リグ ロジック

METAHUMAN フェイスリグの ランタイム評価

ホワイト ペーパー

UNREAL ENGINE

目次

はじめに **3**
- 用語 3
- 背景 5
- リアルタイムのフェイスリグ評価における課題 6

ソリューション **7**
- 要件 7
- ジョイント変形の低減 7
- 詳細度 (Level of Detail：LOD) 9
- MetaHuman DNA ファイル形式 11
- 表情補正 12

ランタイム戦略 **15**
- ジョイント数 15
- ブレンド シェイプ 16
- アニメーション マッピング 16

パフォーマンス **16**

まとめ **17**

はじめに

Rig Logic (リグ ロジック) は、3Lateral によって開発されたランタイム フェイシャル リグ評価ソルバ システムです。高度なキャラクターのカスタマイズとプレイヤーのデジタル化を行うシステムのインフラストラクチャ基盤で、忠実度の高いリグを作成できます。また、用途の広い特性評価を行えるツールでもあります。

図 1:リグ ロジックのユーザー インターフェース

リグ ロジックは、人間の顔の筋肉系を定義するための普遍的なルールセットと、3D オブジェクトのリグとジオメトリの完全な説明を保存するように設計された MetaHuman DNA と呼ばれる 3Lateral の独自のファイル形式に依存しています。

リグ ロジックは、2012 年の設立以来、フェイシャル リギングとアニメーションにフォーカスして研究開発を行っている会社である 3Lateral によって開発されました。同社のプロジェクトには、*Senua's Saga: Hellblade II* や *Marvel's Spider-Man* といった多数の映画やゲームが含まれます。3Lateral は 2019 年に Epic Games の一員になりました。Rig Logic は、Epic Games の MetaHuman Creator ツールで現在使用されているリグ ソルバです。

このホワイトペーパーでは、MetaHuman DNA ファイル形式の背景、リグ ロジック システムを構築する際に行ったその他の決定、そしてこれが顔のポーズやアニメーションのリアルタイム レンダリングで機能する理由について説明します。

用語

リグ ロジックがフェイシャル リギングの完全なリアルタイム ソリューションにどのように進化したのかを説明するには、まずいくつかの用語を定義する必要があります。

表情 - 笑顔、眉をひそめている、口の形などの特定の顔の表現。フェイシャル リギングとアニメーションでは、表情は、一連のジョイント トランスフォーム (平行移動、回転、またはスケーリング)、ブレンド シェイプ (モーフ ターゲットとも呼ばれます)、およびアニメーション マップによって定義されます。

図 2：MetaHuman のキャラクターの表情

ジョイント - フェイシャル リグは、仮想ボーンと、顔の実際の骨、関節、および筋肉をある程度模倣するジョイントで構成されています。実際の筋肉を正確に模倣することは難しいため、仮想の関節がその代わりとなってフェイシャル モデル上にある頂点のアニメーションを動かします。各ジョイントには、アニメートする頂点のセットが割り当てられています。

アニメーション カーブ - 時間の経過による頂点 XYZ の移動を表現します。

コントロール - 単一の基本的な表情のウェイトを保持するメカニズム。これは、表情を形作るジョイント、ブレンド シェイプ、およびアニメーション マップを計算するために使用されます。FACS Action Unit ごとに 1 つのコントロールがあるのが最適です。

リグ コントロール インターフェース - コントロールを組み合わせたり、コントロールを使いやすくしたりするインターフェース。これには、スライダーを備えたグラフィカル ユーザー インターフェース (GUI) から、コントロールを収集して使用して別のアクションを実行する内部インターフェースまで、あらゆるユーザー インターフェースが当てはまります。

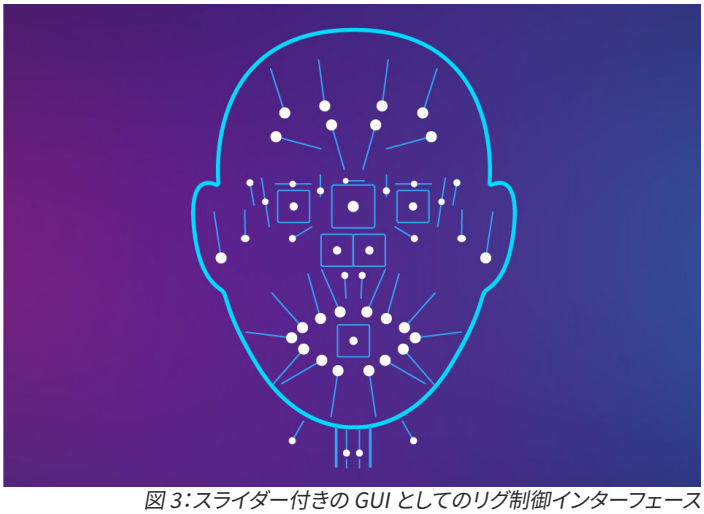

図 3：スライダー付きの GUI としてのリグ制御インターフェース

複雑度 - フェイシャル リグが保持するリグ コントロールの数を指すための一般的な用語。コントロールが 200 のリグは、コントロールが 50 のリグよりも複雑であるとみなします。コントロールが多いほど、表情補正を必要とする表情のバリエーションが広くなり、計算量が増え、メモリ要件が高くなります。

表情補正 - 複雑な表情を実現するための内部調整。表情補正については、このホワイトペーパーの後半で詳しく説明します。

背景

リグ ロジックは、その進化の過程でいくつかの主要なイテレーションを経ています。最新世代は、長年の研究開発を通じて蓄積された研究結果と改良が反映されています。

成功したフェイシャル リグでは、入力値はそれぞれが単一のフェイシャル FACS 式を表すコントロールとなっています。このような入力から、次の出力値を生成するための計算が行われます。

- ジョイントの変形
- ブレンド シェイプのチャンネルのウェイト
- アニメーション マップのシェーダー乗数

これらの 3 つの出力値は、現在使用されている最も成功したフェイシャル リグ システムのコアとなっています。当社の研究は主に、これら 3 つの値の計算と出力をより正確かつ効率的にすることを目的としています。

最初にリアルタイム リグの開発を開始した時、3Lateral では Maya が提供するツールキット、Autodesk のデジタルコンテンツ作成パッケージ、およびゲーム、映画、テレビ、その他の分野の 3D アーティストの間で人気のあるツールを使用して構築していました。Maya ツールキットには、表情補正とシェーダー ネットワークで補完されたドリブ ンキー (DK) が含まれています。変形は、これらの抽象化のみを使用して計算されました。これにより、顔を正常にアニメートするために必要な 3 つの出力値が得られました。

