Labb Inet: Internet/sockets

Denna laboration går ut på att implementera ett enkelt spel för minst två samtidiga spelare med varsin klient.

Syftet med labben är att studera internet-orienterad programmering och då speciellt kommunikation via sockets (det finns en officiell tutorial om sockets att läsa för den som är intresserad) och konstruktion av protokoll. Det är särskilt viktigt att du kan beskriva hur kommunikationen fungerar och därför är det viktigt att ni kan specificera protokollet som programmen använder i detalj.

De flesta studenter skriver denna labb i Python eller Java men det kan vara lärorikt att skriva i t.ex. Go eller Erlang som är konstruerade för att hantera parallellprogrammering. Erlang är däremot inte så bra på att hantera strängar och man kan behöva skriva en liten frontend i Java eller Python för användargränssnittet.

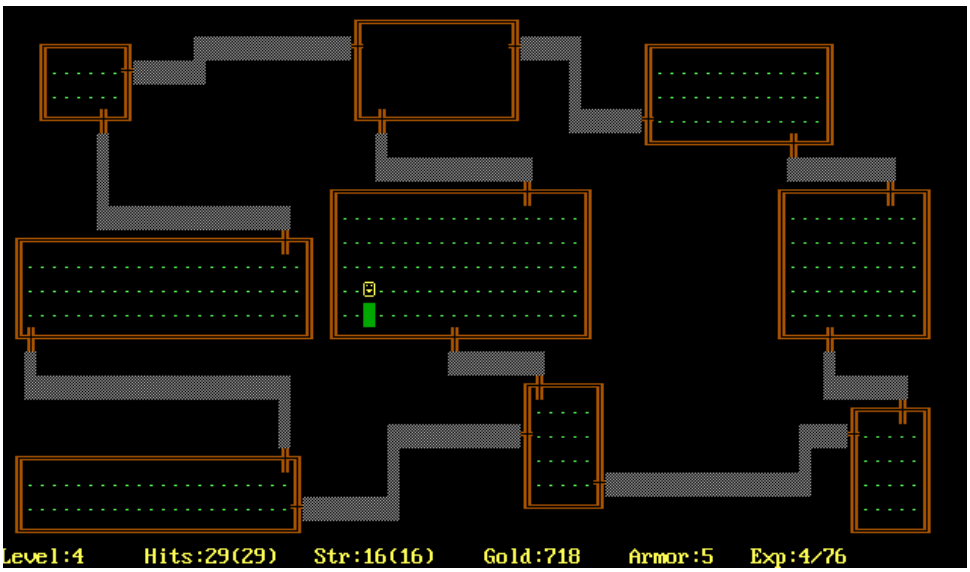
(Denna labb rättas inte på Kattis.)

Bakgrund

1980 kom spelet Rogue och det har gett upphov till en hel uppsjö med Rogueliknande spel. Att implementera en Rogueklon har i alla tider varit den mest klassiska av klassiska programmeringsuppgifter och nu har den letat sig till den här kursen också, men i mycket nedbantad form. Att vinna i Rogueliknande spel handlar om att besegra monster eller andra spelare, ta deras saker och använda dessa för att låsa upp hinder och ta sig vidare till någon slags mål. Vi vill inte begränsa er kreativitet för mycket i vad detta mål är, men samtidigt sätta vissa gränser så att spelen och protokollen varken blir för avancerade (så att uppgiften inte blir klar i tid) eller för enkla (så att de inte utmanar er att lära er kursinnehållet).

Uppgift

Uppgiften går ut på att implementera (1) klient och (2) server samt (3) ett väldokumenterat protokoll för hur klienten och servern kommunicerar med varandra. Skriv både klient- och serverdelen så att kraven på nästa sida uppfylls.



Figur 1: Bild ur spelet Rogue från 1980 (Källa: Wikimedia commons).

Krav

Klienten och servern behöver uppfylla följande krav.

