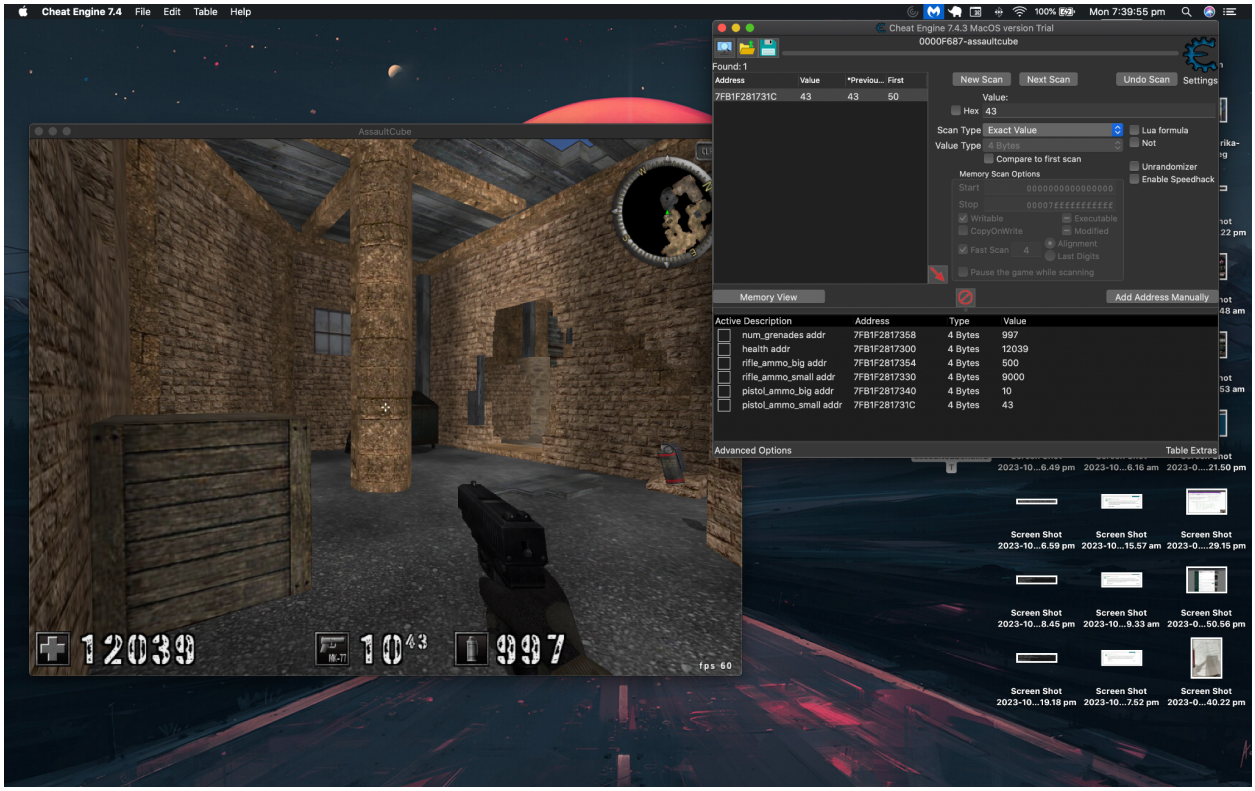
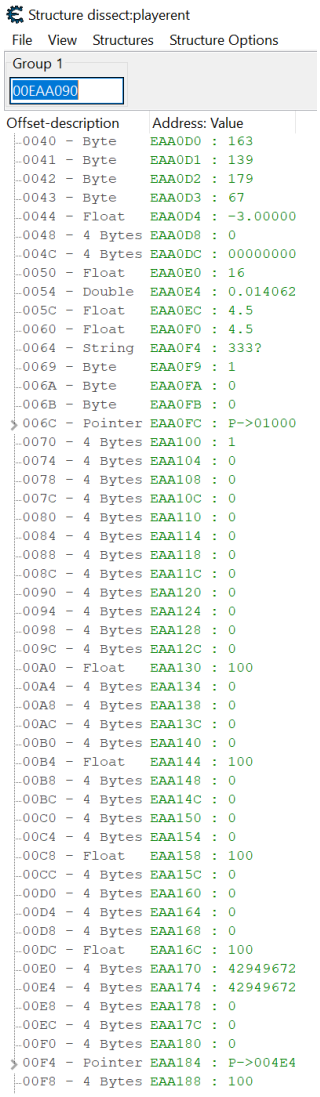
First Cheat Engine Table with Dynamic Addresses



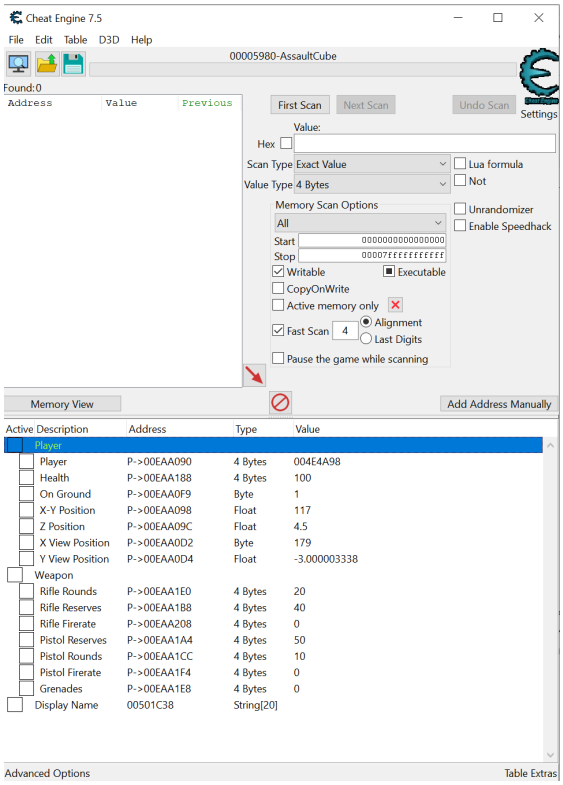
Pointer scans for various addresses:

