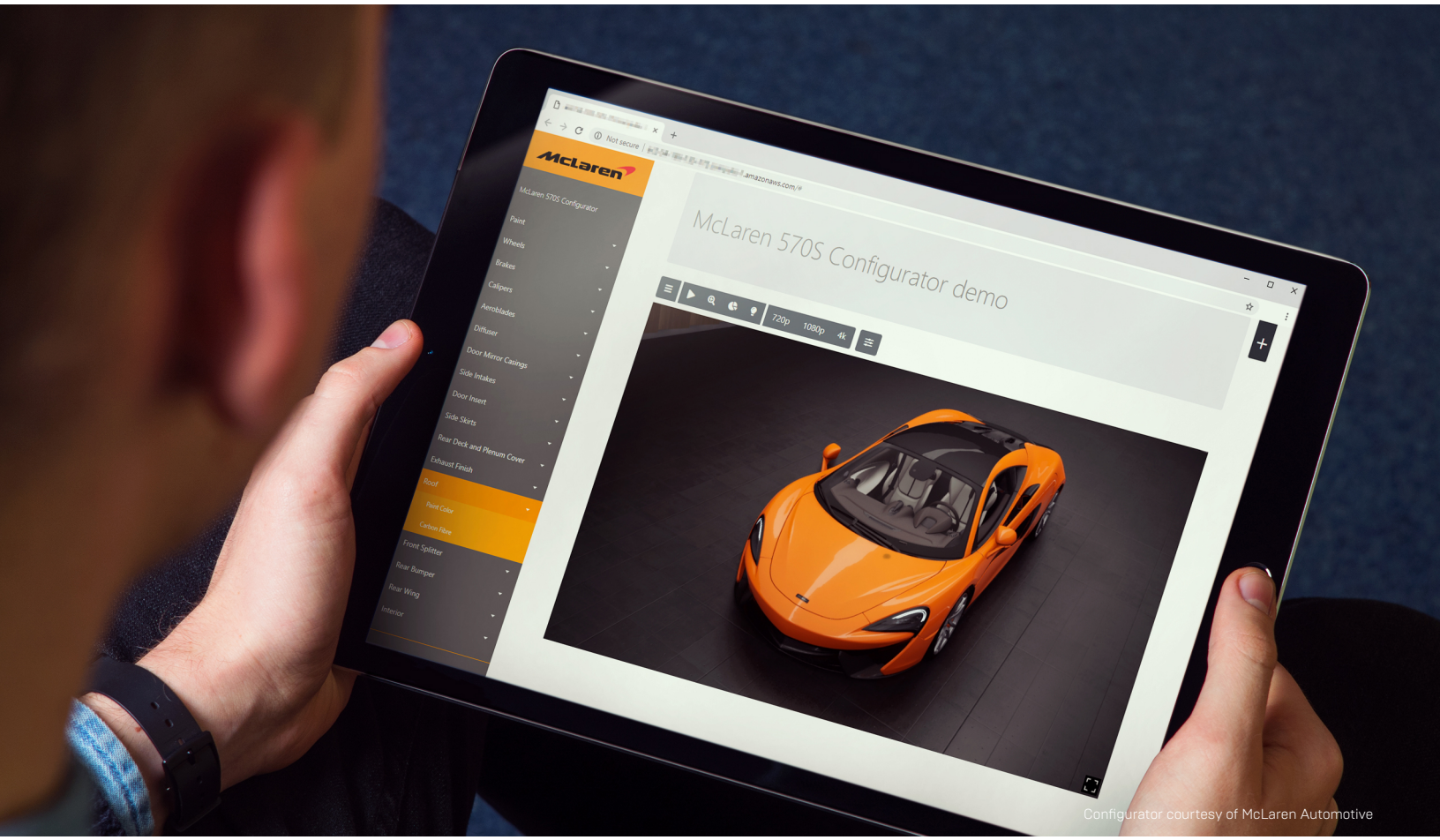
UNREAL ENGINE



Configurator courtesy of McLaren Automotive

Unreal Engine のコンテンツを複数のプラットフォームにストリーミングする

HTML5、WebGL、Pixel Streaming の比較

目次

ページ

1. はじめに 3

2. コンテンツ配信の課題 4
   - UX とパフォーマンスに関する考慮事項 4
   - 技術的な考慮事項 4

3. 配信ソリューションの比較 5
   - WebGL 5
   - HTML5 7
   - Pixel Streaming 8

4. まとめ 13

はじめに

ユーザーを繋ぐエクスペリエンスの開発におけるコンテンツの共有にまつわる課題は、コラボレーション、プロダクション、公開において常に重要な判断が求められます。そして、ユーザーがもしさまざまなデバイス (PC、タブレット、スマートフォン) で共有コンテンツを使用してやり取りする場合、核心的な問題が発生します。「様々なプラットフォームとハードウェア互換性が求められる市場へどのように公開するのか」です。

一般的なのは、最低のエントリー ポイントとターゲット オーディエンスが許容できる品質を見極めて、それを指定プラットフォームとするアプローチです。ところがこのアプローチでは、エンドユーザー全員のエクスペリエンス品質が犠牲になってしまいます。

そのうえ、こうしたデプロイ方法では、エンド ユーザーがリアルタイム コンテンツを利用するデバイスと、データとロジックを保存して結果を画面に描画するデバイスが同じです。デバイスにデータをダウンロードすると、さまざまな問題が生じます。たとえば、ダウンロードを高速化するためにファイル サイズを小さくする必要があり、これにより品質が制限されます。

このドキュメントでは、Unreal Engine ベースのエクスペリエンスを Pixel Streaming で共有するために利用できるオプションやツールを評価し、他の利用可能なオプションと比較します。このドキュメントの目的は、Unreal Engine のゲーム以外のアプリケーションで、品質と機能のエクスペリエンスを最も高い状態に保つことです。

Image courtesy of Zaha Hadid Architects, Line Creative, and Epic Games

コンテンツ配信の課題

>もはや情報化時代を超え、消費者が高品質で魅力的な没入感のあるコンテンツに最も興味を示すエクスペリエンス時代が到来しました。こうしたコンテンツの増加にはインタラクティブなエクスペリエンスを含み、これは 3D モデルはもちろん AR や XR オーバーレイを利用します。そして、大多数の消費者に認識されるには、製作者は、携帯電話やタブレットはもちろん、PC やインタラクティブ ディスプレイまで、さまざまなプラットフォームに向けてこうしたコンテンツを共有する必要があります。