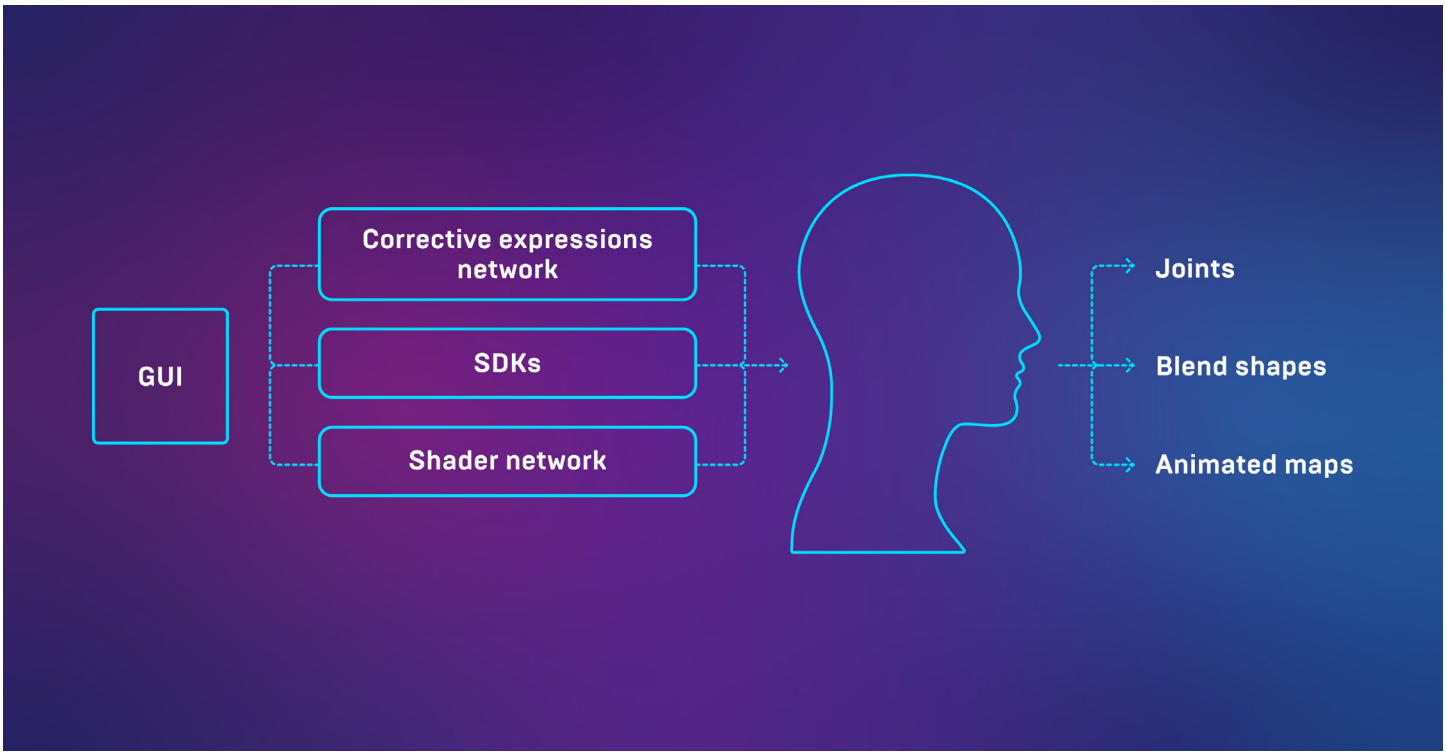

図 4：オリジナルの 3Lateral リアルタイム フェイシャル リグ評価プロセスにおけるデータの流れ

このソリューションはうまく機能し、高レベルな忠実度を実現できました。しかし、リグの評価は非常に遅く、他のテクノロジーとの統合は難しくなりました。

リアルタイムのフェイスリグ評価における課題

当社が直面した課題は、30 fps 以上でリアルタイムに計算しながら、同じまたは似たレベルの忠実度を生成するフェイシャル リギング システムを開発することでした。また、システムを他のテクノロジーと統合できるようにしたいとも考えていました。

リグ ロジックの開発目的の中核としては、Maya が使用するのと同じタイプの入力値に基づいて変形が行われる出力値を計算する一方で、実行速度を Maya よりはるかに高速にすることを目指していました。くわえて、顔の形に関係なく、キャラクター間でリグを再利用できるようにするという目標もありました。これにより、リグがエクスポートしやすくなり、さまざまなテクノロジーではるかに使いやすくなります。

