Klassen- und Methodenübersicht

Im Folgenden wird eine vollständige Übersicht alle Klassen und Methoden der GLOOP Bibliothek (Version 3.7) gegeben. Einige der aufgeführten Klassen und Methoden setzen Kenntnisse voraus, die im Unterricht der gymnasialen Einführungsphase in NRW in der Regel nicht vermittelt werden. Alternativ steht eine *Grundlagenübersicht* der GLOOP Bibliothek zur Verfügung.

1 Unterklassen von GLObjekt

1.1 Klasse GLKegel (Oberklasse GLObjekt)

Konstr. **`GLKegel(double pX, double pY, double pZ, double pRadius, double pHoehe)`**
Erstellt einen Kegel mit der Höhe `pHoehe` und dem Grundflächenradius `pRadius`.

`GLKegel(double pX, double pY, double pZ, double pRadius, double pHoehe, GLTextur pT)`
Erstellt einen Kegel mit der Höhe `pHoehe` und dem Grundflächenradius `pRadius`. Das Objekt wird mit der Textur `pT` überzogen.

`GLKegel(double pX, double pY, double pZ, double pRadius, double pHoehe, String pT)`
Erstellt einen Kegel mit der Höhe `pHoehe` und dem Grundflächenradius `pRadius`. Das Objekt wird mit der Textur in `pT` überzogen.

`GLKegel(GLVektor pPosition, double pRadius, double pHoehe)`
Vektorvariante des Konstruktors.

`GLKegel(GLVektor pPosition, double pRadius, double pHoehe, GLTextur pT)`
Vektorvariante des Konstruktors.

`GLKegel(GLVektor pPosition, double pRadius, double pHoehe, String pT)`
Vektorvariante des Konstruktors.

1.2 Klasse GLKegelstumpf (Oberklasse GLObjekt)

Konstr. **`GLKegelstumpf(double pX, double pY, double pZ, double pRadius1, double pRadius2, double pHoehe)`**
Erstellt einen Kegelstumpf mit der Höhe `pHoehe` und den Radien `pRadius1` und `pRadius2`.

`GLKegelstumpf(double pX, double pY, double pZ, double pRadius1, double pRadius2, double pHoehe, GLTextur pT)`
Erstellt einen Kegelstumpf mit der Höhe `pHoehe` und den Radien `pRadius1` und `pRadius2`. Das Objekt wird mit der Textur `pT` überzogen.

`GLKegelstumpf(double pX, double pY, double pZ, double pRadius1, double pRadius2, double pHoehe, String pT)`
Erstellt einen Kegelstumpf mit der Höhe `pHoehe` und den Radien `pRadius1` und `pRadius2`. Das Objekt wird mit der Textur in `pT` überzogen.

`GLKegelstumpf(GLVektor pPosition, double pRadius1, double pRadius2, double pHoehe)`
Vektorvariante des Konstruktors.

`GLKegelstumpf(GLVektor pPosition, double pRadius1, double pRadius2, double pHoehe, GLTextur pT)`
Vektorvariante des Konstruktors.

`GLKegelstumpf(GLVektor pPosition, double pRadius1, double pRadius2, double pHoehe, String pT)`
Vektorvariante des Konstruktors.

1.3 Klasse GLKugel (Oberklasse GLObjekt)

Konstr. **`GLKugel(double pX, double pY, double pZ, double pRadius)`**
Erstellt an der Stelle `(pX,pY,pZ)` eine Kugel mit dem Radius `pRadius`.

`GLKugel(double pX, double pY, double pZ, double pRadius, GLTextur pT)`
Erstellt an der Stelle `(pX,pY,pZ)` eine Kugel mit dem Radius `pRadius`. Das Objekt wird mit der Textur `pT` überzogen.

`GLKugel(double pX, double pY, double pZ, double pRadius, String pT)`

Erstellt an der Stelle (pX,pY,pZ) eine Kugel mit dem Radius pRadius. Das Objekt wird mit der Textur in pT überzogen.

GLKugel(GLVektor pPosition, double pRadius)
Erstellt an der Stelle pPosition eine Kugel mit dem Radius pRadius.

GLKugel(GLVektor pPosition, double pRadius, GLTextur pT)
Erstellt an der Stelle pPosition eine Kugel mit dem Radius pRadius.