WebGL や HTML5 をベースにした旧来のソリューションでは、表示するコンテンツの忠実度とインタラクティブ性は、消費者のデバイス、特にそのハードウェア、表示メカニズム、オペレーティング システムに依存します。つまり、最も多くのユーザーがコンテンツを利用できるようにするには、デバイスやプラットフォームを共有するアプリケーションを最も要件が厳しいデバイスに要件を定義するか、複数のバージョンのアプリケーションを作成して異なるユーザー グループに対応する必要があります。

ストリーミングを利用するコンテンツを作成する際の課題は、エンド ユーザーのデバイスに関係なく、ブランドの外観と操作感とともにインタラクティブな機能を維持しながら、このコンテンツを複数の通信チャンネルに配信することです。これは、共有アプリケーション自体と、その実行に必要なハイエンド ハードウェアを、ユーザーの表示デバイスから分離することで実現できます。

UX とパフォーマンスに関する考慮事項

複数のユーザー デバイスを対象にした共有アプリケーションを開発するには、ユーザー エクスペリエンスと、それを実現するテクノロジーの両方において、考慮べき要素がいくつかあります。

インタラクション - デプロイを成功させるには、ブラウザ上でユーザーが入力してからホストが反応するまでのインタラクションを許容できる時間内に収め、エクスペリエンスを楽しめるようにする必要があります。これには初期起動時の待機時間が含まれます。一般的に、ゲーム以外のコンテンツの利用者は、テンポが速いゲームをプレイする消費者と比べて、辛抱強さがあります。たとえば自動車コンフィギュレータの場合、ユーザーは反応の確認に 1〜2 秒、さらに最終品質の更新を表示するまで 10 秒間は待ってくれることを期待できます。

再生スピード - リアルタイム パフォーマンスを実現するには、コンテンツ自体の最適化が必要になる場合があります。最初に、対象のアプリケーションとそのユーザーにとって、どの程度の再生速度を許容できるか定義する必要があります。人間が没入感を得るのは 60 fps が標準とされていますが、5 fps だけ必要なアプリケーションもあれば、90 fps を求められるアプリケーションもあります。

画質 - 品質レベルと、ユーザーに表示して体験させる対象の忠実度は、(リアルタイム再生の考慮事項を含む) コンテンツ設定、エンコード品質、搭載する GPU によって、アプリケーション側で決定します。品質の期待値とコストを考慮して、ビジネス事例、そして設定の最終出力を定義できます。

技術的な考慮事項

ホスティングとデータ - 技術的には、分散エクスペリエンスはクライアント側またはアプリケーション側で実行できます。しかし、クライアント プラットフォームの品質制限、そしてクライアントへのデータ転送に伴う時間、ストレージ、セキュリティの問題を回避するために、アプリケーション側からストリーミングするのが理想的です。ユーザー数が限定されたシンプルなアプリケーションでは、手頃なワークステーション 1 台からのストリーミングで十分な場合があります。ユーザーが多数いる場合は、クラウドを利用することで高い可用性とスケーラビリティが得られ、それぞれが独自のバージョンのアプリケーションを完全に制御できます。ユーザー数の少ない複雑なアプリケーションもクラウドでのホスティングに適しています。ただしクラウドを使用する場合は、ストレージにかかるコストとコンテンツのデータ サイズを考慮する必要があります。

ユーザー負荷 - ユーザー数とアプリケーションの複雑さによって、単一インスタンスによるホストで配信できるものと、複数のユーザーに同時に対応するのに必要な負荷分散を定義します。忠実度が高く完全にインタラクティブな自動車コンフィギュレータでは、同じ設定の 10,000 のインスタンスが必要な場合があります。最初のインタラクション時の求められる速さによっては、サーバーの数が重要となります。

ストリーミングのサイズ - モバイル デバイスといったような対象となるユーザーのグループが特定のプラットフォームを使用している場合は、そのプラットフォームの帯域幅制限に準拠するように、異なるビデオ ストリーム サイズの指定が必要となることがあります。

更新 - 更新とバグの修正がどのようにデプロイされるかです。アプリケーション全体を毎回置き換えるのか、またはライブ アップデートを実行する手段を持たせるのかを設定します。こうした更新や修正をより多くのインスタンスに適用する方法を考慮する必要があります。

セキュリティ - ホストとクライアント間のアクセスとセキュリティが、そもそもユースケースを定義する場合があります。デザイン レビューを共有して議論するためにシステムが機密データを要求する場合、または特定の期間内に新しい情報をデプロイする必要があるが期日までアクセスさせない場合は、ネットワークやアクセス サイトにセキュリティ対策を実装する必要があります。

指標 - ユーザーの行動とシーンの統計が KPI である場合は、追跡して分析ソフトウェアにストアする必要があるイベントを、定義、有効化、接続する必要があります。

配信ソリューションの比較

Epic Games が Unreal Engine (UE4) コンテンツの配信ソリューションの提供に着手したとき、開発チームは既存のテクノロジをソリューションのバックボーンに活用することを検討しました。2 つの候補、WebGL と HTML5、を検討した後、Epic は新しい独自のソリューションとして Pixel Streaming プラグインの開発を決定しました。

ここでは、特に前述のニーズを満たすことに関して、コンテンツ配信向けの WebGL と HTML5 に関するチームの調査についてと、その後 Pixel Streaming テクノロジーを開発した理由について説明します。

WebGL

WebGL はインタラクティブな 2D および 3D グラフィックをレンダリングできる JavaScript API であり、ユーザーがアプリケーションやプラグインをダウンロードすることなく結果を Web ブラウザに表示します。WebGL は、ロイヤリティフリーのオープン スタンダードを創り出す非営利組織、「クロノス・グループ」によって設計および管理されています。

WebGL は、スマートフォン、タブレット、ビデオゲーム コンソール、PDA など、組み込みシステム用に設計された「OpenGL ES」(OpenGL for Embedded Systems) をベースにしています。WebGL は、Web ページにグラフィックを描画するために使用する HTML5 Canvas 要素を介して機能します。

WebGL は GPU アクセラレーションを使用した物理や画像の処理、エフェクトをサポートします。

WebGL のデプロイは 2 つの部分で構成されます:

- JavaScript で記述された制御コード
- OpenGL ES Shading Language (OpenGL ES SL) で書かれたシェーダ コード

定義により WebGL はデプロイ前にコンパイルが不要です。

WebGL の作成

WebGL のコンテンツを最も効率的にビルドするには、最終結果を表示するオーサリング ソフトウェアを使用します。これは通常、ブラウザを介してオンラインでテストするツールの使用を意味します。

