Erklärung zu *Cube.ts*, *Fragment.ts* und *Main.ts*

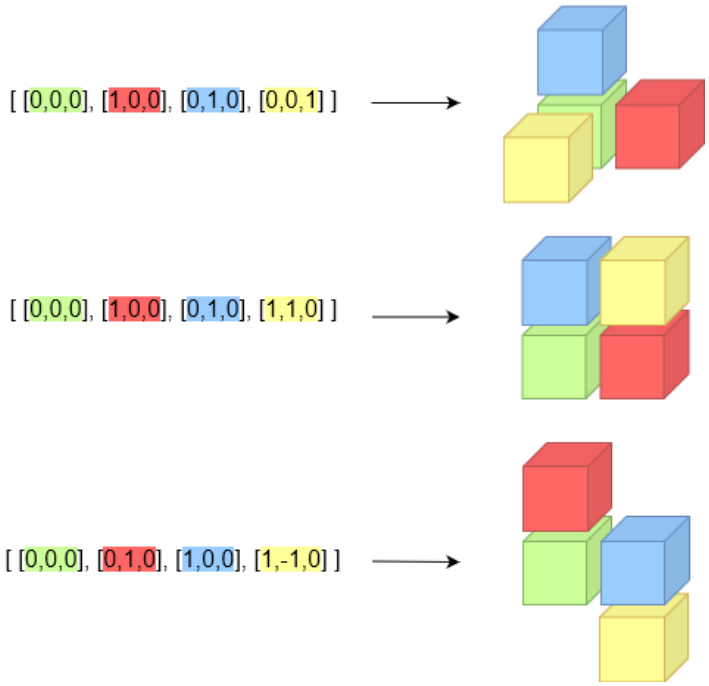
Script Cube.ts:

Das Script *Cube.ts* erzeugt den kleinsten Teil eines Fragments – einen Würfel.
Da jedem Würfel eine Farbe zugewiesen werden soll, werden die Namen der Farben in dem Enum `CUBE_TYPE` gespeichert. Somit werden die Namen auch gegen Schreibfehler gesichert, die bei einem String auftreten können.
Es wird zudem die Variable `Materials` von dem generischen Datentyp *Map* angelegt, um jeder Farbe ein Material zuzuweisen.
Die Klasse `Cube` erbt von `f.Node`, da jeder Würfel auch ein Node sein soll. Mit `super` wird dem Knoten einen Namen zugewiesen.
Die Klasse kennt zwei Variablen – `mesh` und `materials` und die Funktionen `createMaterials` und `constructor`. `createMaterials` füllt die Variable `Materials` mit den passenden Schlüssel-Wert-Paaren.
Die Funktion `constructor` nimmt eine Farbe und eine Position entgegen und generiert daraus einen Würfel. Damit dem Würfel die passende Farbe zugewiesen wird, wird `Matierials` mithilfe der Farbe als Schlüssel nach dem entsprechenden Material durchsucht. Am Ende wird der Würfel noch ein kleinwenig geschrumpft, um dadurch eine Abgrenzung zu einem möglichen Nachbarwürfel zu erzeugen.

Script Fragment.ts:

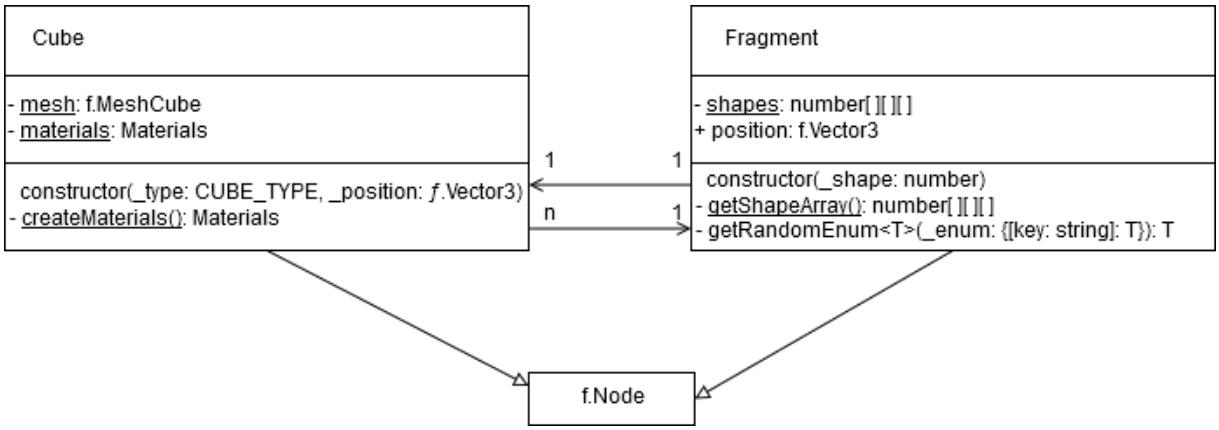
Fragment.ts erzeugt aus den mit *Cube.ts* generierten Würfeln ein Fragment.
Die Klasse `Fragment` erbt ebenfalls von `f.Node` und erhält über den Befehl `super` einen Namen. Die Klasse kennt die Variablen `shapes` und `position`, sowie die Funktionen `constructor`, `getShapeArray` und `getRandomEnum`. `shapes` ist ein dreidimensionales Array, welches durch der Funktion `getShapeArray` gefüllt wird. Danach kennt `shapes` alle möglichen Formen, die ein Fragment annehmen kann. Die Formen liegen als zwei dimensionale Arrays vor, in den die Positionen der einzelnen Würfel gespeichert sind.

Die Grafik zeigt, wie das Fragment zu dem jeweiligen Array aussieht:



Die Funktion `constructor` nimmt eine Zahl entgegen, mit deren Hilfe eine Form für das Fragment ausgewählt wird. Danach wird mit einer For-Of-Schleife das Fragment aus den Würfeln gebaut. Dafür wird mit der Funktion `getRandomEnum` ein zufälliger Wert aus `CUBE_TYPE` ausgewählt, die Position des Würfels als Vektor gespeichert und dann über den Aufruf `let cube: Cube = new Cube(type, vctPosition)` ein Würfel mit der passenden Farbe und Position generiert und dem Fragment-Knoten angehängt.

Das Klassendiagramm veranschaulicht das Verhältnis in dem *Fragment.ts* und *Cube.ts* zueinander stehen:



Script Main.ts:

Wie gewohnt werden in der *Main.ts* ein Viewport und ein Hauptknoten erzeugt, sowie in der Funktion `hndLoad` ein Canvas und eine Kamera.

Indem `f.RenderManager.initialize` der boolsche Wert `true` übergeben wird, wird ein Antialiasing hervorgerufen. Danach werden drei Fragmente erzeugt, an unterschiedliche Positionen verschoben und an den Hauptknoten angehängt. Zusätzlich kennt *Main.ts* noch die Funktion `hndKeyDown`, welche einen Tastendruck registriert und alle drei Fragmente in die entsprechende Richtung dreht. Für die Drehung wird zuerst der Wert der Variablen `rotate` verändert und dann der geänderte Wert der Rotation der Fragmente zugewiesen. Ganz am Ende wird das Bild mit `f.RenderManager.update` und `viewport.draw` erneuert. Somit wird nicht ständig ein neues Bild gezeichnet.