GLKugel(GLVektor pPosition, double pRadius, String pT)
Erstellt an der Stelle pPosition eine Kugel mit dem Radius pRadius.

1.4 Klasse GLLicht (Oberklasse GLObjekt)

Konstr. **GLLicht()**
Erstellt eine weiße Lichtquelle an der Stelle (-10000,10000,10000).

GLLicht(double pX, double pY, double pZ)
Erstellt eine weiße Lichtquelle an der Stelle (pX,pY,pZ).

GLLicht(GLVektor pPosition)
Vektorvariante des Konstruktors.

Auftrag **void setzeAbschwaechung(double pA)**
Setzt, wie stark sich das Licht in der Entfernung abschwächt.

Auftrag **void setzeFarbe(double pR, double pG, double pB)**
Setzt die Farbe der Lichtquelle.

Auftrag **void setzeGlanzlicht(double pR, double pG, double pB)**
Setzt Farbe und Intensität des Glanzlichts.

Auftrag **void setzeHintergrundlicht(double pR, double pG, double pB)**
Setzt den Hintergrundlichtanteil der Lichtquelle.

1.5 Klasse GLPrismoid (Oberklasse GLObjekt)

Konstr. **GLPrismoid(double pX, double pY, double pZ, double pRadius1, double pRadius2, int pEckenzahl, double pHoehe)**
Erstellt einen Prismoiden entlang der Z-Achse.

GLPrismoid(double pX, double pY, double pZ, double pRadius1, double pRadius2, int pEckenzahl, double pHoehe, GLTextur pT)
Erstellt einen Prismoiden entlang der Z-Achse. Das Objekt wird mit der Textur pT überzogen.

GLPrismoid(double pX, double pY, double pZ, double pRadius1, double pRadius2, int pEckenzahl, double pHoehe, String pT)
Erstellt einen Prismoiden entlang der Z-Achse. Das Objekt wird mit der Textur in pT überzogen.

GLPrismoid(GLVektor pPosition, double pRadius1, double pRadius2, int pEckenzahl, double pHoehe)
Vektorvariante des Konstruktors.

GLPrismoid(GLVektor pPosition, double pRadius1, double pRadius2, int pEckenzahl, double pHoehe, GLTextur pT)
Vektorvariante des Konstruktors.

GLPrismoid(GLVektor pPosition, double pRadius1, double pRadius2, int pEckenzahl, double pHoehe, String pT)
Vektorvariante des Konstruktors.

Auftrag **void setzeMantelglaettung(boolean pG)**
Schaltet die Glättung der Kanten zwischen den Mantelflächen ein bzw. aus.

1.6 Klasse GLQuader (Oberklasse GLObjekt)

Konstr. **GLQuader(double pX, double pY, double pZ, double pLX, double pLY, double pLZ)**
Erstellt einen Quader mit den Abmessungen pLX, pLY, pLZ bzgl. der drei Raumdimensionen.

GLQuader(double pX, double pY, double pZ, double pLX, double pLY, double pLZ, GLTextur pT)
Erstellt einen Quader mit den Abmessungen pLX, pLY, pLZ bzgl. der drei Raumdimensionen. Das Objekt wird mit der Textur pT überzogen.

GLQuader(double pX, double pY, double pZ, double pLX, double pLY, double pLZ, String pT)
Erstellt einen Quader mit den Abmessungen pLX, pLY, pLZ bzgl. der drei Raumdimensionen. Das Objekt wird mit der Textur in pT überzogen.

GLQuader(GLVektor pPosition, double pLX, double pLY, double pLZ)
Vektorvariante des Konstruktors.
GLQuader(GLVektor pPosition, double pLX, double pLY, double pLZ, GLTextur pT)
Vektorvariante des Konstruktors.
GLQuader(GLVektor pPosition, double pLX, double pLY, double pLZ, String pT)
Vektorvariante des Konstruktors.

