Sammanfattning

GPSTron är en ny variant av det klassiska spelet Tron. Spelet går ut på att de tävlande styr farkoster på en spelplan med mål att stänga in varandra. Spelarna lämnar “spår” efter sig som i efterhand inte kan passeras. Om en spelare ändå åker in i ett spår eller i spelplanens kant krockar denne och är ute ur spelet. Den som klarar sig längst tid utan att krocka med något spår vinner.

GPSTron gör det möjligt att spela Tron på riktigt med hjälp av en bärbar dator utrustad med mobilt Internet och en GPS. Spelarna kan välja ut ett område, välja en karta över området och sedan spela Tron i området genom att färdas omkring på exempelvis cykel eller i bil medan deras rörelser registreras på kartan.

Bruksanvisning

Spelet består av en server- och en klientdel.

Server

Servern ska köras på en dator, denna behöver inte nödvändigtvis delta i spelet. Servern startas i en terminal med kommandot ”java TronServer” och accepterar argumentet -P för att specificera en port. Standardporten är 4554.

Klient

Klienten är helt grafisk och startas med kommandot ”java GPSTron”. Man kan chatta med övriga spelare genom att skriva text i den nedre textrutan. Klienten sparar en fil med inställningar i ~/.gpstron/config.txt.

Spelet

Den spelare som först ansluter till servern blir host och står högst upp i listan med namn. Denna spelare måste innan spelet kan startas trycka på ”Set coordinates” och välja koordinater för spelområdet. Dessa skickas då till servern för att servern ska veta var spelområdet ligger. Observera att endast spelaren som är host kan göra detta.

För kartor rekommenderas openstreetmap.org. Där kan man gratis ladda ner kartor med tillhörande GPS-gränser. Samtliga spelare bör för sin egen skull använda samma karta och gränser som spelaren som är host.

Spelet startar när samtliga spelare tryckt på knappen ”I'm ready!”. Det finns en 10-sekunders säkerhetsmarginal innan man kan krocka med sitt eget spår. Man kan således inte befinna sig på samma plats i mer än 10 sekunder. Spelare som passerar gränsen för spelplanen ”dör” omgående.

När spelet tagit slut återgår programmet till ursprungsläge efter 10 sekunder och man kan om man vill starta ett nytt spel.

Uppbyggnad

Servern

Serverns viktigaste uppgifter är att sköta kommunikationen mellan spelarna, dvs distribuera koordinater och övrig information. Servern innehåller även kollisionshanteraren som räknar ut om spelare krockar.

TronServer.java

Denna klass utgör det exekverbara objektet. Det samlar upp inkommande anslutningar och gör för varje anslutning ett objekt av klassen PlayerThread. Dessa skickas sedan vidare till ett objekt av ControlThread.

ControlThread.java

Denna klass är huvudinnehållet i servern och sköter de flesta funktioner och spelregler. run()-metoden distribuerar spelarnas koordinater regelbundet.

Publika metoder

delPlayer(PlayerThread p)

Utför de operationer som behövs för att ta bort en spelare då denna kopplat ned från servern.

int getNumber()

Varje spelare har ett unikt id-nummer. Denna metod håller räkningen och returnerar nästa lediga id.

newPlayer(PlayerThread p)

Utför de operationer som behövs när en ny spelare ansluter.

reportDeath(PlayerThread p)

Utför operationer då en spelare har krockat och inte längre ska vara med i spelet.

restart()

Startar om spelet. Tar bort data från tidigare spel, sätter allas status till ”Not ready” osv.

saveBoundaries(double n, double e, double s, double w)

Ändrar spelområdets gränser.

sendToAll(String s)

Skickar ett meddelande s till alla anslutna spelare.

sendToOthers(int n, String s)

Skickar meddelandet s till alla utom spelare med id n.

PlayerThread.java

För varje spelare skapas ett objekt av denna klass. Objektet sköter kommunikationen med sin spelare och innehåller information om spelaren.

Publika metoder

sendMsg(String s)

Skickar meddelandet s till den klient som motsvarar PlayerThread-objektet.

flush()

Tar bort alla tidigare sparade koordinater ur koordinat-vektorn

Map.java

Detta objekt startas ur ControlThread och håller koll på var på spelplanen spelare har passerat. Objektet omvandlar GPS-koordinater till heltal i en 2000x2000-matris. På varje position i matrisen finns sedan ett Spot-objekt som innehåller information om ifall en spelare har passerat där och i så fall också vilken tid och vilken spelare det var.

Publika metoder

Point check(Point p1, Point p2, int id)

Kollar om någon krock har inträffat under förflyttningen från p1 till p2 för spelare nr id. I så fall returneras krockpunkten tillsammans med ett true-värde.

clear()

Rensar fältet och förbereder för ett nytt spel

setEdges(double n, double e, double s, double w)

Specificerar områdesgränser för att möjliggöra omvandling till matrispositioner.

Point.java

Objekt av denna klass representerar en punkt i xy-planet i antingen int- eller doubleformat. Dessutom finns ett booleanvärde som används i check() i Map.java för att tala om om punkten existerar eller inte.

Spot.java

Klassen som bygger upp matrisen i Map.java. Innehåller en boolean som talar om om någon passerat, samt en int för spelarid och en Date för när objektet passerades.