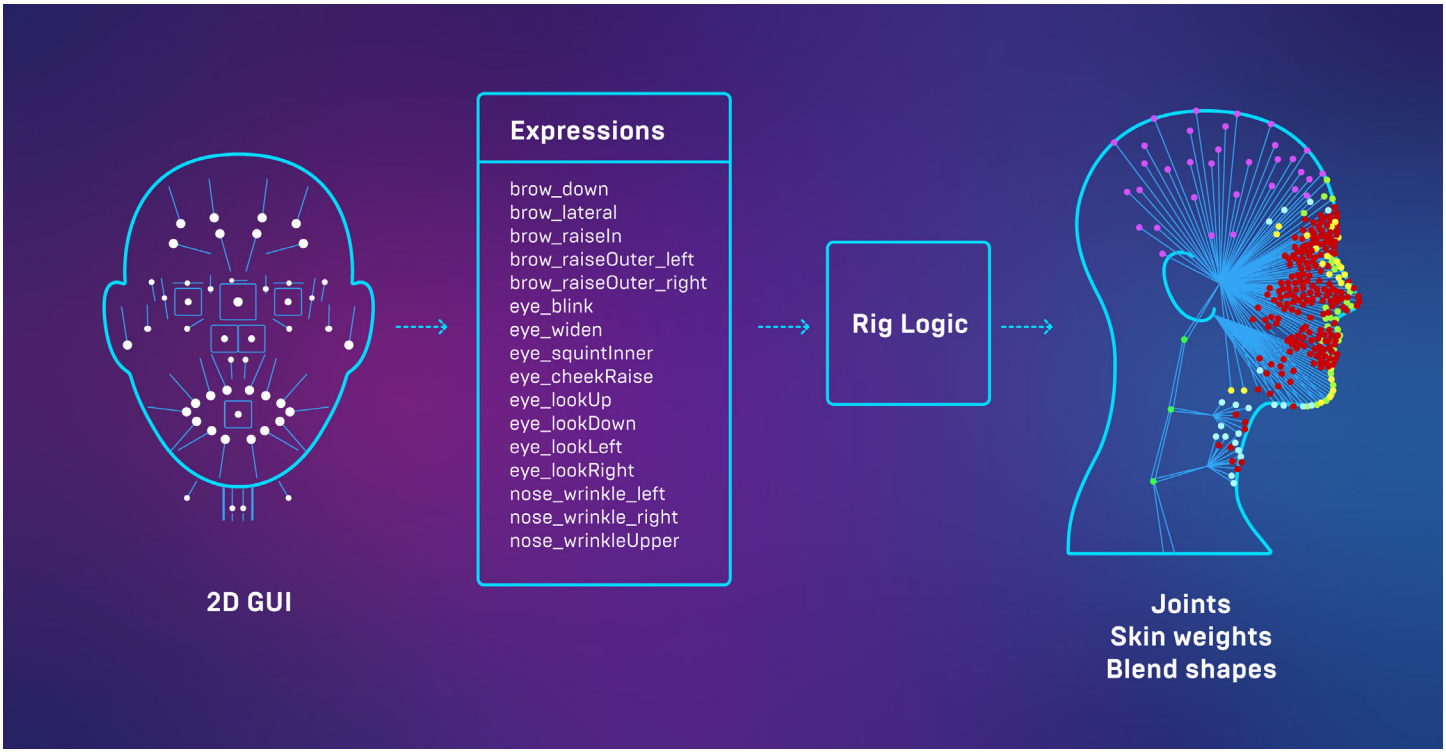

図 5：リグ コントロールから表情、リグ ロジックの計算、そして最後に顔のジョイントへのデータフローから顔を目的の表情に変更。

忠実度の高いフェイシャル アニメーションを実現している最先端のフェイシャル リグには、約 800 のジョイントと 200 を超える表情のコントロールが含まれ、1,000 以上の表情補正が追加されています。つまり、リアルタイム出力に対応できるランタイム ソリューションを開発したということです。

MetaHuman Creator のフェイシャル リグで使用する最も重要な要素は次のとおりです。

- ジョイントとスキン ウェイト デフォーマ
- ブレンド シェイプ デフォーマ
- ベースとアニメーション テクスチャのセットアップ
- Rig Logic プラグインと MetaHuman DNA ファイル
- ユーザー インターフェース
- ビルトイン LOD

ソリューション

要件

フェイシャル リギングとアニメーションの実行可能なランタイム ソリューションを実現するために、リグ ロジックはリグとその操作に関する次の要件の対処に着手しました。

- ランタイム評価。フェイシャル リグのランタイム評価は、30fps 以上などの妥当な速度でアニメーションをリアルタイムで再生できる状態が理想です。
- 入力および出力パラメータの削減。パラメータの数を減らすと、当然、システムの実行が速くなります。
- アニメーションの可逆圧縮。フットプリントが小さくなり処理が高速になります。データを圧縮すると、通常、CPU コストが最少になりデータサイズも 10～15 倍削減できますが、これは可逆圧縮時だけ可能です。
- リグの再利用性。顔の形、プロポーション、分布などにリグが依存しないようにシステムを設計します。これにより、異なるキャラクター間でアニメーション データを簡単に共有できるようになります。
- LOD の柔軟な使用。同じアニメーションが、すべての LOD 仕様を操作できるようにする必要があります。
- より効率的なパイプライン。実行時に評価するリグを作成すると、当然、特定のパイプラインの効率性が向上します。たとえば、これによりデータ サイズがはるかに小さくなるため、アニメーションのエクスポートをより素早く実行できます。システムが正しく設計されている場合、リグの更新を行ってもすべてのアニメーションを再エクスポートする必要がなくなります。
- フェイシャル アニメーション ソフトウェアと簡単にリンク。このソリューションは、リグの筋肉ベースの組織を使用していることから、SpeechGraphics といったオーディオ ベースのフェイシャル アニメーション ソリューションとの統合が簡単です。
- 非線形アニメーション ミキシングをサポート。システムは、実行時に動的な応答をサポートしている必要があります。たとえば、キャラクターは、懐中電灯の光がキャラクターの目を反射するといったように、ゲーム内のイベントにインタラクティブに応答しながら動作し続ける必要があります。このソリューションは、アニメーションによる感情の階層化もサポートしています。

ジョイント変形の低減

リグ ロジックは、リグを Maya から切り離すために、フェイシャル リグを表す必要なデータをキャプチャし、それをリアルタイムで評価するためのランタイム コンポーネントを提供する試みとして開発されました。リグ ロジックの初代では、Maya からデータを抽出し、次のような線形関数でドリブン キーを表現しています。

$y = k*x$

y 値は x が増加するにつれて直線的に増加し、y が増加する程度 (傾き) は *k* によって決定されます。

- *x* は入力制御値であり、この値は [0.0,1.0] の範囲に制限されています。
- *y* は出力変形値です (例：joint.rotation.x パラメータの値)。
- *k* は、関数の動作を定義するリグ パラメータです。

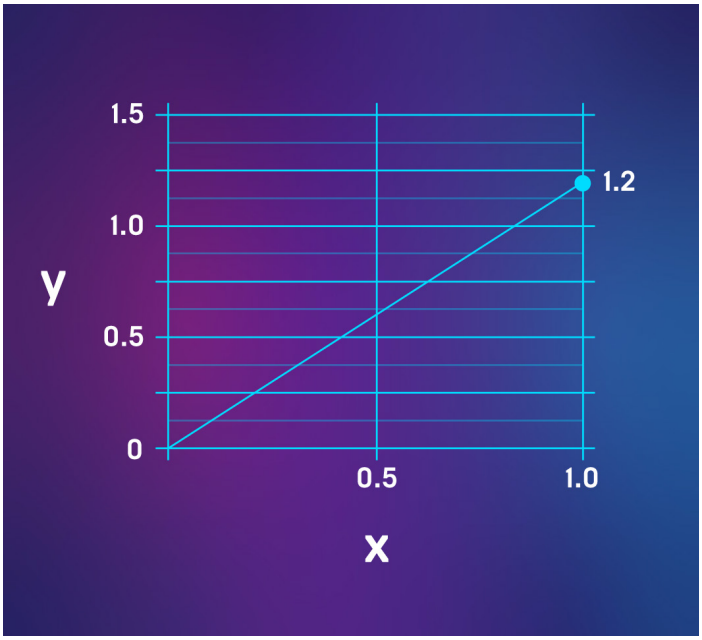

*図 6：x(入力)、y(出力)、および k (リグのパラメータ) 値を決定するための傾き関数 y = k*x の式。*

図 7：MetaHuman のキャラクターの表情

x 値と *y* 値をこのように処理することで、これらの傾き関数のパラメータを行列にエンコードすることが可能になりました。

ストレージ スペースの要件を最小限に抑え、適切な計算パフォーマンスを提供することを目的として、マトリックス タイプにはスパース Compressed row storage (CRS) マトリックス を選択しています。このバージョンのリグ ロジックでは、ジオメトリック プリミティブは対処されていません。つまり、システムはまだ Maya のジオメトリック プリミティブ (メッシュ) に依存していました。

