CG2

Lab 3

Aufgabe 1 Quaternion in $\mathbb{R}^{3\times3}$ konvertieren

(a) Mit welchen Vektoren $\vec{x}_i$ müssen Sie die Matrix $\boldsymbol{A} \in \mathbb{R}^{3\times3}$ multiplizieren, um den $i$ten Spaltenvektor von $\boldsymbol{A}$ zu erhalten?

(b) Gegeben ist nun eine Einheitsquaternion $\boldsymbol{q} \in \mathbb{H}$. Wie lautet die zugehörige Rotationsmatrix?
Hinweis: Transformieren Sie $\vec{x}_i$ mit $\boldsymbol{q}$.

Aufgabe 2 Examiner Controller anbinden

In dieser Aufgabe soll eine Kamerasteuerung per Maus hinzugefügt werden. Dazu dient die Klasse `ExaminerController`. Diese bekommt folgende Mauskommandos:

- `mouseDown`, sobald eine Maustaste gedrückt wird
- `mouseMove`, sobald die Maus bewegt wird
- `mouseUp`, sobald eine Maustaste losgelassen wird.

Diese Mauskommandos werden von `ExaminerController` verarbeitet und in eine Matrix mit Rotations- und Translationsanteil konvertiert. Diese Matrix wird mittels der Methode `getMatrix()` zurürckgegeben. Die Matrix wird aus einer Quaternion `this.rotation` und einem Dreiervektor `this.translation` berechnet.

(a) Fügen Sie in Ihrer `Mesh3DApp` eine neue Membervariable `let mExaminerController = new ExaminerController` ein!

(b) Erstellen Sie in `Mesh3DApp` die Methode `normalizeMouseCoordinates(e)`, welche von einem Mausereignis `e` herausfindet, an welcher Stelle es relativ zum betroffenen UI-Element aufgetreten ist. Geben Sie die Koordinaten als `Vec2` zurück! Die Koordinaten sollen normalisierte Werte zwischen $[-1 \ldots 1]$ annehmen, wenn das Ereignis innerhalb des UI-Elements aufgetreten ist.

(c) Erstellen Sie eine Methode, welche den `ExaminerController` und `Mesh3DApp` verbindet:

```
function registerMouseCallbacks()
{
  gl.canvas.addEventListener('mousedown',
    (event) => { mExaminerController.mouseDown(event.button,
                 normalizeMouseCoordinates(event));});

  gl.canvas.addEventListener('mouseup',
    (event) => { mExaminerController.mouseUp(event.button);});

  gl.canvas.addEventListener('mousemove',
    (event) => { mExaminerController.mouseMove(normalizeMouseCoordinates(event));});

  gl.canvas.addEventListener('contextmenu', (event) => { event.preventDefault();});
}
```

(d) Nutzen Sie die Matrix, die Sie von dem `ExaminerController.getMatrix()` bekommen um das 3D Modell zu transformieren! Sie können davon ausgehen, dass die Matrix nur rotiert und transliert!

Aufgabe 3 Shift-Modus

(a) Wenn die rechte Maustaste (`button = 1`) gedrückt wird, soll der `ExaminerController` in den Shift-Modus gehen. Setzen Sie in diesem Fall `shifting` auf `true`! Wird die rechte Maustaste wieder losgelassen, soll der Shift-Modus verlassen werden!

(b) Während der Shift-Modus aktiv ist und die Maus sich bewegt, soll sich das Objekt in x und z Richtung bewegen. Passen Sie dazu `this.translation` an. Merken Sie sich dazu die normalisierte Mausposition beim Eintreten des Shift-Modus und berechnen Sie die Änderung der normalisierten Mausposition jedes Mal, wenn sich die Maus bewegt. Leiten Sie aus der Änderung der Mausposition einen neuen Translationsvektor `this.translation` her.

(c) Passen Sie die Methode `getMatrix()` an, so dass die so berechnete Translation berücksichtigt wird!

