計算結果のリアルな表現を目指して

- インテル® oneAPI レンダリング・ツールキット -

久保寺　陽子

本日の内容

- インテル® oneAPI レンダリング・ツールキットの概要
  - インテル® Embree
  - インテル® オープン・イメージ・デノイズ
  - インテル® オープン・ボリューム・カーネル・ライブラリー (インテル® オープン VKL)
  - インテル® OSPRay Studio
  - USD Hydra 用インテル® OSPRay プラグイン
  - インテル® オープン・パス・ガイディング・ライブラリー (インテル® オープン PGL)
- インテル® OSPRay とプログラミング・モデル
- デモ
- 技術情報

インテル® oneAPI ツールキット

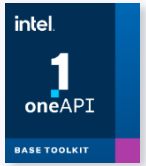
CPU から XPU までカバーする開発ツールの完全なセット

oneAPI

インテル® oneAPI ベース・ツールキット

ネイティブコード開発者

C++、データ並列 C++ アプリケーションおよびインテル® oneAPI ライブラリー・ベースのアプリケーションを構築するためのハイパフォーマンス・ツールの基本セット



ドメイン固有のアドオン・ツールキット

インテル® oneAPI HPC ツールキット

スケーラブルで高速な Fortran OpenMP* および MPI アプリケーションを開発

インテル® oneAPI IoT ツールキット

ネットワークのエッジで実行する、効率的な、信頼性の高いソリューションを構築

特別なワークロード

インテル® AI アナリティクス・ツールキット

最適化された DL フレームワークとハイパフォーマンスの Python* ライブラリーでマシンラーニングとデータ・サイエンス・パイプラインを高速化

インテル® oneAPI レンダリング・ツールキット

ハイパフォーマンスで高忠実度のビジュアライゼーション・アプリケーションを作成

oneAPI 対応のツールキット

データ科学者/AI 開発者向け

OpenVINO™

インテル® ディストリビューションの OpenVINO™ ツールキット

エッジからクラウドまでハイパフォーマンスな推論とアプリケーションをデプロイ

写実性の高いレンダリング

インテル® oneAPI レンダリング・ツールキット

インテル® Embree

ハイパフォーマンスで機能豊富なレイトレーシング & 写実的なレンダリングを実現

インテル® OSPRay

スケーラブルで移植性の高い分散レンダリング API

インテル® オープン・イメージ・デノイズ

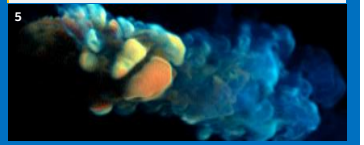
AI によって高速化されたデノイザーが優れたビジュアル品質を実現

インテル® OpenSWR

ハイパフォーマンスでスケーラブルな OpenGL* 互換ラスタライザー

インテル® オープン VKL

3D 空間データ処理のレンダリング & シミュレーション



インテル® OSPRay Studio

新しくシーン・グラフ・アプリケーションの GUI を備えたリアルタイム・レンダリング

インテル® OSPRay for Hydra

ユニバーサル・シーン・ディスクリプション Hydra レンダリング・サブシステムにレンダリング・ツールキット・ライブラリーをプラグインで接続

インテル® オープン PGL (pre -1.0)

物理ベース・レンダリングでアダプティブ分散リダクション技術を実装するライブラリー

インテル® Embree 4 から、外付けGPUを初めてサポート:
- **インテル® Arc™ A シリーズ・グラフィックス**
- **インテル® データセンター GPU フレックス・シリーズ**

新たに
リリース!

1 Avengers: Infinity War - Digital Domain, Marvel Studios, Chaos Group V-Ray
2 Moana Island Scene, Walt Disney Animation Studios , publicly available dataset: 15fps+,~160 billion prims
3 Scene courtesy of Frank Meinl
4 Model from Leigh Orf at University of Wisconsin. For more tornado visualization, visit Leigh Orf's site
5 Smoke volume, data courtesy OpenVDB example repository
6 Copyright Bentley Mortors Limited
7 USD format version: Converted scene by Jay-Artist CC, modified by Bitterli.

詳細は:
www.xlsoft.com/jp/products/intel/oneapi/

インテル® oneAPI レンダリング・ツールキット
プラットフォーム編 (2023.1バージョン)

サポート・プラットフォームの詳細はこちら⇒

- **サポート・ハードウェア**
  - インテル® 64 アーキテクチャー
    - Intel Atom® プロセッサー
    - インテル® Core™ プロセッサー・ファミリー
    - インテル® Xeon® プロセッサー・ファミリー
    - インテル® Xeon® スケーラブル・プロセッサー・ファミリー
  - ARM* アーキテクチャー
    - Apple* M1

注) インテル® SSE 4.2 命令をサポートしていること

- Xe-HPG または Xe-HPC アーキテクチャー以降 (インテル® Embree のみ)
- インテル® Arc™ A シリーズ・グラフィックス
- インテル® データセンター GPU マックス・シリーズ
- インテル® データセンター GPU フレックス・シリーズ

注) GPU を使用する場合には、対応するグラフィックス・ドライバーが必要

- **サポート OS**

ホスト OS/ターゲット OS		開発ツール
Linux*	Red Hat* Enterprise Linux* (RHEL) 8、RHEL 9	インテル® oneAPI ツールキット向け診断ユーティリティー、インテル® oneAPI ツールキット向け Visual Studio* Code (VS Code) 拡張、Visual Studio* 2019、または 2022、Eclipse*
	Rocky Linux* 8、9、Ubuntu* LTS 20.04、22.04	
	SUSE* Linux Enterprise Server 15 SP3、SP4	
	Fedora* 36、37、Debian* 9、10、11、Amazon* 2、2022、WSL* 2	
Windows*	Windows* 10、Windows* 11、Windows Server* 2019、2022	
macOS*	インテル® プロセッサー搭載の Mac*: macOS* 12、13 Apple* M1 プロセッサー搭載のMac*: macOS* 12 以降	

