A presentation on my background in computer graphics

M. Mostajab

www.mmostajab.com

December 31, 2016

Outline

About Me...

- My name is **Morteza Mostajab**
- Bachelor studies:
  Hamedan University of Technology, Iran
- Maseter studies:
  Technische Universität München
- Present:
  Researcher at Fraunhofer IGD, Darmstadt
- Research interests:
  Real-time physically-based rendering
  (Rasterization-based or Ray-tracing)
  Virtual reality
  Computer graphics and visualization
  Game Programming

Inspiration

- Games, Animations, Movies with Special Effects,...


(a) Last Ninja 3


(b) Gears of War


(c) Ratatouille


(d) The lord of the rings

- My firsts...


(e) First Computer


(f) First IBM compatiable PC


(g) First Console (Atari 2600)

Outline

Motivation

- **Goal**: Use virtual reality to help engineers getting a better **understanding** of simulations, make **investigation** in flow field features easier.
- Virtual reality applications demands 75-140 frames per second scene update and rendering to reduce **latency**.
- **Initial motivation**:
  – **Immersive Streamline Demo**: a demo which user could stand in a simulation model to understand flow field features by:
  - Prototype demo which placed the user virtually in simulation model.
  - User could interact with virtual world by moving streamlines' seeding plane.
  - User could walk around in the simulation model, look closer into streamlines to understand flow field's features better.

Immersive Streamline Demo

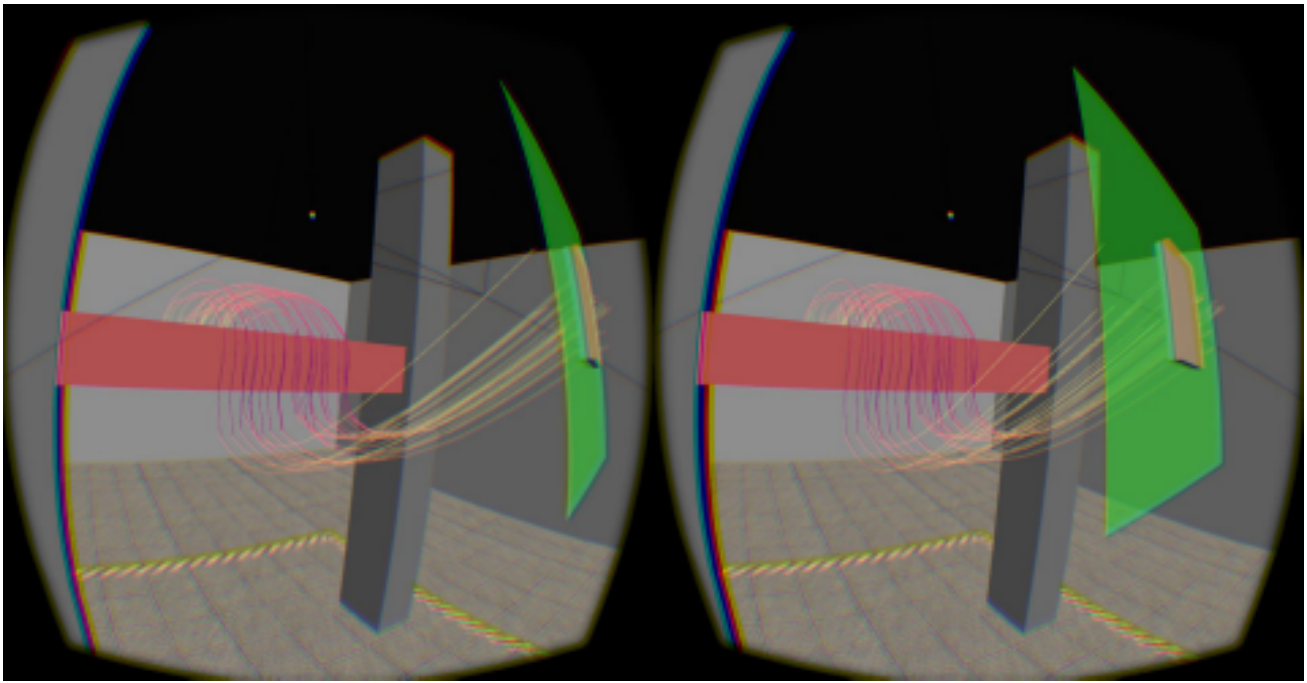


Figure 1: **Immersive Streamlines** was a prototype demo done at Fraunhofer IGD. The user is able to move inside the virtual world, interact with it, and look closely at the flow visualization results. The image shows a stereo pair, which is displayed on Oculus Rift HMD.

