

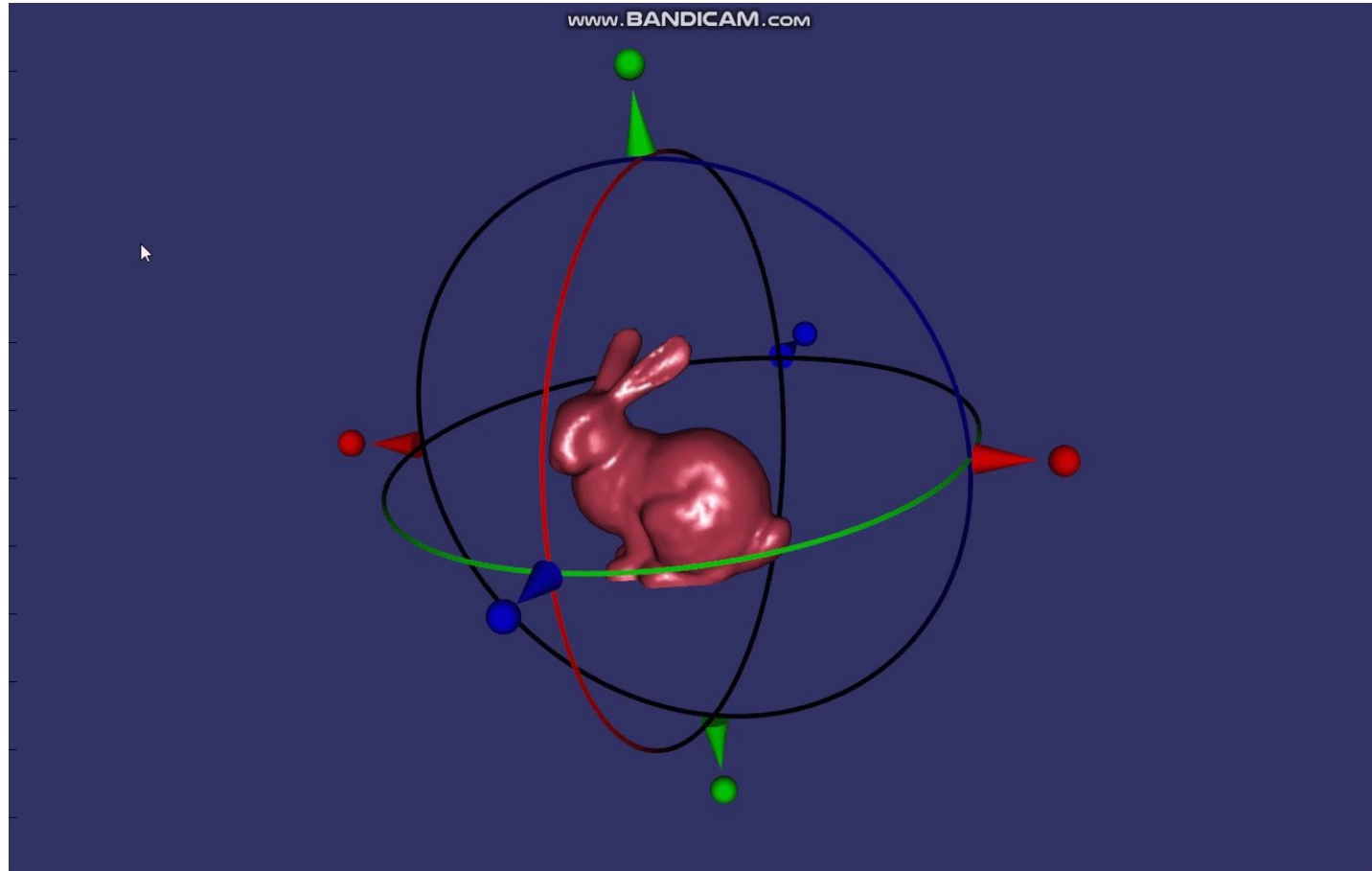
IZG - 3D transformations

Roman Čížmarik

Brno University of Technology, Faculty of information technology
Božetěchova 2, 612 66 Brno
icizmarik@fit.vutbr.cz



- We will implement:
 - Translation
 - Scale
 - Rotation around arbitrary axis



- Quite old, but still maintained
- Implements Scene Graph
- For math consider better options:
 - GLM
 - Eigen



- Row vectors
- Order of multiplication
- Indexing, normalization

```
osg::Matrix model, view, projection;
osg::Matrix MVP = model * view * projection;
```

```
osg::Vec3d vector;
osg::Matrix matrix;
```

```
osg::Vec3d transformedVector = vector * matrix;
```

```
vector.normalize(); //this vector will be normalized
matrix(2,3) = 1.0; //(row, column)
osg::Matrix matrixInv = osg::Matrix::inverse(matrix); //inverse
```

OSG Matrix [documentation](#) and [class reference](#)

$$S(s) = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T(t) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix}$$

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & \sin(\alpha) & 0 \\ 0 & -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_y(\alpha) = \begin{bmatrix} \cos(\alpha) & 0 & -\sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\alpha) & 0 & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