インテル® Embree

高性能かつ高機能なレイトレーシングとフォトリアリスティック・レンダリング

主な機能 / 利点

- 高度に最適化されたレイトレーシング・カーネル・ライブラリー
- プロのレンダリングやサイエンス・ビジュアライゼーション市場で広く採用 (インテル® OSPRay 経由)
- カーネルは最新のインテル® プロセッサー向けに最適化
  - インテル® ストリーミング SIMD 拡張命令 [4.2] からインテル® アドバンスト・ベクトル・エクステンション 512 までサポート

The Grinch, Courtesy Illumination Studios

バージョン 4.0 の新機能

- 外付け GPU をサポートを追加
- インテル® Arc™ GPU
- インテル® データセンター GPU フレックス・シリーズ
- コンパイラーをインテル® oneAPI DPC++/C++ コンパイラーに移行

Intel® Embree with Corona Renderer

https://www.embree.org/ (英語) (Linux*、Windows*、macOS*)

インテル® オープン VKL

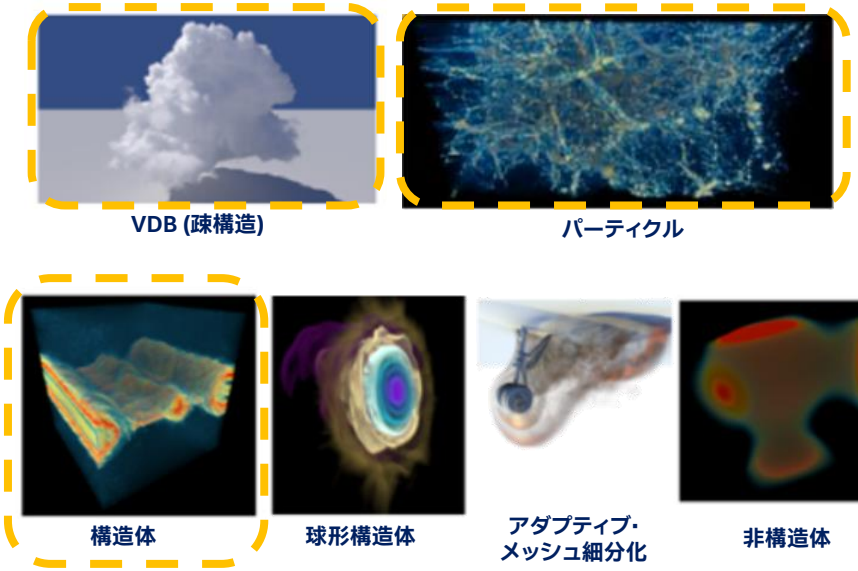
主な機能/利点

- 3D 空間データのレンダリングとシミュレーションを可能に
- インテル® プラットフォーム用に最適化されたボリュームデータ処理アルゴリズム

バージョン 1.3 の新機能　→詳細はこちら (英語)

- Cmake で OPENVKL_ISA_AVX512SKX_8_WIDE オプションの使用により、インテル® AVX-512 8-wide CPU デバイスモードを追加
- VDB ボリューム: nodesPackedDense、nodesPackedTile パラメーターで、性能向上のための一時的なパックド/連続したデータレイアウトのサポートを追加
- パーティクル・ボリューム: メモリーの効率化と性能向上 Superbuild の依存関係が最新バージョンに更新
- ISPC の対応バージョンは v1.18.0 以降
- レンダリング・キットの最新バージョンはインテル® Embree v4.0.0 と rkcommon v1.11.0 が必要 (バージョン1.3.2)
- ARM* アーキテクチャーのサポート

ボリュームタイプ



API

改良版

- Rayをベースにした反復間隔
- ボリューム・オブザーバー
- サンプリング
- 勾配計算
- 陰関数の等値面
- マルチ属性データ

https://www.openvkl.org/ (英語) (Linux*、Windows*、macOS*)

インテル® オープン・イメージ・デノイズ

主な機能/利点

- レイトレースを採用したレンダリング・アプリケーションのレンダリング時間の大幅な短縮
- ピクセルあたりの広範囲のサンプル (spp) を処理するようにトレーニングされた効率的な深層学習ベースのノイズ除去フィルター。プレビューと最終フレームのレンダリングに最適
- 柔軟な C/C++ API によって、ほとんどのレンダリング・ソリューションへ簡単な統合が可能
- ニューラル・ネットワークのトレーニング・コードも含む
- 実行時のユーザーがトレーニングしたモデルを指定可能
- ノイズ除去品質の向上 (例: アーティファクトの発生が少ない、いくつかの場合には不明瞭さが少ない)

バージョン 1.4.3 の新機能

- インストールされた macOS* バイナリーの、ハードコードされたライブラリー・パスを修正
- インテル® oneDNN カーネルのインテル® VTune™ プロファイラーによるプロファイリングをデフォルトで無効化。Cmake のオプションで有効にする
- 公式バイナリーでインテル® oneTBB 2021.5.0 にアップグレード

レイトレーシング用 AI Denoising フィルター
(オフライン & インタラクティブ)

複数入力の機能バッファー
(HDR/LDF カラー、アルベド、通常)

ライトマップのサポート

ほとんどすべての CPU で
実行 (インテル® CPU/
インテル® Core™ CPU)

簡単 & 安定した C/C++ API

PyTorch

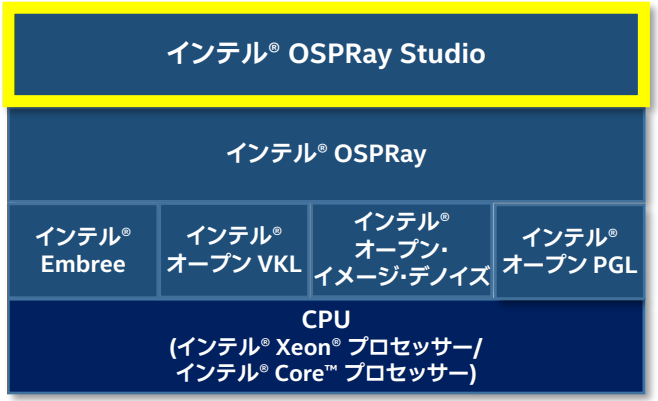
トレーニング・
ツールキット同梱

https://www.openimagedenoise.org/ (英語) (Linux*、Windows*、macOS*)

