Designing and Implementing a Polygon Mesh Library: Can Rust Improve the Status Quo in the Domain of Geometry Processing?

2019-06-12

Lukas Kalbertodt

WG Graphics & Geometric Computing

Designing and Implementing a Polygon Mesh Library: Can Rust Improve the Status Quo in the Domain of Geometry Processing?

(A) Basics

Meshes, geometric computing, data structures, existing libraries

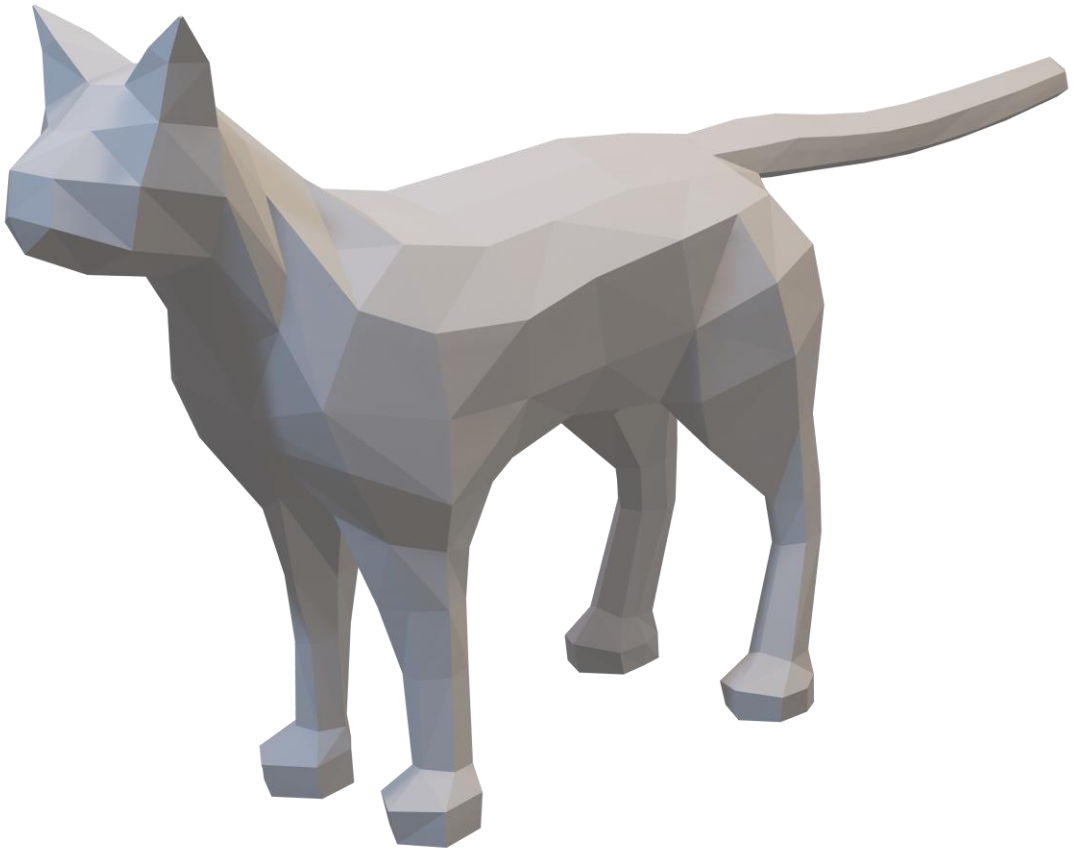
(B) The Library „lox“

Design, Example, Demonstration

(C) Evaluation and Future Work

Benchmarks, problems with Rust, further plans

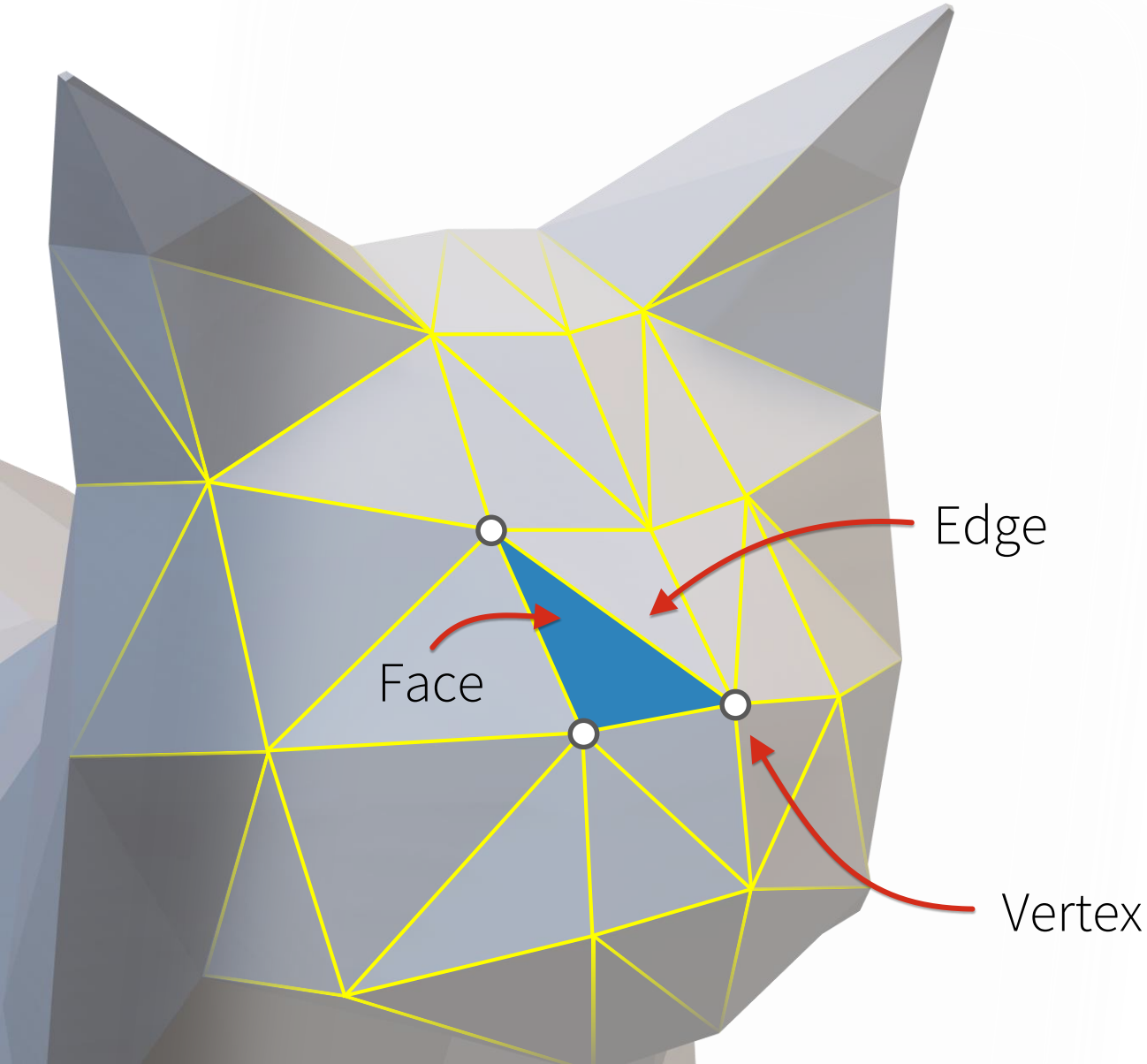
Basics: Meshes



- Use cases:
  - 3D-real time applications (e.g. games)
  - Geometry computing
  - Simulations
  - Processing of 3D sensor data
  - 3D printing
  - …

Cat model by „volkanongun“, CC-BY 4.0
https://sketchfab.com/models/1e7143dfafd04ff4891efcb06949a0b4