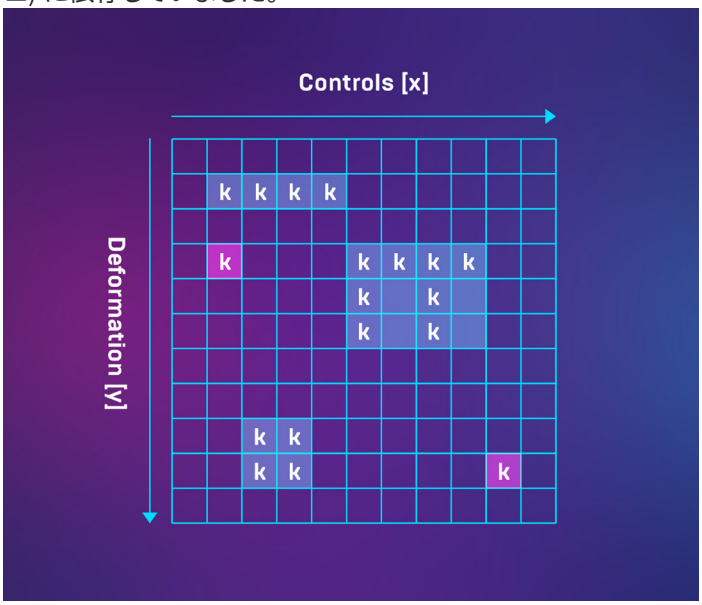

*図 8：一次関数 y=k*x で計算されている、ジョイント変形デルタ (k) の代表的なサンプル マトリックス*

このマトリックスを後で分析すると、これらパラメータの密度が高いグループは、他の点ではほとんど密度の低いマトリックスが形成されていることがわかりました。また、これら同じ密度の高いサブマトリックスには、顔の特定の領域をターゲットとした変形パラメータが含まれていることがわかったため、 *ジョイント グループ* というコンセプトが導入されました。ジョイント グループは、これら密集したサブマトリックスを表しています。それにより、顔の特定の領域をターゲットとするグループへのジョイントを合理的に分離します。

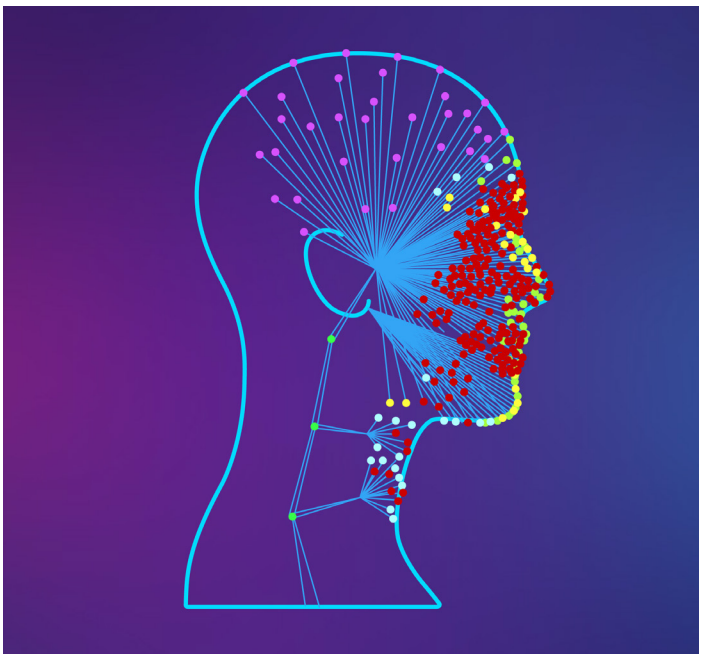

図 9：ジョイント グループ

図 8 に示されているように、パラメータには外れ値、つまり密度の高いサブマトリックスの一部ではないものもありました (図では、外れ値はピンクで表されています)。これらの外れ値は、問題のジョイントがまったく影響を及ぼさないはずの顔の領域における変形を表していることが判明しました。この観察の結果、これらの外れ値を取り除くスマートなプルーニング プロセスが実装されたことから、密度の高い領域のみがマトリックスに残されました。

プルーニングにより、実行時に保存および評価する必要のあるデータを効率的に削減し、誤った変形を取り除くことができました。その結果、残った密度の高い領域は、ジョイントを評価するための非常に高速なアルゴリズムを実装するのに役立ちました。

その後のイテレーションでは、異なるデータ型 (ジョイント、ブレンド シェイプ、アニメーション マップ) も、それぞれは条件付き線形関数を表す別々の行列に分離されました。

詳細度 (Level of Detail：LOD)

顔の表情は、2 つ以上の基本的な表情で構成されているのが一般的です。複雑な表情の例としては、(1) 眉を上げる、(2) 目を大きく開く、(3) 唇を少し開ける、といった 3 つの基本的な表情の組み合わせがあります。

これらの複雑な表情の作成を可能な限り高い効率で実現するため、リグ ロジックは LOD をサポートすることを念頭に置いて設計されました。この際、すべての表情は特定のジョイントの変換によって定義されますが、すべてのジョイントが特定の表情に対して影響を与えるわけではないことに注意する必要があります。最も詳細なレベルである LOD0 では、表情の形成にはその表情に対して参照されているすべてのジョイント変形が使用されます。しかし、下位 LOD (LOD 1、LOD 2 など) は、LOD0 で参照されているジョイントの厳密なサブセットを使用します。