1.7 Klasse GLTafel (Oberklasse GLObjekt)

Konstr. **GLTafel(double pX, double pY, double pZ, double pLX, double pLY)**
Erstellt eine zweidimensionale, beschriftbare Tafel in der Szene.
GLTafel(double pX, double pY, double pZ, double pLX, double pLY, GLTextur pT)
Erstellt eine zweidimensionale, beschriftbare Tafel in der Szene. Das Objekt wird mit der Textur `pT` überzogen.
GLTafel(double pX, double pY, double pZ, double pLX, double pLY, String pT)
Erstellt eine zweidimensionale, beschriftbare Tafel in der Szene. Das Objekt wird mit der Textur in `pT` überzogen.
GLTafel(GLVektor pPosition, double pLX, double pLY)
Vektorvariante des Konstruktors.
GLTafel(GLVektor pPosition, double pLX, double pLY, GLTextur pT)
Vektorvariante des Konstruktors.
GLTafel(GLVektor pPosition, double pLX, double pLY, String pT)
Vektorvariante des Konstruktors.

Anfrage **String gibText()**
Liefert den Schriftzug auf der Tafel.

Auftrag **void setzeAutodrehung(boolean pD)**
Schaltet die automatische Ausrichtung der Tafel zur Kamera ein bzw. aus.

Auftrag **void setzeAutodrehung(boolean pD, int pAchsenfixierung)**
Schaltet die automatische Ausrichtung der Tafel zur Kamera bei Fixierung einer Achse ein bzw. aus.

Auftrag **void setzeBeleuchtung(boolean pB)**
Schaltet die Beleuchtung der Tafel ein bzw. aus.

Auftrag **void setzeFaecherung(int pFaecheranzahl)**
Stellt die Tafel in Form von mehreren Fächern dar.

Auftrag **void setzeKamerafixierung(boolean pF)**
Schaltet die Fixierung der Tafel im Kamerabild ein bzw. aus.

Auftrag **void setzeText(String pText, double pGroesse)**
Setzt einen Schriftzug in der angegebenen Größe auf die Tafel.

Auftrag **void setzeTextfarbe(double pR, double pG, double pB)**
Setzt die Farbe des Schriftzuges auf der Tafel.

1.8 Klasse GLTerrain (Oberklasse GLObjekt)

Konstr. **GLTerrain(double pX, double pY, double pZ, String pHightmap)**
Erstellt eine Landschaftsfläche der Größe `512x512` in der Szene.
GLTerrain(GLVektor pPosition, String pHightmap)

Anfrage **double gibHoehe(double pX, double pZ)**
Liefert die Höhe des Terrains an der Stelle `(pX,pY)`.

Auftrag **void setzeAbmessungen(double pBreite, double pHoehe, double pTiefe)**
Setzt die Abmessungen des Terrains neu. `pHoehe` entspricht der maximal möglichen Höhe des Terrains.

Auftrag **void setzeHoehen(float[][] pH)**
Die Höhen des Terrains können mit Hilfe eines Feldes (`float[512][512]`) übergeben werden.

Auftrag **void zeigeUnterseite(boolean pU)**
Schaltet die Darstellung der Unterseite des Terrains (Backfaceculling) ein bzw. aus.

1.9 Klasse GLTorus (Oberklasse GLObjekt)

Konstr. **GLTorus(double pX, double pY, double pZ, double pRadius, double pDicke)**
Erstellt einen Torus mit dem Radius `pRadius` und der Dicke `pDicke`.
GLTorus(double pX, double pY, double pZ, double pRadius, double pDicke, GLTextur pT)

Erstellt einen Torus mit dem Radius `pRadius` und der Dicke `pDicke`. Das Objekt wird mit der Textur `pT` überzogen.

`GLTorus(double pX, double pY, double pZ, double pRadius, double pDicke, String pT)`

Erstellt einen Torus mit dem Radius `pRadius` und der Dicke `pDicke`. Das Objekt wird mit der Textur in `pT` überzogen.

`GLTorus(GLVektor pPosition, double pRadius, double pDicke)`

Vektorvariante des Konstruktors.

`GLTorus(GLVektor pPosition, double pRadius, double pDicke, GLTextur pT)`

Vektorvariante des Konstruktors.

`GLTorus(GLVektor pPosition, double pRadius, double pDicke, String pT)`

Vektorvariante des Konstruktors.

1.10 Klasse GLWuerfel (Oberklasse GLObjekt)

Konstr. **`GLWuerfel(double pX, double pY, double pZ, double pSeitenlaenge)`**

Erstellt einen Würfel mit der Seitenlänge `pSeitenlaenge`.

