Prismatic Solidにおける XBOX360 活用事例

YO1 KOMORI GAMES

小森　陽一
(ゲームプログラマー)

概要

- XBOX360開発環境
- Prismatic Solid
  - ベベルシェーダ
  - ライトストリング
  - スフィアシェーダ
  - カプセルシェーダ
  - パーティクルシステム
  - バレットシステム
  - コリジョンマップ

XBOX360開発環境

- XNA Framework
- 個人でXBOX360で動くプログラムの開発が可能
- 開発ツールは無料
- プログラムをXBOX360で動かすのに１万円/年
- 開発言語はＣ＃のみ。C++/VB不可。
- Compact .NET Framework

Prismatic Solid

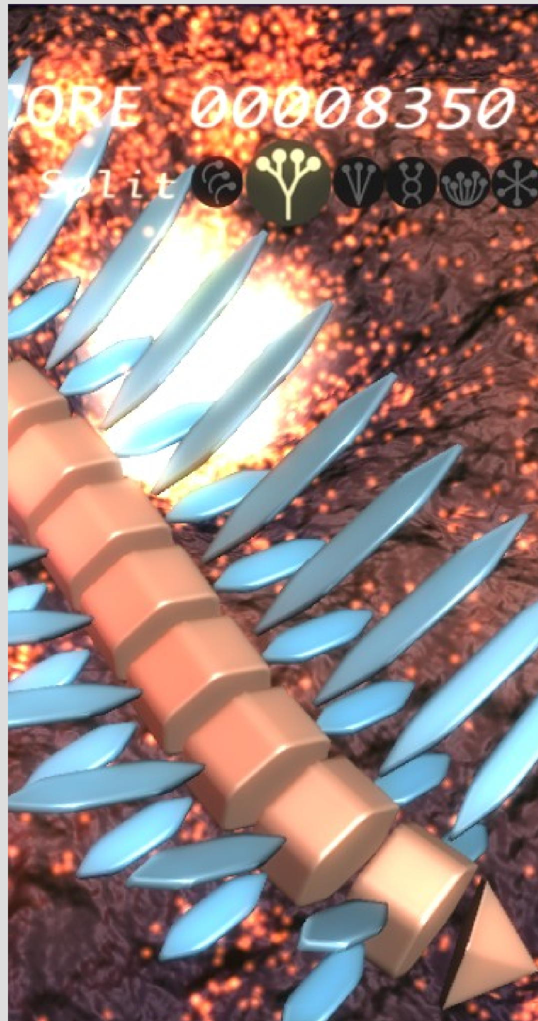
- XNA上で開発されたシューティングゲーム
- Xbox LIVE Indie Games にて80MSPで配信中
- Dream-Build-Play 2010にて2nd Prize受賞

