PROJEKT TRAINS

Sebastian Strindlund

sest1601

05-16-2017

VT17 DT060G Datateknik GR (B),

Objektorienterad programmering i C++, 7,5 hp

Uppgift

Konstruera en prototyp till simuleringsverktyg IronBend Train and Brain Railway Inc vill ha. Skapa klasser för tåg, fordon, stationer, data, och simulering. Läs in filer och använd datan för att simulera tåg som går mellan olika stationer på olika tider, simulera dessutom att tågen letar efter fordon (lok och vagnar) som den behöver på sin avgångsstation. Simulera medelhastighet beräknat på avgångstid, ankomsttid och sträcka. Hantera olika “states” som tåg kommer att ha / “Not assembled”, “Incomplete”, “Assembled”, “Ready”, “Running”, “Arrived” och “Finished”. Lägg in allt som event i en simuleringskö och kör eventen genom diskret händelsestyrd simulering. Under simuleringen ska man kunna få information om tåg, fordon m.m och efter simuleringen ska man kunna skriva ut lite statistik.

Laborationsdata

Sebastian Strindlund
login ID: sest1601
Projekt “TRAINS”
Inlämningsdatum 2017-05-17
Betyg: Siktar på ett C.
Utvecklingsmiljö: **Visual Studio Community 2015**

Filer

sest1601_DT060G_PROJEKT_rapport.pdf

Database.h
Database.cpp
Event.h.
Event.cpp
Fordon.h
Simulation.h
Simulation.cpp
Train.h
Train.cpp
TrainStations.h
TrainStations.cpp
main.cpp

Grupparbete

Jobbade med Peter Näs pens1600, vi gjorde inte ett optimalt lagarbete då vi ibland jobbade på samma saker eller en sak som den andre redan hade blivit klar med. Det vi borde ha gjort var att innan vi började så skulle vi ha bestämt bättre vem som skulle ha gjort vad för att undvika onödigt arbete. Det blev så att vi arbetade egentligen var för sig men vi delade generella ideer för hur man skulle lägga upp funktionerna. En del av funktionerna kommer vara i princip identiska, men det mesta har man skapat själv.

Det som jag inte har skrivit själv är “Simulation::get_StationDistance”, menyerna i simulation.cpp, “Simulation::print_TrainTimeTable()”, Simulation::change_time_for_START_END_or_INTERVAL()

Lösning

För att lösa uppgiften har jag använt mig av följande klasser:

“Database” - Läser in data från filerna, lagrar och servar informationen (tåg, tågstationer och distans mellan stationer) till klasser som frågar efter informationen. Simulation och event är dem som frågar efter informationen, dessa kan även ändra informationen som databasen håller (lägga till tåg i station, ta bort tåg från station, lägga till fordon, ta bort fordon osv.)
Motivering: Skapade denna klass för att det kändes bra att lagra detta på ett separat ställe.

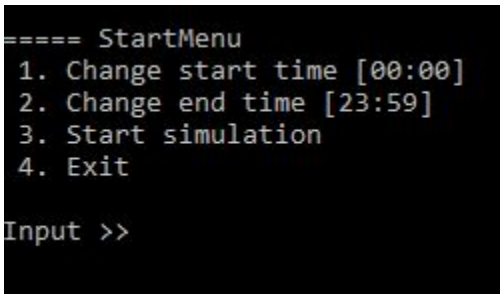
“Train_Cart” - En abstrakt klass som innehåller innehåller Då bara ID, typ, state och historik, sedan har jag deriverat klasserna SleepingWagon, PersonWagon, OpenCargoWagon, CoveredCargoWagon, ElectricLoco och DieselLoco utifrån den. Dessa innehåller då bara saker som är unika för dessa fordon som exempelvis effekt, är unikt för elektriska lok.
Motivering: Var helt enkelt det bästa sättet då man kunde utnyttja dynamisk binding och bara derivera klasserna som vi övat på i tidigare labbar.

“Train” - Hanterar tåg. Tågen innehåller information som tågnr, avgångsstation, ankomst station, avgångstid, ankomsttid, max hastighet, vilket tillstånd tåget är i (NOTASSEMBLED, INCOMPLETE, ASSEMBLED, READY, RUNNING, ARRIVED, FINISHED) vilka fordon tåget har och vilka fordon tåget behöver för att kunna åka iväg. Kommunicerar med simulation, event och trainstation.
Motivering: Behövs såklart en klass för tåg då uppgiften handlar om tåg. Bra att dela upp det i flera klasser dessutom.