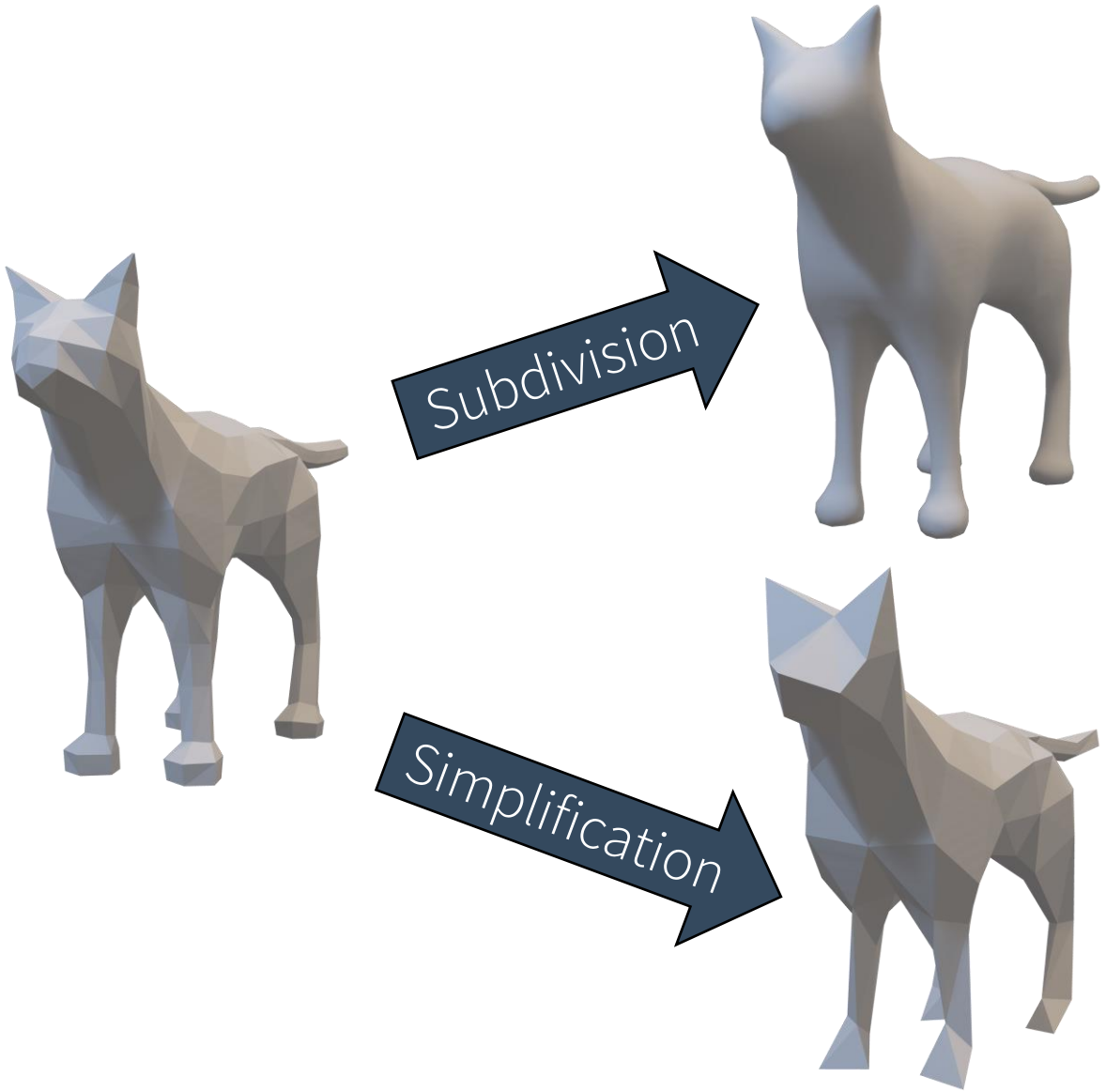
Basics: Meshes



- Arbitrary Data per Element
  - *Vertex*: position, normal, color, textur coordinate, …
  - *Face*: normal, color, …
  - *Edge*: color, costs, …
- Special cases:
  - Triangle Mesh △
  - Quad Mesh □

Cat model by „volkanongun“, CC-BY 4.0
https://sketchfab.com/models/1e7143dfafd04ff4891efcb06949a0b4