1.PTR.results.5	10/10/2023 3:46 PM	5 File	119 KB
1.PTR.results.6	10/10/2023 3:46 PM	6 File	99 KB
1.PTR.results.7	10/10/2023 3:46 PM	7 File	116 KB
1.PTR.results.8	10/10/2023 3:46 PM	8 File	123 KB
2.PTR	10/10/2023 3:50 PM	PTR File	3 KB
2.PTR.results.0	10/10/2023 3:50 PM	zip	2 KB
2.PTR.results.1	10/10/2023 3:50 PM	1 File	2 KB
2.PTR.results.2	10/10/2023 3:50 PM	2 File	2 KB
2.PTR.results.3	10/10/2023 3:50 PM	3 File	2 KB
2.PTR.results.4	10/10/2023 3:50 PM	4 File	1 KB
2.PTR.results.5	10/10/2023 3:50 PM	5 File	1 KB
2.PTR.results.6	10/10/2023 3:50 PM	6 File	2 KB
2.PTR.results.7	10/10/2023 3:50 PM	7 File	2 KB
2.PTR.results.8	10/10/2023 3:50 PM	8 File	5 KB
armourptr.PTR	10/10/2023 4:23 PM	PTR File	3 KB
armourptr.PTR.results.0	10/10/2023 4:23 PM	zip	129 KB
armourptr.PTR.results.1	10/10/2023 4:23 PM	1 File	98 KB
armourptr.PTR.results.2	10/10/2023 4:23 PM	2 File	93 KB
armourptr.PTR.results.3	10/10/2023 4:23 PM	3 File	120 KB
armourptr.PTR.results.4	10/10/2023 4:23 PM	4 File	148 KB
armourptr.PTR.results.5	10/10/2023 4:23 PM	5 File	96 KB
armourptr.PTR.results.6	10/10/2023 4:23 PM	6 File	107 KB
armourptr.PTR.results.7	10/10/2023 4:23 PM	7 File	80 KB
armourptr.PTR.results.8	10/10/2023 4:23 PM	8 File	72 KB
armourptr2.PTR	10/10/2023 4:25 PM	PTR File	3 KB
armourptr2.PTR.results.0	10/10/2023 4:25 PM	zip	2 KB
armourptr2.PTR.results.1	10/10/2023 4:25 PM	1 File	3 KB
armourptr2.PTR.results.2	10/10/2023 4:25 PM	2 File	5 KB
armourptr2.PTR.results.3	10/10/2023 4:25 PM	3 File	3 KB
armourptr2.PTR.results.4	10/10/2023 4:25 PM	4 File	3 KB
armourptr2.PTR.results.5	10/10/2023 4:25 PM	5 File	3 KB
armourptr2.PTR.results.6	10/10/2023 4:25 PM	6 File	2 KB
armourptr2.PTR.results.7	10/10/2023 4:25 PM	7 File	4 KB
armourptr2.PTR.results.8	10/10/2023 4:25 PM	8 File	2 KB
grav1.PTR	17/10/2023 6:08 PM	PTR File	3 KB
grav1.PTR.results.0	17/10/2023 6:08 PM	zip	5,060 KB
grav1.PTR.results.1	17/10/2023 6:08 PM	1 File	5,483 KB
grav1.PTR.results.2	17/10/2023 6:08 PM	2 File	3,839 KB
grav1.PTR.results.3	17/10/2023 6:08 PM	3 File	3,040 KB
grav1.PTR.results.4	17/10/2023 6:08 PM	4 File	2,717 KB

Player entity struct that contains all the offsets and their corresponding values. These are guessed by Cheat Engine so not 100% reliable but still quite reliable.



Cheat table with base pointers before moving onto writing scripts.



Script for finding the list of entities in preparation for AssaultCube triggerbot.

```python
import pymem

ENTITY_LIST = 0x10f4f8
LOCAL_PLAYER = 0x509b74
HEALTH_OFFSET = 0xF8
CLIENT = 0x400000
ENTITY_SIZE = 0x4

def main():
    print("Trigger has been launched")
    pm = pymem.Pymem("ac_client.exe")

    # entity list base addr
    entity_list_base = pm.read_int(CLIENT + ENTITY_LIST)
    print(f"Entity list base address: {hex(entity_list_base)}")

    # 16 for testing rn. need to find way to get entity count
    # skip first range since always empty
    for i in range(1, 16):
        # calculate the address of the entity
        entity_address = pm.read_int(entity_list_base + i * ENTITY_SIZE)
        health = pm.read_int(entity_address + HEALTH_OFFSET)
        print("entity address: ", entity_address)
        # skip invalid addresses
        if health == 0 or health == None:
            continue

        # read the health value of the entity
        health = pm.read_int(entity_address + HEALTH_OFFSET)
        print(f"Entity {i} health: {health}")

if __name__ == "__main__":
    main()
```

```
OUTPUT DEBUG CONSOLE TERMINAL PORTS
[Running] python -u "c:\Users\bryan\localcompscimage\python\ACtriggerbot.py"
Trigger has been launched
Entity list base address: 0x2b42280
entity address:  227319832
Entity 1 health: 100
entity address:  45587744
Entity 2 health: 100
entity address:  45568880
Entity 3 health: 100
entity address:  227052216
Entity 4 health: 100
entity address:  45413568
```

Simple script for bunnyhopping.

