# CPU で Unreal* Engine のパーティクル・エフェクトを使用する

この記事は、インテル® デベロッパー・ゾーンに公開されている「Use Unreal Engine* Particle Effects on the CPU」の日本語参考訳です。

---

Unreal* Engine バージョン 4.19 は、インテル® マルチコア・プロセッサーのパフォーマンスをゲームで利用できるようにし、よりエキサイティングなエクスペリエンスを実現します。この記事は、ゲーム「Sinner: Sacrifice for Redemption」を例に Unreal* Engine の CPU パーティクル・エフェクトについて説明します。

## CPU パーティクルを使用する理由

Unreal* Engine のパーティクル・エフェクトは、GPU または CPU のいずれでも利用できます。サポートされるパーティクルの最大数は、GPU のほうが CPU よりもはるかに大きくなります。それにもかかわらず CPU パーティクルを使用するのはなぜでしょうか? GPU パーティクルは数では絶対優位にありますが、Light モジュールなどのすべての特性をサポートすることができません。また、強力なマルチコア CPU 処理能力により、GPU の負荷を軽減することができます。

## はじめに

Sinner ゲームの最初のシーンでは、ゲームの雰囲気を表現するためキャンドル装飾が追加されています。キャンドルの光は炎の状態により調整されます。キャンドルの炎エフェクトは CPU パーティクルによって実現され、ダイナミック・ライト・エフェクトには CPU パーティクルの Light モジュールが使用されています。

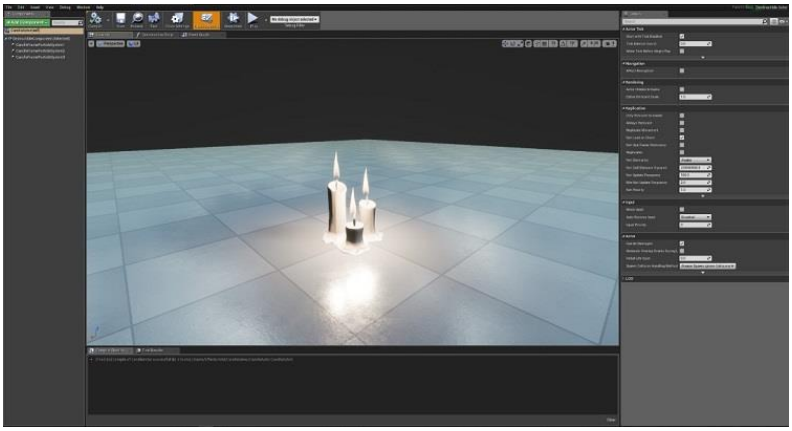

図 1. キャンドルのプレビュー

図 2. ゲーム画面のパフォーマンス

## キャンドルモデルの準備

キャンドルの作成は 2 つのパートに分けられます。

- キャンドルのメッシュとマテリアルを含むキャンドルモデルの作成
- キャンドルの炎の作成

キャンドルモデルの作成には、特殊な 3D モデリング・ソフトウェアが使用されます。この例では、すでにコンテンツを作成およびインポート済みです。キャンドルごとに異なるモデルを作成する代わりに、3 つの異なるキャンドルを 1 つのモデルにして作成と使用を容易にします。次の図 3 を参照してください。

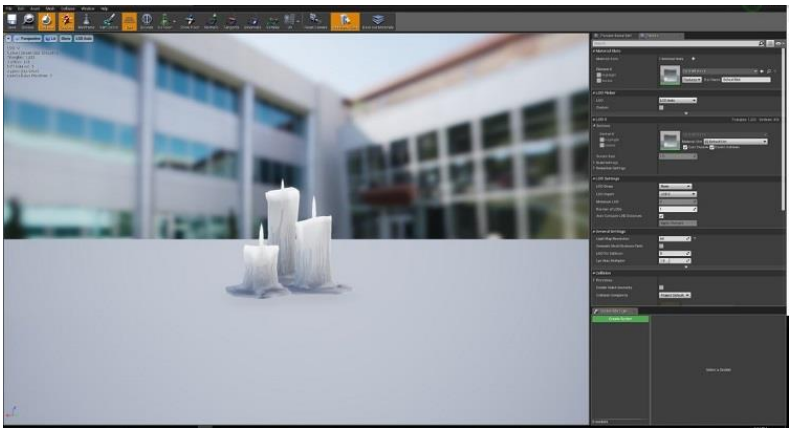
図 3. キャンドルモデルのプレビュー

## 炎のマテリアルの作成

炎のメインビジュアル表現は炎のエフェクトです。このエフェクトを作成するには、炎のマテリアルを作成して、それをパーティクル・マテリアルとして使用します。燃焼により炎が揺らめく効果は、炎のオリジナルのテクスチャーにノイズ・テクスチャーを重ねて炎のマテリアルで表現されます。

