Mathematics in game development: Pythagorean theorem

For game development, in the sense of programming, you don’t need to be a mathematician. For the development of real simple games you only need the understanding of summations, subtractions, multiplications and divisions. For the development of more complicated games though, the understanding of some mathematical theorems is a must. For example the Pythagorean theorem is very often used in the programming of games. You probably heard about the Pythagorean theorem before, but perhaps you can’t quite remember what it was all about, or perhaps you don’t see what it could be used for. Because mathematics is a real useful tool for programming we’ll discuss what the Pythagorean theorem is, and what some of its applications are in game development. If you already know what the Pythagorean theorem is, you could skip the next paragraph.

The Pythagorean theorem

Pythagoras once stated and proved that the squared length of \(a\) plus the squared length of \(b\) equals the squared length of \(c\) if \(a\), \(b\) and \(c\) form a triangle where angle \(ab\) is 90° or a 1/2 \(\pi\) radian.

This results in the equation:

\[
a^2 + b^2 = c^2
\]

Solved for \(c\) it gives:

\[
c = \sqrt{a^2 + b^2}
\]

The last equation derived directly from the first equation and gives the length of \(c\), with the lengths of \(a\) and \(b\) as input. So now we can directly calculate the length of the longest side of the triangle, if the other sides are given.

For example, if \(a = 3\) and \(b = 4\), then what is the length of \(c\)? First write down the equation. Fill in the known values and start solving.

\[
c = \sqrt{a^2 + b^2} = \sqrt{3^2 + 4^2} = \sqrt{9 + 16} = \sqrt{25} = 5
\]

So the length of \(c\) is five in this case.

There are other equations that look very similar, but give the lengths of \(a\) or \(b\) instead of the length of \(c\). These equations are not so useful for programming, so we will not pay attention to them here.

Now you’ve learned or remembered how the Pythagorean theorem works, it is time to apply it in game development.
In practice, calculating distance

In the illusive world of games, every object in a game is located at a certain coordinate in the world. The game always knows at which x- and y-coordinate a certain object in the game is. For example point $P$, the player, and point $E$, the enemy. By subtraction it is fairly easy to calculate the horizontal distance, the difference in x-coordinates ($\delta x$), and the vertical distance, the difference in y-coordinates ($\delta y$), between the two objects. However neither one of those distances represent the absolute distance between the objects.

The absolute distance is represented by the dashed distance $D$. Because the x-axis and the y-axis make an angle of 90°, the Pythagorean theorem can be applied to calculate the distance of $D$, with the use of $\delta x$ and $\delta y$ as input. This gives the following equation:

$$D = \sqrt{\delta x^2 + \delta y^2}$$

Once the absolute distance is acquired, it can be used for collision detection, or to check if the player is close enough so that it can hit the enemy. You could also trigger a event when the player comes within a specified range of a certain point in the world.

For example: you are programming a fighting game and you want to check if the player is close enough to hit the enemy with his sword. The player $P$ is currently at point (2; 4) while the enemy $E$ is at point (14; 9). What is the absolute distance between the two?

First we need the know what the horizontal and vertical distance is between the player and the enemy. After we know that, we can use the equation to calculate the absolute distance between the two. Now the horizontal distance is the difference in x-values, while the vertical distance is the difference in y-values:

$$\delta x = E_x - P_x = 14 - 2 = 12$$
$$\delta y = E_y - P_y = 9 - 4 = 5$$

Now solve the equation for that input:

$$D = \sqrt{\delta x^2 + \delta y^2} = \sqrt{12^2 + 5^2}$$
$$= \sqrt{144 + 25} = \sqrt{169} = 13$$

So we found that the absolute distance between the player and the enemy is 13. Now that you know the absolute distance you could decide whether your player can hit the enemy with his sword or not.
In practice, calculating speed

Moving objects have certain speed vectors. This means that the object has an horizontal speed (x-speed) and a vertical speed (y-speed). These two speeds are components of the absolute speed. This means that combining the x-speed and y-speed gets you the real speed the object has on the monitor. This absolute speed can be in any direction.

Perhaps you won’t see it right away, but this figure actually persists from two equally shaped triangles. Both with one corner of 90°, which means the Pythagorean theorem can be applied.

