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

12.06.2019

Lukas Kalbertodt

AG Computergrafik

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

(A) Grundlagen

Meshes, Geometrieverarbeitung, Datenstrukturen, existierende Bibliotheken

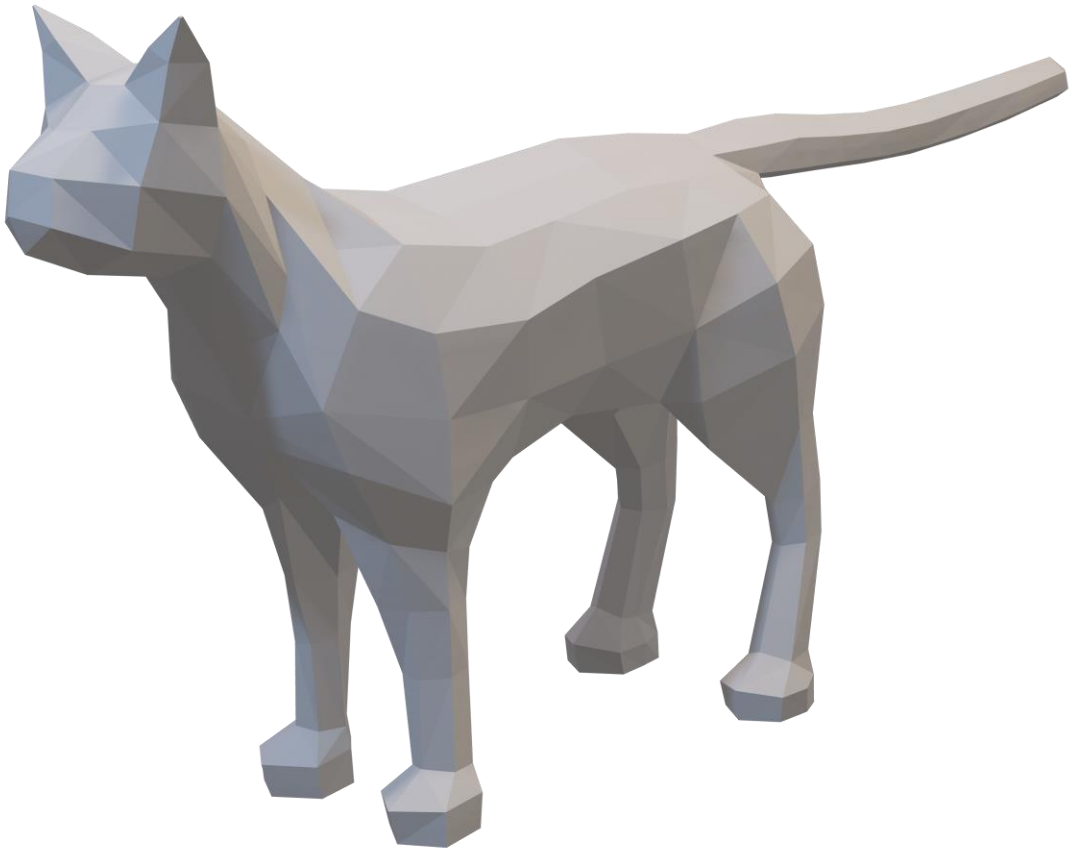
(B) Die Bibliothek „lox“

Design, Beispiel, Demo

(C) Evaluation und Ausblick