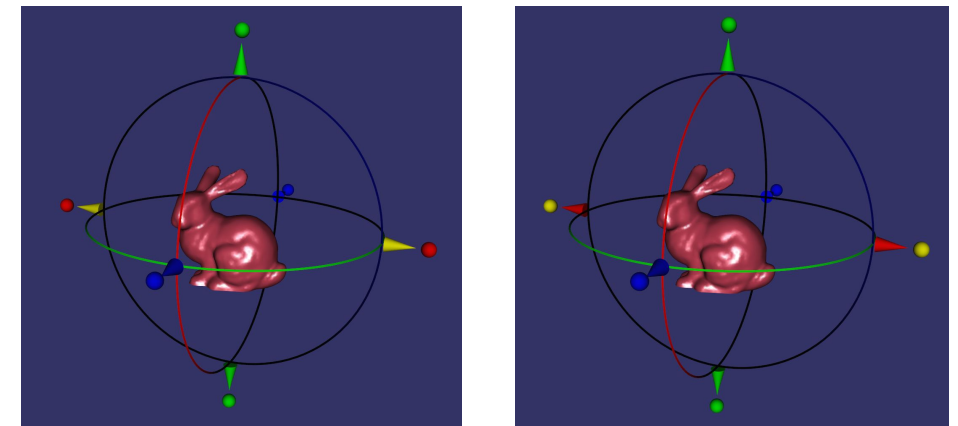
$$R_z(\alpha) = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & 0 & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Modify Student.cpp **ONLY** !!
- Build matrices by hand, do **NOT** use built-in methods of `osg::Matrix` in tasks 1-6.
- Fill in methods:

```
osg::Matrix getScaleMatrix(const osg::Vec3d& scale);           (1)
```

```
osg::Matrix getTranslationMatrix(const osg::Vec3d& translation); (2)
```

- Return correct matrix representing *scale* and *translation*



- Fill in methods

```
osg::Matrix rotateAroundX(double angle); (3)
```

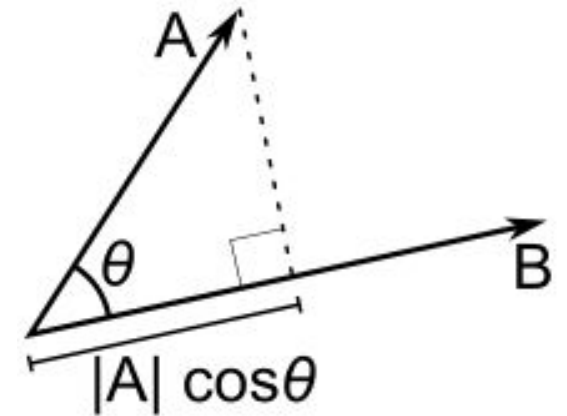
```
osg::Matrix rotateAroundY(double angle); (4)
```

```
osg::Matrix rotateAroundZ(double angle); (5)
```

```
double angleBetweenVectors(osg::Vec3d u, osg::Vec3d v) (6)
```

- Computing angle between two vectors

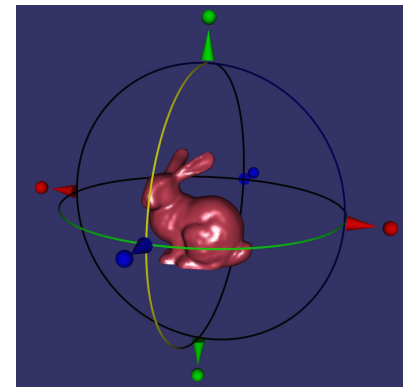
- Normalize vectors! (function `normalize`)
- Dot product (function `dot`)
- Arcus cosine (`std::acos`)