`GLWuerfel(double pX, double pY, double pZ, double pSeitenlaenge, GLTextur pT)`

Erstellt einen Würfel mit der Seitenlänge `pSeitenlaenge`. Das Objekt wird mit der Textur `pT` überzogen.

`GLWuerfel(double pX, double pY, double pZ, double pSeitenlaenge, String pT)`

Erstellt einen Würfel mit der Seitenlänge `pSeitenlaenge`. Das Objekt wird mit der Textur in `pT` überzogen.

`GLWuerfel(GLVektor pPosition, double pSeitenlaenge)`

Vektorvariante des Konstruktors.

`GLWuerfel(GLVektor pPosition, double pSeitenlaenge, GLTextur pT)`

Vektorvariante des Konstruktors.

`GLWuerfel(GLVektor pPosition, double pSeitenlaenge, String pT)`

Vektorvariante des Konstruktors.

1.11 Klasse GLZylinder (Oberklasse GLObjekt)

Konstr. **`GLZylinder(double pX, double pY, double pZ, double pRadius, double pHoehe)`**

Erstellt einen Zylinder mit der Höhe `pHoehe` und mit dem Radius `pRadius`.

`GLZylinder(double pX, double pY, double pZ, double pRadius, double pHoehe, GLTextur pT)`

Erstellt einen Zylinder mit der Höhe `pHoehe` und mit dem Radius `pRadius`. Das Objekt wird mit der Textur `pT` überzogen.

`GLZylinder(double pX, double pY, double pZ, double pRadius, double pHoehe, String pT)`

Erstellt einen Zylinder mit der Höhe `pHoehe` und mit dem Radius `pRadius`. Das Objekt wird mit der Textur in `pT` überzogen.

`GLZylinder(GLVektor pPosition, double pRadius, double pHoehe)`

Vektorvariante des Konstruktors.

`GLZylinder(GLVektor pPosition, double pRadius, double pHoehe, GLTextur pT)`

Vektorvariante des Konstruktors.

`GLZylinder(GLVektor pPosition, double pRadius, double pHoehe, String pT)`

Vektorvariante des Konstruktors.

2 Methoden der Oberklasse GLObjekt

Auftrag **`void drehe(double pWX, double pWY, double pWZ)`**
Dreht das Objekt um durch den Mittelpunkt des Objektes gehende Parallelen der Koordinatenachsen.

Auftrag **`void drehe(double pWX, double pWY, double pWZ, double pX, double pY, double pZ)`**
Dreht das Objekt um durch den Punkt `(pX,pY,pZ)` gehende Parallelen der Koordinatenachsen.

Auftrag **`void drehe(double pWX, double pWY, double pWZ, GLVektor pPunkt)`**
Dreht das Objekt um durch den Punkt `pPunkt` gehende Parallelen der Koordinatenachsen.

Anfrage **`GLVektor gibPosition()`**
Liefert den Mittelpunkt des Objektes als Ortsvektor.

Anfrage **`GLTextur gibTextur()`**
Liefert das Texturobjekt, an welches das Objekt aktuell gebunden ist.

Anfragen **`double gibX()`**
`double gibY()`
`double gibZ()`
Liefert die entsprechende Koordinate des Mittelpunktes des Objekts.

Auftrag **`void loesche()`**
Löscht das Objekt.

Auftrag **`void rotiere(double pWinkel, double pX, double pY, double pZ, double pRX, double pRY, double pRZ)`**
Rotiert das Objekt um die angegebene Achse im Raum. Der Punkt `(pX,pY,pZ)` ist derjenige Punkt, durch den die Achse läuft (Ortsvektor). Die Richtung der Achse wird mit dem Richtungsvektor `(pRX,pRY,pRZ)` angegeben.

Auftrag **`void rotiere(double pWinkel, GLVektor pOrt , GLVektor pRichtung)`**
Rotiert das Objekt um die angegebene Achse im Raum.

Auftrag **`void setzeDrehung(double pWX, double pWY, double pWZ)`**
Dreht das Objekt um durch den Mittelpunkt des Objektes gehende Parallelen der Koordinatenachsen, unabhängig von der aktuellen Ausrichtung des Objektes, auf die angegebenen Drehwinkel.