Benchmarks, Probleme mit Rust, weitere Pläne

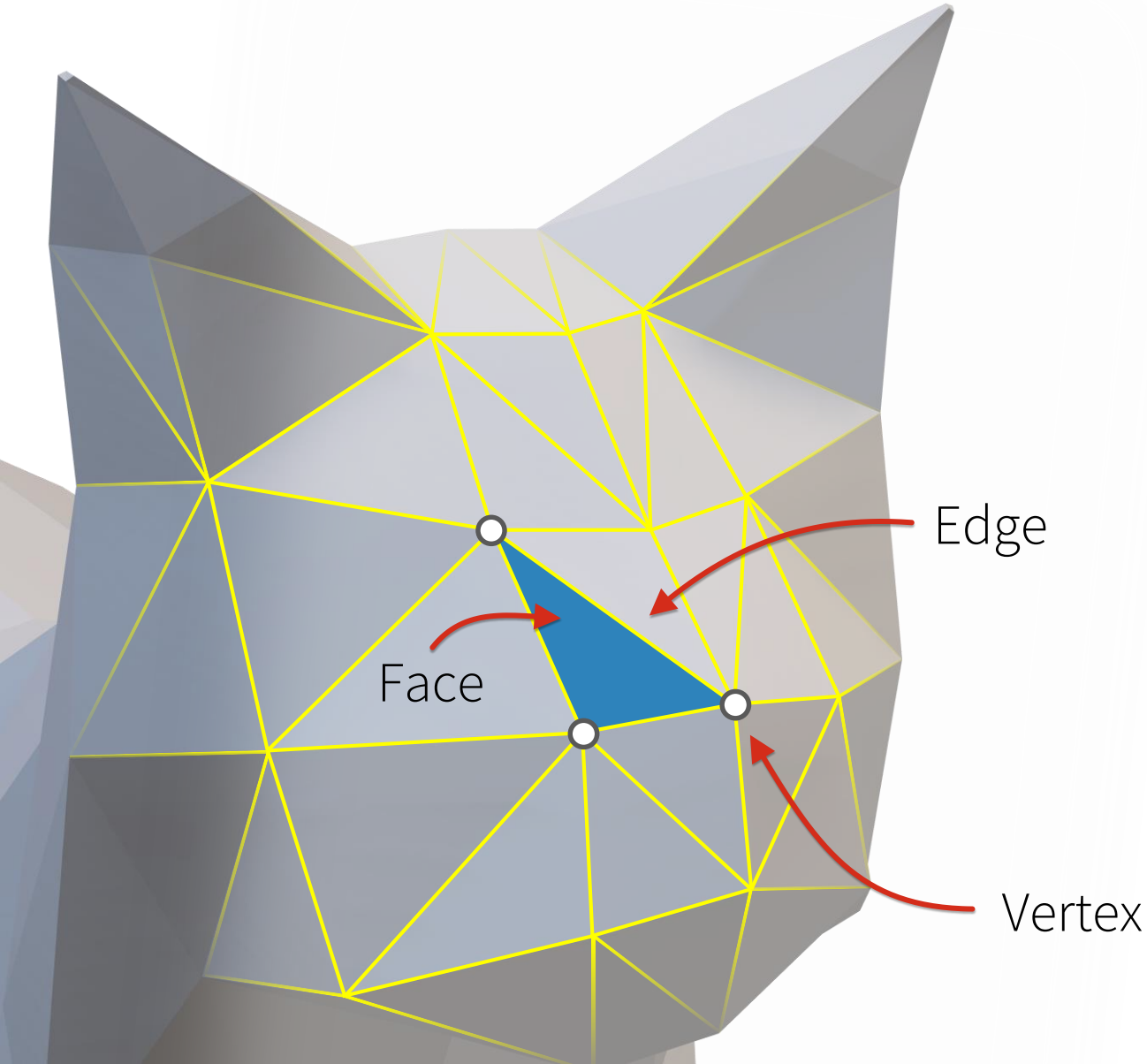
Grundlagen: Meshes



- Verwendung:
  - 3D-Echtzeitanwendungen (insb. Spiele)
  - Geometrieverarbeitung
  - Simulationen
  - Verarbeitung von 3D-Sensordaten
  - 3D-Druck
  - …

Katzenmodell von „volkanongun“, CC-BY 4.0
https://sketchfab.com/models/1e7143dfafd04ff4891efcb06949a0b4