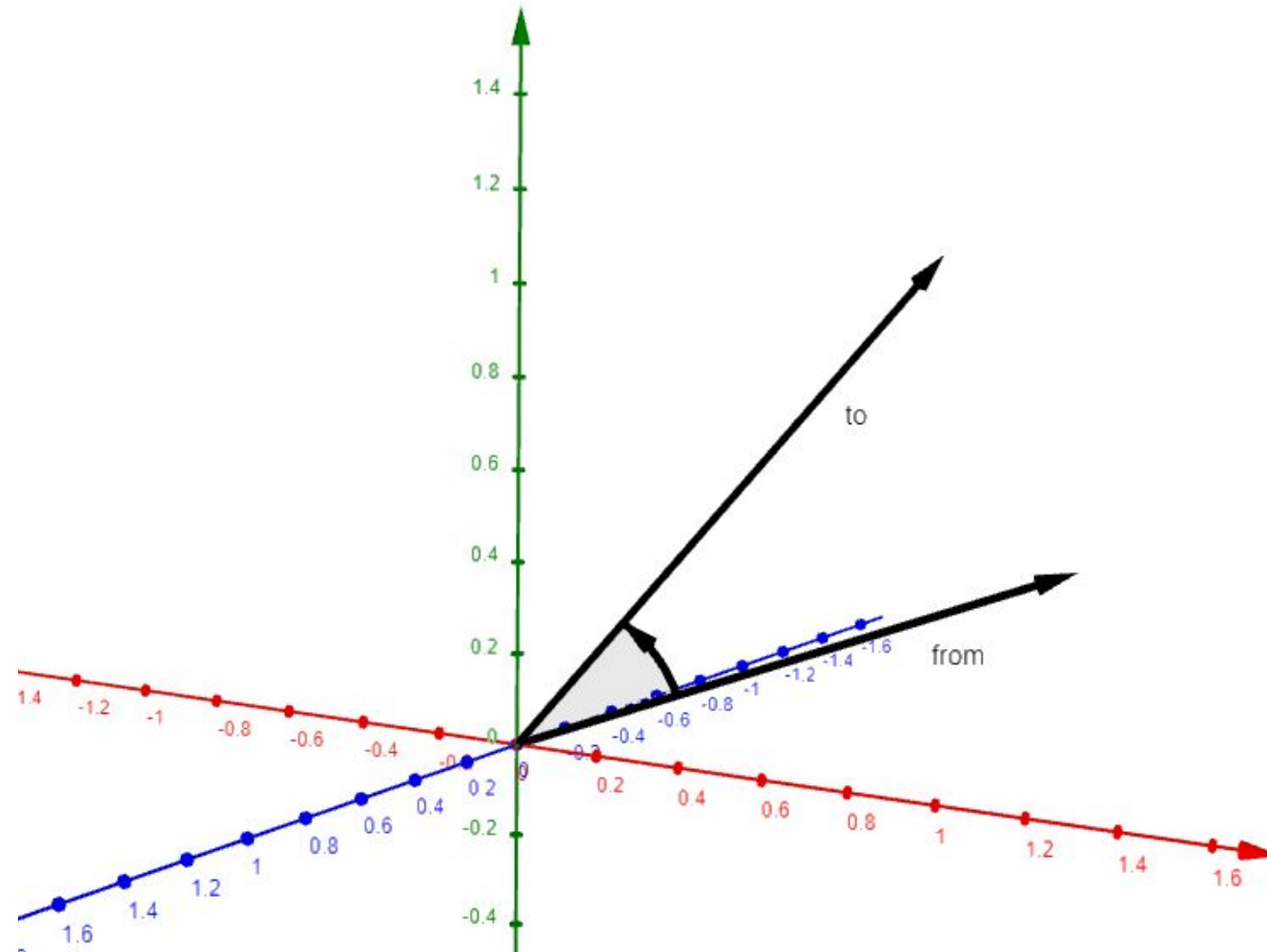


- Rotate around **X axis**
- Fill in method

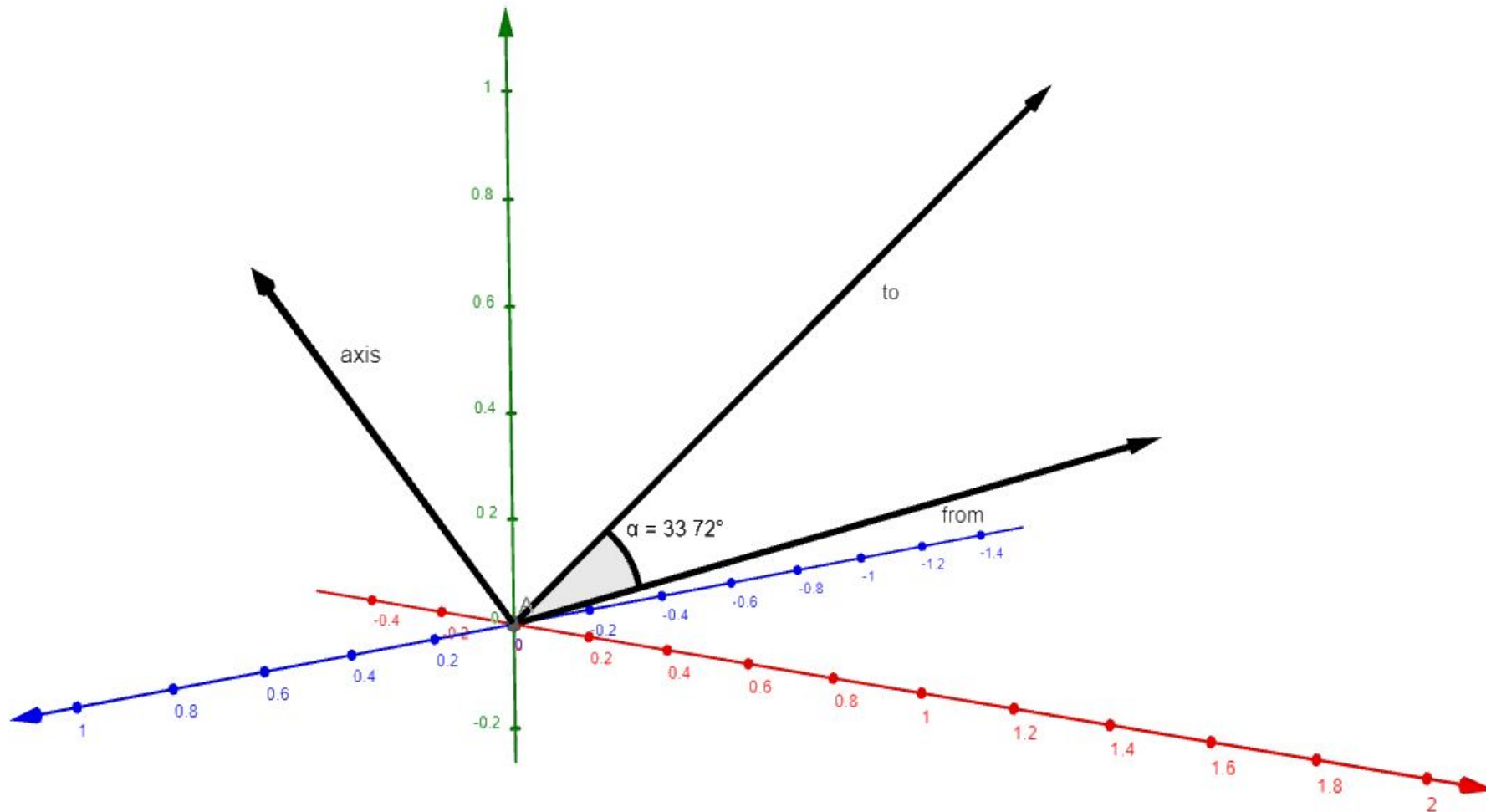
```
osg::Matrix getRotationMatrix(const osg::Vec3d& fromVector, const osg::Vec3d& toVector); (7)
```

