
GPU上でのリアルタイム ヘア・シミュレーション
Bryan Dudash
NVIDIA.


NVIDIA

結果

- 髪の毛166本のシミュレート
- 99万トライアングル

- 静止状態: 64 fps
- 運動状態: 41 fps

- 8800GTX, 1920x1200,
- 8XMSAA

結果

- 髪の毛166本のシミュレート
- 210万トライアングル

- 静止状態: 24fps
- 運動状態: 17.5fps

- 8800GTX, 1280x1024,
- 2x SSAA, 8XMSAA

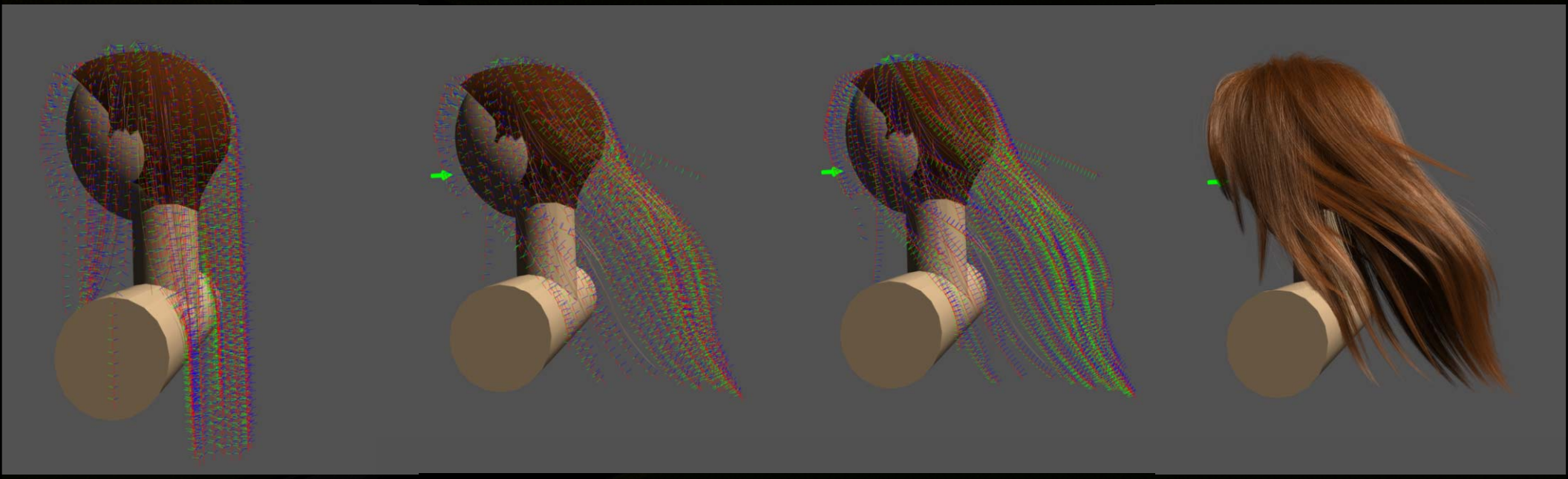
Main Contributions

- 全てGPU上で動作
  - GPU上でシミュレーション
  - GPU上でテセレーションと補間
- Robust安定した inter hair forces
- 補間した髪の毛の衝突判定と衝突回避

Previous work

- リアルなリアルタイム・ヘアに関して多くの人が興味深い研究をしています:
  - **Dual Scattering Approximation for Fast Multiple Scattering in Hair. Zinke and Yuksel**
  - **A practical self-shadowing algorithm for interactive hair animation. Bertails et al.**
  - **Algorithms for Hardware Accelerated Hair Rendering. Tae-Yong Kim**
  - **Real-Time Approximate Sorting for Self Shadowing and Transparency in Hair Rendering. Sintron and Assarsson**
  - **Deep Opacity Maps. Yuksel and Keyser**

Process



ガイドとなる髪の毛を取り込む

ガイド・ヘアのシミュレート

ガイド・ヘアのテセレートと補間

最終的な髪の毛のレンダリング

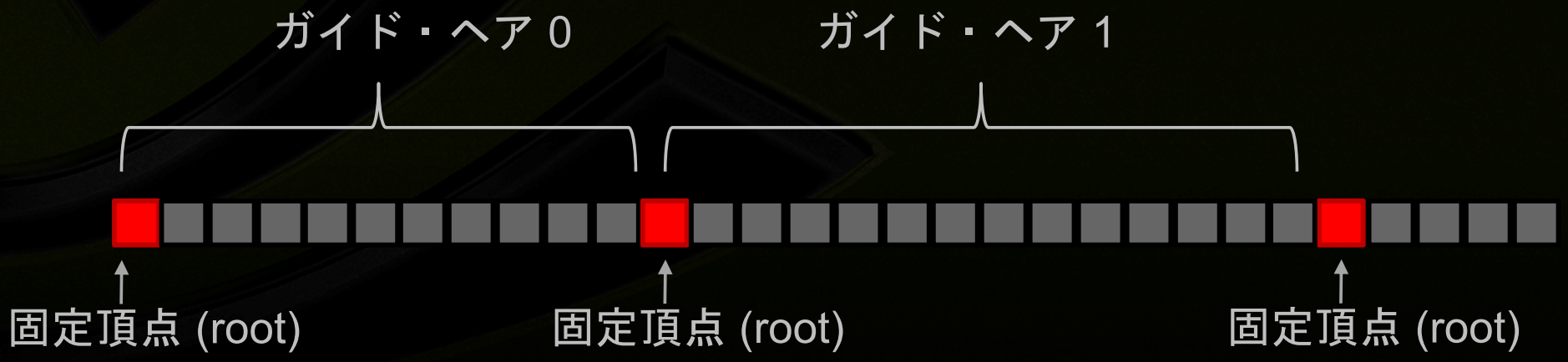
シミュレーション

シミュレーション

- パーティクル制限システムとしての
  ヘア・シミュレーション
  - 髪の毛のバーテックスはパーティクルとしてシミュレーション
  - 髪の毛のバーテックス同士を**距離束縛**結合する
    - これらの制限によって髪の毛の長さを保つ
  - 各々の髪の毛のバーテックスの**角力**が髪の毛の形を保つ
    - ２次元の角力を考える (ねじれ方向は無視する)
  - **衝突制限**によって障害物をヘア・パーティクルが避けるようにできる

