

Enkelt DRM system för spel

Emil Lindgren | T4 | 12/5-17

# Abstract

This report explores the possibility of a simple DRM limitation in a game, the limitations of such a system system and the process of developing the system. The report also discusses what DRM can be used for and how it works in principle.

Innehåll

[Abstract 1](#_Toc484130612)

[Inledning 3](#_Toc484130613)

[Bakgrund 3](#_Toc484130614)

[Syfte och frågeställning 3](#_Toc484130615)

[Metod 3](#_Toc484130616)

[Avgränsningar 3](#_Toc484130617)

[Arbete 4](#_Toc484130618)

[Spelet 4](#_Toc484130619)

[Hemsidan 6](#_Toc484130620)

[Resultat 7](#_Toc484130621)

[Diskussion 7](#_Toc484130622)

[Hjälpmedel 7](#_Toc484130623)

[Spelverktyg 7](#_Toc484130624)

[Databasverktyg 8](#_Toc484130625)

[Svagheter med DRM systemet 8](#_Toc484130626)

[Styrkor med DRM systemet 8](#_Toc484130627)

[Källor 9](#_Toc484130628)

# Inledning

Jag valde att göra en rapport om ett enkelt DRM system för att det är ett område som är tekniskt intressant, speciellt eftersom att DRM oftast är stora komplexa system. Ett enkelt system skulle kunna användas för att demonstrera implementeringar i olika program, vilket i sin tur kan vara användbart för att lära ut hur man använder sig av digitala lås.

## Bakgrund

DRM (*Digital Rights Management*)[[1]](#footnote-1) är ett omdiskuterat ämne för många, stora spelutvecklare vill använda sig av det för att förhindra piratkopiering, spelare brukar inte vilja ha det för att förhindra att spelet plötsligt slutar att fungera om eller då login servicen av spelet slutar fungera. Det finns även andra användningsområden för DRM t.ex. demo versioner av spel som är semi låsta med DRM. Vissa DRM system kan man även gå omkring med olika medel.

## Syfte och frågeställning

Syftet med denna rapport är att se om det är möjligt att skapa ett enkelt DRM system för ett spel. Frågeställningen är:

* Vilka problem kan uppstå?
* Finns det några för och nackdelar till det enkla DRM systemet?

## Metod

Arbetet började med att skapa ett enkelt spel i GameMaker: Studio[[2]](#footnote-2) som bestod av flera banor och ett mål man skulle nå, när man nådde målet blev man skickad till nästa bana. Sedan skapades ett DRM system för att låsa ner funktionalitet baserat på en inloggningsvariabel. Efter detta skapades en server för att ta emot inloggningar via nätet.

## Avgränsningar

Utvecklingsmiljön behöver inte alls ha lika många verktyg som den i rapporten, det går lika bra att bara använda GameMaker: Studio för att skapa spelet och använda en lokal server på samma dator för webbsidan.

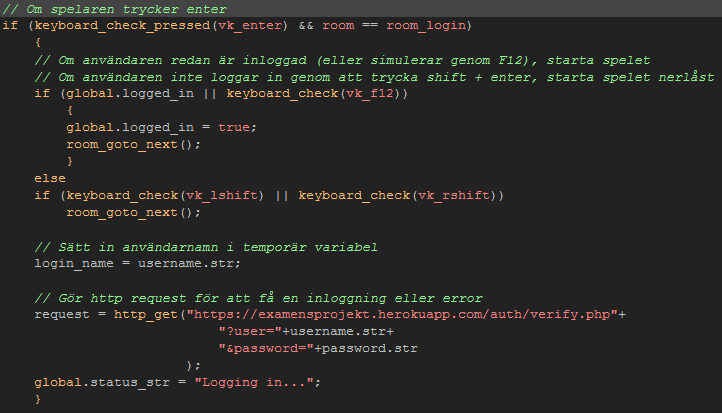
Eftersom att inloggningssystemet inte var tänkt för något annat än demonstrativa skäl blev den inte designad på ett säkert sätt, och bör därför inte användas för ett kommersiellt projekt.