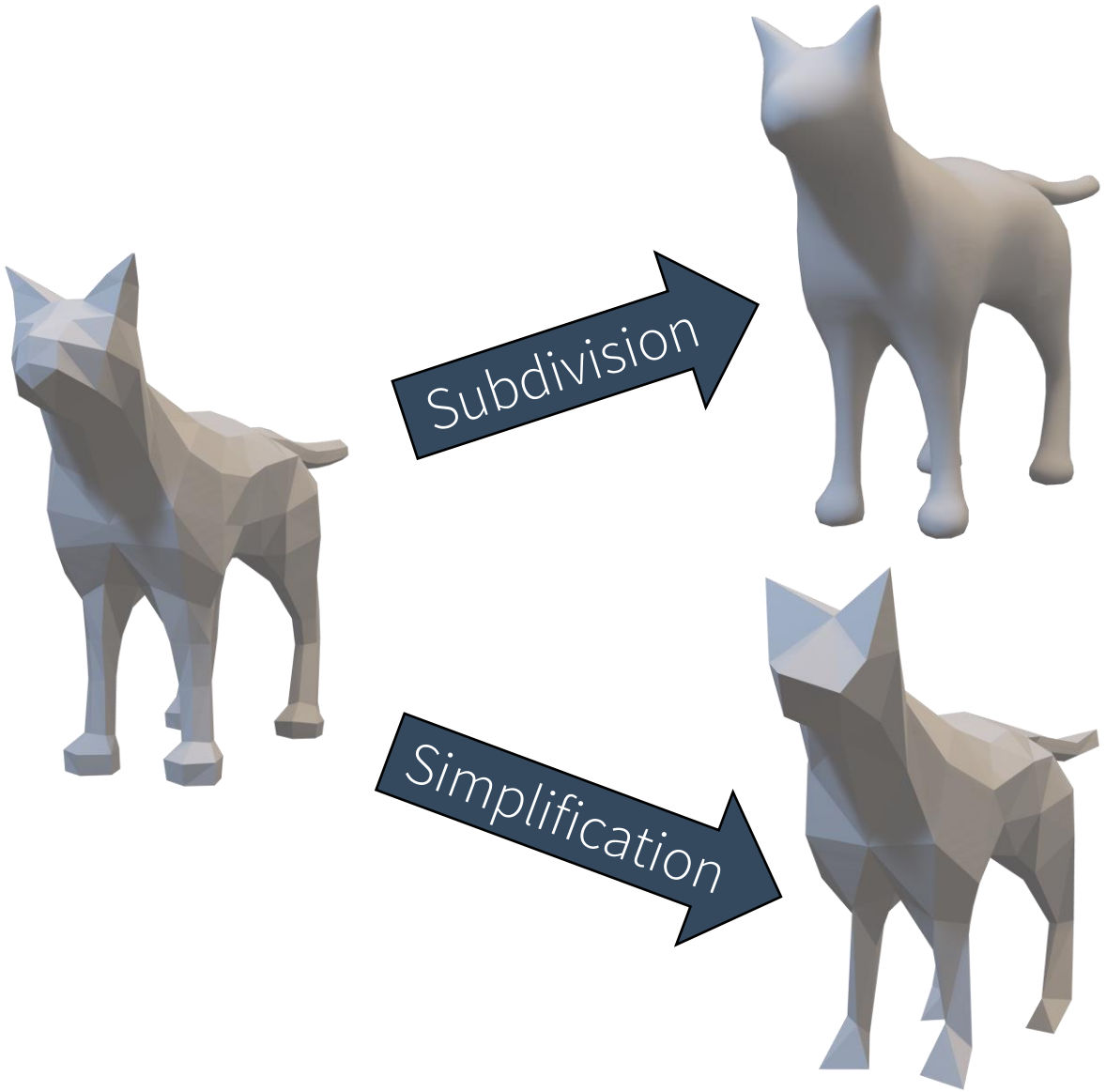
Grundlagen: Meshes



- Beliebige Daten pro Element
  - *Vertex*: Position, Normale, Farbe, Textur Koordinaten, …
  - *Face*: Normale, Farbe, …
  - *Edge*: Farbe, Kosten, …
- Spezielle Meshes:
  - Triangle Mesh △
  - Quad Mesh □

Katzenmodell von „volkanongun“, CC-BY 4.0
https://sketchfab.com/models/1e7143dfafd04ff4891efcb06949a0b4

Algorithmen



- Andere Algorithmen:
  - Remeshing
  - Reperaturen
  - Berechnungen
  - …

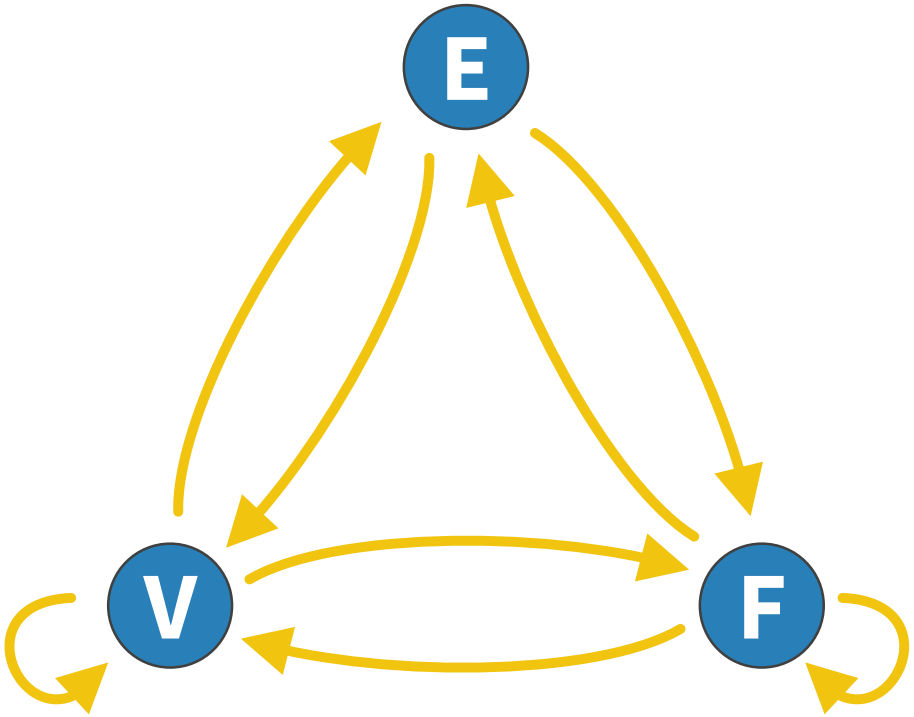
Algorithmen brauchen Adjazenzinformationen!

Mesh-Datenstruktur

Anforderungen:

- *Schneller* Zugriff auf Adjazenzinfos
- *Schnelle* Modifikationen
- Speichereffizient

