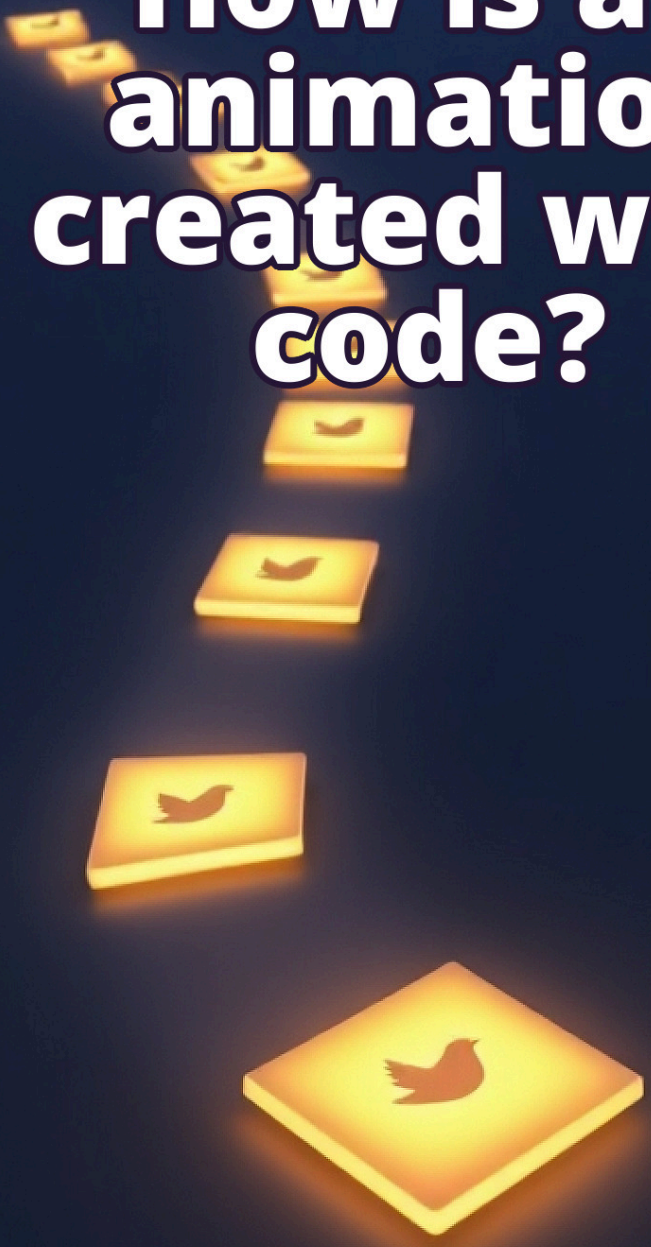
How is an animation created with code?