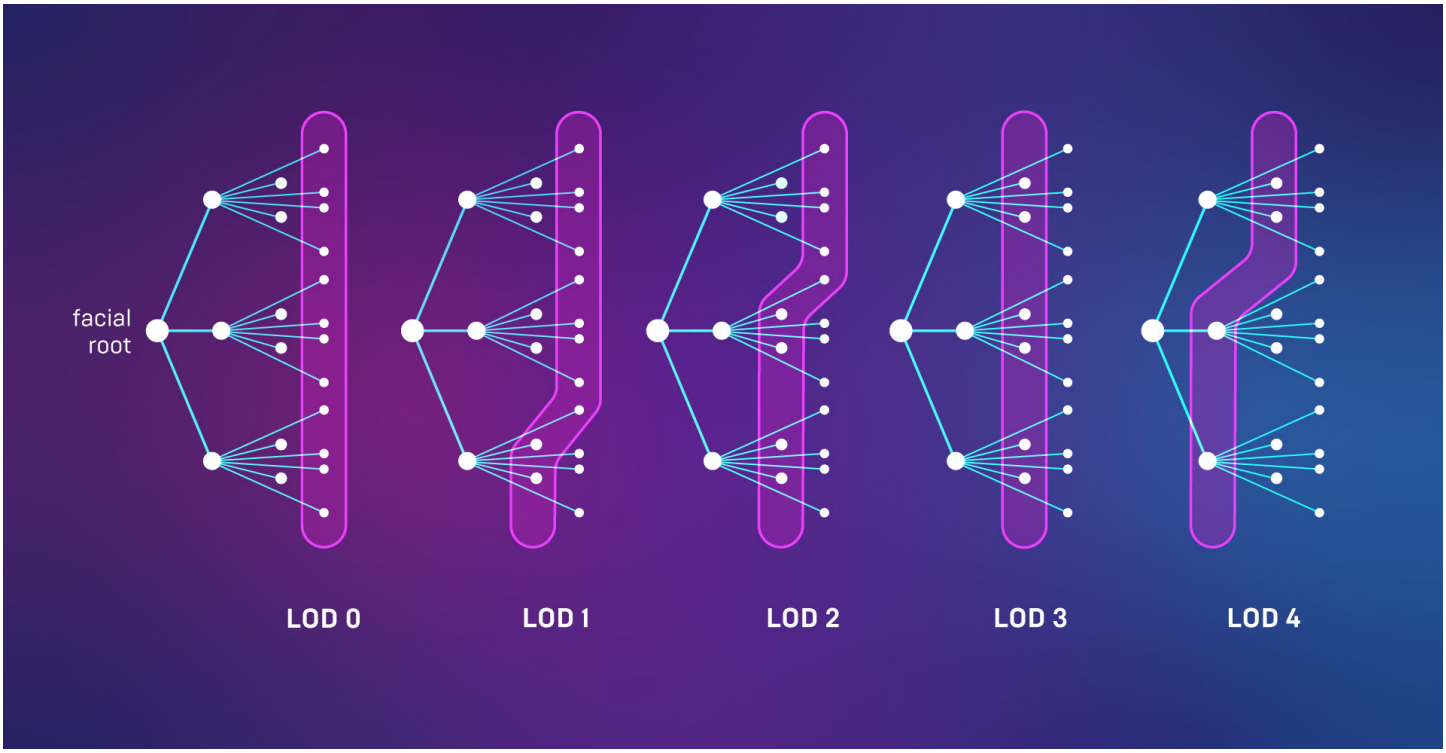

図 10：関連するジョイントの数が LOD0 から LOD4 に変化し、最も密度の高いトポロジと最大のジョイント数 (LOD 0) から、最も密度の低いトポロジと最小のジョイント数 (LOD 4) に変化した図。

下位 LOD で参照されるジョイントを LOD0 で参照される同じジョイントに制限すると、データの浪費を最小限に抑えながら、LOD を切り替えるための手間がゼロになります。低い LOD から高い LOD に変更した場合、指定されたジョイントが既にロードされているデータに追加されます。一方、高い LOD から低い LOD に変更した場合、参照されていないジョイントが無視されます。

システムには 8 つの LOD がビルドされていますが、必要に応じて、リグ ロジックをカスタマイズしてさらに追加することもできます。ジョイント数の制約と顔のさまざまな部分に必要な変形のクオリティに応じて、さまざまなトポロジのジョイントとスキンのウェイトを簡単に組み合わせることで、色々な LOD を形成できます。