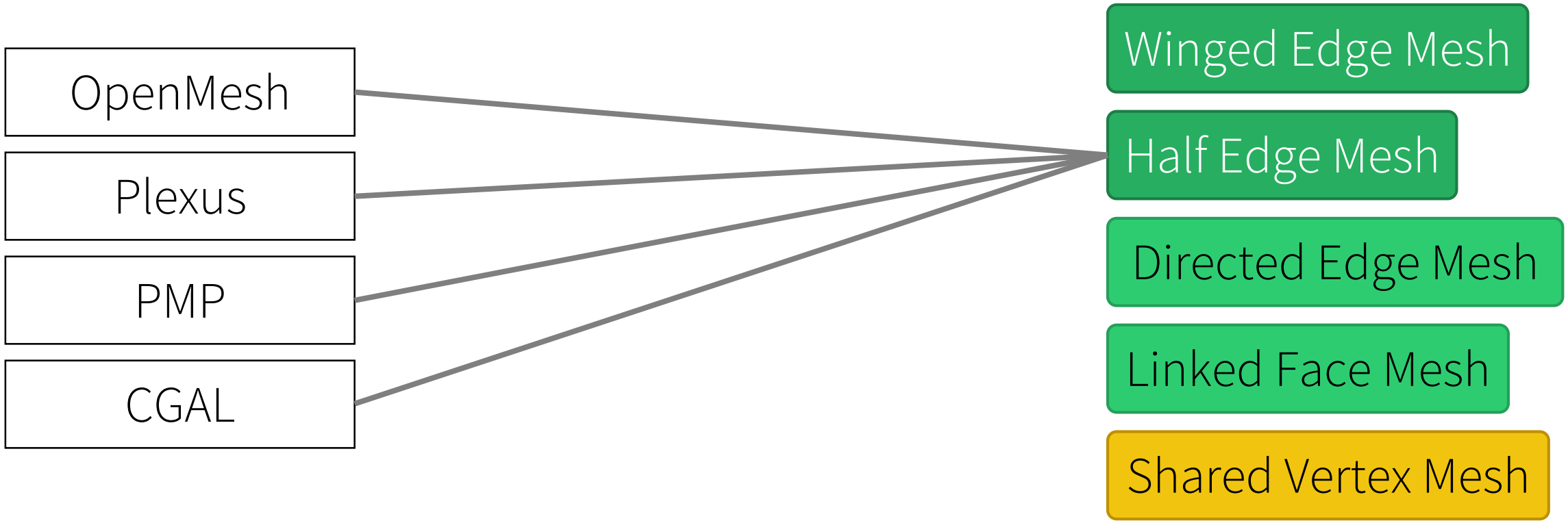
Nicht trivial!



Existierende Datenstrukturen

Adjazenzinfos & Edges	Half Edge Mesh, Winged Edge Mesh
Volle Adjazenzinfos	Linked Face Mesh, Directed Edge Mesh
Partielle Adjazenzinfos	Shared Vertex Mesh
	Triangle Soup, Vertex-Vertex Mesh

Existierende Datenstrukturen



Warum nur eine DS?

Die Bibliothek „lox“

- Open Source Bibliothek (MIT & Apache 2.0)
- Der Kern: Abstraktion über Datenstrukturen

Ziel: Besser™ als OpenMesh/PMP!

Abstraktion in C++ und Rust

C++

Abstrakte Klasse

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


void smooth(Mesh& m) { … }
```

- Overhead (virtual call)
- Limitiert

Templates

```
// No „interface“ definition




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

- Kein „Interface“
- Schlechte Fehlermeldungen

Rust

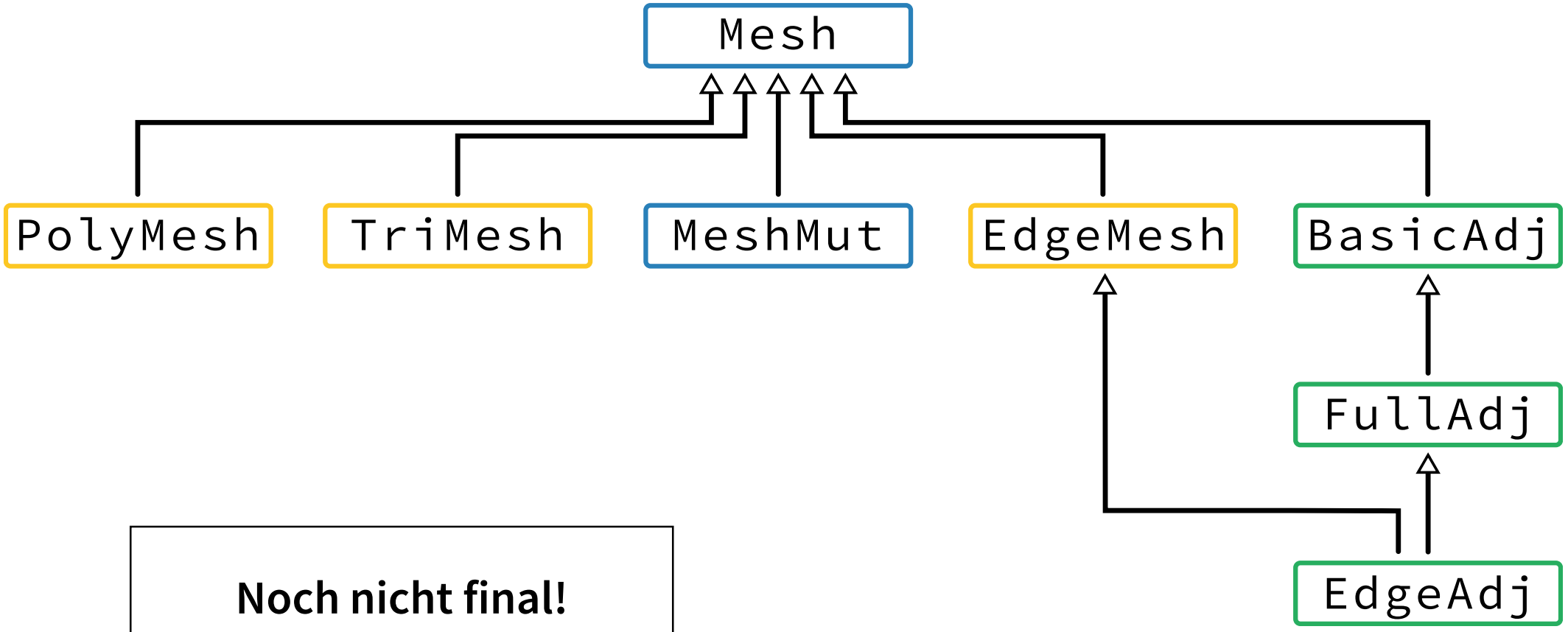
Traits

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

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

- Kein Overhead
- Gute Fehlermeldungen

Mesh Traits



Noch nicht final!