PlayCanvas - PlayCanvas エディタは高度な WebGL オーサリング環境であり、基本的に WebGL のゲーム エンジンです。2D や 3D のグラフィック シミュレーションのプログラミングには JavaScript を使用します。コードはすべて標準に準拠したクロスプラットフォーム HTML5 で記述されており、主要なブラウザとデバイスにすべて対応しています。

Sketchfab - Sketchfab は 3D コンテンツを共有および配信するためのオンライン マーケットプレイスです。3D ビューアは、VR と AR をサポートするブラウザでプラグインを使用せずにデプロイが可能です。すべてのコンテンツを Sketchfab サーバーでホストして、そこから配信する必要があります。

WebGL の制限事項

広範な配信ツールと見なす場合、WebGL の主な制限はコンテンツの作成と配信に関するものです。コンテンツは OpenGL ES SL シェーダ言語で機能するように特別に作成する必要があります。これは現状では、アプリケーションのインタラクティブな Web デプロイでのみ使用できます。Unreal Engine のようなリアルタイム エンジンで作成したコンテンツをデプロイするには、WebGL フレームワークで動作するようにコンテンツとインタラクティブ要素を設定する必要があります。

さらに WebGL は、クライアント側の受信ブラウザとハードウェアの機能に完全に依存します。グラフィックの品質はブラウザの表示機能に依存するのです。データはクライアント側にダウンロードする必要があり、このダウンロードにかかる時間がエクスペリエンスを開始するまでの待機時間を決めます。大規模なデータセットで複雑なエクスペリエンスを実現するためには、クライアント側にそのデータを格納できる十分な容量が必要です。

機密データを扱う場合、クライアント側でデータを保護する追加の手順が必要になります。

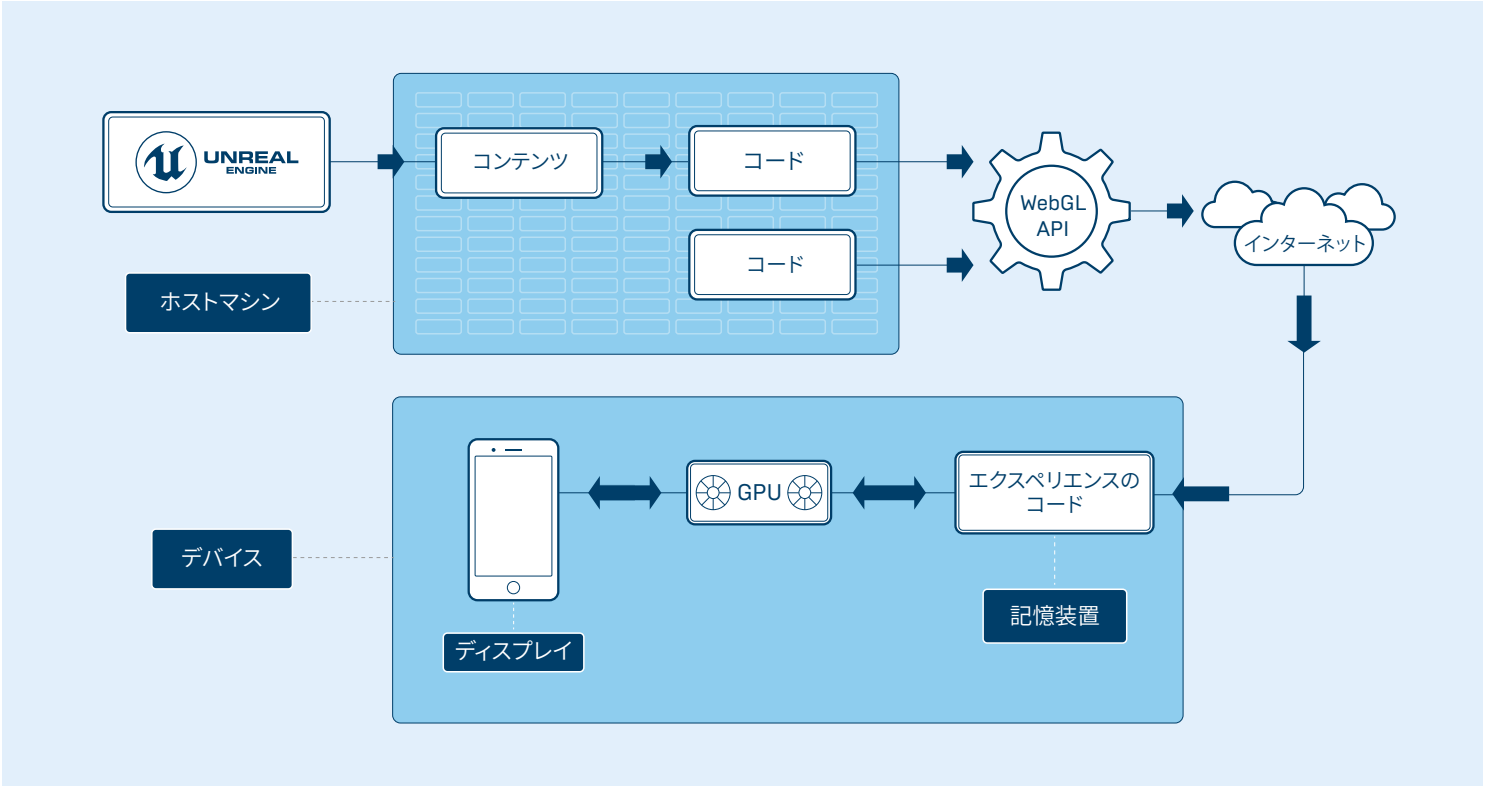


図 1: デプロイのデータフロー

WebGL コンテンツが他で制作したコンテンツと混在する環境では、WebGL の品質がエクスペリエンスの決定的な基準になります。プラットフォームのより広い観点では、異なるエンド チャンネルで操作する場合、システムは少なくとも 2 つのデータセット、シェーダ モデル、クリエイティブの設定を維持する必要があります。

指標に関しては、ログインとセッション時間のデータを通常の Web サイトのチャンネルを介して収集できます。

こうした制限はありますが、WebGL は非常にシンプルなデプロイ手段で、データセットが小さく、セキュリティ懸念が低く、基本的な品質要件が求められるエクスペリエンスに適しています。こうした種類のエクスペリエンスをサポートするために、Epic は UE4 から WebGL に直接エクスポートする Unreal Engine のオプションに取り組んでおり、これには glTF 形式のファイルや、交換シェーダー ライブラリのサポートを含みます。このエクスポーターは 2020 年第 1 四半期に公開される予定です。