Auftrag **`void setzeFarbe(double pR, double pG, double pB)`**
Setzt die Farbe des Objektes. `pR` = Rotanteil, `pG` = Grünanteil, `pB` = Blauanteil.

Auftrag **`void setzeGlanz(double pR, double pG, double pB, int pHaerte)`**
Setzt die Farbe und die Intensität (`pHaerte`) des Glanzes des Objektes.

Auftrag **`void setzeMaterial(float[][] pM)`**
Setzt die Materialeigenschaft des Objektes.

Auftrag **`void setzePosition(double pX, double pY, double pZ)`**
Setzt die Position des Objekts auf die Position `(pX,pY,pZ)`.

Auftrag **`void setzePosition(GLVektor pPosition)`**
Vektorvariante der Methode.

Auftrag **`void setzeQualitaet(int pQ)`**
Setzt die Darstellungsqualität des Objekts auf `pQ`.

Auftrag **`void setzeSelbstleuchten(double pR, double pG, double pB)`**
Setzt das Selbstleuchten eines Objektes auf die angegebene Farbe.

Auftrag **`void setzeSichtbarkeit(boolean pS)`**
Macht das Objekt sichtbar bzw. unsichtbar.

Auftrag **`void setzeSkalierung(double pG)`**
Absolute Variante von `skaliere`.

Auftrag **`void setzeSkalierung(double pX, double pY, double pZ)`**
Absolute Variante von `skaliere`.

Auftrag **`void setzeTextur(GLTextur pTex)`**
Überzieht das Objekt mit der übergebenen Textur.

Auftrag **`void setzeTextur(String pDateiname)`**
Erstellt aus einer Datei ein Texturobjekt und überzieht das Objekt mit dieser Textur.

Auftrag **`void skaliere(double pG)`**
Verändert die Größe des Objektes um den Faktor `pG`.

Auftrag **`void skaliere(double pX, double pY, double pZ)`**
Verändert die Größe des Objektes in Richtung jeder Achse um einen separaten Wert.

Auftrag **`void verschiebe(double pX, double pY, double pZ)`**
Verschiebt das Objekt entlang der drei Koordinatenachsen.

Auftrag **`void verschiebe(GLVektor pVek)`**
Vektorvariante der Methode.

3 Weitere Grafikklassen

3.1 Klasse GLBoden (Oberklasse Object)

Konstr. **`GLBoden(GLTextur pBoden)`**
`GLBoden(String pBoden)`
Erstellen eine endlose Ebene in der Szene, die mit der im Parameter übergebenen Textur bzw. Bilddatei gekachelt ist.

Anfrage **`GLTextur gibTextur()`**
Liefert das Texturobjekt, an welches das Objekt aktuell gebunden ist.

Auftrag **`void loesche()`**
Löscht das Objekt.

Auftrag **`void setzeFarbe(double pR, double pG, double pB)`**
Setzt die Farbe des Objektes. `pR` = Rotanteil, `pG` = Grünanteil, `pB` = Blauanteil.

Auftrag **`void setzeSichtbarkeit(boolean pS)`**
Macht das Objekt sichtbar bzw. unsichtbar.

Auftrag **`void setzeTextur(GLTextur pTex)`**
Überzieht das Objekt mit der übergebenen Textur.

Auftrag **`void setzeTextur(String pDateiname)`**
Erstellt aus einer Datei ein Texturobjekt und überzieht das Objekt mit dieser Textur.

3.2 Klasse GLHimmel (Oberklasse Object)

Konstr. **`GLHimmel(GLTextur pHimmel)`**
`GLHimmel(String pHimmel)`
Erstellt eine Himmelssphäre, die auf der Innenseite die im Parameter übergebene Textur bzw. Bilddatei zeigt.

Anfrage **`GLTextur gibTextur()`**
Liefert das Texturobjekt, an welches das Objekt aktuell gebunden ist.

Auftrag **`void loesche()`**
Löscht das Objekt.

Auftrag **`void setzeFarbe(double pR, double pG, double pB)`**
Setzt die Farbe des Objektes. `pR` = Rotanteil, `pG` = Grünanteil, `pB` = Blauanteil.

Auftrag **`void setzeSichtbarkeit(boolean pS)`**
Macht das Objekt sichtbar bzw. unsichtbar.

Auftrag **`void setzeTextur(GLTextur pTex)`**
Überzieht das Objekt mit der übergebenen Textur.