python > bhop.py > bhop

```python
import pymem, keyboard, time
import pymem.process
import win32api

LOCAL_PLAYER = 14596508
FORCE_JUMP = 86756744
HEALTH = 256
FLAGS = 260

def bhop() -> None:
    print('Bhop has launched.')
    pm = pymem.Pymem('csgo.exe')

    # module address
    client = pymem.process.module_from_name(pm.process_handle, "client.dll").lpBaseOfDll

    # hack loop
    while True:
        time.sleep(0.01)

        # check space bar is not pressed
        if not win32api.GetAsyncKeyState(0x20):
            continue

        local_player: int = pm.read_uint(client + LOCAL_PLAYER)

        # check valid player
        if not local_player:
            continue

        # is alive
        if not pm.read_int(local_player + HEALTH):
            continue

        if pm.read_uint(local_player + FLAGS) & 1 << 0:
            pm.write_uint(client + FORCE_JUMP, 6)
            time.sleep(0.01)
            pm.write_uint(client + FORCE_JUMP, 4)

if __name__ == '__main__':
    bhop()
```

CSGO script for triggerbot

python > trigger.py > ...

```python
import keyboard
import pymem
import pymem.process
import time

dwEntityList = (81793068)
dwForceAttack = (52620952)
dwLocalPlayer = (14596508)
m_iCrosshairId = (71736)
m_iTeamNum = (244)

def main():
    print("Trigger has launched.")
    pm = pymem.Pymem("csgo.exe")
    client = pymem.process.module_from_name(pm.process_handle, "client.dll").lpBaseOfDll

    while True:
        localPlayer = pm.read_int(client + dwLocalPlayer)
        crosshairID = pm.read_int(localPlayer + m_iCrosshairId)
        getTeam = pm.read_int(client + dwEntityList + (crosshairID - 1) * 0x10)
        localTeam = pm.read_int(localPlayer + m_iTeamNum)
        crosshairTeam = pm.read_int(getTeam + m_iTeamNum)

        if crosshairID > 0 and crosshairID < 32 and localTeam != crosshairTeam:
            pm.write_int(client + dwForceAttack, 6)

        if keyboard.is_pressed('x'):
            time.sleep(2)

        time.sleep(0.1)

if __name__ == '__main__':
    main()
```

Script for finding updated offsets for CSGO from [hazedumper on Github](https://github.com/frk1/hazedumper/blob/master/csgo.json).

python > update_offsets.py > ...

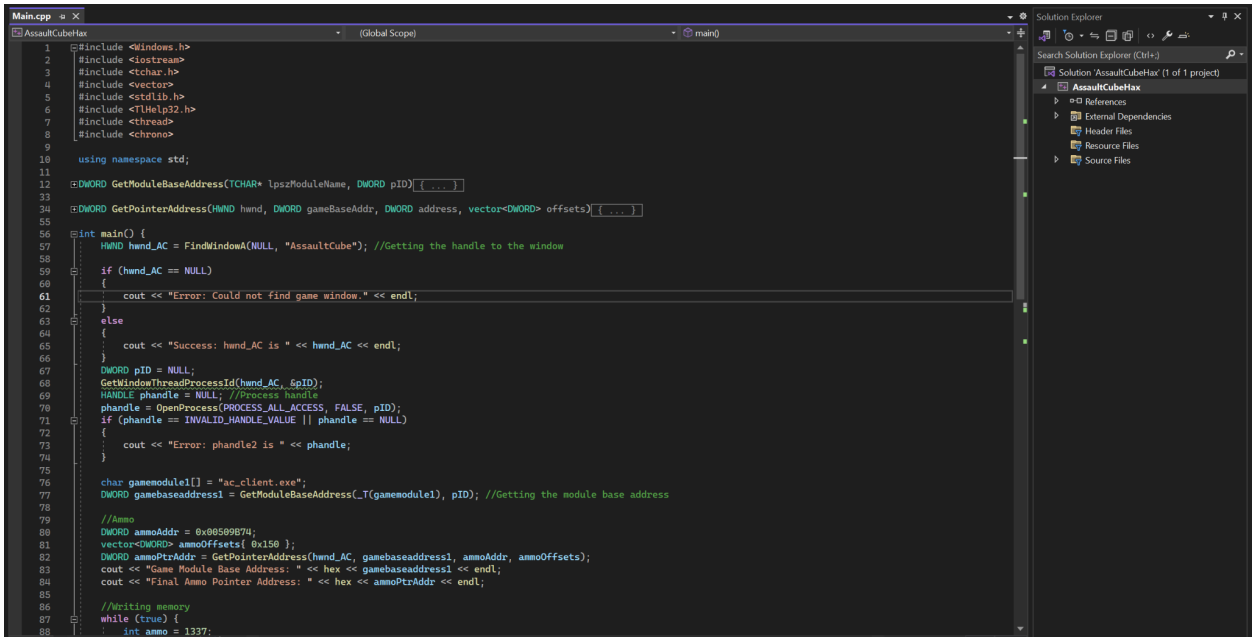
```python
import requests
import json

def write_items(dictionary, file):
    for k, v in dictionary.items():
        sigs = f"{k}: {v}"
        file.write(sigs)
        file.write('\n')

def update():
    r = requests.get("https://github.com/frk1/hazedumper/blob/master/csgo.json").json()
    rawLines = r["payload"]["blob"]["rawLines"]
    data_dict = json.loads("".join(rawLines))
    signatures = data_dict['signatures']
    netvars = data_dict['netvars']
    with open("offsets.txt", "w") as f:
        write_items(signatures, f)
        write_items(netvars, f)
        f.close()

if __name__ == "__main__":
    update()
```

Result of previous script into a file called offsets.txt. Contains updated offsets for CSGO.

python > offsets.txt

```
anim_overlays: 10640
clientstate_choked_commands: 19760
clientstate_delta_ticks: 372
clientstate_last_outgoing_command: 19756
clientstate_net_channel: 156
convar_name_hash_table: 197024
dwClientState: 5894556
dwClientState_GetLocalPlayer: 384
dwClientState_IsHLTV: 19784
dwClientState_Map: 652
dwClientState_MapDirectory: 392
dwClientState_MaxPlayer: 904
dwClientState_PlayerInfo: 21184
dwClientState_State: 264
dwClientState_ViewAngles: 19856
dwEntityList: 81793068
dwForceAttack: 52620952
dwForceAttack2: 52620964
dwForceBackward: 52621024
dwForceForward: 52621012
dwForceJump: 86756744
dwForceLeft: 52620880
dwForceRight: 52620892
dwGameDir: 6532608
dwGameRulesProxy: 87229844
dwGetAllClasses: 14733220
dwGlobalVars: 5893728
dwGlowObjectManager: 87407312
dwInput: 86369792
dwInterfaceLinkList: 10080132
dwLocalPlayer: 14596508
dwMouseEnable: 86221408
dwMouseEnablePtr: 86221360
dwPlayerResource: 52613584
dwRadarBase: 86211332
```