Unreal Engine の需要が高いエクスペリエンスについて、Epic はより適切なソリューションを目指し研究開発を続けました。

HTML5

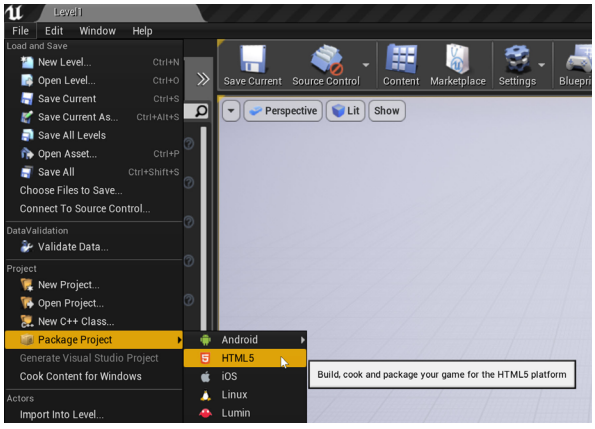
HTML5 は、Web ページ開発用マークアップ言語である HTML の最新バージョンです。HTML5 自体にさまざまなソリューションがあり、WebGL をまったく使用せずに Canvas 要素で 3D エクスペリエンスをビルドします。こうしたソリューションは、すべて GPU プログラミングの知識とエンド クライアントがダウンロードする広範なデータのパッケージ化を必要とすることから、その結果、エクスペリエンスの品質と機能が制限されてしまいます。

HTML5 の制限事項

この設定では、クライアント側のハードウェアでアプリケーションを実行するため、エクスペリエンスを開始する前にデータセット全体をクライアントがダウンロードする必要があります。このようなデプロイではダウンロード速度と必要な容量の確保が重要です。クライアントは豊かなエクスペリエンスを得るために数 GB のデータをダウンロードしなければならない場合がありますが、これは特にモバイル デバイスにおいて、ダウンロード待機時間とストレージ容量の観点から現実的ではありません。

WebGL と同様に、この方法ではシステムが 2 つのデータ標準を維持する必要があり、元のコンテンツと配信したコンテンツの間で一貫したユーザー エクスペリエンスを保証できません。さらに、配信済みのコンテンツを更新および維持するには完全に置き換えるしかありません。そのため、これは頻繁に変更するコンテンツで問題となる可能性があります。HTML5 のセキュリティに関する懸念は WebGL の懸念と同じです。

Unreal Engine はプロジェクトを HTML5 プラットフォームに公開するツールを含みます。ただし UE 4.24 の時点で Epic は、HTML5 サポートをコミュニティがサポートするプラットフォームの拡張として GitHub に移行しました。現在は公式のサポート対象ではありません。



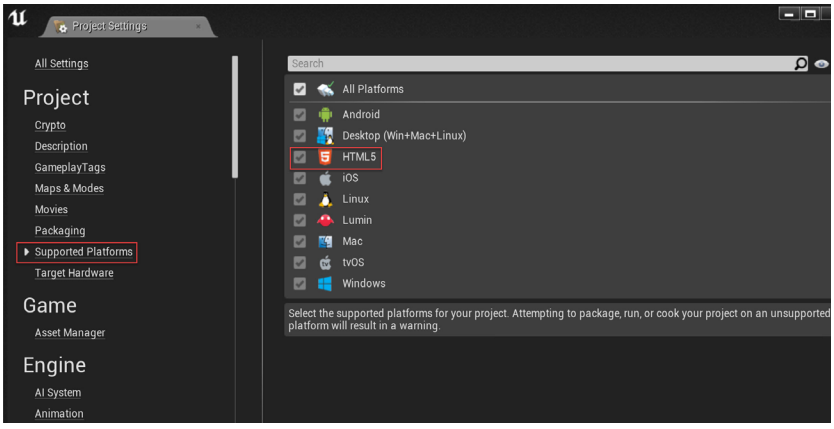


図 2：Unreal Engine インターフェースで HTML5 を選択する

Pixel Streaming

Epic Games はリアルタイム コンテンツを複数の種類のデバイスに配信する最適な方法を模索し、WebGL と HTML5 の両方から提供されているツールと機能を検討しました。しかし、これらの技術には制限があったため、目的を達成するために Pixel Streaming と呼ばれる Epic 独自の技術を開発することにしました。

Epic は、リアルタイムでインタラクティブなコンテンツを複数の種類のデバイスに配信する理想的なソリューションには、次の機能が必要だと判断しました。

- クライアント側でデータをダウンロードする必要がなく、ホストからピクセルをストリーミングする機能
- ハイエンドとローエンドのユースケースに両方対応する単一のデータ設定
- プラットフォームに依存しないデプロイ
- すべてプラットフォームとデバイスで決定的な品質
- 高い忠実度と品質
- Unreal Engine の機能を完全に共有する機能
- 明確でシンプルなメンテナンスと更新
- エクスペリエンス開始時の素早いアクセス
- デプロイ前、デプロイ中、デプロイ後にデータと設定を保護
- さまざまなユースケースで有効なさまざまな構成
- 単一のサーバーからクラウドにまでデプロイできるスケーラビリティ
- ユーザーとセッションの指標を取得する機能

Pixel Streaming は Unreal Engine のプラグインとして実装されました。このプラグインは、ホスト サーバー上のグラフィック ストリームをエンコードして、WebRTC プロトコルを介して受信するブラウザとデバイスに送信します。事実上、ハイエンド ホスト システムから実行される Unreal Engine アプリケーションが、ホスト デバイスと同じ品質でエンド ユーザーのデバイス上にある Unreal Engine の機能すべてをアクティブにします。

データはホストするサーバーに残り、ピクセルだけが表示デバイスに配信されることから、Pixel Streaming のようなストリーミングを用いた方法は、クライアントによってダウンロードさせる方法より本質的に高速で安全です。また、ユーザー セッション データの必要な指標を UE4 で取得できます。

