大規模粒子データからの
ドーム投影用映像制作

武田隆顕
VASA エンターテイメント株式会社

2016/11/25 ビジュアリゼーションカンファレンス

広角度映像（プリレンダ or リアルタイム）

広視野の映像といえば、
旬の話題はVRだけれども…

Oculus Rift，プレステVRなど
ヘッドマウントディスプレイ（HMD）が
ある程度普及する気配を見せている。

リアルタイムレンダリングではなく
プリレンダで全天周映像を作る話です。

データが大きすぎて、リアルタイムレンダリングは
無理、という場合を念頭に置いています。

広角度映像（プリレンダ/リアルタイム）

プラネタリウムといった限られたドーム投影施設でしか
堪能する手段が無かったものが、

Oculus Rift, プレステVR などHMDが
ある程度普及する気配を見せている。

現時点で研究室に１つとか、施設に数台あるという状態は現実的？

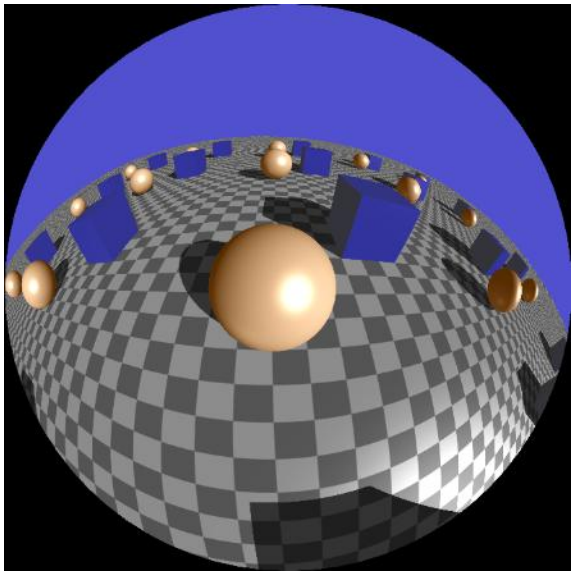
スマートフォンをあちこちに向けることで360度映像
（の一部の方向）を再生することができる

(Youtubeなどにもそういう360度映像が置いてある)

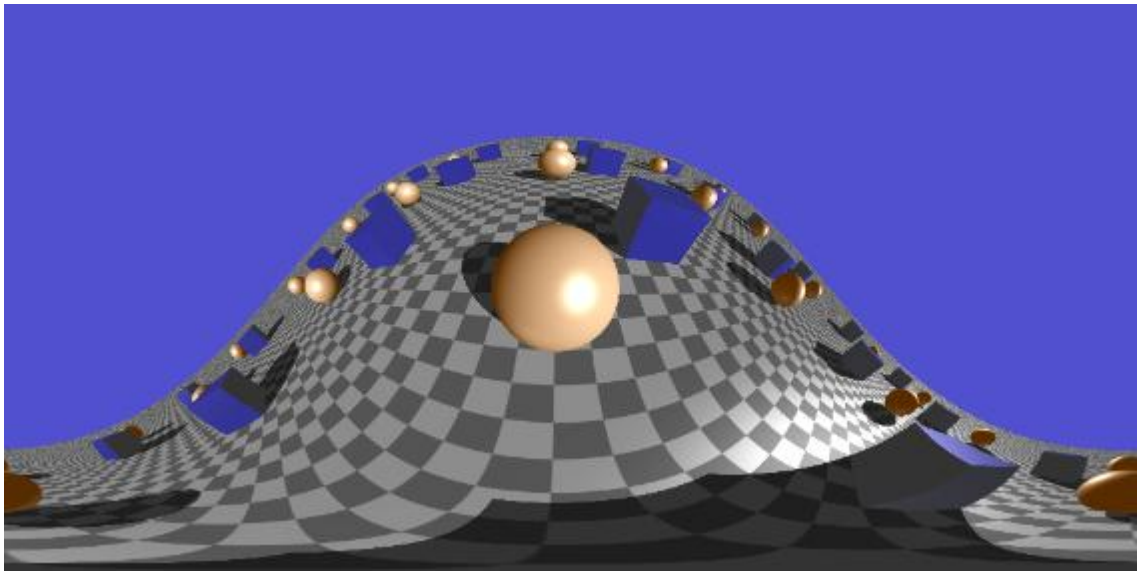
Pokemon GO などARのヒットで
スマートフォン越しに見る行為も充分広まった

半球ドーム、もしくは(360°全天投影用映像のフォーマット)

ドームマスター

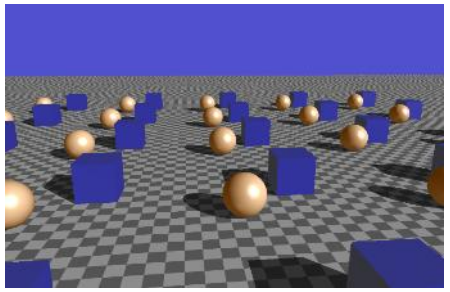


Equirectangular

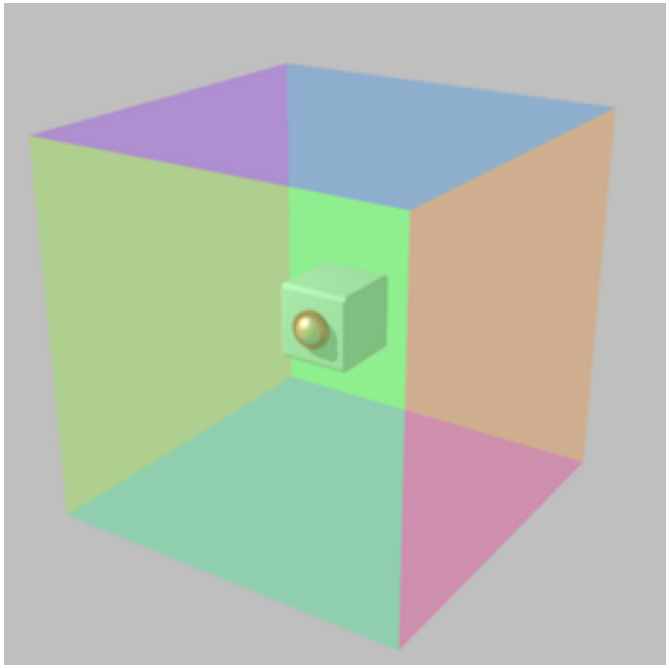


レイトレース法などのCG作成では、光線の方向を制御しやすいので
普通にこういった画像をレンダリングして作れる（、ようなCGソフトも多い）

OpenGL等で、ポリゴンを高速で描画する場合、
（普通は）一度にこういった表示はできない



広角度映像（プリレンダ/リアルタイム）



真ん中にカメラを置いて、
画角90度ずつ６枚レンダリングすれば、
360度全方位分の超広角情報が得られる。

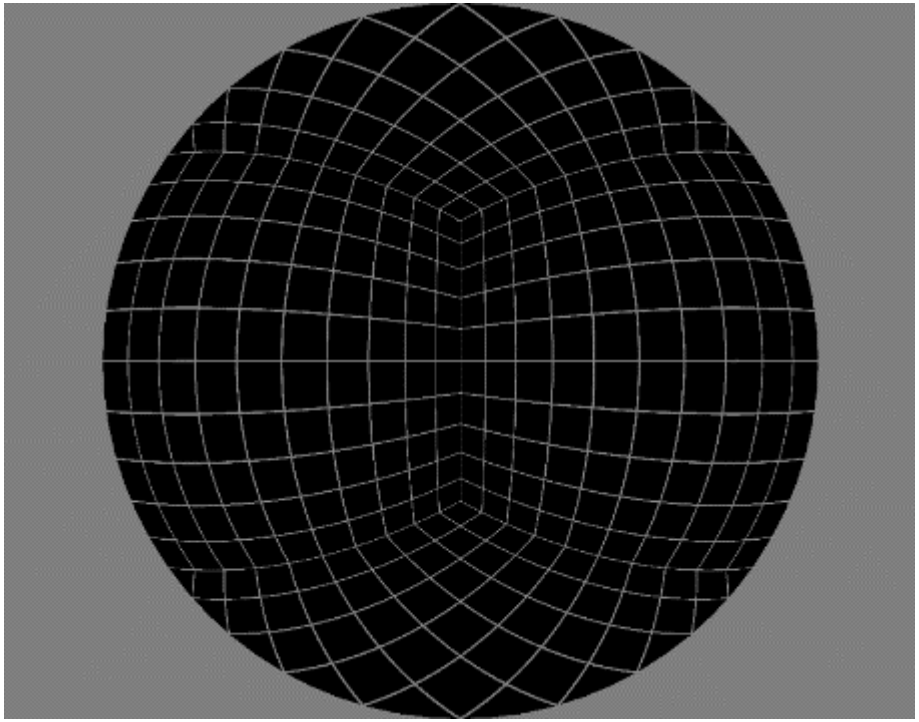
(6回のレンダリング、もしくは６並列で1回)