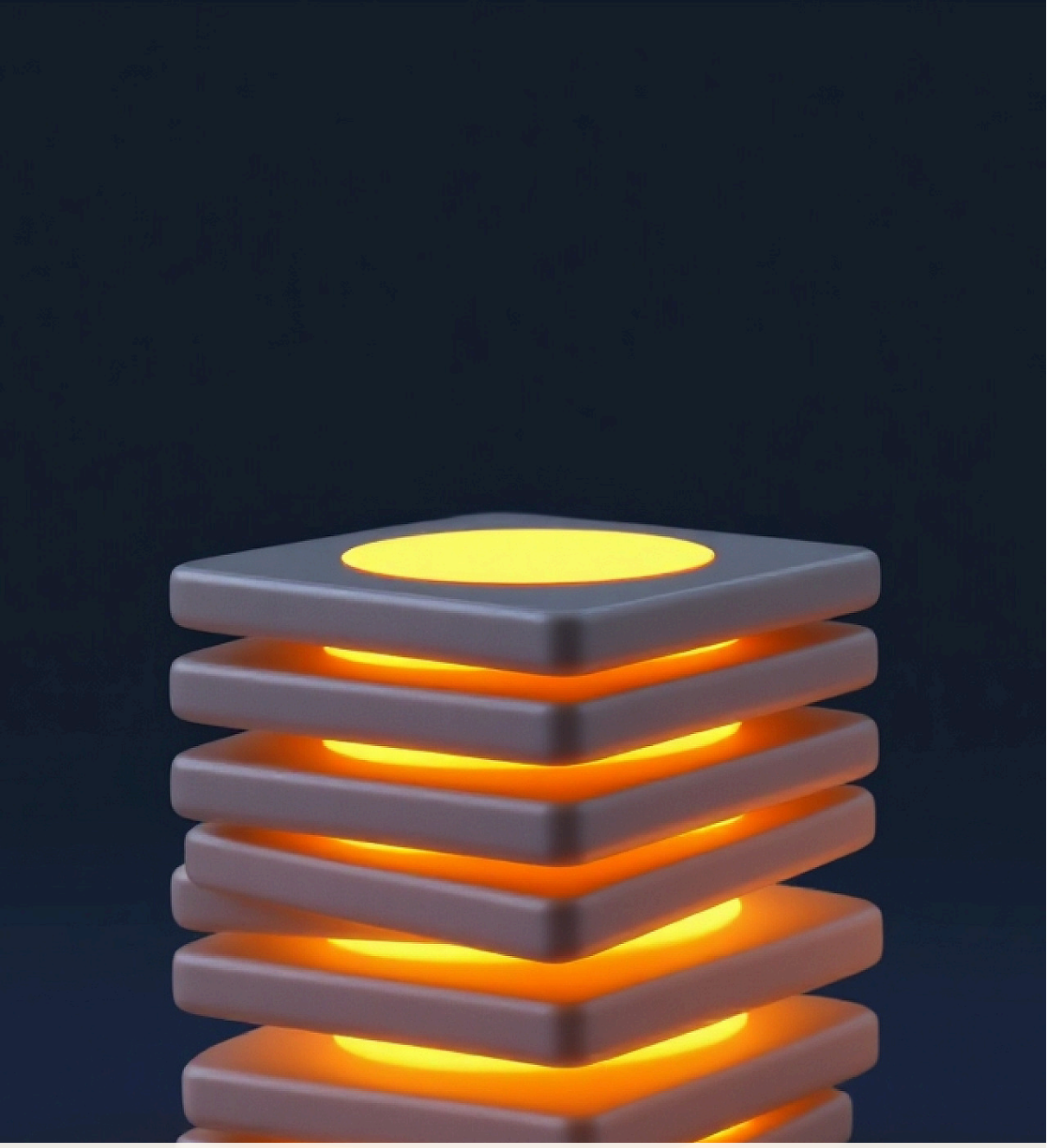


www.yzokul.com.tr

How is an animation created with code?



How does code make an animation? The short answer is that code changes an image's properties over time, and your eye sees that change as motion. Properties are things like where the image is, how big it is, or what color it has. When these change step by step, the picture appears to move. No magic is involved; it is just repeated changes.



An animation is simply a picture whose properties keep changing. The same image is drawn again and again, each time with small differences. Your eye blends these slightly different pictures into smooth movement. Nothing magical happens; the computer just redraws the picture with new values. This repeated redrawing is the core of all code animation.



Code can change several properties of an image: its position on the screen, its size, its color, and which picture is shown. Changing any one of these alone can already create a visible animation. For example, moving an object's position makes it slide, while changing its size makes it grow. You can combine them for richer effects.



To make an object slide to the right, code increases its horizontal position by a small amount each time the picture is drawn. For example, start at position 0 and add 5 each frame. The object appears to move smoothly across the screen. The horizontal position is often called the x-coordinate. This simple addition creates sliding motion.