# Arbete

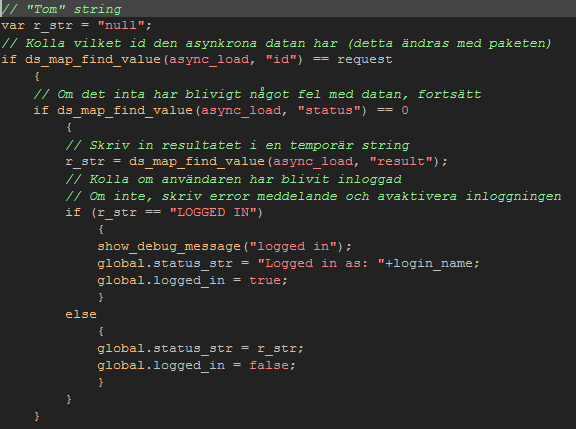
Utvecklingen började med att sätta upp miljön för att skapa själva spelet.

## Spelet

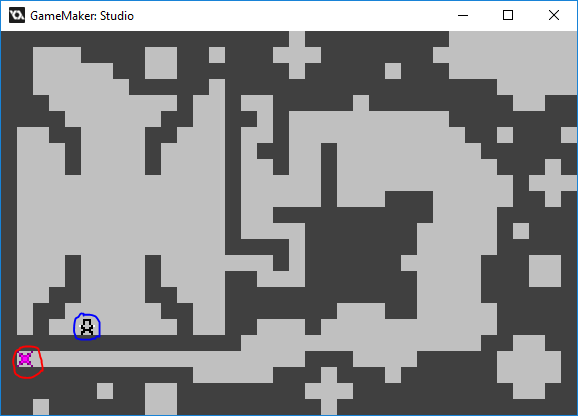
Det första som gjordes var att skapa ett interface som skulle göra förfrågningar till servern beroende på vad användaren gjorde, detta åstadkommes med hjälp av ett objekt som både interagerade med globala variabler och kunde sammansätta en hyperlänk med en login förfrågning.



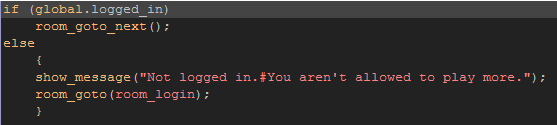
Eftersom att det returnerade resultatet kommer anlända asynkront (när som helst) används speciella rutiner för att använda informationen, dessa hamnar i andra delar av programmet. Detta är hur koden för att läsa av resultatet ser ut:



Efter detta skapades en spelare med enkel rörelse och solida block som inte går att röra sig igenom, och för att flera banor skulle fungera skapades också ett målobjekt som sade åt spelet att skicka spelaren till login skärmen med ett meddelande om spelaren inte var inloggad.

  
(Spelare inringad med blå färg, mål inringat med röd färg, solida block mörkgrå)

Detta hanterades genom at kolla en global variabel som *o\_drm* objektet hade satt till sant eller falskt före själva spelet började.