Klienten

GPSTron.java

Denna klass innehåller main-metoden och bygger upp fönstret. Fönstret består i sin tur av ett View- och ett Manipulation-objekt. Storleken är anpassad för att spelet ska kunna spelas på 1024x600, vilket är en vanlig upplösning på Netbooks.

GPSTron.java startar även objekt av klasserna Protocol och Updater vilka utgör stommen i programmet.

View.java

Synligt objekt som ritar upp kartan och allt som ska vara ovanpå den. Till detta används en dubbelbuffrad Canvas, DBCanvas, från Brownsk rörelse-labben. Klassen innehåller metoder för att omvandla GPS-koordinater till rit-koordinater samt metoder för att rita tjocka linjer och kryss.

Publika metoder

newFile(File f)

Laddar en ny kartbild som bakgrund

newCoords(double n, double, e, double s, double w)

Sätter gränser för kartan.

Manipulation.java

Synligt objekt som innehåller alla knappar och textfält till höger i programmet. För att ordna objekten används en GridBagLayout som tillåter bra kontroll över placering.

Publika metoder

updatebuttons()

Används av Updater-objektet för att se till att texten på Ready- och Connectknappen är rätt. Är man ansluten ska det exempelvis stå ”Disconnect” istället för ”Connect” på knappen.

Protocol.java

Denna klass innehåller operationer som ska utföras för alla kommandon som kan mottagas från servern. Det mesta av informationen rörande spelaren finns sparad i instansvariabler här eftersom alla objekt känner till Protocol-objektet.

Publika metoder

setSocket(String s, int i, String n, String c)

Metod för att ansluta till server på adressen s på port i. När anslutningen har upprättats skickas namn n och färg c till servern.

disconnect()

Saker att utföra för att koppla ner från servern. Meddelar servern att nedkoppling sker och stänger socketen.

sendMsg(String s)

Skickar meddelande s genom socketen till servern.

saveConf(String s)

Tar en string s på formen ”asd=123” och skickar det till Config-objektet för att spara det i en fil.

String getConf(String s)

Om s är asd returnerar getConf 123 om inställningarna är som i exemplet ovan.

getNewPos()

Kontaktar navagationsobjektet och uppdaterar positionen som finns lagrad som instansvariablar.

Updater.java

Detta objekt ser till att saker sker med jämna mellanrum. Saker som att få ny position, skicka denna till servern, rita om kartan och uppdatera chatfältet är exempel på saker som kontrolleras av detta objekt.

GPSReader.java / Keyboard.java

Dessa klasser syftar till att bestämma spelarens position. Keyboard.java är en hjälpklass som möjliggör simpel navigering med tangentbordet för att lättare kunna testa programmet. GPSReader.java läser GPS-koordinater från programmet gpsd som måste vara konfigurerat och startat. Utåt har klasserna samma gränssnitt med två instansvariabler där koordinater för longitud och latitud lagras.

För att växla mellan GPS och tangentbordsstyrning måste man ändra i Protocol.java innan programmet kompileras. Observera att startkoordinaterna i Keyboard.java bör ändras för att motsvara de koordinater man har inställda.

För GPS-styrning ska rad 20 och 30 vara aktiverade och rad 21 och 31 utkommenterade. För tangentbordsstyrning ska rad 21 och 31 vara aktiverade medan rad 20 och 30 är utkommenterade.

Player.java

Objekt av denna klass representerar övriga spelare på servern. Medan informationen om den egna spelaren lagras i ett Protocol-objekt skapas ett Player-objekt för de andra spelarna. I en Vector lagras de punkter som spelaren passerat och som ska ritas upp. View-objektet ansvarar för att lägga in nya punkter i Vectorn innan varje omritning eftersom de först ska konverteras till ritkoordinater.

Publika metoder

changeCol(String c)

Används då servern meddelar att spelaren har valt en ny färg. Den nya färgen tolkas och lagras i en instansvariabel.

changeName(String n)

Används då spelaren byter namn. Namnet lagras i en instansvariabel.

currPos(double x, double y)

Ändrar spelarens position.

die(double x, double y)

Används då spelaren krockat för att tala om att spelaren är ute ur spelet samt lagra punkten där krocken skedde.

Config.java

Detta objekt ansvarar för filen med inställningar som sparas i ~/.gpstron/config.txt. Där lagras senast använda karta, dess gränser, namn samt färg. Config-objektet nås via Protocol där de två metoderna för att spara en inställning och hämta en inställning finns.

Point.java

Objekt av denna klass representerar en punkt i xy-planet i antingen int- eller doubleformat.

SetCoords.java

Ett objekt av denna typ startas när man trycker på ”Set coordinates” i programmet. Ett nytt fönster öppnas i vilket man kan specificera gränserna för kartan. Klassen har inga publika metoder utan skickar bara de nya koordinaterna till Protocol för att sparas i inställningsfilen och till View så att de kan användas då positioner ritas ut.

Källor

Till hjälp vid programmeringen har jag enbart använt Google för att hitta information och olika exempel på kodsnuttar. Ett urval av sidor jag haft stor nytta av är:

Java API	http://java.sun.com/j2se/1.5.0/docs/api/
Metoden connectGPS()	http://lists.berlios.de/pipermail/gpsd-users/2005-January/000216.html
Java2s	http://www.java2s.com/
Roseindia	http://www.roseindia.net/
Om sockets	http://java.sun.com/docs/books/tutorial/networking/sockets/
Inspiration	http://codeninja.de/tron/