COMP270: 3D Computational Geometry Worksheet 2 – **Answers**

1. A quaternion $\boldsymbol{q}$ to rotate through an angle θ is written as $\boldsymbol{q} = [w \quad \boldsymbol{v}] = \left[\cos\left(\frac{\theta}{2}\right) \quad \sin\left(\frac{\theta}{2}\right)\hat{\boldsymbol{n}}\right]$.
    a. Construct a quaternion to rotate 30° about the x-axis.

    $$\left[\cos\left(\frac{30°}{2}\right) \quad \sin\left(\frac{30°}{2}\right)\begin{pmatrix}1\\0\\0\end{pmatrix}\right] = [0.966 \quad (0.259 \quad 0 \quad 0)]$$

    b. What is the magnitude of this quaternion?

    1 – because all rotation quaternions have a magnitude of 1! (You can do the maths to check if you like, bearing in mind the identity: $\cos^2\theta + \sin^2\theta \equiv 1$).

    c. What is its conjugate, $\boldsymbol{q}$*?

    $$\boldsymbol{q}^* = [w \quad -\boldsymbol{v}] = [0.966 \quad (-0.259 \quad 0 \quad 0)]$$

    d. Assume the quaternion is used to rotate points from object space to world space. What would the position of the point (0.5, -0.7, 2.3) be under this rotation?

    To apply a quaternion rotation to a point, we first express the point in quaternion form, $\boldsymbol{p} = [0 \quad (0.5 \quad -0.7 \quad 2.3)]$. We then multiply $\boldsymbol{p}$ in a sandwich between $\boldsymbol{q}$ and its inverse, noting that:

    i. $\boldsymbol{q}^{-1} = \boldsymbol{q}^*$ for rotation quaternions, and
    ii. $\boldsymbol{q}_1\boldsymbol{q}_2 = [w_1w_2 - \boldsymbol{v}_1 \cdot \boldsymbol{v}_2 \quad w_1\boldsymbol{v}_2 + w_2\boldsymbol{v}_1 + \boldsymbol{v}_1 \times \boldsymbol{v}_2]$

    This gives us: $\boldsymbol{qpq}^{-1} = \boldsymbol{qpq}^* =$
    $[0.966 \quad (0.259 \quad 0 \quad 0)][0 \quad (0.5 \quad -0.7 \quad 2.3)][0.966 \quad (-0.259 \quad 0 \quad 0)]$

    Quaternion multiplication is associative, so it doesn't matter which order we calculate this in. Breaking it down, we get:

    $$w_{\boldsymbol{qp}} = 0.966 \times 0 - (0.259 \times 0.5 + 0 \times -0.7 + 0 \times 2.3) = -0.130$$

    $$\begin{aligned}\boldsymbol{v}_{\boldsymbol{qp}} &= (0.966(0.5 \quad -0.7 \quad 2.3) + 0(0.259 \quad 0 \quad 0) + (0.259 \quad 0 \quad 0) \times (0.5 \quad -0.7 \quad 2.3))\\ &= ((0.483 \quad -0.676 \quad 2.222) + (0 \times 2.3 - 0 \times -0.7 \quad -(0.259 \times 2.3 - 0 \times 0.5) \quad 0.259 \times -0.7 - 0 \times 0.5))\\ &= ((0.483 \quad -0.676 \quad 2.222) + (0 \quad -0.596 \quad -0.181))\\ &= (0.483 \quad -1.272 \quad 2.041)\end{aligned}$$

    Now we perform the second multiplication, with our result above and the inverse of $\boldsymbol{q}$, $[-0.13 \quad (0.483 \quad -1.272 \quad 2.041)][0.966 \quad (-0.259 \quad 0 \quad 0)]$, in the same way:

    $$\begin{aligned}w_{\boldsymbol{qpq}^{-1}} &= -0.13 \times 0.966 - (0.483 \times -0.259 \pm 1.272 \times 0 + 2.401 \times 0)\\ &= -0.126 - (-0.126) = 0\end{aligned}$$

    $$\begin{aligned}\boldsymbol{v}_{\boldsymbol{qpq}^{-1}} &= (-0.13(-0.259 \quad 0 \quad 0) + 0.966(0.483 \quad -1.272 \quad 2.041) + (0.483 \quad -1.272 \quad 2.041) \times (-0.259 \quad 0 \quad 0))\\ &= ((0.034 + 0.467 \quad -1.229 \quad 1.972) + (-1.272 \times 0 - 2.041 \times 0 \quad -(0.483 \times 0 - 2.041 \times -0.259) \quad 0.483 \times 0 - (-1.272) \times -0.259))\\ &= ((0.51 \quad -1.229 \quad 1.972) + (0 \quad -0.529 \quad -0.329))\\ &= (0.51 \quad -1.758 \quad 1.643)\end{aligned}$$