カメラを固定すればプリレンダのムービー

ヘッドトラッキングしてカメラを動かせば
リアルタイム系のVR

ドーム用映像（ドームマスター）
はそのうち４枚（もしくは3枚）の画像を
変換して作ることができる

ドームマスター（魚眼）

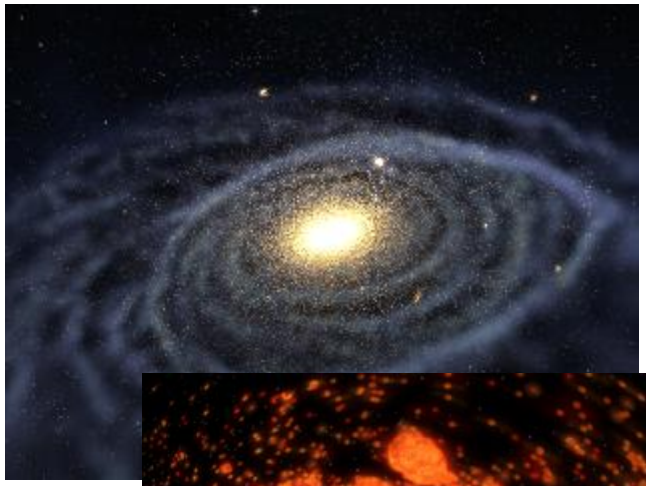


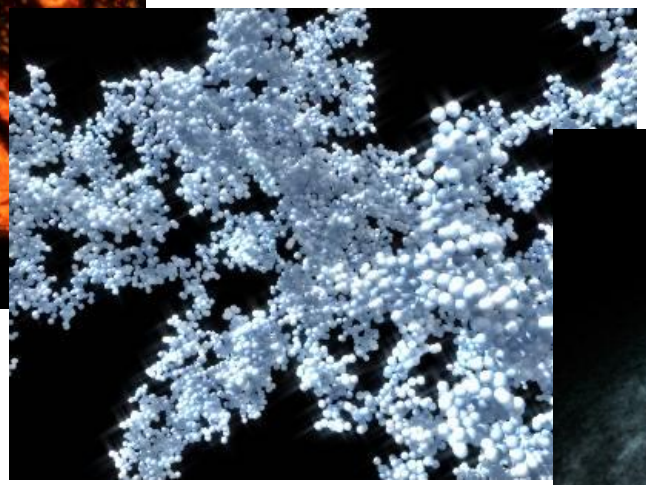
もっとも素直な方法では、
キューブマップの４枚を使い
座標変換をして合成する

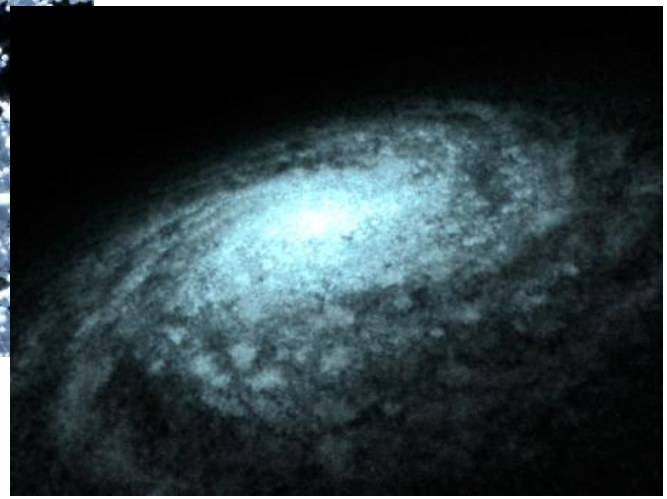
一枚の画角を大きくすれば
3枚から魚眼化可能
（丸からはみ出る部分は大きくなって
解像度的には損）

つなぎ目の部分に不連続があると
ドームに投影すると困る。

粒子データのドームマスター表示







天文学では、よく大規模な粒子データを使う

（銀河のシミュレーションなどが典型的）

とても重いので、CGを作るときも
レイトレースではなくOpenGL等を使いたい

粒子データのドームマスター表示

超広角の映像を作成する際、
(OpenGL等の表示では)複数のビューのつなぎ合わせが必要

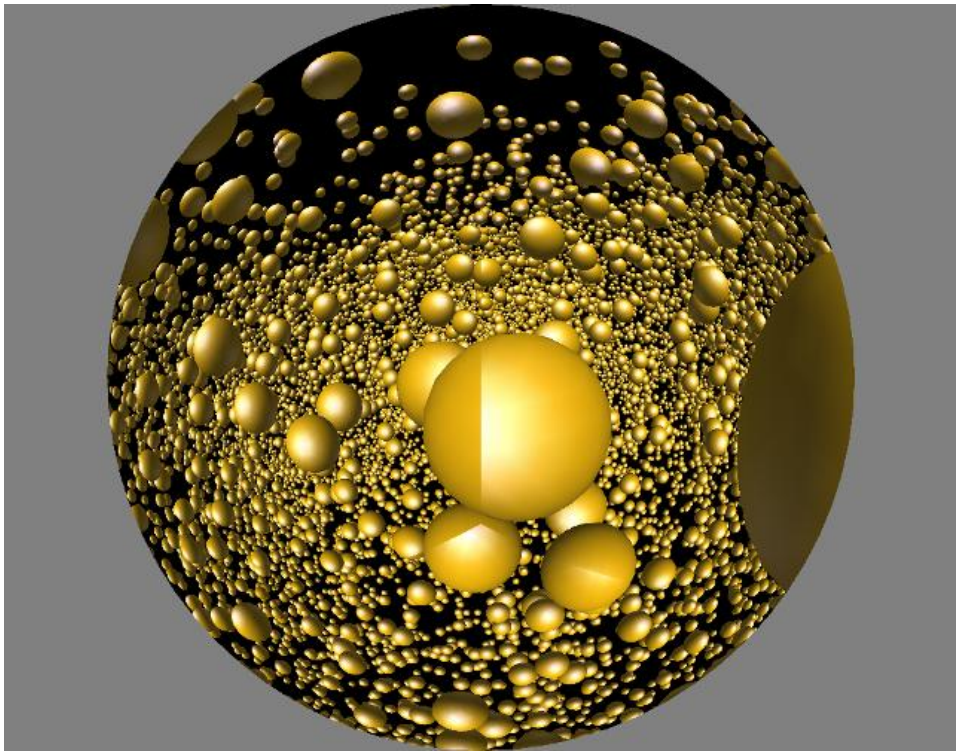
大規模なデータの表示などをする際は
（速度を十分に担保するために、プリレンダでも） OpenGL等での表示が必要

