

## Re: Writing float to process memory

**Source:** <http://coding.derkeiler.com/Archive/Delphi/comp.lang.pascal.delphi.misc/2004-10/0349.html>

---

**From:** Rob Kennedy (*me3\_at\_privacy.net*)

**Date:** 10/11/04

Date: Sun, 10 Oct 2004 23:14:09 -0500

Daniel Hjerholm wrote:

> *I am trying to make a trainer for Age of Empires Expansion that gives me  
> t.ex. 10000 wood. I have found the address with ArtMoney, and i can also  
> change the value from ArtMoney, but not from Delphi. The problem is that the  
> value is a 4 byte float variable, and I only know how to write integer  
> variables to the memory. If i change the value to an integer I get 0 woon in  
> AoE.*

I find it strange that Age of Empires would use a floating-point value for that; you can only have integral amounts of wood. A four-byte floating-point value is called a Single in Delphi.

> *Here is my code:*

>  
> *procedure TForm1.Button1Click(Sender: TObject);*  
> *const*  
> *PokeValue = 10000;*  
> *var*  
> *h: hwnd;*  
> *ThreadId: Integer;*  
> *ProcessId: Integer;*

Thread and process IDs are unsigned values. Declare those as DWord, not Integer.

> *MemHandle: Integer;*

What you have is a handle to a process, not a handle to memory. Picking accurate names for variables goes a long way toward understanding what the code does. Also, since it's a handle, declare it as a THandle.

> *buf: PChar;*  
> *write: cardinal;*  
> *begin*  
> *h := FindWindow(nil, 'Age of Empires Expansion');*  
> *if h = 0 then Exit;*  
> *ThreadId := GetWindowThreadProcessId(h, @ProcessId);*  
> *MemHandle := OpenProcess(PROCESS\_ALL\_ACCESS, False, ProcessId);*

Don't request *\*all\** access unless you've determined that you need all access. For instance, it looks like you only need write access, not read access, or synchronization access. Requesting only the minimum permissions increases the chances that the function will return a valid handle instead of returning an error code. Don't forget to check for error codes, by the way.

```
> GetMem(buf, 8);
```

What's the significance of the magic number 8? If you need a four-byte buffer, then why are you allocating an eight-byte buffer?

```
> buf^ := Chr(PokeValue);
```

That doesn't do what you think it does. First, it converts the value 10000 into a Char. (Since the maximum Char value is #255, I'm not really sure what you'll get — probably the least significant byte of 10000, which is #10.) Next, that Char value is stored into the first byte of the buffer. The remaining seven bytes are left untouched, so they continue holding whatever garbage values they had before you allocated the memory.

You really don't need to allocate any additional memory at all. All you need to pass to WriteProcessMemory is a pointer to a Single, right? So declare a Single value and pass a pointer to it:

```
var
```

```
    WoodAmount: Single;
```

```
WoodAmount := 10000;
```

```
Win32Check(WriteProcessMemory(MemHandle, Addr($f675b4), @WoodAmount,  
    SizeOf(WoodAmount), BytesWritten));
```

```
Assert(SizeOf(WoodAmount) = BytesWritten);
```

```
> WriteProcessMemory(MemHandle, Pointer($00F675B4), buf, NumberOfBytes,
```

```
> write);
```

Use a named constant for that address. "WoodValueLocation" might be a good name.

And remember to check the return value of that function. Trampling on another process's memory is a delicate task, so that function is bound to fail someday, and when it does, you should have some plan for what to do about it.

Rather than blithely writing to that address, it might make sense to read the current value and some values nearby to ensure that you're writing to the field you intended. I imagine the magic address would change if a patch or update were released for the program.

```
--
```

comp.lang.pascal.delphi.misc: Re: Writing float to process memory

Rob