    Giving the transformed point as (0.51, -1.758, 1.643). Notice that the x coordinate ought to be unchanged; due to rounding errors, it has moved a little, but a higher degree of precision should avoid this! Also notice that the w component of the final quaternion has cancelled out to zero, as we would hope for a point.

2. Compute a quaternion that performs twice the rotation of the quaternion $[0.965 \quad (0.149 \quad -0.149 \quad 0.149)]$.

First, we extract the half-angle and axis of rotation:

$$\alpha = \frac{\theta}{2} = \operatorname{acos} w = \operatorname{acos} 0.965 \approx 15°$$

$$\hat{\boldsymbol{n}} = \operatorname{normalise}\begin{pmatrix} 0.149 \\ -0.149 \\ 0.149 \end{pmatrix} \approx \begin{pmatrix} 0.577 \\ -0.577 \\ 0.577 \end{pmatrix}$$

Now we form a new quaternion using the new half-angle, $\alpha' = 2\alpha \approx 30°$:

$$\begin{aligned} &[\cos\alpha' \quad (n_x \sin\alpha' \quad n_y \sin\alpha' \quad n_z \sin\alpha')] \\ &= [0.867 \quad (0.577 \times 0.5 \quad -0.577 \times 0.5 \quad 0.577 \times 0.5)] \\ &= [0.867 \quad (0.288 \quad -0.288 \quad 0.288)] \end{aligned}$$

Note that $\boldsymbol{q}^2$ also represents twice the angular displacement of $\boldsymbol{q}$, so we could use the exponentiation formula and the identities for log and exp to give exactly the same result:

$$\boldsymbol{q}^2 = \exp(2\log\boldsymbol{q}) = \exp(2[0 \quad \alpha\hat{\boldsymbol{n}}]) = \exp([0 \quad 2\alpha\hat{\boldsymbol{n}}]) = [\cos 2\alpha \quad \hat{\boldsymbol{n}}\sin 2\alpha]$$

3. Consider the quaternions:

$$\boldsymbol{a} = [0.233 \quad (0.060 \quad -0.257 \quad -0.935)]$$
$$\boldsymbol{b} = [-0.752 \quad (0.286 \quad 0.374 \quad 0.459)]$$

   a. Compute the dot product ***a*·*b***.

$$\begin{aligned} \boldsymbol{a}\cdot\boldsymbol{b} &= 0.233 \times -0.752 + 0.060 \times 0.268 + -0.257 \times 0.374 + -0.935 \times 0.459 \\ &= -0.175 + 0.017 - 0.096 - 0.429 = -0.683 \end{aligned}$$

   b. Compute the quaternion product ***ab***.

$$w_{\boldsymbol{ab}} =$$
$$0.233 \times -0.752 - (0.060 \times 0.286 + (-0.257) \times 0.374 + (-0.935) \times 0.459)$$
$$= -0.175 - 0.017 + 0.096 + 0.429 = 0.333$$

$$\boldsymbol{v_{ab}} = 0.233(0.286 \quad 0.374 \quad 0.459) + (-0.752)(0.060 \quad -0.257 \quad -0.935) + (0.060 \quad -0.257 \quad -0.935) \times (0.286 \quad 0.374 \quad 0.459)$$
$$= (0.067 - 0.045 \quad 0.087 + 0.193 \quad 0.107 + 0.703) + (-0.257 \times 0.459 - (-0.935) \times 0.374 \quad -(0.060 \times 0.459 - (-0.935) \times 0.286) \quad 0.060 \times 0.374 - (-0.257) \times 0.286)$$
$$= (0.022 \quad 0.28 \quad 0.81) + (-0.118 + 0.35 \quad -0.028 - 0.267 \quad 0.022 + 0.074)$$
$$= (0.022 \quad 0.28 \quad 0.81) + (0.232 \quad -0.295 \quad 0.096) = (0.254 \quad -0.015 \quad 0.906)$$

$$\boldsymbol{ab} = [0.333 \quad (0.254 \quad -0.015 \quad 0.906)]$$

   c. Compute the difference from ***a*** to ***b*** (as a quaternion).

