Abgabemodalitäten

- **Testatstermin:** Mittwoch, 30. November 2016 in der Übungsstunde.
- Zu diesem Termin stellen alle Gruppenmitglieder gemeinsam die Lösungen vor.

Vorbereitung

Die Programmiersprache, das Betriebssystem und die Entwicklungsumgebung sind Ihnen freigestellt. Wir empfehlen allerdings C++ [1], [2]. Das bereitgestellte Framework benutzt C++ 12 Features.

a) Richten Sie sich eine Entwicklungsumgebung ein. Sie können entweder eine integrierte Entwicklungsumgebung (*Visual Studio*, *Eclipse CDT*, ...) [3], oder einen Texteditor (*Vim*, *Emacs*, ...) zusammen mit Make [4] verwenden.

b) Installieren Sie OpenGL. Unter Windows und Linux ist dies in der Regel **nicht** nötig. Auf der offiziellen Homepage [5] finden Sie alle Downloads und auch Hilfestellung. Das OpenGL-Wiki [6] gibt eine detaillierte Anleitung für fast alle Betriebssysteme.

c) Um die OpenGL Programmierung zu vereinfachen verwenden wir die `Free-GLUT`-Bibliothek. Unter [7] finden Sie sowohl den Source Code als auch vorkompilierte Bibliotheksdateien (siehe Prepackaged Releases) und Tutorials zum Einrichten. Wenn Sie die vorkompilierten Windows Bibliotheksdateien nutzen, müssen Sie nur den heruntergeladenen *freeglut* Ordner in den selben Ordner wie das Framework ablegen. Liegt der Ordner woanders, müssen Sie in der `CMakeLists.txt` des Frameworks den Pfad zu `Free-GLUT` anpassen. `(set(GLUT_ROOT_PATH...)`

d) Kompilieren Sie nun das Framework, um festzustellen, ob alles richig eingerichtet ist. Um Makefiles bzw. Projektdateien plattformunabhängig zu generieren, empfehlen wir `CMake` [8] zu verwenden. Unter Linux wechseln Sie in den `build`-Ordner und führen nacheinander `cmake ..` sowie `make` aus. Unter Windows können Sie Projektdateien mittels der IDE von `CMake` generieren.

   In der CMake-Gui (Windows) müssen erst sowohl der Pfad für die Quelldateien, als auch der Pfad für das Build-Verzeichnis angegeben werden. Anschließend auf *Configure* drücken. Wenn keine Fehlermeldung ausgegeben wird, kann mit *Generate* das Projekt erstellt werden.

   Unser Framework öffnet ein OpenGL-Fenster und zeichnet ein Koordinatensystem.

Erzeugung und Visualisierung einer Halbkanten-Datenstruktur

In dieser Programmieraufgabe soll eine Halbkanten-Datenstruktur implementiert werden. Wir stellen ein Framework bereit, welches die Elemente der Datenstruktur bereits definiert und ein Minimalbeispiel erzeugt. Beachten Sie, dass das Minimalbeispiel derzeit noch **keine** valide Topologie erstellt. Es werden nicht alle Elemente der Datenstruktur erzeugt (Body, Shell, ...) und

viele Verknüpfungen sind nicht gesetzt (oppHe, prevHE, toLoop, toFace, ...). Das Framework bezieht sich auf die Datenstruktur aus den Folien, ist aber vereinfacht:

- Es gibt keine ID-Einträge.
- Es gibt keine unsortierten next und previous pointer. Lediglich HalfEdges und Loops besitzen next und previous pointer. Diese sind allerdings sortiert und zeigen auf den topologischen Nachbarn. Für eine unsortierte Liste können die `std::list<...*>` Objekte aus der `HalfEdgeDS.h` verwendet werden.
- Es gibt keine Geometrieinformationen zu den faces. Nur Vertices haben 3D-Koordinaten.
- Das Element `Shell` fehlt. Die Objekte, die erzeugt werden können haben somit immer nur eine Oberfläche.

Die Einzelaufgaben stellen eher die Minimalanforderung dar. Auch das bereitgestellte Framework ist darauf ausgelegt möglichst einfach zu sein und liefert nur einen sehr beschränkten Funktionsumfang. Es gibt viele Möglichkeiten das Programm besser zu gestalten. Nach Lust und Laune kann der Funktionsumfang, die Visualisierungsqualität oder der Bedienkomfort erheblich verbessert werden. Es steht euch offen die Minimalanforderung zu erarbeiten oder euch im Programm auszutoben. Zu vervollständigende Zeilen sind im Framework mit `TODO:` gekennzeichnet.

Aufgaben

- Visualisieren Sie die Halbkanten-Datenstruktur als Drahtgittermodell. Zeichnen Sie dazu die Linien der Kanten und Punkte der Vertices.
- Erstellen Sie ein valides Testobjekt mit allen topologischen Elementen und Verknüpfungen. Dieses Objekt sollte volumetrisch sein und mindestens ein Loch besitzen.
- Ermöglichen Sie die Navigation durch die Datenstruktur über die Halbkanten (next, previous, opposite). Visualisieren Sie die derzeit aktive Halbkante. Über Tastendruck soll zu einer verknüpften Halbkante gesprungen werden können.
- Erweitern Sie die Navigation, so dass die Halbkante auf innere Loops springen kann, falls welche existieren.
- Erweitern Sie das Interface der Datenstruktur um mindestens drei Euleroperatoren. Vorschlag: `MEVVLS`, `MEV`, `MEL`, `KPMH` genügen, um Objekte mit Löcher und Durchbohrungen zu erstellen. Dies soll die dynamische Modifikation der Datenstruktur erlauben, so dass man nicht auf ein unveränderliches Testobjekt beschränkt ist.
- Demonstrieren Sie die Euleroperatoren anhand von Beispielen, z.B. auf Tastendruck.
- Überprüfen Sie die Datenstruktur mit der Euler-Poincaré-Formel. Die einzige schwer bestimmbare Variable ist die Anzahl der Volumendurchbrüche (Rings). Stellen Sie die Gleichung um und geben Sie die Anzahl der Rings auf der Konsole aus. Dies kann dann visuell überprüft werden.

Literatur

[1] C++ Language Tutorial http://www.cplusplus.com/doc/tutorial/

[2] Thinking in C++ Vol 1 & 2 http://mindview.net/Books/TICPP/ThinkingInCPP2e.html

[3] Eclipse CDT http://www.eclipse.org/cdt/

[4] GNU Make http://www.gnu.org/software/make/

[5] OpenGL Homepage. http://www.opengl.org/

[6] OpenGL Wiki. http://www.opengl.org/wiki/Getting_started#How_to_make_your_first_OpenGL_Program

[7] Free-GLUT OpenGL Utility Toolkit. http://freeglut.sourceforge.net/

[8] CMake http://www.cmake.org