XBOX LIVE
indie games

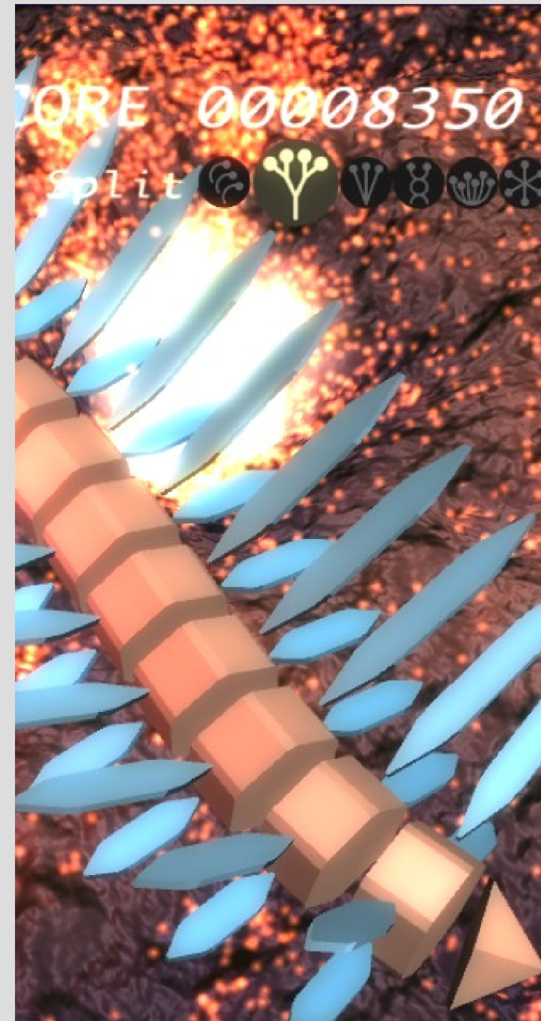
ベベルシェーダ

- 日本語では「面取りシェーダ」
- エッジを斜めに削る加工法
- 質感向上、エッジにハイライトとアウトライン効果

ベベルシェーダの例

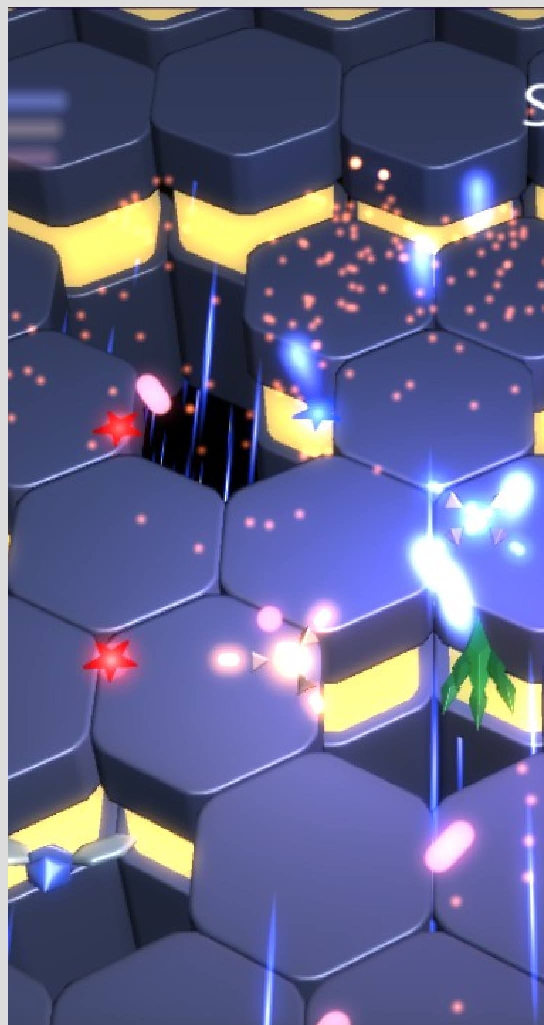


ベベルシェーダあり



ベベルシェーダなし

ベベルシェーダの例

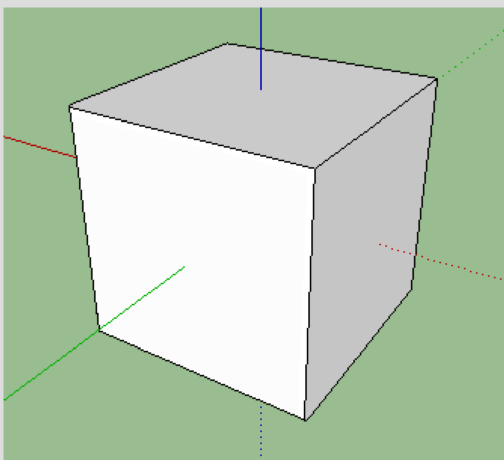
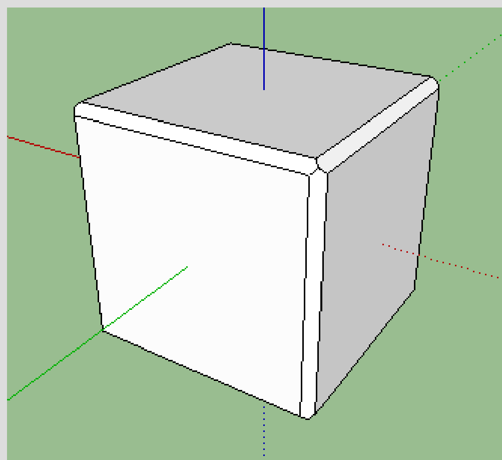
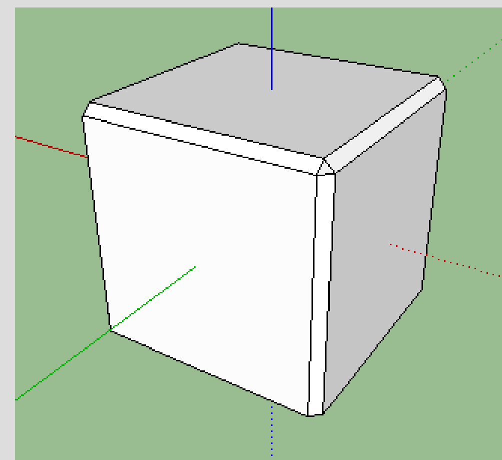


ベベルシェーダあり

ベベルシェーダなし

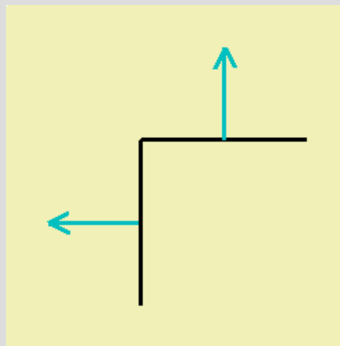
正しい面取り手法

- エッジにポリゴンを追加する
  - モデルデータ作成時に追加
    - ３～４倍のポリゴン数の増大。
  - ジオメトリシェーダで生成
    - XBOX360にジオメトリシェーダはない。