Beispiel und Demo

```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(())
}
```

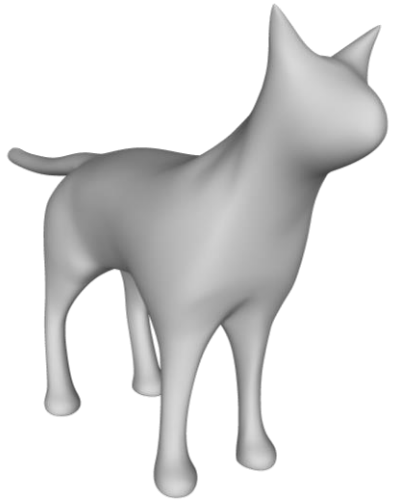
Mini-Demo

Benchmarks

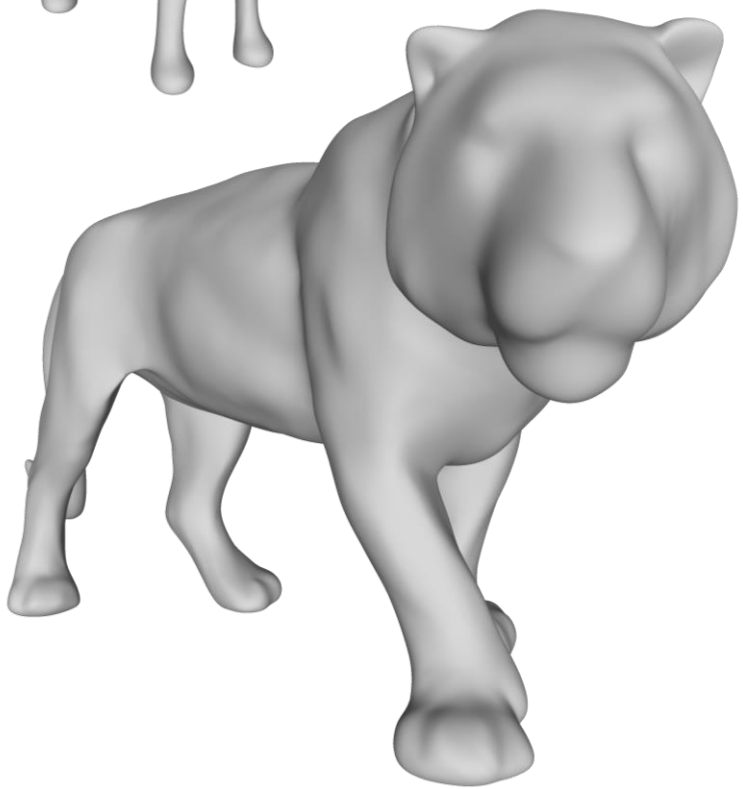
- „Cat“ und „Tiger“ Mesh
- IO von RAM-Disk
- 100 bzw. 500 Wiederholungen pro Benchmark