「つなぎ目で不整合が起きてはいけない」という条件があるので
高速化のための工夫などに、いろいろと制約が出てくる。

OpenGL等で作ったキューブマップから
超広角映像を作る際に
発生しがちな問題について

キューブマップに不連続の出る原因

３Ｄオブジェクトとしての表示



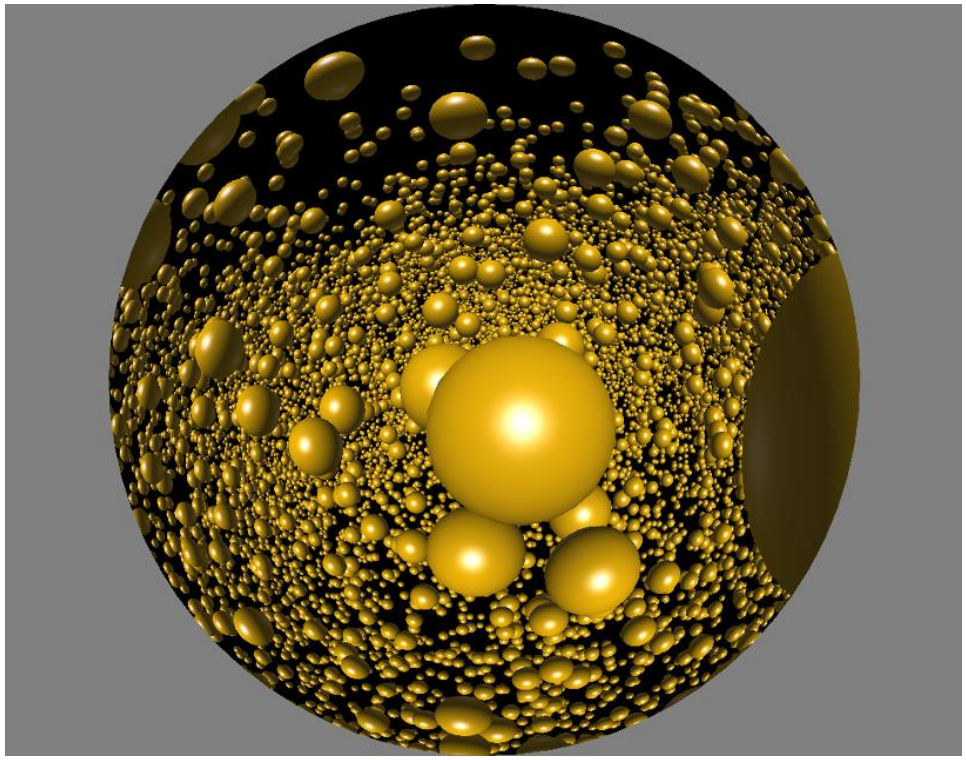
スペキュラー（光沢）

実は(クラシカルな)
OpenGLデフォルト状態の
スペキュラーの計算は正確ではない
（カメラの**向き**に依存）

本当はカメラの向きではなくて
(ポリゴンの位置 - カメラの位置)
に依存しないといけない

キューブマップの境界では
カメラの向きが90度変わることになるので、
スペキュラーの計算が不連続を起こす

GLSLを使ったシェーディング

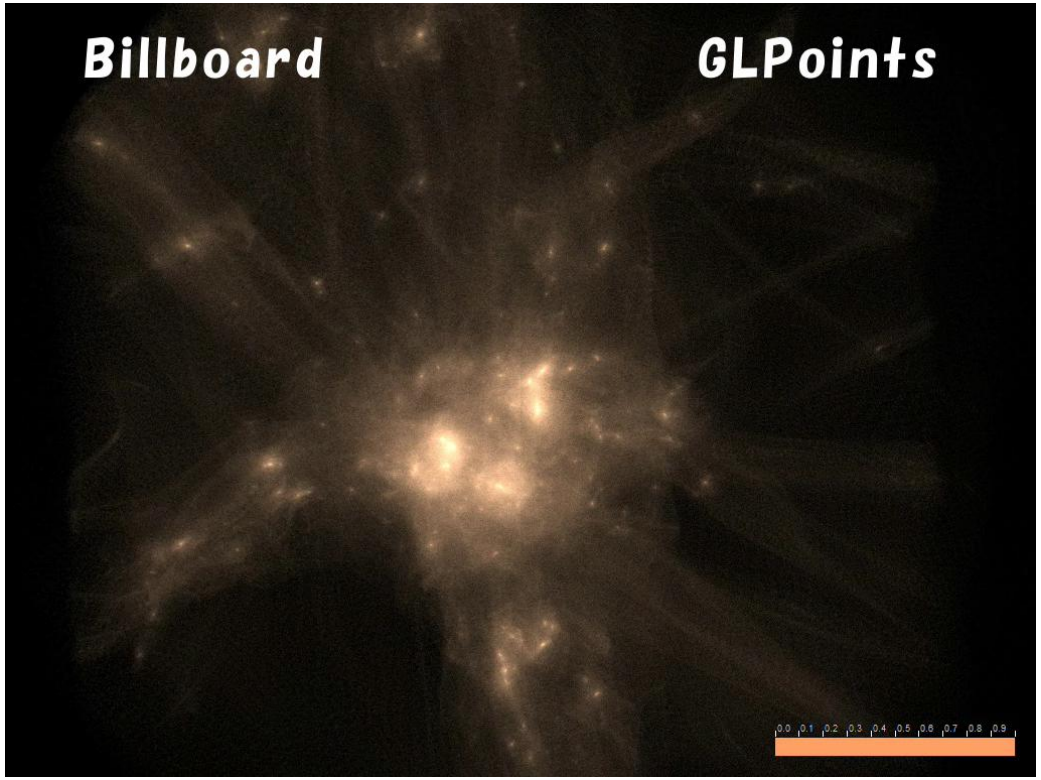


解決法：
クラシカルなOpenGLのレンダリング
ではなくて、
シェーダーを用いて反射の計算をさせる

シェーダー言語を使った表示の例としてよくある
フォンシェーディングなどでも、
光沢を(ポリゴン位置-カメラ位置)で計算をしているので、
つながるようになる。

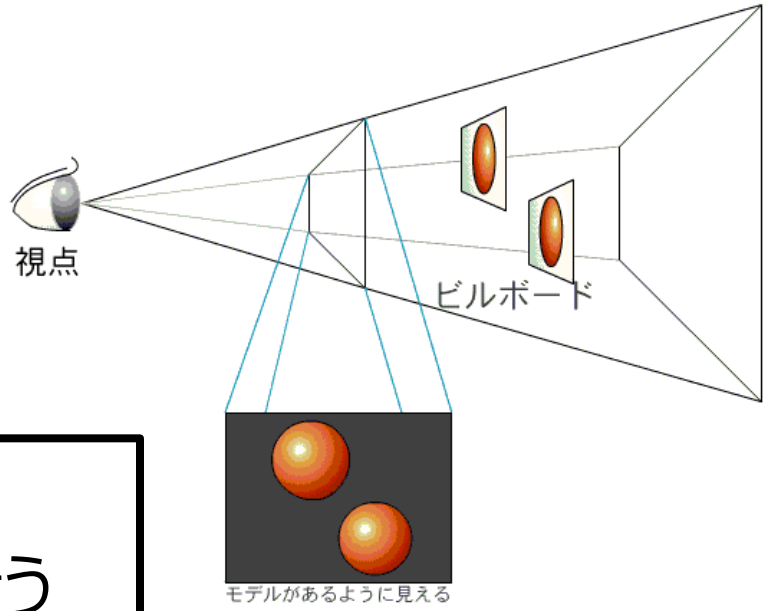
光点の描画

最速はpixelサイズの点を描画するGL_POINTS命令



ただし、距離によって
見かけの密度が変化

図の引用 : マイクロソフト アカデミック ポータル:
DirectX によるゲームプログラミング入門 : 第 6 章 実践ゲーム開発