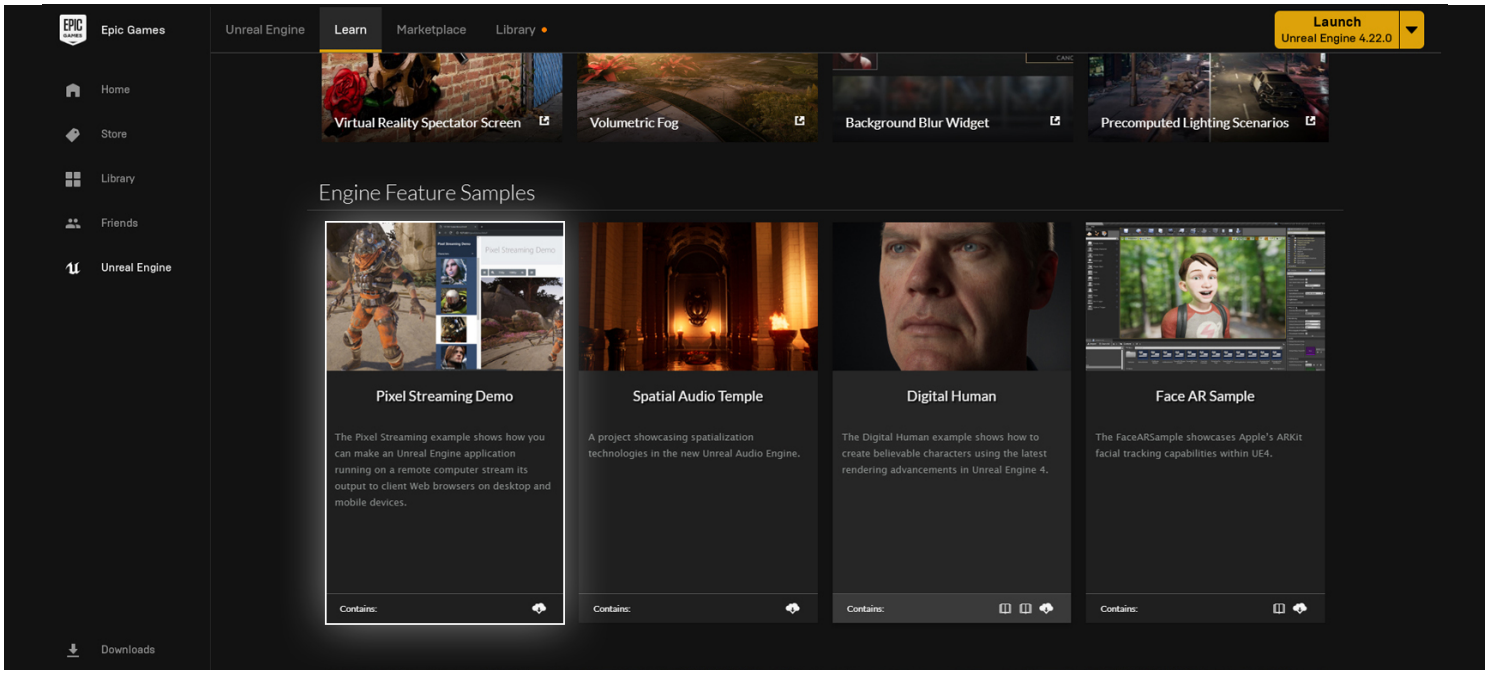


図 3 : ランチャー インターフェースの Pixel Streaming プラグイン

WebRTC プロトコル

WebRTC (Web Real-Time Communication) は、Web ブラウザとモバイル アプリケーションを介したリアルタイム通信 (RTC) プロトコルです。このプロトコルでは、オーディオとビデオは直接リンクを介して転送され、ユーザーがプラグインやアプリをダウンロードする必要がありません。通信用のコマンドは API (Application Programming Interface) を介して送信されます。

要件

Pixel Streaming は、単一のサーバーで実行することも、適切なハードウェアの動的なスケーリングとプロビジョニングを可能にする GPU クラウド環境で実行することもできます。こうした状況で重要となるのは、必要となるスケールの分析です。なぜなら、これは最終的なホスト環境のコストとユーザー エクスペリエンスの速度に直接関係するからです。

Pixel Streaming プラグインは、ホスト サーバー上の WebRTC を介して、接続したサーバーやクライアントと通信します。最も単純な形式はローカル ホストで、ローカル IP アドレスとネットワーク ポート (80、8124、8888) を介してアクセスします。

ビデオカード

NVIDIA GPU (Kepler 世代以降) は NVENC と呼ばれるハードウェア ベースのエンコーダを含み、グラフィック パフォーマンスから独立した、完全に高速化されたハードウェア ベースのビデオ エンコーディングを提供します。NVENC にオフロードされた完全なエンコードの計算が複雑なタスクで、グラフィック エンジンと CPU を他の処理に回すために解放します。NVENC で、以下のことができるようになります。

- CPU を利用しない、ゲームとアプリケーションの高品質かつ極端に低いレイテンシでのエンコードおよびストリーミング
- アーカイブ、OTT ストリーミング、Web ビデオの非常に高い品質でのエンコード
- ストリームごとの超低消費電力 (ワット/ストリーム) でのエンコード

ビデオ エンコーディング ハードウェア NVENC は、NVIDIA Video Codec SDK から利用できます。この専用アクセラレータは、Windows と Linux で一般的なビデオ コーデックのハードウェア アクセラレーション エンコーディングを多数サポートします。

詳細は 「NVIDIA Video Codec SDK のドキュメント」 を参照してください。

開発標準と、GPU およびドライバーの使いやすさの向上により、Unreal Engine 4.24 以降は Pixel Streaming の AMD オプションをサポートするようになりました。今後のリリースも、サポートされるハードウェアを引き続き増やしていく予定です。