Algorithms



- More algorithms:
  - Remeshing
  - Repair
  - Various calculations
  - …

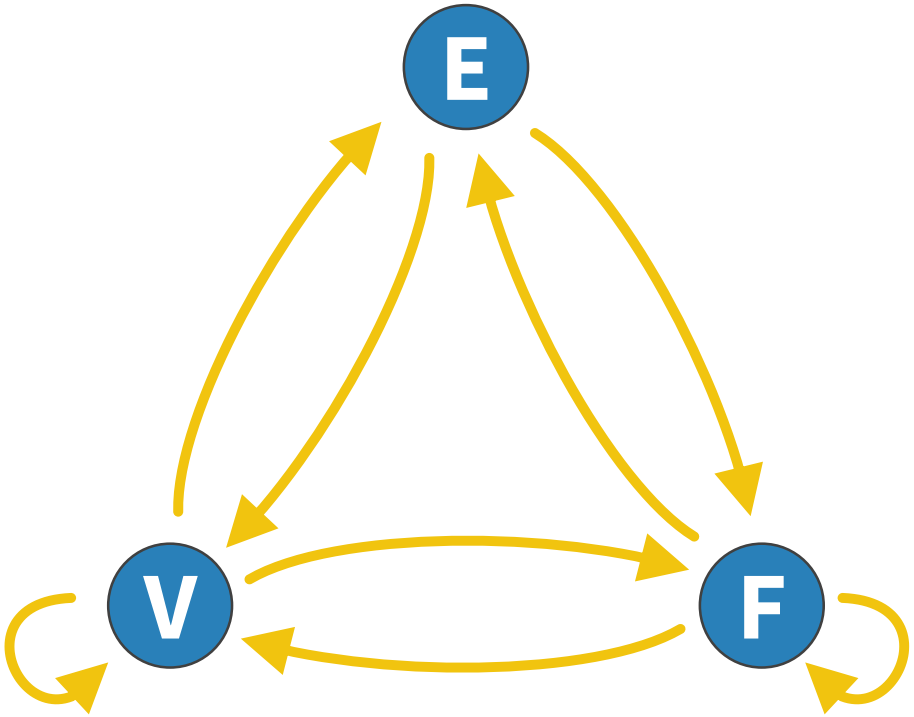
Algorithms need adjacency-information!

Mesh Data Structure

Requirements:

- *Fast* access to adjacency information
- *Fast* modifications
- Memory efficient

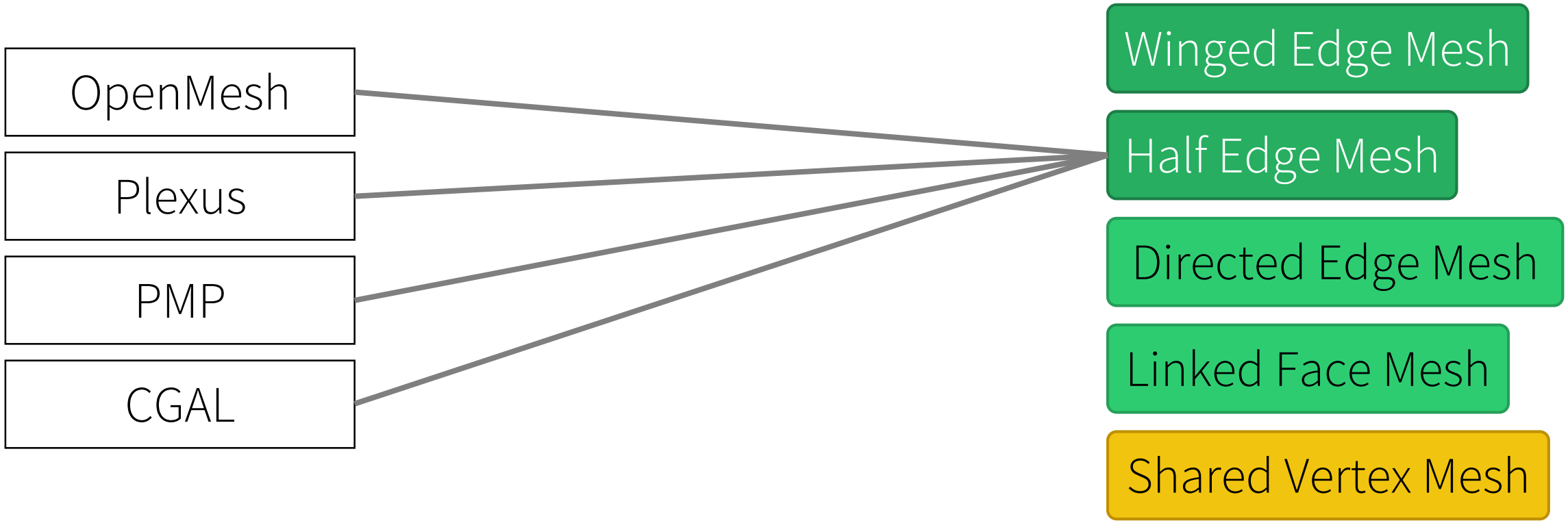
Design not trivial!