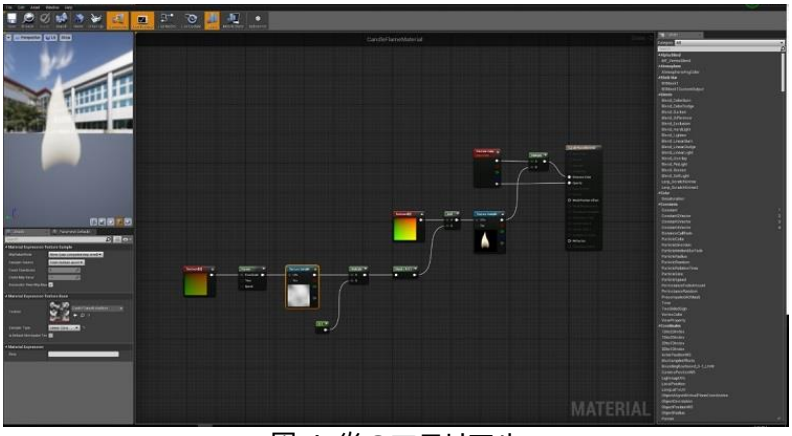

図 4. 炎のマテリアル

## フレーム･パーティクルの作成方法

必要なモジュールを選択してパーティクル･システムを構築し、トランスミッターによって使用されるマテリアルとして作成した炎のマテリアルを指定します。次のステップを実行します。

1. 画面の配置を **[PSA (Particle Screen Alignment) Rectangle]** に設定します。これにより、常にカメラに対面し、SizeX と SizeY により炎の形状 (縦横比) を制御できます。(図 5 を参照。)

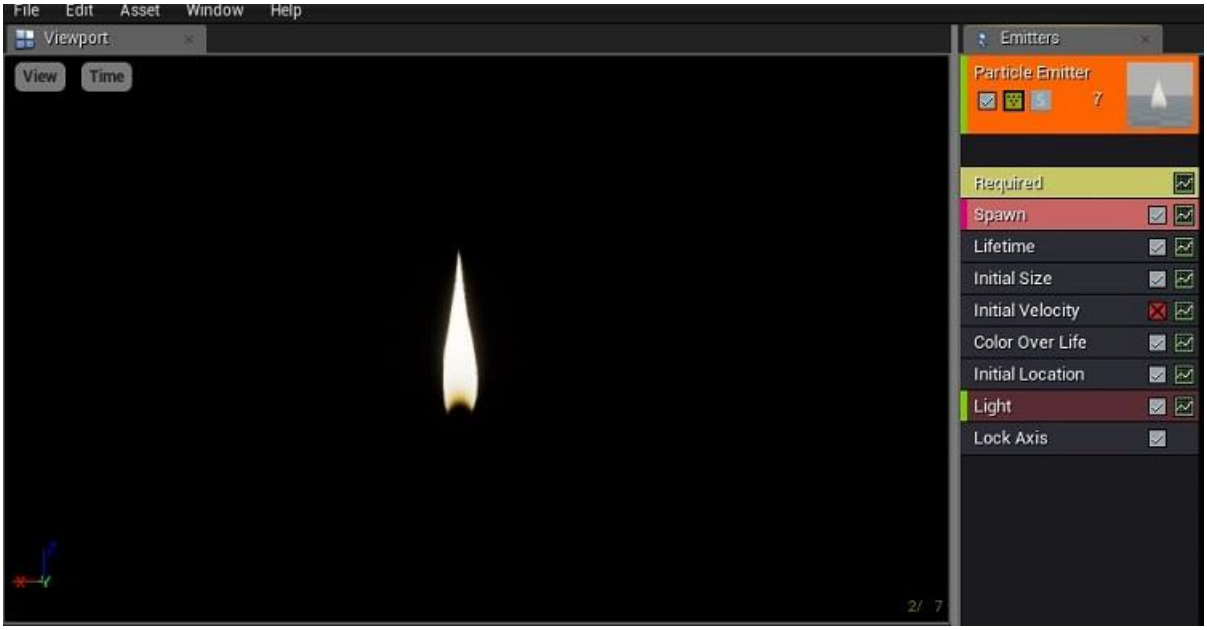

図 5. 炎のパーティクルのパラメーター

2. Initial Size モジュールを使用して炎の初期サイズを設定し、**[Distribution Vector Constant]** 分布を選択して、X と Y パラメーターのみ設定します。パーティクルは常にカメラと対面するため、Z 軸の設定は有効になりません。

