

Dividing by Zero

Prepared by: Teton Multimedia (www.TetonMultimedia.com)

Computers cannot divide by zero. Does this mean that computers are stupid? Well, yes... and no.

Let's walk through the logic. What is one divided by one? One. What is one divided by one tenth? Ten. What is one divided by one hundredth? One hundred. What is one divided by one billionth? A billion. See the pattern here? As the denominator gets smaller and smaller, and approaches zero, the resultant gets bigger and bigger, approaching... infinity.

What is infinity? Well gosh, that's hard to say exactly. It's um... really really big.

Getting back to the nature of computers, they are machines that work with things that are exact. And infinity is not exact. You cannot point at it and say, "See this thing here? This is infinity."

More correctly, computers cannot divide by extremely small numbers. This is because they cannot handle the resulting numbers that are *bigger* than a specific limit. This limit is imposed by the precision of their math — and this precision is defined by their human designers.

All computational devices have this limit. For example, I have an old Hewlett-Packard calculator. If I ask the calculator to take 99 to the 99th power, the result is a blinking "9.999999 E+99". This simply means that the actual answer is bigger than the value shown, and the value shown is the biggest the calculator can handle.

This type of problem is known as an "overflow error." Note that this type of error *can* occur when performing addition, subtraction, multiplication, or division. However, it rarely occurs except as a consequence of division (when dividing by zero or an extremely small number).

Now you may be thinking, "Gosh, this seems a bit esoteric. Why is this guy babbling on like this? It sure seems unlikely that I need to worry about dividing by zero." Well, as it turns out, "divide by zero" errors seem to pop up all the time. For example, in our basic trigonometry lesson you'll see that:

1.1

$$\text{Tan}(A) = y / x$$

This implies that division by zero is always something to worry about in basic trigonometry. Whenever you have a vertical line and are interested in its angle or slope, then division by zero *will be a concern*.

Lingo! (Handling the Problem)

Spinning wheels, spinning clock hands, etc. Division by zero will happen unless you prevent it from happening. This type of prevention is often referred to as "trapping." You

trap the zeroes (or small numbers) and handle these cases in a specific manner. For example, we usually aren't actually interested in the Tangent of an angle; instead we're actually interested in the angle itself. In Director, distances between two sprites are expressed in pixels. Because sprite locations are whole numbers of pixels (not fractions), the distance between them will be zero whenever it is less than one. (Note that we're talking about absolute values when referring to distances.) Consequently, we can often trap the zeroes simply by testing for $\text{abs}(x) < 1$.

Here's a code snippet from our "spinning arrow" example, in which an arrow automatically rotates to point at the mouse. Note that without the "divide by zero" trap, the code crashes whenever the mouse is directly above or below the arrow's center of rotation.

```
x = float(the mouseh - MyS.Loch) -- x from Equation 1.1.  
y = float(the mousev - MyS.LocV) -- y from Equation 1.1.
```

```
if abs(x) < 0.1 then -- trap (avoid) divide by zero.
```

```
  if y > 0 then  
    myRads = Pi * 0.5 -- (point straight down)  
  else  
    myRads = -Pi * 0.5 -- (point straight up)  
  end if
```

```
else -- regular case (no divide by zero).
```

```
  myTan = y / x  
  myRads = ATan(myTan)
```

```
  if x < 0 then  
    if y > 0 then  
      myRads = myRads + Pi  
    else  
      myRads = myRads - Pi  
    end if  
  end if
```

```
end if
```

```
myDegrees = myRads * 180.00 / Pi  
MyS.rotation = myDegrees
```

Note that other situations may require a different technique to handle the division by zero. As the programmer, you'll simply have to consider the range of possibilities, and decide how best to handle them.