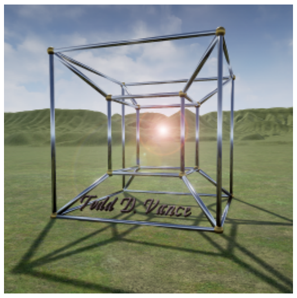

Pointers

How To

WHAT ARE THEY AND HOW DO THEY WORK

January 5, 2017

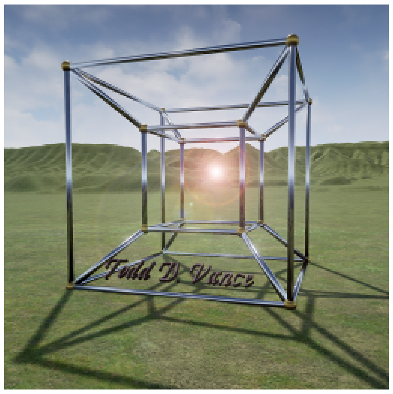

Figure 1: Hypercube projected into the South Branch Valley

Contents

1 Introduction

Pointers are hard. They are difficult enough that modern languages like Python, Ruby, and Java were designed to not use pointers. But if you are programming C, C++, and sometimes even C# (using the "unsafe" code), you will encounter pointers. Understanding pointers will enrich the understanding of languages that don't use them, like Python, Ruby, and Java, in the same way that understanding some automobile mechinics enhances the understanding of the car, even when all you do is drive it. There is a reason that the metaphor "under the hood" is used for some difficult programming concepts, after all.

So, if you are taking the Udemy Unreal course (`https://www.udemy.com/unrealcourse/learn/v4/overview`) taught by Ben Tristem and Sam Pattuzzi, you will encounter some lectures on pointers, because writing games for Unreal requires at least some knowledge of pointers. This HowTo is intended to go beyond the very basics and put a person on a path of "Grokking" (as Robert A. Heinlein would say) pointers "in the fullest".

The primary prerequisite for this HowTo is a knowledge of some programming language, specifically a knowledge and some experience with arrays. I shall assume you know how to work with arrays in, for example, C++, or really any language that uses them (which is most languages, Lisp and Assembly being notable exceptions).

2 Start with Arrays

Most student programmers are introduced to arrays before being introduced to pointers, because it is easier to understand. An added advantage is that, pointers can be “modeled” by arrays so this is a big step toward understanding pointers.

In summary, if a variable a is an array then it has some properties (which are specified differently in different languages), the main one being the *length*, sometimes called *dimension*. So, the following commands define an array of length 10 in various languages:

BASIC:	DIM A(10)
C or C++:	int a[10];
Java:	int[] a = new int[10];
Python:	a = [0]*10;

So, arrays have a length. They also have *elements* indexed by integers. In BASIC, the elements are $A(1)$, $A(2)$, $A(3)$, $A(4)$, $A(5)$, $A(6)$, $A(7)$, $A(8)$, $A(9)$, and $A(10)$. In the other languages, the elements are $a[0]$, $a[1]$, $a[2]$, $a[3]$, $a[4]$, $a[5]$, $a[6]$, $a[7]$, $a[8]$, and $a[9]$. Since almost all modern languages follow a pattern simimilar to C, C++, Java, and Python, that is what I shall use in this HowTo.

These are the main properties of arrays. An array is a single variable that holds an indexed (starting with zero, ending with length - 1 in most langauges) list of values.

Of course, the examples were arrays of integers (except BASIC, where the default is floating point numbers for most dialects). But arrays in most languages can hold any value, though except for some exceptions like Python, the values must all be the same type.

3 The Memory Array

At the lowest level, the memory in a computer holds numbers. So, the memory looks something like:

```
int[] a = new int[NUMBER_OF_WORDS];
```