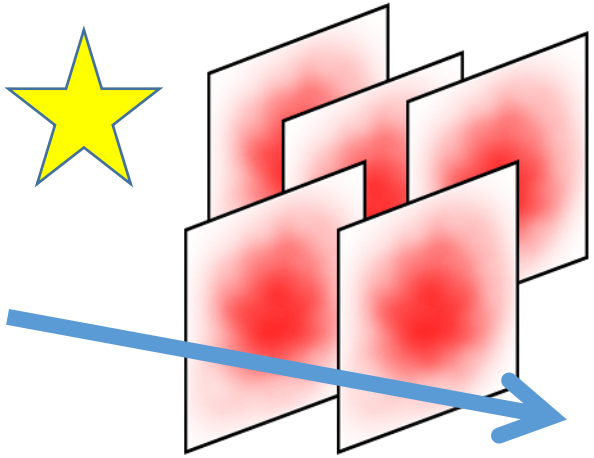
結局カメラ方向を向いた
板（ビルボード）による描画を行う

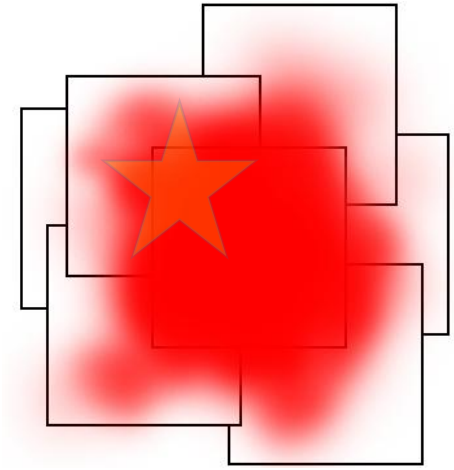
光点およびテクスチャによる粒子の描画

板ポリゴン（ビルボード）表示

表面を持ったオブジェクトではなく
ガスである場合。
星など光点だが
大きさやにじみなど構造が欲しい場合

大量のビルボード表示

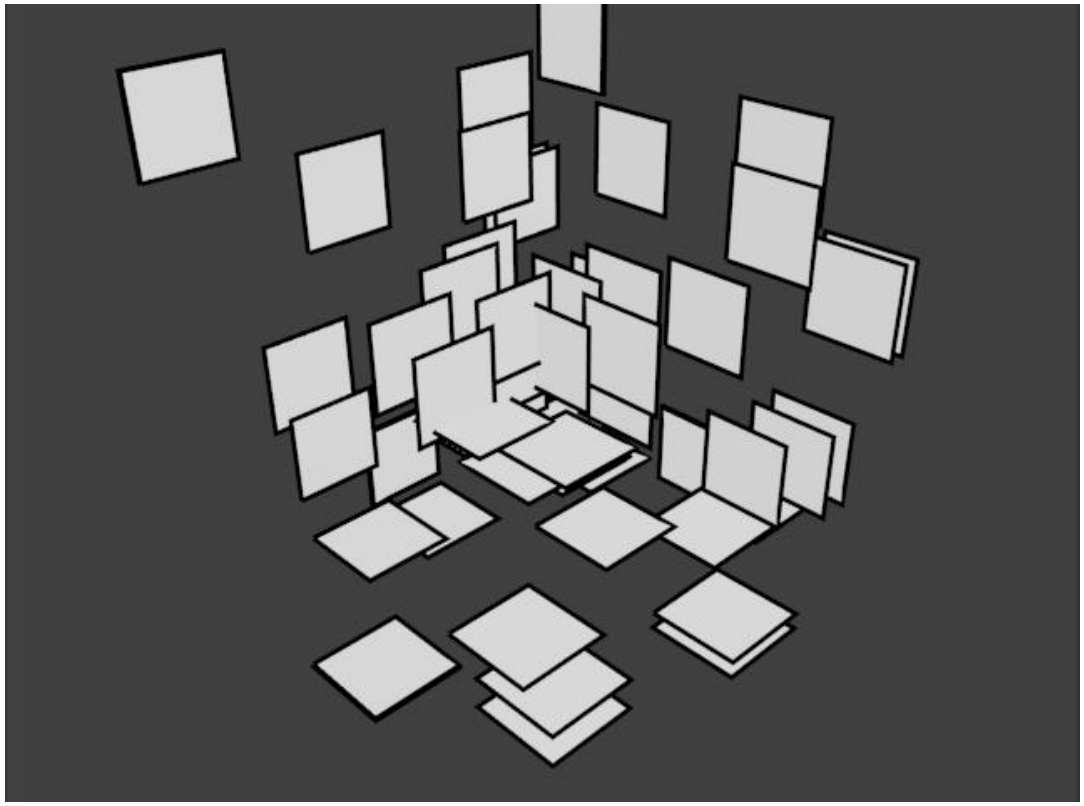




ソートして奥から描画

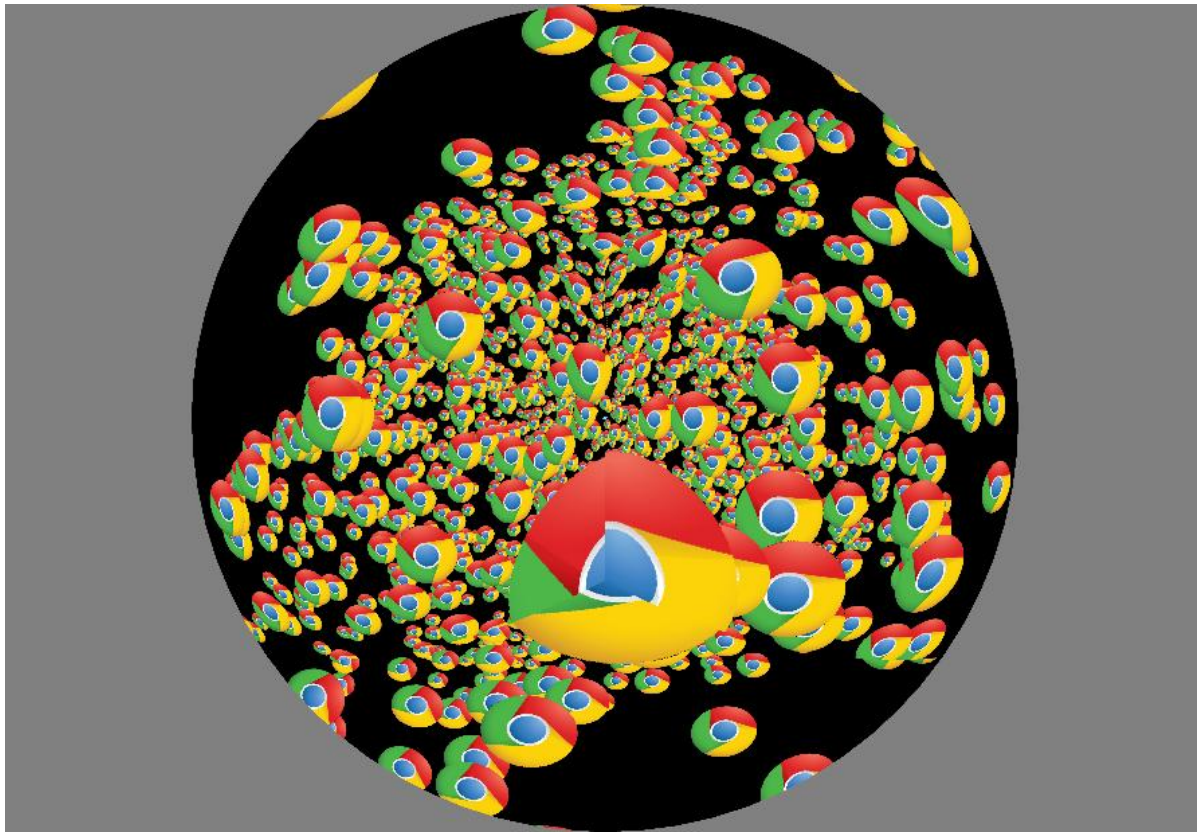
背後にあるものは隠れる

ビルボードで不連続の出る原因



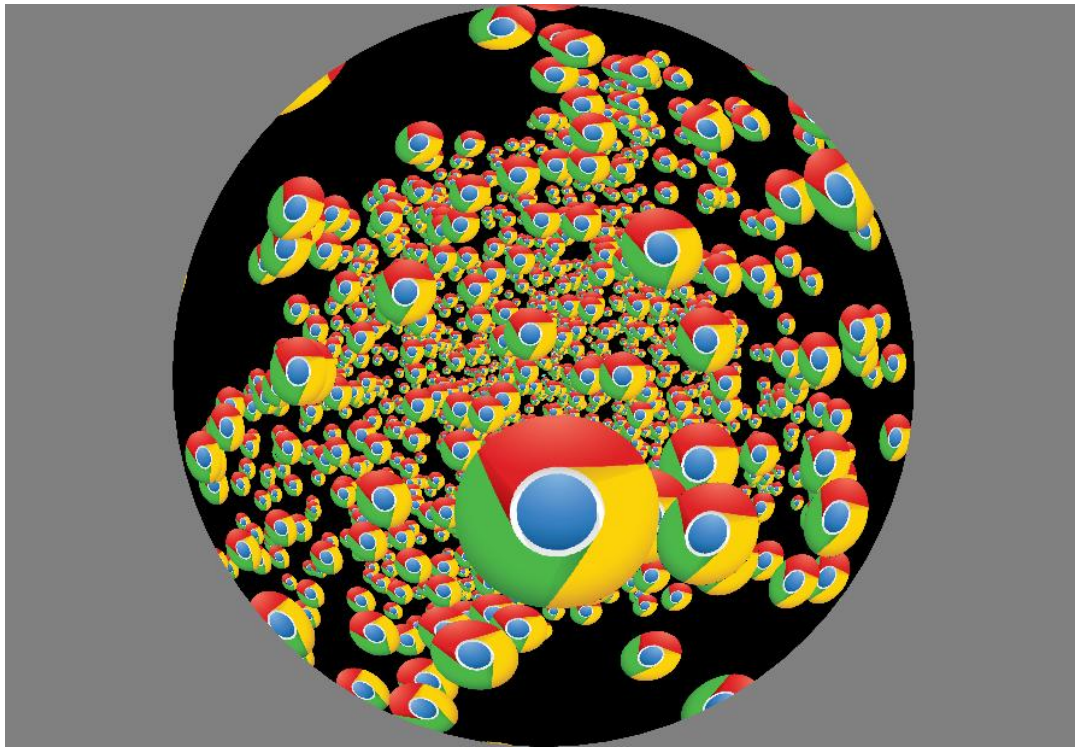
カメラの「視線方向」に従ったビルボード表示
見るからに不連続が生じそうな予感がしますが…

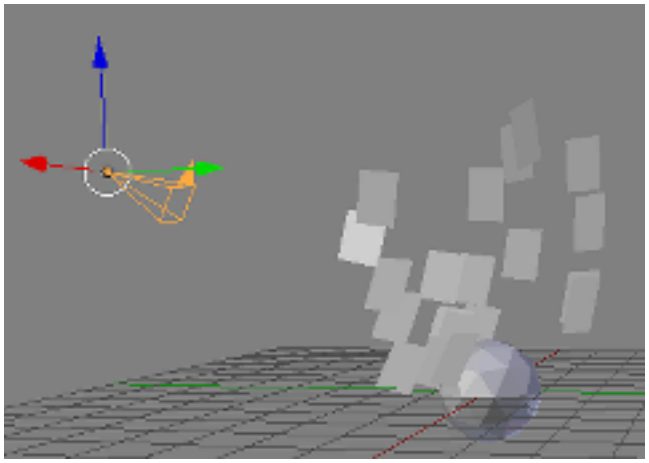
ビルボードで不連続の出る原因



丸くて色のわかりやすいアイコンを
テクスチャとして表示して合成
明らかにつなぎ目に不連続が生じている

ビルボードで不連続の出る原因

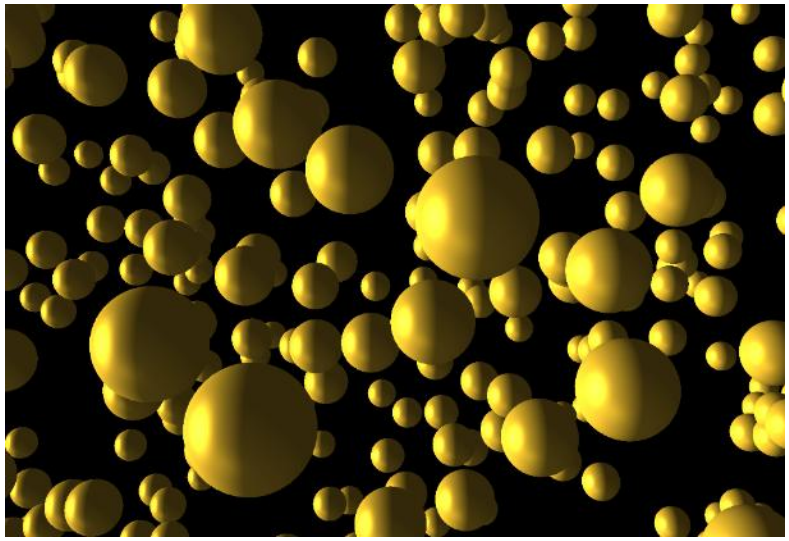




板ポリゴンを
カメラの「位置」を向くように
表示する

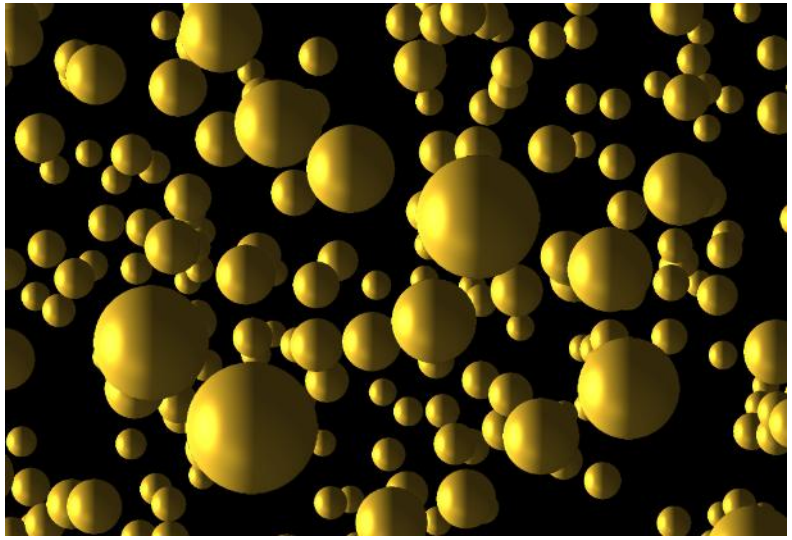
これで魚眼に変換も境界なしで行うことができる

ビルボード表示による高速化



球を大量に表示する

1つで240面体(240ポリゴン)