改良版

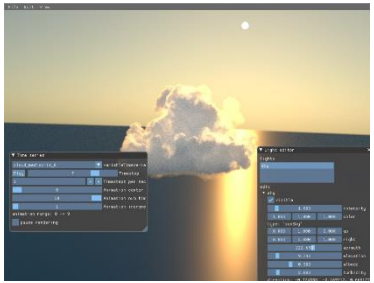
インテル® OSPRay Studio

インテル® OSPrayの機能のすべてをサポートするシーングラフ・ベースの使いやすいアプリケーション



主な機能:

- プログラミンは不要。サポートされた形式のファイルを読み込んで、表示させるだけ
- サポートしているファイル形式 – obj/mtl、glTF、vdb、構造化/非構造化のボリューム・フォーマット
  https://github.com/ospray/ospray_studio/blob/master/FEATURES.md (英語)
- OSPRay 光源タイプをすべてサポート (平行光源、点光源、スポットライト、面光源、HDRI、環境光、Sun-Sky、放射光)
- マテリアル、光源、カメラエディター、カメラパス制御用のシーンビルダー/エディター GUI を備える
- シーンとイメージのエクスポート: ウィンドウサイズに依存しない非常に大規模なイメージ出力をサポートし、オフラインでのイメージ生成用のヘッドレスのバッチモードにも対応
- プラグイン経由のカスタマイズ: プラグインは実行時にロードする共通オブジェクト・ライブラリーでシーングラフとアプリケーション UI をさまざまな面で拡張



バージョン 0.12.0 の新機能:

- 新フォーマットのサポート: Tiny OpenEXR*、Tiny DNG* (TIFF フォーマットのサポート)、OpenImageIO*

https://www.ospray.org/ospray_studio (英語) (Linux*、Windows*、macOS*)

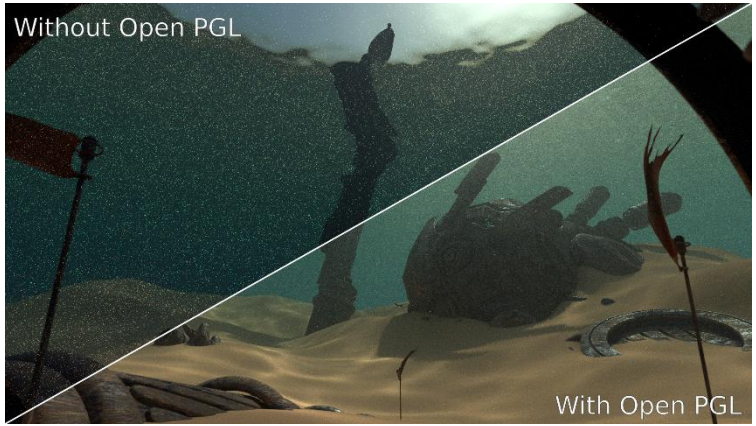
USD Hydra 用インテル® OSPRay プラグイン

ピクサー社のユニバーサル・シーン・ディスクリプション (USD) 向けのレンダリング・プラグインで、Maya*、Houdini* などのアプリケーションを含む Hydra をサポートするアプリケーションに有用

https://github.com/ospray/hdospray (英語) (Linux*、macOS*)

インテル®オープン・パス・ガイディング・ライブラリー
(プレリリース版)

- 物理ベース・レンダリングでアダプティブ分散リダクション技術を実装するライブラリー
- レンダラーに統合したパスガイドのトレーニング・アルゴリズムにより、レンダリングの大幅な性能向上
- 堅牢で効率的
- 画像データの劣化はなく、性能向上のみ
- インテル® SSE、インテル® AVX、インテル® AVX2、インテル® AVX-512 命令セットで最新のインテル® プロセッサーに最適化
- インテル® オープン PGL はインテル® Embree、インテル® TBB が必要



Path traced image of a variation of the Nishita Sky Demo scene from Blender Studio (CC0) without and with using Open PGL to guide directional samples (i.e., on surfaces and inside the water volume).