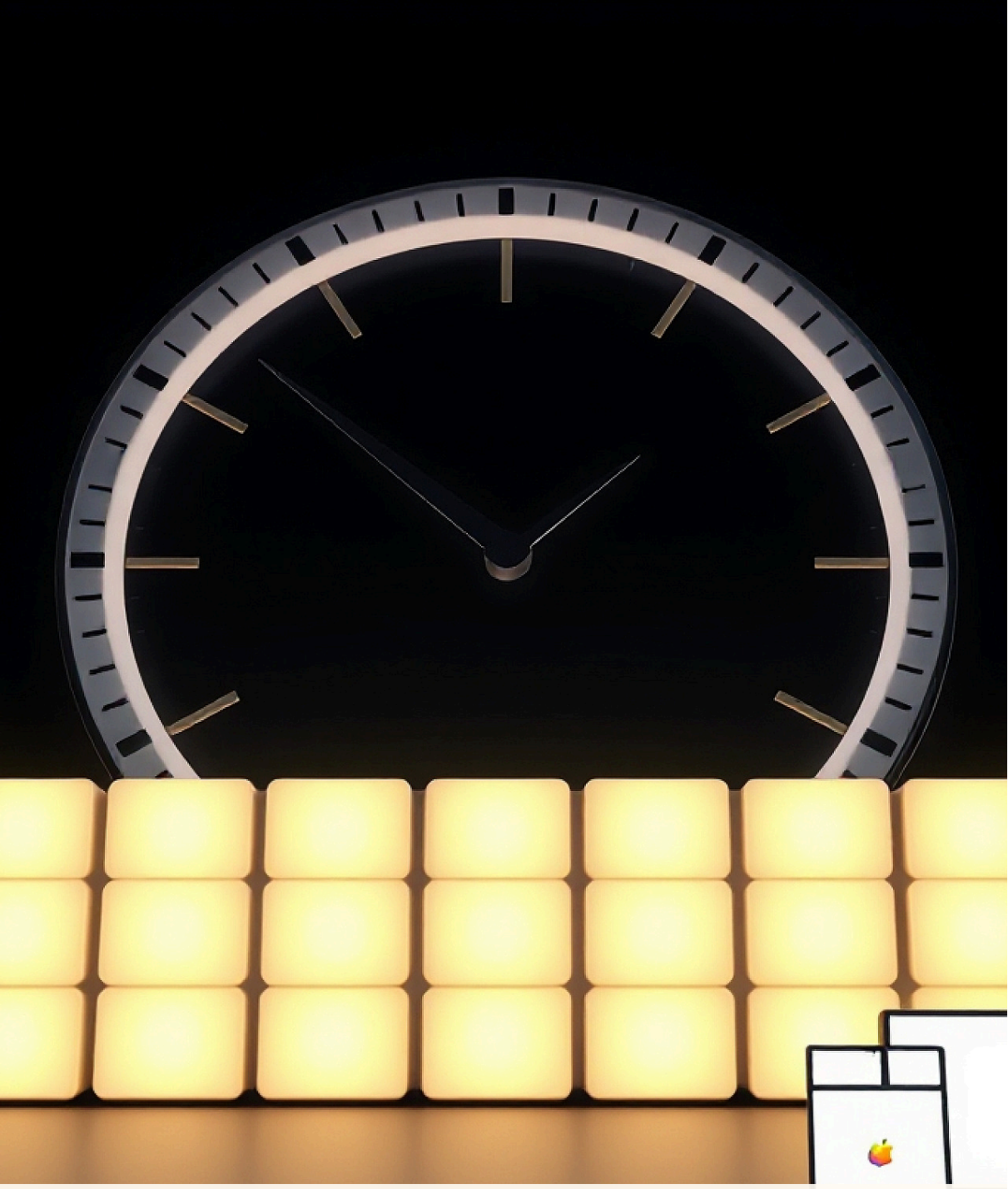
Changing an object's size step by step makes it grow bigger or smaller. For example, increase its width and height by 2 pixels each frame. Changing its color gradually, such as adding a little red each time, makes it shift hue. These property changes also read as animation, even without any movement across the screen. They show that motion is not the only kind of animation.