- Compute rotation axis using cross product (function cross) (7a)
- Project rotation axis into XY plane (function projectOnPlane) (7b)
- Compute the angle between projection and X axis (7c)
- Rotate around the Z axis (7d)
- Rotate to Y axis - carefully consider(negative) orientation! (7e)
- Rotate around X axis (7f)
- Compose the final rotation (function inverse) (7g)

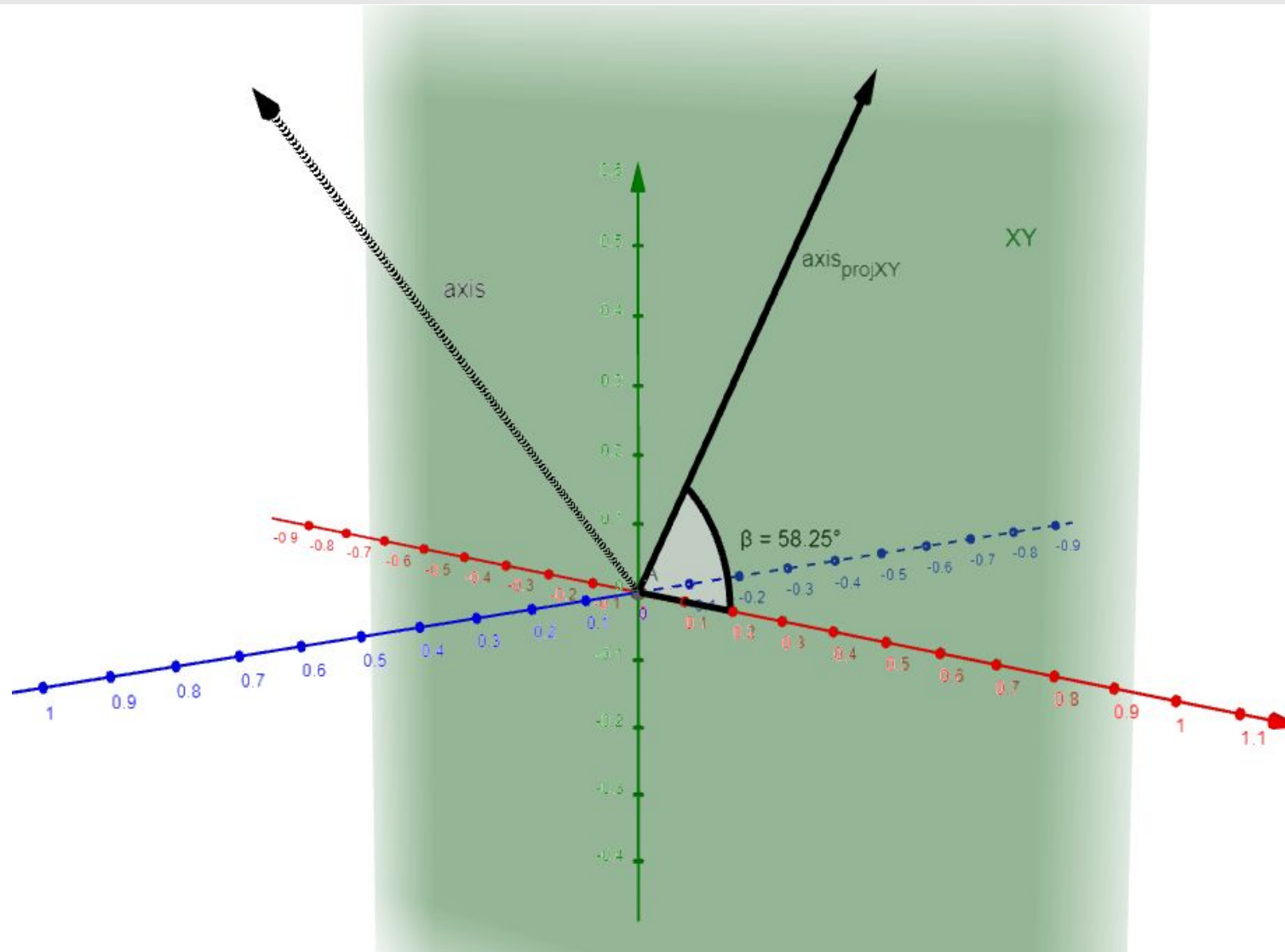