https://github.com/OpenPathGuidingLibrary/openpgl **(英語)**

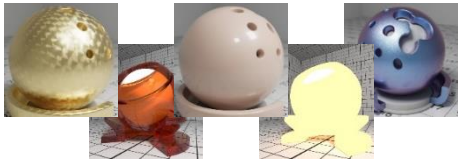
インテル® OSPRay

スケーラブルでポータブルな分散型レンダリング API

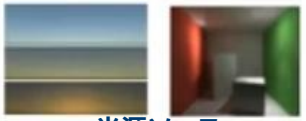
すべてのプラットフォームで実行

統合されたジオメトリー& ボリューム・レンダリング

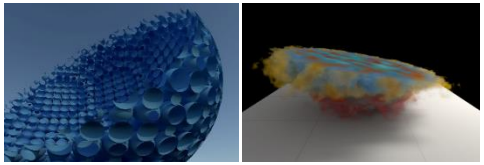
サイエンス・ビジュアライゼーション・ レンダラー & パストレーサー



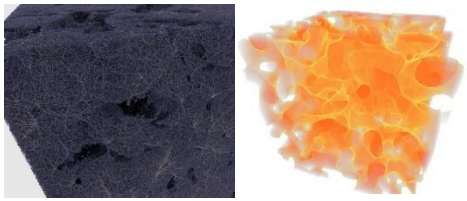
新素材



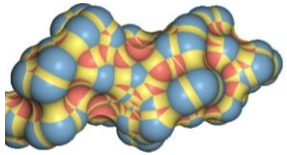
光源ソース
太陽/空　フォトメトリック



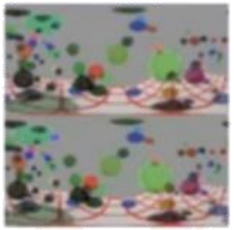
ジオメトリーの切断



12B パーティクル/10TB 大規模 データセット



拡張性
例: SES ジオメトリー、 UStuttgart



VR 用ステレオ
3D パノラマ

流線/パーティクル　球体/陰関数等値面
非多角形ジオメトリー

新機能
改良版

インテル® OSPRay

主な機能/利点

- サイエンス・ビジュアライゼーション向けに開発
- クラスターもサポート可能な、スケーラブルで拡張可能なレイトレーシング・レンダリング・ライブラリーでパストレースとボリューム・レンダリングも含む
- ビジュアライゼーション・アプリケーションを開発するための SDK であり、高レベルのレンダリング API
- スレッディングとベクトル化を効果的に使用して、インテル® CPU 上でインタラクティブで忠実度の高いアプリケーションを作成
- プラグインモジュール経由で拡張性を確保
- ボリューム・レンダリング機能を強化するためにインテル® オープン VKL を統合

バージョン 2.10 & 2.11 の新機能:　→詳細はこちら (英語)

- フレーム・バッファー・チャネルとして、プリミティブ、オブジェクト、インスタンス ID バッファーのサポートを追加
- メッシュとサブディビジョンのジオメトリー向けに面変化属性のサポートを追加
- MPI モジュールの性能向上
- Windows* 版で単一 ISPC ターゲットをサポート
- ARM64 アーキテクチャーで浮動小数点演算 NEON 命令をサポート

https://www.ospray.org/ (英語) (Linux*、Windows*、macOS*)

インテル® OSPRray プログラムの流れ

API の詳細はこちら（英語、PDF版）

1. レンダラーの初期化:

関数 ospInit(int *argc, const char **argv) でインテル® OSPRay にコマンドラインで指定した変数を渡す

2. レンダリングを行うオブジェクトの作成と管理:

レンダラー、カメラ、ワールドなどのオブジェクトを作成

- レンダラー作成: ospNewRenderer() scivis/ao/pathtracer からタイプを選択
- カメラ作成: ospNewCamera() perspective/orthographic/panoramic からタイプを選択
- ワールドの作成: ospNewWorld()

オブジェクトの管理: ospCommit()、ospRelease()、ospRetain()

3. レンダリング結果を格納するフレームバッファの確保:

作成 ospNewFrameBuffer() とクリア ospResetAccumulation()

4. レンダリングの実行: 同期 ospRenderFrameBlocking()/非同期 ospRenderFrame()

5. レンダラーの終了: 関数 ospShutdown()

初期化 (ospInit) のコマンドライン・オプション

詳細はこちら (英語)

パラメーター	詳細
--osp:debug	マルチスレッドを無効にして、さまざまな追加チェックとデバッグ出力を有効にする
--osp:num-threads=<n>	デフォルトで設定されたすべての検知されたハードウェア・スレッドを使用する代わりに n スレッドを使用
--osp:log-level=<str>	ログレベルの設定。使用可能な値: none、error、warning、info、debug
--osp:warn-as-error	エラー・コールバックを通して警告とエラーメッセージを送る。それ例外は、メッセージ・コールバックを通して警告を送る。警告を有効にするには、充分な logLevel が必要
--osp:load-modules=<name>[,...]	初期化の際に 1 つ以上のモジュールをロード。ospLoadModule(name) と同じ
--osp:device=<name>	インテル® OSPRay が作成するデバイスのタイプとして name を使用。 例: --osp:device=cpu はデフォルトで CPU デバイス。使用するデバイスがモジュールで定義されている場合には、最初に --osp:load-modules=<name> を渡す
--osp:set-affinity=<n>	1 の場合、ソフトウェア・スレッドをハードウェア・スレッドに紐付け、0 の場合には無効とする。デフォルトは 0
--osp:device-params=<param>:<value>[,...]	1 つ以上の他のデバイス・パラメーターをセット。ospDeviceSet*(param, value) と同じ

インテル® OSPRay

API のカテゴリー:

- **OSPFrameBuffers:**
  レンダリングされたフレームの最終結果を保持。ピクセルのカラー、深度値、累積情報を含む
- **OSPData:**
  1D データ配列であり、GPGPU の「バッファー」に類似。データ配列は、スカラーおよびベクトルデータ (2、3、4 次元)、デバイスを抽象化するほかのアクター (ほかのデータ配列を含む) への参照
- **OSPGeometry:**
  三角形、球、円柱などの幾何学的サーフェス・プリミティブ
- **OSPVolumes:**
  任意の 3D 位置に対してボリュームレンダラーがサンプリングできるスカラー値を生成可能な 3D スカラーフィールド
- **OSPTransferFunctions:**
  スカラーを RGBA カラーにマップ
- **OSPModels:**
  ジオメトリーとボリュームの集合で、階層の親オブジェクト。OSPModel のベクトルは時間的に変化するデータを表す
- **OSPCameras:**
  レンダラーが計算する主光線を生成
- **OSPRenderers:**
  カメラやモデルを使用してピクセルをレンダリング。scivis と pathtracer の 2 つのレンダラーをサポート

インテル® OSPRay のシーン階層
ワールド (OSPWorld)/インスタンス (OSPInstance)/グループ (OSPGroup)

- **ワールド (OSPWorld)**
  - 複数のインスタンス (OSPInstance)
  - ワールドの光源 (OSPLight)

レンダリングの実行
ospRenderFrame(OSPFrameBuffer, OSPRenderer, OSPCamera, OSPWorld);

- **インスタンス (OSPInstance)**
  - 1 つのグループ (OSPGroup)
  - アフィン変換
  - スケール
  - 回転
  - クォータニオン
  - 移動

ワールド空間への変換

- **グループ (OSPGroup)**
  - ジオメトリック･ボリューム[ ]
  - ジオメトリック･モデル[ ]
  - 光源[ ]