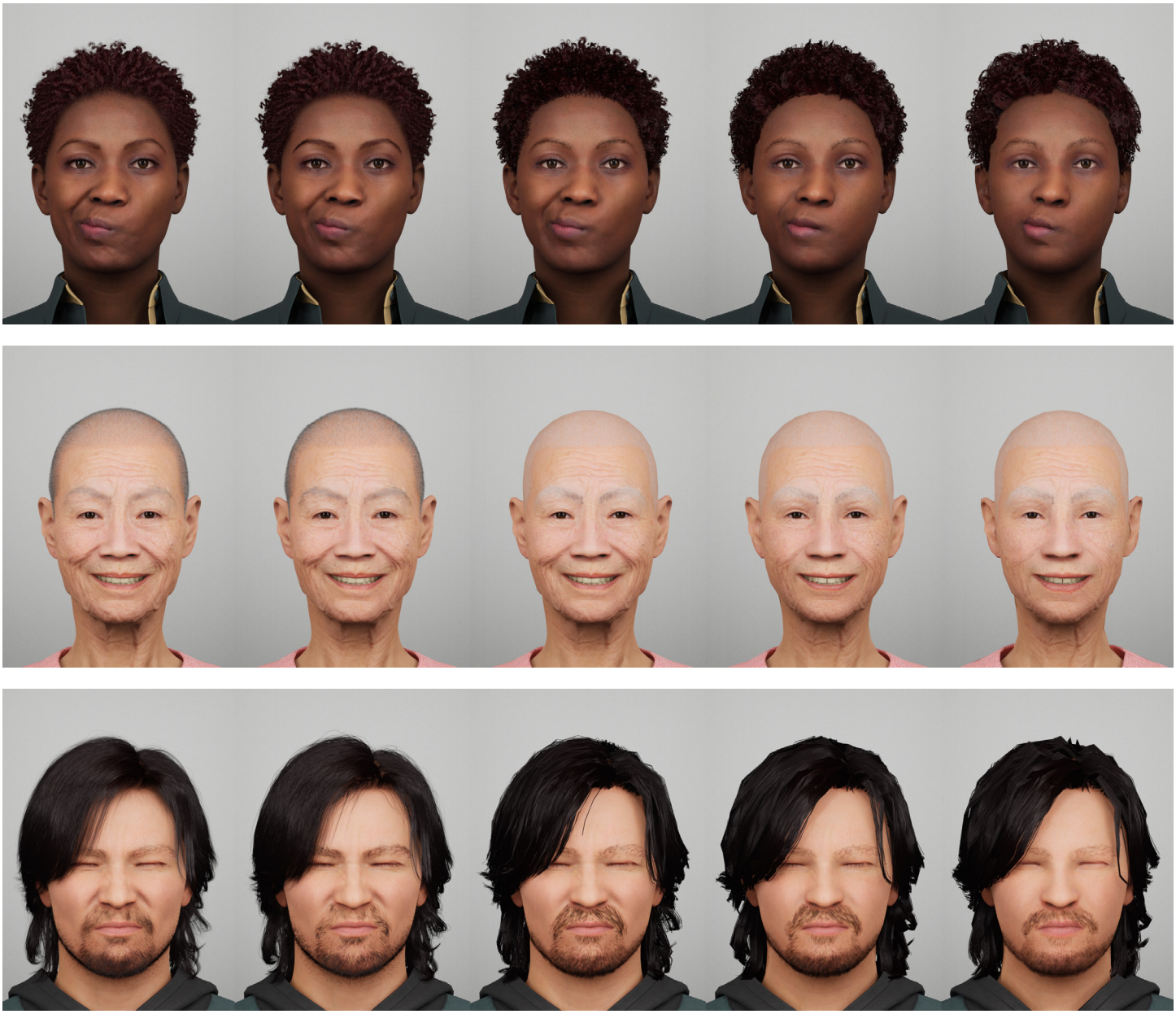

図 11：LOD をゼロコストで高いものから低いものへと切り替えた表情の例。

メモリ使用量をさらに削減するために、実行時に必要となる予想 LOD を指定することもできます
(または、最大 / 最小 LOD を指定できます)。選択した LOD によって参照されていないデータはロードされません。

MetaHuman DNA ファイル形式

3Lateral の MetaHuman DNA ファイル形式は、3D オブジェクトのリグとジオメトリの完全な設計図を保存しています。MetaHuman DNA ファイルを使用すると、オブジェクトの完全なメッシュを再構築し、完全にリギングして、アニメートする準備を整えることができます。現実問題としては、一般的に MetaHuman DNA ファイルは人間のキャラクターの顔のみを保存するのによく使用され、体、小道具、またはその他のオブジェクトを保存するのに使用されることはあまりありません。

MetaHuman DNA ファイルの内容は、独自のバイナリ形式を使用してエンコードされていますが、形式の仕様とそれを読み取るコードはオープンソースです。

レイヤー

MetaHuman DNA ファイルに含まれるデータは、いくつかの論理レイヤーに分割されています。レイヤーは緩い階層で接続されており、MetaHuman DNA ファイルの後続の各レイヤーは、その下のレイヤーに格納されているデータに依存しています。

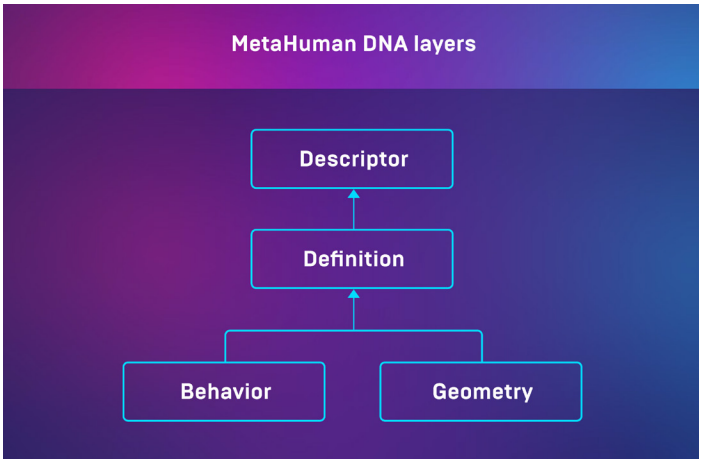

図 12：MetaHuman DNA ファイルの論理レイヤー

指定されたレイヤーまでのみ、MetaHuman DNA データを選択的にロードすることもできます。図 12 に示されているように、Behavior レイヤーと Geometry レイヤーは相互に依存していません。MetaHuman DNA ファイルがジオメトリ データにアクセスすることなくリグの操作だけを行うといったユースケース (Behavior レイヤーを使用したランタイム評価) に対応するには、このような独立性が重要となります。

独立性によりソリューションは移植可能なので、Maya ジオメトリやその他の特定のジオメトリに適用することなく、これをフェイシャル ジオメトリが存在する他のアプリケーションで使用できます。このファイル形式は、リアルタイムなフェイシャル アニメーションを実現するために効率的かつ完全な形式を生成できるように設計されています。

Descriptor レイヤー

Descriptor レイヤーには、リグに関する次の基本的なメタデータが含まれています。

- キャラクターの名前
- 年齢
- フェイシャル アーキタイプ
- キーと値のペアの形式の任意の文字列メタデータ
- 必要となる互換パラメータ (MetaHuman DNA ファイルのミキシングなどの高レベルシステムに関連したもの)

Definition レイヤー

Definition レイヤーには、次のようなリグの静的なデータが含まれています。

- コントロール、ジョイント、ブレンド シェイプ、アニメーション マップ、メッシュの名前
- ジョイント、ブレンド シェイプ、アニメーション マップ、およびメッシュの個々の LOD へのマッピング
- ジョイントの階層
- バインディング ポーズ (T ポーズなど) のジョイントの変形

このレイヤーには、選択した LOD に基づいて後続のレイヤーでフィルタリングを実行するために必要な情報が含まれています。

Behavior レイヤー

Behavior レイヤーには、リグの動的なデータが含まれています。これは、次の目的で使用されます。

- GUI コントロールの生のコントロール値へのマップ
- 表情補正の計算
- ジョイントの変形の計算
- ブレンド シェイプ チャンネルのウェイトの計算
- アニメートされたマップのウェイトの計算

Geometry レイヤー

Geometry レイヤーには、キャラクターのメッシュを再構築するために必要なすべてのデータと、そのスキン ウェイトおよびブレンド シェイプ ターゲット デルタが含まれています。メッシュの情報自体は、OBJ ファイル形式に似た形式で構造化されています。

図 13：MetaHuman のキャラクターの表情

表情補正

複雑な表情の場合、いくつかのジョイントが複数の基本的な表情の影響を受けるといったオーバラップが発生します。このような場合、指定されたすべての変形をアクティブ化するだけだと変形が「掛け合わされる」ことになり、通常、外観に悪影響をもたらします。

表情補正は、複雑な表情のジョイントのオーバーラップを修正するために用います。表情の問題のある領域に発生した歪みを元に戻すため、複雑な表情それぞれに表情補正を定義する必要があります。そして、このような複雑な表情は、基本的な表情の組み合わせ (または少なくとも重要と考えられる組み合わせ) ごとに定義する必要があります。

したがって、サポートされる基本的な表情の数が増えると、複雑な表情の数も指数関数的に増えることになります。考えられるすべての組み合わせに対して表情補正を有効にするのは、メモリと計算のコストを考慮すると不可能です。したがって、その中で最も影響の大きい組み合わせだけに対して行うのが良いでしょう。