Existing Data Structures

Adjacency info & edges	Half Edge Mesh, Winged Edge Mesh
Full adjacency info	Linked Face Mesh, Directed Edge Mesh
Partial adjacency info	Shared Vertex Mesh
	Triangle Soup, Vertex-Vertex Mesh

Existing Data Structures



Why only one DS?

The Library „lox“

- Open Source library (MIT & Apache 2.0)
- The core: abstraction over data structures

Goal: better™ than OpenMesh/PMP!

Abstractions in C++ and Rust

C++

Abstract Class

```
class Mesh {
public:
    virtual VertexHandle
        addVertex() = 0;
}


void smooth(Mesh& m) { … }
```

- Overhead (virtual call)
- Limited

Templates

```
// No „interface“ definition




template <typename M>
void smooth(M& m) { … }
```

- No formal „Interface“
- Often bad error mesages

Rust

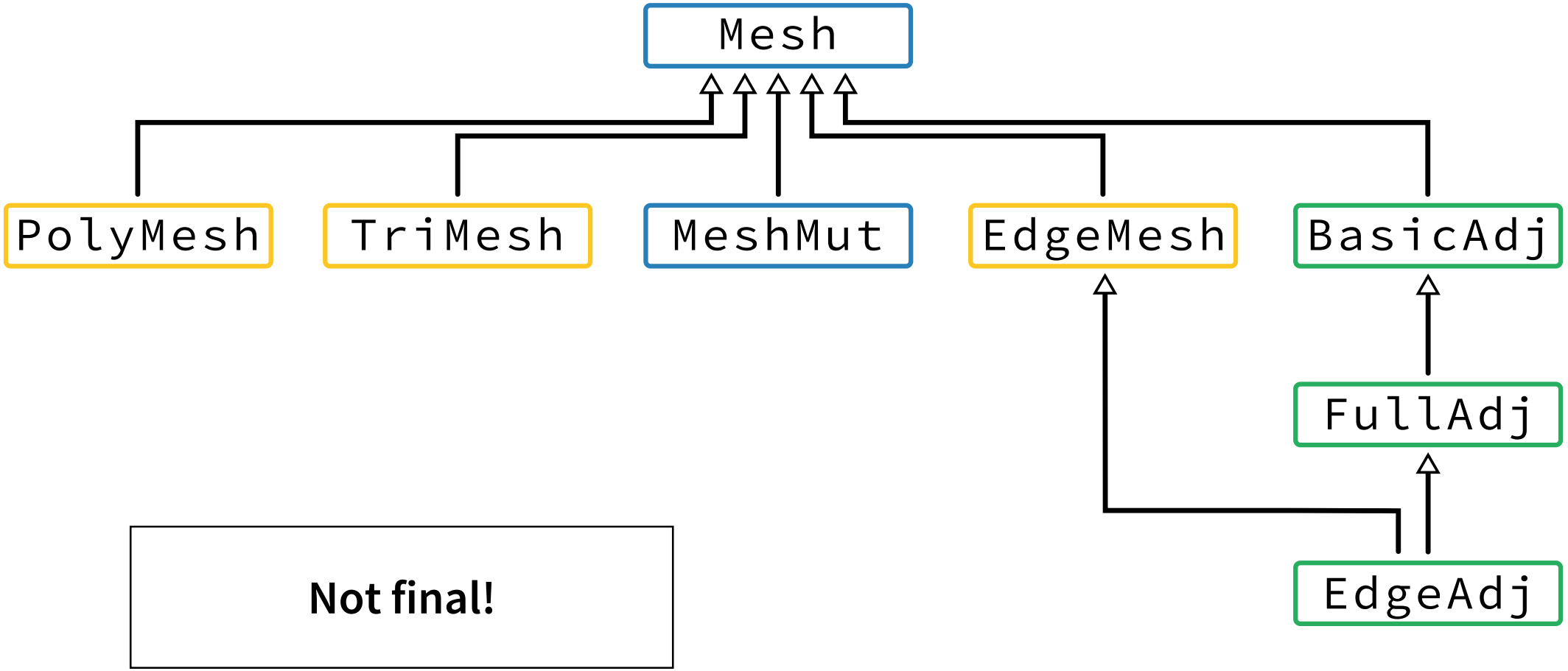
Traits

```
trait Mesh {
    fn add_vertex(&mut self)
        -> VertexHandle;
}

fn smooth<M>(m: &mut M)
    where M: Mesh,
{ … }
```

- No overhead
- Mostly good error messages

Mesh Traits


Mesh
PolyMesh
TriMesh
MeshMut
EdgeMesh
BasicAdj
FullAdj
EdgeAdj
Not final!