Representation

- 全てのガイド・ヘアのバーテックスで一つの Vertex Buffer (VB)を構成する

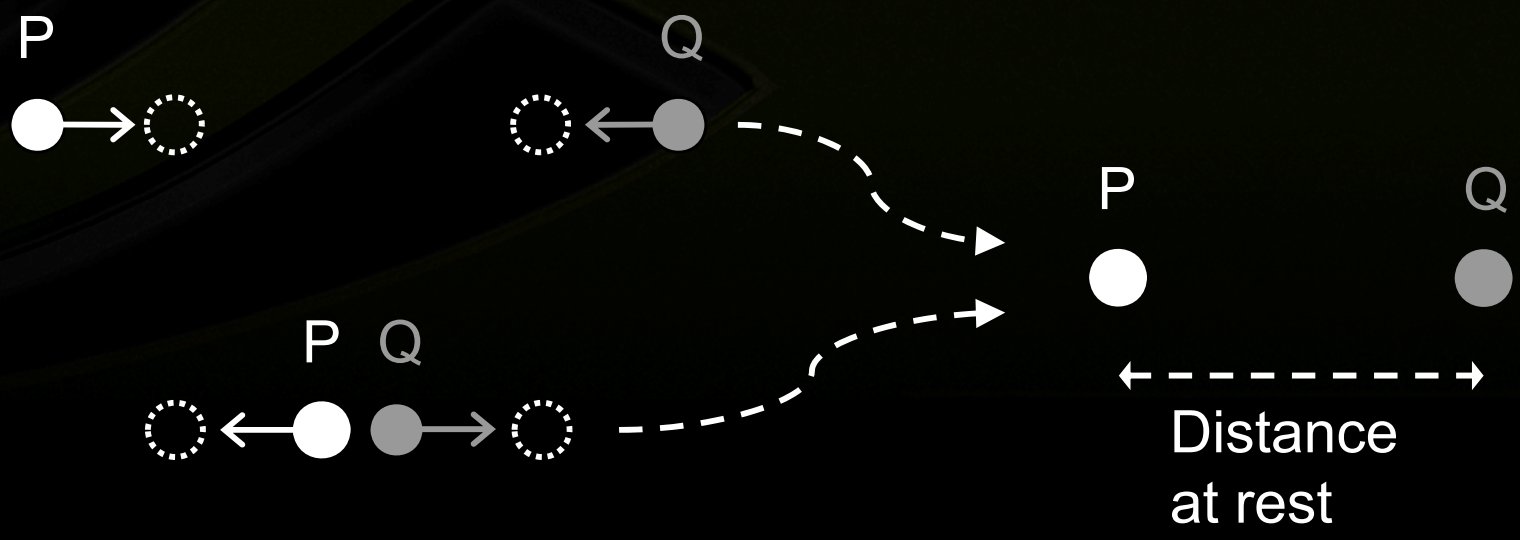


GPU上での物理演算

- VertexシェーダとStream Outputでシミュレーションは完結する
  - Vertexシェーダからラスタライザはスキップして直接Vertex Bufferへ書き出す
- 二つのガイド・ヘアVB間でピンポン
  - VB1をバインドしてvertexシェーダを走らせ、VB2にStream Outする
  - VB2をバインドして次のvertexシェーダを走らせ、VB1にStream Outする
  - 続く・・・

例：距離束縛

- 二つのパーティクルP, Q間での距離束縛 DC(P, Q) はお互いの位置関係を決定する:

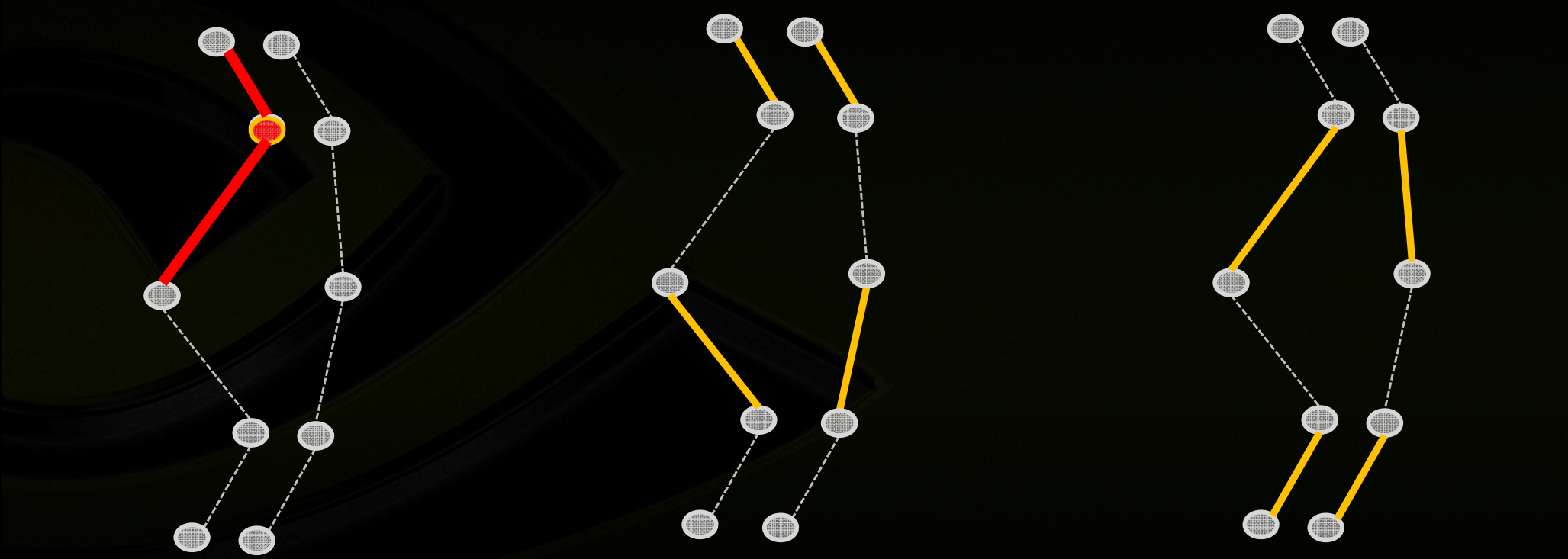


例: 距離束縛

- 距離束縛を満足するためには、
  二つのバーテックスを動かす必要がある

- Geometryシェーダを使う
  - ２頂点を入力
  - 距離束縛を満足する位置に動かした結果のバーテックスを出力

例: 距離束縛



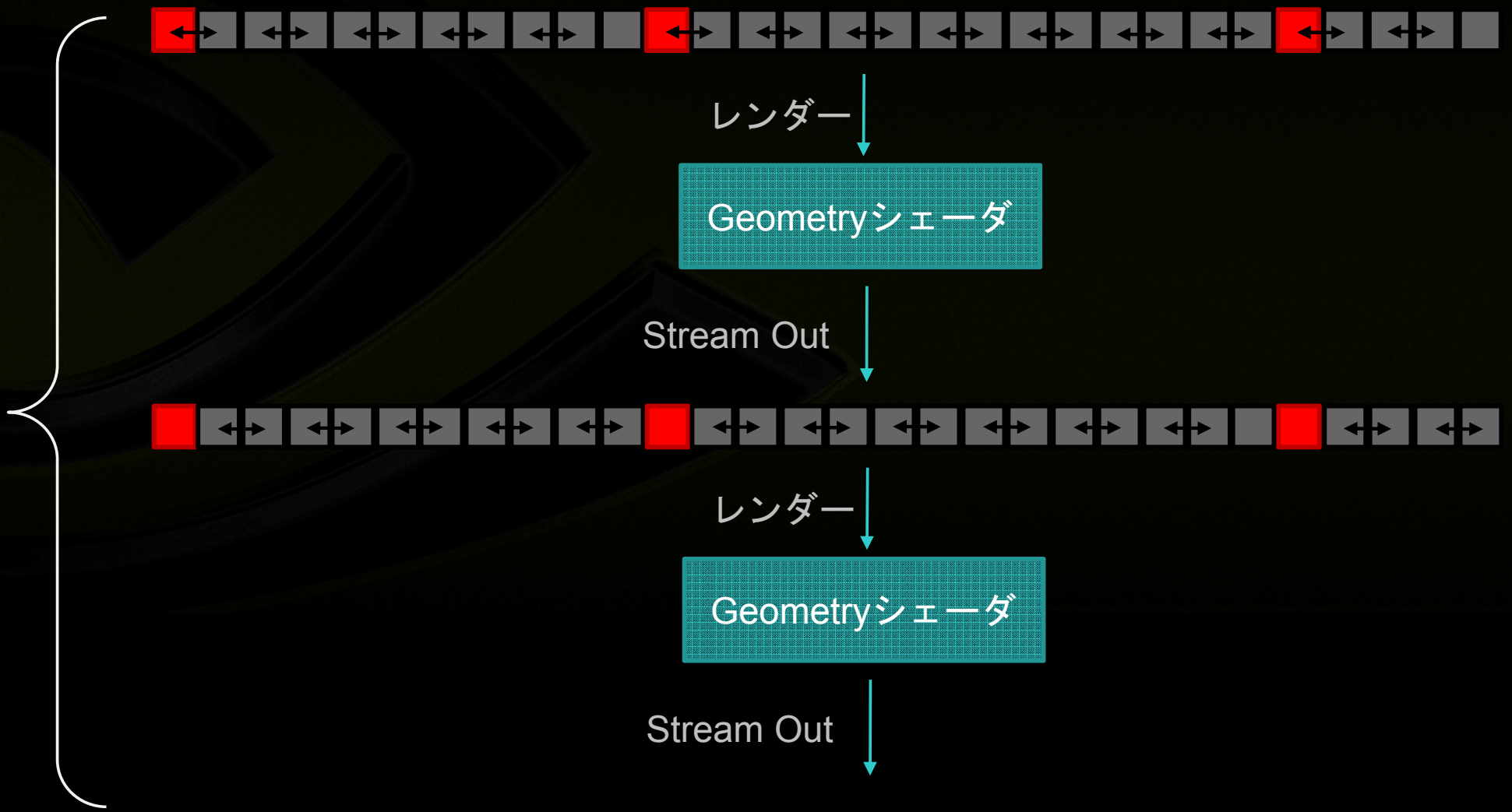
任意のパーティクルは複数の束縛制限下にあるので、いっぺんに並列処理することは出来ない