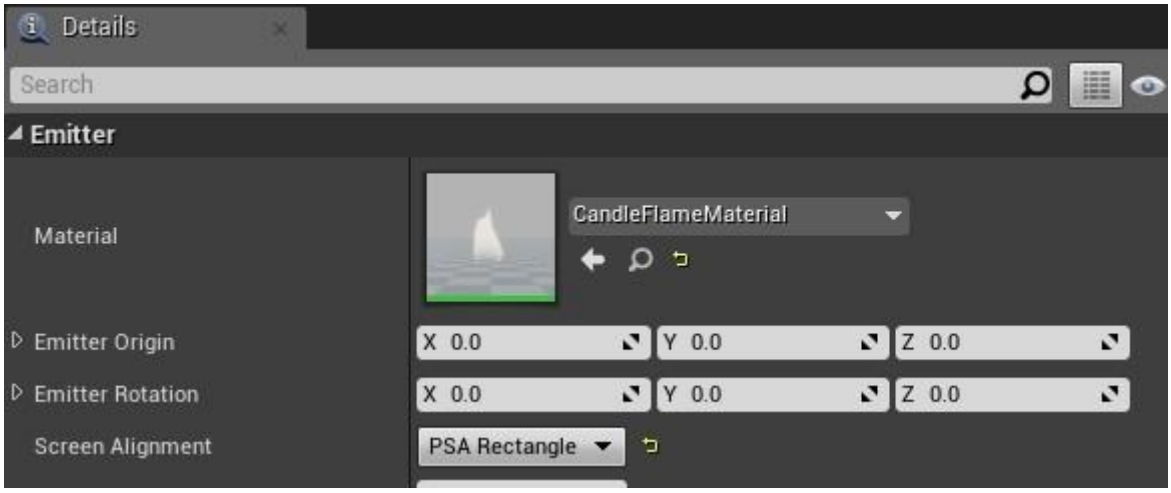


図 6. マテリアルと画面の配置の設定

3. パーティクルのスポーンモジュールの生成レートと Lifetime モジュールのパーティクルの存続期間を設定して、可能な限り実際の炎のエフェクトをシミュレーションします。