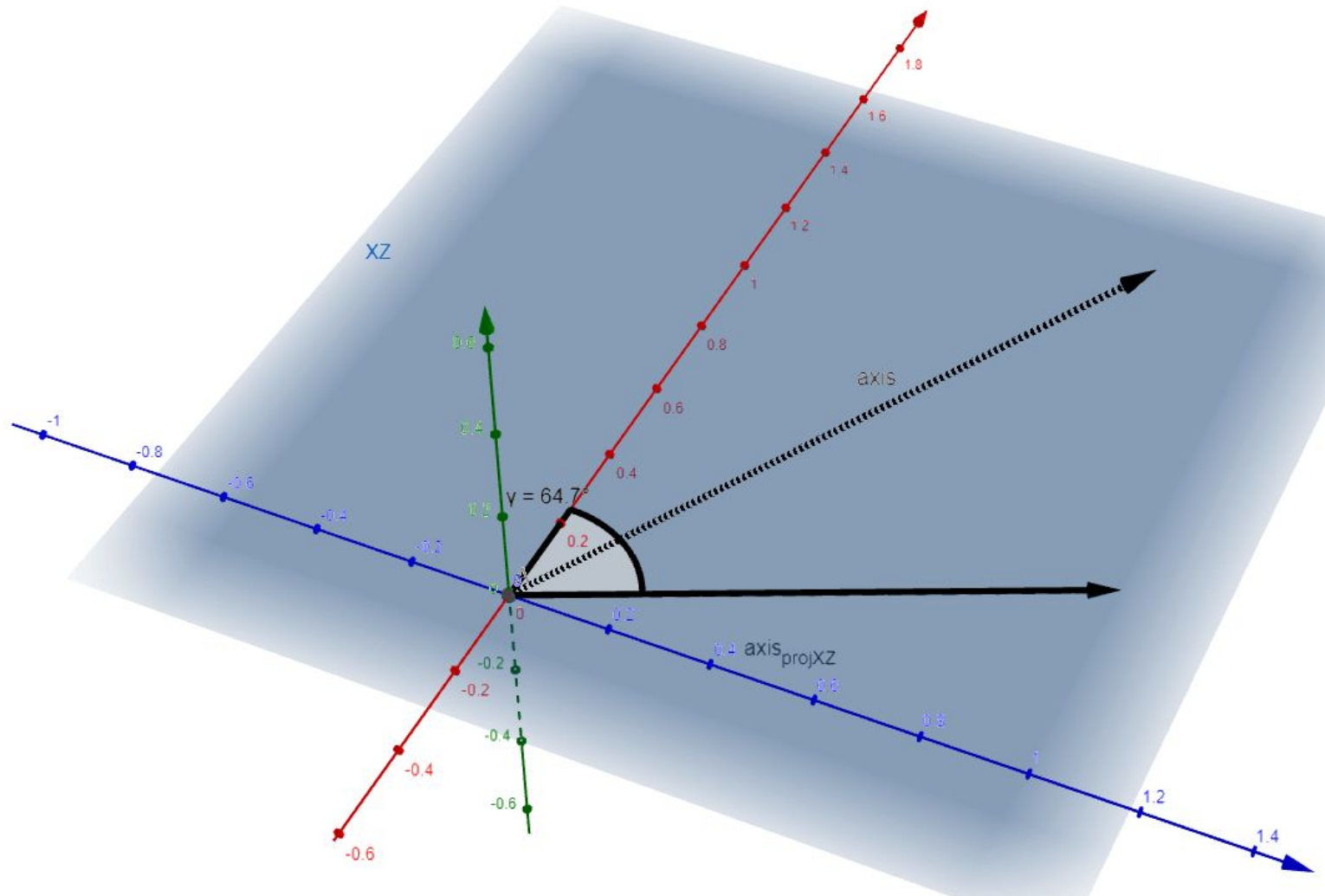
Input



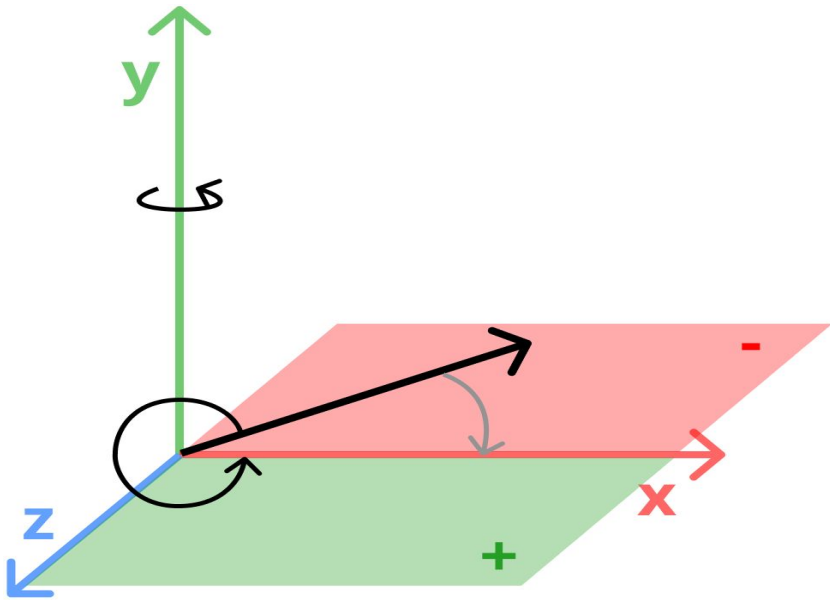
7a - Angle between vectors from and to, axis of rotation (cross product)



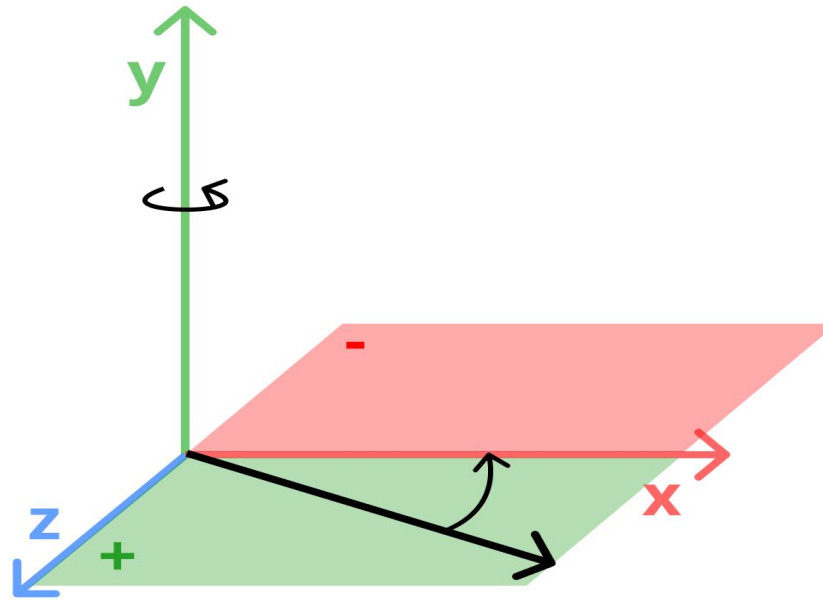
7b,c,d - Projection of the axis of rotation on XY plane, angle to X axis



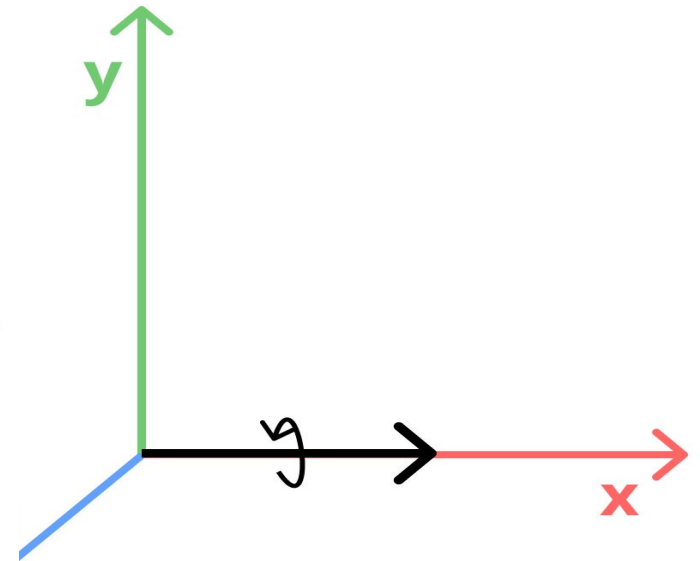
7e - Projection of the axis of rotation on XZ plane, angle to X axis