最初の束縛処理

次の束縛処理

繰り返し適用

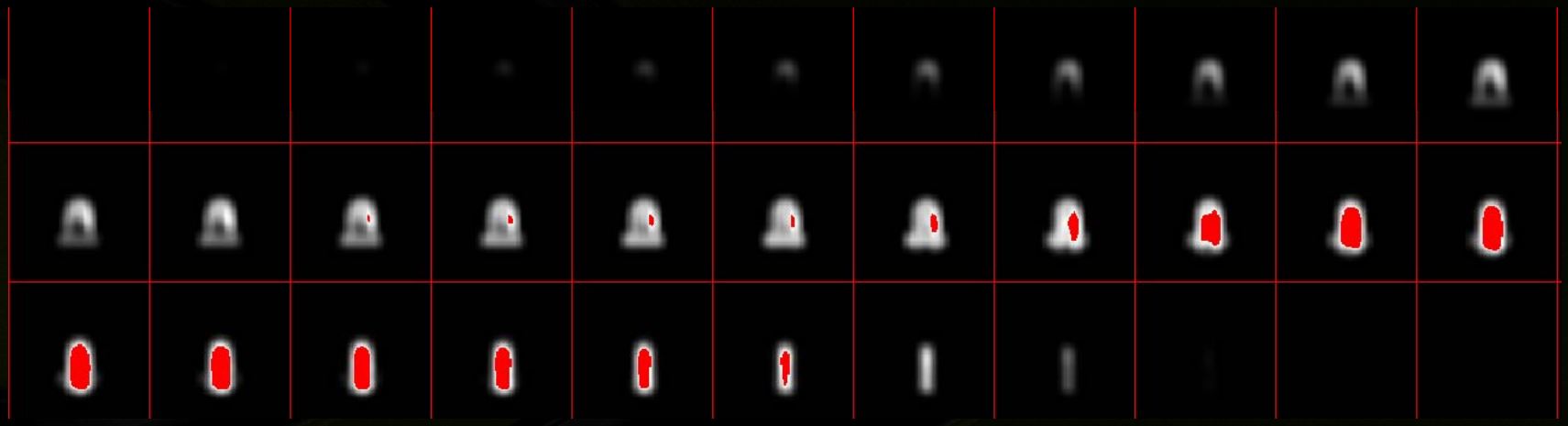
例: 距離束縛

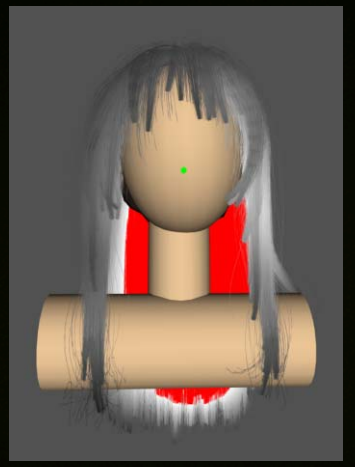

レンダー
Geometryシェーダ
Stream Out
レンダー
Geometryシェーダ
Stream Out

例: 髪の毛の衝突

- 髪の毛の衝突はグリッドベースな枠組みで処理
- 髪の毛と障害物(頭や体等)はLowレゾリューションなグリッドにボクセル化
- 髪の毛の頂点は高密度な領域へ押し出し

例: 髪の毛の衝突





- 力は密度の負勾配の方向に作用
  - ボクセル化された密度は不明瞭なので、この時点では各頂点は密度勾配で見つかった高密度領域へと落とされる
- この手法は髪の毛の衝突のThis approach tries to achieve volume preserving quality of inter-hair collisions