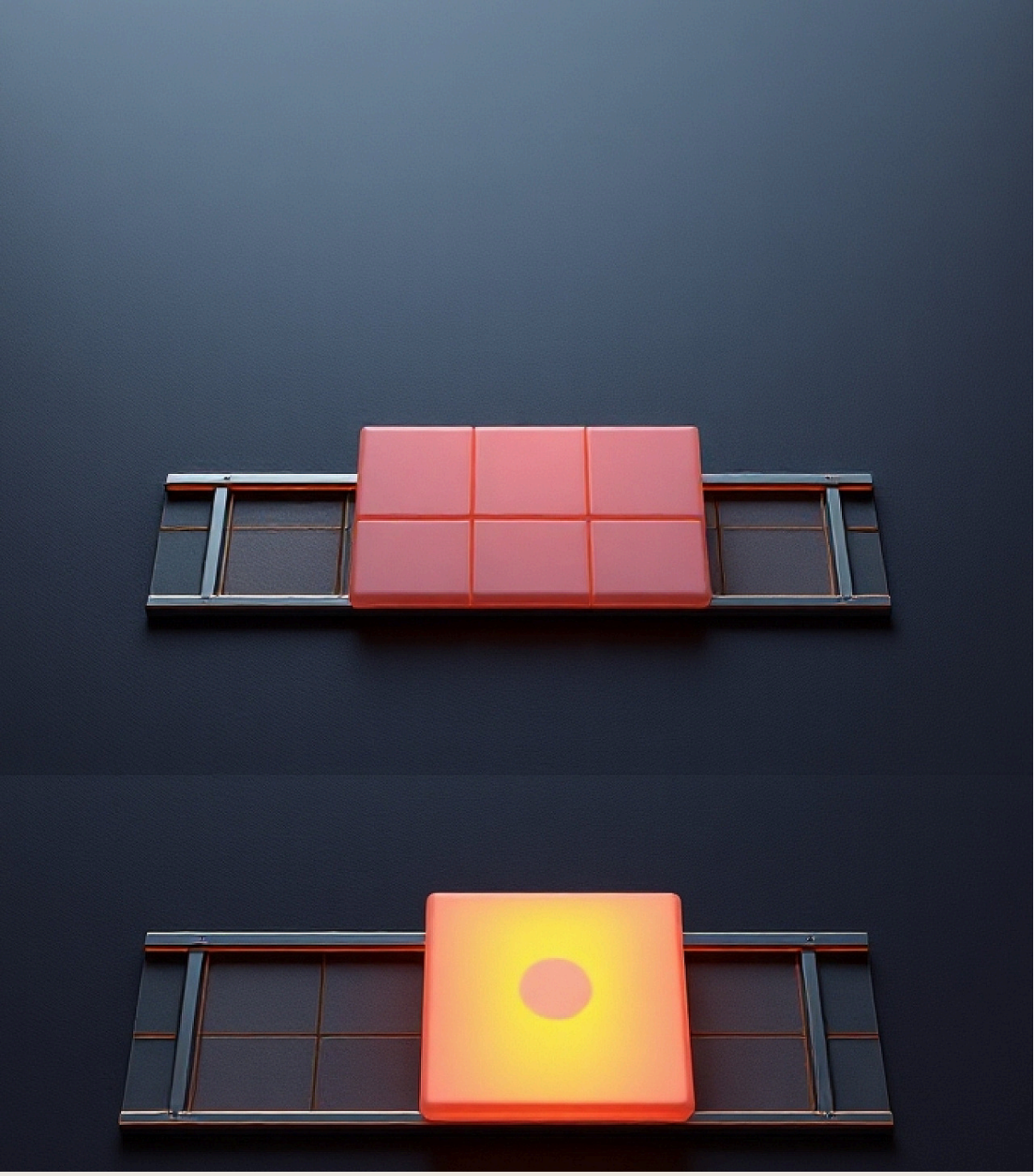
Animation is a series of still pictures shown one after another. Each still picture is called a frame. When frames are shown quickly, your eye sees them as continuous motion. Think of a flipbook: each page is a frame, and flipping fast creates movement. Code creates frames by drawing the picture with updated properties.



To make a character look like it is moving, code shows different drawings of that character in sequence. For example, show a leg lifted, then straight, then lifted again. Each drawing is a frame. When shown in order quickly, the character appears to walk. The code simply switches which picture is displayed at each frame.