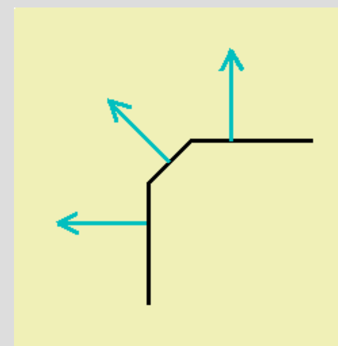




擬似的面取り手法

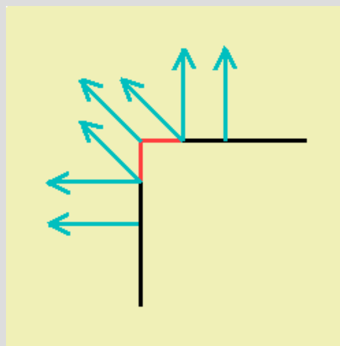
- ポリゴンは変更しない
  - 擬似的にエッジの法線を変更する



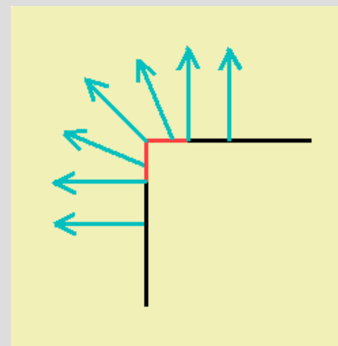
面取りなし



エッジにポリゴンを追加



エッジ付近の法線を変更



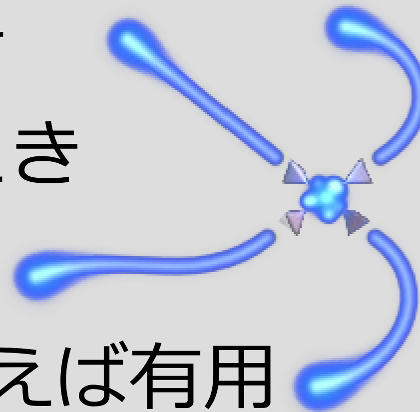
滑らかに変化させれば丸角に

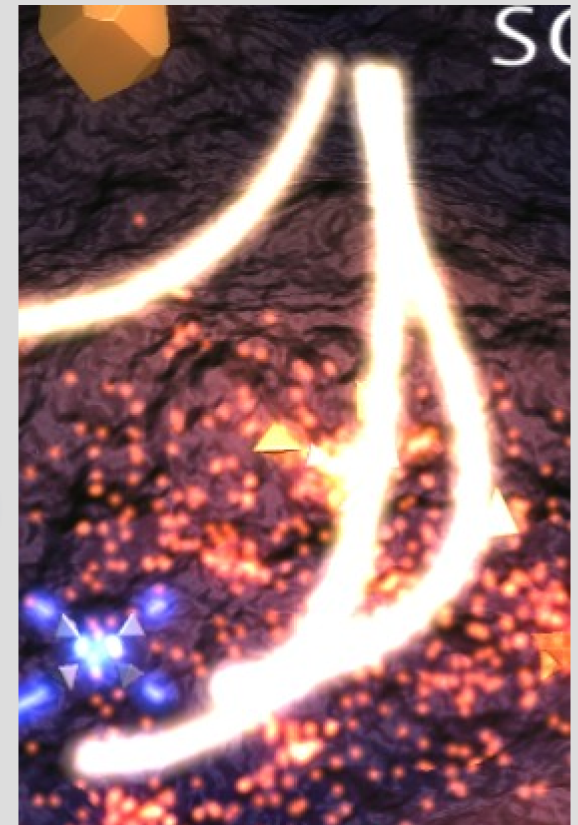
ベベルシェーダの実装

- ピクセルシェーダでエッジ近辺の法線を変更
- ピクセル近傍の辺情報が必要
- 理論上無限大数の辺が近傍にありうる
- ５つの辺に限定する
- 頂点付近でアーティファクトが発生しうる
  - 拡大しなければクォリティ的に問題ない
- 辺情報を事前に生成し頂点情報に追加
  - 辺の方向と辺の向こう側の面の法線

ライトストリング

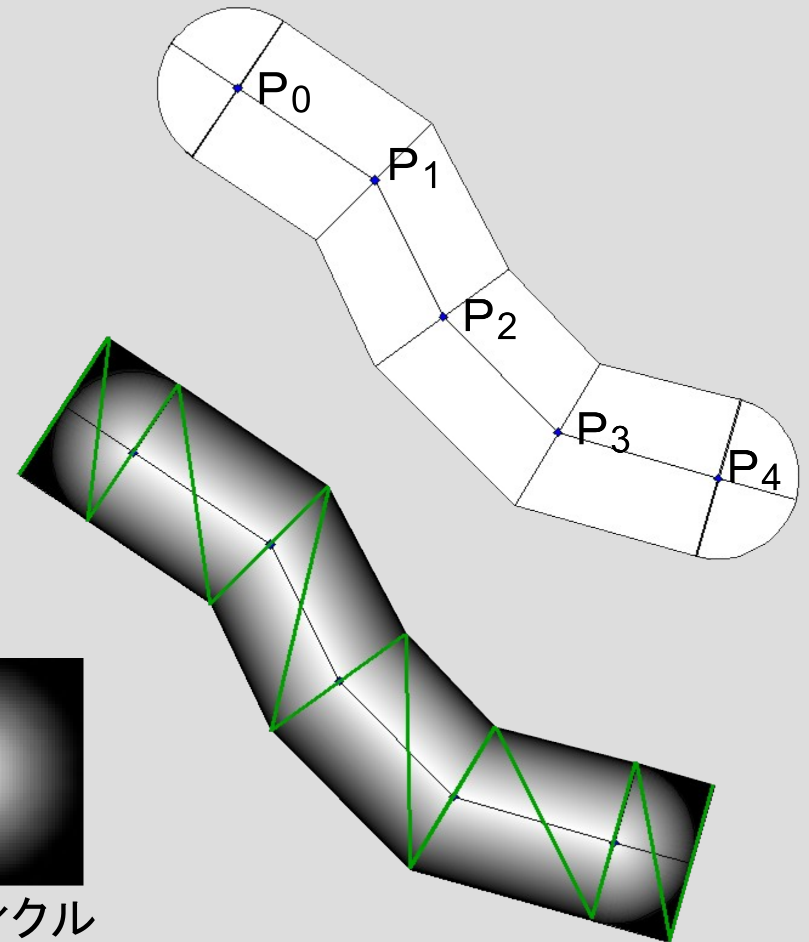
- 光の紐
- 使用テクスチャ
- スクリーン空間２Ｄポリゴン生成
- 擬似的ゆえのアーティファクト
  - Z方向に伸びて
    パースがきついとき
  - 曲率が小さすぎるとき
  - 幅が太すぎるとき
    - 特性を理解して使えば有用