髪の毛の衝突

密度を可視化

Before

After

最終レンダリング結果

Before

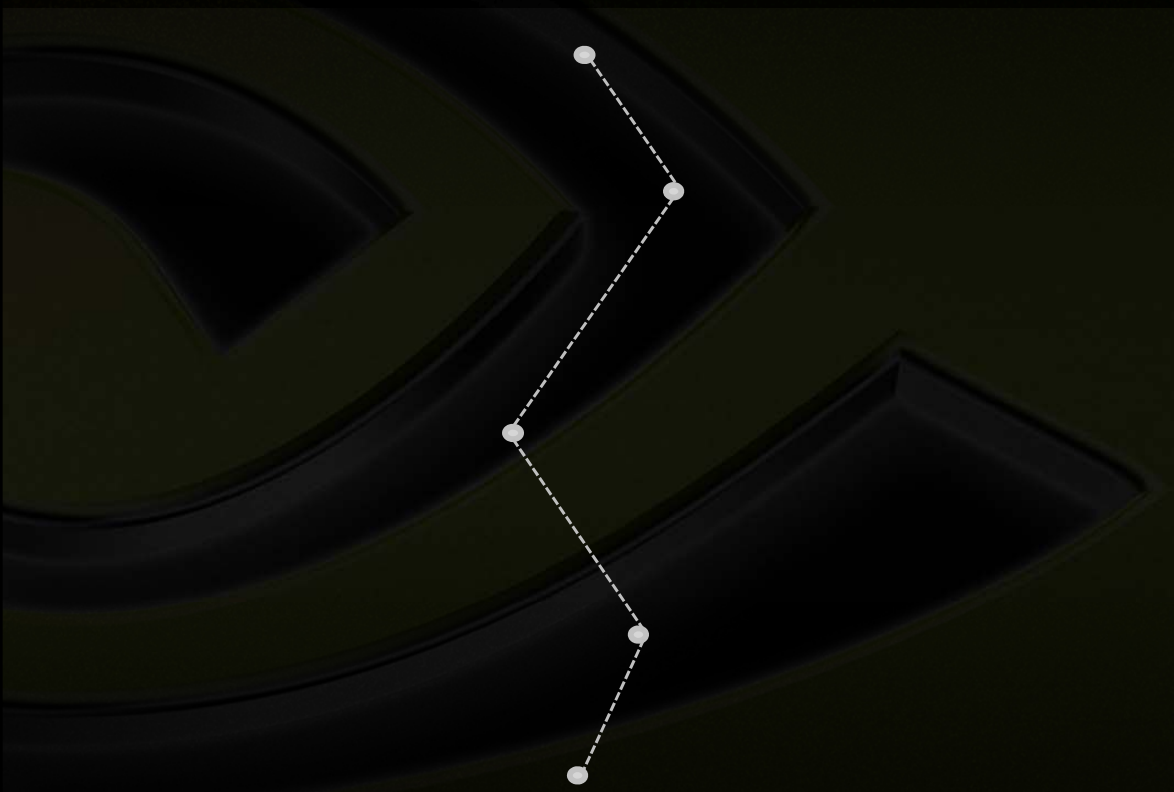
After

風の力

- 風力は粗いグリッドでのセミ・ラグランジュ流体シミュレーションを利用
- ボクセル化した髪の毛とメッシュも風に対する障害物としてグリッドに追加される

テセレーションと補間

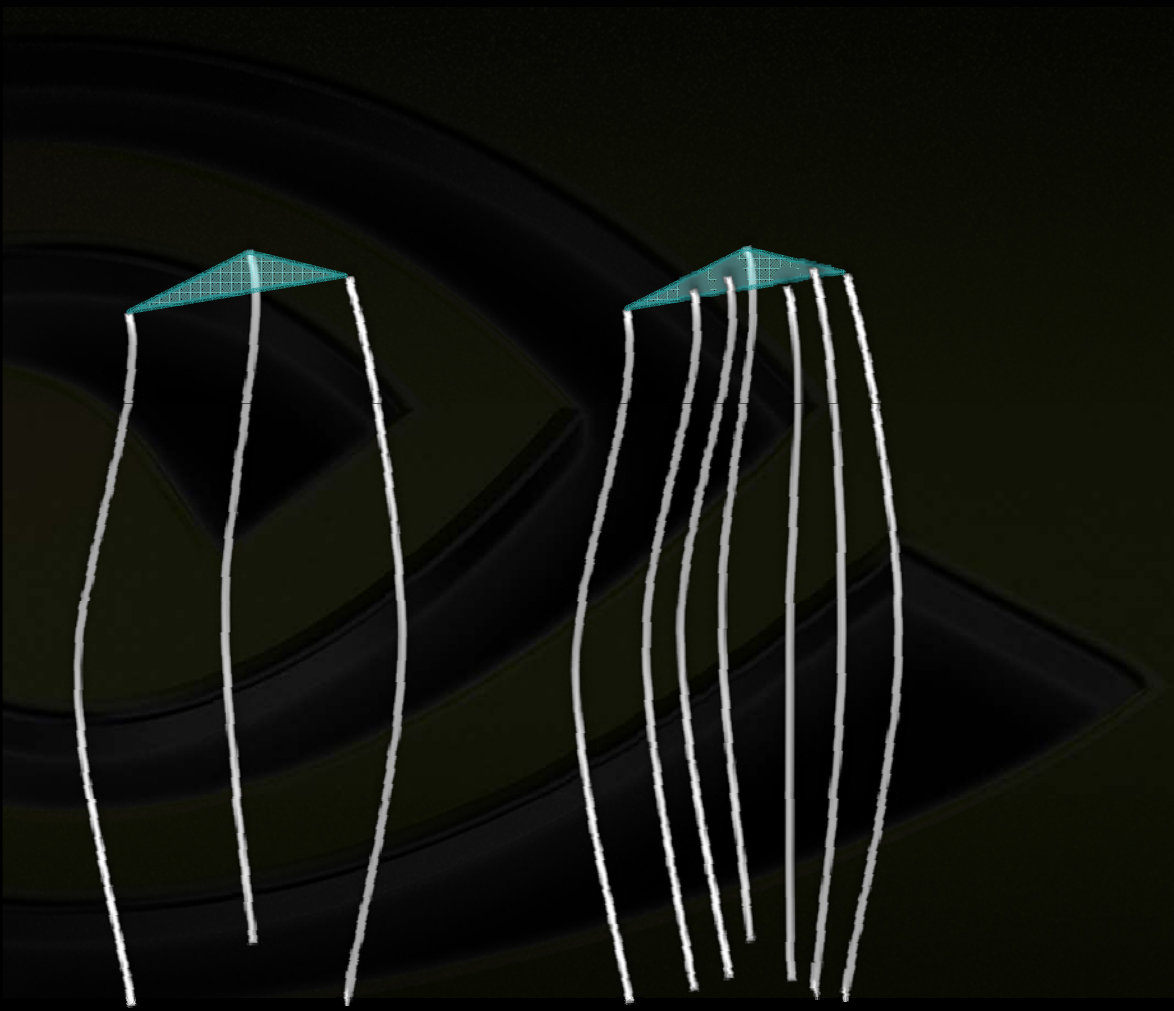
テセレーション



シミュレートされた頂点

滑らかにテセレートされた髪の毛

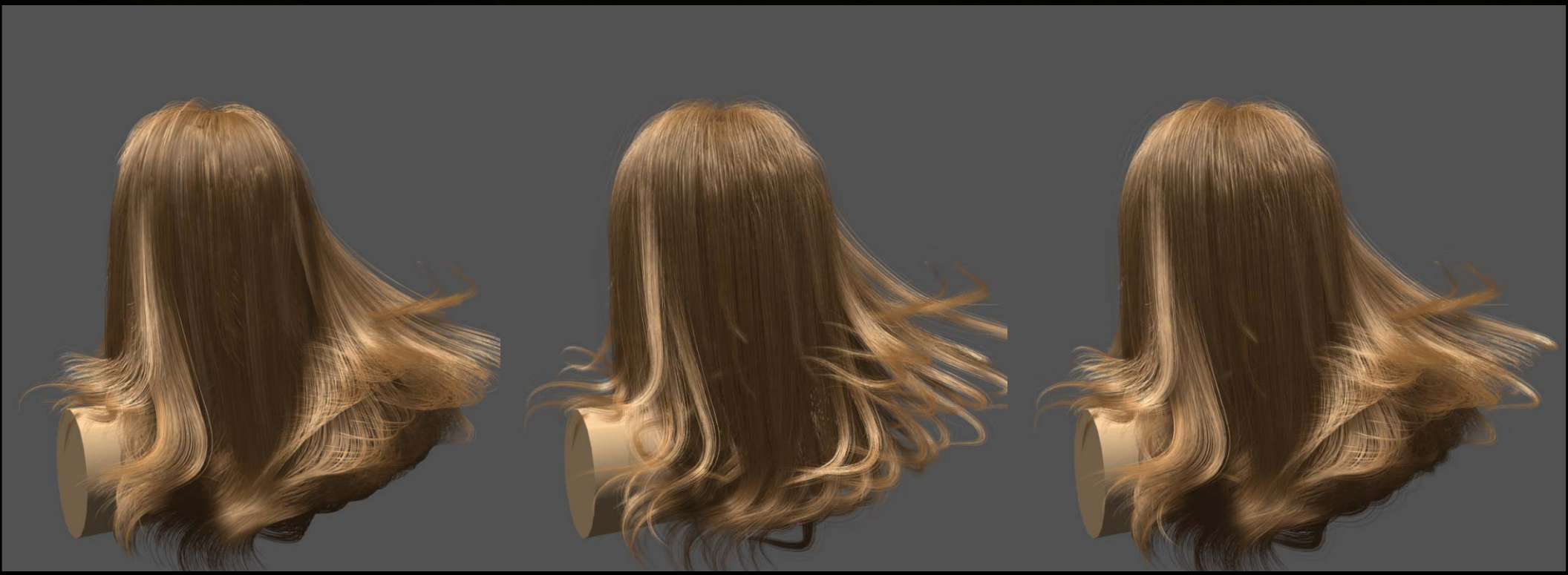
補間



マルチ・ストランド補間

クランプベース補間

補間



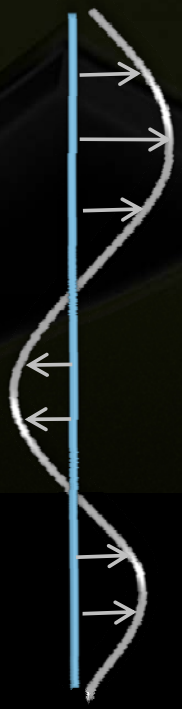
マルチストランド補間

クランプベース補間

コンビネーション

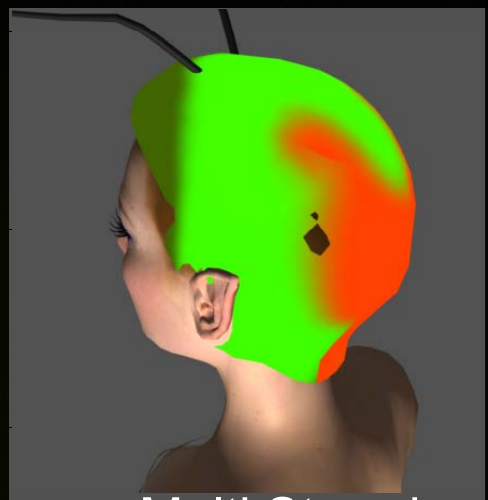
カーリーヘアー

- 定数バッファにカールのオフセットを符号化して追加
- クランプオフセットにこれらのオフセットを加算
- 手続き的に作成することも、デザイナーが基本となるカールオフセットを作ってそれを基に生成する事も可能

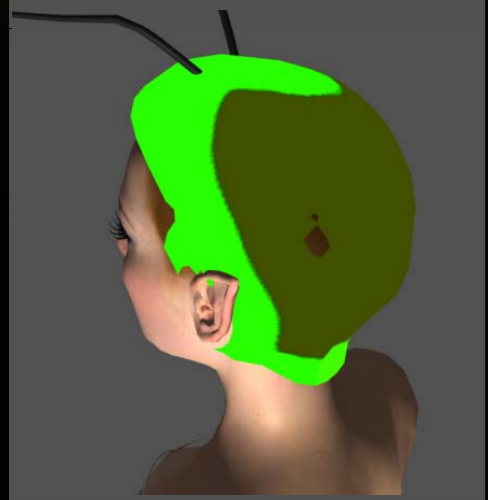


頭皮上の密度と厚さの調節

- Green: 髪の毛の局所密度
  - クランプベースの髪の毛では前頭部の近くではより密度が高い