To see this more clearly, it can be useful to place the y-speed at the other side of the rectangle, so it forms a triangle with the x-speed and the absolute speed. Now we can clearly see that the Pythagorean theorem can be used for calculating the absolute speed with the x- and y-speed as inputs. This gives the following equation:

\[ a = \sqrt{x^2 + y^2} \]

For example: you are developing a racing game. Your car is traveling with an horizontal speed of 8, and a vertical speed of 6. What is the absolute speed of the car? Therefore we write down the equation, fill in the known values and solve it:

\[ a = \sqrt{8^2 + 6^2} = \sqrt{64 + 36} = \sqrt{100} = 10 \]

So now we know the absolute speed of the car is 10. Now you can for example show the speed to the player on a speedometer, or trigger an event that the cops come chasing you for speeding.
Programming the equation

Fine all those theories about how things could be calculated, but now you'll probably wonder how to actually program the computer to calculate it for you. Therefore I will now show you two small codes that calculate the two examples that we discussed earlier. These codes will be in C#, a programming language very often used for games and applications. If you are not programming in C#, it will still be useful to look at these codes because most programming languages do have a lot in common, and all principles are the same.

```csharp
// Calculating horizontal and vertical distances
double xDistance = enemy.position.X - player.position.X;
double yDistance = enemy.position.Y - player.position.Y;

// Calculating absolute distance
double absoluteDistance = Math.Sqrt(Math.Pow(xDistance, 2) + Math.Pow(yDistance, 2));
```

This is a code that calculates the distance between the player and the enemy. First it calculates the horizontal distance (\(xDistance\)), and the vertical distance (\(yDistance\)), between the objects. After that, it calculates the absolute distance using the Pythagorean theorem. I used build-in mathematical functions for this, since there is no other way. \(\text{Math.Sqrt()}\) gives back the square root of the given value. \(\text{Math.Pow()}\) gives back the given value to the given power. So \(xDistance\) and \(yDistance\) both to the power of two. So this code calculates the square root of the squared horizontal distance plus the squared vertical distance, which is the Pythagorean theorem. The calculated values are in this case set into \textit{double} variables, which are big decimal numbers. This is done because answers of this equation are often numbers with a lot of decimals.

The following code calculates the speed of a car in a racing game, like the example in the previous paragraph.

```csharp
// Calculating absolute speed
```

This code calculates the square root of the squared horizontal speed plus the squared vertical speed. The acquired value is the absolute speed of the car. Yet again the build-in mathematical functions are used for the calculation, and the final value is set into a \textit{double} variable because of the decimal size of the answer.
The third dimension

This entire article covered the Pythagorean theorem and some applications in game development. Perhaps you noticed we only discussed horizontal and vertical distances and speeds. We did not take the third dimension into account. This does not mean that the theorem is only useful for two dimensional games, it also applies to three dimensional games. For 3D games however, you have one more value, the distance or speed in depth. Instead of only taking the x-axis and the y-axis into account, you now also have to think about the z-axis. This means that the equation has to be altered a little. The new equation is:

\[ x^2 + y^2 + z^2 = d^2 \]

Basically it is the same equation, only one input value has been added, which represents the depth. Solved for \( d \):

\[ d = \sqrt{x^2 + y^2 + z^2} \]

For example: you have an object that has a width of 6, a depth of 2 and a height of 3. What is the distance from the lower left front corner to the upper right back corner?

We take the new equation, fill in the given values and start solving:

\[ d = \sqrt{6^2 + 2^2 + 3^2} = \sqrt{36 + 4 + 9} = \sqrt{49} = 7 \]

So the asked distance in this case is 7. Talking of mathematics, this isn’t any harder than in two dimension. The equation is as easy to use as the two dimensional version. The most difficult thing is to imagine the third dimension and the distance you want to solve. Once you can picture the third dimension, the theorem is fairly simple to apply.
Conclusion

In this article we discussed the Pythagorean theorem for two and for three dimensions. We showed where this theorem could be used for in practice, and how you can program these equations into your game. Of course there are more mathematical theorems that are used in game development, some of them we will discuss in other articles. We hope you now have a better picture of where mathematical theorems could be used for in game development, and how they should be applied to codes. The point we want to make with this article, is that mathematical theorems are required for the calculation of some fundamental variables in a game. And that therefore the understanding of some mathematical theorems is a must for game programmers.

Have any questions? Want to comment?
Contact the author:
krister@softlion.nl