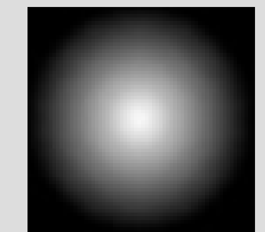




ライトストリングの生成

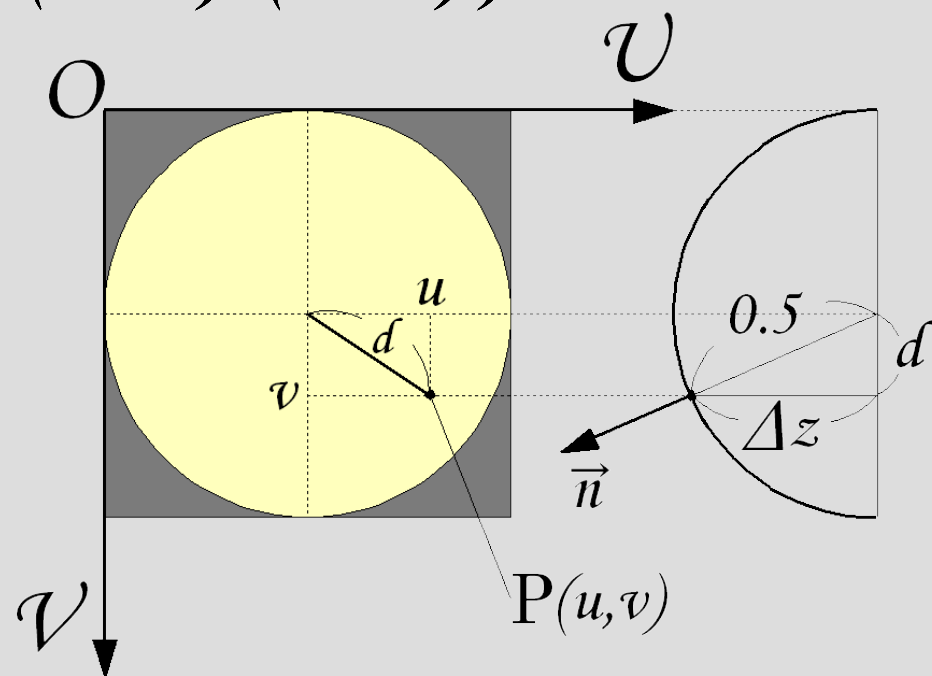
- 頂点シェーダ
  - 頂点列P_0~P_nから TriangleStripを生成
  - 透視変換後のスクリーン座標系
- パーティクル用のテクスチャを引き伸ばして適用




パーティクル
テクスチャ

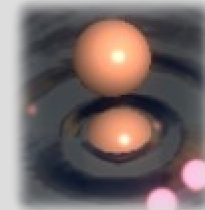
スフィアシェーダ

- ポイントスプライト
- 座標P(u,v)でのＺ補正量を計算
  - ＵＶ平面上で中心からの距離＝ｄ
  - $d = sqrt((u-0.5)*(u-0.5) + (v-0.5)*(v-0.5))$
  - $\Delta z = sqrt(0.5*0.5 - d*d)$
- dが0.5以上ならクリップ
- 法線を生成
  - $n = (u-0.5,v-0.5,\Delta z)/0.5$



ポイントスプライト

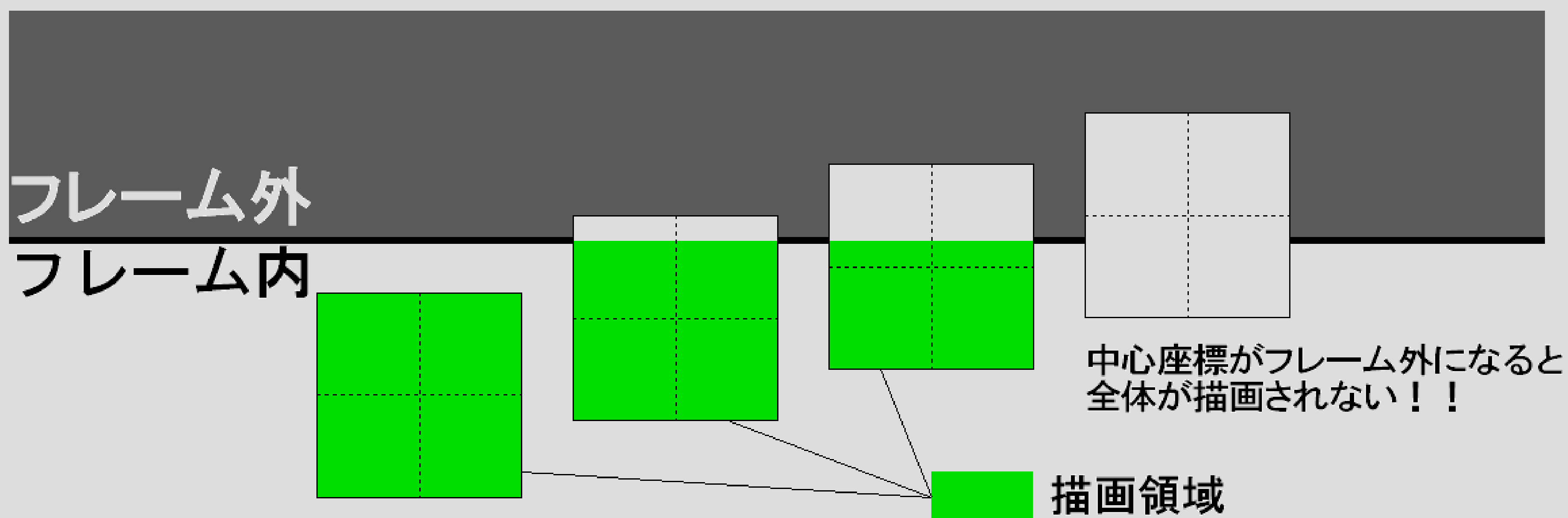
- 正方形のプリミティブ
  - 中心座標とサイズ情報を持つ



- 利点
  - １つの頂点情報で描画可能
    - TriangleStripの場合４頂点＋縮退ポリゴン用に２頂点で合計６頂点必要
- 欠点
  - 最大サイズに限界あり (最大64x64)
  - 画面端でシザリングされない