- Red: 髪の毛の局所的な厚さ
  - クランプベースの髪の毛は前頭部近くで厚さが足りない



Multi Strand



Clump Based

プロセス

- テセレートされたダミーヘアをＮ回レンダリングする。Ｎは最終的な髪の毛の本数。

- バーテックスシェーダにて、シミュレートされたストランドの属性を保持しているバッファをロード
  - ストランドのテクスチャ座標や長さ、幅などの定数属性
  - 頂点位置や座標フレームなどの属性

プロセス

- 再計算を最小にするために各ステージ後にデータをStream out する
  - シミュレートされたストランドをテセレートし、Stream out
  - テセレートされたストランドを補間し、Stream out
  - Shadow mapへのシェーディング用に最終ヘアをレンダリング
  - 最終レンダリング用に最終ヘアをレンダリング
- 各ステージは前ステージで計算しStream Outされたデータを利用する

その他の最適化に関する詳細

- より良い結果を得るためにはAlpha to Coverageを使うと良いが、
  - Depthの読み書きのためにはearlyZを無効にしなければならない
  - Depthプリパス以前にearlyZを得るには、最終描画中depthテストのみを行い書き込むことができない
- ヘア・ストランドを作成するためにはGSを使わない
  - GSをラインやトライアングルを拡張するために使うことができるが、パフォーマンスはパイプラインの負荷に依存する