図 14 は、怒りながら叫んでいる顔のポーズを表現するための顔の変形の例を示しています。このようなポーズの作成には、下記のいくつかの基本的な表情の組み合わせが必要になる場合があります。

- jawOpenExtreme
- mouthCornerPull
- mouthStretch
- neckStretch
- noseWrinkler
- squintInner
- browsLower

これらの表情には、ジョイントのオーバーラップがいくつか存在することがあります。たとえば、口と鼻の表情を修正せずに合わせると、顔のその領域の周りに極端なストレッチが発生する可能性があります。

図 14：基本的な表現の特定の組み合わせに表情補正を適用する前 (左) と後 (右) の顔のポーズ。

左側の顔は、上記の表情を単純に合わせており、不自然なポーズとなっています。これは、この特定の表情の組み合わせがアクティブになるたびに特定の表情補正を適用してポーズを調整することで解決できます。

右側の顔は、この表情の組み合わせに修正表情を適用しており、より自然で説得力のあるポーズになっています。

動作中の表情補正の他の例を図 15 に示します。

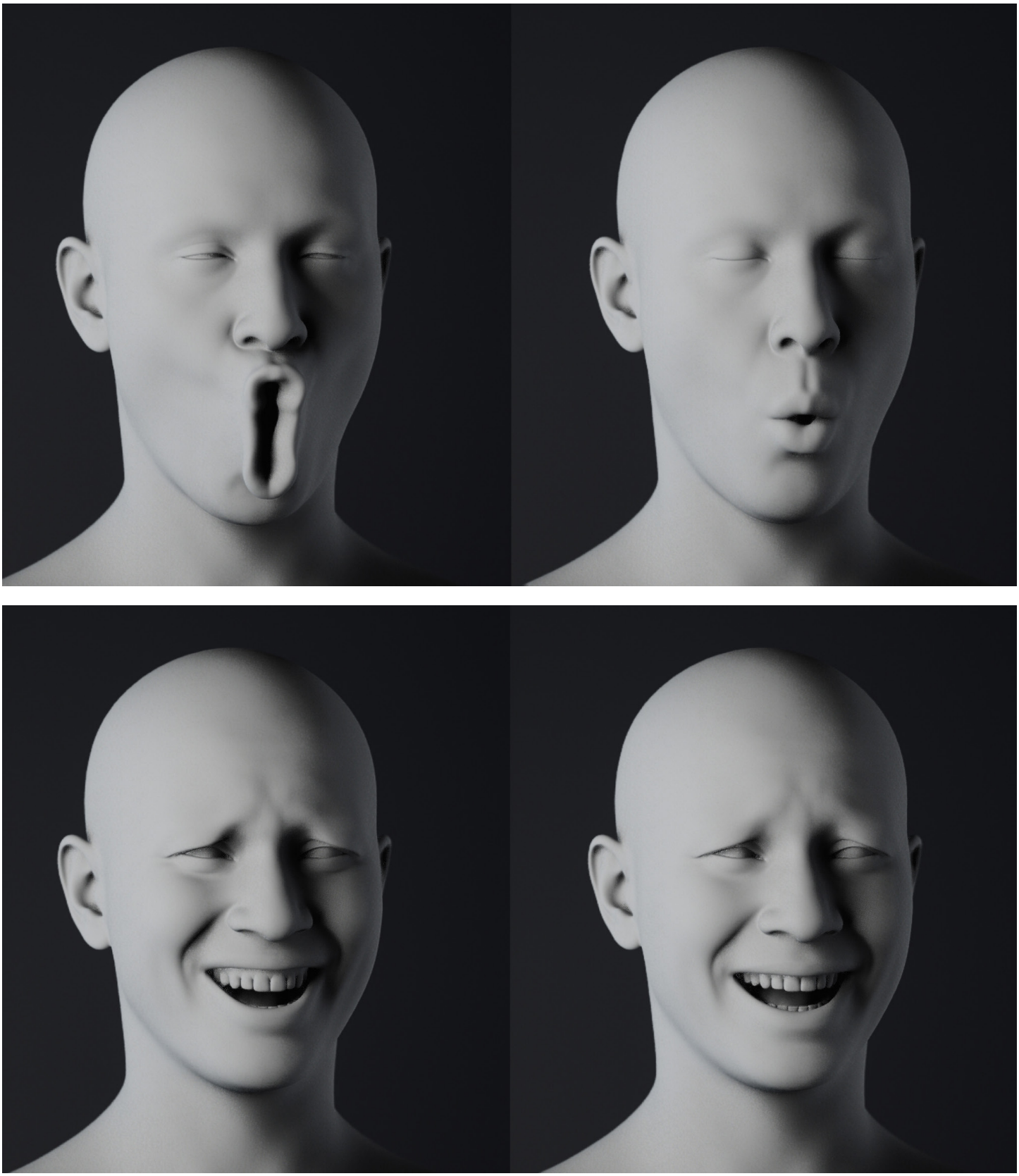

図 15：変形が「掛け合わされた」顔のポーズ (右)、それに表情補正をかけた顔のポーズ (左)。

ランタイム戦略

ジョイント数

リグ ロジックでは、プルーニング処理によってジョイントがジョイント グループに分けられています。MetaHuman DNA ファイルの準備中は、結果に悪影響を与えかねない望ましくないデータを取り除くことでジョイント変形を引き起こすデータを削除できます (この作業によるダメージは特に大きくありません)。また、この作業は保存する必要のあるデータの量を減らし、実行時に実行する必要のある計算の量を減らせるというプラスの効果があります。

プルーニングのプロセスの概要は次のとおりです。

- 顔をいくつかの大まかな領域 (額、顎、あごなど) に分ける。
- 影響を受ける顔の領域に基づいてジョイントをグループに分ける。
- 各表情と表情の影響を受けるジョイント グループとの間にマッピングを確立する。
- ジョイント グループによる制御を想定していない領域に対して影響を与えるデータを含まないように制約を定義する。

つまり、ジョイント グループが組み立てられて MetaHuman DNA ファイルにエクスポートされる場合、基本的に、ジョイント グループが割り当てられている顔の領域に属するジョイントのデータのみが含まれ、そのデータは処理中のジョイント グループに影響を与える可能性のある特定の表情のみの変形が含まれるようにフィルタリングされるということです。