ポイントスプライトの画面端問題

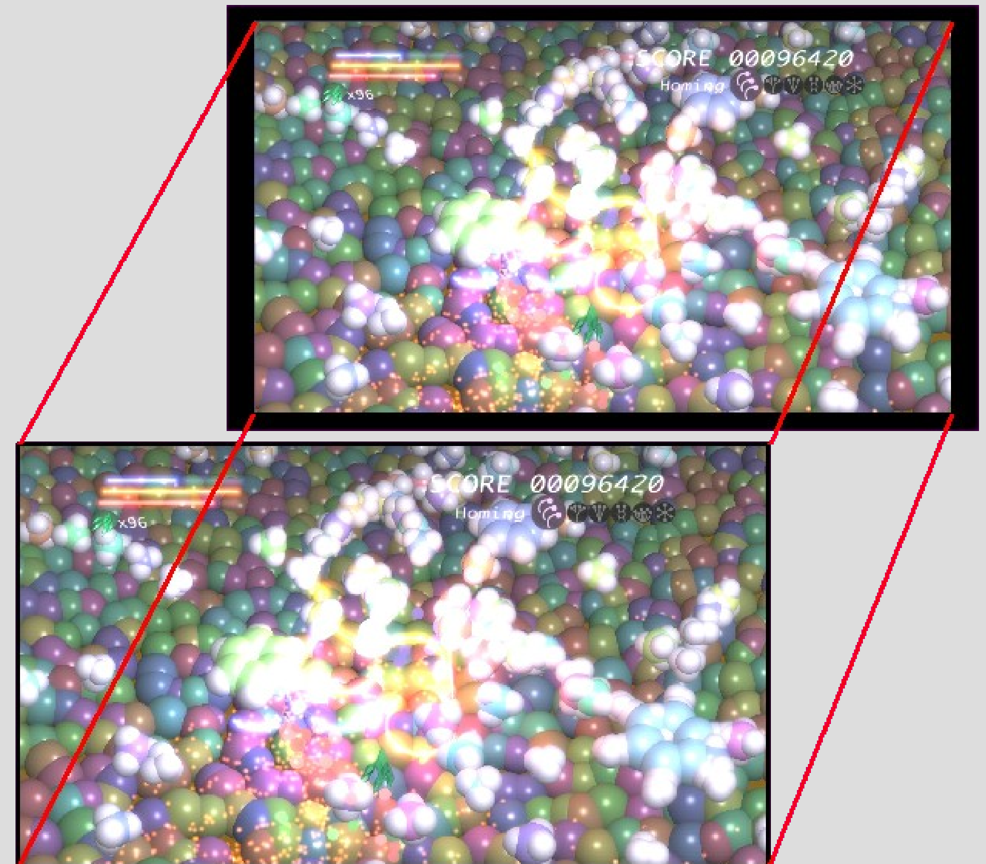
- 中心座標が画面外に出ると全体が描画されない



ポイントスプライトの画面端問題解決方法

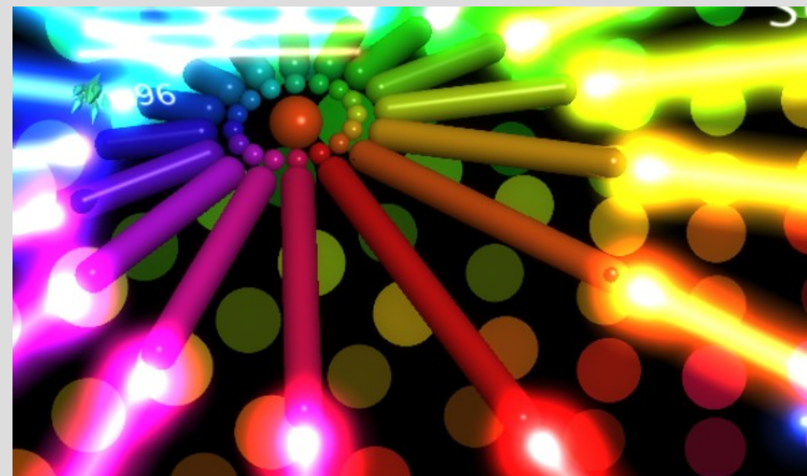
- ポイントスプライトの最大サイズは64x64
- 描画フレームの端に最低32ピクセルの余白
- 表示領域は余白の内側を拡大して使用