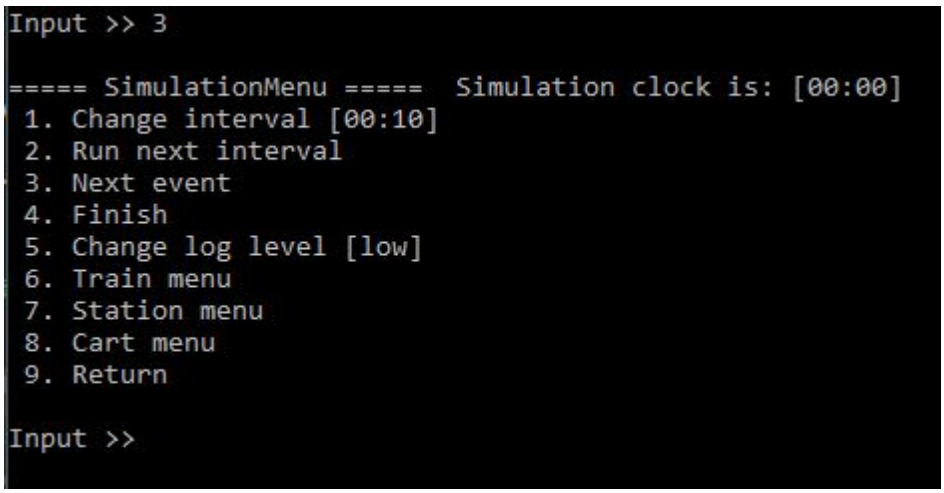
“Trainstation” - Hanterar stationer. Innehåller namn, vilka fordon som finns i stationen och vilka tåg som finns i stationen. Kommunicerar databas, simulation, train och event.
Motivering: Även denna kändes rätt självklar eftersom att en station är en hörnpelare när det kommer till tåg - Det finns alltid avgång och ankomst stationer. Förutom detta så är det ju bra eftersom det blir en bra struktur på det hela.

“Event” - Abstrakt klass som innehåller det som krävs för att skapa event som man kan lägga till i en eventkö som sorterar eventen baserat på det event som har minst tid (tidigaste eventet ligger först i kön).innehåller tid för eventet. Sedan har jag deriverat alla olika event typer utifrån denna. Alla event har en egen process() funktion som gör olika saker, som att lägga till ett nytt event för nästa steg (exempelvis från arrived till finished).
Motivering: Eftersom det tipsades om en Diskret händelsestyrd simulering och vi skulle då använda oss av olika event så var det även här lämpligt att göra en Abstrakt klass som man deriverade ut till de andra eventen där själva process() dvs funktionen var unik för varje event.

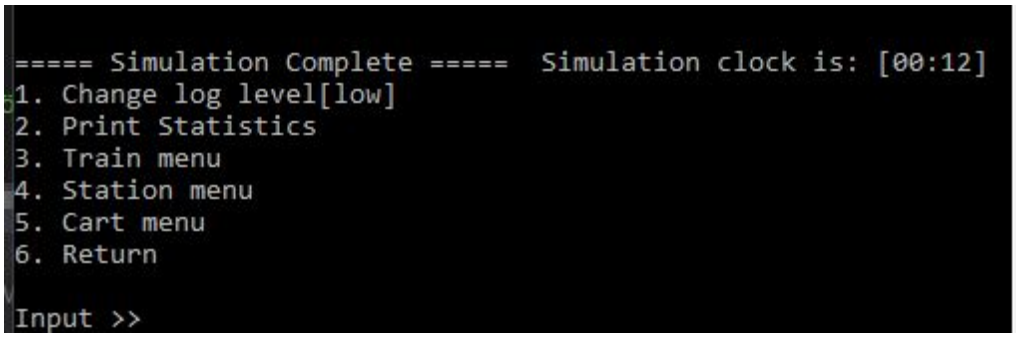
“Simulation” - Sköter det mesta som användaren ser, skriver ut menyer, tar in och kontrollerar användarinput, stegar igenom eventkön baserat på hur långt fram användaren vill hoppa i varje steg, sköter start och stopptid som användaren valt, söker tåg, stationer och presenterar information, baserat på log-level, loggar varje statusförändring som sker för tåget inom intervallet som användaren valt till Trainsim.log. Det är även i Simulation som programmet körs i run().



Startmenyn för programmet.



Simuleringsmenyn, här kan man byta hur stora hoppen i alternativ 2 kommer att vara, köra bara nästa event (3), Köra igenom hela simuleringen (4), toggla log-level mellan low och high (5), starta tågmenyn (6), starta stationsmenyn (7), starta fordonsmenyn (8).



Denna meny dyker upp när simuleringen är slutförd. Här kan man skriva ut statistik för simuleringen, i trainmenu kan man skriva ut tidtabellen, söka efter ett specifikt tågid. Söker man specifikt tågid kommer information skrivas ut om tåget beroende på log-level.

I stationsmenyn kan man skriva ut alla stationsnamn, söka efter en specifik station, om stationen man sökte hittades kommer den skriva ut vilka tåg och vilka fordon som finns på stationen.

I “Cart menu” (fordonsmenyn) så kan man skriva ut alla fordon (informationen beror på log-level), söka efter ett specifikt id, om det id’t då hittades kommer man att få information beroende på log-level och då även historik om vart fordonet befunnits

Slutsatser och kommentarer

Ett väldigt lärorikt projekt som jag lade ned mycket tid på, körde fast flera gånger och var ganska svårt att komma igång med. Tycker att beskrivningen av uppgiften var lite väl otydlig så hade gärna sätt att den var tydligare med vad man ska presentera vad och vad som gäller i olika fall. Att förstå hur diskret händelsestyrd simulering var lite svårt att förstå till en början, men efter att ha läst igenom och kört McSmack som följde med projektet så klarnade det mer och mer, lysande sätt att utföra simuleringen på! Som vanligt lossnade det mer och mer under projektets gång om hur saker fungerar och hur det struktureras upp bäst. I det stora hela så var det ändå ett roligt projekt som var roligt att göra då man även var tvungen att planera lite i förväg hur allt skulle läggas upp eftersom att projektet var större än något annat vi gjort hittills.