# **Utvärdering av Eclipse ett fusk till CS:GO**

# **Projektutvärdering i programmering 2**

### Resurser

Programmet är kodat i Visual Studio Community 2019. Källor som används till programmet är:

* <https://www.unknowncheats.me/wiki/Counter-Strike:_Global_Offensive>
* <https://github.com/frk1/hazedumper>
* <https://www.youtube.com/watch?v=pv9L30A_f-o>
* <https://www.youtube.com/watch?v=N46PtrT6hew>
* <https://www.youtube.com/watch?v=GgTQod8Kp0k>

### 

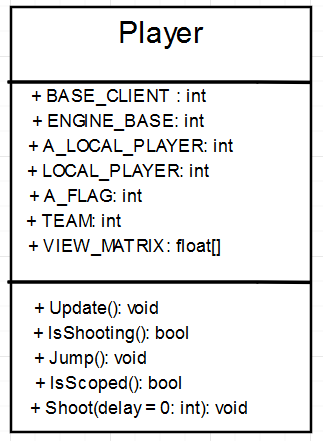
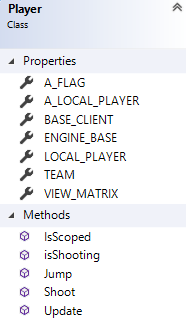
### Klassdiagram

**Player Klassen**

Denna klass innehåller information om spelaren som använder fusket. Som information om vad spelaren hade för Id (i spelet), vilket lag som spelaren var på samt funktioner som användes i fusken.

<https://i.imgur.com/6y7Yqhy.png>

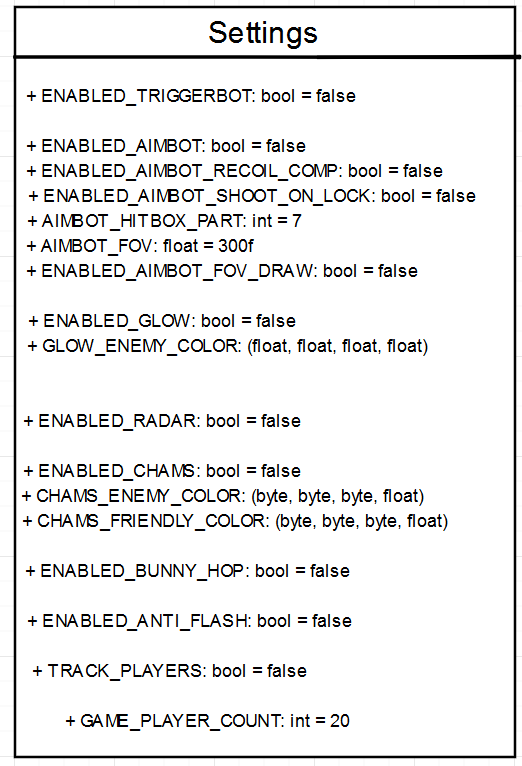
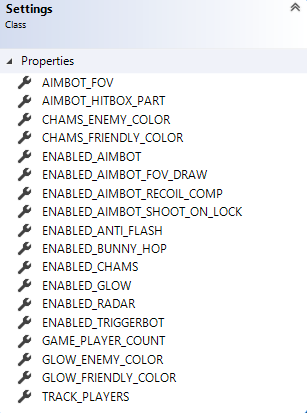
<https://i.imgur.com/dqOccoZ.png>



**Settings Klassen**

Denna klass håller koll på vilka fusk som är aktiva och vilka värden som ska sparas till varje fusk. Till exempel så när vi ändrar färg på spelaren så sparas det här och när vi aktiverar ett fusk så sparas det här att det fusket är aktivt.   
<https://i.imgur.com/daqa6Ta.png>

<https://i.imgur.com/JGSOhvF.png>



### Hur har projekt gått?

För mig så har projekt gått mycket bra, jag har lärt mig många saker angående hur klasser kan integreras i en applikation för att underlätta processen i kodningen. Jag ävent lärt mig mycket hur själva tekniken i hur spel använder sig av ram och hur data är sparat i spel. Detta har varit väldigt intressant och lärande för mig. Jag satsade även på att dela upp koden i detta projekt för att göra det mer läsbart. Till exempel flyttade jag funktioner till en separat fil och sedan kallar jag på dem där istället för att ha allt i samma fil.