- 副次的ジャギ低減効果



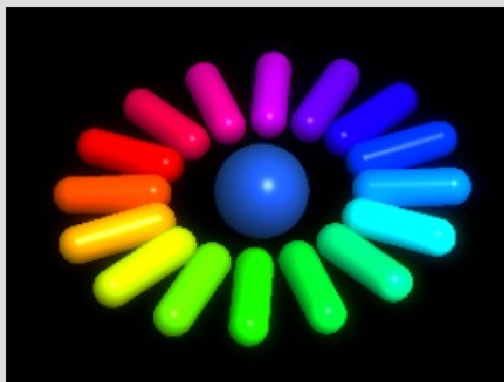
カプセルシェーダ

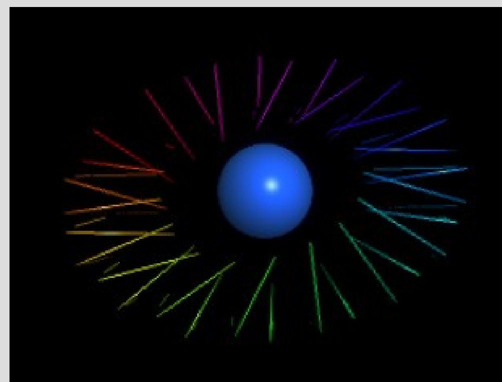
- カプセル形状
  - 円柱と２つの半球の組み合わせ
  - 平面で区切られた二次曲面の集合
  - 座標２つと半径で定義可能
- 形状をポリゴンで表現せずレイキャストで実現



カプセルシェーダの実装

- 頂点シェーダ
  - 形状を覆うスクリーン空間上の２Ｄポリゴンを生成。ライトストリングと同様のポリゴン生成
- レイキャスト（ピクセルシェーダ）
  - 視線ベクトルとカプセル形状との交差判定
  - 交点と法線を求めてシェーディング





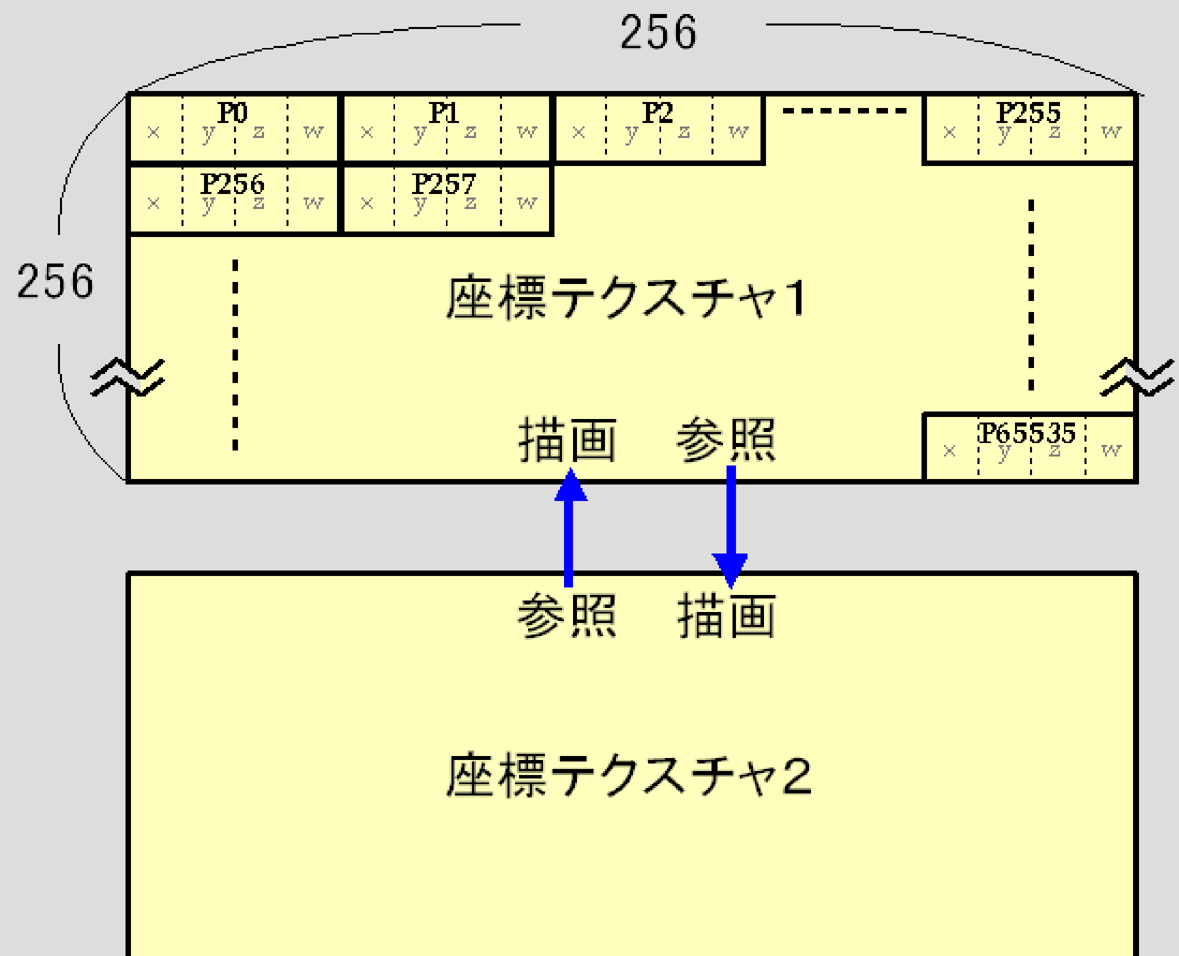
ワイヤーフレーム表示

パーティクルシステム

- ポイントスプライト
- 座標と速度をFloat16のテクスチャに保存
  - 描画で更新
  - ピンポンバッファ
  - 表示時は頂点シェーダから参照
- 地面に当たって跳ね返る
  - ハイトマップテクスチャを参照
- 最大６５５３６個 (256×256テクセル)