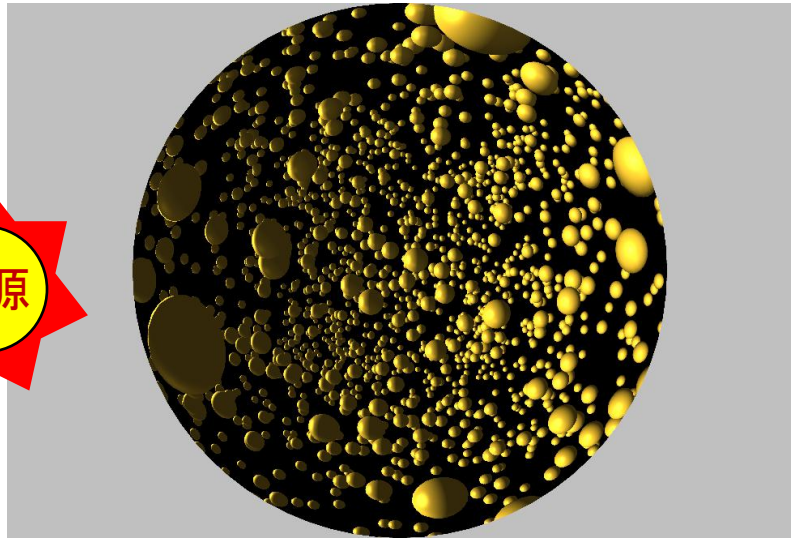


球を描いたビルボードを大量に表示する

1ポリゴン

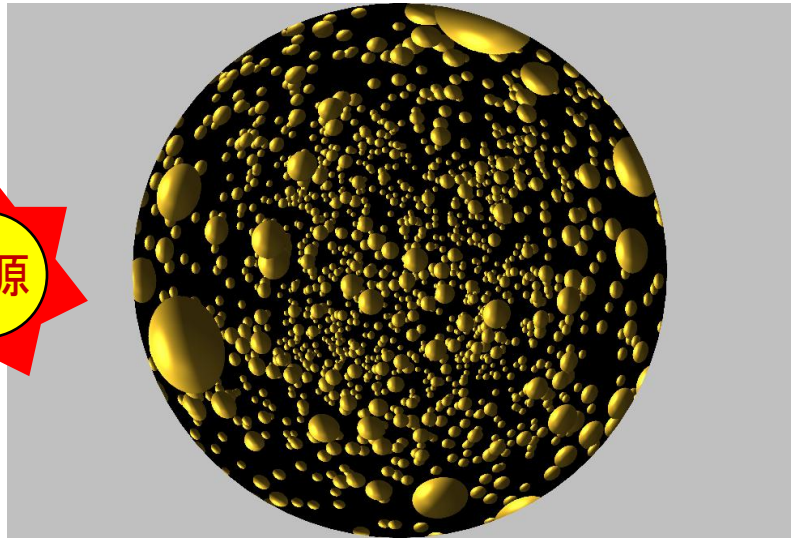
ビルボードで済ませられるときは
速度的には圧倒的にビルボード優位

光源問題



球を大量に表示する

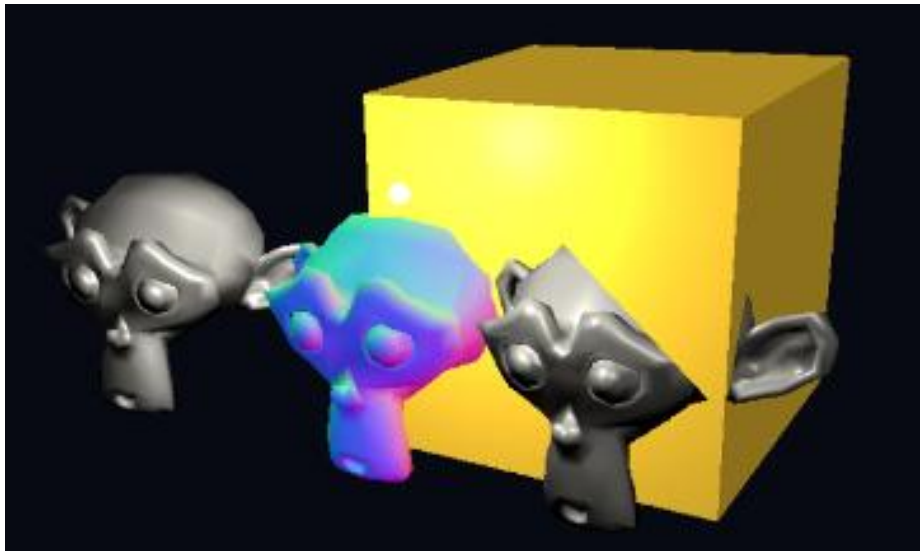
視野が広いと、照らされた側、
陰の側、同時に表示される



固定された画像だと
どちらも同時に描くことができない

光源の向きの不整合が起きる

ノーマルマップを使ったビルボード

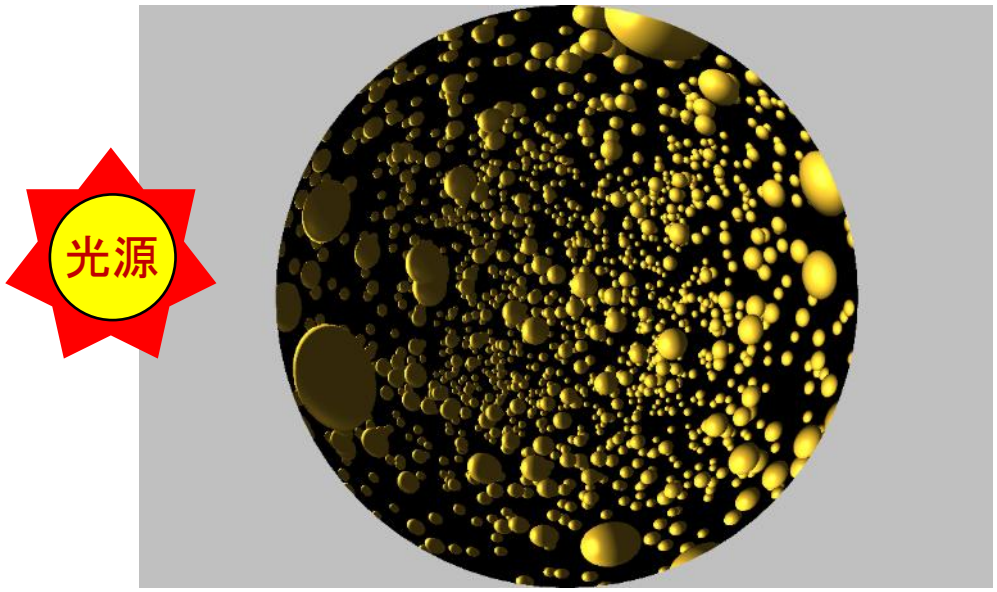


ノーマルマップ ＝ 法線の向きを色情報としてテクスチャとする

表示の時に、色を法線の向きと解釈して、
光源の向きに従って陰影をつける

一枚のテクスチャでも、いろいろな光源の方向に対応できる
広視野角の表示の時の光源問題も解決

ノーマルマップを使ったビルボード



テクスチャー枚（正確には色＋ノーマル情報で２枚）で
光源の向きも正確に反映させることができる

「球」の表示であれば、粒子の「向き」情報はいらないため
これでほぼ実用的に使える

ノーマルマップを使ったビルボード

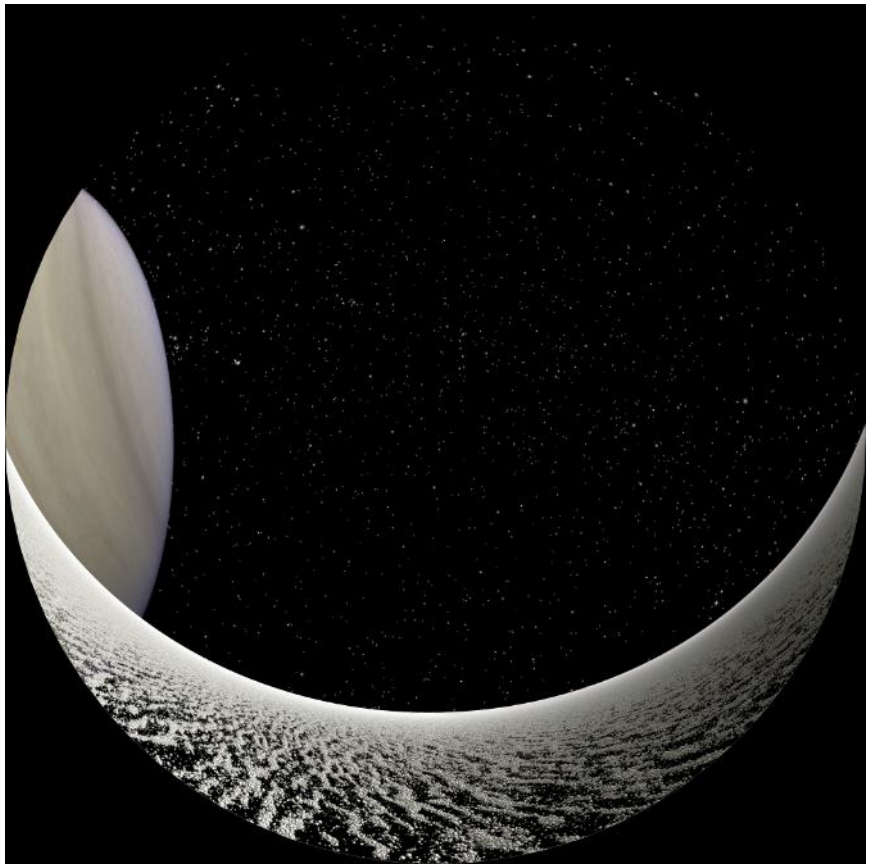
全部板なのに光源が反映される！

球の代わりに3Dモデルを表示

複雑な形状のオブジェクトでも
1ポリゴンの表示で済むので
高速に大量の表示ができる。

ただし、当然一枚絵が元になるので
オブジェクトの向きなどは変えることは
できないことに注意！

以上のテクを使っての映像作成



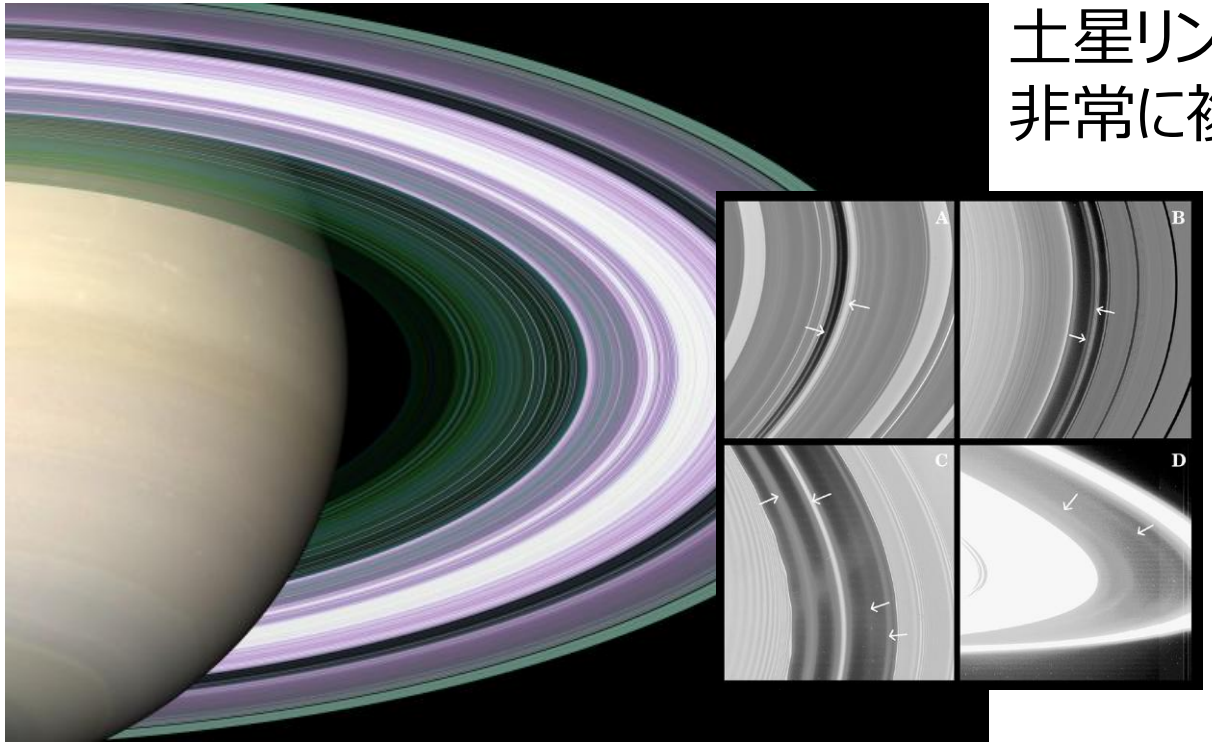
土星リングシミュレーション(2016)
(国立天文台4D2U映像)