- Alle Details in der schriftlichen Ausarbeitung

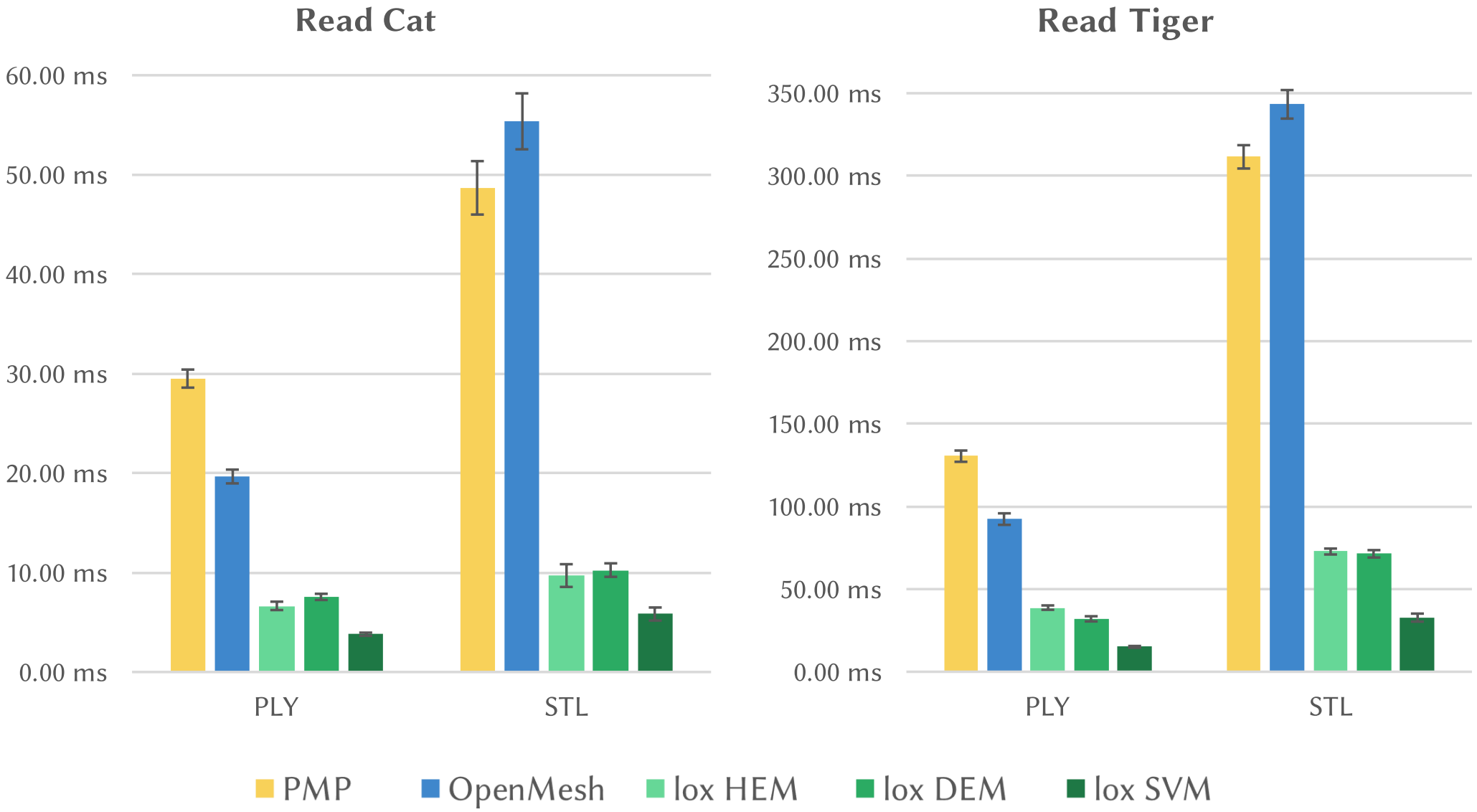
Cat
59 292 Faces