Example and Demonstration

```rust
use lox::{
    algo,
    ds::{HalfEdgeMesh, half_edge::TriConfig},
    fat::MiniMesh,
    io,
};

fn main() -> Result<(), io::Error> {
    type MyMesh = MiniMesh<HalfEdgeMesh<TriConfig>>;

    let mut mesh: MyMesh = io::read_file("input.stl")?;
    algo::subdivision::sqrt3(&mut mesh.mesh, &mut mesh.vertex_positions, 1);
    io::write_file(&mesh, "output.ply")?;

    Ok(())
}
```

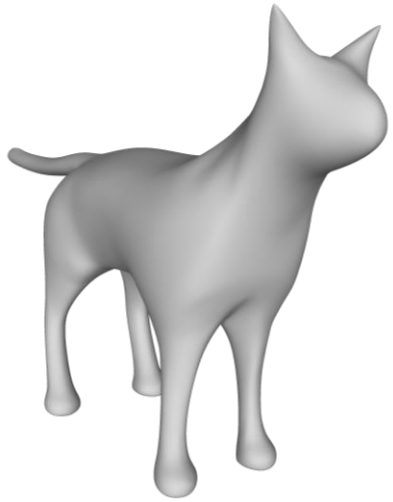
Mini-Demo

Benchmarks

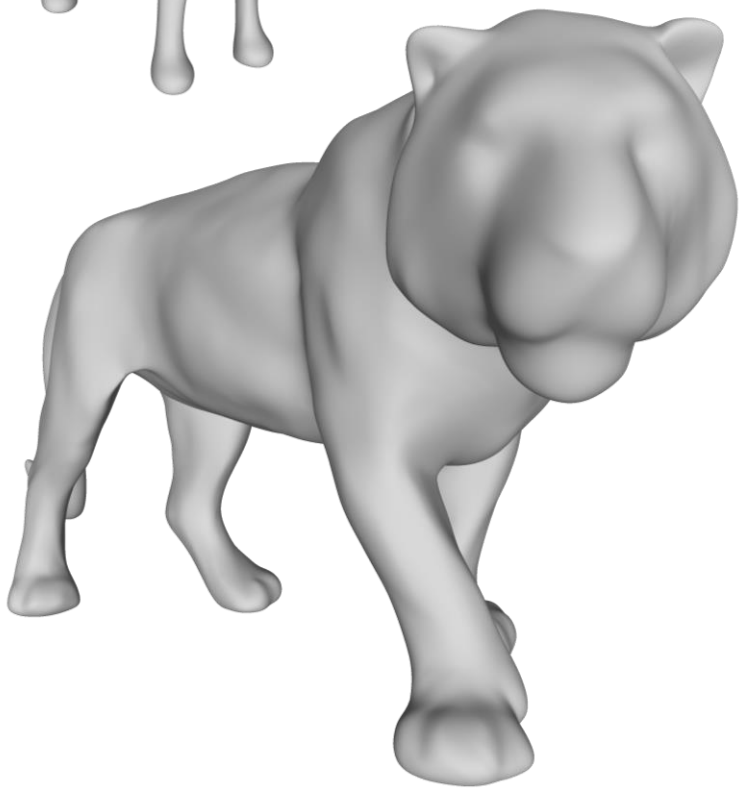
- „Cat“ and „Tiger“ Mesh
- IO benchmarks use a RAM-disk
- 100 or 500 iterations per benchmark