First attempt at c++ script for infinite ammo hack.




```cpp
#include <Windows.h>
#include <iostream>
#include <tchar.h>
#include <vector>
#include <stdlib.h>
#include <TlHelp32.h>
#include <thread>
#include <chrono>

using namespace std;

DWORD GetModuleBaseAddress(TCHAR* lpszModuleName, DWORD pID) { ... }

DWORD GetPointerAddress(HWND hwnd, DWORD gameBaseAddr, DWORD address, vector<DWORD> offsets) { ... }

int main() {
    HWND hwnd_AC = FindWindowA(NULL, "AssaultCube"); //Getting the handle to the window

    if (hwnd_AC == NULL)
    {
        cout << "Error: Could not find game window." << endl;
    }
    else
    {
        cout << "Success: hwnd_AC is " << hwnd_AC << endl;
    }
    DWORD pID = NULL;
    GetWindowThreadProcessId(hwnd_AC, &pID);
    HANDLE phandle = NULL; //Process handle
    phandle = OpenProcess(PROCESS_ALL_ACCESS, FALSE, pID);
    if (phandle == INVALID_HANDLE_VALUE || phandle == NULL)
    {
        cout << "Error: phandle2 is " << phandle;
    }

    char gamemodule1[] = "ac_client.exe";
    DWORD gamebaseaddress1 = GetModuleBaseAddress(_T(gamemodule1), pID); //Getting the module base address

    //Ammo
    DWORD ammoAddr = 0x00509B74;
    vector<DWORD> ammoOffsets{ 0x150 };
    DWORD ammoPtrAddr = GetPointerAddress(hwnd_AC, gamebaseaddress1, ammoAddr, ammoOffsets);
    cout << "Game Module Base Address: " << hex << gamebaseaddress1 << endl;
    cout << "Final Ammo Pointer Address: " << hex << ammoPtrAddr << endl;

    //Writing memory
    while (true) {
        int ammo = 1337;
```

```cpp
    //Writing memory
    while (true) {
        int ammo = 1337;
        int currentAmmo;
        if (ReadProcessMemory(phandle, (LPVOID*)ammoPtrAddr, &currentAmmo, sizeof(currentAmmo), 0)) {
            cout << "Current Ammo: " << currentAmmo << endl;
        }
        else {
            cout << "Failed to read current ammo value!" << endl;
        }
        WriteProcessMemory(phandle, (LPVOID*)(ammoPtrAddr), &ammo, 4, 0);
        if (!WriteProcessMemory(phandle, (LPVOID*)(ammoPtrAddr), &ammo, 4, 0)) {
            cout << "Failed to write to memory!" << endl;
        }
        this_thread::sleep_for(chrono::milliseconds(100));
    }
}
```

Simple infinite health and ammo hack for AssaultCube

python > ACstatshack.py > ...

```python
import pymem
import time

def infinitehealthammo():
    pm = pymem.Pymem('ac_client.exe')

    LOCAL_PLAYER = 0x00509b74
    HEALTH = 0xF8
    AMMO = 0x150

    local_player = pm.read_int(LOCAL_PLAYER)

    while True:
        time.sleep(0.01)
        health_address = local_player + HEALTH
        pm.write_int(health_address, 420)

        ammo_address = local_player + AMMO
        pm.write_int(ammo_address, 420)

if __name__ == "__main__":
    infinitehealthammo()
```

Simple bhop script for AssaultCube

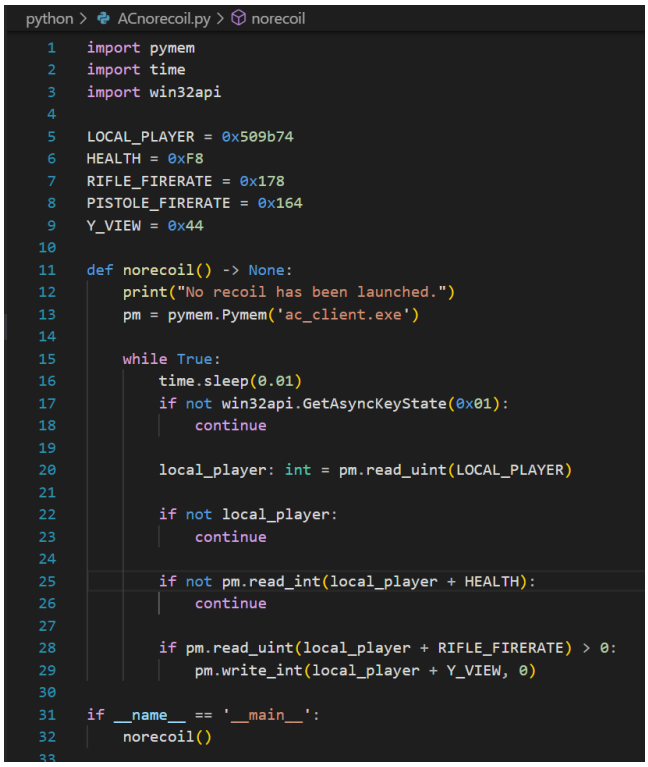
python > ACbhop.py > ...

```python
import pymem, keyboard, time
import win32api

LOCAL_PLAYER = 0x509b74
FORCE_JUMP = 0x6B
HEALTH = 0xF8
ON_GROUND = 0x69

def bhop() -> None:
    print('Bhop has launched.')
    pm = pymem.Pymem('ac_client.exe')

    # hack loop
    while True:
        time.sleep(0.01)

        # space bar
        if not win32api.GetAsyncKeyState(0x20):
            continue

        local_player: int = pm.read_uint(LOCAL_PLAYER)

        if not local_player:
            continue

        # is alive
        if not pm.read_int(local_player + HEALTH):
            continue

        # on ground
        if (pm.read_uint(local_player + ON_GROUND) == 1):
            pm.write_uint(local_player + FORCE_JUMP, 1)


if __name__ == '__main__':
    bhop()
```