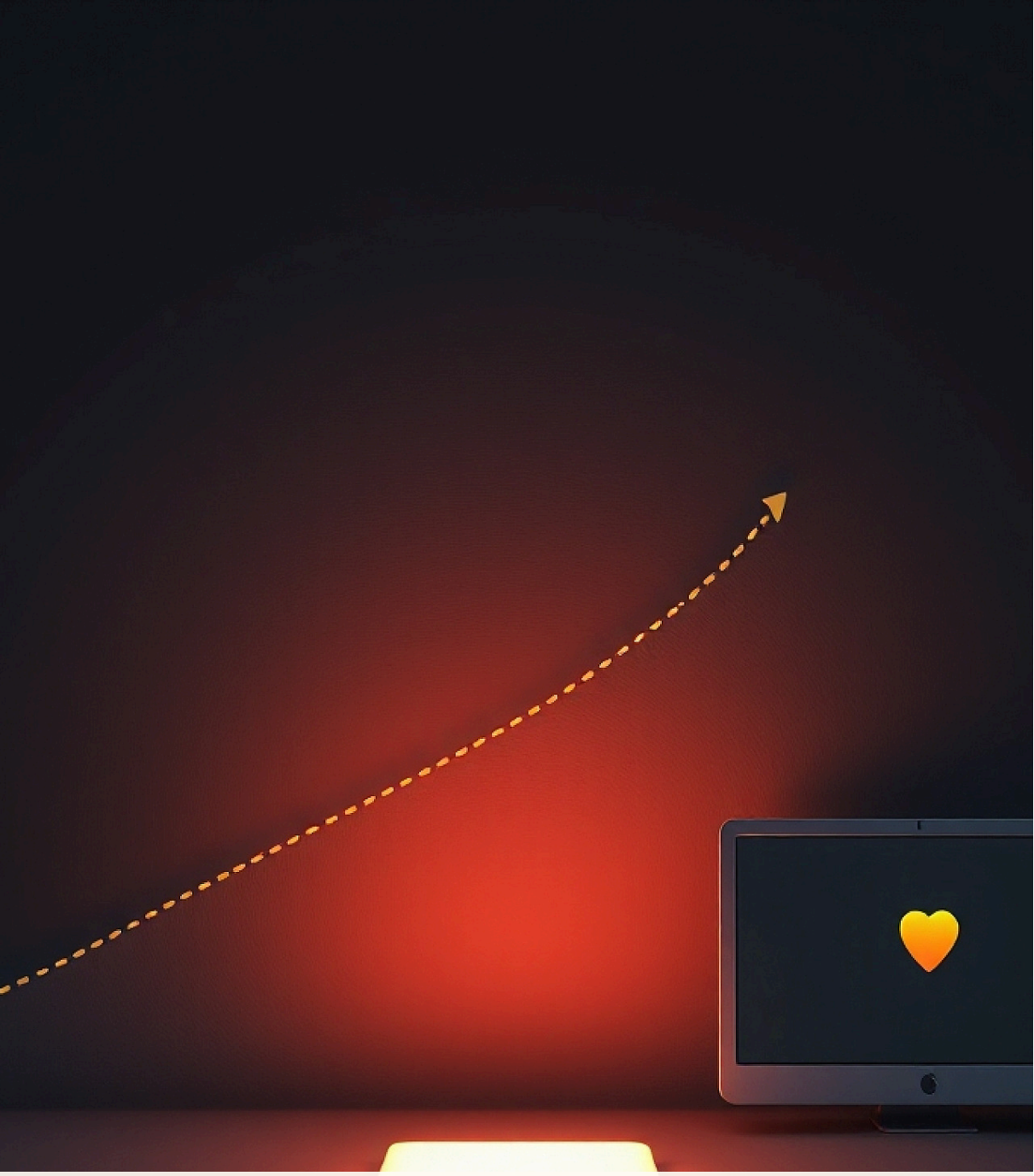
The time between frames matters a lot. If frames come too slowly, the motion looks jumpy, like a slideshow. If they come fast enough, usually many frames per second, the motion looks smooth. Code uses a timer or a loop to control when the next frame is drawn. Consistent timing is key to smooth animation.