- All details in the written thesis and in the linked repository

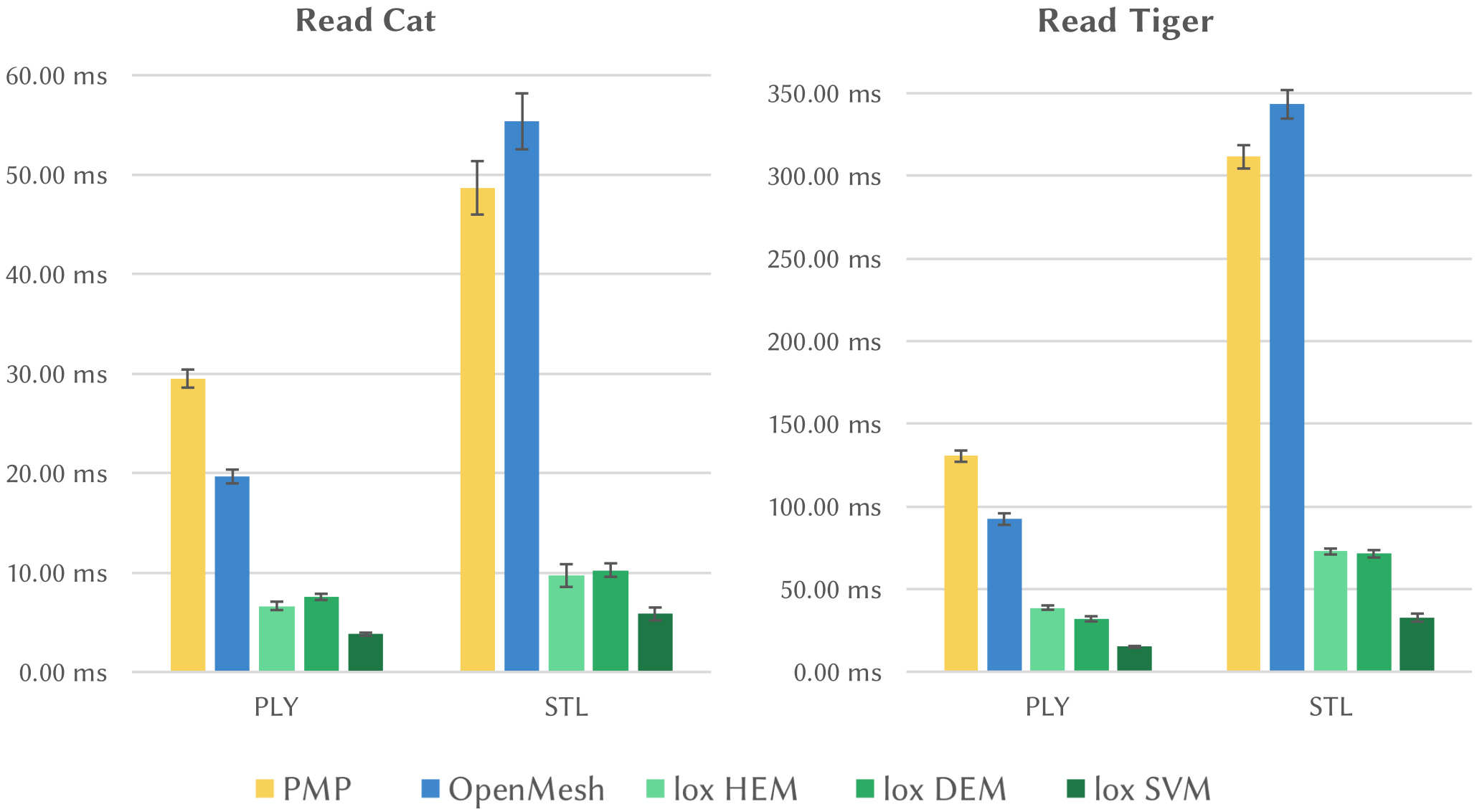
Cat
59 292 Faces



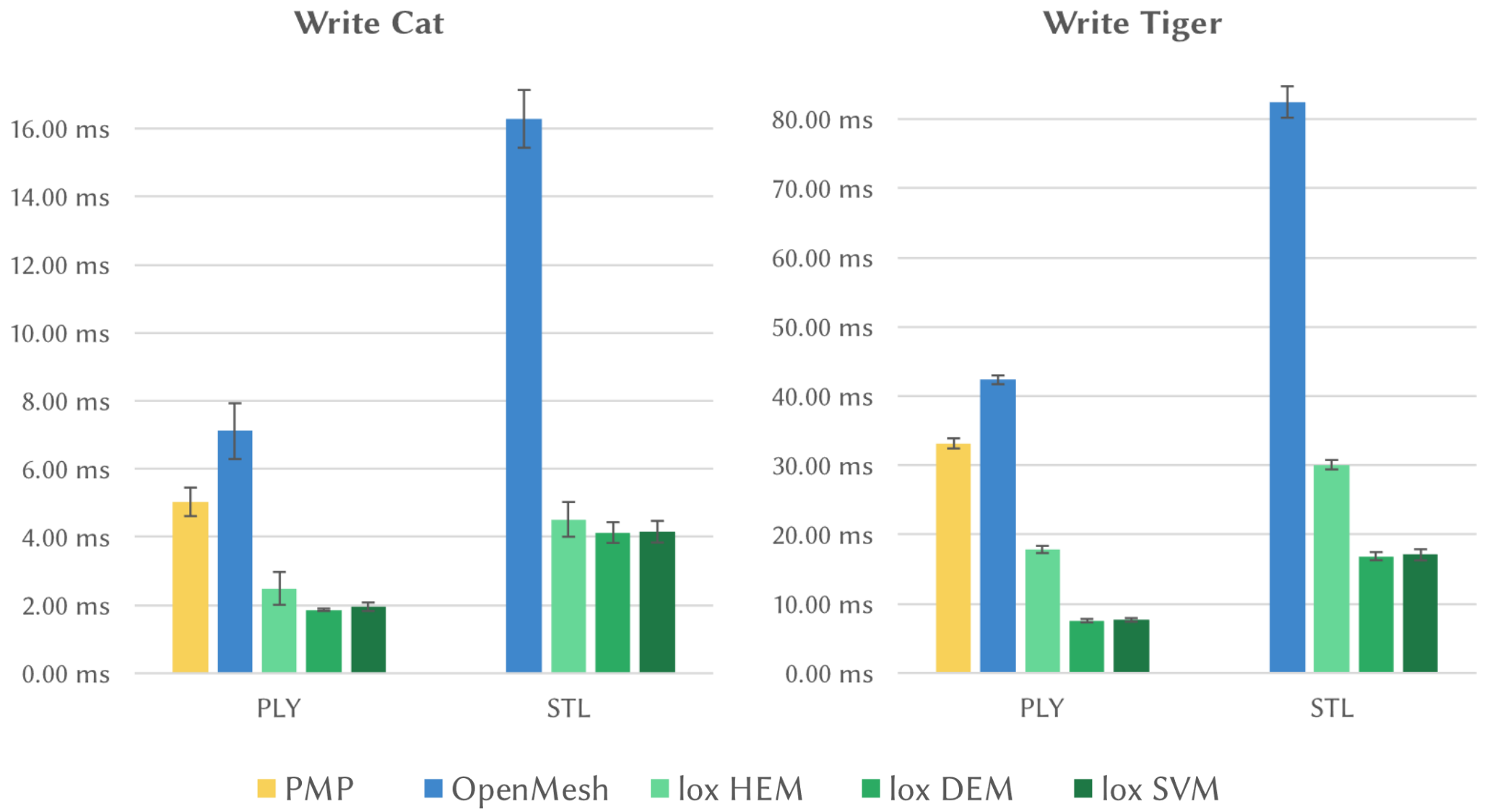