土星リングの正体　＝
氷の塊（小衛星）の集合体

大量の氷の塊が
地平線の向こうまで分布している状態

これを全天映像に落とし込みます

土星リングのシミュレーション



土星リングは
非常に複雑な構造

リング粒子の運動は
興味深い研究ターゲット

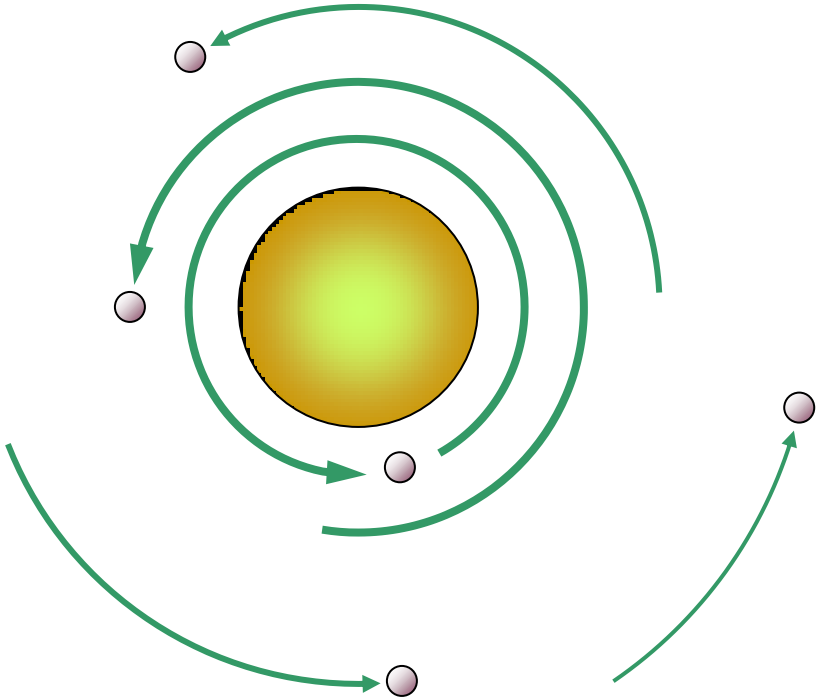
■運動をまともにシミュレーションしたいが

メートルサイズの氷の塊が、
地球の１０倍以上の広さに
撒き散らされている

全部計算
するのは絶対無理

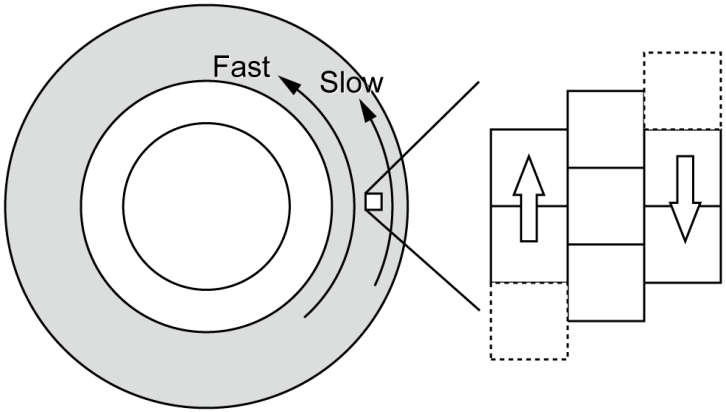
一部領域を計算する

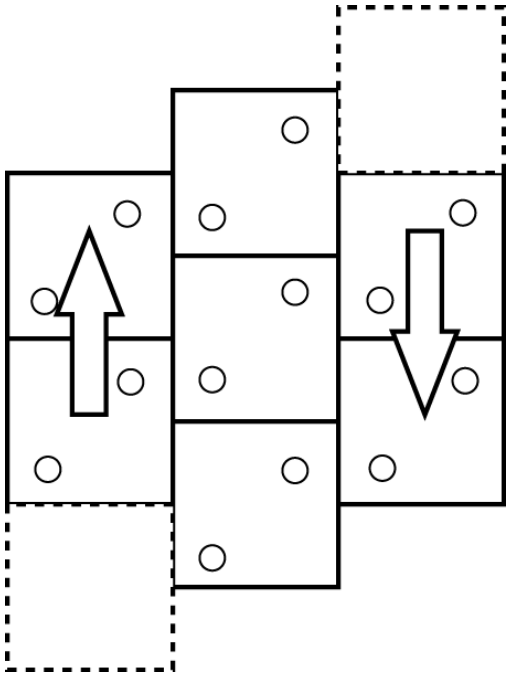
土星リングのシミュレーション



速度差を加味して
差動回転する箱の中で
粒子の運動を計算

ケプラーの法則：
内側の方が速い
外側の方が遅い

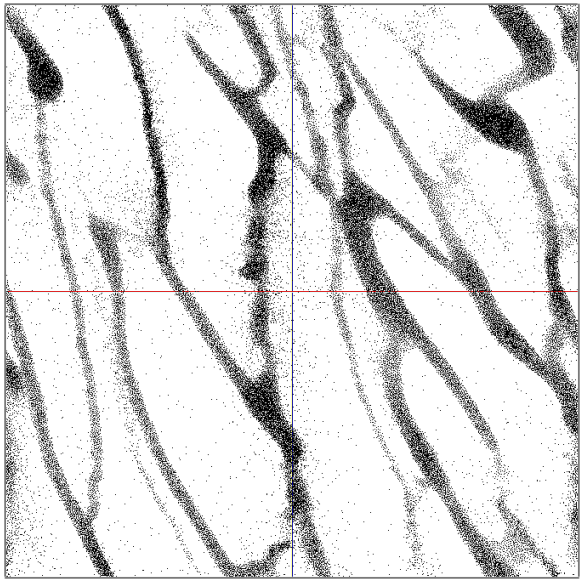


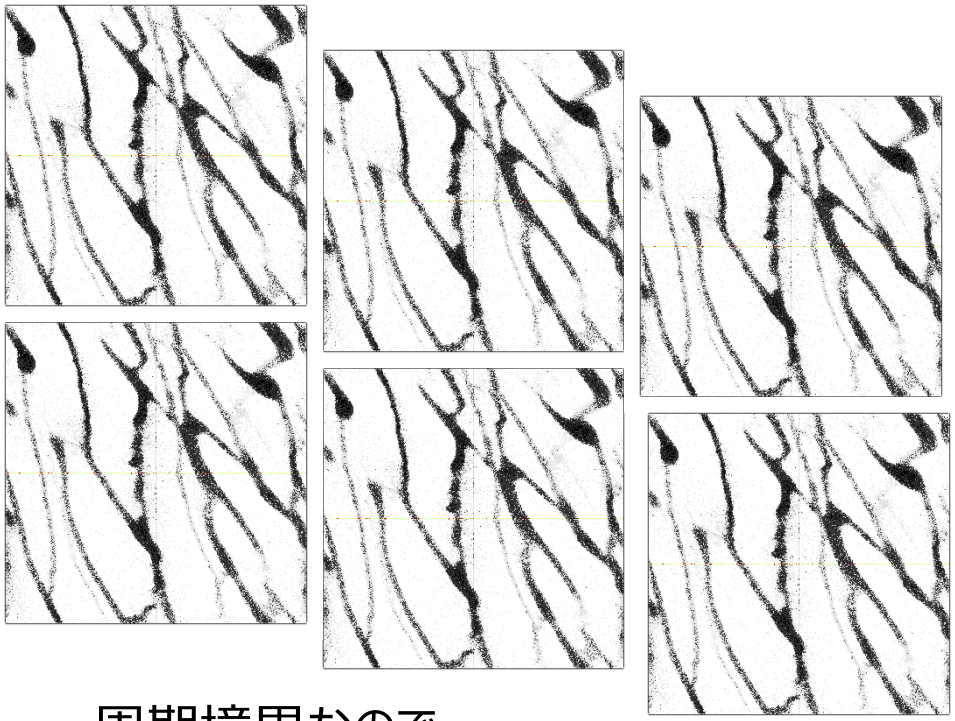


土星リングのシミュレーション

こんな構造が生まれる

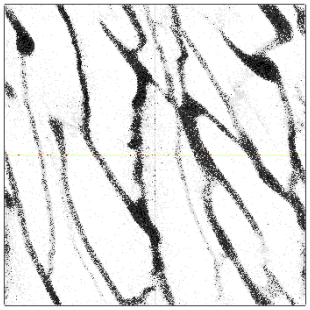
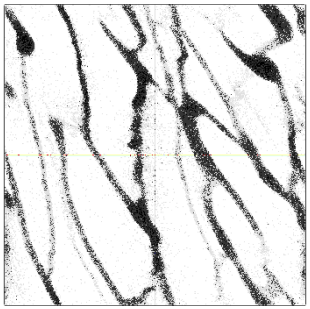
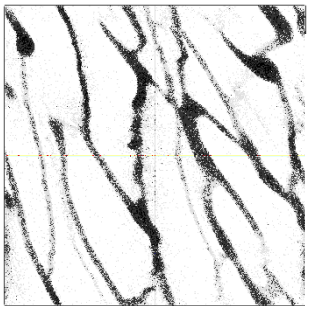
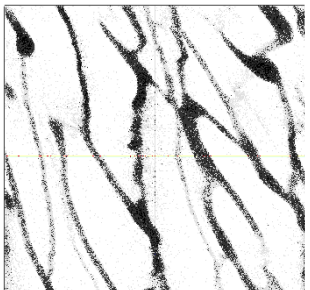
