



Lucas PARSY



*"It's
pronounced
Aine jean"*

Video game hacking:

Cheat Engine Workshop



Resources

- Cheat Engine: cheatengine.org
- My game, Flag Quest:
See QR code
Game, writeup &more
- USB key with everything!



tuxlu.fr/talk_vghacking



- Memory scanner and debugger

Cheat Engine 7.5

File Edit Table D3D Help

00006764-flag_quest_release.exe

Found: 678

Address	Value	Previous	First
E80A9FF35C	55.5	55.5	-0.00003051...
E8B89FF2DC	255.2008057	255.2008057	250.9992065
E8B89FF2F4	255.2008057	255.2008057	250.9992065
E8B89FF30C	55.08114624	55.08114624	-0.00003051...
E8B89FF32C	255.2008057	255.2008057	250.9992065
E8B89FF33C	255.2008057	255.2008057	250.9992065
E8B89FF34C	0	0	0
E8B8D9FF2E0	62.46905518	62.46905518	59.99999987E14
E8B8E9FF2E0	27.27392578	27.27392578	86.62145996
E8B8E9FF334	256.563446	256.563446	250.9993896
E8B8E9FF34C	55.08119202	55.08119202	-0.00003051...
E8B8E9FF36C	256.563446	256.563446	250.9993896
E8B8E9FF3DC	256.563446	256.563446	250.9993896
E8B8F1FF350	45.99002075	45.99002075	59.99999987E14
E8B8F1FF35C	262.0027161	262.0027161	250.9993896
E8B8F1FF3BC	262.0027161	262.0027161	362
E8B8F1FF3D4	262.0027161	262.0027161	362
E8B8F1FF90C	257.9138489	257.9138489	362
E8B8F1FF97C	257.9138489	257.9138489	362
E8B8F9FF3D0	36.50170898	36.50170898	89.88651611

New Scan Next Scan Undo Scan Settings

Scan Type: Increased value Simple values only

Value Type: Float Compare to first scan

Memory Scan Options:

- All
- Start: 0000000000000000 Stop: 00007FFFFFFEFFFFE
- Writable Executable
- CopyOnWrite
- Active memory only
- Fast Scan Alignment: 4 Last Digits
- Pause the game while scanning

Speed: 1 Apply

Memory View Add Address Manually

Active	Description	Address	Type	Value
<input type="checkbox"/>	No description	28ADC91EE0	4 Bytes	5259
<input type="checkbox"/>	No description	28ADC7B33FC	Float	2064
<input type="checkbox"/>	No description	28ADC7EC544	Float	0
<input type="checkbox"/>	No description	28ADCBE410	Float	2.087934712E-43
<input type="checkbox"/>	No description	28ADCDE9528	Float	4.593582483E-40
<input type="checkbox"/>	No description	28ADCDE96A0	Float	8.407790786E-45
<input type="checkbox"/>	No description	28ADCFC090	Float	-1.892436462E33
<input type="checkbox"/>	No description	28ADC699BC0	Float	Nan
<input type="checkbox"/>	No description	28ADC7B3250	Float	0
<input type="checkbox"/>	No description	28ADD2C7CB8	4 Bytes	100004
<input type="checkbox"/>	No description	28ADD2D17F8	4 Bytes	1000000
<input type="checkbox"/>	No description	28AF6EB4040	4 Bytes	999992



- Memory scanner and debugger
- AutoAssembly and LUA scripting

```
Hook :  
retGetGamePlayers_o:  
readmem( retGetGamePlayers, 6 )  
mov [LocalPlayer],rax  
mov rcx, [rax+30]  
test rcx,rcx  
je short @f  
    mov [OakPlayerController],rcx  
    mov rcx, [rcx+488]  
    test rcx, rcx  
    je short @f  
    mov rcx, [rax+30]  
    mov rcx, [rcx+1988]  
    test rcx,rcx  
    je short @f  
        mov [OakDeveloperPerks],rcx  
        test byte ptr [rcx+C8],40  
        jne short @f  
            or byte ptr [rcx+C8],40  
  
@@:  
jmp retGetGamePlayers+6
```



- Memory scanner and debugger
- AutoAssembly and LUA scripting

```
function AOBScanAA(script, symbol)
    local success, disableInfo = autoAssemble(script)
    if not success then return nil, disableInfo end -- disable
    local addr = getAddress(symbol)
    autoAssemble(script, disableInfo) -- disable script and
    return addr, 'success'
end

function AOBScanRegion(bytestr, start, stop)
    local script = ([[[
[ENABLE]
aobscanregion(luaAOBScanRegionSymbol,%X,%X,%S)
registersymbol(luaAOBScanRegionSymbol)
[DISABLE]
unregistersymbol(luaAOBScanRegionSymbol)
]]]):format(getAddress(start), getAddress(stop), bytestr)
    return AOBScanAA(script, 'luaAOBScanRegionSymbol')
end

function AOBScanModule(bytestr, module)
    local script = ([[[
[ENABLE]
aobscanmodule(luaAOBScanModuleSymbol,%S,%S)
registersymbol(luaAOBScanModuleSymbol)
[DISABLE]
unregistersymbol(luaAOBScanModuleSymbol)
]]]):format(module, bytestr)
    return AOBScanAA(script, 'luaAOBScanModuleSymbol')
end
```



- Memory scanner and debugger
- AutoAssembly and LUA scripting
- GUI 'trainer' generator





- Memory scanner and debugger
- AutoAssembly and LUA scripting
- GUI 'trainer' generator
- **Not limited to video games**



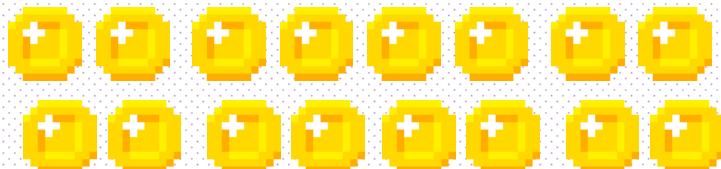
Finding values

COINS : 0



Finding values

COINS : 4



New Scan Next Scan

Value: Hex

Scan Type: Exact Value

Found: 113

Address	Value	Previous
1318B...	0	4
1318B...	0	4
13193...	4	4
13193...	4	4
13193...	4	4
13193...	4	4
13193...	4	4
13193...	4	4
13193...	4	4
15C0A...	4	4

Finding values

COINS : 20



New Scan Next Scan

Value:
 Hex

Found:4

Address	Value	Previous
15C0D...	20	20
15C0D...	20	20
15C0D...	20	20
15C0E...	20	20

Finding values



COINS : 4000



Activ	Description	Address	Type	Value
<input type="checkbox"/>	coins	00000000		
<input type="checkbox"/>	coins	15C0DD859784 Bytes	20	
<input type="checkbox"/>	coins	15C0DD859AC4 Bytes	20	
<input type="checkbox"/>	coins	15C0EC83FD84 Bytes	20	
<input type="checkbox"/>	coins	15C0DD859C84 Bytes	20	

Change Value

what value to change this to?