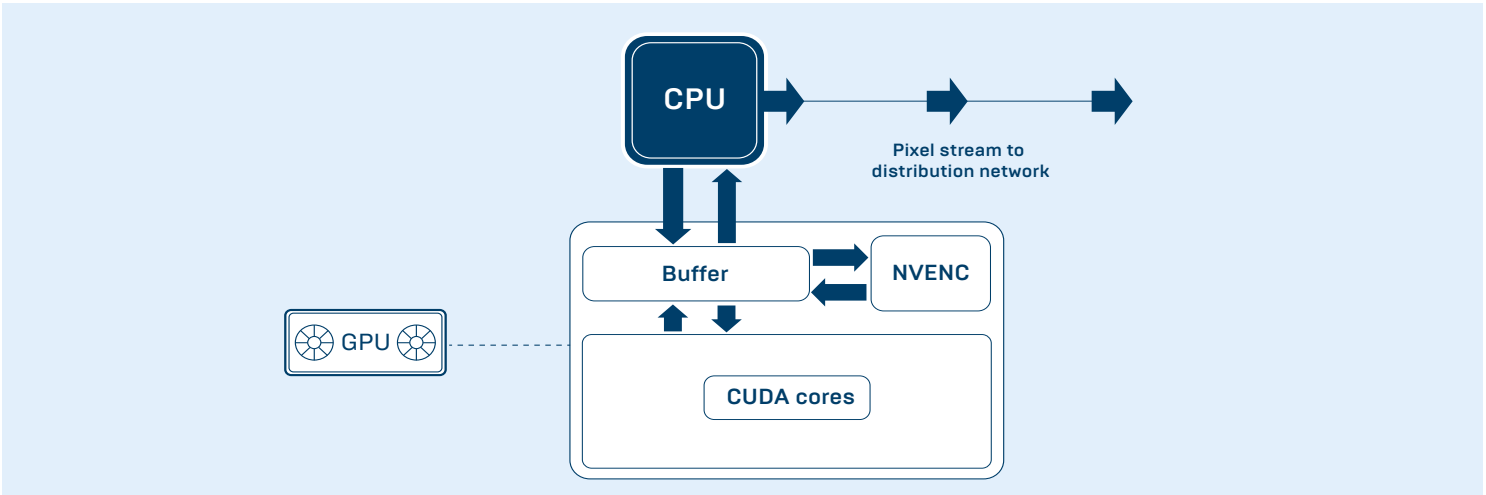


図 4 : ハードウェア ベースのエンコードおよびピクセル ストリームの生成。NVENC エンコーダはバッファを介して CPU および CUDA コアと連携し、ピクセル ストリームを生成します。このピクセル ストリームは CPU を介して配信ネットワークに押し出されます。

Pixel Streaming のユースケース設定

ここでは Pixel Streaming の一般的な設定と、それを機能させるための付随するテクノロジーを確認します。詳細については、Unreal Engine Pixel Streaming のドキュメントで「ホスティングおよびネットワーキング ガイド」トピックを参照してください。

ホストとクライアントがネットワーク ファイアウォールで分離されている場合、ホストとユーザーの通信を確立するには STUN (Session Traversal Utilities for NAT) および TURN (Traversal Using Relays around NAT) のサーバーが必要となる可能性があることに留意してください。一般的な制限により、一部のクライアント ファイアウォールはホスト サービスにアクセスを許可しません。STUN/TURN サーバーはこのような状況で、許可されたさまざまな方法を試し、ホストと要求するクライアント間に有効な通信回線を確保します。この方法で STUN/TURN サーバーはファイアウォールの外側にいるユーザーの障壁を取り除きます。そうしない場合はストリーミング アプリを使用するたびに、毎回システム管理者に特別な許可を求める必要があります。

LAN 内のコラボレーション

この最も基本的な共有設定では、ホストおよび、開発環境を共有する必要のある複数のパーティがすべて同じ Local Area Network (LAN) に存在します。コンテンツは付属するサーバー プラグインを介してストリーミングされ、ホストと参加者の通信には WebRTC API を利用します。

LAN がファイアウォールでセグメント化されている場合は、STUN/TURN サーバーが必要になる場合があります。

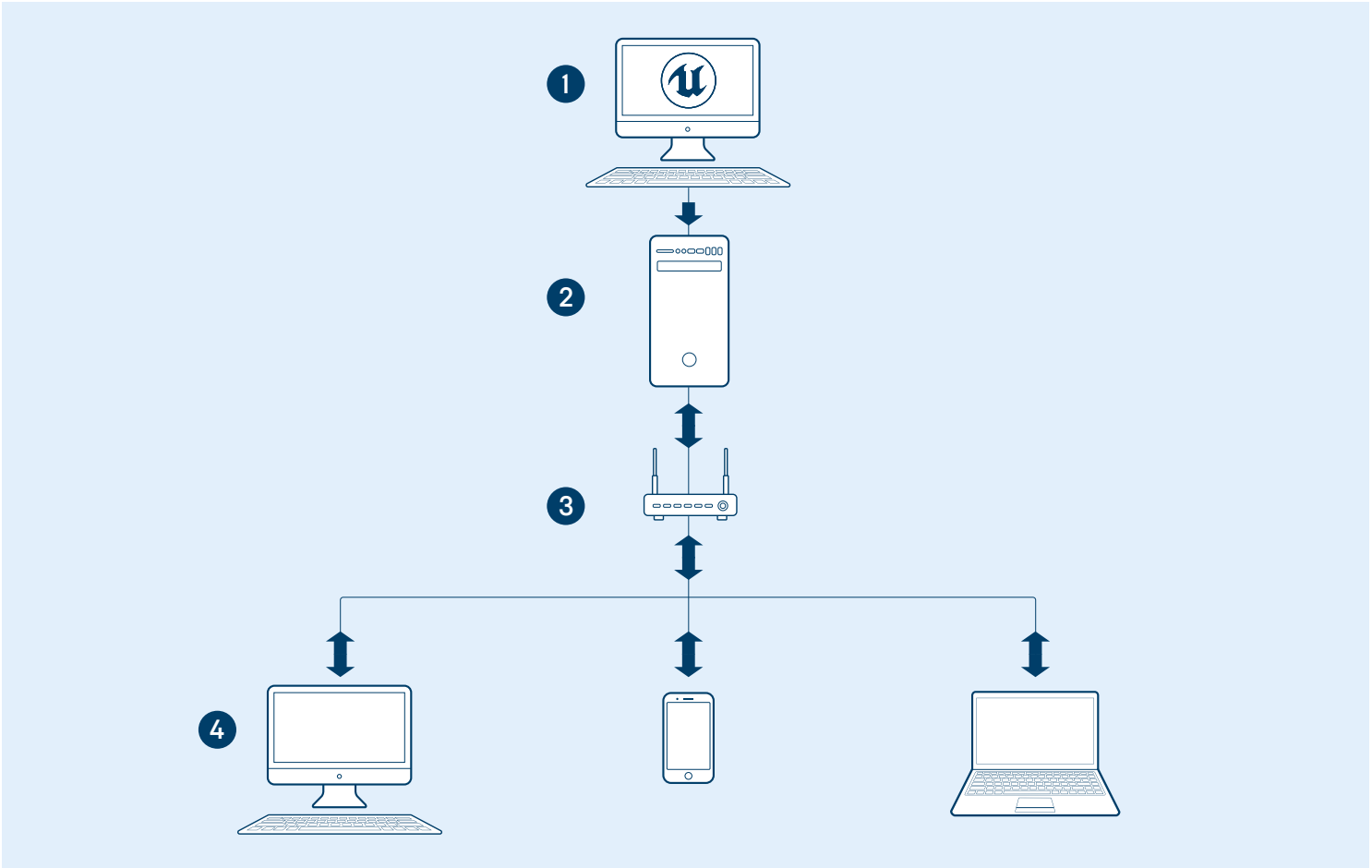


図 5：LAN 内のシンプルなコラボレーションの設定
要点：1.Unreal Engine 開発 2. WebRTC を使用した LAN 上の UE4 サーバー アプリ 3. ルーター 4. ユーザーの表示/インタラクション