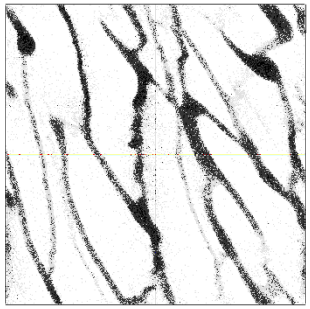
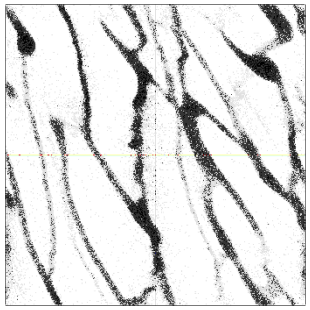
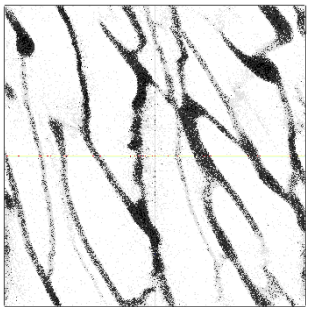
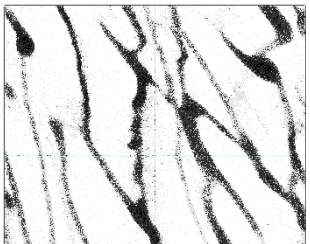
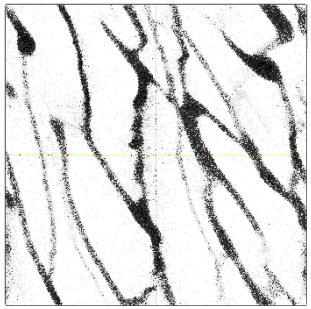
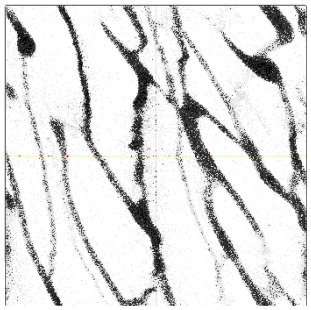
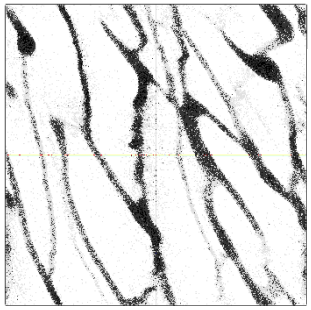
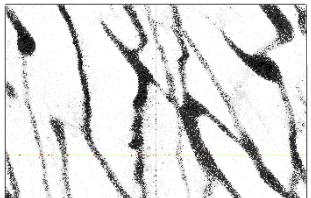
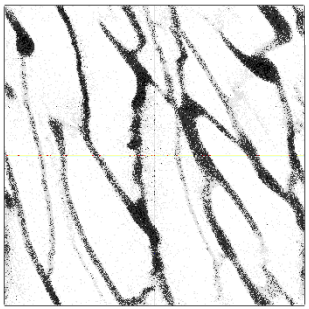
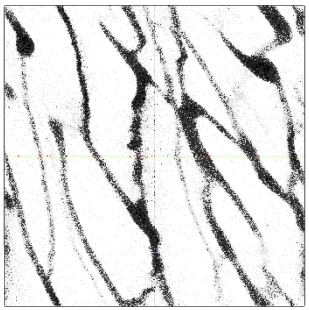
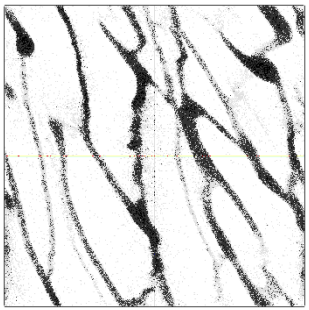
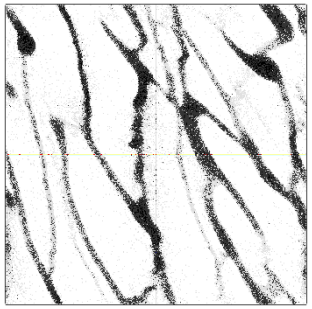
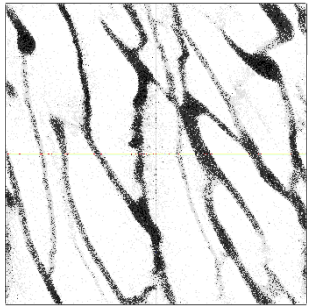
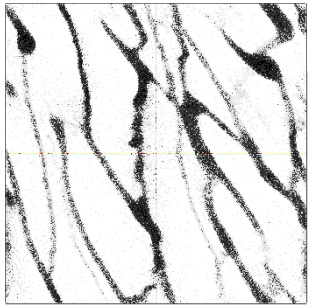
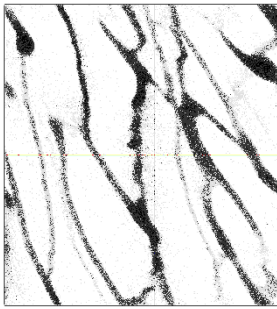
重力でまとまろうとする効果と、まとまりかけた塊が、
潮汐力で引き伸ばされる効果が釣り合って縞構造が生まれる