4000

OK

Finding values: advanced features

- Exact Value
- Exact Value
- Bigger than...
- Smaller than...
- Value between...
- Increased value
- Increased value by ...
- Decreased value
- Decreased value by ...
- Changed value
- Unchanged value
- Ignore value

Value:
value % 2 == 1 and value > previousvalue * 3

Lua formula

Hex 20

Scan Type Exact Value

Value Type 4 Bytes

Lua formula

Not

Memory Scan Options

All

Start 000000000000000000000000

Stop 00007FFFFFFFFF

Writable Executable

CopyOnWrite

Active memory only

Fast Scan 4 Alignment Last Digits

Pause the game while scanning

- 4 Bytes
- Binary
- Byte
- 2 Bytes
- 4 Bytes
- 8 Bytes
- Float
- Double
- String
- Array of byte
- All
- Grouped

Memory viewer

Description	Address	Type	Value
health	15C0DD85978	4 Bytes	1337

Memory View

Protect:Read/Write	AllocationBase=15C0DD40000	Base=
address	80 81 82 83 84 85 86 87 89	ABCDEF01234567
15C0DD85978	39 05 00 00 64 00 00 00 9...	d... \...
15C0DD85988	C8 00 00 00 16 00 00 00
15C0DD85998	80 51 69 29 5C 01 00 00 Qi)	\.....
15C0DD859A8	B0 6A 74 0E 5C 01 00 00 jt.	\.....
15C0DD859B8	C6 17 00 00 00 20 00 00Qm) \....
15C0DD859C8	13 37 00 00 00 01 00 00 .7.....	B.. \....

```
struct Player
{
    int health      = 1337;
    int ???        = ???;
    int ???        = ???;
}
```

Memory viewer

Description	Address	Type	Value
health	15C0DD85978	4 Bytes	1337

Memory View Display Type > • 4 Byte decimal

Protect:Read/Write	AllocationBase=15C0DD40000	Base=
address	78	7C
15C0DD85978	1337	100
15C0DD85988	200	22
15C0DD85998	694768000	348
15C0DD859A8	242510512	348
15C0DD859B8	6086	8192
15C0DD859C8	14099	256

```
struct Player
{
    int health    = 1337;
    int strength = 100;
    int defense   = 200;
}
```

Data structures

Description	Address	Type	Value
health	15C0DD85978	4 Bytes	1337

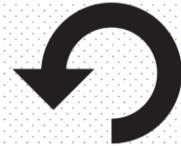
Memory View Tools  Dissect data/structures

Offset-description	Address: Value
Player	
0000 - 4 Bytes	5DA650 : 1337
0004 - 4 Bytes	5DA654 : 100
0008 - 4 Bytes	5DA658 : 200
000C - 4 Bytes	5DA65C : 22

```
struct Player
{
    int health    = 1337;
    int strength = 100;
    int defense   = 200;
}
```

Persisting memory addresses

Activ	Description	Address	Type	Value
<input type="checkbox"/>	coins	00000000		
<input type="checkbox"/>	coins	15C0DD859784 Bytes	??	
<input type="checkbox"/>	coins	15C0DD859AC4 Bytes	??	
<input type="checkbox"/>	coins	15C0EC83FD84 Bytes	??	
<input type="checkbox"/>	coins	15C0DD859C84 Bytes	??	



**Reload game
Lose everything!**

Persisting memory addresses

Solution: search *potentially static* addresses
search for all code that points to the address

coins	2216AF30F68	4 Bytes	58
	🔍 Pointer scan for this address		
Recursive scan	 <pre>op [(Address - 1) + 01] op [(Address - 2) + 02] op [(Address - 3) + 03] ...</pre>		
7FF7BB2E2724 - 48 03 41 08 - add rax,[rcx+08]	RCX=000002216AF30F60		
7FF7BB165E68 - 49 89 44 24 08 - mov [r12+08],rax	R12=000002216AF30F60		
7FF7BB1B9D40 - 49 8B 55 08 - mov rdx,[r13+08]	R13=000002216AF30F60		



Persisting memory addresses

Problem 1: no results :/

Pointerscanner scanoptions

Scan for address Scan for addresses with value Generate pointermap
 Use saved pointermap

0152E268

Nr of threads scanning: 12 Normal

Maximum offset value: 128 Max level 6

OK Cancel

4 Bytes Pointer paths 0

Base Address Offset 0 Poi



Persisting memory addresses

Problem 2: too many results!

Pointerscanner scanoptions

Scan for address Scan for addresses with value Generate pointermap
 Use saved pointermap

0152E268

Nr of threads scanning: 12 Normal

Maximum offset value: 128 Max level: 6

OK Cancel

4 Bytes Pointer paths 230860

Base Address	Offset 0	Poi
"godot.windows.opt.tools.64.exe"+07212940	10	-
"godot.windows.opt.tools.64.exe"+071DE070	60	-
"godot.windows.opt.tools.64.exe"+07212940	10	-
"godot.windows.opt.tools.64.exe"+071DE070	60	-



Persisting memory addresses

solution: rescan, and compare results

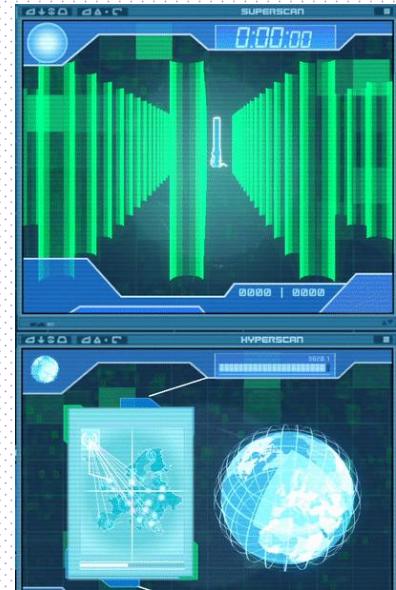
Generate pointermap → Pointer scan for this address

Filename	Address
pointermap_coins5.scandata	248C053F6E8
pointermap_coins3.scandata	1D847EDC898
pointermap_coins1.scandata	247DF1D46D8

4 Bytes Pointer paths 1

Base Address: "godot.windows.opt.tools.64.exe"+0717F820 Offset 0: 3B8 Points to: 2216AF30F68 = 206

coins	2216AF30F68	4 Bytes	74
pointerscan result	P->2216AF30F68	4 Bytes	74



What if we don't search a value?

How to search for a condition?



```
def player_move():
    if collision("coin"):
        coins += 1
    if collision("door"):
        if has_key:
            open_door()
    if button("down"):
        crouch()
```

What if we don't search a value?

How to search for a condition:
code filter



Addresses executed since last filter operation: 0		
Has been executed		Load address list
Has not been executed		From Trace
<input type="button" value="Start"/>		From Disassembler
<input type="button" value="Stop"/>		From File

Address List (46093)

Address	Executed
Tutorial-i386.exe.text+1BB5	No
Tutorial-i386.exe.text+1BBA	No
Tutorial-i386.exe.text+1BBF	No
Tutorial-i386.exe.text+1BC8	No

Memory View

Tools



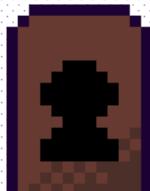
Code Filter

```
def player_move():
    • if collision("coin"):
        coins += 1
    • if collision("door"):
        if has_key:
            open_door()
    • if button("down"):
        crouch()
```

What if we don't search a value?

How to search for a condition?:

code filter



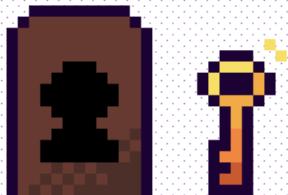
Addresses executed since last filter operation: 1791	
Has been executed	
Has not been executed	
Start	Stop
Address List (44302)	
Address	Executed
Tutorial-i386.exe.text+1BB5	No
Tutorial-i386.exe.text+1BBA	No
Tutorial-i386.exe.text+1BBF	No
Tutorial-i386.exe.text+1BC8	Yes

```
def player_move():
    if collision("coin"):
        coins += 1
    ● if collision("door"):
    ●     if has_key:
            open_door()
    if button("down"):
        crouch()
```

What if we don't search a value?

