Lecture 12 - Surface Reconstruction

Philip Caplan

CSCI 0422 - Geometric Modeling (Spring 2022)
Learning objectives

By the end of this lecture you will be able to:

- define and visualize an isosurface on a 3d grid,
- reconstruct a surface from a point cloud and an implicit function using the Marching Tetrahedra algorithm,
- discuss software development culture.
Getting started...

Switch Host & Client today!

```bash
$ git pull
$ make update
$ cmake .
$ make template_class12_isosurface
$ cmake .
```

Compiling and running the exercise:

```bash
$ make class12_isosurface
```

Compiling and running the solution (after class):

```bash
$ make class12_isosurface_sol
```
Motivation: supposed you want to visualize constant values of a function.
Warm-up exercise: define the distance to a sphere.

- define a Grid\(<\text{Tet}>\) mesh,
- compute the distance from every vertex to a sphere with center \((0.5, 0.5, 0.5)\) and radius 0.25,
- create and attach a vertex field to the grid corresponding to the distance,
  use `create_vertex_field(name, vertex_values)`
- visualize the distance in flux360
  (turn off lighting and use the clip feature)
The Marching [insert shape] algorithm for surface reconstruction.

- define a Grid<[insert shape]> mesh,
- define the function value at the vertices of the mesh,
- specify a desired isosurface value (0 if using the distance to the surface),
- for every [insert shape]:
  - determine which vertices of the [insert shape] are < or > isosurface value,
  - "polygonize" the [insert shape] using the function value at the vertices

(CC BY-SA 2.5, https://commons.wikimedia.org/w/index.php?curid=470798)
The Marching Cubes algorithm for surface reconstruction.
The Marching Tetrahedra algorithm for surface reconstruction.
Calculating the coordinates of the intersection point.

- Let vertex coordinates of intersected edge be \((e_0, e_1)\),
- Let function values at vertices of intersected edge by \((f_0, f_1)\),
- First: check if we already create an intersection point on edge,
- Otherwise, compute intersection coordinates:

\[
\begin{align*}
  w_0 &= \frac{d_1}{d_0 + d_1}, \\
  w_1 &= \frac{d_0}{d_0 + d_1} \\
  x &= w_0 e_0 + w_1 e_1
\end{align*}
\]

where \(d_0 = |f_0|\) and \(d_1 = |f_1|\).
Surface reconstruction examples.

- using analytic distance to a sphere
  (20 x 20 x 20 grid)

- using distance to a mesh
  (50 x 50 x 50 grid)
Calculating the distance to a mesh.

1. read in a mesh and calculate vertex normals,
2. scale the initial grid so the mesh fits inside,
3. build a kd-tree of the surface mesh points,
4. for each point in the grid (denote as \( \mathbf{g} \)):
   - use the kd-tree to find the closest point on the surface mesh, denote as \( \mathbf{c} \),
   - denote the normal at this closest point as \( \mathbf{n} \),
   - calculate distance: \( d = (\mathbf{c} - \mathbf{g}) \cdot \mathbf{n} \)

note: improvements are possible, particularly in the search for the closest surface mesh point.
Software Development Culture
Software development is a team sport.

Please read the following excerpts (one per group of 4-5) and consider the following:

- why is the story/theme important in the context of software development?
- have you had any similar experiences (classes, internship) before?
- discuss the different points of view of the excerpt,

  please assign a volunteer(s) to summarize the story and share any thoughts your group had
We lived through the early 1990s and witnessed the amazing run of championships by the Chicago Bulls. We were both in Chicago during this period, and the national media was saturated for years with stories about this amazing team.

What did we mostly hear about on TV and in newspapers? Not the team, but Michael Jordan, the superstar. Every player around the world wanted to be MJ. We watched him dance circles around other players. We watched him in television commercials. We went to see silly movies where he played basketball with cartoon characters. He was a star, and every kid on every court practicing hoops secretly wished to grow up and follow his path.

Programmers have that same instinct—to find idols and worship them. Linus Torvalds, Richard Stallman, Bill Gates—all heroes who changed the world with heroic feats. Linus wrote Linux by himself, right?
Excerpt #1: the Genius myth & impostor syndrome
Imagine you’re a bicycle-design enthusiast, and one day you get a brilliant idea for a completely new way to design a gear shifter. You order parts and proceed to spend weeks holed up in your garage trying to build a prototype. When your neighbor—also a bike advocate—asks you what’s up, you decide not to talk about it. You don’t want anyone to know about your project until it’s absolutely perfect. Another few months go by and you’re having trouble making your prototype work correctly. But because you’re working in secrecy, it’s impossible to solicit advice from your mechanically inclined friends.

Then one day your neighbor pulls their bike out of the garage with a radical new gear-shifting mechanism. Turns out they’ve been building something very similar to your invention, but with the help of some friends down at the bike shop. At this point you’re exasperated. You show them your work. They point out that your design had some simple flaws—ones that might have been fixed in the first week if you had shown them.
Excerpt #2: Hiding your code until it’s "ready"
John Tukey almost always dressed very casually. He would go into an important office and it would take a long time before the others realized that this is a first-class person and they had better listen. For a long time John had to overcome this kind of hostility. It’s wasted effort! I didn’t say you should conform; I said, ‘The appearance of conforming gets you a long way.’ If you chose to assert your ego in any number of ways, ‘I am going to do it my way,’ you pay a small steady price throughout the whole of your professional career. And this, over a whole lifetime, adds up to an enormous amount of needless trouble. […] By realizing you have to use the system and studying how to get the system to do your work, you learn how to adapt the system to your desires. Or you can fight it steadily, as a small, undeclared war, for the whole of your life.
Suppose you start a new job as a programmer. After your first week you really started digging into the code base. Because you care about what was going on, you started gently questioning other teammates about their contributions. You send simple code reviews by email, politely asking about design assumptions or pointing out places where logic could be improved. After a couple of weeks you’re summoned to your director’s office. “What’s the problem?” you ask. “Did I do something wrong?” The director looked concerned: “We’ve had a lot of complaints about your behavior. Apparently you’ve been really harsh toward your teammates, criticizing them left and right. They’re upset. You need to tone it down.” You’re baffled. In a strong culture based on Humility, Respect and Trust, your code reviews should have been welcomed and appreciated by your peers.
Excerpt #4: Learn to Both Deal Out and Handle Criticism
There's a well-known (and clichéd) urban legend in the business world about a manager who makes a mistake and loses an impressive $10 million. They dejectedly go into the office the next day and start packing up their desk, and upon getting the inevitable “the CEO wants to see you” call, the manager trudges into the CEO’s office and quietly slides a piece of paper across the desk to the CEO.

“What’s this?” asks the CEO.

“My resignation,” says the exec. “I assume you called me in here to fire me.”

“Fire you?” responds the CEO, incredulously. “Why would I fire you? I just spent $10 million training you!”
HRT ❤: The three pillars

- **Humility:**
  You are not the center of the universe. You’re neither omniscient nor infallible. You’re open to self-improvement.

- **Respect:**
  You genuinely care about others you work with. You treat them as human beings, and appreciate their abilities and accomplishments.

- **Trust:**
  You believe others are competent and will do the right thing, and you’re OK with letting them drive when appropriate.

  Your self-worth shouldn’t be connected to the code you write.
  You are not your code.
Your TODO list . . .

- nothing added to flux-base that requires testing today,
- finish Project 2!
- for further reading, see Chapter 2 in *Software Engineering at Google*:
  https://www.oreilly.com/library/view/team-geek/9781449329839/ch01.html
  (full textbook posted on Canvas in files->books)