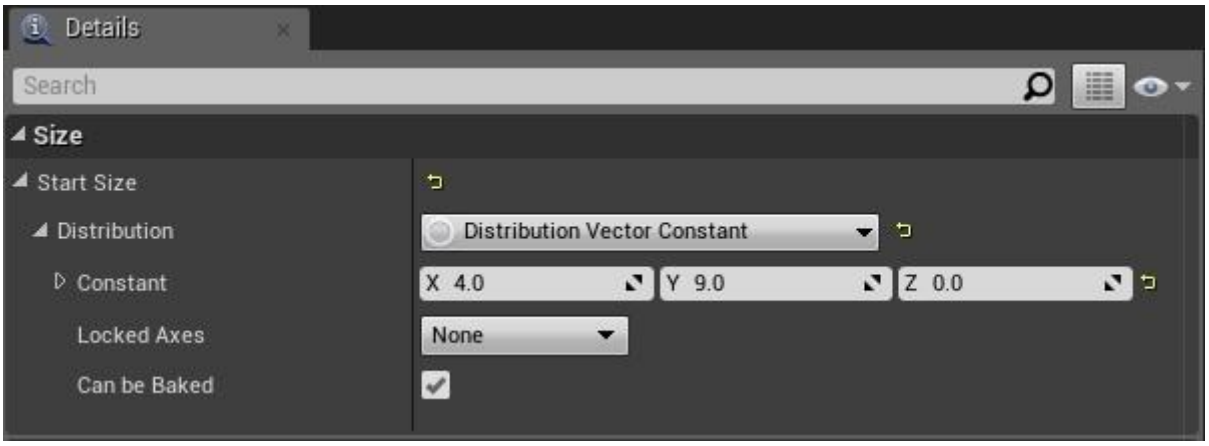


図 7. パーティクルの初期サイズの設定

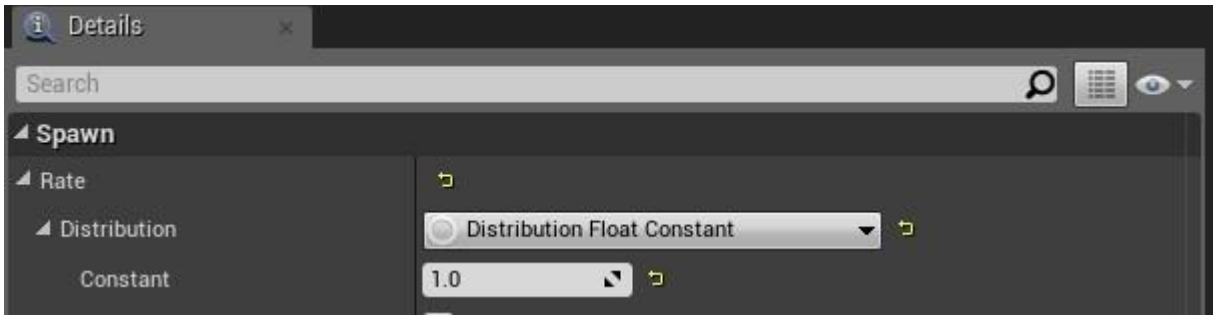


図 8. パーティクルの生成レートの設定

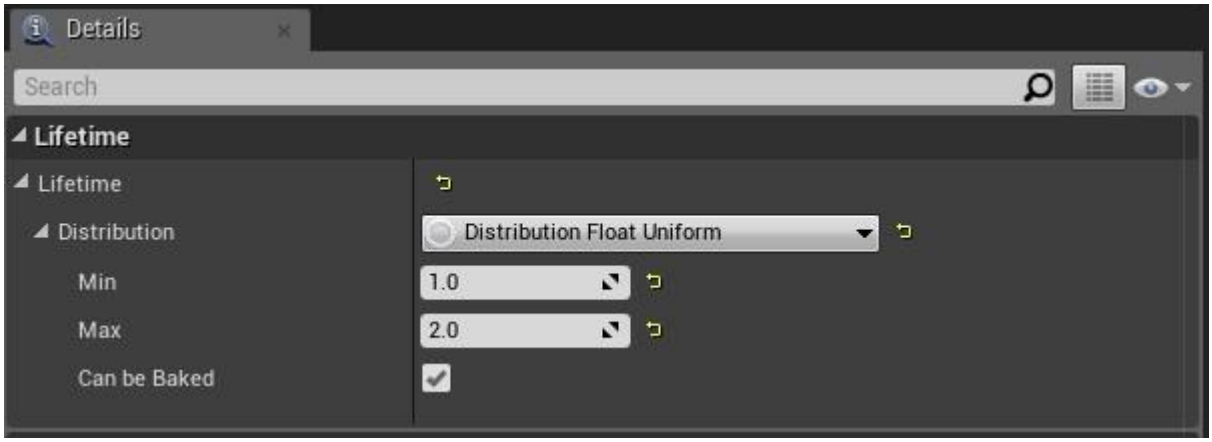

図 9. パーティクルの存続期間の設定

## パーティクル・ライトの追加

パーティクルの基本エフェクトを作成した後、トランスミッターに Light モジュールを追加して、炎のライフサイクルに伴う照明効果を実現します。[Brightness Over Life] でパーティクルのライフサイクルに応じた光の輝度、[Radius Scale] で光の半径、[Light Exponent] で光度をそれぞれ設定します。