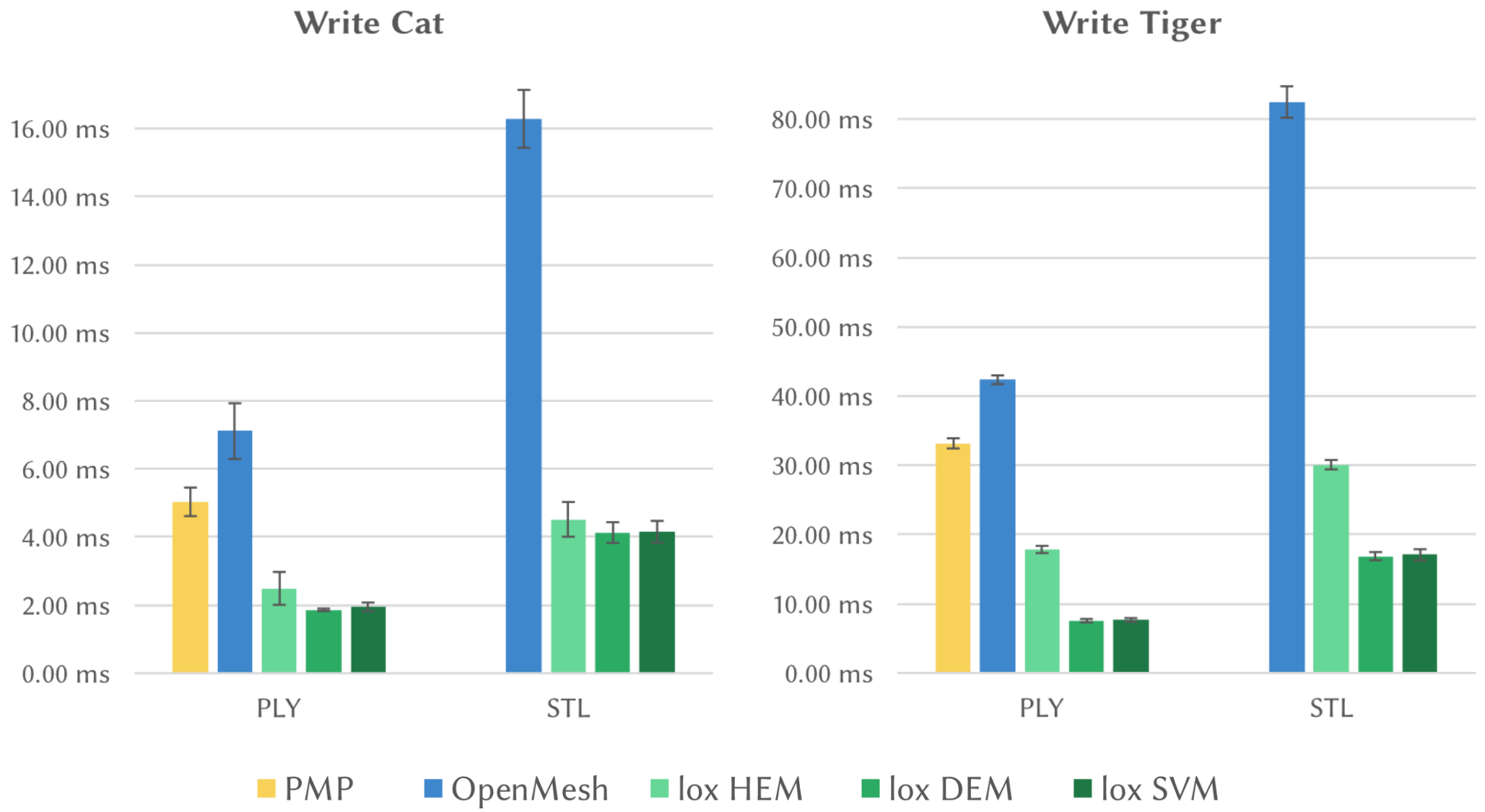
Tiger
232 956 Faces



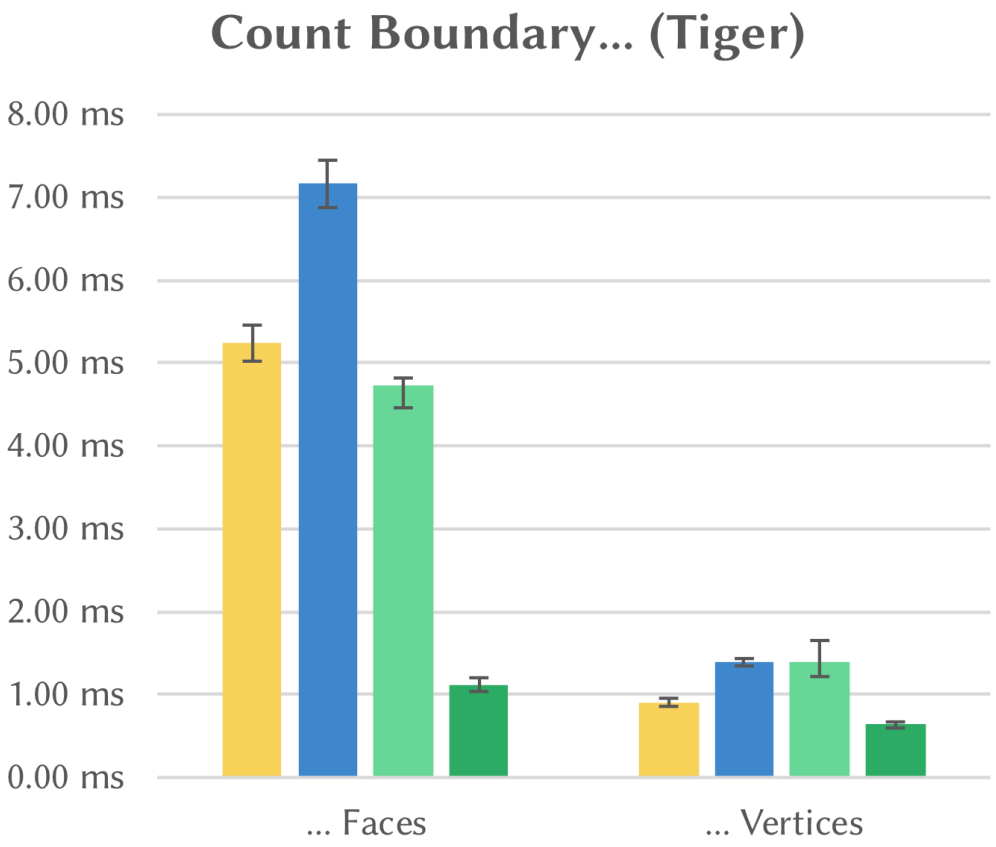
Benchmark: IO (Lesen)