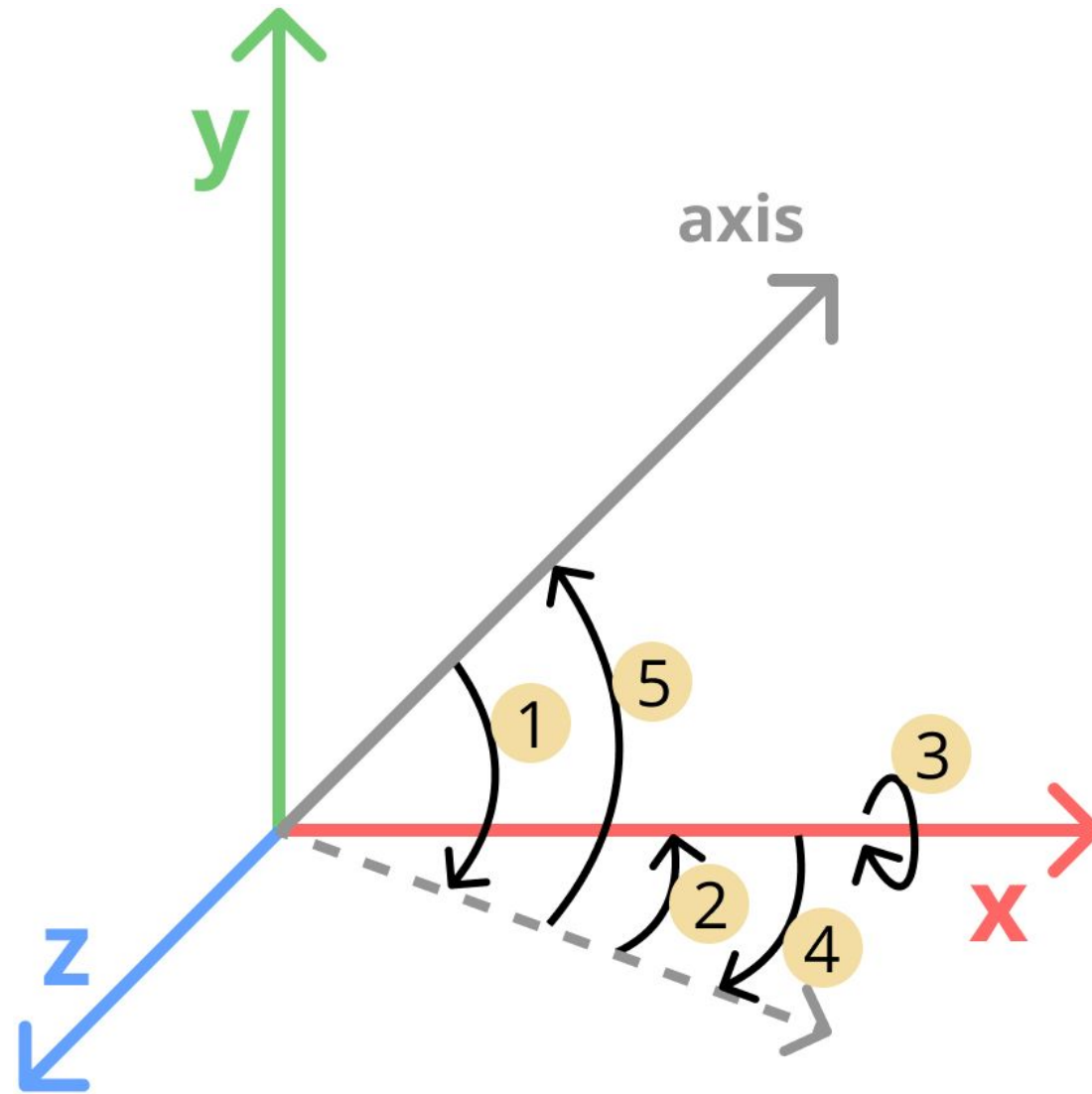
7(e)



(7e)