Aufgabe 4 Pitch-Modus

Wird die mittlere Maustaste (`button = 2`) gedrückt, soll der `ExaminerController` das Objekt in x und y Richtung verschieben. Passen Sie die Methoden des `ExaminerControllers` entsprechend an!

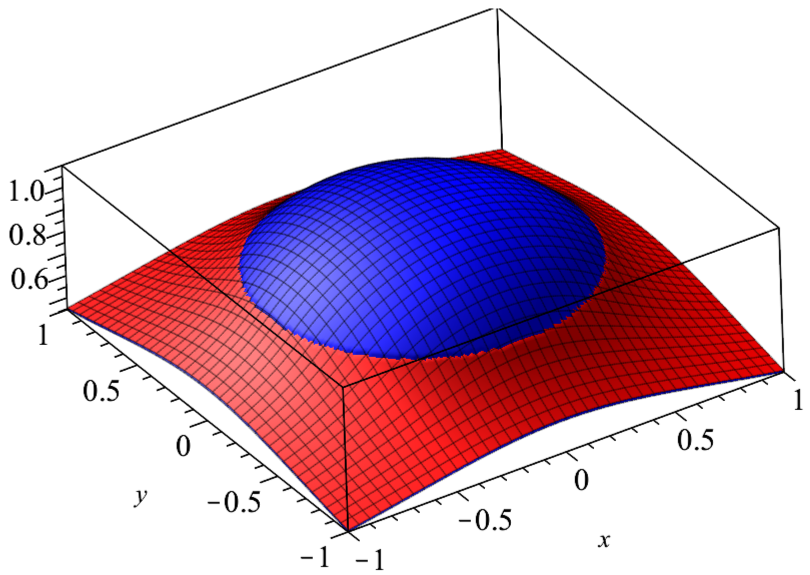
Aufgabe 5 Quaternionen Klasse

Implementieren Sie die Methoden der Klasse `Quaternion` gemäß der Beschreibung im jeweiligen Kommentar!

Aufgabe 6 Rotations-Modus

(a) Der Rotations-Modus wird aktiv, sobald die linke Maustaste (`button = 0`) gedrückt wird und entsprechend inaktiv, falls diese wieder losgelassen wird. Implementieren Sie dieses Verhalten!

(b) Aus der normalisierten 2D Mausposition x, y wird ein Punkt auf eine 3D Fläche projiziert. Für den Menschen am intuitivsten für Rotationen ist es, diese Position auf eine Halbkugel zu projizieren, d.h. $\left[x, y, \sqrt{1 - x^2 - y^2}\right]^\top$. Da allerdings der Ausdruck unter der Wurzel negativ werden kann, wird, falls $\sqrt{x^2 + y^2} \geq \frac{1}{2}$ ein Punkt auf einem Hyperboloid zurückgegeben, $\left[x, y, \frac{1}{2\sqrt{x^2+y^2}}\right]^\top$. Berechnen Sie diesen z Wert in der Methode `projectToSurface(p)`. Die Fläche sieht wie folgt aus (blau ist der Kugelanteil, rot der Hyperboloidanteil).



(c) Aus zwei aufeinanderfolgenden, sich ändernden, Mauspositionen kann nun eine Quaternion berechnet werden. Dazu bestimmt man die 3D Position der beiden Mauskoordinaten $\vec{a}$ und $\vec{b}$, wie in der vorherigen Teilaufgabe beschrieben. Der Vektor $\vec{k}$ wobei $\vec{k} \perp \vec{a}$ und $\vec{k} \perp \vec{b}$, ist die Rotationsachse. Der Rotationswinkel ist

$$\alpha = \angle(\vec{a}, \vec{b}) = 2\,\mathrm{asin}\frac{\|\vec{a} - \vec{b}\|}{2}.$$

Zaubern Sie aus der Achse und dem Winkel eine Rotationsquaternion! Aktualisieren Sie mit dieser dann die Rotationsquaternion `this.rotation`!

(d) Nutzen Sie die Methode `Quaternion.toMatrix` um die Matrix in `ExaminerController.getMatrix()` mit der Rotation zu befüllen.