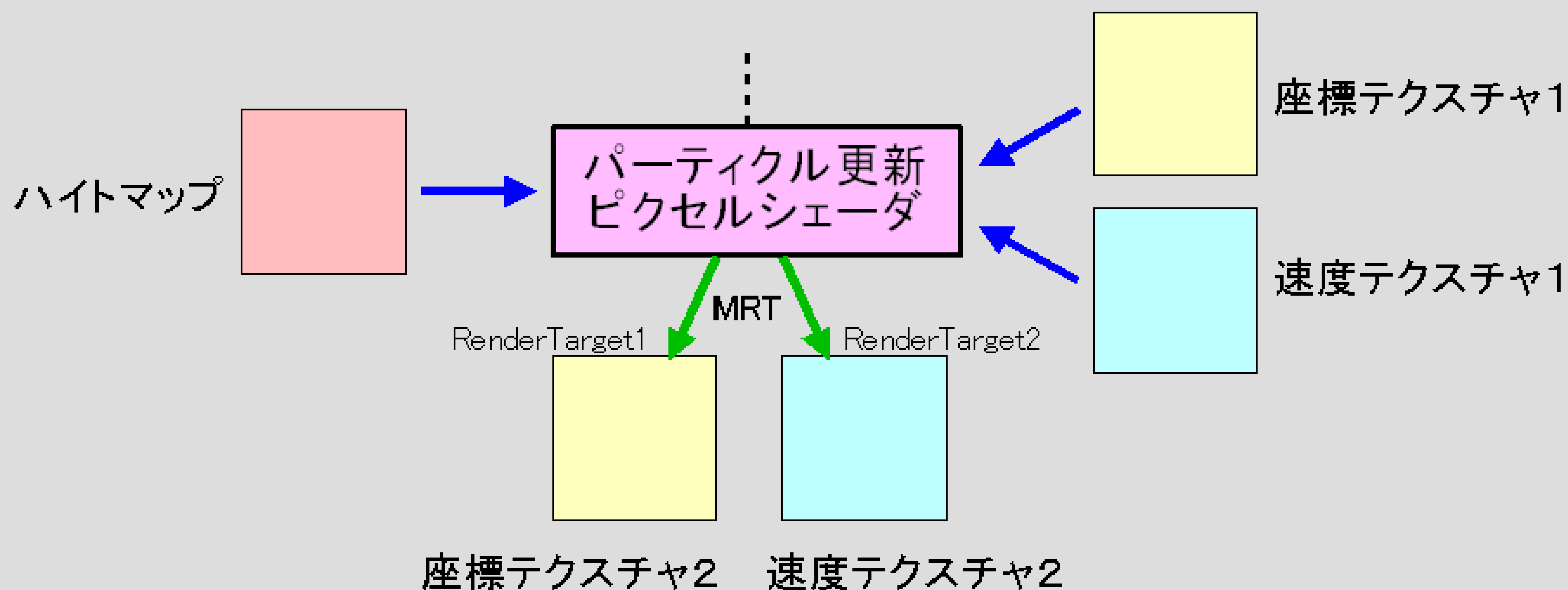
パーティクルの座標と速度

- Float16x4テクスチャに座標と速度を保存
- ダブルバッファ
  更新毎にレンダーターゲットと参照テクスチャが入れ替わるピンポンバッファ



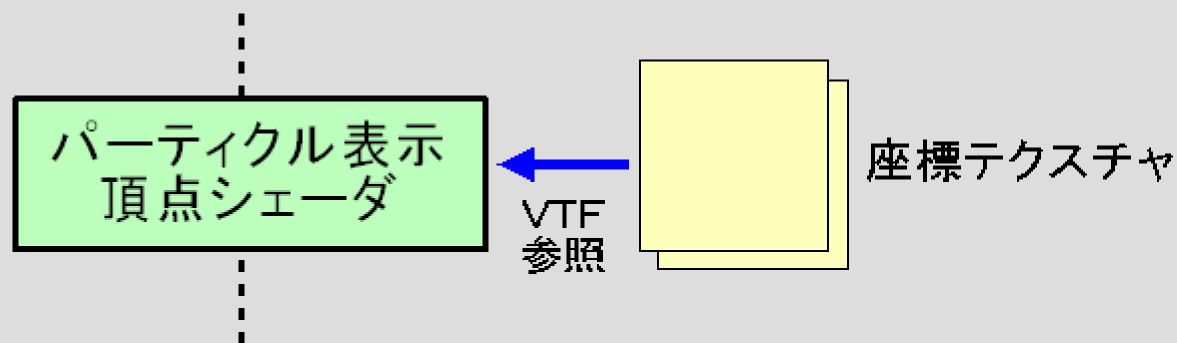
パーティクルの更新

- ピクセルシェーダで更新
- マルチレンダーターゲット
- パーティクルの生成も同様、テクスチャに描画



パーティクルの表示

- 頂点シェーダで座標テクスチャを参照
- ポイントスプライト
  - 入力頂点は１個
- 座標のw成分に寿命を保存
  - 寿命に近づくとフェードアウトして消える