Jag testade även att använda mig utav Structs och Enums för att lättare kunna skapa listor med värden och sedan koppla dem med ett namn för att göra det lättare att läsa koden. Exempel på detta är Hitbox enum som kopplar en kroppsdel till en int.

### Vilka fel har du stött på i koden och i programmet? Hur har du löst dem?

Det första problemet som jag stötte på var att jag fick ett problem där jag inte kunde köra ändra delar i formen under tiden ett fusk kördes, detta var på grund av att jag hade en while true loop i fusket som behövdes för att kunna köra den delen konstant. Vilket resulterade i att formen väntade bara på att fusket skulle bli klar och slutade svara när man gav den kommandon. Detta löste jag genom att använda något som kallas threads. Threading gör det möjligt att kunna hantera flera begärande från användaren samtidigt utan att köra flera kopiera av samma program samtidigt. Detta löste det problemet och ökade presterande på varje fusk, då det blev som sin egna lilla del som kördes. En tråd skapas på detta sätt

|  |
| --- |
| Thread myThread = new Thread(() => MyThreadFunctionThatIWantToCall()); myThread.Start(); |

Det andra stora problemet som jag stötte på var att jag inte visste hur jag skulle kod det mesta, och när man utvecklas fusk så görs det oftast i C++, då det är lättare att jobba med RAM och spelkod, då C++ är närmare maskinkod än C#. Detta resulterade i att jag läste väldigt mycket C++ kod och konvertera det till C# kod, som jag sedan använde i mitt program. Jag läste även väldigt mycket på forum som är dedikerade till fusk och hittade en wiki sida som hjälpte jättemycket, dock var den också skriven i C++ så det blev att konverterade koden där också .<https://www.unknowncheats.me/wiki/Counter-Strike:_Global_Offensive>

Värt att notera så genom hela kod processen så har jag googlat och sökt väldigt mycket och lärt mig väldigt mycket hur jag bör tänka och hur jag bör hantera delar i koden som använder bland annat trådning för att optimera prestandan. Hade till exempel ett problem där jag använde nästan all CPU kraft och programmet crashed, vilket jag löste genom att lägga till en Thread.Sleep i koden för att låta programmet vila lite mellan varje loop.

|  |
| --- |
| Thread.Sleep(50); |

### 

### Delar i koden som är värt att notera

**Varför använder jag “f” efter float värden?**

Detta är pågrund av att säga till kompilatorn att vi verkligen vill använda oss av en float och inte en double, som den ibland kan tro att det är.

**Vad är offsets i koden?**

Offsets är en memory address i spelet som är kopplad till det offsets värdet. Till exempel ditt HP i spelet har en offsets som bestämmer hur ett program kan hitta det värdet, vad ditt crosshair har för id eller hur vi kan tvinga spelet att skjuta. Det man gör är att man skriver till en av dessa offsets med ett startvärde, till exempel vill vi ha HP på en fiende så måste vi lista ut vad fienden har för startvärde och sedan söker vi efter start värdet + HP offseten vilket ger oss fiendens HP. Men allt utgår från bas spelar adressen (värdet på spelaren)

|  |
| --- |
| public const Int32 m\_iHealth = 0x100; public const Int32 m\_iCrosshairId = 0xB3E4; public const Int32 dwForceAttack = 0x31D54B4; |

|  |
| --- |
| int entity = Memory.Read<int>(Player.BASE\_CLIENT + Offsets.dwEntityList + 1 \* 0x10); // get the player entity int hp = Memory.Read<int>(entity + Offsets.m\_iHealth); // entity hp (57) |

### Är slutprodukten så bra som den kan bli?

Nej jag tror att slutprodukten kunde ha blivit mer optimerade med trådarna till exempel. Men då jag inte hade en större kunskap om detta innan jag började så är jag väldigt nöjd med vad jag har lyckats åstadkomma från att ha börjat från noll.