Auftrag **`void setzeTextur(String pDateiname)`**
Erstellt aus einer Datei ein Texturobjekt und überzieht das Objekt mit dieser Textur.

3.3 Klasse GLNebel (Oberklasse Object)

Konstr. **`GLNebel()`**
Erstellt ein Nebelobjekt, das die Szene mit gleichmäßigem Dunst ausfüllt.

Auftrag **`void loesche()`**
Entfernt den Nebel aus der Szene.

Auftrag **`void setzeFarbe(double pR, double pG, double pB)`**
Setzt die Farbe des Nebels.

Auftrag **`void setzeNebelbereich(double pAnfang, double pEnde)`**
Der Nebelbereich wird gesetzt.

4 Verschiedene Kameraklassen

4.1 Klasse GLKamera (Oberklasse Object)

Konstr. **`GLKamera()`**
Erstellt eine Kamera im Vollbildmodus.
`GLKamera(int pB, int pH)`
Erstellt eine Kamera, deren Fenster die Breite `pB` und die Höhe `pH` hat.

Auftrag **`void an()`**
Die Kamera wird eingeschaltet und erzeugt Bilder der Szene.

Auftrag **`void aus()`**
Die Kamera wird ausgeschaltet.

Auftrag **`void erstelleEinzelbild()`**
Ist die Kamera aus, kann mit diesem Befehl ein einzelnes Bild erstellt werden.

Anfrage **`GLVektor gibBlickpunkt()`**
Liefert den Blickpunkt der Kamera als Vektor.

Anfragen **`double gibBlickpunktX()`**
`double gibBlickpunktY()`
`double gibBlickpunktZ()`
Gibt die entsprechende Komponente des Blickpunktes der Kamera zurück.

Anfrage **`GLVektor gibBlickrichtung()`**
Liefert die Blickrichtung der Kamera als Vektor.

Anfrage **`int gibBreite()`**
Liefert die Breite des Kamerafensters.

Anfrage **`int gibHoehe()`**
Liefert die Höhe des Kamerafensters.

Anfrage **`GLVektor gibPosition()`**
Liefert die Position der Kamera als Vektor.

Anfrage **`GLVektor gibScheitelrichtung()`**
Liefert die Scheitelrichtung der Kamera als normierten Vektor.

Anfragen **`double gibX()`**
`double gibY()`
`double gibZ()`
Gibt die entsprechende Koordinate der Position der Kamera zurück.

Auftrag **`void loesche()`**
Entfernt die Kamera und schließt das entsprechende Fenster.

Auftrag **`void rotiere(double pWinkel, double pX, double pY, double pZ, double pRX, double pRY, double pRZ)`**
Rotiert die Kamera um die angegebene Achse im Raum (vgl. Methode von `GLObjekt`).

Auftrag **`void rotiere(double pWinkel, GLVektor pOrt, GLVektor pRichtung)`**
Vektorvariante der Methode.

Auftrag **`void schwenkeHorizontal(double pWinkel)`**
Dreht die Kamera in der Art eines Horizontalschwenks (links / rechts) um den Winkel `pWinkel`.

Auftrag **`void schwenkeVertikal(double pWinkel)`**
Dreht die Kamera in der Art eines Vertikalschwenks (oben / unten) um den Winkel `pWinkel`.

Auftrag **`void setzeAugendistanz(double pAugendistanz)`**
Setzt im Stereomodus die Distanz zwischen den beiden Augen des Betrachters.

Auftrag **`void setzeBlickpunkt(double pX, double pY, double pZ)`**
Setzt den Blickpunkt der Kamera auf den Punkt `(pX,pY,pZ)`.

Auftrag **`void setzeBlickpunkt(GLVektor pPunkt)`**
Vektorvariante der Methode.

Auftrag **`void setzeFensterposition(int pX, int pY)`**
Setzt die Position des Fensters auf dem Bildschirm.

Auftrag **`void setzePosition(double pX, double pY, double pZ)`**
Setzt die Position der Kamera auf den Punkt `(pX,pY,pZ)`.

Auftrag **`void setzePosition(GLVektor pPos)`**
Vektorvariante der Methode.