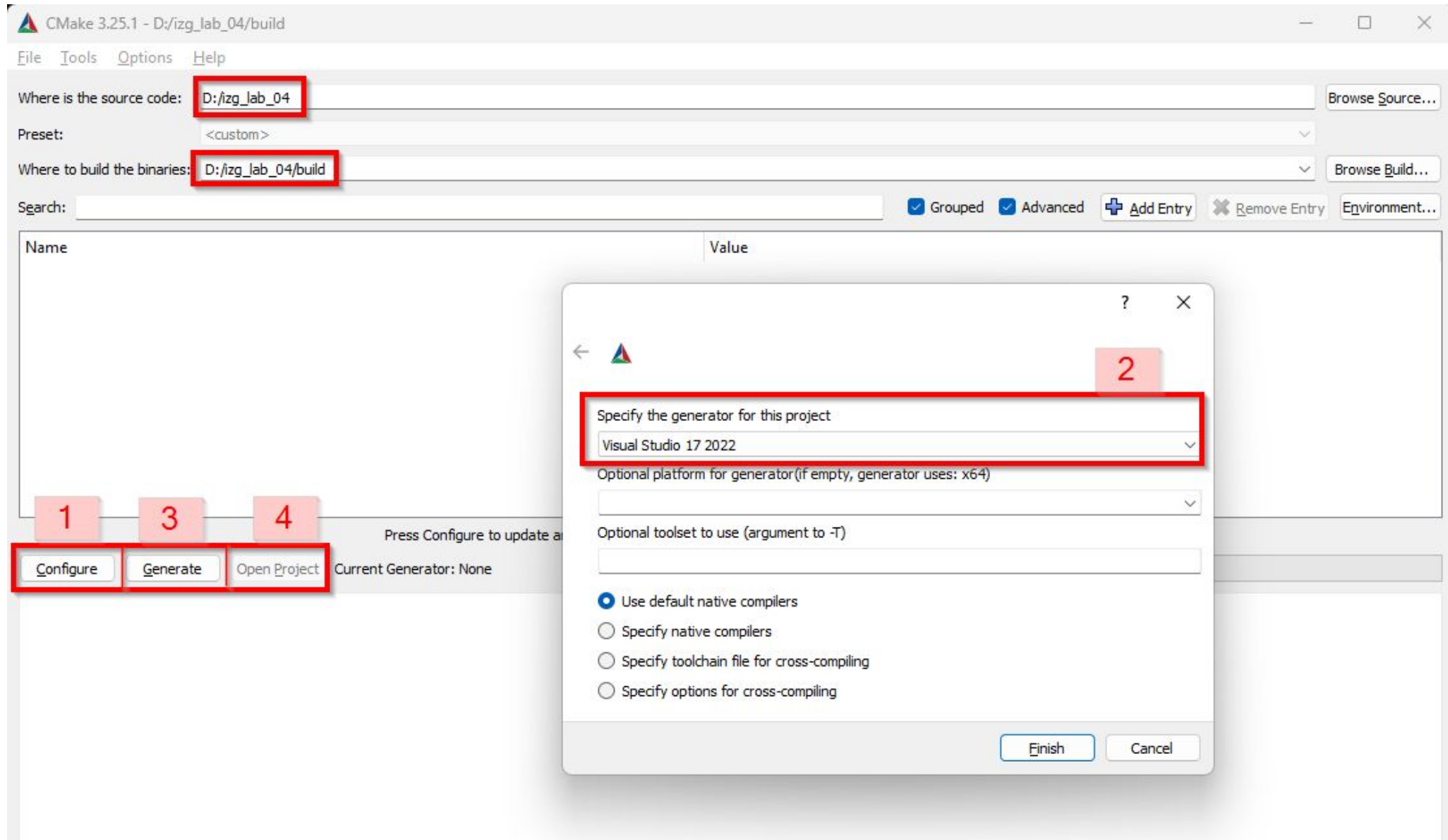


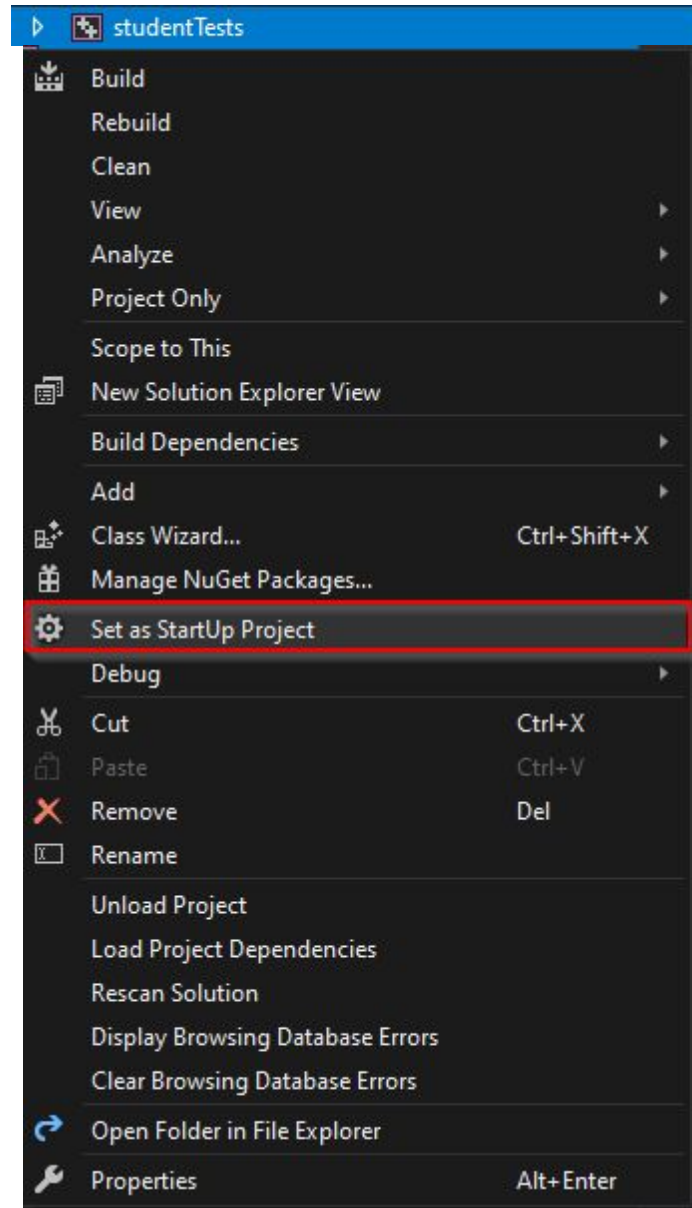
(7f)



(7g)

TIP: Input vectors (in task 7) have the Z-coordinate always 0





Thank you for your attention!