディタ	「PhyreEngine™ アニメーションチャンネル」オブジェクトなどの波形特性を持つデータを表示/編集するためのPhyreStation エディタ。
余白部分	ウィンドウの、何も表示されていない領域。
ワークエリア	PhyreStation のメインウィンドウの領域の1つ。この領域内では「フレーム ウィンドウ」または「ツールバー」をドッキングさせることができます。
ワークスペース	PhyreStation の作業環境。ここには一連の「PhyreEngine™ データベース」、関連する「スクリプト」ファイル、PhyreStation グループなどのワークスペース「オブジェクト」を含めることができます。
Workspace Explorer ウィンドウ	利用可能なフィルタのリストから「ビューフィルタ」を選択できるようにするウィンドウ。これを使えば、表示する情報の種類を管理し、情報またはオブジェクトを選択した後、最近の選択に対してタスクを実行することができます。
ワークスペースファイル	「ワークスペース」の情報とユーザプリファレンスが格納された XML ファイル。
ワークスペースプリファレンス	PhyreStation の「ワークスペース」の設定に対する「ユーザプリファレンス」。「アプリケーションプリファレンス」を参照してください。