$$\boldsymbol{d} = \boldsymbol{ba}^{-1}$$
$$= [-0.752 \quad (0.286 \quad 0.374 \quad 0.459)][0.233 \quad (-0.060 \quad 0.257 \quad 0.935)]$$

$$\begin{aligned} w_{\boldsymbol{d}} &= -0.752 \times 0.233 - (0.286 \times -0.060 + 0.374 \times 0.257 + 0.459 \times 0.935) \\ &= -0.175 + 0.017 - 0.096 - 0.429 = -0.683 \end{aligned}$$

$$\boldsymbol{v_d} = -0.752(-0.060 \quad 0.257 \quad 0.935) + 0.233(0.286 \quad 0.374 \quad 0.459) + (0.286 \quad 0.374 \quad 0.459) \times (-0.060 \quad 0.257 \quad 0.935)$$
$$= (0.045 + 0.067 \quad -0.193 + 0.087 \quad -0.703 + 0.107) + (0.374 \times 0.935 - 0.459 \times 0.257 \quad -(0.286 \times 0.935 - 0.459 \times -0.060) \quad 0.286 \times 0.257 - 0.374 \times -0.060)$$
$$= (0.112 \quad -0.106 \quad -0.596) + (0.350 - 0.118 \quad -0.267 - 0.028 \quad 0.074 + 0.022)$$
$$= (0.112 \quad -0.106 \quad -0.596) + (0.232 \quad -0.295 \quad 0.096)$$
$$= (0.344 \quad -.401 \quad -0.5)$$

$$\boldsymbol{d} = [-0.683 \quad (0.343 \quad -0.401 \quad -0.5)]$$

4. An object initially had its axes and origin coincident with the world axes and origin. It was first rotated 30° about the y-axis, and then -22° about the world x-axis.

   a. What is the 3x3 matrix that can be used to transform column vectors from object space to world space?

The “transformation from object space to world space” is the same as “the transform of the object in world space” – so, we first need a matrix to represent a rotation of 30° about the y-axis:

$$R_y(30°) = \begin{pmatrix} \cos 30° & 0 & \sin 30° \\ 0 & 1 & 0 \\ -\sin 30° & 0 & \cos 30° \end{pmatrix} = \begin{pmatrix} 0.866 & 0 & 0.5 \\ 0 & 1 & 0 \\ -0.5 & 0 & 0.866 \end{pmatrix}$$

We also need a rotation of -22° about the x-axis:

$$R_x(-22°) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(-22°) & -\sin(-22°) \\ 0 & \sin(-22°) & \cos(-22°) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.927 & 0.375 \\ 0 & -0.375 & 0.927 \end{pmatrix}$$

Next, we need to combine them from right to left in the order they should be applied (since the column vector will be on the far right):

$$R_x(-22°)R_y(30°) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.927 & 0.375 \\ 0 & -0.375 & 0.927 \end{pmatrix}\begin{pmatrix} 0.866 & 0 & 0.5 \\ 0 & 1 & 0 \\ -0.5 & 0 & 0.866 \end{pmatrix}$$

$$= \begin{pmatrix} 1 \times 0.866 + 0 \times 0 + 0 \times -0.5 & 1 \times 0 + 0 \times 1 + 0 \times 0 & 1 \times 0.5 + 0 \times 0 + 0 \times 0.866 \\ 0 \times 0.866 + 0.927 \times 0 + 0.375 \times -0.5 & 0 \times 0 + 0.927 \times 1 + 0.375 \times 0 & 0 \times 0.5 + 0.927 \times 0 + 0.375 \times 0.866 \\ 0 \times 0.866 + (-0.375) \times 0 + 0.927 \times -0.5 & 0 \times 0 + (-0.375) \times 1 + 0.927 \times 0 & 0 \times 0.5 + (-0.375) \times 0 + 0.927 \times 0.866 \end{pmatrix}$$

$$= \begin{pmatrix} 0.866 & 0 & 0.5 \\ -0.188 & 0.927 & 0.325 \\ -0.464 & -0.375 & 0.803 \end{pmatrix}$$

b. What about the matrix to transform vectors from world space to object space?