Simple rapid fire script for AssaultCube

```python
import pymem, keyboard, time
import win32api

LOCAL_PLAYER = 0x509b74
HEALTH = 0xF8
RIFLE_FIRERATE = 0x178
PISTOLE_FIRERATE = 0x164

def rapidfire() -> None:
    print("Rapid fire has been launched.")
    pm = pymem.Pymem('ac_client.exe')

    while True:
        time.sleep(0.01)
        # left-click
        if not win32api.GetAsyncKeyState(0x01):
            continue

        local_player: int = pm.read_uint(LOCAL_PLAYER)

        if not local_player:
            continue

        # is alive
        if not pm.read_int(local_player + HEALTH):
            continue

        if pm.read_uint(local_player + RIFLE_FIRERATE) == 120:
            pm.write_uint(local_player + RIFLE_FIRERATE, 60)

        # Pistol kinda useless though
        if pm.read_uint(local_player + PISTOLE_FIRERATE) == 160:
            pm.write_uint(local_player + RIFLE_FIRERATE, 60)

if __name__ == '__main__':
    rapidfire()
```

Simple no recoil script for AssaultCube before fix. Issue was that while shooting, couldn't look up or down.



```python
import pymem
import time
import win32api

LOCAL_PLAYER = 0x509b74
HEALTH = 0xF8
RIFLE_FIRERATE = 0x178
PISTOLE_FIRERATE = 0x164
Y_VIEW = 0x44

def norecoil() -> None:
    print("No recoil has been launched.")
    pm = pymem.Pymem('ac_client.exe')

    while True:
        time.sleep(0.01)
        if not win32api.GetAsyncKeyState(0x01):
            continue

        local_player: int = pm.read_uint(LOCAL_PLAYER)

        if not local_player:
            continue

        if not pm.read_int(local_player + HEALTH):
            continue

        if pm.read_uint(local_player + RIFLE_FIRERATE) > 0:
            pm.write_int(local_player + Y_VIEW, 0)

if __name__ == '__main__':
    norecoil()
```

Simple no recoil script for AssaultCube after fix. Realised that recoil was only going up so made the player look downwards and wrote value to Y_VIEW offset

python > ACnorecoil.py > norecoil

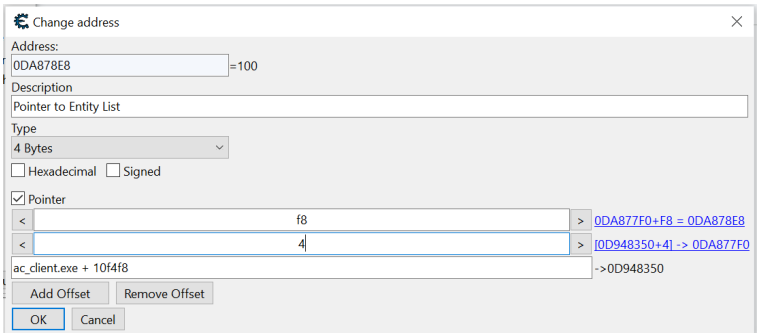
```python
import pymem
import time
import win32api

LOCAL_PLAYER = 0x509b74
HEALTH = 0xF8
RIFLE_FIRERATE = 0x178
PISTOLE_FIRERATE = 0x164
Y_VIEW = 0x44

def norecoil() -> None:
    print("No recoil has been launched.")
    pm = pymem.Pymem('ac_client.exe')

    while True:
        time.sleep(0.01)
        if not win32api.GetAsyncKeyState(0x01):
            continue

        local_player: int = pm.read_uint(LOCAL_PLAYER)

        if not local_player:
            continue

        if not pm.read_int(local_player + HEALTH):
            continue

        if pm.read_uint(local_player + RIFLE_FIRERATE) > 0:
            current_y = pm.read_float(local_player + Y_VIEW)
            recoil_amount = 0.28
            new_y = current_y - recoil_amount
            pm.write_float(local_player + Y_VIEW, new_y)

if __name__ == '__main__':
    norecoil()
```

Failed attempt at rifle jump script for AssaultCube

python > ACriflejump.py > {} pymem

```python
import pymem, keyboard, time
import win32api

LOCAL_PLAYER = 0x509b74
FORCE_JUMP = 0x6B
HEALTH = 0xF8
ON_GROUND = 0x69
FORCE_ATTACK = 0x224
Y_VIEW = 0x44

def bhop() -> None:
    print('Bhop has launched.')
    pm = pymem.Pymem('ac_client.exe')

    # hack loop
    while True:
        time.sleep(0.01)

        # space bar
        if not win32api.GetAsyncKeyState(0x20):
            continue

        local_player: int = pm.read_uint(LOCAL_PLAYER)

        if not local_player:
            continue

        # is alive
        if not pm.read_int(local_player + HEALTH):
            continue

        # on ground
        if win32api.GetAsyncKeyState(0x20) and win32api.GetAsyncKeyState(0x01):
            pm.write_uint(local_player + FORCE_JUMP, 6)
            pm.write_uint(local_player + Y_VIEW, -90)
            # pm.write_uint(local_player + FORCE_ATTACK, 1092831209)


if __name__ == '__main__':
    bhop()
```

Checking entity offset for entity list



First attempt at triggerbot script for AssaultCube.
Uses entity list acquired previously and accesses each entity in the list to get their names, health and team. Then compares the team of the entity to the team of the player and if they are not on the same team, the player shoots. Otherwise, the player stops shooting. Issue with the player constantly shooting after looking at the enemy and only stops shooting when looking at teammates.