コミュニケーション、承認、プレゼンテーション

プロジェクトを議論したり、独自のファイアウォールに阻まれた個別の LAN に接続した既知のリモート参加者にプレゼンテーションを行うために、忠実度の高いシステムが必要な状況向けに、Pixel Streaming は製品データを転送する必要がない安全な通信方法を提供します。このような設定では、ホストとクライアントの通信を処理するために STUN/TURN サーバーが必要です。

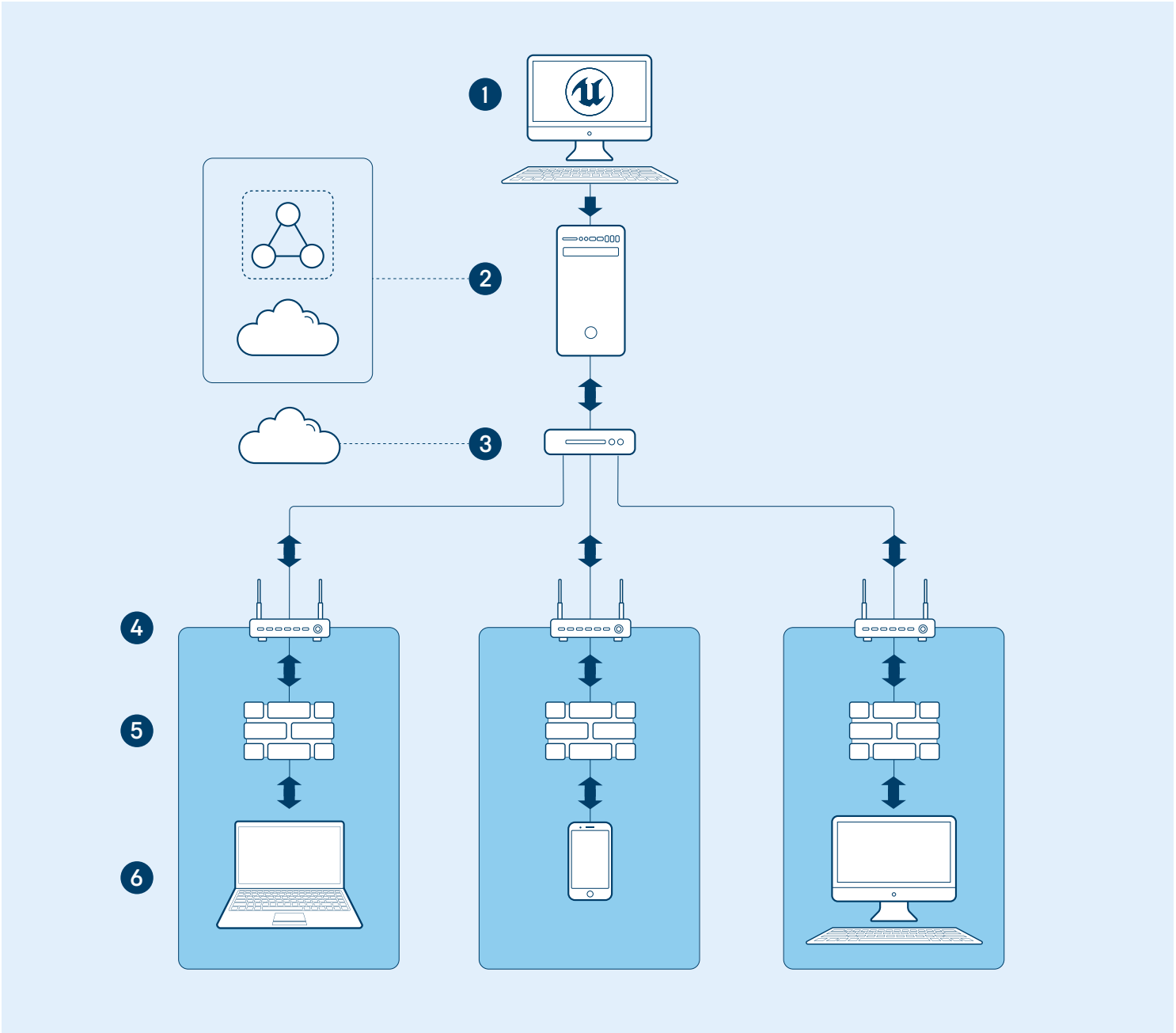


図 6：オンプレミス ネットワークまたはクラウドからの単一インターフェース接続
要点：1.UE4 開発 2. オンプレミス ネットワークやクラウドの UE4 サーバー アプリ 3. クラウドの STUN/TURN サーバー 4. ルーター/クライアント ネットワークの入り口 5. ファイアウォール 6. 表示/インタラクション デバイス

リモート協同デザイン、共有エクスペリエンス、消費者アプリケーション、コンフィギュレータ

アプリケーションが、さまざまなハードウェアとソフトウェアの設定を持った大勢のエンド ユーザーとやり取りしないといけない場合は、Pixel Streaming をスケーラブルなクラウド環境で実行できます。これは、安全なデータ アクセスと追跡可能なエクスペリエンスが必要となる自動車コンフィギュレータや他の共有エクスペリエンスで一般的な設定です。この種類の設定は、製品デザインについて共同作業するリモート チームにも適しています。