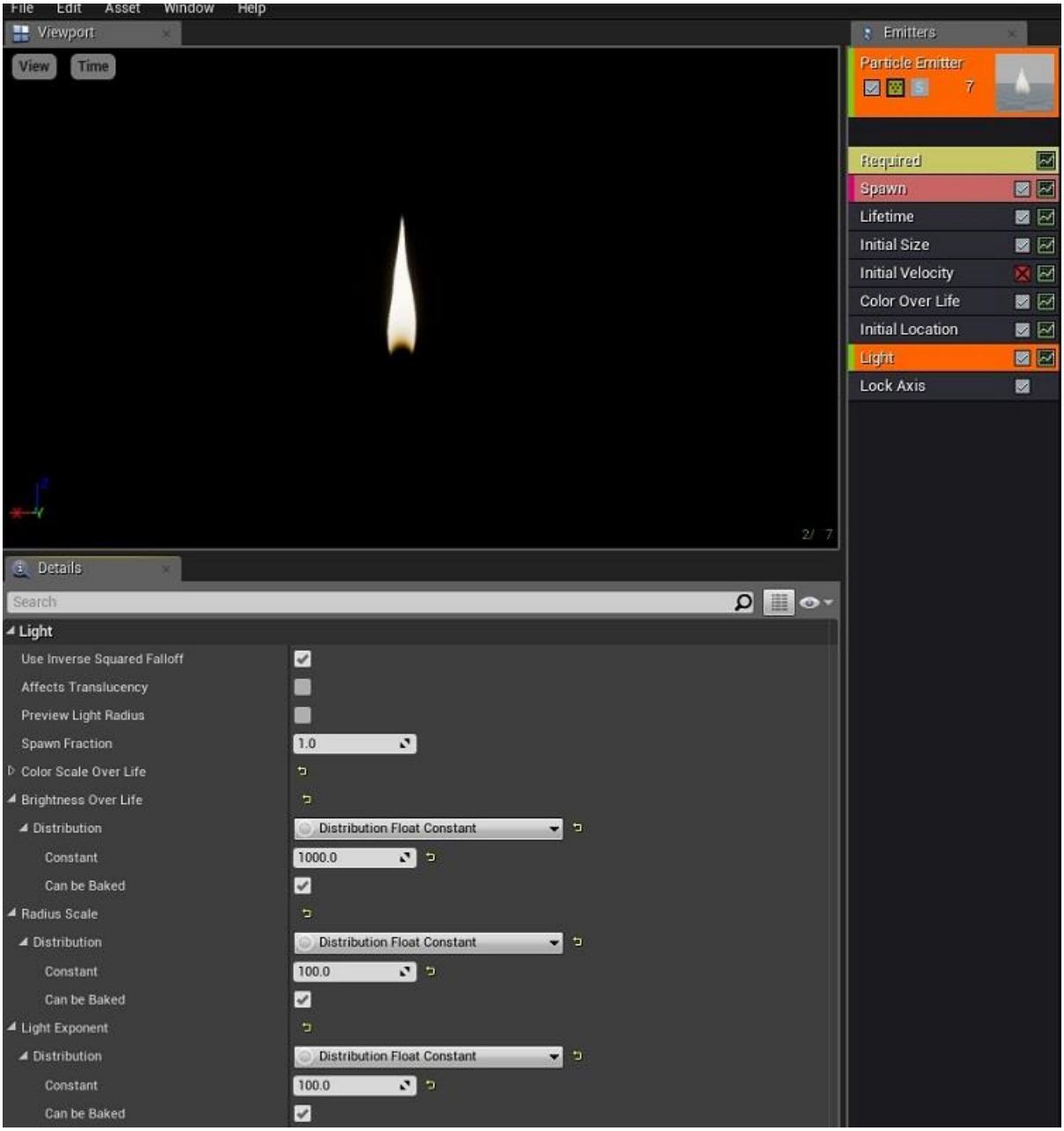


図 10. Light モジュールの追加

さらに、Color Over Life モジュールでパーティクルとパーティクル・ライトの色を設定します。

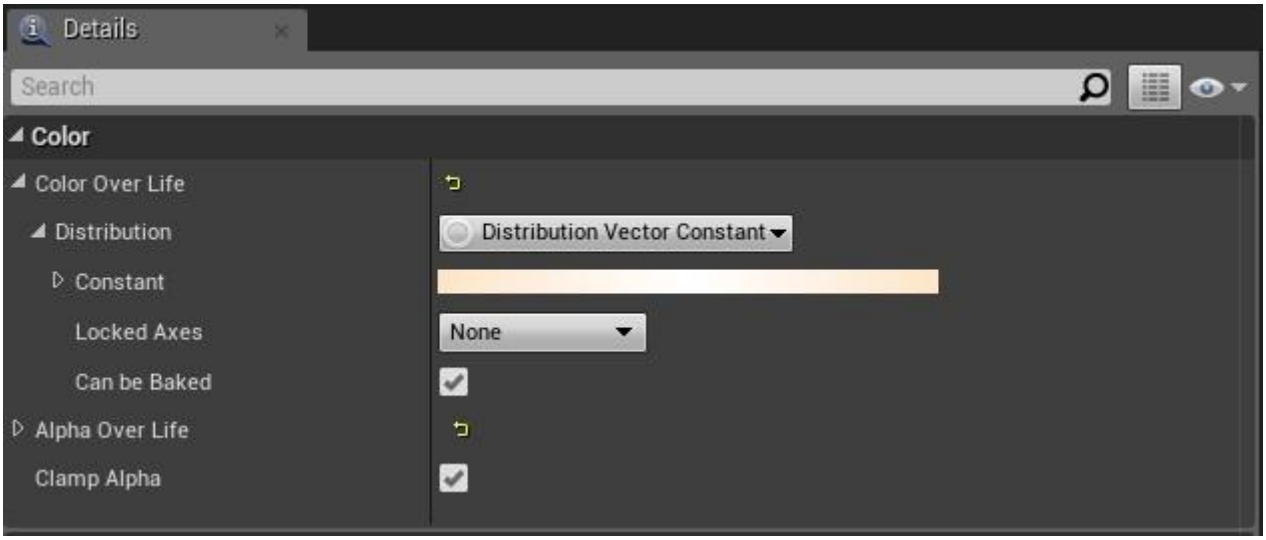

図 11. パーティクルと光の色の設定

## 統合プレビュー

炎とキャンドルのモジュールコンテンツの準備ができたら、ゲームシーンに簡単に追加できるようにそれらを結合する必要があります。これは、Actor を親クラスとして Blueprint クラスを作成することで達成できます。次に、キャンドルモデルをインポートして、作成済みの炎のパーティクル・システムを追加します。この例では、キャンドルモデルに 3 つのキャンドルがあるため、3 つの炎のパーティクル・システムを追加して、炎とモデルの相対位置を調整する必要があります。

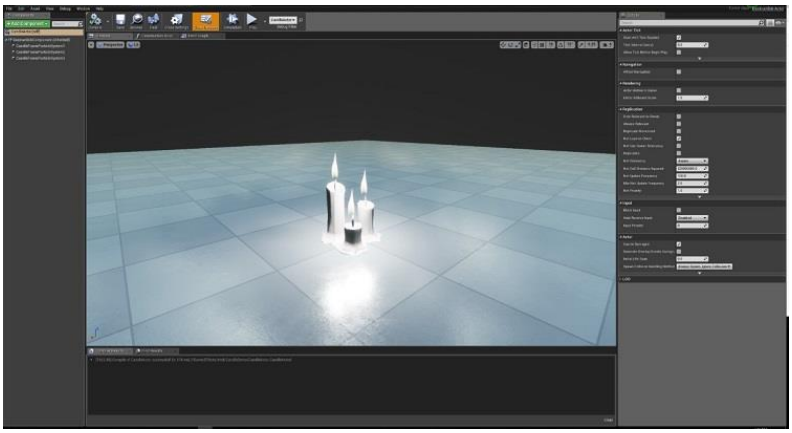
図 12. キャンドル作成のプレビュー

これで、キャンドルの装飾の主なコンテンツの作成は完了です。テストレベルにアクターを配置して、ゲームでそのエフェクトを確認します。

図 13. キャンドルレベルのプレビュー

## ゲームビデオ

より詳細なビューについては、次のビデオで最終的なゲームの結果を比較しています。

https://www.youtube.com/embed/P3Ao3qTID_g?feature=oembed

## マテリアルのダウンロード

この記事の制作資料は、「Sinner: Sacrifice for Redemption」の開発チームである Dark Star Studio (英語) から提供され、インテルの支援を受けて完成しました。Dark Star Studio は、MIT ライセンスの下にサンプル・プロジェクトを公開しています。

サンプル・プロジェクトは次のレポジトリーから入手できます。

GitHub* のサンプル・プロジェクトを表示 (英語)

## ゲームのステップアップ

次のゲームの想像、計画、設計に役立つ強力な開発ツールとチュートリアルにアクセスできます。インテル® Game Developer Program (英語) は、ゲームを効率良くコーディングして、インテル® アーキテクチャー向けに最適化するのを支援する無料のライブラリー、パフォーマンス・アナライザー、その他の機能を提供しています。排他的なコードサンプルへのアクセスを含む利点 (英語) を調査し、ゲーム開発者が執筆した市場投入ガイドを参考にしてください。

参加する (英語)

コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。