Auftrag **`void setzeScheitelrichtung(double pX, double pY, double pZ)`**
Setzt die Scheitelrichtung der Kamera auf `(pX,pY,pZ)`.

Auftrag **`void setzeScheitelrichtung(GLVektor pRich)`**
Vektorvariante der Methode.

Auftrag **`void setzeStereomodus(boolean pM)`**
Schalten den Stereomodus (Rot-Cyan Anaglyphenbilder) der Kamera ein bzw. aus.

Auftrag **`void verschiebe(double pX, double pY, double pZ)`**
Verschiebt die Kamera um den Wert `pX` auf der X-Achse, `pY` auf der Y-Achse und `pZ` auf der Z-Achse.

Auftrag **`void verschiebe(GLVektor pVerschiebung)`**
Vektorvariante der Methode.

Auftrag **`void vor(double pWeite)`**
Lässt die Kamera in Richtung des Blickpunktes um `pWeite` vorfahren.

Auftrag **`void zeigeAchsen(boolean pAn)`**
Blendet die Koordinatenachsen im Kamerabild ein.

Auftrag **`void zeigeFenster(boolean pS)`**
Zeigt oder versteckt das Kamerafenster.

4.2 Klasse GLSchwenkkamera (Oberklasse GLKamera)

Konstr. **`GLSchwenkkamera()`**
Erstellt eine Schwenkkamera im Vollbildmodus.
`GLSchwenkkamera(int pB, int pH)`
Erstellt eine Schwenkkamera, deren Kamerafenster die Breite `pB` und die Höhe `pH` hat.

4.3 Klasse GLEntwicklerkamera (Oberklasse GLSchwenkkamera)

Konstr. **`GLEntwicklerkamera()`**
Erstellt eine Entwicklerkamera im Vollbildmodus.
`GLEntwicklerkamera(int pB, int pH)`
Erstellt eine Entwicklerkamera, deren Kamerafenster die Breite `pB` und die Höhe `pH` hat.

5 Eingabeklassen

5.1 Klasse GLTastatur (Oberklasse Object)

Konstr. **`GLTastatur()`**
Erstellt ein neues Tastaturobjekt.

Anfrage **`boolean alt()`**
Liefert `true`, wenn die angefragte Taste gedrückt ist.

Anfrage **`boolean backspace()`**
Liefert `true`, wenn die angefragte Taste gedrückt ist.

Anfrage **`boolean enter()`**
Liefert `true`, wenn die angefragte Taste gedrückt ist.

Anfrage **`boolean esc()`**
Liefert `true`, wenn die angefragte Taste gedrückt ist.

Anfrage **`char gibZeichen()`**
Liefert das erste Zeichen aus dem Tastaturpuffer und löscht es.

Anfrage **`boolean istGedrueckt()`**
Liefert `true`, wenn irgendeine Taste gedrückt ist.

Anfrage **`boolean istGedrueckt(char pT)`**
Liefert `true`, wenn die dem Zeichen `pT` entsprechende Taste gedrückt ist.

Anfrage **`boolean links()`**
Liefert `true`, wenn die angefragte Taste gedrückt ist.

Auftrag **`void loeschePuffer()`**
Löscht alle Zeichen aus dem Tastaturpuffer.

Anfrage **`boolean oben()`**
Liefert `true`, wenn die angefragte Taste gedrückt ist.

Anfrage **`boolean rechts()`**
Liefert `true`, wenn die angefragte Taste gedrückt ist.

Anfrage **`boolean shift()`**
Liefert `true`, wenn die angefragte Taste gedrückt ist.

Anfrage **`boolean strg()`**
Liefert `true`, wenn die angefragte Taste gedrückt ist.

Anfrage **`boolean tab()`**
Liefert `true`, wenn die angefragte Taste gedrückt ist.

Anfrage **`boolean unten()`**
Liefert `true`, wenn die angefragte Taste gedrückt ist.

Anfrage **`boolean wurdeGedrueckt()`**
Liefert `true`, wenn Zeichen im Tastaturpuffer sind.

5.2 Klasse GLMaus (Oberklasse Object)

Konstr. **`GLMaus()`**
Erstellt ein neues Mausobjekt.

Anfrage **`boolean doppelklick()`**
Liefert `true`, wenn ein Doppelklick durchgeführt wurde.

