Missile command rapport

Mikael Bohlin Liljekvist

Mitt projekt strävar efter att vara ett A. Därför redan från början la jag upp projektet efter att kunna implementera alla extra krav. För att skapa en överblick av projektet finns et klassdiagram med i zip filen men också en doxygen genererad dokumentation med grafer för funktionsanrop.

För att göra spelet så skapades en Game klass för att köra gameloopen. Därefter delades gameloopen upp i flera funktioner. Dom olika funktionerna representerar ett steg som görs varje frame. Jag börjar med att hantera input från användaren i HandleInput() som sedan lägger inputen i en lista av en Action (ett knapptryck) och en std::any som innehåller kompletterande information beroende på vilken Action som utfördes. Från början var tanken är göra HandleInput() trådad så den körs utanför gameloopen för input hanteringen inte ska beroende på hastigheten av spelarens grafikkort eller andra limitationer på hastighet. Därefter uppdateras spelet i UpdateGame(). Sen uppdateras skärmobjekt i UpdateScreen(). Till sist skapas en frame som visas för andvändaren i ComposeFrame().
Alla dessa funktioner gör olika saker beroende på vilket State spelet är. I många fall har olika States olika Scene:er. GameScene:en är där alla spel relaterade objekt finns såsom Tower, Metiorite och Missile. Medans i MainMenuScene:en finns bara et MainMenu objekt. Dessa gör att man lätt kan byta vilken scene man vill uppdatera och rita till skärmen utan att allokera nytt minne varje gång man byter State.
En scene består av en vektor av SceneObject. Allt som ritas till skärmen ärver från SceneObject direkt eller indirekt. SceneObject är en abstrakt klass som innehåller update och draw funktioner som används för att uppdatera samt rita till skärmen. Dessa overload:as i alla objekt som ärver från SceneObject.
En spelare representeras av Player objektet och innehåller en GameHud instans. Player klassen innehåller variabler för liv, poäng och om spelaren är död eller inte. GameHud läggs sedan till som en pekare i GameScene:en.
WaveMngr är en statisk klass som skapar motståndare samt håller koll på waves. ScoreFile är en annan statisk klass som läser och skriver till score filen för att spara highscores.

Min implementation av spelet blev överdrivet komplicerad enligt mig. Jag tror att jag skulle kunnat tagit bort alla sprites och dess logik för att göra koden mycket simplare. Däremot eftersom det är väldigt uppdelat så är det lätt att följa koden då funktioner är självförklarande.
Det jag lärde mig mest av var hur jag implementerade Button klassen och hur ett Button objekt ska utföra en action. Det problemet jag stötte på var hur ska jag göra så att en Button klass kan utföra olika actions beroende på vilken sorts knapp det är. Jag testade först att göra en funktion i knappen för varje möjlig användning sen vid konstruktion så väljer man vilken knapptyp det är med ett enum. Jag insåg nästan direkt att den lösningen inte var bra då den snabbt blev komplicerad. Jag tänkte då använda en lambda som skickas in i konstruktorn och sparas som en member variabel. Sedan kan anropas med doAction(). JAg stötte på direkt att lambdas har olika signaturer och istället fick använda std::function för att kunna spara den.

Om jag hade börjat om projektet idag hade jag nog ändrat väldigt lite. Jag hade som jag sa försökt simplifiera projektet genom att dra ner på klasser. Jag hade även delat upp större funktioner som i slutenden blev för komplexa. Jag hade nog även planerat lite mindre då eller göra planen mer flexibel. Jag la nu väldigt mycket tid på att planera sen uppstod något problem så jag fick ändra planen rätt många gånger. Det jag inte hade ändrat var hur klass hierarkin ser ut. Just nu är den enligt mig lätt förstod. Däremot skulle fler kommentarer behövas.

Överlag tycker jag att projektet har gått bra. Det har varit väldigt få problem och oftast har det varit problem med hur jag ska namnge saker. Det har även varit roligt. Jag tror jag kanske fortsätter utveckla projektet lite på fritiden utanför kursen. Vid intresse kommer jag göra mitt github repo för koden publikt efter inlämning deadline. Länken till repot är https://github.com/liljekvist/Missile-Command.