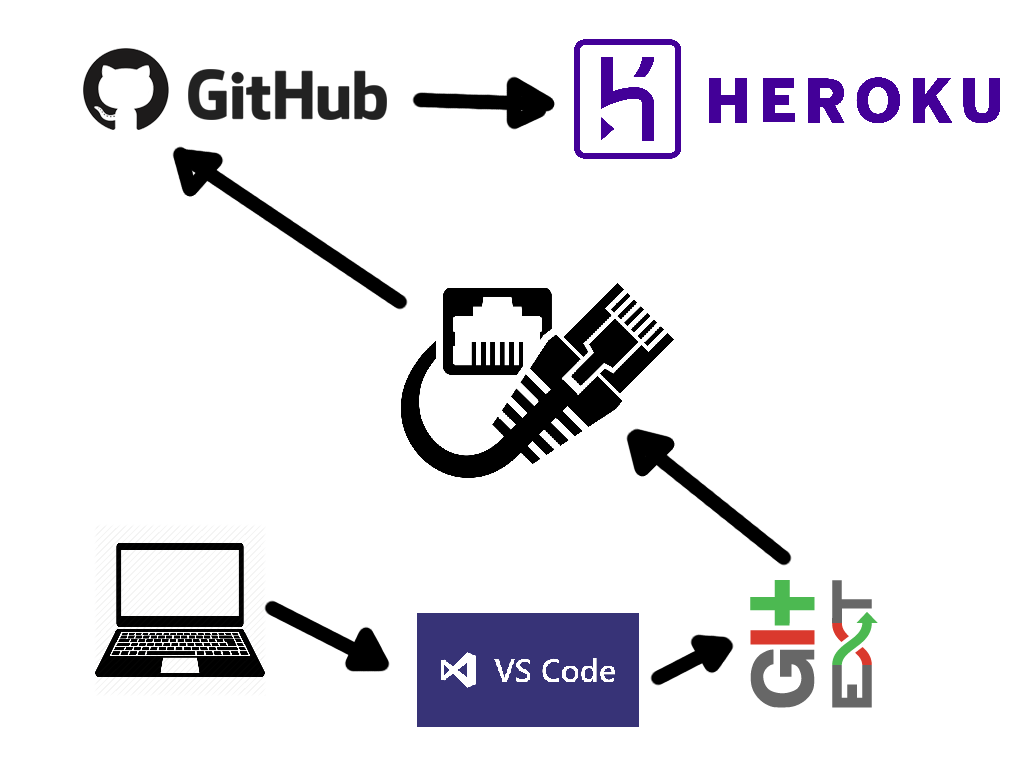


## Hemsidan

Det andra steget i utvecklingen var att skapa en hemsida som kunde ta emot hyperlänkar och ge tillbaka resultat som spelet kunde använda som en inloggningsbekräftelse.

Till att börja med skapades ett Heroku projekt för att agera som värd för hemsida och login tjänst. Efter detta skapades ett GitHub projekt som skulle fungera som basmappen där webbsidan låg, efter detta kopplades GitHub projektet in i Heroku projektet genom Herokus användarmiljö som automatiskt kan uppdatera hemsidan när GitHub projektet uppdateras. Efter detta användes Git Extensions[[3]](#footnote-3) för att lättare uppdatera GitHub projektet lokalt och sedan uppdatera det synliga GitHub projektet, till sist användes Visual Studio Code[[4]](#footnote-4) för att skapa och göra ändringar i hemsidan och login tjänsten.

Arbetsflödet blev då slutligen Visual Studio Code->Git Extensions ->GitHub->Heroku.



Som förfining av spelet fick spelaren grafik beroende på vad som händer, t.ex. när spelaren rör sig åt vänster kommer den spela en animation av att gå till vänster. Spelet fick även en logg där man kunde se om man faktiskt hade loggat in.

# Resultat

Inloggningsfunktionen fungerar som den ska och kan verifiera om användaren har loggat in eller inte, den var mycket enklare att implementera i spelet än att implementera i serverform, detta är för att man enkelt kan byta ut inloggningsfunktionen mot en knapptryckning eller andra medel.

Hemsidan fungerar också och när rapporten blev skriven kan man hämta programmet och källan av programmet via <https://examensprojekt.herokuapp.com/> eller alternativt <https://github.com/metroidchild/examensprojekt>.

Prestandan av systemet är mycket bra när antalet användare är nog litet, i teorin kommer det dock att ta linjärt mer tid och resurser för servern med ökningar av antalet användare.

Det enda problem som uppstod under utvecklingen var med verktygen som inte fungerade, om den inbyggda Git funktionaliteten i Visual Studio Code hade fungerat skulle kedjan från uppdatering av kod till att hemsidan uppdaterats gått mycket fortare. Detta är det största problemet som hände under utvecklingen.