(ローカル空間座標を共有)

Structured Regular/Spherical, AMR(Adaptive Mesh Refinement), Unstructured, VDB Volume

Mesh, Subdivision,Spheres, Curves, Boxes, Planes, Isosurfaces

Photometric, Directional/Distant, Point/Sphere, Spotlight/Ring, Quad, Cylinder, HDRI, Ambient, Sun-Sky Light

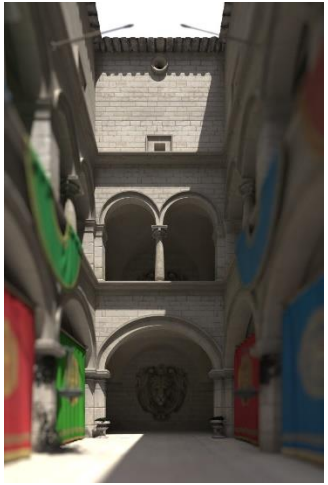
インテル® OSPRay のレンダラー

インテル® OSPRay は複数のレンダラーを提供: ospNewRenderer (scivis/ao/pathtracer)

- **SciVis レンダラー:** サイエンス・ビジュアライゼーション用の高速なレイトレーサーでボリューム・レンダリングとアンビエント・オクルージョン (AO) をサポート
- **アンビエント・オクルージョン (AO) レンダラー:** 性能向上のために SciVis レンダラーの機能のサブセットのみをサポート。メインのシェーディングは、光源を全く考慮しない AO であり、ボリューム・レンダリングをサポート
- **Path Tracer:** ソフトシャドウ、間接照明、写実的なマテリアル、多重散乱のあるボリュームをサポート

インテル® OSPRay のカメラ

詳細はこちら (英語)



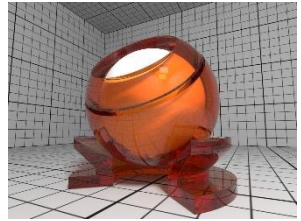
インテル® OSPRay は複数のカメラを提供:

ospNewCamera(perspective/orthographic/panoramic)

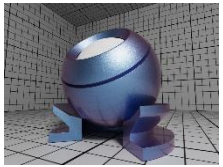
- **Perspective** カメラ: 透視投影を実装する単一レンズを実装。レンダラーに pathtracer を設定した場合は、オプションとして被写界深度とステレオ・レンダリングをサポート
- **Othographic** カメラ: 深度を考慮しない正投影の単一レンズを実装
- **Panoramic** カメラ: ステレオ・レンダリングをサポートする簡単なカメラを実装。カメラの周囲全体を、緯度/経度のマッピングでキャプチャーする。そのためレンダリング後の画像の比率は 2:1 が最適

インテル® OSPRay のマテリアル

詳細情報はこちら (英語)



- **OBJ マテリアル**
  - 拡散、反射、光沢、不透明、法線マップ
  - (Lightware/Wavefront の MTL をベース)
- **Principled**
  - 複数のレイヤーとローブを組み合わせた最も複雑なマテリアル
  - 微小平面分布関数 GGX、Oren-Nayar 拡散反射、エネルギーの節約
- **CarPaint**
  - Principled マテリアルの特別なバージョンで、さまざな種類の自動車用塗料をサポートしており、MettalicPain マテリアルよりも柔軟
- **MettalicPaint**
  - オプションのフレークを含むベースコートとクリアコートで構成される金属塗装マテリアル
- **Metal**
  - パラメーター eta および k、波長に依存する複雑な屈折率によって制御される物理的な金属



- **Alloy**
  - Metal マテリアルと同様だが、より直感的でカラー制御が柔軟
- **Glass**
  - 屈折と体積減衰をサポートしているリアルなガラスマテリアル
- **ThinGlass**
  - 1 つの表面が薄くて、透明な板をモデル化するオブジェクトに有用
- **Luminous**
  - どんな物理オブジェクトでも、四方八方に一様に放射する 1 つの光源とするマテリアル
- **Transfer 関数:** ボリューム・レンダリングのマテリアル
  - Transfer 関数は、ボリュームのスカラー値を色と不透明度にマッピングし、ボリュームの特定の機能を視覚的に強調

インテル® OSPRayのレンダリング
レンダリングの実行:

- **同期レンダリング ospRenderFrameBlocking:**
  - ospray_util.h で定義され、以下と同じ動作

    f = ospRenderFrame(); ospWait(f, OSP_TASK_FINISHED); return ospGetVariance(fb);
- **非同期レンダリング ospRenderFrame:**
  - OSPFuture ospRenderFrame(OSPFrameBuffer, OSPRenderer, OSPCamera, OSPWorld): 関数の戻り値 OSPFuture ハンドルがレンダリングの制御に用いられる
  - ospGetProgress(OSPFuture): レンダリング進行状況の確認
  - ospCancel(OSPFuture): 進行中の非同期レンダリングをキャンセル
  - ospWait(OSPFuture, OSPSyncEvent = OSP_TASK_FINISHED): 非同期レンダリングの終了まで待つ
  - ospGetTaskDuration(OSPFuture): 非同期のレンダリング時間
- **マルチデバイス・レンダリング :**
  - 開発中
  - 現バージョンでは、ISPCDevice delegate 自動作成のみに制限

インテル® OSPRay オブジェクト管理用 API

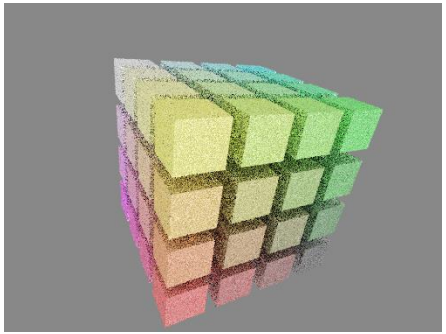
インテル® OSPRay オブジェクト (レンダラー、インスタンス、グループ、カメラなど) のための共通の管理用 API