python > ACtriggerbot.py > FORCE_ATTACK

```python
ENTITY_LIST = 0x10f4f8
LOCAL_PLAYER = 0x509b74
HEALTH_OFFSET = 0xF8
CLIENT = 0x400000
ENTITY_SIZE = 0x4
FORCE_ATTACK = 0x224
TEAM_OFFSET = 0x32C
NAME_OFFSET = 0x225
DISPLAY_NAME = 0x101C38


def get_character_entities(pm):
    # entity list base addr
    entity_list_base = pm.read_int(CLIENT + ENTITY_LIST)

    entities_data = {}
    # 16 for testing rn. need to find way to get entity count
    # skip first range since always empty
    for i in range(1, 16):
        try:
            entity_address = pm.read_int(entity_list_base + i * ENTITY_SIZE)
            health = pm.read_int(entity_address + HEALTH_OFFSET)
            team = pm.read_int(entity_address + TEAM_OFFSET)
            name = pm.read_string(entity_address + NAME_OFFSET)
            # skip invalid addresses or if health is zero
            if health == 0 or health == None:
                continue
            entity_data = {'name': name, 'health': health, 'team': team}
            entities_data[i] = entity_data
        except pymem.exception.MemoryReadError as e:
            continue
    return entities_data

def get_entity_by_name(entities_info):
    return {info['name']: entity_num for entity_num, info in entities_info.items() if info['name']}

def main():
    print("Trigger has been launched")
    pm = pymem.Pymem("ac_client.exe")
    entities = get_character_entities(pm)
    print(entities)
    # lookup dict for names to entity numbers
    entity_lookup_by_name = get_entity_by_name(entities)
    while True:
        time.sleep(0.01)
        local_player: int = pm.read_uint(LOCAL_PLAYER)
        display_name = CLIENT + DISPLAY_NAME
        current_name = pm.read_string(display_name)

        # check if current name matches any known entity's name
        if current_name in entity_lookup_by_name:
            entity_num = entity_lookup_by_name[current_name]
            entity_details = entities[entity_num]
            team = entity_details['team']
            if team == 0:
                pm.write_uint(local_player + FORCE_ATTACK, 1)
                # print(pm.write_bytes(LOCAL_PLAYER + FORCE_ATTACK, (1).to_bytes(1, byteorder='little'), 1))
                # print(pm.write_uint(LOCAL_PLAYER + FORCE_ATTACK, 1853441793))
            elif team == 1:
                pm.write_uint(local_player + FORCE_ATTACK, 0)
                continue

if __name__ == "__main__":
    main()
```

First fix for triggerbot.
Realised issue was with display_name storing name even though the player wasn't looking at the enemy anymore. Created prev_name to store the name from previous iteration to compare to current name to decide to shoot or not. If the names are the same, don't shoot, otherwise, shoot. Issue was only one bullet shot at the enemy when looking at them instead of multiple if constantly looking.

```
prev_name = None
can_shoot = False
while True:
    time.sleep(0.01)
    local_player: int = pm.read_uint(LOCAL_PLAYER)
    player_team = pm.read_int(LOCAL_PLAYER + TEAM_OFFSET)
    current_name = pm.read_string(CLIENT + DISPLAY_NAME)

    print(f"prev_name1: ", prev_name)
    print(f"current_name1: ", current_name)

    # check if current name matches any known entity's name
    if prev_name != current_name:
        prev_name = current_name
        if current_name in entity_lookup_by_name:
            entity_num = entity_lookup_by_name[current_name]
            entity_details = entities[entity_num]
            team = entity_details['team']
            if team != player_team:
                can_shoot = True
            else:
                can_shoot = False
        else:
            can_shoot = False
    else:
        can_shoot = False

    if can_shoot:
        pm.write_uint(local_player + FORCE_ATTACK, 1)
    else:
        pm.write_uint(local_player + FORCE_ATTACK, 0)

    print(f"prev_name2: ", prev_name)
    print(f"current_name2: ", current_name)
```

Second fix for triggerbot. Instead of keeping track of prev_name, I realised I could write null terminators to the display name address. This way, current_name still gets updated when looking at another character/entity but if not looking, current_name is just an empty string. Hence, there is constant shooting at enemy and looking away stops shooting.