# Diskussion

Visual Studio Code har Git funktionalitet och kan teoretiskt sett uppdatera GitHub projektet direkt utan att behöva externa verktyg, men eftersom att detta inte fungerade för mig använde jag mig av ett verktyg som jag hade använt förut och visste fungerade. För övrigt borde arbetet ha blivit bättre tidsanpassat, mer tid än nödvändigt gick åt till att få igång verktyg för att över huvud taget utveckla, t.ex. från början skulle projektet använda en fysisk server istället för ett webhotell.

## Hjälpmedel

Projektet hade inte varit möjligt utan hjälp av digitala verktyg som är inriktade för utveckling av både spel och websidor, detta är en snabb överblick på de verktyg som användes vid utvecklingen.

### Spelverktyg

Speldelen av projektet skapades med hjälp av GameMaker: Studio.

* GameMaker: Studio är ett nu mera avvecklat program som går att göra spel med, det är designat för att göra väldigt enkla spel, men möjligheten att göra mer avancerade spel och funktioner finns även där, detta projekt använde specifikt nätverksfunktioner.

### Databasverktyg

Databasdelen skapades med hjälp av Visual Studio Code, GitHub, Git Extensions och Heroku.

* Heroku var den slutliga plattformen för websidan som används som en primitiv och osäker databas och login tjänst, den använder sig av PHP för logik och en JSON fil som databas. Detta val gjordes för att GitHub projektet är öppet för alla att se, och redan då fallerar den säkerhet som kunde finnas. Därför lades inte mycket tid på själva databasdelen och istället på att få spelet att fungera. Om Heroku blir kopplat till ett GitHub projekt kan man välja att göra automatiska uppdateringar så fort GitHub projektet blir uppdaterat, detta gör cykeln mellan kod och testande mycket snabbare.
* GitHub användes som den centrala källhanteringen av projektet, eftersom att GitHub även kan kopplas till Heroku för att spara utvecklingstid var GitHub ett bra val.
* Git Extensions är ett grafiskt verktyg som speglar funktionaliteten i verktyget Git, detta användes för att enkelt uppdatera GitHub projektet och där med även Heroku.
* Visual Studio Code är en textredigerare som är designad för webbutveckling, men fungerar även bra till andra projekt.

## Svagheter med DRM systemet

Saker som skulle kunna ha förbättras är databasimplementeringen, trots att den är väldigt enkel är den mycket svag när det kommer till både skalning och säkerhet, om jag skulle gör om något liknande projekt i framtiden skulle jag nog istället använda mig av mer traditionella databaser och hålla mig till säkerhetsstandarder.

Ett problem med DRM systemet är att GameMaker: Studio har problem med att hantera unicode bokstäver, så error meddelanden visas inte alltid korrekt; å, ä och ö är helt osynliga.

## Styrkor med DRM systemet

Det enkla DRM systemet är väldigt starkt när det kommer till att snabbt testa låsta funktioner, cykeln mellan att ändra på kod och att faktiskt testa är väldigt kort. Detta hjälpte till och med till under utvecklingen av DRM systemet.

# Källor

Margaret Rouse. (den 12 maj 2017). *digital rights management (DRM)*. Hämtat från TechTarget:

<http://searchcio.techtarget.com/definition/digital-rights-management>

Michael Rohde. (den 12 maj 2017). *What Is GameMaker: Studio?* Hämtat från Dummies:

<http://www.dummies.com/programming/programming-games/what-is-gamemaker-studio/>

Heroku. (den 12 maj 2017). Hämtat från Heroku:

<https://www.heroku.com/>

Visual Studio. (den 12 maj 2017). Hämtat från Visual Studio Code:

<https://code.visualstudio.com/>

1. http://searchcio.techtarget.com/definition/digital-rights-management [↑](#footnote-ref-1)
2. http://www.dummies.com/programming/programming-games/what-is-gamemaker-studio/ [↑](#footnote-ref-2)
3. https://gitextensions.github.io/ [↑](#footnote-ref-3)
4. https://code.visualstudio.com/ [↑](#footnote-ref-4)