Problem Definition

- Streamline's computation was 10 frames per second (latency was very visible to user).
- Streamline's accuracy was not high enough (straight lines in streamlines are visible in the screenshot).
- Stream surfaces can provide more information about flow fields features.
- So, we defined **Parallel Stream Surface Computation and Rendering** master thesis to solve these problems.

Main Contributions

- Investigation techniques from ray tracing field for application in accurate streamline computation:
  - Using acceleration structures
  - Using ray-packing
- Using heterogeneous computing:
  - Scale streamline computation on all capable devices.
  - Scale rendering on all graphic processing units.

Benefits

- Accurate streamline computation method (using **Runge-Kutta** integration method).
- **Adaptive Runge-Kutta** integration method to adapt steps to maximum integration error.
- Heterogeneous stream surface computation.
- Multi GPU stream surface computation and result

Related Works

- Parallel stream surface computation for large data sets [CCG+12]. proposed a distributed stream surface computation system.
  – **My method is not distributed but uses all available computation devices.**
- Interactive Streak Surface Visualization on the GPU [BFTW09]. Bue+09 samples the simulation mesh on a regular grid, then compute the streak surface.
  – **Reduces accuracy, ignores a lot of information exist in simulation mesh by sampling it on a regular grid.**

Related Works

- Interactive particle tracing for the exploration of flow fields in virtual environments [Sch08].
  Sch08 uses the neighboring graph instead of acceleration structure on GPU.
  – **step size needed to be small enough.**
- Fast, Memory-Efficient Cell Location in Unstructured Grids for Visualization [GJ10].
  introduces cell-tree acceleration structure use it for particle tracing.
  – **I have evaluated more acceleration structures to classify them based on memory requirements and performance.**

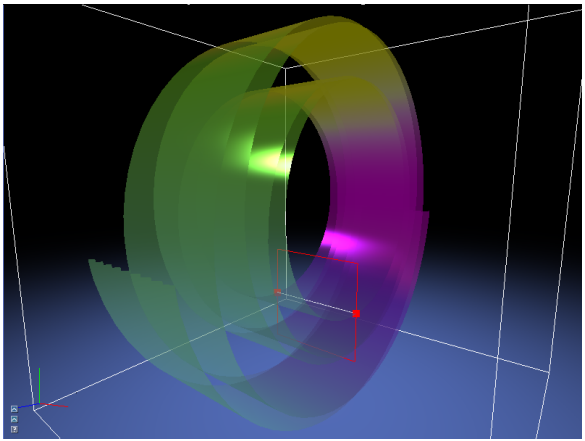
Stream Surface Computation Algorithm [CCG+12]

Data: *seeds* list which contains all initial seeding points.
Result: Stream Surface Points
while *seeds list is not empty* **do**
  Tracing: Trace streamlines originating from seeding points stored in *seeds* list.
  Refinement:**for** *each pair of adjacent streamlines* s_1 *and* s_2 **do**
    $d \leftarrow Distance(s_1, s_2)$
    if $d \leq D_{disc}$ **then**
      Discontinuity: The surface is discontinued in that area.
    else
      $s_{new} \leftarrow (s_1[0] + s_2[0])/2$
      add s_{new} to list *seeds*.
    end
  end
end

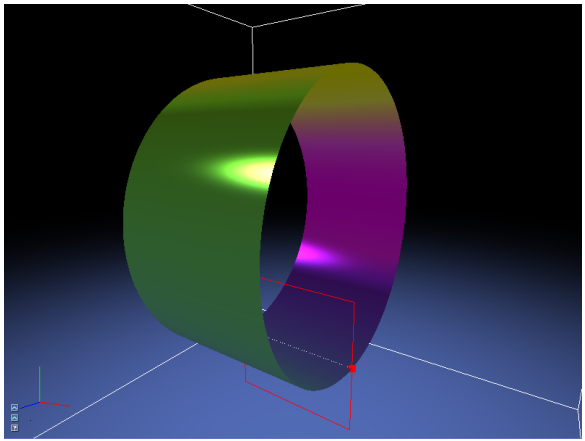
Algorithm 1: Stream Surface Computation Algorithm [CCG+12].

Accurate Streamline Computation

- Point-Inside-Cell checks & interpolating results inside cells.
- Using *adaptive Runge-Kutta* integration method (Cash-Karp).
- Adaptive stepping (faster and more accurate integration).
- Avoiding early terminations.