前述のジョイントの LOD メカニズムは、実際には、各 LOD ごとに、各ジョイント グループに対して個別に定義されます。

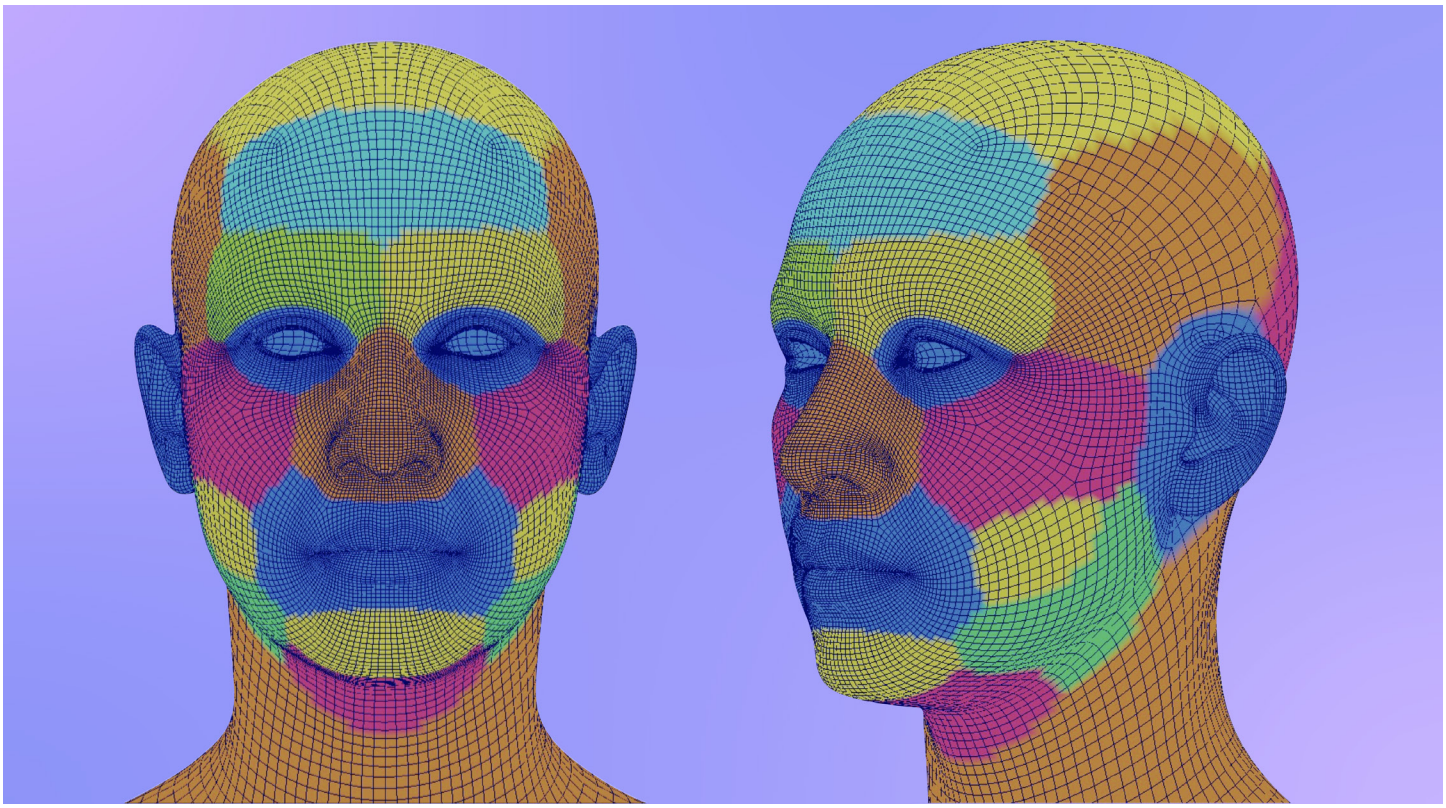

図 16：色で定義された顔の領域

ブレンド シェイプ

リグ ロジックによって計算されたブレンド シェイプ チャンネルのウェイトは、ブレンド シェイプ ターゲット デルタをアクティブ化する必要がある強度を表します。ブレンド シェイプのターゲット デルタは、リグ ロジック自体によって生成されるのではなく、MetaHuman DNA ファイルのジオメトリ セクションに保存されます。計算では、リグ ロジックが受け取る入力制御のウェイトの単純な再マッピング処理を行います。

ジョイントの場合と同様に、ブレンド シェイプ チャンネルのウェイトを計算するためのデータも LOD によって順序付けられます。したがって、選択した LOD の計算を制限するには、単純なスライス操作で十分です。

アニメーション マッピング

アニメートされたマップによって生成される出力は乗数であり、しわ、皮膚の下の血流の影響などを制御するシェーダーに直接影響を与えます。

アニメートされたマップは他のソリューションと同じ仕組みで LOD によって順序付けられるので、これも同様に LOD スイッチングをゼロコストで行えます。

パフォーマンス

ジョイントをジョイント グループに分離するプロセスを通じて、ジョイント ストレージにスパース行列の代わりに密度の高いサブマトリックスを利用することが可能でした。この方法は、ジョイント変形を計算するための非常に効率的なアルゴリズムを実装するのに有益であることが証明されています。当社は、キャッシュと SIMD に適した方法でデータを小さなブロックに分割するカスタム マトリックス レイアウトを使用し、スカラー剰余ループが不要になるように最小限のパディングを行いました。したがって、コードパス全体が完全にベクトル化されています (パディング メモリのオーバーヘッドは 2% 未満です)。

同じ操作に単純な CSR マトリックスを使用していた前世代のリグ ロジックと比較して、この新しいバージョンでは、評価時間が約 6 倍向上しています。同時に、評価されるジョイントの数も大幅に増加しました。結果のコードは、ほぼ理想的な CPU パイプライン使用率を実現しています。

まとめ

リグ ロジックは、従来のリアルタイムのフェイシャル アニメーション ソリューションに比べて大幅に改善されています。多くの側面で効率性や移植性が高く、機能も多いため、これは MetaHuman Creator およびその他の実装で実践的なソリューションであることが証明されています。

将来的には、Epic Games のプロジェクトのニーズに対応する技術の開発を継続することにより、リグ ロジックをさらに改善する予定です。また、私たちはリグ ロジックの将来のバージョンの開発に生かして推進するため、最新のテクノロジーにもキャッチアップしています。

このドキュメントについて

技術開発
3Lateral

著者
Andrean Franc

寄稿者
Pavel Kovac

編集
Michele Bousquet

グラフィックとレイアウト
Jung Kwak

注記：リグ ロジックは、MetaHuman Creator、MetaHuman キャラクター アセット、および Unreal Engine に組み込まれています。リグ ロジックの使用は、これらの製品に適用されるライセンスの対象となります。

Copyright © 2022 Epic Games, Inc.All rights reserved.