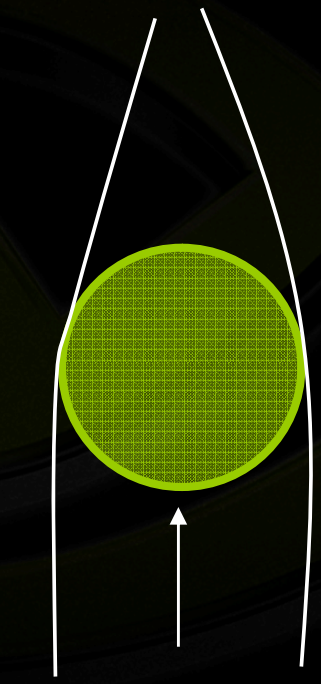
補間された髪の毛のコリジョン回避

- ガイド・ストランドを補間することによって障害物をすり抜けてしまう場合がある

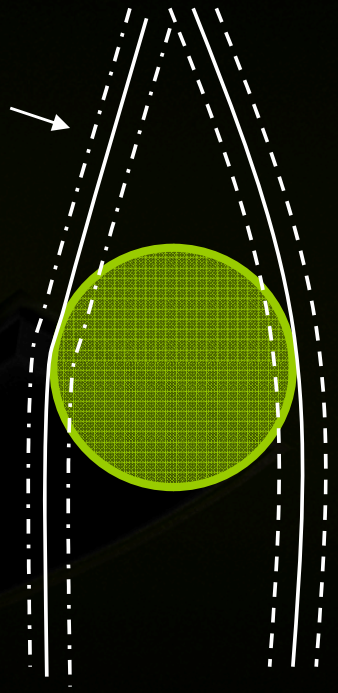
補間したヘアがコリジョンボリュームと交差

追加シミュレーション無しで衝突を修正

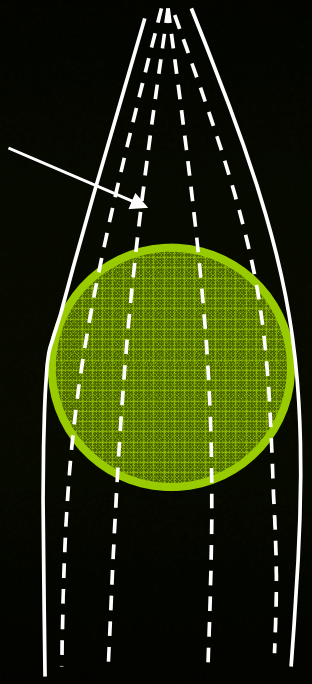
補間された髪の毛のコリジョン回避



シミュレーションされたヘア

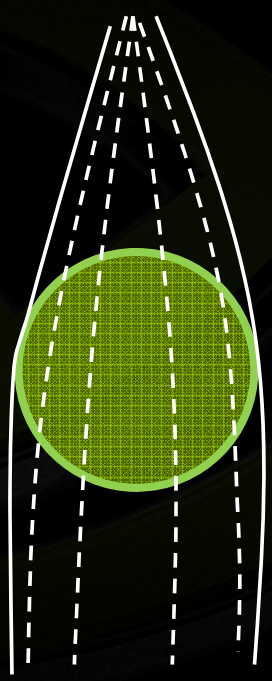


シングルヘア補間

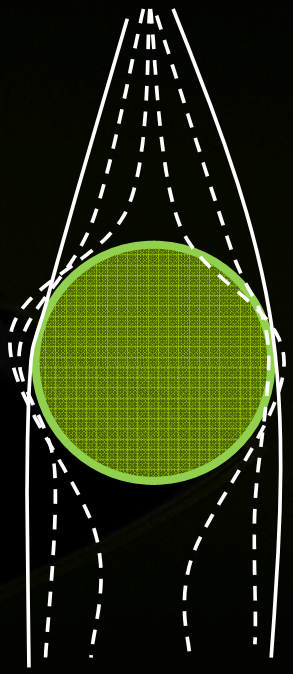


マルチヘア補間

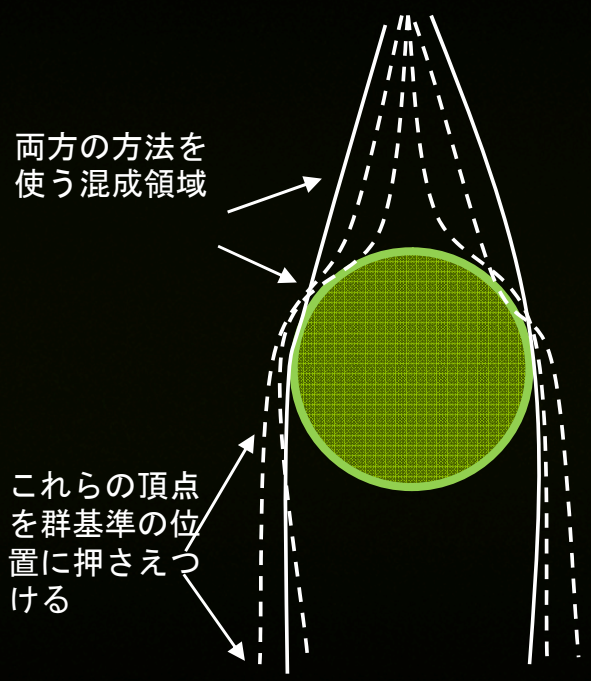
補間された髪の毛のコリジョン回避



衝突回避無し

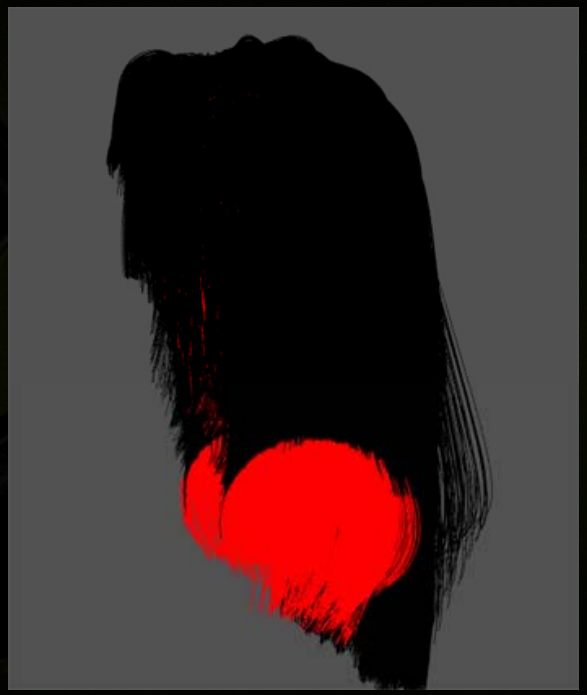


貫通している頂点のみの修正



私たちの手法

補間された髪の毛のコリジョン回避

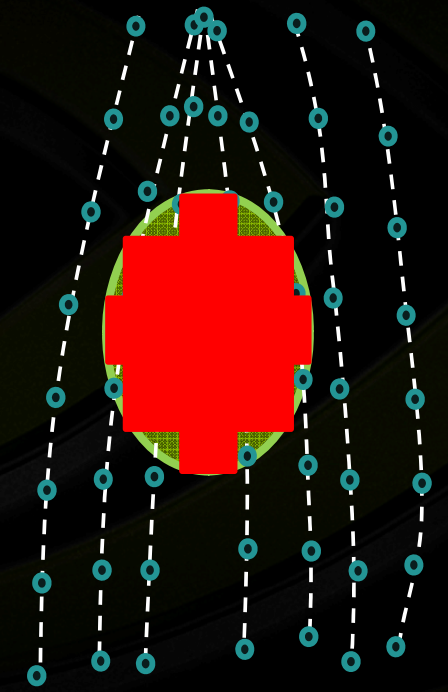


補間された髪の毛のコリジョン回避

各ヘア・ストランドをーピクセルへレンダリング

該当頂点が衝突していた場合は頂点IDを出力

最小限の混成を使用

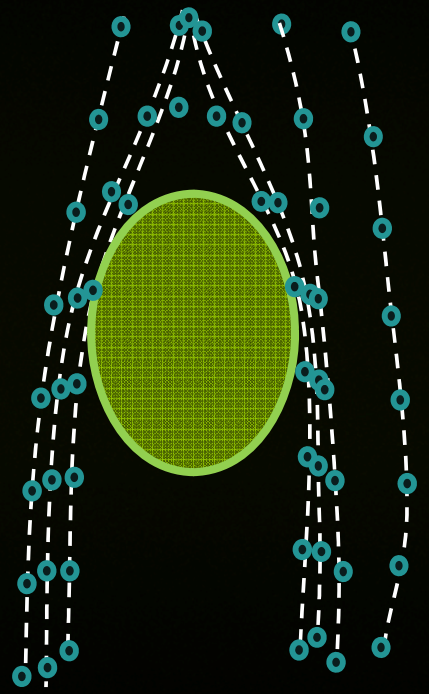


∞	3	2	2	3	∞	∞

パッチ毎の補完される髪の毛の最大数

パッチの数

∞	3	2	2	3	∞	∞



レンダリング

レンダリング

- ラインの問題点:
  - 浮動小数点数な幅を持たない
  - ラインをまたがったテクスチャがない
    - 多量の髪の毛の見た目をシミュレーションするのに役立つ
    - 複雑なカラーバリエーションでレンダリング
- トライアングルストリップに向いたカメラをレンダリング
  - ラインをストリップにGSで拡張することでも可能
  - それか、インスタンス化されたトライアングルストリップをレンダリング

シェーディング: Kajiya and Kay

- **Kajiya and Kay** ***[Rendering fur with three dimensional textures*** **(SIGGRAPH '89)]**

- **Diffuse = sin(T,L) = sqrt(1 – T . L^2)**
  Specular = [T . L * T . E + sin(T,L) sin(T,E)] ^{p}
  = [T . L * T . E + sqrt(1 – T . L^2) sqrt(1 – T . E^2)] ^{p}

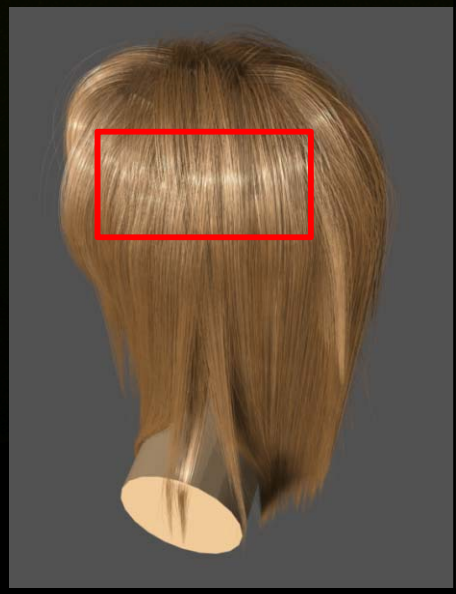
- **Ivan 2006**
- **２重スペキュラハイライトの偽装**
  - 主要なハイライトを先端方向へずらす
  - セカンダリハイライトを根元方向へずらす