I frågan om trådar och om jag är nöjd med dem, så vill jag svarar ja och nej. Jag har nyligen börjat använd träningen för detta projekt och det finns säkert mängder med sätt att optimera hur trådar användes i ett program, då jag märkte att träningen är det som står för den mesta resurstaningen i programmet.

En annan del som jag skulle vilja utveckla är delar i koden som återanvändes kan brytas upp och delas upp i sin egna funktioner för att öka läsbarheten av koden och om vi skulle vilja ändra något senare så går det snabbare att bara ändra det på ett ställe istället för att ändra samma kod överallt.

### Fungerar den som den ska?

Efter dem testerna som jag har gjort så fungerade alla fusk utan några problem, och jag testade även dem 3 punkterna som jag skrev om i projektplanen:

**Hur fungerar projekt på en annan bana än den som projekt har utvecklas på?**

Detta funkade utan några problem, och jag hade ingen problem att köra på en annan banan än dem som jag testade och utvecklade på.

**Fungerar alla knappar och funktioner korrekt**

Alla knappar och inställningar fungerade utan några problem som jag kunde finna under testning av programmet.

**Kan programmet automatiskt ändra sig när olika spelar variabler ändras?**

Ja och nej, jag han inte utveckla denna delen än, men det skulle gå att ha en klocka i bakgrund som kör var femte sekunden och kallar på Player klass funktionen Update() för att automatiskt ändra detta. Istället så la jag till en knapp som går att klicka på som kallar på funktionen om något inte skulle fungera helt rätt.

Jag hade även inga syntax fel samt hittade jag inga kör problem under testning av programmet.

Testing bestod av att jag testade olika kombination av knappar och aktiverade och de aktiverade fusk för att testa om något skulle kollidera med varandra. Jag testade även att byta karta och spelläge under tiden programmet kördes och efter ha klickat på update player knappen så fungerade allt igen.

### I vilka situationer gör det inte det?

Under testning av programmet så stötte jag på ett problem som gjorde att jag hamnade i en oändlig loop vilket resulterade i att programmet frös och sedan kraschade, detta var på grund av att jag hade en while loop som aldrig tog slut och i den så gjordes massor av uträkningar som aldrig tog slut för att jag hade missat att avlsuta loopen när uträkingen var klar. Detta var en enkel fix som kunde implementerade . Detta var ett problem som jag hittade under tiden som jag testade programmet och det kom inte upp i kompileringen då det inte var något fel med koden utan hur koden kördes.

### Vad är det som kan förbättras?

I framtiden så skulle jag vilja utveckla 3 delar i programmet helst.

Den första delen är användarvänlighet i programmet, jag vill göra det lättare att navigera i formen, kanske använde ett bibliotek för att skapa snyggare forms som då blir lättare för användaren att navigera runt i. Detta hade gjort så att programmet såg finare ut och kanske då även kunnat locka flera använder till att använda programmet.

Som tidigare nämnt så är jag en nybörjare till att använda threads (trådar) och är något som jag skulle vilja lära mig mer om. Jag tror även att jag skulle kunna optimera koden mera för att använda trådar och göra så att programmet kör bättre och med mer prestanda. Detta är värt att utforska mera.

Som sista del i det jag vill förbättra är att programmet själv ska kunna hitta offset sen varje gång programmet startar. Detta då varje gång CS:GO släpper en uppdatering till spelet så måste dessa värden ändras, då i kompileringen (av CS:GO) så ändras dem. Detta hade gjort att man inte varje gång behöver lista ut (eller googla) dessa offsets värden och hade gjort det lättare att för användaren som använder programmet att inte behöva installera en ny version varje gång det kommer en uppdatering.