How to search for a condition?:

code filter



Addresses executed since last filter operation: 595

Has been executed
Has not been executed

Start Stop

Address List (595)

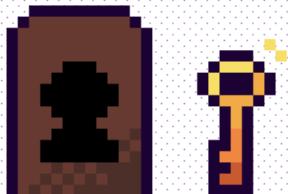
Address	Executed
Tutorial-i386.exe.text+1BB5	No
Tutorial-i386.exe.text+1BBA	No
Tutorial-i386.exe.text+1BBF	No
Tutorial-i386.exe.text+1BC8	Yes

```
def player_move():
    if collision("coin"):
        coins += 1
    ● if collision("door"):
    ●     if has_key:
            open_door()
    if button("down"):
        crouch()
```

What if we don't search a value?

How to search for a condition?:

code filter



Addresses executed since last filter operation: 1		
Has been executed		
Has not been executed		
Start	Stop	
Address List (1)		
Address	Executed	
Tutorial-i386.exe.text+1BB5	Yes	

```
def player_move():
    if collision("coin"):
        coins += 1
    ● if collision("door"):
    ●     if has_key:
            open_door()
    if button("down"):
        crouch()
```

Instruction patching

ASM, help!

74 02	je	Tutorial-i386.exe.text+26687
EB 49	jmp	Tutorial-i386.exe.text+266D0
A1 B0666500	►mov	eax,[Tutorial-i386.exe+2566B0]
3B 45 E8	cmp	eax,[ebp-18]
74 02	je	Tutorial-i386.exe.text+26693
EB 1F	jmp	Tutorial-i386.exe.text+266B2
C7 45 E8 00000000	►mov	[ebp-18],00000000
6A 00	push	00



Instruction patching

ASM primer

je if ==
jne if !=

jg if >
jl if <

add +=
sub -=

mov x=y

nop do nothing
(padding)

Instruction patching

Replace the *has_key* condition

74 02	je	Tutorial-i386.exe.text+26687
EB 49	jmp	Tutorial-i386.exe.text+266D0
A1 B0666500	►mov	eax,[Tutorial-i386.exe+2566B0]
3B 45 E8	cmp	eax,[ebp-18]
74 02	je	Tutorial-i386.exe.text+26693
EB 1F	jmp	Tutorial-i386.exe.text+266B2
C7 45 E8 000...	►mov	[ebp-18],00000000
6A 00	push	00

```
def player_move():
    if collision("coin"):
        coins += 1
    if collision("door"):
        if has_key:
            open_door()
    if button("down"):
        crouch()
```

Instruction patching

Replace the *has_key* condition

74 02	je	Tutorial-i386.exe.text+26687
EB 49	jmp	Tutorial-i386.exe.text+266D0
A1 B0666500	►mov	eax,[Tutorial-i386.exe+2566B0]
3B 45 E8	cmp	eax,[ebp-18]
74 02	je	Tutorial-i386.exe.text+26693
EB 1F	jmp	Tutorial-i386.exe.text+266B2
C7 45 E8 000...	►mov	[ebp-18],00000000
6A 00	push	00

```
def player_move():
    if collision("coin"):
        coins += 1
    if collision("door"):
        if has_key:
            open_door()
    if button("down"):
        crouch()
```

Instruction patching

Replace the *has_key* condition

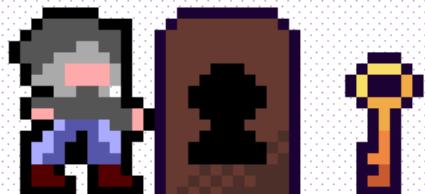
74 02	je	Tutorial-i386.exe.text+26687
EB 49	jmp	Tutorial-i386.exe.text+266D0
A1 B0666500	mov	eax,[Tutorial-i386.exe+2566B0]
3B 45 E8	cmp	eax,[ebp-18]
75 02	jne	Tutorial-i386.exe.text+26693
EB 1F	jmp	Tutorial-i386.exe.text+266B2
C7 45 E8 000...	mov	[ebp-18],00000000
6A 00	push	00

```
def player_move():
    if collision("coin"):
        coins += 1
    if collision("door"):
        if not has_key:
            open_door()
    if button("down"):
        crouch()
```

Instruction patching

Replace the *has_key* condition

74 02	je	Tutorial-i386.exe.text+26687
EB 49	jmp	Tutorial-i386.exe.text+266D0
A1 B0666500	mov	eax,[Tutorial-i386.exe+2566B0]
3B 45 E8	cmp	eax,[ebp-18]
75 02	jne	Tutorial-i386.exe.text+26693
EB 1F	jmp	Tutorial-i386.exe.text+266B2
C7 45 E8 000...	mov	[ebp-18],00000000
6A 00	push	00



Instruction patching

What if we could do it from known memory addresses?

coins 01723548 4 Bytes 100 Find out what writes to this address

The following opcodes write to 01723548

Count	Instruction
1	004272D7 - 89 02 - mov [edx],eax

.....

Tutorial-i386.exe.text+262D7:
004272CE - 8B 15 B0666500 - mov edx,[Tutorial-i386.exe+2566B0]
004272D4 - 8B 45 F0 - mov eax,[ebp-10]
004272D7 - 89 02 - mov [edx],eax <<

EAX=0000037F
EBX=00000000

- [Replace](#)
- [Show disassembler](#)
- [Add to the codelist](#)
- [More information](#)
- [copy memory](#)

Instruction patching: problems

Less obvious instructions

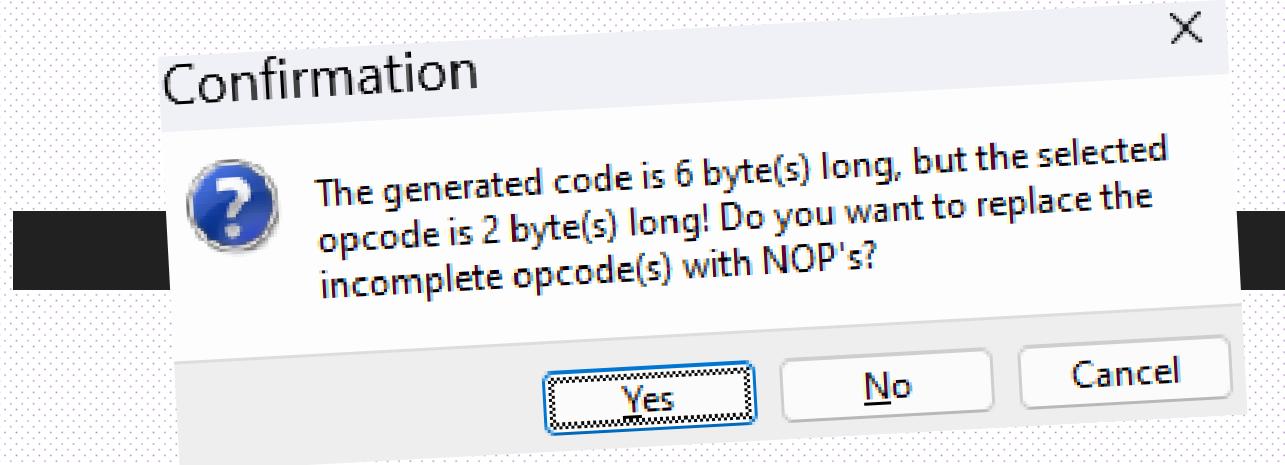
```
mov      [edx],eax
```

where **add** ?