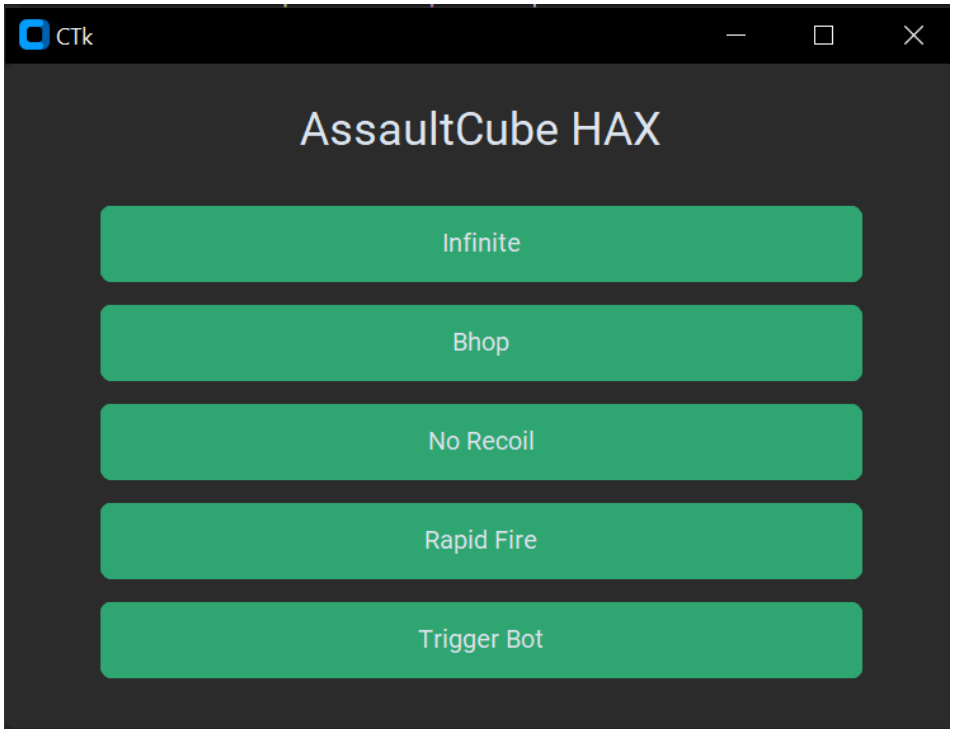
```python
def triggerbot():
    print("Trigger has been launched")
    pm = pymem.Pymem("ac_client.exe")
    entities = get_character_entities(pm)
    entity_lookup_by_name = get_entity_by_name(entities)
    player_team = get_player_team(entities)

    can_shoot = False
    while True:
        time.sleep(0.01)
        local_player: int = pm.read_uint(LOCAL_PLAYER)
        current_name = pm.read_string(CLIENT + DISPLAY_NAME)
        pm.write_string(CLIENT + DISPLAY_NAME, "\0" * 20)

        # check if current name matches any known entity's name
        if current_name in entity_lookup_by_name:
            entity_num = entity_lookup_by_name[current_name]
            entity_details = entities[entity_num]
            team = entity_details['team']
            if team != player_team:
                can_shoot = True
            else:
                can_shoot = False
        else:
            can_shoot = False

        if can_shoot:
            pm.write_uint(local_player + FORCE_ATTACK, 1)
        else:
            pm.write_uint(local_player + FORCE_ATTACK, 0)
```

Code for GUI mod menu for AssaultCube cheats.

python > ACmain.py > bhop

```python
import customtkinter
import tkinter
from ACinfinitehealthammo import infinitehealthammo
from ACbhop import bhop
from ACnorecoil import norecoil
from ACrapidfire import rapidfire
from ACtriggerbot import triggerbot

customtkinter.set_appearance_mode("dark")
customtkinter.set_default_color_theme("green")

menu = customtkinter.CTk()
menu.geometry("500x350")

frame = customtkinter.CTkFrame(master=menu)
frame.pack(fill="both", expand=True)

label = customtkinter.CTkLabel(master=frame, text="AssaultCube HAX", font=("Roboto", 24))
label.pack(pady=20, padx=60)

infinitebutton = customtkinter.CTkButton(master=frame, text="Infinite", command=infinitehealthammo, width=400, height=40)
infinitebutton.pack(pady=6, padx=8)

bhopbutton = customtkinter.CTkButton(master=frame, text="Bhop", command=bhop, width=400, height=40)
bhopbutton.pack(pady=6, padx=8)

norecoilbutton = customtkinter.CTkButton(master=frame, text="No Recoil", command=norecoil, width=400, height=40)
norecoilbutton.pack(pady=6, padx=8)

rapidfirebutton = customtkinter.CTkButton(master=frame, text="Rapid Fire", command=rapidfire, width=400, height=40)
rapidfirebutton.pack(pady=6, padx=8)

triggerbotbutton = customtkinter.CTkButton(master=frame, text="Trigger Bot", command=triggerbot, width=400, height=40)
triggerbotbutton.pack(pady=6, padx=8)

menu.mainloop()
```

Mod menu GUI for AssaultCube Cheats

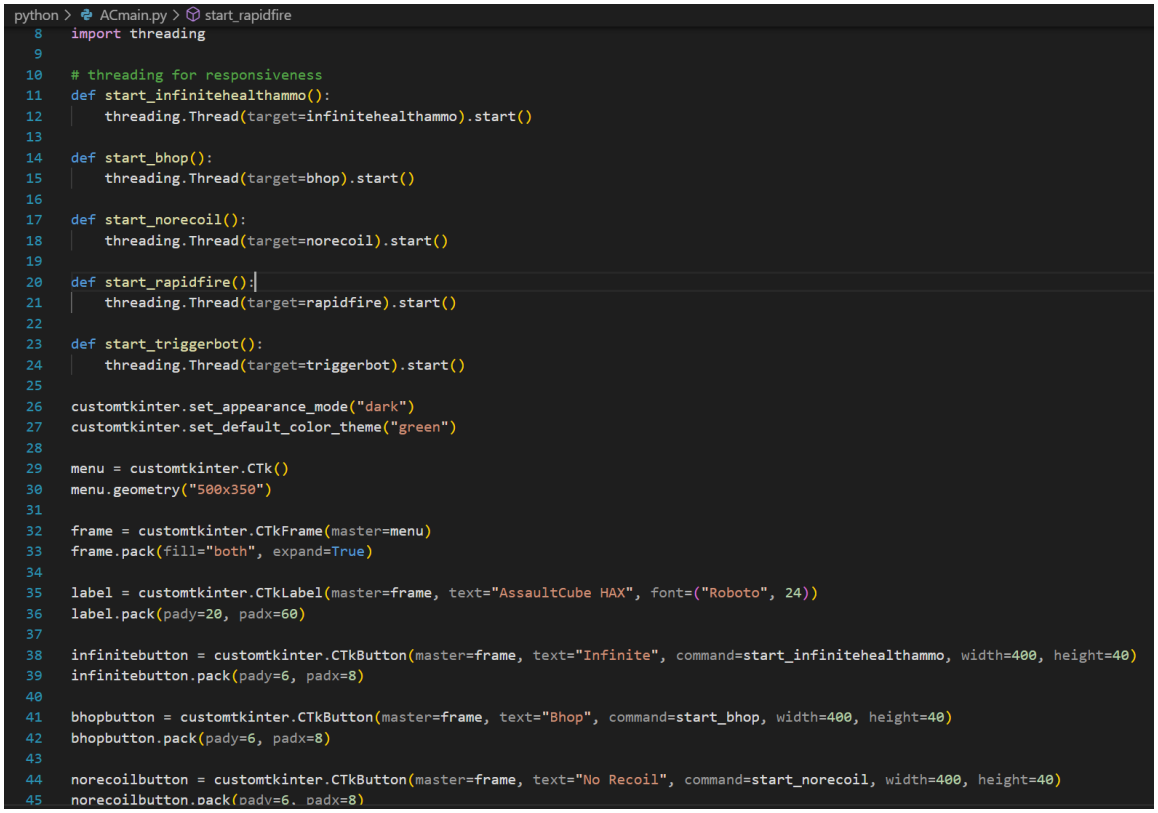


ChatGPT suggestions for why the mod menu crashes after running one cheat. Realised it was because of threading.