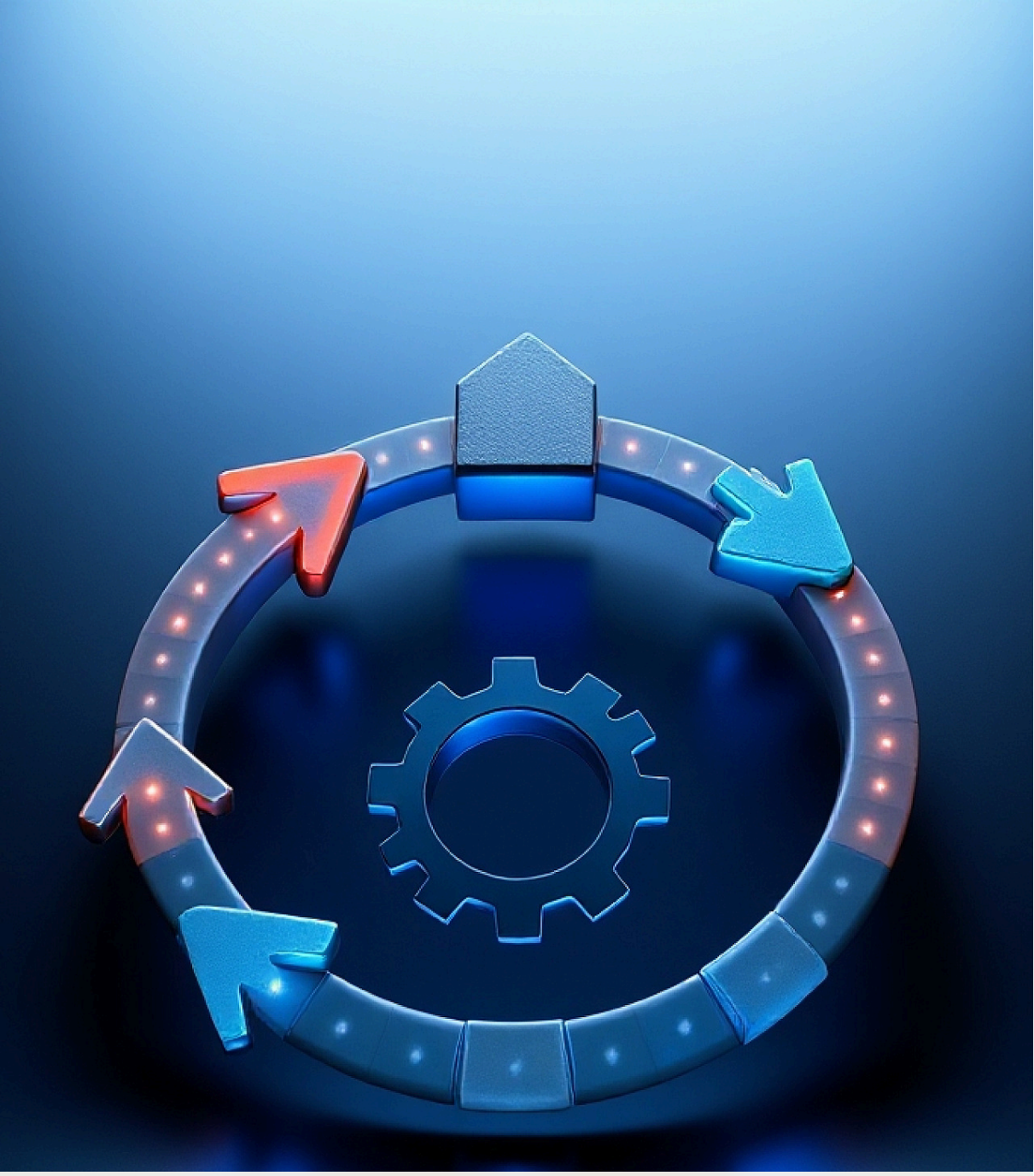
A bigger change per frame makes faster motion, and a smaller change makes slower motion. For example, adding 10 to the position each frame moves twice as fast as adding 5. Speed is set by how much you change each time. You can also change the time between frames to affect speed. Both the step size and the timing control how fast the animation appears.



Code uses simple calculations to decide the next frame. For example, it adds a number to the current position to get the new position. These calculations control how the motion looks. You can use multiplication or other math to create curves or acceleration. The calculations are the rules that drive the animation forward.