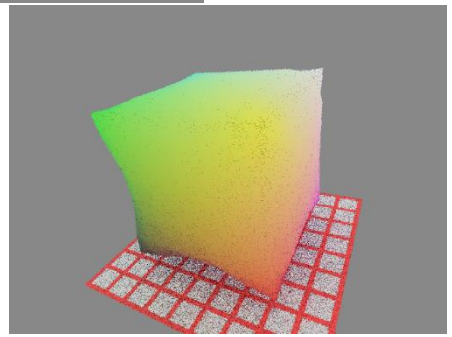
API	詳細
ospCommit()	ospCommit 関数を呼ぶまで、パラメーターはオブジェクトに渡されない。ospCommit 関数を呼ぶ直前まで追加/修正されたすべてのパラメーターは、ospCommit(Object) で同時に渡される
ospRelease()	インテル® OSPRay ではオブジェクトの有効期間を管理するのに参照カウントを使用するので、オブジェクトを明示的に削除することはできない。ospRelease 関数は、削除の代わりにアプリケーションが指定したオブジェクトを必要とせずアクセスしないことを示して、参照カウントを減少させる。カウントが 0 に達すると、オブジェクトは自動的に削除
ospRetain()	アプリケーションがオブジェクトへの参照カウントを複数保持したい場合に使用
ospShutdown()	アプリケーションでインテル® OSPRay の使用を終了したいときに使用。この関数により、使用しているデバイスをクリーンアップするので、静的なオブジェクトを割り当てている場合には、呼び出し側のアプリケーション・プロセスが終了する前に、アプリケーション内で ospShutdown 関数を呼び出すこと

デモ (OSPExamples)

プログラム手順

- プログラムで使用するヘッダーの宣言
- コマンド入力判定と、初期化
- ワールドの設定
- レンダラーの設定とカメラの作成
- フレームバッファー作成とレンダリング





デモ プログラムで使用するヘッダーの宣言

```
#define _GLIBCXX_USE_CXX11_ABI 0
// rkcommon
/* インテル (R) oneAPI レンダリング・ツールキットのさまざまなコンポーネントで使用される C++ のインフラと Cmake ユーティリティーの共通ライブラリー */
#define OSPRAY_CPP_RKCOMMON_TYPES

// rkcommon
#include "rkcommon/math/rkmath.h"
#include "rkcommon/math/vec.h"
#include "rkcommon/utility/ParameterizedObject.h"
#include "rkcommon/containers/TransactionalBuffer.h"
#include "rkcommon/math/AffineSpace.h"

// インテル(R) OSPRay のインクルード
#include "ospray/ospray_cpp.h"
#include "ArcballCamera.h"

// ospray_testing のインクルード
#include "ospray/ospray_testing/ospray_testing.h"

// GL 数値演算ライブラリーをインクルード
#include "glm/vec2.hpp"
#include "glm/vec3.hpp"
#include "glm/vec4.hpp"
#include "glm_box3.h"

// RenderKit_Learning_Path/utils のファイルをインクルード
#include "utils.h"
```

シーンごとにワールドを作成し直すための builder

```
// C++ 標準入出力ライブラリーをインクルード
#include <iostream>
#include <stdexcept>
#include <exception>
#include <vector>
#include <functional>
#include <string>
// argh コマンドライン・パーサー
#include "argh.h"
using namespace ospray;
using namespace rkcommon::math;
using namespace std;
// リストされたさまざまなシーンを表す文字列のベクトル
static const std::vector<std::string> g_scenes = {
  "boxes",
  "cornell_box",
  "curves",
  "gravity_spheres_volume",
  "gravity_spheres_isosurface",
  "perlin_noise_volumes",
  "random_spheres",
  "streamlines",
  "subdivision_cube",
  "unstructured_volume",
  "planes",
  "clip_with_spheres",
  "clip_with_planes",
  "clip_gravity_spheres_volume",
  "clip_perlin_noise_volumes"
};
```

デモ コマンド入力判定と、初期化

```
int main(int argc, const char* argv[]) try {

  // コマンドライン・パーサーの設定: プログラム ospExamples.cpp 実行時のコマンドラインが入力
  auto parser = argh::parser(argc, argv);
  parser.parse(argc, argv, argh::parser::PREFER_PARAM_FOR_UNREG_OPTION);

  // コマンドライン入力から szRenderer 変数を抜き出して設定
  std::string szRenderer;
  parser({"-szRenderer","--szRenderer"}, "scivis") >> szRenderer;

  // コマンドライン入力から nScene 変数を抜き出して設定
  int nScene;
  parser({ "-nScene", "--nScene"}, 0) >> nScene;

  // インテル(R) OSPRay レンダラー/デバイスの初期化
  utils::initializeOSPRay(argc, argv);

  // レンダリングする画像サイズ
  vec2i imgSize;
  imgSize.x = 1024; // width
  imgSize.y =  768; // height
  // インテル(R) OSPRay オブジェクト
  cpp::Renderer renderer;
  cpp::Camera   camera{"perspective"};
  cpp::World    world;
  // アークボール・カメラのインスタンス
  std::unique_ptr<ArcballCamera> arcballCamera;

  vec4f backgroundColor { 0.25f, 0.25f, 0.25f, 1.f };
```

インテル® OSPRay の初期化と使用可能なデバイスの設定 & エラー処理

ユーザー定義のアークボール・カメラ

デモ ワールドの設定

```
// コミットするインテル(R) OSPRay オブジェクトの実行リスト
rkcommon::containers::TransactionalBuffer<OSPObject> objectsToCommit;

auto builder = testing::newBuilder(g_scenes[nScene]);
testing::setParam(builder, "rendererType", szRenderer);
testing::commit(builder);

world = testing::buildWorld(builder);
testing::release(builder);

world.commit();
```

シーンを選択

```
static const std::vector<std::string> g_scenes = {
  "boxes",
  "cornell_box",
  "curves",
  "gravity_spheres_volume",
  "gravity_spheres_isosurface",
  "perlin_noise_volumes",
  "random_spheres",
  "streamlines",
  "subdivision_cube",
  "unstructured_volume",
  "planes",
  "clip_with_spheres",
  "clip_with_planes",
  "clip_gravity_spheres_volume",
  "clip_perlin_noise_volumes"
};
```