6. External Dependencies:
   - If your mod menu relies on external libraries or APIs, ensure that they are functioning correctly and are compatible with your code. Any issues with external dependencies can affect the stability of your mod menu.
7. Thread Synchronization:
   - If you are using multiple threads in your program, make sure you have proper thread synchronization mechanisms in place to avoid race conditions and other threading-related issues.
8. Debugging:
   - Use debugging tools and techniques to identify the specific point in your code where the mod menu becomes unresponsive. This will help you pinpoint the problem and find a solution more effectively.
9. Profiling:
   - Profile your code to identify performance bottlenecks and areas that may be causing the slowdown. Profiling tools can help you optimize your code for better performance.
10. Game Anti-Cheat Measures:
    - Some games have anti-cheat measures that detect and block cheats or mods. If your cheats are being detected by the game's anti-cheat system, it could lead to instability or unresponsiveness. Be cautious when using cheats in games with active anti-cheat systems.

Without specific details about your code and the game you are modifying, it's challenging to provide a more precise solution. However, by investigating these common issues and debugging your code thoroughly, you should be able to identify and address the root cause of the mod menu's unresponsiveness.

Applied threading to mod menu GUI code to prevent crashing mod menu from crashing.



```
python > ACmain.py > start_rapidfire
 8  import threading
 9
10  # threading for responsiveness
11  def start_infinitehealthammo():
12      threading.Thread(target=infinitehealthammo).start()
13
14  def start_bhop():
15      threading.Thread(target=bhop).start()
16
17  def start_norecoil():
18      threading.Thread(target=norecoil).start()
19
20  def start_rapidfire():
21      threading.Thread(target=rapidfire).start()
22
23  def start_triggerbot():
24      threading.Thread(target=triggerbot).start()
25
26  customtkinter.set_appearance_mode("dark")
27  customtkinter.set_default_color_theme("green")
28
29  menu = customtkinter.CTk()
30  menu.geometry("500x350")
31
32  frame = customtkinter.CTkFrame(master=menu)
33  frame.pack(fill="both", expand=True)
34
35  label = customtkinter.CTkLabel(master=frame, text="AssaultCube HAX", font=("Roboto", 24))
36  label.pack(pady=20, padx=60)
37
38  infinitebutton = customtkinter.CTkButton(master=frame, text="Infinite", command=start_infinitehealthammo, width=400, height=40)
39  infinitebutton.pack(pady=6, padx=8)
40
41  bhopbutton = customtkinter.CTkButton(master=frame, text="Bhop", command=start_bhop, width=400, height=40)
42  bhopbutton.pack(pady=6, padx=8)
43
44  norecoilbutton = customtkinter.CTkButton(master=frame, text="No Recoil", command=start_norecoil, width=400, height=40)
45  norecoilbutton.pack(pady=6, padx=8)
```

Final Cheat Table

Active	Description	Address	Type	Value
☐	Player			
☐	Player	P->00EBA090	4 Bytes	004E4A98
☐	Name	P->00EBA2B5	String[10]	unarmed
☐	Health	P->00EBA188	4 Bytes	100
☐	Force Attack	P->00EBA2B4	Byte	0
☐	Force Jump	P->00EBA0FB	Byte	0
☐	On Ground	P->00EBA0F9	Byte	1
☐	X-Y Position	P->00EBA098	Float	91
☐	Z Position	P->00EBA09C	Float	12.5
☐	X View Position	P->00EBA0D2	Byte	135
☐	Y View Position	P->00EBA0D4	Float	0
☐	Weapon			
☐	Rifle Rounds	P->00EBA1E0	4 Bytes	20
☐	Rifle Reserves	P->00EBA1B8	4 Bytes	40
☐	Rifle Firerate	P->00EBA208	4 Bytes	0
☐	Pistol Reserves	P->00EBA1A4	4 Bytes	50
☐	Pistol Rounds	P->00EBA1CC	4 Bytes	10
☐	Pistol Firerate	P->00EBA1F4	4 Bytes	0
☐	Grenades	P->00EBA1E8	4 Bytes	0
☐	Display Name	00501C38	String[20]	Shrike
☐	Pointer to Entity List	P->0D856968	4 Bytes	0
☐	Team	P->00EBA3BC	4 Bytes	1

Video Demos

Video for Infinite health and ammo demo - Demonstrates infinite health and ammo hack
Video for Bhop demo - Demonstrates bhop script
Video for rapid fire demo - Demonstrates rapid fire script
Video for no recoil demo #1 - Demonstrates no recoil script being unable to look up or downwards while shooting
Video for no recoil demo #2 - Demonstrates no recoil script being able to look around while shooting
Video of triggerbot demo #1 - Demonstrates triggerbot script shooting constantly
Video of triggerbot demo #2 - Demonstrates triggerbot shooting one bullet at a time
Video of triggerbot demo #3 - Demonstrates triggerbot fixed but shooting at wrong targets (allies)
Video for GUI demo #1 - Demonstrates mod menu for one hack before crashing
Video for GUI demo #2 - Demonstrates mod menu working for multiple hacks
Video for triggerbot and mod menu demo - Demonstrates mod menu working and triggerbot shooting at enemies.