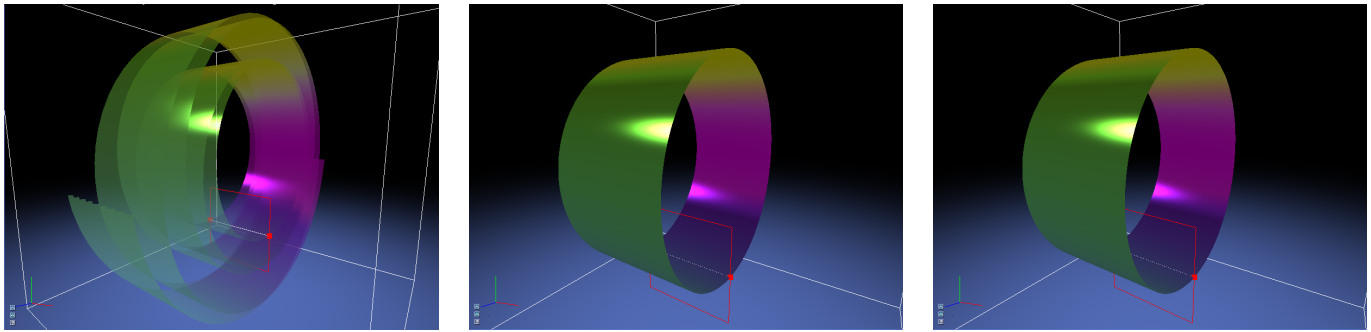
(a) Euler



(b) Cash-Karp

Figure 2: Euler and Runge-Kutta integration methods in a perfect rotation around center which are integrated with the same step size.

Refinement Strategy Results



(a) No Refinements (b) The first refinement iteration (c) The second refinement iteration

Figure 3: The result streamlines making skeleton of stream surface after 2 steps refinement.

Heterogeneous computing model

What was in my thesis.

Static Distribution of Seeding Points

What was in my thesis.

Dynamic Distribution of Seeding Points

What will be implemented in holidays 2016.

Heterogeneous computing results

What I will measure.

Acceleration Structures

- Grid
- Kd-tree
- Bounding Volume Hierarchy (BVH)
- Cell-tree [GJ10]

Figure 4: method

Why Acceleration Structures?

- Accelerating cell look up for faster streamline computation.
- Ray tracing uses acceleration structures to find **nearest intersection point** in contrast to streamline computation which uses it to **sample result in a specific point**.
- Differences:
  - In streamline computation instead of *Surface Area* Heuristics, textitVolume Heuristics should be used.
  - *Traversal algorithms* for space partitioning cases does not need *stack*.
  - Hierarchy traversal algorithms need a greedy method to prefer one child to other one during traversal.

Acceleration Structures Comparison Results

- The origin...

Outline

Ray tracing revisited for rendering CSG models consisting of higher order primitives

