**3. Fazit**

Dieses Kapitel berichtet über aufgetretene Schwachstellen, sowie Vorteile die die entwickelte Sprache im Gegensatz zu den herkömmlichen Alternativen bietet. Außerdem wird über mögliche Features berichtet die zu signifikant Verbesserungen in der Benutzung führen könnten.

**3.1 A**

Grundsätzlich hat dieses Projekt gezeigt, dass es durchaus Sinnvoll sein kann gewisse Problemstellung mit der Sprache 3D-Code zu visualisieren, da sie ohne großes Vorwissen oder Kenntnisse im Bereich Open GL auskommt. Sie ermöglich auf verschiedene Weisen mit den selbstdefinierten Objekten zu interagieren und diese zu verändern. Außerdem lässt sich die Sprache auf verschiedensten Endgeräten ausführen, da sie auf Kotlin basiert.

**3.1 A**

Wie Beispiel A, B und C zeigen ist es mit wenig Aufwand möglich verschiedene komplexe Gebilde darzustellen. Dies wird erreicht indem der Nutzer einer Liste Objekte hinzufügt, diese werden nach der Vollendung des Codes gerendert. Diese Objekte bestehen aus den Daten einer .obj Datei und zusätzlichen Positons/ - Farbwerten.

Ein wesentlicher Vorteil der eigenentwickelten Sprache ist, dass besondere Features sich leicht hinzufügen lassen, die sonst unüblich in normalen Programmiersprachen sind. Des Weiteren ist die Syntax leicht zu verstehen und komplexe System vor dem Nutzer abgekapselt. Ein Nachteil der Sprache 3Dcode ist, dass sie wesentlich langsamer ist als die Sprache worauf 3D-Code aufbaut. Die Sprache ist durchschnittlich um den Faktor X langsamer.

In Anbetracht der oben genannten Punkte, wäre eine andere valide Option einer Entwicklung einer Kotlin Bibliothek, die ähnlich wie die entwickelte Sprache fungiert. Diese Option hätte zu dem jetzigen Entwicklungsstand noch Vorteile gegenüber der selbstentwickelten Sprache. Durch zusätzliche Features der Sprache 3D - Code und weiteren Optionen könnte, könnten die Vorteile kompensiert werden. (Siehe Ausblick)

**3.2 Ausblick**

Damit die Sprache 3D Code einen wesentlichen Vorteil gegenüber anderen Optionen [siehe LINK] hätte, müsste sie in manchen Punkten ergänzt werden.

* Das manuelle Aktualisieren des Anzeigebildes während der Code ausgeführt wird.
* Überschreiben der Kamera Bewegung
* Injektion von eigenen Shaderprogrammen
* Umwandlung des 3D Codes in Maschinensprache, statt Evaluation in Kotlin
* Vererbung/ Polymorphie
* Statische Funktionen / Klassen
* Ausarbeitung der Bibliotheken
* Beleuchtung und Kollisionserkennung
* Eine Entwicklungsumgebung