Navigation API

Einbinden der API:

API-Javascript File in Dateistruktur an die gewünschte Stelle kopieren und im HTML-Dokument nach x3dom.js einbinden:

```
<script type="text/javascript" src="navigationAPI.js"></script>
```

Initialisierung der Kamera:

1. Viewpoint im HTML X3D Container durch Viewfrustum ersetzen:

```
<Viewpoint id="camera" position="-2.06146 3.06806 133.89495" orientation="0.00000 0.00000 0.00000 0.00000" zNear="0.01000" zFar="10000.00000"></Viewpoint>

<Viewfrustum id="camera"></Viewfrustum>
```

2. Erzeugung eines Viewstate Objektes welches ViewMatrix und ProjectionMatrix enthält und damit die Kamera repräsentiert:

```
/**
 * Defines a camera state object that contains the view and projection matrix
 * @param translation Vector to position the camera
 * @returns {Viewstate}
 */
Viewstate(translation: x3dom.fields.SFVec3f)


/**
 * Example for a production of a concrete instance
 */
var viewState = new Viewstate(
     x3dom.fields.SFVec3f(0.0, 1.8, -20.0)
);
```

3. Initialisierung der API durch Setzen der CameraID und Setzen des erzeugten Viewstate

```
NavigationAPI.SetCamera("camera");
NavigationAPI.SetViewState(viewState);
```

Aktualisierung der Kamera:

1. Verschieben der Kamera:

```
/**
 * Adds the given translation to the current ViewMatrix
 * @param {x3dom.fields.SFVec3f} translation
 * @returns {void}
 */
viewState.TranslateView(new x3dom.fields.SFVec3f(x, y, z));
NavigationAPI.SetView(viewState.ViewMatrix);
NavigationAPI.Render();
```

2. Rotieren der Kamera:

```
/**
 * Adds the given angles to the current ViewMatrix
 * @param {number} xAngle
 * @param {number} yAngle
 * @returns {void}
 */
viewState.RotateView(x, y);
NavigationAPI.SetView(viewState.ViewMatrix);
NavigationAPI.Render();
```

Navigation Mathe Funktionen

```
// Creates a Vector2
var vec2 = new x3dom.fields.SFVec2f(x, y);

// Creates a Vector3
var vec3 = new x3dom.fields.SFVec3f(x, y, z);
```