Tiger
232 956 Faces



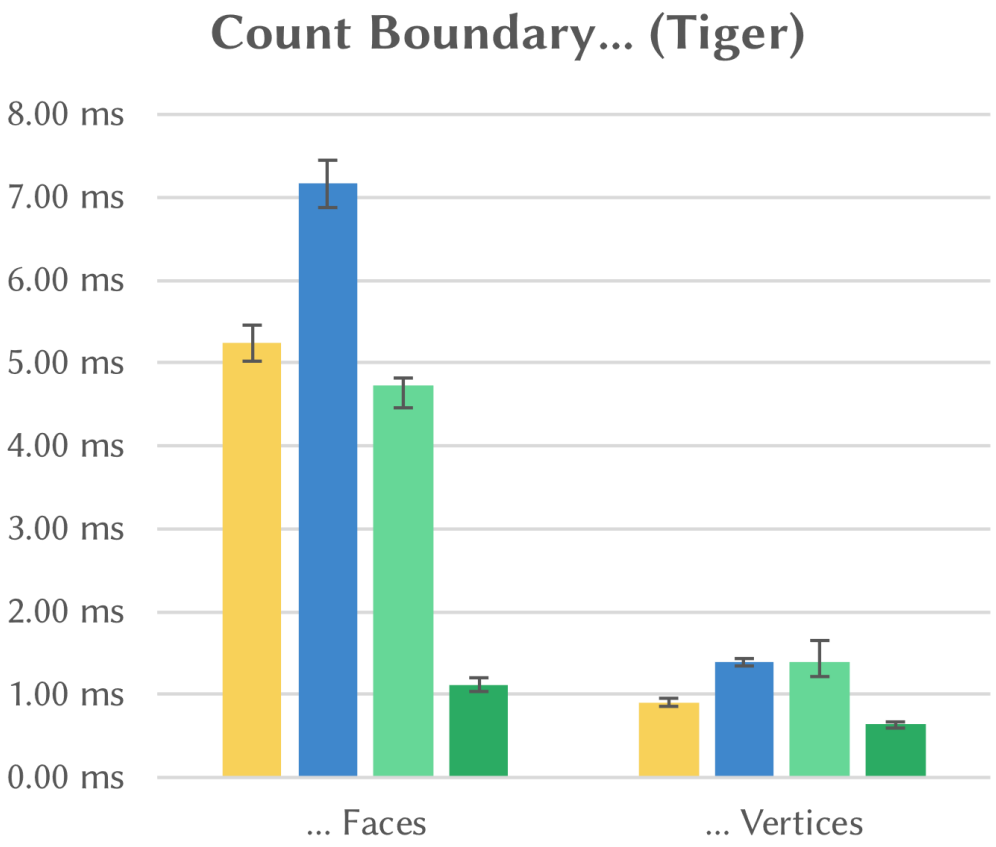
Benchmark: IO (Read)

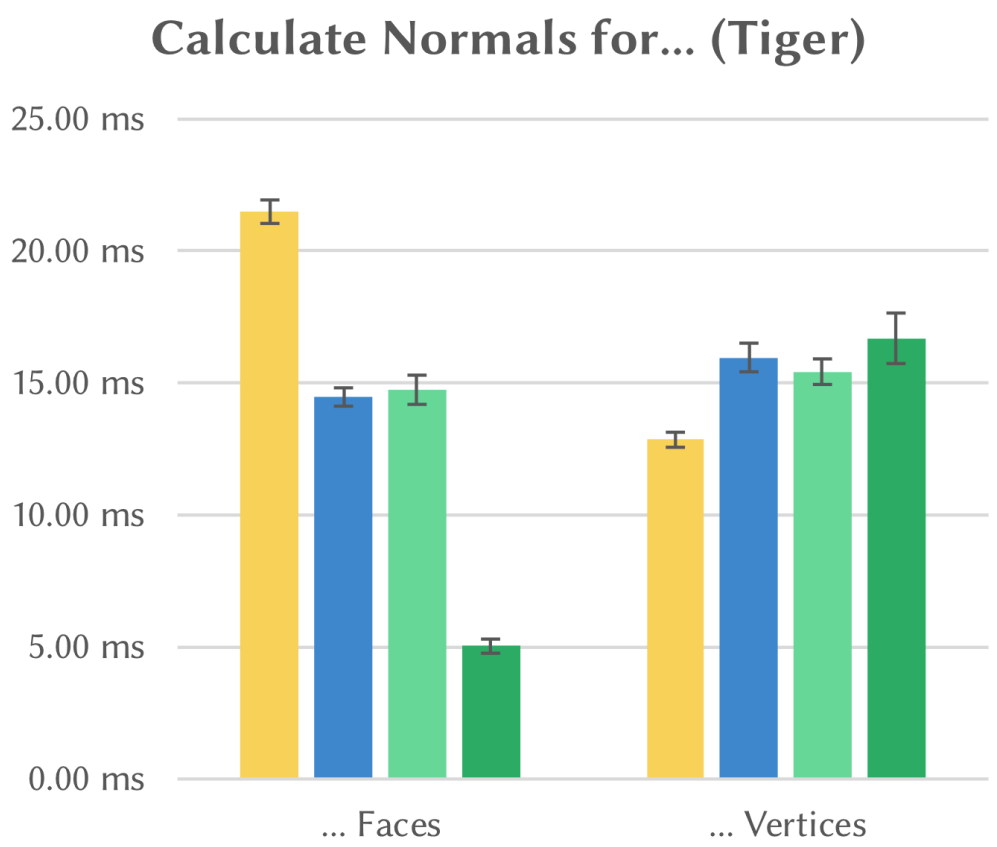


Benchmark: IO (Write)



