

Known Slugs

This lesson deals with functions which make your C program slower.

We'll cover the following ^

- Slow math functions

There are some C functions that are known to be slow.

Slow math functions

`pow()`, `sqrt()`, and trigonometric functions (e.g. `sin()`, `cos()`, `tan()`, etc.

When using `pow()` with integers, just use basic operators instead. For example instead of this:

```
double y = 3.14;
double x = pow(y, 3);
```



Use the `*` operator instead:

```
double y = 3.14;
double x = y*y*y;
```



You can sometimes get around having to compute `sqrt()`, e.g. by squaring instead (but not using `pow()`). For example, let's say we're testing whether the distance of a point (x1,y1) from another point (x2,y2) is less than some minimum (mindist).

Instead of computing the actual distance like this:

```
double the_dist = sqrt( pow(x2-x1,2) + pow(y2-y1,2) );
if (the_dist < mindist) {
    printf("it is less\n");
}
```



You can test the squared distance against the squared mindist:

```
double xdif = x2-x1;
double ydif = y2-y1;
double the_dist_squared = (xdif*xdif) + (ydif*ydif);
```



```
double the_dist_squared = (xdif * xdif) + (ydif * ydif);  
if (the_dist_squared < (mindist * mindist)) {  
    printf("it is less\n");  
}
```

Note how we have also replaced `pow()`, and we have made temporary variables `xdif` and `ydif` so we only compute each difference once.

In our lab, we got rid of a bunch of `pow()` function calls in a C function that represented a muscle model in an arm model simulation, and we sped up the code by a factor of about two (twice as fast).

Another way to run code faster is to use optimizer flags. We'll get familiar with these flags in the next lesson.