Instruction patching: problems

Instruction size clash



Instruction patching: problems

Instruction size clash

Where to add our code?

```
mov      eax,[ebp-10]
```

```
mov      [edx],00001000
```

```
mov      edx,[T000010006.exe+2566B0]
```

Autoassemble

Solution: auto assembler!

- Auto-allocate memory
- Create complex methods
- Persisting and toggleable

```
[ENABLE]
alloc(newmem,2048)
label(return)
alloc(multiplier, 2)
registerSymbol(multiplier)
```

```
multiplier:
dd (int)5
```

```
newmem:
mov eax, [edx]
add eax, [multiplier]
mov [edx],eax
mov eax, [Tutorial-i386.exe+2566B0]
jmp return
```

```
"Tutorial-i386.exe"+272D7:
jmp newmem
nop 2
return:
```

```
[DISABLE]
dealloc(multiplier)
unregisterSymbol(multiplier)
```

```
dealloc(newmem)
"Tutorial-i386.exe"+272D7:
db 89 02 A1 B0 66 65 00
```



Autoassemble

Solution: **auto assembler!**

- Auto-allocate memory

```
alloc(newmem,2048)
label(return)
```

```
newmem:
// your code here
```

```
jmp return
```

```
"Tutorial-i386.exe"+272D7: //original
jmp newmem
nop 2
return:
```

Autoassemble

Solution: **auto assembler!**

- Auto-allocate memory
manages labels, variables...

<input type="checkbox"/>	multiplier	018E0800	4 Bytes	5
--------------------------	------------	----------	---------	---

```
alloc(newmem,2048)
label(return)
alloc(multiplier, 2)
registerSymbol(multiplier)
```

```
multiplier:
    dd (int)5
```

```
newmem:
mov eax, [edx]
add eax, [multiplier]
```

```
jmp return
```

```
"Tutorial-i386.exe"+272D7:
jmp newmem
nop 2
return:
```

Autoassemble

Solution: **auto assembler!**

- Auto-allocate memory
- **Create complex methods**

```
alloc(newmem,2048)
label(return)
alloc(multiplier, 2)
registerSymbol(multiplier)
```

```
multiplier:
    dd (int)5
```

```
newmem:
mov eax, [edx] //coins += multiplier
add eax, [multiplier]
mov [edx],eax
mov eax, [Tutorial-i386.exe+2566B0]
jmp return
```

```
"Tutorial-i386.exe"+272D7:
jmp newmem
nop 2
return:
```

Autoassemble

Solution: **auto assembler!**

- Auto-allocate memory
- Create complex methods
- **Persisting and toggleable**

```
newmem:  
    mov eax, [edx] //coins += multiplier  
    add eax, [multiplier]  
    mov [edx],eax  
    mov eax, [Tutorial-i386.exe+2566B0]  
    jmp return
```

```
"Tutorial-i386.exe"+272D7:  
    jmp newmem  
    nop 2  
return:
```

```
[DISABLE]  
dealloc(multiplier)  
unregisterSymbol(multiplier)
```

```
dealloc(newmem)  
"Tutorial-i386.exe"+272D7:  
db 89 02 A1 B0 66 65 00
```

Autoassemble

Solution: **auto assembler!**

- Auto-allocate memory
- Create complex methods
- **Persisting and toggleable**

```
newmem:  
    mov eax, [edx] //coins += multiplier  
    add eax, [multiplier]  
    mov [edx],eax  
    mov eax, [Tutorial-i386.exe+2566B0]  
    jmp return
```

```
"Tutorial-i386.exe"+272D7:  
    jmp newmem  
    nop 2  
return:
```

```
[DISABLE]  
dealloc(multiplier)  
unregisterSymbol(multiplier)
```

```
dealloc(newmem)  
"Tutorial-i386.exe"+272D7:  
db 89 02 A1 B0 66 65 00
```

Autoassemble

Solution: **auto assembler!**

- Auto-allocate memory
- Create complex methods
- **Persisting and toggleable**

```
[ENABLE]
alloc(newmem,2048)
registerSymbol(multiplier)
registerSymbol(INJECT)
aobscanmodule(INJECT,Tutorial-i386.exe,
89 02 A1 B0 66 65 00)
```

```
multiplier:
    dd (int)5
```

```
newmem:
mov eax, [edx] //coins += multiplier
add eax, [multiplier]
mov [edx],eax
mov eax, [Tutorial-i386.exe+2566B0]
jmp return
```

```
INJECT:
jmp newmem
nop 2
return:
```

Autoassemble

Solution: auto assembler!

- Auto-allocate memory
- Create complex methods
- Persisting and toggleable

```
[ENABLE]
alloc(newmem,2048)
registerSymbol(multiplier)
registerSymbol(INJECT)
aobscanmodule(INJECT,Tutorial-i386.exe,
89 02 A1 B0 66 65 00)

multiplier:
dd (int)5

newmem:
mov eax, [edx] //coins += multiplier
add eax, [multiplier]
mov [edx],eax
mov eax, [Tutorial-i386.exe+2566B0]
jmp return

INJECT:
jmp newmem
nop 2
return:

[DISABLE]
dealloc(multiplier)
unregisterSymbol(multiplier)

dealloc(newmem)
INJECT:
db 89 02 A1 B0 66 65 00
unregistersymbol(INJECT)
```

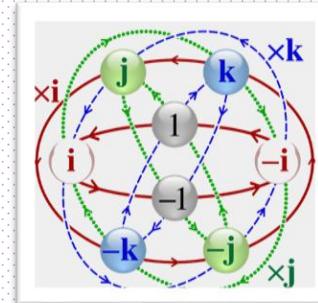
Level 3: thinking 3D

- Where 3D angles?
- Uses quaternions, rotation matrices and radian angles
- See Godot docs for more infos.

Memory View						
2B4CC81B1AC	0.00	0.00	0.00	-6798912...	0.00	0.00
2B4CC81B1C4	0.00	0.00	0.00	-1.00	SX	0.00
2B4CC81B1DC	0.00	1.00	SY	0.00	0.00	-1.00
2B4CC81B1F4	-33.93	14.65	-20.00	-1.00	0.00	0.00
2B4CC81B20C	0.00	1.00	0.00	0.00	0.00	-1.00
2B4CC81B224	-20.20	X	6.64	Y	-27.89	Z
2B4CC81B23C	0.00	1.00	1.00	1.00	0.00	0.00

Properties

Basis	basis	Basis(1, 0, 0, 0, 1, 0, 0, 0, 1)
Vector3	origin	Vector3(0, 0, 0)



Finding shared instructions

Modify an instruction, suddenly:

- Bugs appear
- Making yourself invincible makes enemies invincible too.
- Game crashes

```
player.takeDamage(100);
```



```
sub rax, (int)100
```

```
enemy.takeDamage(10);
```



```
class Entity {  
    int health = 100;  
  
public:  
    void takeDamage(int damage) {}  
};  
  
class Player : public Entity {};  
class Enemy : public Entity {};
```

100042D13 - 29 50 60 - sub [rax+60],edx

Finding shared instructions

Solution 1:  Dissect data/structures

- "Find out what addresses this code access"
- Select all: "dissect data".
- Compare value in groups