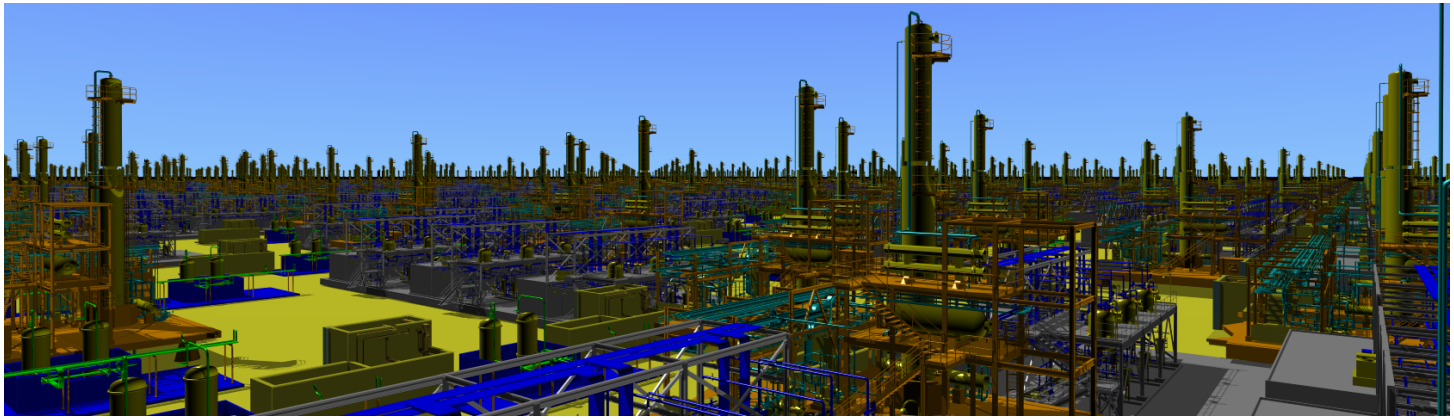


Figure 5: A complex CAD scene built out of 8,000 individual plant models. In total, the scene consists of more than 100,000,000 non-planar second-order (cylinders, cones, etc.) and forth-order (tori) primitives, ray traced at more than 10 frames per second on 16 CPU cores, including pixel-accurate shadows.

Benefits

- **On-the-fly compositing.**
  – Boolean set operations are performed on a per-ray basis immediately during rendering.
  – No Preprocessing.
- **Low storage requirement.**
  – small set of parameters for each primitive.
  – No prior triangulation of primitives. – Holding scenes in memory would be possible without triangulation.
- **Pixel-accurate higher order surfaces.**
  – Ray-primitive intersection can be tested directly. – Usually a quadratic equation which is easy to solve (tori has a quartic equation).
  – No discretization artifacts are visible.
  – Surfaces appear perfectly smooth.

Main Contributions

- a method to apply CSG operations in real-time during ray tracing while keeping the memory overhead low.
- Evaluation of the proposed method in comparison to other well-known rasterization-based real-time CSG model rendering techniques([GHF86], and [SLJ02]).

Optimized Hitpoint Calculation

ray: current ray hitting a primitive

if *entering primitive* **then**
 | *delta* := +1;
else
 | *delta* := −1;
end
if *positive primitive hit* **then**
 | *ray.posDepth* += delta;
else
 | *ray.negDepth* += delta;
end
if *(ray.posDepth* > 0*)* && *(ray.negDepth* <= 0*)* **then**
 | ReportHit(); // final hit in layer found
else
 | ContinueRay(*ray*); // still inside a negative medium
end

Algorithm 2: RayTraceLayer(*ray*)