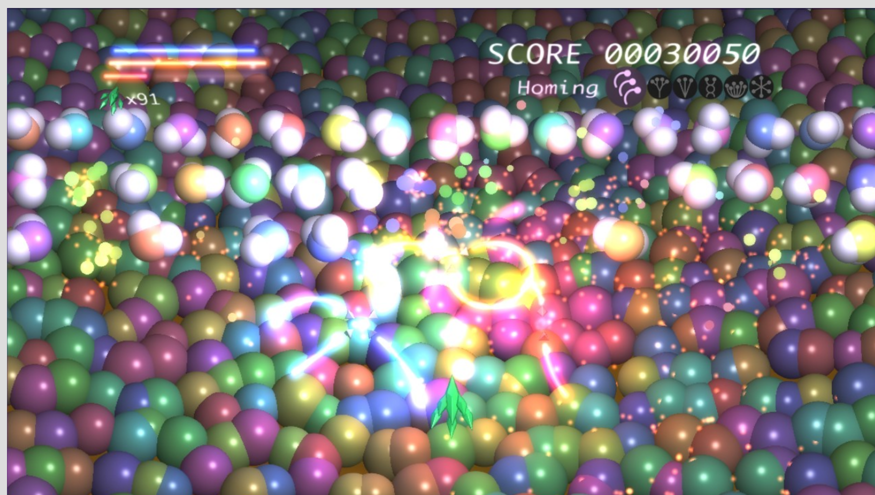


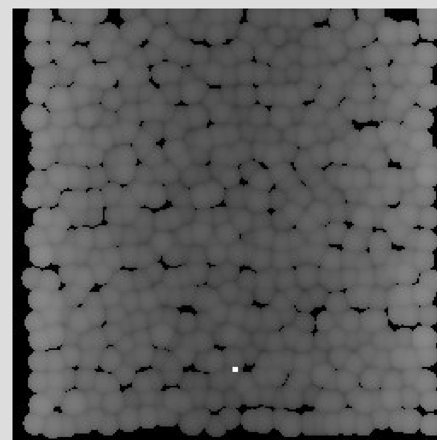
バレットシステム

- パーティクルシステムの変形拡張
- ２次元座標(Y軸=0)
- 当たり判定付き
  - 自機
  - ハイトマップ
  - コリジョンマップ
- 最大１６３８４個 (128×128テクセル)
- 画面内で消えない程度に寿命を設定。

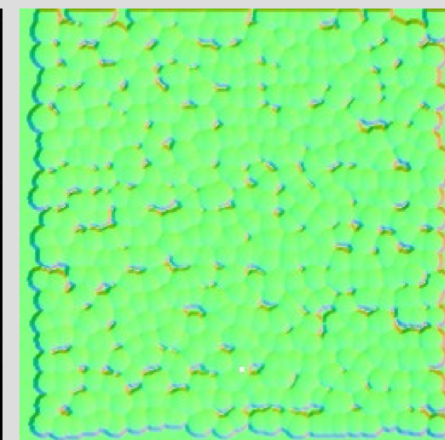
ハイトマップとノーマルマップ

- 地形のハイトマップとノーマルマップを動的に生成
- Ｙ軸平行投影でシェイプを描画
- ハイトマップからノーマルマップを生成





ハイトマップ



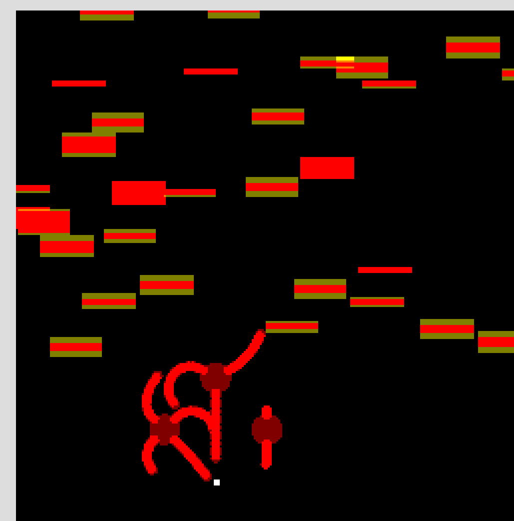
ノーマルマップ

コリジョンの通知

- ＧＰＵで判定したコリジョン結果をＣＰＵに通知する必要がある。
- 描画したテクスチャを直接読む手法
  - テクスチャのロックが必要。ロック可能になるまでブロックされるのでスレッド化またはマルチバッファ化すれば可能かもしれない。
- 今回採用した手法
  - OcclusionQueryを使う。描画されたピクセル数が分かる。クエリ完了をポーリングできる。

コリジョンマップ

- 当たり判定用の動的生成テクスチャ
- Ｙ軸平行投影でシェイプを描画
- シェイプのY=0での切断面が当たり形状
- if (Y>0)xor(表ポリゴン) then R+=1 else G+=1
- 参照時当たり判定を行うテクセルがR>Gで当たり



その他

- 多面体切断処理
  - 別スレッドで切断処理を実行。
- プロシージャルシェイプ
  - モデリングツールは一切使用していない。
  - シェイプは全てコードで生成。
  - 隕石はランダムに切断して生成。
  - 山岳はパーリンノイズで生成。
- 水面（波処理）
  - ＧＰＵによるテクスチャ更新、法線マップ化

多面体切断処理

まとめ

- ＣＰＵのパフォーマンスを引き出すのは難しい。
  - 中間言語。ＧＣあり。
- ＧＰＵのパフォーマンスはかなり引き出せる。
  - シェーダはＨＬＳＬとアセンブラで記述可能。
- 重いＣＰＵ処理はＧＰＵにオフロードすることを検討したい。
- 単純で物量的な処理ほどＧＰＵパワーが生きる。

連絡先

メール

yo1@heloli.com

サイト

http://www.heloli.com/

ツイッター

http://twitter.com/YO1KOMORI