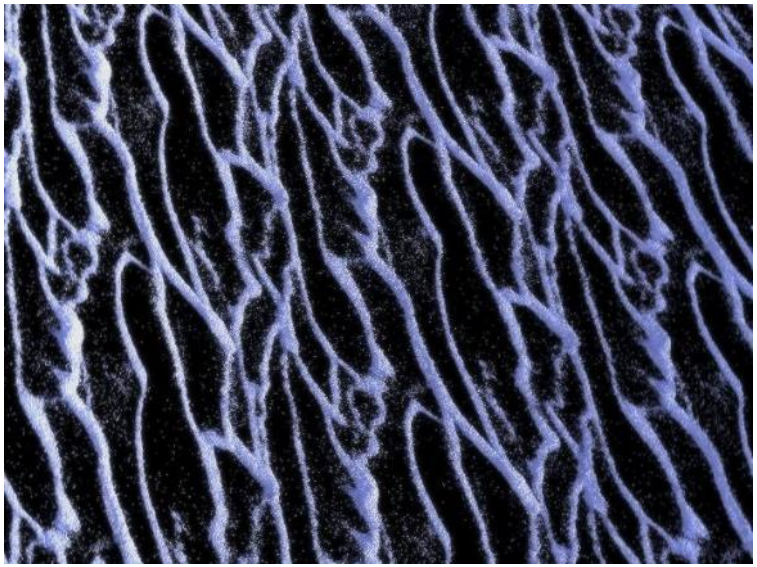
周期境界なので、
コピーを展開するとこんな感じ

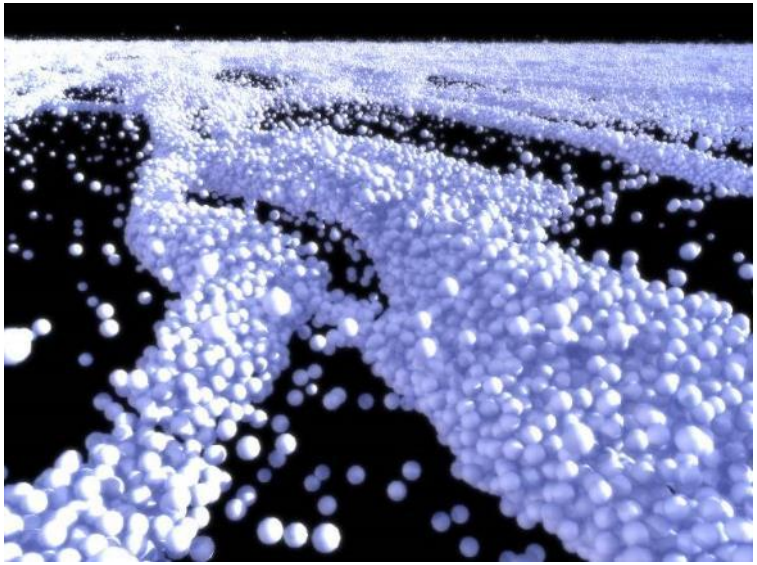
土星リングのシミュレーション

地平線の果てまで広がっている

以前のものとの比較





以前、別の土星リングシミュレーションを作成して
カンファレンスで紹介したことがありました。

当時の映像でできなかったことは…

氷粒子にディテールが無い
球の集合体として表示（ピンポン玉）

球として表示をしていたので、
表示速度的にもきつかった

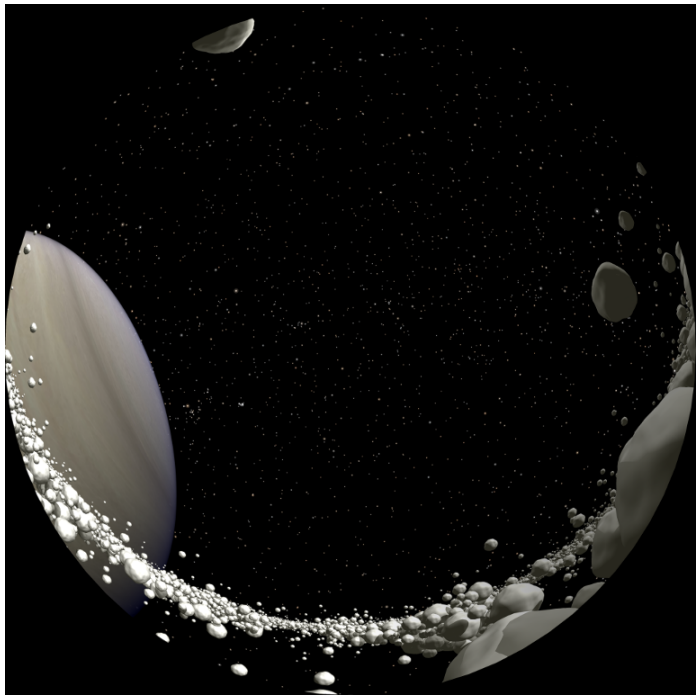
地平線の果てまでは表示できず、
途中でカット。遠くは間引きながら200万粒子ほど

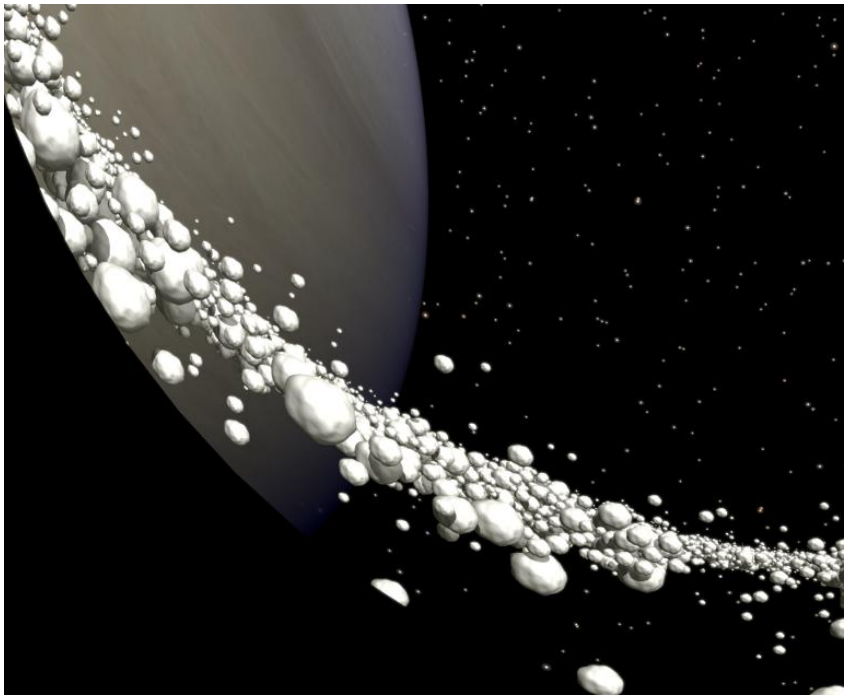
土星や背景の星空などは同時に表示しなかった

以前のものとの比較

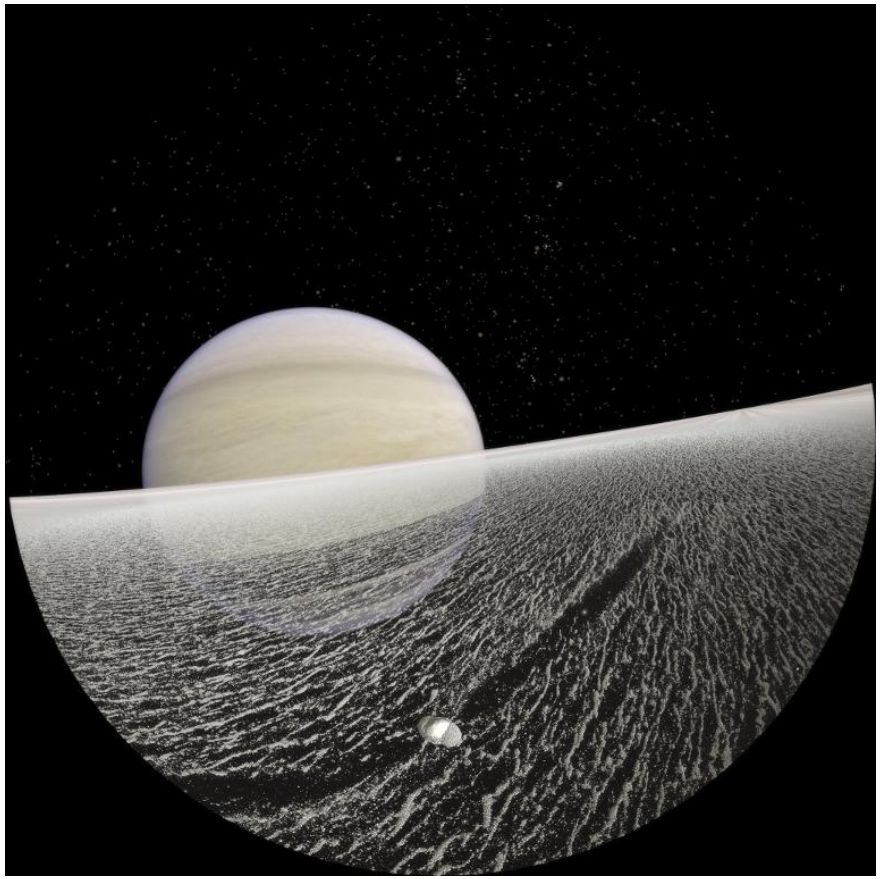
氷粒子を球ではなく、凹凸のある表現へ（もうピンポン玉ではない）

ビルボードとして表示することでより大量の表示。
（厳密には、近くはオブジェクト、遠くはビルボード表示での合成）
地平線の果てまで広がる土星リングの再現（最大約60億粒子ほどが表示）



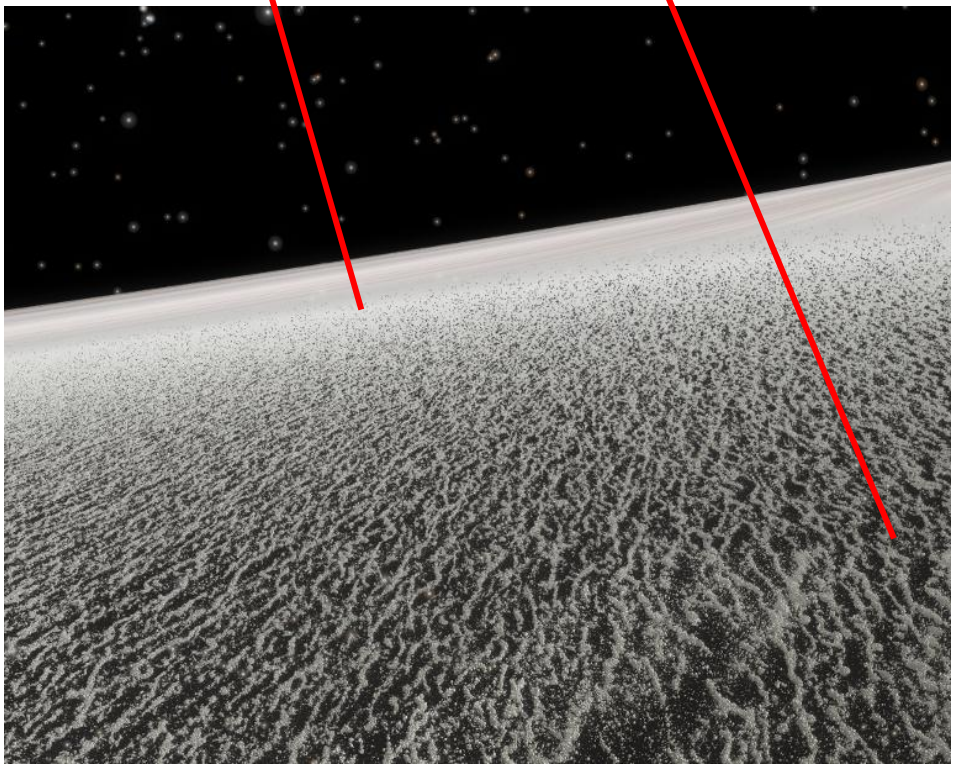


以前のものとの比較



大体この辺りまで表示

ビルボードとの切り替え
よく見ると若干色調の差は
見える



映像再生

まとめ

超広角の映像を作成する際、
(OpenGL等の表示では)複数のビューのつなぎ合わせが必要

大規模なデータの表示などをする際は
（速度を十分に担保するために、プリレンダでも） OpenGL等での表示が必要

「つなぎ目で不整合が起きてはいけない」という条件があるので
高速化のための工夫などに、いろいろと制約が出てくる。

土星リングのドーム映像の制作での経験から、
使ったテクニックやおきがちな落とし穴などをまとめて紹介した。