Systemet

1. Spelarna behöver kunna flyttas omkring i 4 riktningar genom att trycka på piltangenterna eller klicka på knappar i ett gränssnitt.
2. Det ska finnas föremål som kan plockas upp och då ska dessa försvinna från spelplanen för övriga spelare. Åtminstone vissa föremål behöver vara relevanta för målet.
3. Spelarna behöver kunna bära på saker som plockats upp från spelbrädet.
4. Det ska finnas hinderrutor '#'. Om en spelare försöker flytta sig in i ett hinder så ska spelet fortsätta men utan att utföra förflyttningen.
5. Spelarna blockerar varandra, så om en spelare försöker gå dit som en annan spelare redan står så ska spelet fortsätta men utan att utföra förflyttningen.
6. Det ska finnas ett mål med spelet.
   (a) Målet måste involvera båda spelarna.
   (b) Målet får till exempel vara att besegra den andra spelaren.
   (c) Målet får till exempel bygga på samarbete.
   (d) När målet är uppfyllt ska spelet avslutas med en text (eventuellt olika) till båda spelarna.
7. Koden ska struktureras i funktioner och klasser om du använder det. Ingen funktion eller klass får bli för stor (mört, mört, mört, blåval-mönstret).
8. Koden ska dokumenteras bra.

Spelet bedöms helt på sina tekniska meriter. Ni behöver inte...

1. implementera inloggning eller någon form av persistens (databas).
2. implementera ett avancerat grafiskt användargränssnitt. Teckengrafik i ncurses går bra.
3. ha ett stort spel med flera våningar, olika nycklar och många olika monster.
4. skriva en story till spelet.

Det är inte tillåtet att...

1. använda web sockets för protokollet.
2. förenkla blockerande rutor eller spelare genom att göra dessa till omedelbara vinster eller förluster.
3. förenkla målet så att en spelare kan göra allt som krävs för att vinna utan att interagera med någon annan spelare.
4. förenkla labben genom att bara göra labb F3 spelbar via Internet.

Protokollet

Protokollet ska dokumenteras väl. Dokumentationen ska:

1. Innehålla en utförlig beskrivning av datan som skickas mellan server och klient.
2. Innehålla ett tillståndsdiagram (state diagram) som beskriver vilka tillstånd klienten och servern kan ha samt vad som gör att de övergår från ett tillstånd till ett annat.
3. Vara så detaljerat att en godtycklig student som har klarat kursen kan bygga en ny klient som fungerar med din server utifrån endast protokollet.

Protokollet ska dessutom uppfylla följande krav:

1. Protokollet ska antingen skicka text (ASCII, ISO8859-1 eller helst UTF8) eller vara binärt.
2. Protokollet måste vara på operativsystemsnivå och inte på webbnivå (web sockets är en helt annan sorts utmaning än denna labb).
3. Ha viss datasäkerhet. Det ska inte gå att krascha servern (eller klienten) genom att skicka trasig data. Ni behöver inte skydda er mot timingattacker.
4. Protokollet får inte vara programspråkspecifikt. Det är till exempel inte tillåtet att förvänta sig att mottagaren automatiskt kan deserialisera ett Python-objekt.
5. Servern behöver ha viss robusthet. Den ska till exempel kunna hantera att en klient disconnectar.
6. Protokollet ska delas upp i olika funktioner/metoder.

Tips

1. Använd ncurses för klienten och en enkel meny för servern. Servern kan logga med utskrifter i terminalen.
2. Låt klienten vara en tunn klient och lägg istället mer logik på servern. Det är tillåtet att strömma all grafik från servern.

Vid redovisning

1. ... ska du ha tre terminalfönster uppe med en server och minst två spelklienter som körs.
2. ... ha ditt protokoll redo för att kunna visa upp det.
3. ... ska du kunna redogöra för olika tillstånd i klient och serverdelen.
4. ... ska du ha en editor uppe med all källkod.
5. ... ska du kunna redogöra för alla detaljer i koden.