# Gesällprov Rapport

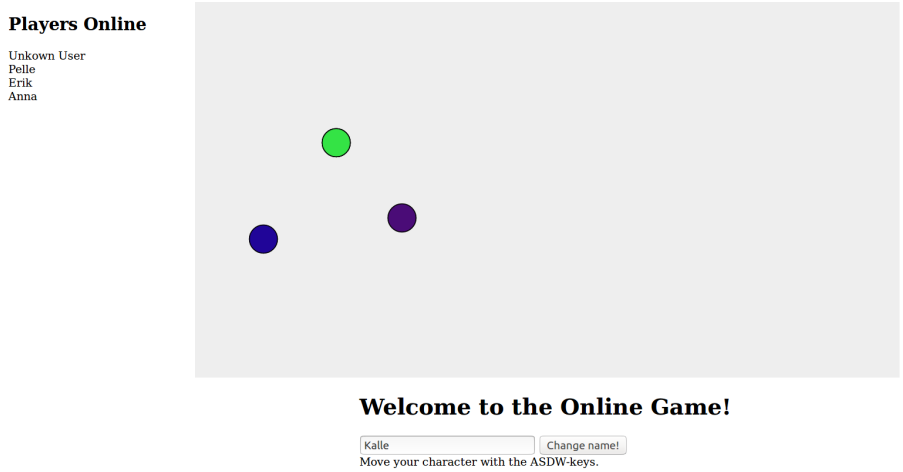
Aroshine Munasinghe

January 2019

## 1 Introduktion

För mitt Gesällprov har jag valt att bygga en spelmotor för ett 2D multiplayer-spel. Syftet för mig har varit att bygga en bas ovanpå vilken man ska kunna utveckla olika 2D-spel.

Spelet består av en backend och en frontend, båda skrivna i JavaScript. Frontenden använder sig av *jQuery* för att hantera diverse input från användaren och *d3* för att visualisera spelet. Backenden använder sig av NodeJs. Både front- och backenden använder sig av SocketIO för att kommunicera.




I spelet kan man styra en karaktär som ser ut som en färgglad circel. Med ASWD-tangenterna kan man flytta runt circeln. Om andra spelarna ansluter sig till spelet får de egna karaktärer och alla spelare som är anslutna kan se varandra i realtid. På vänstersidan finns en lista med spelare som är online för tillfället och under spelplanen har man möjligheten att ändra namnet på sin spelare. En livedemo finns på denna URL:en http://13.53.43.168/

# 2 Frontend

## 2.1 Main Loop

Spelet är frame-baserat, vilket betyder att en "huvudloop" körs kontinuerligt och alla visuella uppdateringar av spelet ska ske i denna loop, X antal gånger per sekund. Denna loop ritar bland annat ut alla spelare på skärmen. Här flyttas också spelaren genom att röra det grafiska objektet några pixlar per iteration av loopen.

## 2.2 Input från användaren

I javascript finns mig veterligen bara support för triggra funktioner när tanger trycks ner och när tangenter släpps. Det finns däremot ingen support för tangents som *hålls nertryckna*. Därför används några olika globala variabler (som MOVING_RIGHT, MOVING_LEFT, MOVING_UP etc) för att hålla koll på om spelaren håller någon tangent nertryckt. Denna information används sedan i main-loopen för att faktiskt flytta spelaren på skärmen.

Spelaren har också möjligheten att registrera ett smeknamn för sig själv. Detta propageras sedan till servern som propagerar det till alla andra användare.

## 2.3 Kommunikation med servern

Parallellt med main-loopen sker läsning från servern via en så kallad websocket. Websocket är ett alternativ till typisk kommunikation som sker via XMLHttpRequest (även känt som Ajax). Både HTTP och Websocket är baserade på protokollet TCP, men kommunikationen är bidirektionell och kan ses som en kontinuerligt ström av data istället för data uppdelad i requests.

Det finns två typer av meddelanden som frontenden tar emot från servern. Det ena är *joinGame* som skickas så fort en anslutning med servern har skapats. Detta låter frontenden veta att servern har bekräftat frontendens anslutning. *joinGame*-meddelandet innehåller information om användaren som har genererats av servern, t.ex användar-id, koordinater, m.m.

Det andra meddelandet som frontenden tar emot är *gameState*. Detta meddelande skickas av servern och innehåller information om hela spelet. GameState-objektet innehåller bland annat information om alla andra spelare och deras koordinater. Detta används sedan av main-loopen för att rita ut de andra spelarna.

# 3 Backend

## 3.1 HTTP

Med NodeJs kan backenden fungera som en helt vanlig webserver. Denna webserver kan ta emot besökare och servera diverse filer. När besökare går in på hemsidan serveras en *index.html*-fil. Denna innehåller själva frontend-koden. Webservern kan också servera andra statiska filer med sin */files*-endpoint.

## 3.2 Gamestate

För att ett onlinespel ska fungera är det viktigt att man håller koll på vilket state som spelet befinner sig i. Detta görs genom variabler som hålls i RAM-minnet. De viktigaste av dessa variabler är är *sockets* och *users*. Båda är javascript-objekt som är nycklade med användar-id:n. Med andra ord, varje fält i objekten representerar en användare och nyckeln till det fältet är användarens id. *Sockets*-objektet innehåller alla sockets som används för att kommunicera med de olika frontends som har anslutit sig till spelet. *Users*-objektet innehåller all information om de användarna som är anslutna till spelet i realtid, t.ex koordinater, namn etc.

## 3.3 Websockets

Parallellt med att servera HTTP-anrop sköter backenden också kommunikationen med de websockets som har etablerats med frontenden. Så fort frontenden laddas kommer frontenden skickas ett request till backenden som etablerar en kanal för websocket-kommunikation. Så fort denna websocket har etablerats genereras ett unikt användar-id samt användardata från en mall. T.ex börjar alla spelare på plats $(300, 300)$ och de heter *"Unkown User"*. Denna användardata skickas sedan till frontenden via *joinGame*-meddelandet beskrivet ovan. Om antalet spelare överstiger 10 kommer den äldsta spelare att sparkas från spelet.

Servern lyssnar efter tre meddelanden från frontenden. Det första är *registerUsername* vilket redigerar den givna spelarens namn på skärmen. Det andra är *move* vilket uppdaterar spelarens position i *users*-objektet. Detta meddelande skickas av frontenden kontinuerligt så fort spelaren rör på sig. Det sista är *disconnect* vilket är ett meddelande som skickas automatiskt när spelaren stänger ner sitt fönster i frontenden. När servern tar emot detta meddelande tas den givna användaren bort ur *users* och *sockets*-objekten.

### 3.4 Main loop

Även backenden har en main loop som körs 25 gånger per sekund. Denna loop propagerar ut information till alla spelarna och utan denna loop skulle man inte kunna uppnå multiplayer-effekten i spelet. Loopen vandrar igenom samtliga anslutna användare och skickar hela gameState:t till deras respektive sockets. På detta sättet kan information propageras ut till spelarna så fort gameState:t ändras. Exempelvis om en användare byter namn kommer det att laddas upp till servern, servern redigerar *users*-objektet och detta skickas sedan direkt ut till alla användare i nästa iteration av main-loopen.