Anfrage **`boolean gedruecktLinks()`**
Liefert `true`, wenn die linke Maustaste gerade gedrückt ist.

Anfrage **`boolean gedruecktRechts()`**
Liefert `true`, wenn die rechte Maustaste gerade gedrückt ist.

Anfrage **`int gibX()`**
`int gibY()`
Liefert die entsprechende Mauskoordinate im Kamerafenster.

Anfrage **`boolean linksklick()`**
Liefert `true`, wenn ein Linksklick durchgeführt wurde.

Anfrage **`boolean rechtsklick()`**
Liefert `true`, wenn ein Rechtsklick durchgeführt wurde.

Anfrage **`boolean wirdBewegt()`**
Liefert `true`, wenn die Maus gerade bewegt wird.

6 Hilfsklassen

6.1 Klasse GLVektor (Oberklasse Object)

Konstr. **`GLVektor(double pX, double pY, double pZ)`**
Der Vektor `(pX,pY,pZ)` wird erstellt.
`GLVektor(double pX1, double pY1, double pZ1, double pX2, double pY2, double pZ2)`
Der Vektor `(pX2-pX1, pY2-pY1, pZ2-pZ1)` wird erstellt.
`GLVektor(GLVektor pV)`
Erstellt den Vektor als Kopie des Vektors `pV`.
`GLVektor(GLVektor pV1, GLVektor pV2)`
Erstellt den Vektor als Differenzvektor `pV2 - pV2`;

Auftrag **`void addiere(GLVektor pV)`**
Addiert `pV` auf den Vektor auf.

Auftrag **`void drehe(double pWX, double pWY, double pWZ)`**
Die Spitze des Vektors wird gedreht (vgl. Methoden von `GLObjekt`).

Anfrage **`double gibBetrag()`**
Liefert den Betrag des Vektors.

Anfrage **`GLVektor gibKreuzprodukt(GLVektor pV)`**
Errechnet das Kreuzprodukt aus dem Vektor und `pV` und liefert es als neues Objekt vom Typ `GLVektor` zurück.

Anfrage **`double gibSkalarprodukt(GLVektor pV)`**
Liefert das Skalarprodukt des Vektors und `pV`.

Anfrage **`double gibX()`**
`double gibY()`
`double gibZ()`
Liefert die entsprechende Komponente des Vektors.

Auftrag **`void multipliziere(double pS)`**
Multipliziert den Skalar `pS` mit dem Vektor.

Auftrag **`void normiere()`**
Normiert den Vektor.

Auftrag **`void rotiere(double pWinkel, double pNX, double pNY, double pNZ)`**
Rotiert die Spitze des Vektors um die durch `(pRX, pRY, pRZ)` gegebene Achse.

Auftrag **`void rotiere(double pWinkel, GLVektor pVN)`**
Vektorvariante der Methode.

Auftrag **`void skaliereAuf(double pBetrag)`**
Skaliert den Vektor auf die Länge `pBetrag`.

Auftrag **`void subtrahiere(GLVektor pV)`**
Subtrahiert `pV` von dem Vektor.

6.2 Klasse Sys (Oberklasse Object)

Auftrag **`static void beenden()`**
Beendet das aktuelle Programm.

Auftrag **`static void erstelleAusgabe(String pM)`**
Gibt den String `pM` auf einer am unteren Bildrand eingeblendeten Konsole aus.

Auftrag **`static void erstelleAusgabe(String pT, String pM)`**
Gibt den String `pM` auf einer am unteren Bildrand eingeblendeten Konsole aus und übertitelt sie mit `pT`.

Anfrage **`static String erwarteEingabe()`**
Blendet am unteren Bildschirmrand eine Konsole ein und wartet auf die Eingabe eines String.

Anfrage **`static String erwarteEingabe(String pTitel)`**
Blendet am unteren Bildschirmrand eine Konsole ein und wartet auf die Eingabe eines String. Die Konsole wird mit `pTitel` übertitelt.

Anfrage **`static GLObjekt gibObjekt(double pX, double pY)`**
Gibt das Objekt zurück, welches an der Stelle `(pX,pY)` im Kamerafenster zu sehen ist.

Auftrag **`static void warte()`**
Lässt das System eine Millisekunde warten.

Auftrag **`static void warte(int pM)`**
Lässt das System `pM` Millisekunden warten.