To go from world space “back” to object space, we simply apply the opposites of each transform that took us from world space to object space, in the opposite order:

$$R_y(30°)R_x(-22°) = \begin{pmatrix} 0.866 & 0 & 0.5 \\ 0 & 1 & 0 \\ -0.5 & 0 & 0.866 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.927 & 0.375 \\ 0 & -0.375 & 0.927 \end{pmatrix}$$

$$= \begin{pmatrix} 0.866 & -0.188 & -0.464 \\ 0 & 0.927 & -0.375 \\ 0.5 & 0.325 & 0.803 \end{pmatrix}$$

This is the inverse of the matrix in part (a), which, since it is a rotation, is equal to the transpose of the original.

c. Express the object’s z-axis using world coordinates.

The object’s z-axis in local space is simply $\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$, so to express this in world coordinates we can apply the object-to-world transform from part (a):

$$\begin{pmatrix} 0.866 & 0 & 0.5 \\ -0.188 & 0.927 & 0.325 \\ -0.464 & -0.375 & 0.803 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0.5 \\ 0.325 \\ 0.803 \end{pmatrix}$$

Notice that this the same as the last column of the matrix – because the matrix represents the basis vectors of the transformed space.

5. Construct a 4x4 matrix to translate by $\begin{pmatrix} 4 \\ 2 \\ 3 \end{pmatrix}$.

$$\begin{pmatrix} 1 & 0 & 0 & 4 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

6. Construct a 4x4 matrix to rotate 20° about the x-axis and then translate by $\begin{pmatrix}4\\2\\3\end{pmatrix}$.

The 4x4 matrix to perform the rotation is:

$$R_x(20°) = \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & \cos(20°) & -\sin(20°) & 0\\ 0 & \sin(20°) & \cos(20°) & 0\\ 0 & 0 & 0 & 1\end{pmatrix} = \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 0.940 & -0.342 & 0\\ 0 & 0.342 & 0.940 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}$$

Applying this after the translation matrix from question (5) gives:

$$\begin{pmatrix}1 & 0 & 0 & 4\\ 0 & 1 & 0 & 2\\ 0 & 0 & 1 & 3\\ 0 & 0 & 0 & 1\end{pmatrix}\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 0.940 & -0.342 & 0\\ 0 & 0.342 & 0.940 & 0\\ 0 & 0 & 0 & 1\end{pmatrix} = \begin{pmatrix}1 & 0 & 0 & 4\\ 0 & 0.940 & -0.342 & 2\\ 0 & 0.342 & 0.940 & 3\\ 0 & 0 & 0 & 1\end{pmatrix}$$

7. Construct a 4x4 matrix to translate by $\begin{pmatrix}4\\2\\3\end{pmatrix}$ and then rotate 20° about the x-axis.

This time we're applying the same transforms in the opposite order:

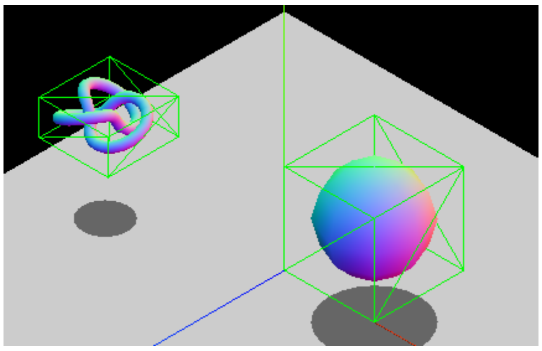
$$\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 0.940 & -0.342 & 0\\ 0 & 0.342 & 0.940 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}\begin{pmatrix}1 & 0 & 0 & 4\\ 0 & 1 & 0 & 2\\ 0 & 0 & 1 & 3\\ 0 & 0 & 0 & 1\end{pmatrix}$$

$$= \begin{pmatrix}1 & 0 & 0 & 4\\ 0 & 0.940 & -0.342 & 2 \times 0.940 + 3 \times -0.342\\ 0 & 0.342 & 0.940 & 2 \times 0.342 + 3 \times 0.940\\ 0 & 0 & 0 & 1\end{pmatrix}$$

$$= \begin{pmatrix}1 & 0 & 0 & 4\\ 0 & 0.940 & -0.342 & 0.854\\ 0 & 0.342 & 0.940 & 3.504\\ 0 & 0 & 0 & 1\end{pmatrix}$$