Find out what addresses this code accesses

 Accessed addresses by 100042D13

Code Address	Value	Count
The following 3 addresses have been accessed by the code		
Address	Value	Count
01541CA0	48	20
01541D40	192	5
015429C0	195	5

 Open dissect data with selected addresses

 Structure dissect:unnamed structure

Offset-description	Address: Value	Address: Value	Address: Value
unnamed structure			
0000 - Pointer	1541C40 : P->1541CE0 : P->1002B4478	1542960 : P->1002B4478	
0008 - 4 Bytes	1541C40 : 0	1541C88 : 0	1542960 : 0
000C - 4 Bytes	1541C4C : 0	1541CEC : 0	154296C : 0
0010 - 4 Bytes	1541C50 : 0	1541CF0 : 0	1542970 : 0
0014 - Double	1541C54 : 0.0(1541CF4 : 0.0078125	1542974 : 0.0078125	
001C - Float	1541C5C : 1	1541CFC : 1	154297C : 1
0020 - 4 Bytes	1541C60 : 0	1541D00 : 0	1542980 : 0
0024 - Double	1541C64 : 0.0(1541D04 : -0.0008789066	1542984 : -0.0008789066	
002C - 4 Bytes	1541C6C : 0	1541D0C : 0	154298C : 0

Finding shared instructions

Solution 2:  Find commonalities between addresses

- Auto-find common values between different calls
- Compare structures with only unique distinct values

€ Accessed addresses by 100042D13

Code Address	10	
The following 3 addresses have been accessed by the code		
Address	Value	Count
01541CA0	48	Mark selection as group 1
01541D40	192	5
015429C0	195	Mark selection as group 2

Scan for commonalities

€ Commonality scanner

Register	Status
RAX	Doubleclick to launch structure compare
RCX	Doubleclick to launch structure compare
RDX	"Group 2" has common value 0x1

€ Structure Compare : RAX

Offset	0	G1:01541C40	G2:01541CE0	G2:01542960
Max Level	28	01541C68 : 10...	01541D08 : 3209481421	01542988 : 32...
	64	01541CA4 : 100	01541D44 : 200	015429C4 : 200
Structsize	4096	01541CB0 : 0	01541D50 : 1	015429D0 : 1
element compare size	4	01541CB8 : 0	01541D58 : 256	015429D8 : 256
	70	01541CD4 : 11...	01541DB4 : 0	01542A34 : 0
	78	01541CD8 : 0	01541D5E : 0	01542A64 : 0
	D4	01541D14 : 11...	01541DF8 : 0	01542A78 : 0
	104	01541D44 : 200	01541DE4 : 0	
	118	01541D58 : 256	01541DF8 : 0	

Only find matching groups

New Scan Rescan

Finding shared instructions

Solution 3: manual search

- Some “static” values change only when relaunching the game
- “*Find commonalities*” does not work across game reboots
- Use some scripts to help scans