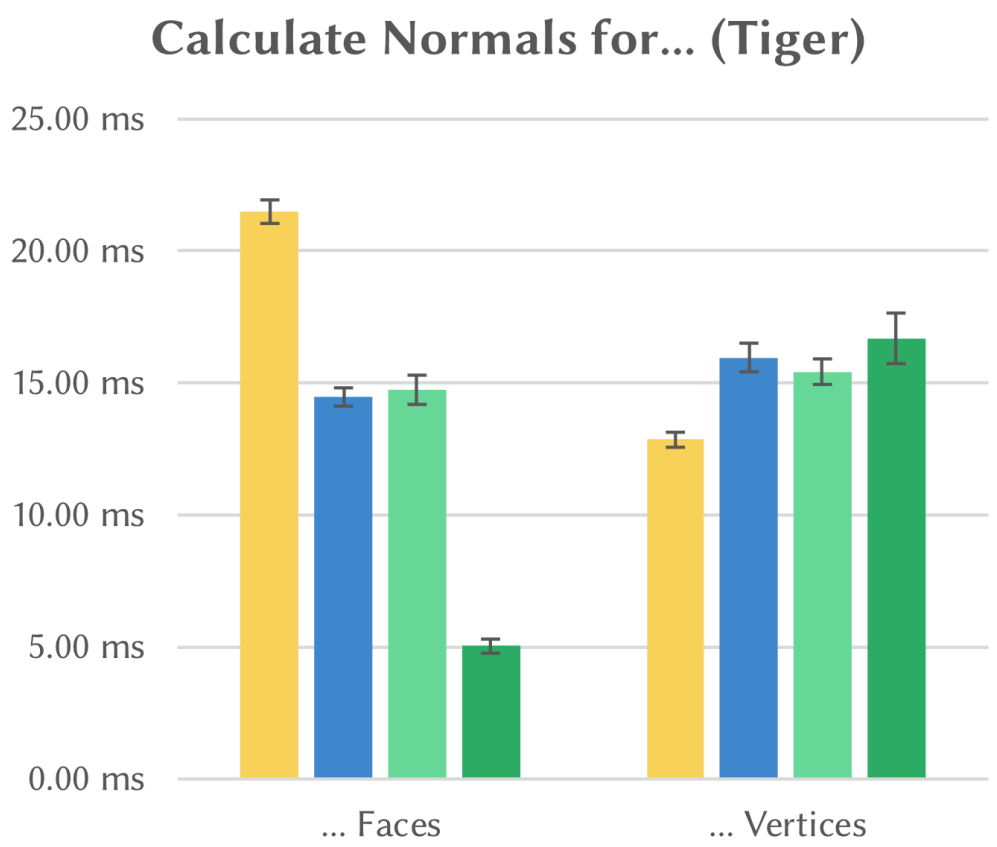


Benchmark: IO (Schreiben)



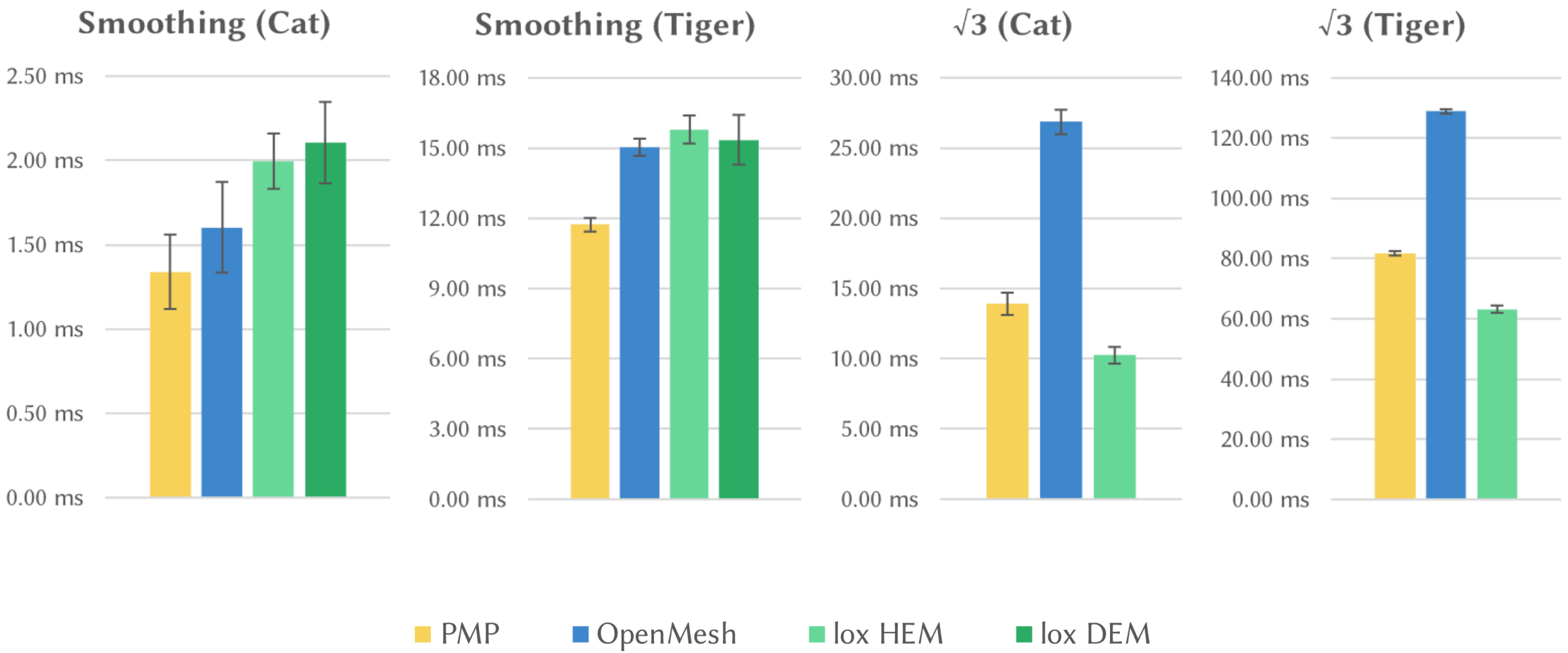
Benchmark: Algorithms





PMP OpenMesh lox HEM lox DEM

Benchmark: Algorithms



Probleme mit Rust

- Einige Typsystem-Features fehlen *noch*
  - Generic Associated Types (GATs)
  - Specialization
- Unnötige Bound Checks
- „Aliasing *nand* Mutability“ stellenweise problematisch

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

Weitere Pläne und Fazit

- Bibliothek weiterentwickeln
  - Vervollständigen: Datenstrukturen, Dateiformate, Algorithmen, …
  - Initiales Release & Nutzerfeedback einarbeiten
  - Experimente mit Typsystem
- Vergleich aller Datenstrukturen

- Hohe Geschwindigkeit
- Abstraktion funktioniert
- Kurzer & lesbarer Code

Vielen Dank für die Aufmerksamkeit!

Fragen?

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

Vielen Dank für die Aufmerksamkeit!

Fragen?

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

Referenzen & Quellen

- Katzenmodell von „volkanongun“, CC-BY 4.0, https://sketchfab.com/models/1e7143dfafd04ff4891efcb06949a0b4
- Tigermodell von „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.