接線

- スムーズな接線が必要
  - テセレートされ補完された接線を計算

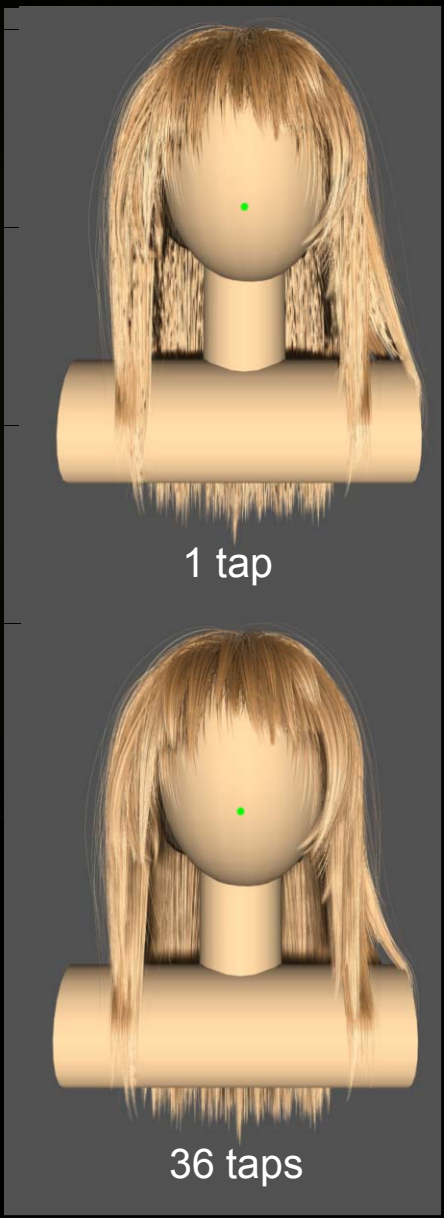
$$x(t) = [T_0 \quad T_1 \quad T_2] \begin{bmatrix} 0.5 & -1 & 0.5 \\ -1 & 1 & 0.5 \\ 0.5 & 0 & 0 \end{bmatrix} \begin{bmatrix} t^2 \\ t \\ 1 \end{bmatrix}$$

- 強すぎるハイライトをなくすためにジッタを接線に加算する
  - 定数バイアスを接線に対して根元方向ないしは逆方向にランダムに
  - 接線にピクセル毎のノイズを加算



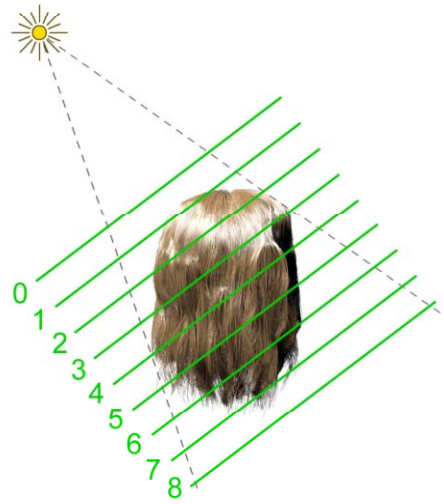
影

- マテリアル・モデル: 不透明ヘア
- 本質的条件
  - フリッカーなし、スムーズな影
  - ソフト・シャドウ
- マルチサンプルでPCFを
  - tShadowMap.SampleCmpLevelZero(ShadowSampler, texcoord, z, int2(dx, dy));
  - 側頭部や空間的なエリアシングを減らすのに役立つ
  - VSでシャドウの計算をして髪長全域で補間する事でよりエリアシングを減らせる

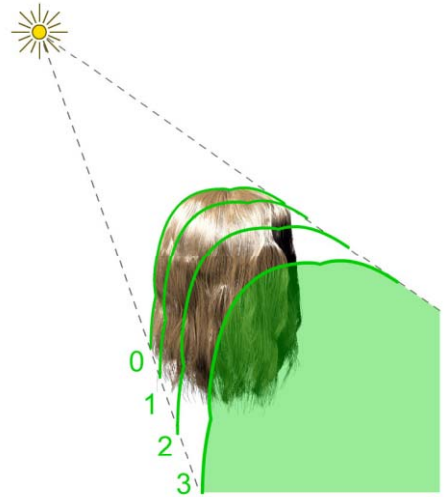


影

- マテリアル・モデル: 透明ヘア
- 髪の毛が半透明の場合ボリューメトリックシャドウが必要
  - [Yuksel and Keyser 08], [Kim and Neuman 01], [Lokovic and Veach 01]
  - 空間をレイヤーに分離



不透明シャドウマップ



ディープシャドウマップ

[Images courtesy of Yuksel 08]

影

- 重み付きPCF取り入れる
  - [Halen 06] と同様
  - 重み付きPCF は次式でサンプルされる
    1 - exp(g_SigmaA * d)
    - d は現在のシェード位置とライトから最も近い位置との差分

アンチ・エリアシング

- 人間の髪は非常に細い
- 例によってアルファブレンディングがエリアシングを隠すのに使われる
  - 時間を浪費してしまうジオメトリのソートが必要
  - デプス・ピーリングが使える [Everitt 01], [Bavoil and Myers 07]
    - 拡張可能: 例えば最初の４つのデプスレイヤだけをレンダリングするようにできる
  - それか [Sintorn and Assarsson 08]

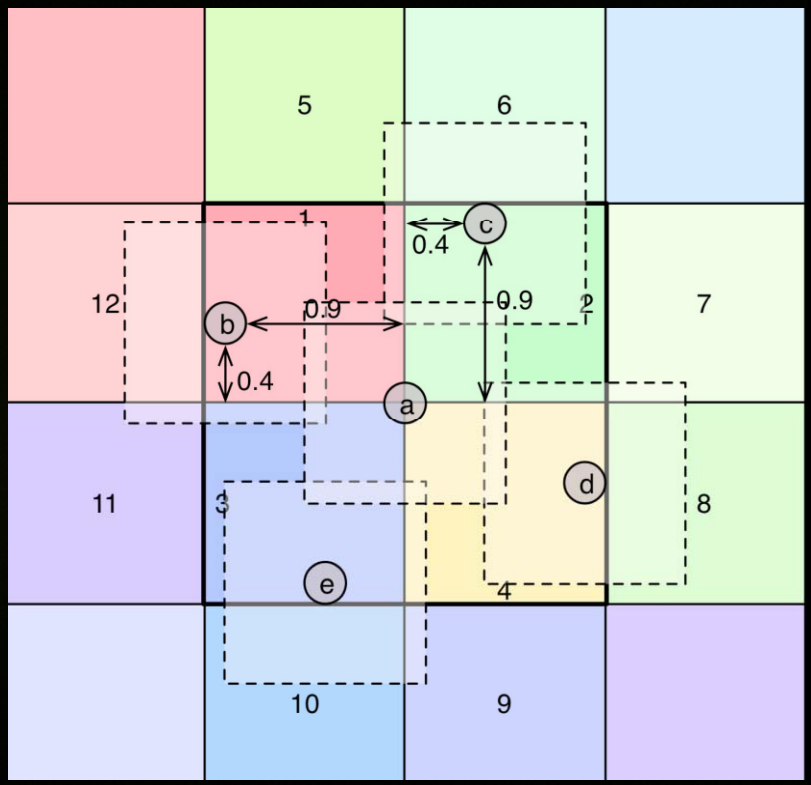
アルファブレンディング

アルファブレンディングお

[Sintorn and Assarsson 08]