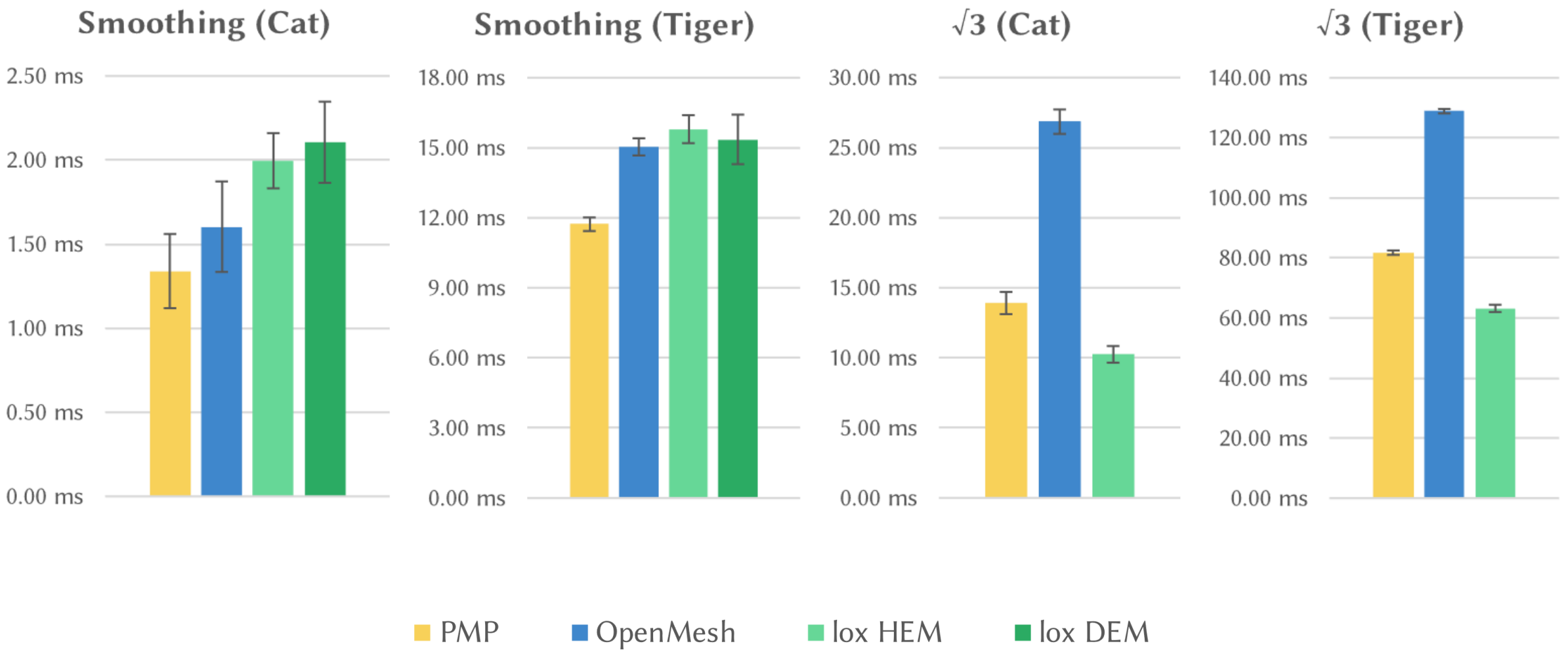
Benchmark: Algorithms





PMP OpenMesh lox HEM lox DEM

Benchmark: Algorithms



Problems with Rust

- Some type system features are still missing
  - Generic Associated Types (GATs)
  - Specialization
- Unnecessary Bound Checks
- „Aliasing *nand* Mutability“ problematic in some cases

```
for f in mesh.faces() {
    mesh.add_vertex();
}
```

Further Plans and Summary

- Further developing the library
  - Add more: data structures, file formats, algorithms, …
  - Initial release & using feedback by users
  - Experiments with Rust‘s type system
- Comparing all data structures

- Fairly high speed
- Abstraction works!
- Short & concise code

Thanks for your attention!

Questions?

https://github.com/LukasKalbertodt/masters-thesis

Vielen Dank für die Aufmerksamkeit!

Fragen?

https://github.com/LukasKalbertodt/masters-thesis

Referenzen & Quellen

- Cat model by „volkanongun“, CC-BY 4.0, https://sketchfab.com/models/1e7143dfafd04ff4891efcb06949a0b4
- Tiger model by „Jérémie Louvetz “, CC-BY-NC 4.0, https://sketchfab.com/3d-models/sumatran-tiger-95c4008c4c764c078f679d4c320e7b18
- OpenMesh: https://openmesh.org/
- PMP (Polygon Mesh Processing): http://www.pmp-library.org/
- CGAL: https://www.cgal.org/
- SIEGER, Daniel; BOTSCH, Mario. Design, implementation, and evaluation of the surface_mesh data structure. In: Proceedings of the 20th international meshing roundtable. Springer, Berlin, Heidelberg, 2011. S. 533-550.
- KOBBELT, Leif. √ 3-subdivision. In: Proceedings of the 27th annual conference on Computer graphics and interactive techniques. ACM Press/Addison-Wesley Publishing Co., 2000. S. 103-112.