Texture-mapping starts with an interesting image

Let’s say that we want to do bump-mapped displacements with these bricks. For certain types of textures, like this one, you could write a program to examine the texture texel-by-texel and come up with an approximate normal vector at each texel and then encode this into another texture image. This is called a normal map.
Getting the normals by analyzing the texture – the Normal Map

Red : nx
Green : ny
Blue : nz

Much red: nx ~ +1.
No red:    nx ~ -1.
Much green: ny ~ +1.
No green:  ny ~ +1.
Much blue: nz ~ +1.

Interpreting this image is a little tricky. Normal vector components run from -1. to +1. But, color channels run from 0. to 1. So, a color value of 0. is needed to correspond to a normal component of -1., and a color value of 1. is needed to correspond to a normal component of +1. In this case, green is encoded upside-down.

Original Texture Map and Normal Texture Map
We can use the color texture image on top of a surface

Geometry you are displaying

And then you get something like this
But, what if the surface really has displacements, but you would only see them if you were using more geometric detail?

Geometry you are displaying

What you really have

Even turning on texture-mapping only puts the flat texture on the flat surface

Geometry you are displaying

What you really have
We could get the normals from the normal map and perform bump-mapping

That is good, but . . .
... we can do even better – Parallax Mapping

Geometry you are displaying

... it would be displaying this one

... it displays this texture color.

But if the displacements were really here, ...

When the eye looks here...

The inner-loop of Parallax Mapping

Slopes are perpendicular to the normal map