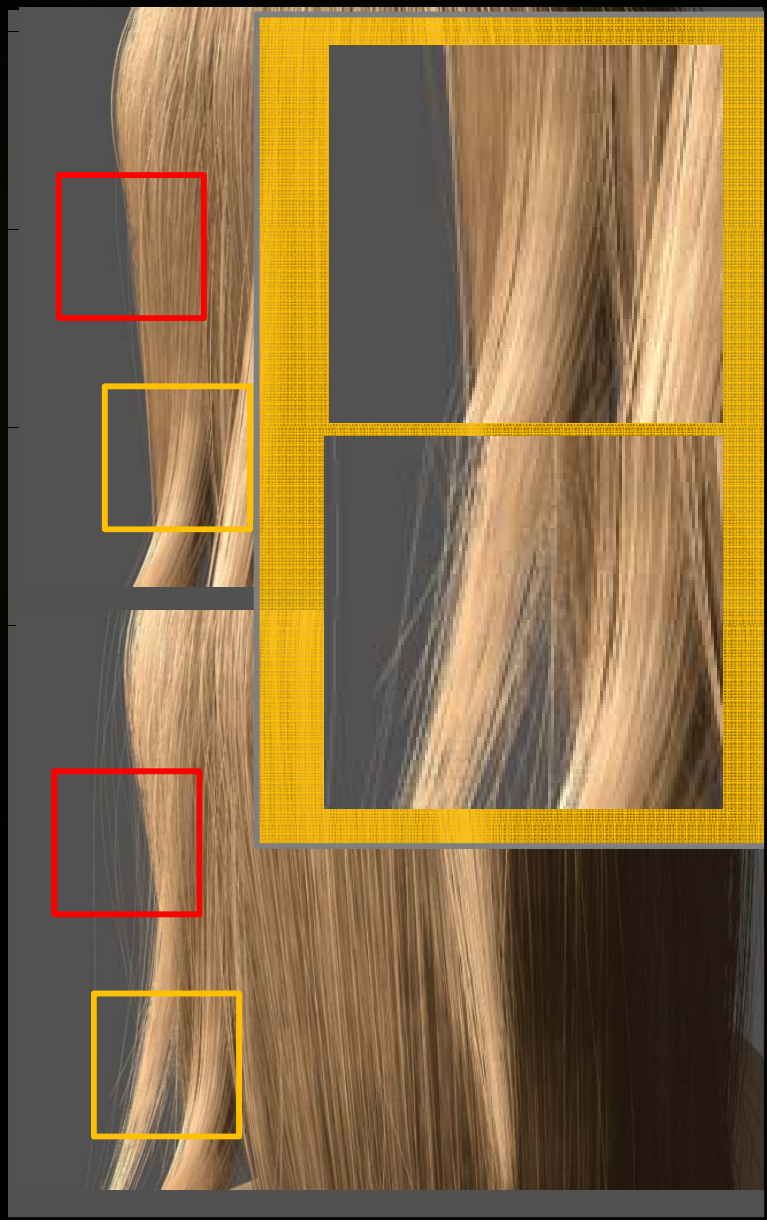
アンチエリアシング

- Alpha To Coverageも使える
  - ソートが必要でない
  - MSAAが必要
  - earlyZを得るためにデプスプリパスが必要
- MSAAとSSAAのコンビネーションが使用可能
  - 8xMSAA
  - 5サンプリングの2xSSAA



ランダムな偏差を髪に加える

- 髪全体に渡って補間オフセットに加える偏差をあらかじめ作って保持しておく
  - 大抵の髪は先端で逸れる
  - 少量の細い髪の毛は非常に大きく逸れる
- その他
  - 髪の毛の先端で幅が細くなる
  - ヘア・ストランド毎にランダムな幅

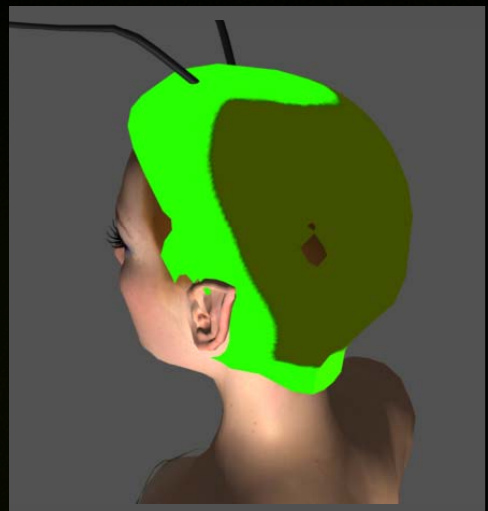


LOD

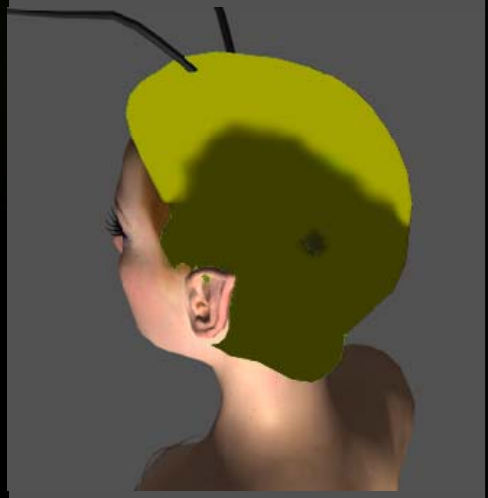
- LODを決めるために投影されたセグメントのサイズを使う
  - Low LOD レベルは少ないライン数、少ないセグメント、細いラインを使うかもしれない

LOD

- デザイナーが決めたLODを使うこともできる
  - ヘアスタイルに応じてデザイナーは違うLODを違う見た目のために作ることが出来る
    - 例えば、低LODでは頭頂部に大きなストリップを持つようにすることが出来る
  - これらのLODはテクスチャに書き写すことも出来る
  - Hullシェーダはラインの密度と厚さを得るために適切なLOD textureでlerp()する



Near



Far

Thank you!

Thank you!

10:40 - 12:00
Displacement Subdivision Surfaces in DX11
Takayuki Kazama

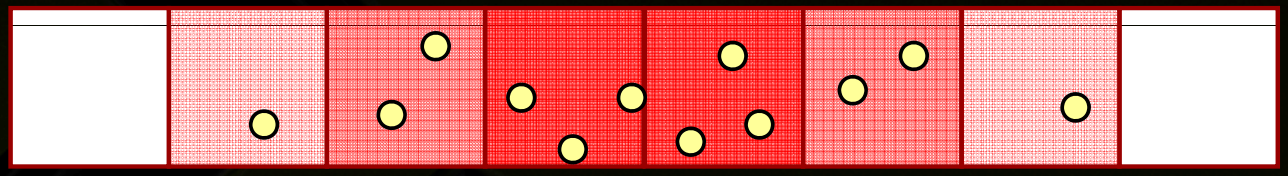
13:00 - 14:20
GPU Physics and CUDA
Kitty Vongsay

14:50 - 16:10
Real time Hair Rendering
Bryan Dudash

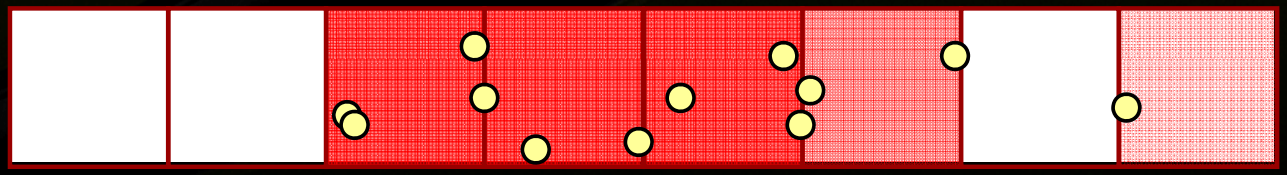
16:40 - 18:00
Horizon Based Ambient Occlusion
Bryan Dudash

developer.nvidia.com

Bertailsとの比較



•*The original hair vertices and corresponding cell densities*



•*New hair vertex positions and cell densities after applying fixed magnitude local collision forces*