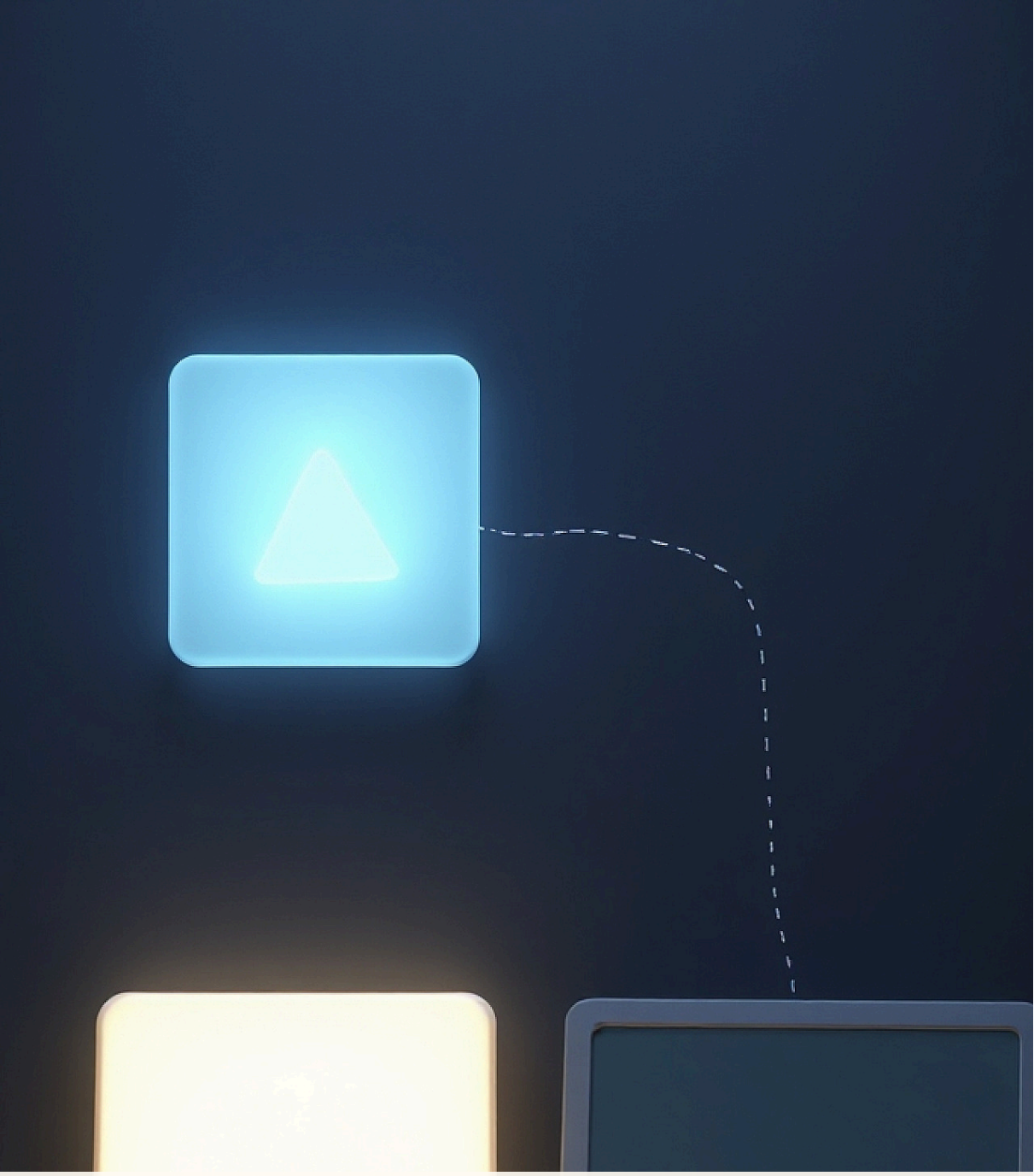
The way values change over time gives the transition style. A steady change, like adding the same amount each frame, looks even and linear. Other patterns, like starting with small changes and increasing them, make motion start slow and speed up. Code can use different formulas to create these styles. Transition style affects the feel of the animation.



Code repeats the same steps over and over: change a property, draw the picture, wait a short time, and repeat. This repeating loop is what keeps the animation going. Without the loop, the animation would stop after one change. The loop runs continuously, updating and redrawing the picture many times per second.



Follow one full example: start an object at position 0. Each frame, add 5 to its position, wait 1/60th of a second, and redraw it. The object slides to the right smoothly. Seeing all the parts together—starting value, change per frame, wait, and redraw—shows how the motion appears. This is the essence of code animation.



A common beginner mistake is changing a value but never drawing the picture again. For example, you update the position variable but forget to call the draw function. Without a redraw, nothing on the screen changes. Always remember to redraw after changing properties. The screen only shows what you last drew.



Another mistake is wrong timing: frames shown too fast, too slow, or at uneven times. If frames are too fast, the motion may be too quick to see; if too slow, it looks jerky. Uneven timing makes the animation look broken. Use a consistent timer or loop to control frame timing. Test and adjust the timing for smooth results.



To sum up, animation is a picture whose properties change over time, shown frame by frame in a loop. Timing and calculations control the speed and style of the motion. Change position, size, color, or the displayed picture to create different effects. Remember to redraw and keep timing consistent. With these basics, you can start creating your own code animations.