Notice how applying the translation first has caused it to be rotated, too!

8. An *axis aligned bounding box (AABB)* is the smallest box whose edges are aligned with the coordinate axes that entirely contains a geometric object, defined by its minimum and maximum vertices $\boldsymbol{p}_{min}$ and $\boldsymbol{p}_{max}$. AABBs are commonly used to accelerate the collision testing process.
Describe (in English and/or pseudocode) how one might test for the following intersections, giving an example of when each might be used:
   a. Two AABBs.
   b. A plane and an AABB.
   c. A ray (line) and an AABB.



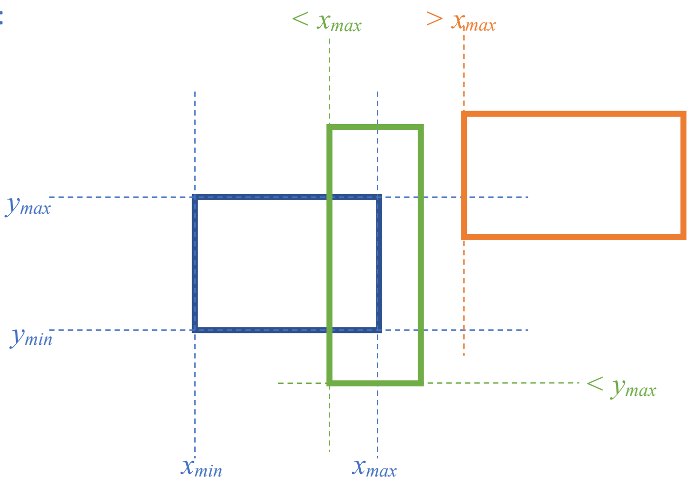
Some bounding boxes, from https://developer.mozilla.org/en-US/docs/Games/Techniques/3D_collision_detection

AABB-AABB intersection test

This test is likely to be used as an initial pass to detect collisions between two more complex geometries: if the bounding boxes do not intersect, there is no need to do any further tests.

In busy scenes with more than a few objects, this test will potentially be run many times; fortunately, it's very simple! The important point to bear in mind is that, in order for the boxes to intersect, the coordinates of their min/max points must overlap in every dimension: that is to say, the min value of one box cannot be greater than the max of the other.

Visualised in 2D:



The code for this is very straightforward, as we just need to compare the max and min points against each other:

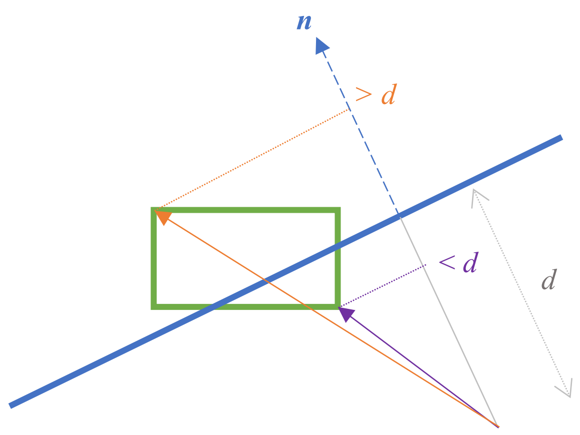
```
bool aabbsIntersect(AABB a, AABB b)
      if (a.min.x >= b.max.x) return false
      if (a.max.x <= b.min.x) return false
      if (a.min.y <= b.min.y) return false
      // … etc. …
      return true
```

Plane-AABB intersection test

This test somewhat less common, but the technique could have applications in slicing operations. For the plane to pass through the AABB, at least one of the vertices must be on the opposite side of the plane from the others, which we can test by comparing the dot products of the vertices with the plane normal.