- シーンの名前から対応する testing::Builder 構造体のインスタンスを作成
- Buider は cpp ラッパーの形式で提供され、インテル® OSPRay シーン階層とインテル® OSPRay API の使用ガイドに最適
- cpp::Geometry、cpp::Volume、cpp::Light のようなシーンのオブジェクトをインスタンス化して管理

Buider のソースプログラムは、インテル® OSPRay の master よりダウンロード可能
→apps/common/ospray_testing/builders/ (英語)

デモ レンダラーの設定とカメラの作成

```
// レンダー・オブジェクト (scivis または pathtracer) の作成
renderer = cpp::Renderer(szRenderer.c_str());

// レンダラーの変更時にセットした背景色を保持
renderer.setParam("backgroundColor", backgroundColor);
if(szRenderer == "sciviz") {
   renderer.setParam("aoSamples", 16);
   renderer.setParam("aoRadius", 100.f);
   renderer.setParam("aoIntensity", 1.f);
   renderer.setParam("volumeSamplingRate", 1.f);
} else if(szRenderer == "pathtracer") {
   renderer.setParam("lightSamples",1);
   renderer.setParam("geometryLights",true);
   renderer.setParam("roulettePathLength",5);
   renderer.setParam("maxContribution", 10.f);
}
renderer.commit();


// アークボール・カメラ・モデルを作成
arcballCamera.reset(new ArcballCamera(world.getBounds<box3f>(), imgSize));
arcballCamera->updateWindowSize(imgSize);
arcballCamera->rotate(vec2f(0.f), vec2f(0.25f));

camera.setParam("aspect",   imgSize.x / float(imgSize.y));
camera.setParam("position",  arcballCamera->eyePos());
camera.setParam("direction", arcballCamera->lookDir());
camera.setParam("up",       arcballCamera->upDir());
camera.commit();
```

ユーザー定義の
アークボール・カメラ

デモ フレームバッファー作成とレンダリング

```
  // フレームバッファーの作成と設定。フレームバッファーはレンダリングされた画像の x 方向、y 方向のサイズとフォーマット、色とチャネルを持った 2D イメージデータを保持
  // 使用可能な画像フォーマット:
    // OSP_FB_NONE:   フレームバッファーはアプリケーションによってマッピングされない
    // OSP_FB_RGBA8:  ビット [0–255] 線形色空間、赤、緑、青、アルファ
    // OSP_FB_SRGBA:  ビット sRGB ガンマ補正色空間と線形アルファ
    // OSP_FB_RGBA32F: 32 ビット浮動小数点コンポーネント 赤、緑、青、アルファ
  ospray::cpp::FrameBuffer framebuffer(
    imgSize.x, imgSize.y, OSP_FB_SRGBA, OSP_FB_COLOR | OSP_FB_ACCUM);
  framebuffer.clear();

  // 1 フレームをレンダリング
  framebuffer.renderFrame(renderer, camera, world);

  // フレームバッファーにアクセスして、内容を PNG ファイルに書き込む
  uint32_t *fb = (uint32_t *)framebuffer.map(false ? OSP_FB_ALBEDO : OSP_FB_COLOR);

  utils::writePNG("boxes.png", glm::ivec2(1024, 768), fb);

  framebuffer.unmap(fb);

  return 0;
} catch( std::exception &ex ) {
  std::cout << ex.what() << std::endl;
}
```

フォーマット

チャネル

アルファ値を含むRGB カラー

プログレッシブ・リファインメント用のアキュムレーション・バッファー

フレームバッファーの OSP_FB_COLOR チャネルにアクセス

最初にヒットしたときのレンダリングして格納されたマテリアルの反射率 (光源なしの色)

インテル® oneAPI レンダリング・ツールキット

サンプルプログラムの入手方法

- インテル社のコードサンプルから入手
  https://www.intel.com/content/www/us/en/developer/tools/oneapi/rendering-toolkit.html
  (英語)