Browse this memory region

address	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15	B16	B17	B18	B19	B20	B21	B22	B23	B24	B25	B26	B27	B28	B29	B30	B31	B32	B33	B34	B35	B36	B37	B38	B39	B40	B41	B42	B43	B44	B45	B46	B47	B48	B49	B50	B51	B52	B53	B54	B55	B56	B57	B58	B59	B60	B61	B62	B63	B64	B65	B66	B67	B68	B69	B70	B71	B72	B73	B74	B75	B76	B77	B78	B79	B80	B81	B82	B83	B84	B85	B86	B87	B88	B89	B90	B91	B92	B93	B94	B95	B96	B97	B98	B99	B100	B101	B102	B103	B104	B105	B106	B107	B108	B109	B110	B111	B112	B113	B114	B115	B116	B117	B118	B119	B120	B121	B122	B123	B124	B125	B126	B127	B128	B129	B130	B131	B132	B133	B134	B135	B136	B137	B138	B139	B140	B141	B142	B143	B144	B145	B146	B147	B148	B149	B150	B151	B152	B153	B154	B155	B156	B157	B158	B159	B160	B161	B162	B163	B164	B165	B166	B167	B168	B169	B170	B171	B172	B173	B174	B175	B176	B177	B178	B179	B180	B181	B182	B183	B184	B185	B186	B187	B188	B189	B190	B191	B192	B193	B194	B195	B196	B197	B198	B199	B200	B201	B202	B203	B204	B205	B206	B207	B208	B209	B210	B211	B212	B213	B214	B215	B216	B217	B218	B219	B220	B221	B222	B223	B224	B225	B226	B227	B228	B229	B230	B231	B232	B233	B234	B235	B236	B237	B238	B239	B240	B241	B242	B243	B244	B245	B246	B247	B248	B249	B250	B251	B252	B253	B254	B255	B256	B257	B258	B259	B260	B261	B262	B263	B264	B265	B266	B267	B268	B269	B270	B271	B272	B273	B274	B275	B276	B277	B278	B279	B280	B281	B282	B283	B284	B285	B286	B287	B288	B289	B290	B291	B292	B293	B294	B295	B296	B297	B298	B299	B300	B301	B302	B303	B304	B305	B306	B307	B308	B309	B310	B311	B312	B313	B314	B315	B316	B317	B318	B319	B320	B321	B322	B323	B324	B325	B326	B327	B328	B329	B330	B331	B332	B333	B334	B335	B336	B337	B338	B339	B340	B341	B342	B343	B344	B345	B346	B347	B348	B349	B350	B351	B352	B353	B354	B355	B356	B357	B358	B359	B360	B361	B362	B363	B364	B365	B366	B367	B368	B369	B370	B371	B372	B373	B374	B375	B376	B377	B378	B379	B380	B381	B382	B383	B384	B385	B386	B387	B388	B389	B390	B391	B392	B393	B394	B395	B396	B397	B398	B399	B400	B401	B402	B403	B404	B405	B406	B407	B408	B409	B410	B411	B412	B413	B414	B415	B416	B417	B418	B419	B420	B421	B422	B423	B424	B425	B426	B427	B428	B429	B430	B431	B432	B433	B434	B435	B436	B437	B438	B439	B440	B441	B442	B443	B444	B445	B446	B447	B448	B449	B450	B451	B452	B453	B454	B455	B456	B457	B458	B459	B460	B461	B462	B463	B464	B465	B466	B467	B468	B469	B470	B471	B472	B473	B474	B475	B476	B477	B478	B479	B480	B481	B482	B483	B484	B485	B486	B487	B488	B489	B490	B491	B492	B493	B494	B495	B496	B497	B498	B499	B500	B501	B502	B503	B504	B505	B506	B507	B508	B509	B510	B511	B512	B513	B514	B515	B516	B517	B518	B519	B520	B521	B522	B523	B524	B525	B526	B527	B528	B529	B530	B531	B532	B533	B534	B535	B536	B537	B538	B539	B540	B541	B542	B543	B544	B545	B546	B547	B548	B549	B550	B551	B552	B553	B554	B555	B556	B557	B558	B559	B560	B561	B562	B563	B564	B565	B566	B567	B568	B569	B570	B571	B572	B573	B574	B575	B576	B577	B578	B579	B580	B581	B582	B583	B584	B585	B586	B587	B588	B589	B590	B591	B592	B593	B594	B595	B596	B597	B598	B599	B600	B601	B602	B603	B604	B605	B606	B607	B608	B609	B610	B611	B612	B613	B614	B615	B616	B617	B618	B619	B620	B621	B622	B623	B624	B625	B626	B627	B628	B629	B630	B631	B632	B633	B634	B635	B636	B637	B638	B639	B640	B641	B642	B643	B644	B645	B646	B647	B648	B649	B650	B651	B652	B653	B654	B655	B656	B657	B658	B659	B660	B661	B662	B663	B664	B665	B666	B667	B668	B669	B66A	B66B	B66C	B66D	B66E	B66F	B66G	B66H	B66I	B66J	B66K	B66L	B66M	B66N	B66O	B66P	B66Q	B66R	B66S	B66T	B66U	B66V	B66W	B66X	B66Y	B66Z	B66A1	B66B1	B66C1	B66D1	B66E1	B66F1	B66G1	B66H1	B66I1	B66J1	B66K1	B66L1	B66M1	B66N1	B66O1	B66P1	B66Q1	B66R1	B66S1	B66T1	B66U1	B66V1	B66W1	B66X1	B66Y1	B66Z1	B66A2	B66B2	B66C2	B66D2	B66E2	B66F2	B66G2	B66H2	B66I2	B66J2	B66K2	B66L2	B66M2	B66N2	B66O2	B66P2	B66Q2	B66R2	B66S2	B66T2	B66U2	B66V2	B66W2	B66X2	B66Y2	B66Z2	B66A3	B66B3	B66C3	B66D3	B66E3	B66F3	B66G3	B66H3	B66I3	B66J3	B66K3	B66L3	B66M3	B66N3	B66O3	B66P3	B66Q3	B66R3	B66S3	B66T3	B66U3	B66V3	B66W3	B66X3	B66Y3	B66Z3	B66A4	B66B4	B66C4	B66D4	B66E4	B66F4	B66G4	B66H4	B66I4	B66J4	B66K4	B66L4	B66M4	B66N4	B66O4	B66P4	B66Q4	B66R4	B66S4	B66T4	B66U4	B66V4	B66W4	B66X4	B66Y4	B66Z4	B66A5	B66B5	B66C5	B66D5	B66E5	B66F5	B66G5	B66H5	B66I5	B66J5	B66K5	B66L5	B66M5	B66N5	B66O5	B66P5	B66Q5	B66R5	B66S5	B66T5	B66U5	B66V5	B66W5	B66X5	B66Y5	B66Z5	B66A6	B66B6	B66C6	B66D6	B66E6	B66F6	B66G6	B66H6	B66I6	B66J6	B66K6	B66L6	B66M6	B66N6	B66O6	B66P6	B66Q6	B66R6	B66S6	B66T6	B66U6	B66V6	B66W6	B66X6	B66Y6	B66Z6	B66A7	B66B7	B66C7	B66D7	B66E7	B66F7	B66G7	B66H7	B66I7	B66J7	B66K7	B66L7	B66M7	B66N7	B66O7	B66P7	B66Q7	B66R7	B66S7	B66T7	B66U7	B66V7	B66W7	B66X7	B66Y7	B66Z7	B66A8	B66B8	B66C8	B66D8	B66E8	B66F8	B66G8	B66H8	B66I8	B66J8	B66K8	B66L8	B66M8	B66N8	B66O8	B66P8	B66Q8	B66R8	B66S8	B66T8	B66U8	B66V8	B66W8	B66X8	B66Y8	B66Z8	B66A9	B66B9	B66C9	B66D9	B66E9	B66F9	B66G9	B66H9	B66I9	B66J9	B66K9	B66L9	B66M9	B66N9	B66O9	B66P9	B66Q9	B66R9	B66S9	B66T9	B66U9	B66V9	B66W9	B66X9	B66Y9	B66Z9	B66A10	B66B10	B66C10	B66D10	B66E10	B66F10	B66G10	B66H10	B66I10	B66J10	B66K10	B66L10	B66M10	B66N10	B66O10	B66P10	B66Q10	B66R10	B66S10	B66T10	B66U10	B66V10	B66W10	B66X10	B66Y10	B66Z10	B66A11	B66B11	B66C11	B66D11	B66E11	B66F11	B66G11	B66H11	B66I11	B66J11	B66K11	B66L11	B66M11	B66N11	B66O11	B66P11	B66Q11	B66R11	B66S11	B66T11	B66U11	B66V11	B66W11	B66X11	B66Y11	B66Z11	B66A12	B66B12	B66C12	B66D12	B66E12	B66F12	B66G12	B66H12	B66I12	B66J12	B66K12	B66L12	B66M12	B66N12	B66O12	B66P12	B66Q12	B66R12	B66S12	B66T12	B66U12	B66V12	B66W12	B66X12	B66Y12	B66Z12	B66A13	B66B13	B66C13	B66D13	B66E13	B66F13	B66G13	B66H13	B66I13	B66J13	B66K13	B66L13	B66M13	B66N13	B66O13	B66P13	B66Q13	B66R13	B66S13	B66T13	B66U13	B66V13	B66W13	B66X13	B66Y13	B66Z13	B66A14	B66B14	B66C14	B66D14	B66E14	B66F14	B66G14	B66H14	B66I14	B66J14	B66K14	B66L14	B66M14	B66N14	B66O14	B66P14	B66Q14	B66R14	B66S14	B66T14	B66U14	B66V14	B66W14	B66X14	B66Y14	B66Z14	B66A15	B66B15	B66C15	B66D15	B66E15	B66F15	B66G15	B66H15	B66I15	B66J15	B66K15	B66L15	B66M15	B66N15	B66O15	B66P15	B66Q15	B66R15	B66S15	B66T15	B66U15	B66V15	B66W15	B66X15	B66Y15	B66Z15	B66A16	B66B16	B66C16	B66D16	B66E16	B66F16	B66G16	B66H16	B66I16	B66J16	B66K16	B66L16	B66M16	B66N16	B66O16	B66P16	B66Q16	B66R16	B66S16	B66T16	B66U16	B66V16	B66W16	B66X16	B66Y16	B66Z16	B66A17	B66B17	B66C17	B66D17	B66E17	B66F17	B66G17	B66H17	B66I17	B66J17	B66K17	B66L17	B66M17	B66N17	B66O17	B66P17	B66Q17	B66R17	B66S17	B66T17	B66U17	B66V17	B66W17	B66X17	B66Y17	B66Z17	B66A18	B66B18	B66C18	B66D18	B66E18	B66F18	B66G18	B66H18	B66I18	B66J18	B66K18	B66L18	B66M18	B66N18	B66O18	B66P18	B66Q18	B66R18	B66S18	B66T18	B66U18	B66V18	B66W18	B66X18	B66Y18	B66Z18	B66A19	B66B19	B66C19	B66D19	B66E19	B66F19	B66G19	B66H19	B66I19	B66J19	B66K19	B66L19	B66M19	B66N19	B66O19	B66P19	B66Q19	B66R19	B66S19	B66T19	B66U19	B66V19	B66W19	B66X19	B66Y19	B66Z19	B66A20	B66B20	B66C20	B66D20	B66E20	B66F20	B66G20	B66H20	B66I20	B66J20	B66K20	B66L20	B66M20	B66N20	B66O20	B66P20	B66Q20	B66R20	B66S20	B66T20	B66U20	B66V20	B66W20	B66X20	B66Y20	B66Z20	B66A21	B66B21	B66C21	B66D21	B66E21	B66F21	B66G21	B66H21	B66I21	B66J21	B66K21	B66L21	B66M21	B66N21	B66O21	B66P21	B66Q21	B66R21	B66S21	B66T21	B66U21	B66V21	B66W21	B66X21	B66Y21	B66Z21	B66A22	B66B22	B66C22	B66D22	B66E22	B66F22	B66G22	B66H22	B66I22	B66J22	B66K22	B66L22	B66M22	B66N22	B66O22	B66P22	B66Q22	B66R22	B66S22	B66T22	B66U22	B66V22	B66W22	B66X22	B66Y22	B66Z22	B66A23	B66B23	B66C23	B66D23	B66E23	B66F23	B66G23	B66H23	B66I23	B66J23	B66K23	B66L23	B66M23	B66N23	B66O23	B66P23	B66Q23	B66R23	B66S23	B66T23	B66U23	B66V23	B66W23	B66X23	B66Y23	B66Z23	B66A24	B66B24	B66C24	B66D24	B66E24	B66F24	B66G24	B66H24	B66I24	B66J24	B66K24	B66L24	B66M24	B66N24	B66O24	B66P24	B66Q24	B66R24	B66S24	B66T24	B66U24	B66V24	B66W24	B66X24	B66Y24	B66Z24	B66A25	B66B25	B66C25	B66D25	B66E25	B66F25	B66G25	B66H25	B66I25	B66J25	B66K25	B66L25	B66M25	B66N25	B66O25	B66P25	B66Q25	B66R25	B66S25	B66T25	B66U25	B66V25	B66W25	B66X25	B66Y25	B66Z25	B66A26	B66B26	B66C26	B66D26	B66E26	B66F26	B66G26	B66H26	B66I26	B66J26	B66K26	B66L26	B66M26	B66N26	B66O26	B66P26	B66Q26	B66R26	B66S26	B66T26	B66U26	B66V26	B66W26	B66X26	B66Y26	B66Z26	B66A27	B66B27	B66C27	B66D27	B66E27	B66F27	B66G27	B66H27	B66I

