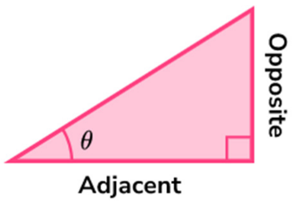
Addendum: use of the tangent function in the initialization of the DDA algorithm

<u>Introduction:</u>

- In the DDA algorithm you check on your ray to collide with horizontal grid lines and with vertical grid lines;
- To do this efficiently, you want to calculate:
  - What is the horizontal displacement of the ray assuming a unit vertical displacement;
  - What is the vertical displacement of the ray assuming a unit horizontal displacement;

This is done in the initialization steps of the DDA algorithm. The tangent function is very helpful here. Check out the definition[6] of tan():

$$\tan(\theta) = \frac{Opposite}{Adjacent}$$



It tells us how many times bigger the opposite is than the adjacent.

If all you have is the angle[7] (called theta in this example), tan() can give you either the opposite length or the adjacent length, but you have to assume the other one being 1.

<u>Horizontal grid line checking:</u>

1. You're extending the ray until it collides to the next **horizontal** grid line;
2. This means the vertical displacement is kept constant at 1 cell height (either up or down);
3. And you're looking for the corresponding horizontal displacement to find the endpoint of the ray (i.e. the collision point with the next horizontal gridline);

Express this using a tangent function:

- You keep the opposite side equal to 1 cell height [see point 2. above]
- You want to calculate the length of the adjacent side [see point 3. above]

Therefore it makes sense to use 1 / tan(), and not tan(). If you keep the opposite length to 1, then 1/tan(theta) = Adjacent / Opposite = Adjacent / 1 = Adjacent. So 1/tan() gives you the adjacent length, assuming the opposite length is 1.

[6] Taken from: https://thirdspacelearning.com/gcse-maths/geometry-and-measure/trigonometric-functions/

[7] This is the ray angle, derived from the player angle and the field of view

Vertical grid line checking:

4. You're extending the ray until it collides to the next **vertical** grid line;
5. This means the horizontal displacement is kept constant at 1 cell width (either left or right);
6. And you're looking for the corresponding vertical displacement to find the endpoint of the ray (i.e. the collision point with the next vertical gridline);

Express this using a tangent function:

- You keep the adjacent side equal to 1 cell width [see point 5. above]
- You want to calculate the opposite side [see point 6. above]

Therefore it makes sense to use tan(), and not 1/tan(). If you keep the adjacent length to 1, then tan(theta) = Opposite / Adjacent = Opposite / 1 = Opposite. So tan() gives you the opposite length, assuming the adjacent length is 1.