Supported Higher order primitives


Pyramid


Box


Circular torus


Rectangular torus

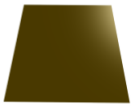

Spherical dish


Elliptical dish


Snout

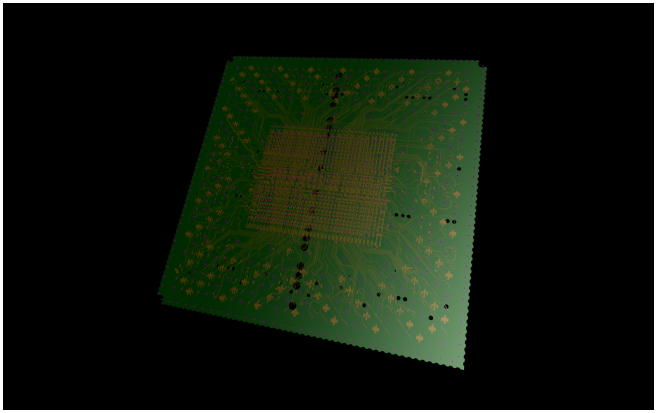

Cylinder

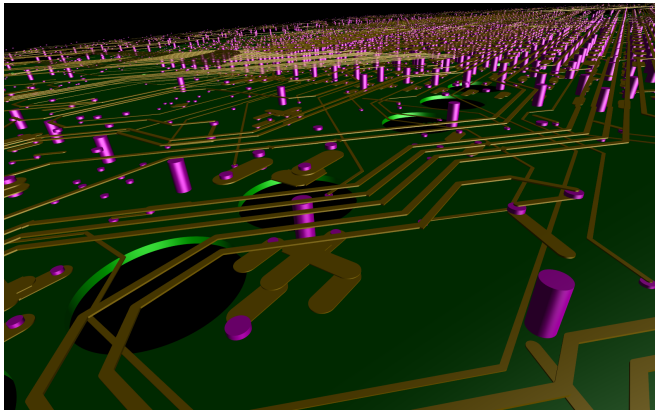

Sphere

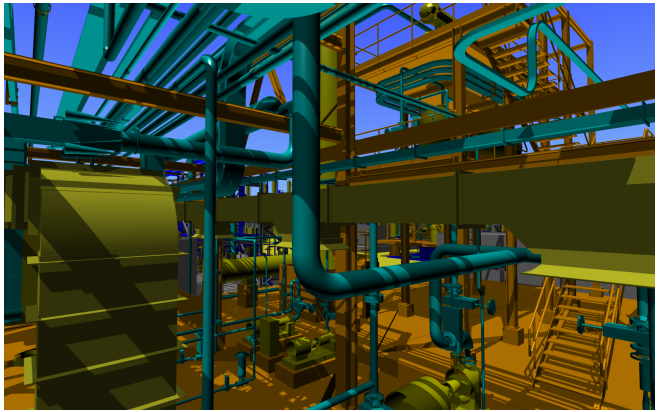

Plane


Triangulated mesh

More Results







Outline

Ray tracing revisited for rendering CSG models consisting of higher order primitives

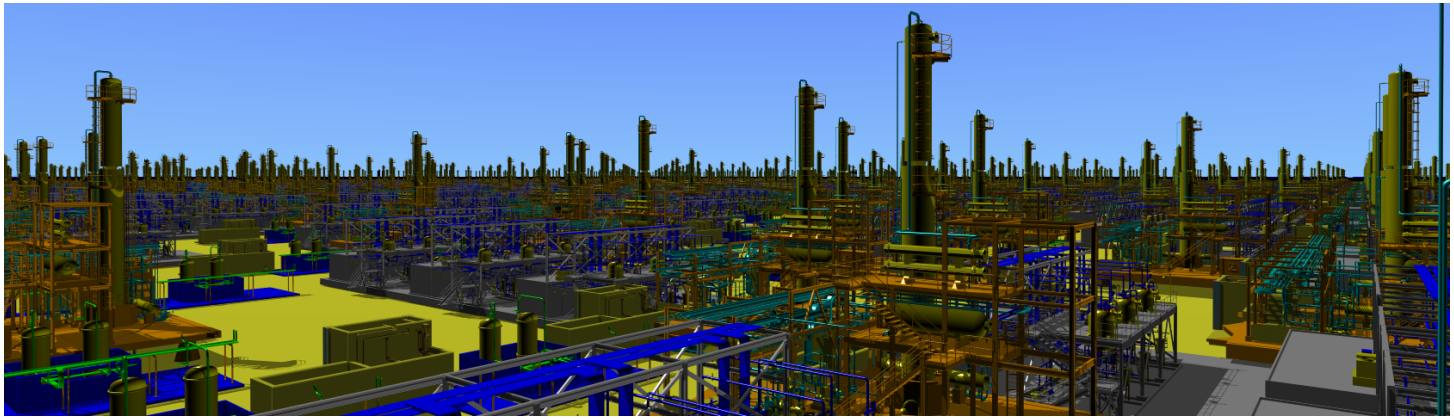


Figure 6: A complex CAD scene built out of 8,000 individual plant models. In total, the scene consists of more than 100,000,000 non-planar second-order (cylinders, cones, etc.) and forth-order (tori) primitives, ray traced at more than 10 frames per second on 16 CPU cores, including pixel-accurate shadows.

Outline

Practical Courses

Computer Graphics related courses

MetaIO

Zull Dutch

IDP

Outline

What do you want to be known for in next.

References I

- K. Buerger, F. Ferstl, H. Theisel, and R. Westermann, *Interactive streak surface visualization on the gpu*, IEEE Transactions on Visualization and Computer Graphics **15** (2009), no. 6, 1259–1266.
- D. Camp, H. Childs, C. Garth, D. Pugmire, and K. I. Joy, *Parallel stream surface computation for large data sets*, Large Data Analysis and Visualization (LDAV), 2012 IEEE Symposium on, Oct 2012, pp. 39–47.
- Jack Goldfeather, Jeff P. Hultquist, and Henry Fuchs, *Fast Constructive-Solid Geometry Display in the Pixel-Powers Graphics System*, Computer Graphics (Proceedings of ACM SIGGRAPH), 1986, pp. 107–116.
- C. Garth and K.I. Joy, *Fast, memory-efficient cell location in unstructured grids for visualization*, Visualization and Computer Graphics, IEEE Transactions on **16** (2010), no. 6, 1541–1550.

References II


Marc Schirski, *Interactive particle tracing for the exploration of flow fields in virtual environments*, 2008, pp. X, 139 S. : Ill., graph. Darst.


Nigel Stewart, Geoff Leach, and Sabu John, *Linear-time csg rendering of intersected convex objects*, In 10th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision - WSCG 2002 (2002, 2002, pp. 437–444.