LUA autoclicker

Code complex scripts with LUA

- Complete CE API
- Timers, memory scan,
AutoAssembly integration
- GUI development

```
{$lua}
if syntaxcheck then return end
[ENABLE]
```

```
l2sh_counter = 0
l2sh_timer = createTimer(nil)

speedhack_setSpeed(500)
l2sh_timer.Enabled = false
l2sh_timer.Interval = 100 -- spam keys
every 0.1s
```

```
function l2SH_loop(v_timer)
    if ((l2sh_counter == -1) or
        (l2sh_counter == 750)) then
        l2sh_timer.Enabled = false
        -- stop after 75s or manual interrupt
        speedhack_setSpeed(1)
        return
    end

    if (getForegroundWindow() ~=
findWindow(null, "flag quest")) then
        return -- do not keypresses when
game window is not focused
    end
```

LUA autoclicker

Code complex scripts with LUA

- Complete CE API
- Timers, memory scan,
AutoAssembly integration
- GUI development

```
function l2SH_loop(v_timer)
    if ((l2sh_counter == -1) or
        (l2sh_counter == 750)) then
        l2sh_timer.Enabled = false
        -- stop after 75s or manual interrupt
        speedhack_setSpeed(1)
        return
    end

    if (getForegroundWindow() ~=
        findWindow(null, "flag quest")) then
        return -- do not keypresses when
        game window is not focused
    end

    if ((l2sh_counter % 10) == 0) then
        doKeyPress(VK_SPACE)
        -- 1/10 space keypress
    else
        doKeyPress(VK_RIGHT)
    end
    l2sh_counter = l2sh_counter + 1
end

-- put this AFTER method definition
l2sh_timer.OnTimer = l2SH_loop
```

LUA autoclicker

Code complex scripts with LUA

- Complete CE API
- Timers, memory scan,
AutoAssembly integration
- GUI development

```
speedhack_setspeed(1)
return
end

if (getForegroundWindow() ~= findWindow(null, "flag quest")) then
    return -- do not keypresses when game window is not focused
end

if ((l2sh_counter % 10) == 0) then
    doKeyPress(VK_SPACE)
    -- 1/10 space keypress
else
    doKeyPress(VK_RIGHT)
end
l2sh_counter = l2sh_counter + 1
end

-- put this AFTER method definition
l2sh_timer.OnTimer = l2SH_loop
l2sh_timer.Enabled = true

[DISABLE]
speedhack_setspeed(1)
l2sh_timer.destroy() --important!
```

Nice CheatEngine features

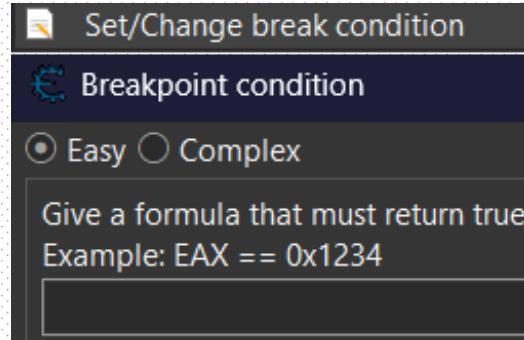
- Debugger: ultra powerful

stack trace

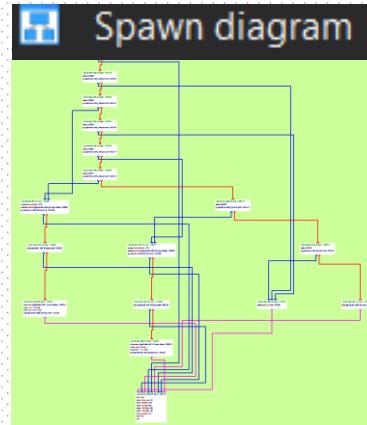
The screenshot shows the 'Tracer' tab in CheatEngine. At the top, there's a toolbar with icons for Break and Trace, Stop, and others. Below the toolbar is a menu bar with File and Search. The main area displays a stack trace for the process 'gtutorial-x86_64.exe'. The trace shows the following sequence of instructions:

```
gtutorial-x86_64.exe.text+4083C - mov rcx,[rbx+38]
gtutorial-x86_64.exe.text+40840 - call gtutorial-x86_64.exe.text+41B50
gtutorial-x86_64.exe.text+40845 - test al,al
gtutorial-x86_64.exe.text+40847 - je gtutorial-x86_64.exe.text+40898
gtutorial-x86_64.exe.text+40898 - mov rdx,[rbx+40]
gtutorial-x86_64.exe.text+4089C - movsxd rax,r12d
```

conditional breakpoints

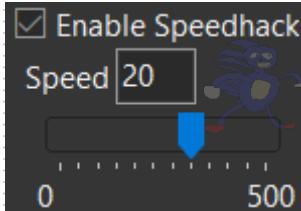


Flow charts

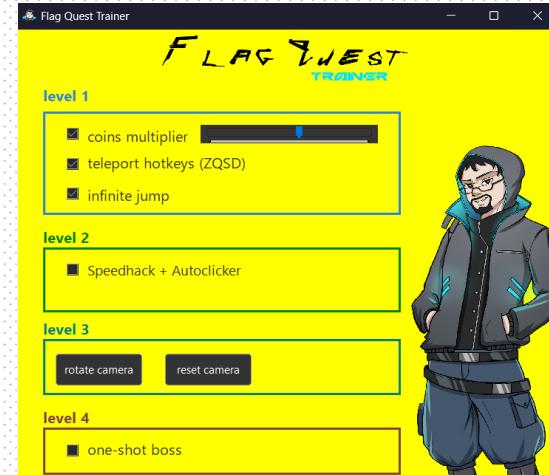


Nice and useless CheatEngine features

- Speedhack: it just works



- GUI generator: very powerful
but long topic



- Unrandomizer: never works



- Mono Features: better use Unity Explorer directly

The best

CheatEngine feature

- Luacode: put LUA inside your ASM
- Write complex code inside of ASM patch
- Have full LUA API access
- Rename register as variables

code:

```
cmp r9,rax  
jne end
```

```
{$LUACODE pos=RBX}
```

```
local npos = pos+0x424 + 8 -- Z pos addr  
if npos ~= getAddressSafe("lev3_z") then  
    print("Updating all CE 3D position vars")
```

```
    local names = {"z", "y", "x", "sz2",  
"sz1", "sz0", "sy2", "sy1", "sy0", "sx2",  
"sx1", "sx0"}
```

```
    for _, name in ipairs(names) do  
        registerSymbol("lev3_"..name, npos)  
        -- going backward in memory  
        npos = npos - 4  
    end
```

```
end
```

```
{$ASM}
```

end:

```
    movups [rbx+00000420],xmm2 // original  
instruction for setting sz2  
    jmp return
```