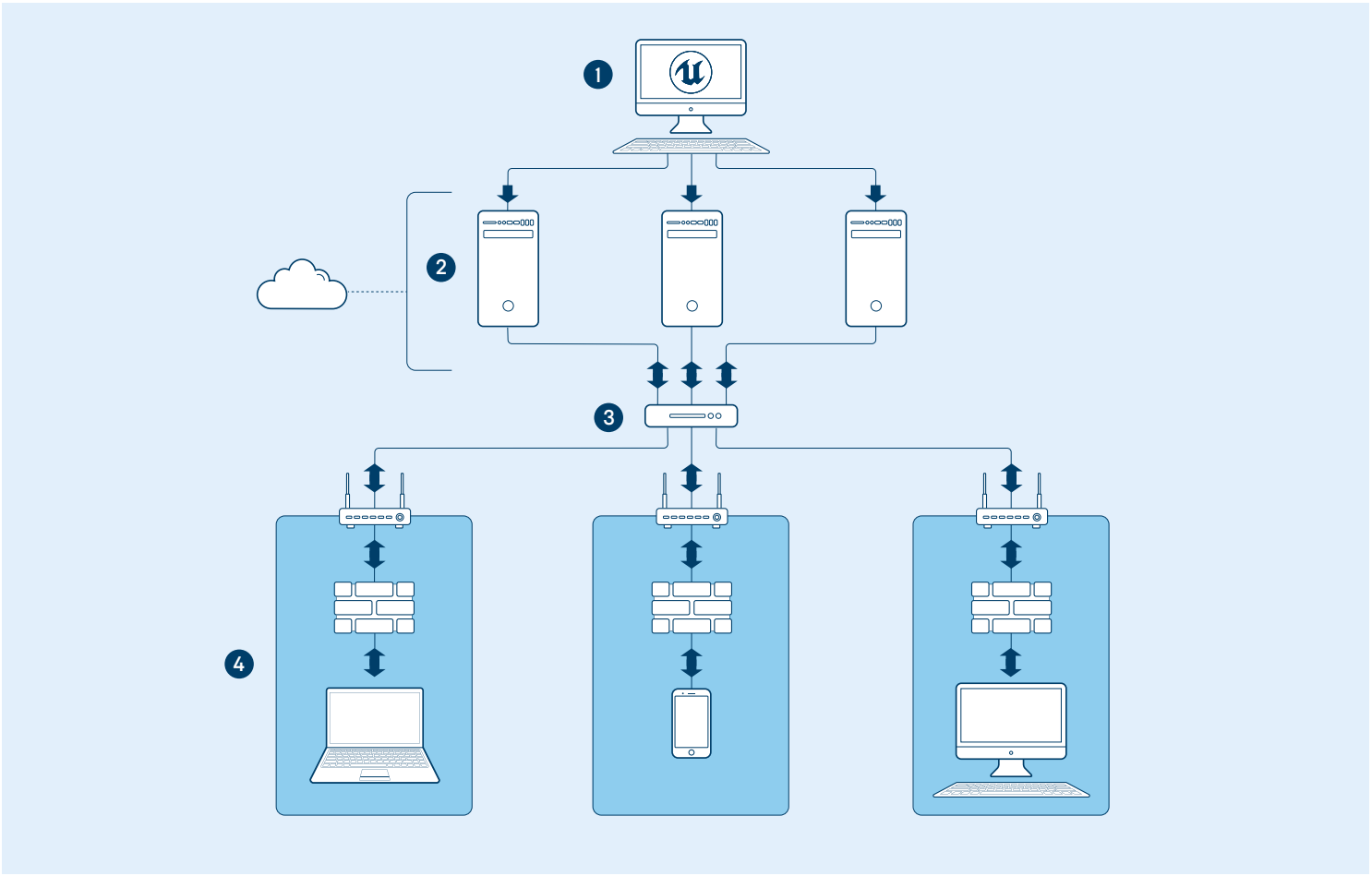


図 7 : ユーザーごとに一意の単一エクスペリエンス
要点 : 1.Unreal Engine 開発 2. クラウドで Web RTC を使用する UE4 サーバー アプリ 3. STUN/TURN サーバー 4. ユーザーの表示/インタラクション

ほとんどの商用ユースケースには、各ユーザーに独自のインタラクティブなエクスペリエンスとストリームが必要です。システムは、ユーザーごとに Pixel Streaming コンポーネントのスタックを個別に実行し、各ユーザーを個別の Web サーバーと (可能な場合は) 個別のホストに誘導します。

各スタックがエクスペリエンスを制御するには、一意の識別子とポートが必要です。多くのコンシューマ レベルのグラフィック カードは、同時に最大 2 つのエンコーダしか実行できないため、コンピュータで実行可能なインスタンス数は制限されます。NVIDIA Quadro や Tesla など、プロ グレードのカードやクラウド ベースの GPU インスタンス (AWS) には、このような制限がありません。

Matchmaker システムは、各リクエスタを独自のシグナリングおよび Web サーバーにリダイレクトを行い、クライアントとその WebRTC プロキシ サーバー間の接続を設定します。Matchmaker はユーザーがサーバー上でアクティブである限り単一のユーザーへのストリームを維持します。Matchmaker コンポーネントは、Unreal Engine や他のサーバー コンポーネントにあります。詳細については Unreal Engine Pixel Streaming に関するドキュメントの「ホスティングおよびネットワーキング ガイド」トピックの「マッチメイキングのある複数フルスタック」セクションを参照してください。

まとめ

Epic Games は、高品質な Unreal Engine のコンテンツを複数の種類のデバイスにストリーミングする方法を検討して選択肢を調べた結果、Pixel Streaming を開発することにしました。

WebGL は現在のホスト環境にデプロイする場合は低負荷であり、シンプルなユースケースには十分かもしれません。Epic は Unreal Engine の WebGL エクスポータに取り組んでいますが、これはシンプルな実装のみに適しています。

HTML5 を使用した方法で利益を出すのは既に限界を迎えており、この分野での開発は商用利用に向きません。こうした類のデプロイは、対象のユースケースとアプリケーションに重点を置いたカスタム開発次第となります。

Pixel Streaming は、利用可能な最高の忠実度で最高のコンテンツをデプロイすることに焦点が当てられている、エクスペリエンス時代でのコンテンツ配信に関したすべてのユーザーと技術的な考慮事項に対処しています。Pixel Streaming は、エンド クライアントやプラットフォームを考慮することなくハイエンド エクスペリエンスを共有することが可能で、オフライン コンテンツとオンライン コンテンツの両方にさまざまなチャンネルを持つ大規模なプラットフォーム設定で機能します。

次のステップ

UE4 の Pixel Streaming の設定を理解するには、このショーケース設定をご覧ください：

https://launcher-website-prod07.ol.epicgames.com/ue/learn/pixel-streaming-demo

詳細な手順は以下を参照してください。

https://docs.unrealengine.com/ja/Resources/Showcases/PixelStreamingShowcase/index.html

このドキュメントについて

著者

Heiko Wenczel

貢献者

Sébastien Miglio

Marco Anastasi

編集者

Michele Bousquet

Copyright © 2020 Epic Games, Inc. All rights reserved.