If the plane is defined in the implicit manner as $\boldsymbol{x}\cdot\boldsymbol{n} = d$, where $d = \boldsymbol{a}\cdot\boldsymbol{n}$ with $\boldsymbol{a}$ being a known point on the plane, then points on one side of the plane will have a dot product greater than d, while for those on the other side it is less than d (this is equivalent to taking the dot product with a vector from $\boldsymbol{a}$ to the AABB vertex and checking the sign of its dot product, with fewer vector differences to compute).

The calculation can be simplified by finding only the minimum and maximum values of the dot product, instead of computing it for all eight vertices. Since all of the AABB vertices are just different combinations of the min and max x, y and z values (and every combination of these values is one of the vertices), the smallest dot product would come from the smallest x, y and z values combined with a positive corresponding coefficient of n – or, if the coefficient of n is negative, the largest x, y and z.



```
bool planeIntersectsAABB(AABB a, Vector3 n, float d)
      float minD = 0, maxD = 0;
      for each component c in x, y, z
            if (nc > 0)
                  minD += nc * minc;
                  maxD += nc * maxc;
            else
                  minD += nc * maxc;
                  maxD += nc * minc;

      return (minD < d and maxD > d)
```

Ray-AABB intersection test

This test is commonly used to speed up ray-casting or ray-tracing of complex objects, by first performing a rejection test on the objects' AABBs.

One way to perform this test is to note that a cuboid is formed of six intersecting planes, each of which is aligned with two of the axes – i.e. there are two planes parallel to each of the xy, yz and zx-planes. We can define these planes using the values of the AABB's min/max points – for example, $y = y_{min}$, $z = z_{max}$ – which constrains the values along one of the axes while allowing the plane to extend infinitely along the other dimensions.

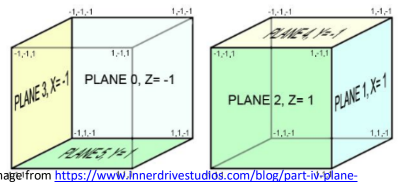


Image from https://www.innerdrivestudios.com/blog/part-iv-plane-detection-a-first-step-to-implementing-panorama-hotspots/

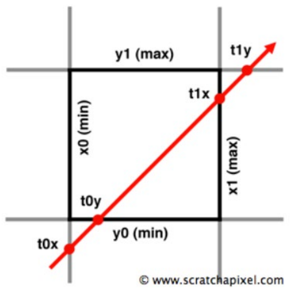
For the AABB to be intersected means that the ray must pass through at least one of these planes, within the bounds of the vertices. With the line expressed in the usual manner, as $\boldsymbol{x} = \boldsymbol{p} + t\boldsymbol{v}$ (with $\boldsymbol{x}$ being any point (x, y, z) on the line, $\boldsymbol{v}$ the direction vector and t the scalar parameter), we can find the point of intersection of the ray with each of the AABB planes via the standard method of substituting the plane "equation" for the relevant component, in this case a single scalar, into the ray equation.

For example, to find the intersection with the plane defined by the minimum x value of the AABB:

$$x_{min} = p_x + tv_x \Longrightarrow t = \frac{x_{min} - p_x}{v_x}$$

(In other words, we already know the value of the x coordinate at the point of intersection, since it's the same everywhere on the plane, so we can substitute this directly to find t using only the x components).

Once we've found the intersection points for all the planes (testing first to make sure that the ray is not parallel to any of them – note the division in the equation above!), we need to check that they are actually within the box. We could compare all the points against all the AABB vertices, however a faster way to check is to notice that if an intersection occurs within the box boundaries, it must include the one that is the furthest from the ray origin, i.e. with the greater value of t, for the planes defined by the box's minimum boundaries (assuming the ray source is on that side of the box).



Comparisons of the t values can also be used to discover which rays miss the box; further details on this computation (along with code examples) can be found here: https://www.scratchapixel.com/lessons/3d-basic-rendering/minimal-ray-tracer-rendering-simple-shapes/ray-box-intersection
and in Appendix A.18 of "3D Math Primer for Graphics and Game Development" by Fletcher Dunn and Ian Parberry.