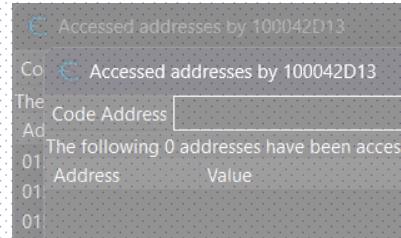
The best worst CheatEngine feature

- Luacode: put LUA inside your ASM
- Bugged at the moment
 - Need to add a space after your comments lines at the start of your cheat
- Not optimized
- May crash your game
- Consider `{$CCODE}`
write C inside your ASM



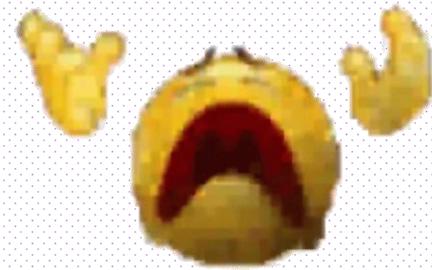
Common Cheat Engine problems 1

- I changed a value and my game crashed!
-> *welcome to CE. It will do that.*
- My “find what writes/access” address search has no result
-> *you already have a scan running, but the window is hidden behind another.*
- I added My AutoAssembly cheat to the codelist, but I can't enable it
-> *you clicked on Execute and it already patched the game. Relaunch the game.*



Common Cheat Engine problems 2

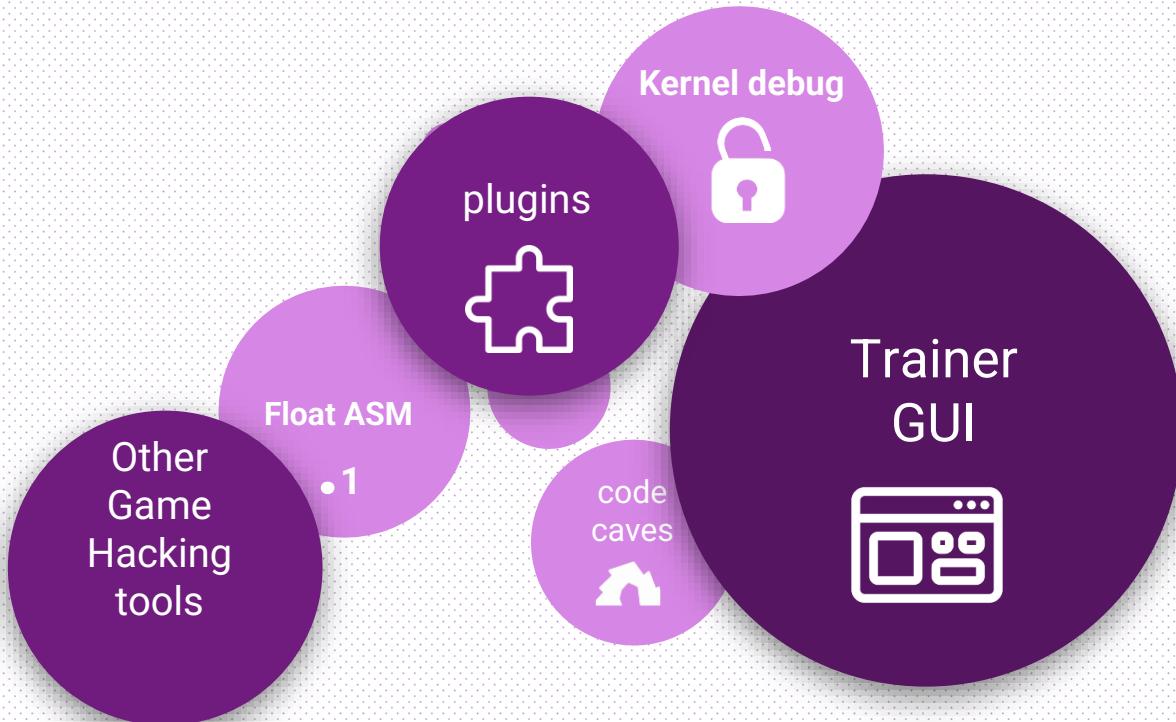
- I did a scan, saved my results in a file.
Redid a scan, saved to the same file.
Now the scan doesn't work.
-> *Just don't do that*

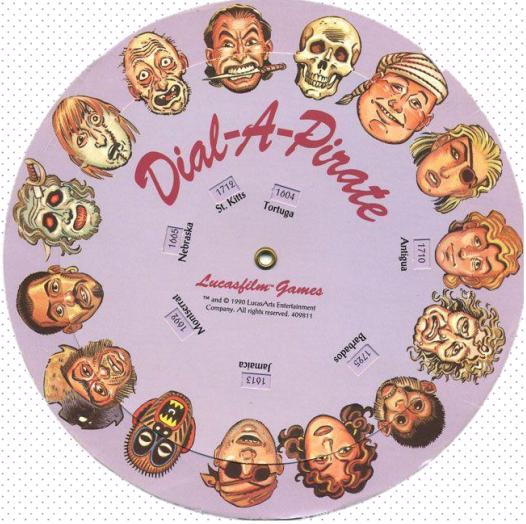


- My LUA script runs twice!
-> *you forgot to add if syntaxcheck then return end*

I launched a LUA script that never terminates,
modified it and restarted it. Now I have bugs.
-> *Relaunch Cheat Engine :/*

Next steps



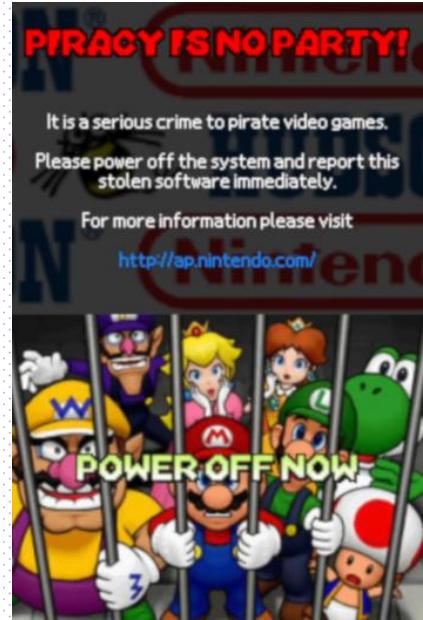


tuxlu.fr/talk_vghacking





tuxlu.fr/talk_vghacking





thank you.