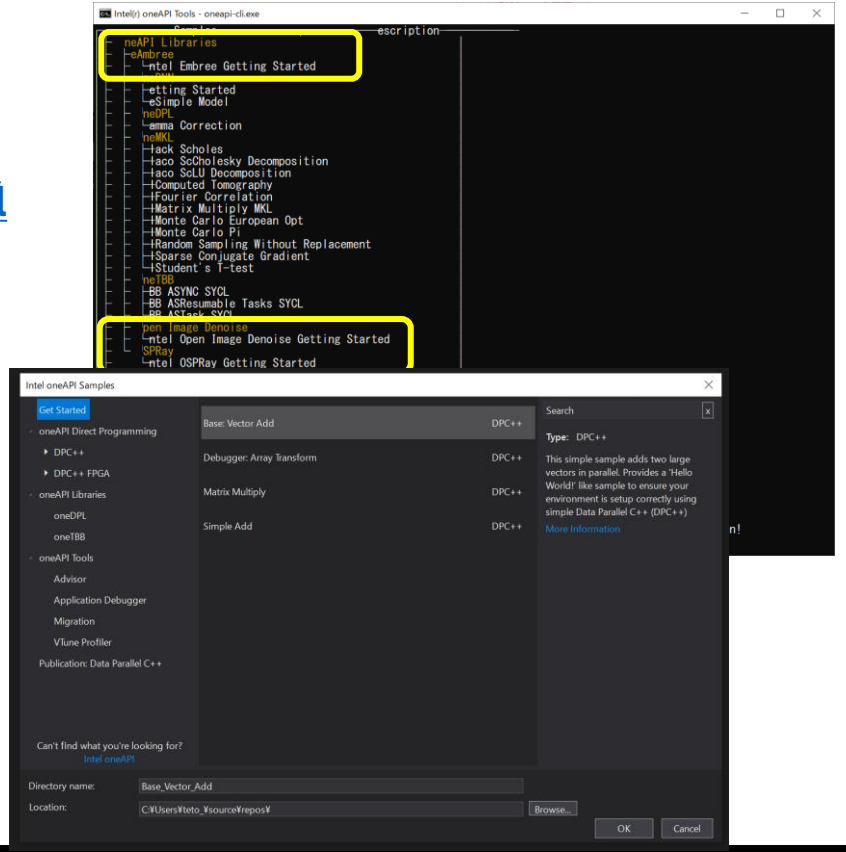
- インテル® oneAPI CLI サンプルブラウザーの使用

  $ oneapi-cli の実行

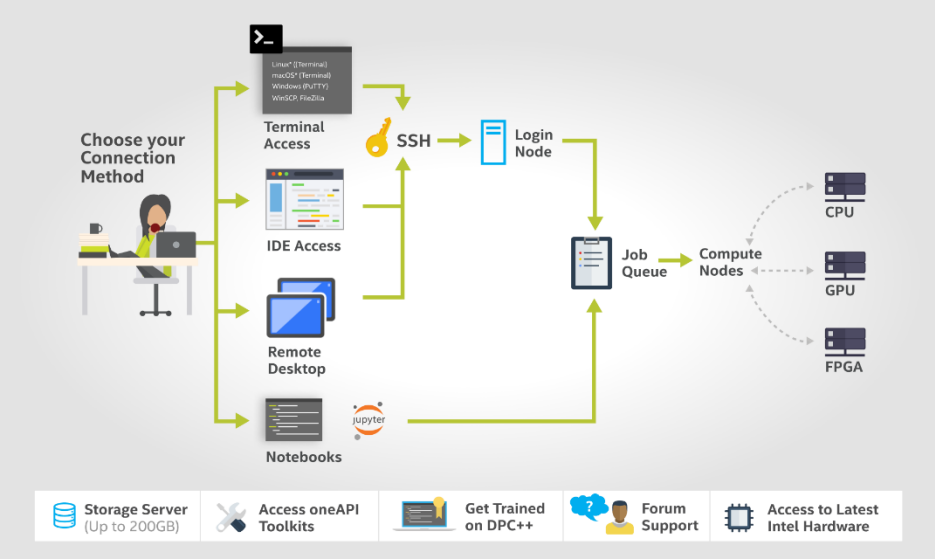
  ($ /opt/intel/oneapi/setvars.sh の実行後)

- インテル® oneAPI Visual Studio* コード拡張機能を使用

  [拡張機能] > [Intel] > [Browse Intel oneAPI Samples]



インテル® DevCloud for oneAPI によるビジュアル開発



- アプリケーションの実行、ビジュアライゼーション・パフォーマンスの最適化にインテル® oneAPI レンダリング・ツールキット
- リモート・デスクトップ機能でレンダリングを可視化
- 最新のインテル製ハードウェアでワークロードの評価が可能
- ガイド付きサンプル・アプリケーションの詳細 https://devcloud.intel.com/oneapi/home/ (英語)
- サインアップ software.intel.com/DevCloud/oneAPI/sign-up (英語)

フリーアクセス、インストール不要、セットアップ & 設定不要

さまざまなインテル® アーキテクチャー & インテル® oneAPI ツールのテストドライブ

インテル® oneAPI ツールキット

サポート体制

- **インテル社エンジニアのサポート、日本語サポート、日本語の最新技術資料が欲しい**
  - インテル® oneAPI ツールキットのライセンスを購入
  - 優先サポートユーザーへ
- **さまざまな最新のプラットフォーム上で動作確認したい**
  - インテル® DevCloud for API にサインアップ (英語)
  - インテル® oneAPI ツーキットなどの開発環境はすでに準備
  - リモート・デスクトップで動作確認
- **ターゲット・プラットフォームのバージョンがない、オープンソースでテストしたい**
  - オープンソースの各サイトからダウンロード可能
  - バイナリー版でもプラットフォームでの検証はない
  - 質問があればオープンソースのコミュニティー内で解決

お得情報

インテル® oneAPI 関連のドキュメント

(日本語)

HPSC サイトで期間限定公開中の日本語ドキュメント (https://hpc-event.jp/contents.html)

*期間終了後は、購入者のみ

- インテル® ソフトウェア開発製品に関する並列化マガジン
  - The Parallel Universe 日本語版
- 最新の日本語版パッケージ
  - インテル® DevCloud for oneAPI
  - インテル® VTune™ プロファイラー
- インテル® VTune™ プロファイラー・パフォーマンス解析クックブック
- インテル® Advisor クックブック
- アーキテクチャー向け最適化リファレンスおよびホワイトペーパー

oneAPI 関連資料

- oneAPI 仕様
- oneAPI、SYCL*、OpenMP* 関連ガイド
- インテル® oneAPI ツールキット、同梱コンポーネント関連ガイド
- oneAPI for NVIDIA* GPU および AMD* GPU 関連ガイド
- インテル® oneAPI ツールキット製品カタログ

  ベース・ツールキット/HPC ツールキット/IoT ツールキット/レンダリング・ツールキット

https://www.isus.jp/devcloud/devcloud-for-oneapi-jp/

追加の技術情報とデモビデオ (英語)

- Tech.Decoded レンダリング・ウェビナー (https://techdecoded.intel.io/?s=rendering)
- インテル® OSPRay Studio を使用したベントレーモーターズ社のインタラクティブ・バーチャル・ショールーム (https://youtu.be/hAEAu6DVySM)
- Interactive Volumetric Path Traced Cloudscape with Intel® Open Volume Kernel Library, Intel® OSPRay, and Intel® OSPRay Studio (http://youtu.be/cpRMNuyiTig)
- SIGGRAPH 2020 インテル® DevMesh (https://bit.ly/siggraph20)
- University of Chicago Uses SciVis for a Billion Cells, COVID-19, and Invisible Monsters

お問い合わせはこちらまで

https://www.xlsoft.com/jp/qa