CPROG Rapport för Programmeringsprojektet

[Gruppnummer: 20]
[Gruppmedlemmar: (Daniel Kårvall 9510318497)]

Programmeringsprojektet är uppbyggt som sådant att man kan skapa spelare som påverkas av vinkeln av vad man siktar för att kunna skjuta ut skott. Det finns också en klass som kallas för boulder, kanske borde omnämnas till EnemyObject eller liknande som kan träffas. Spelaren har även ett antal liv, den får inte träffas av Boulder för många gånger. Spelarens hastighet och hp sätts av speltillverkaren. Även Boulder(EnemyObject:s) hastighet och startposition samt slutposition sätts i förhållande till var objektet som skall skjutas ned startar och slutar. Man kan även lägga till olika textrutor i början av spelet som informerar spelaren om vad som komma skall. Sökvägen sker genom namnyrmden constants och ger en sökväg till spelets resurser. Spelaren förväntas röra sig genom ”A” och ”D” till vänster och höger men kan ändras till att röra sig uppåt och nedåt genom en enkel förädring. Det finns även en ObjectSpawner som spawnar objekt av av en viss typ i ett visst intervall.

Den ifyllda Rapportmallen lämnas in tillsammans med Programmeringsprojektet. Spara rapporten som en PDF med namnet CPROG_RAPPORT_GRUPP_NR.pdf (där NR är gruppnumret).

1. Beskrivning

Detta är en spelmotor som du kan bestämma hastigheten av olika objekt och dess position. Du kan även bestämma en startposition för vissa objekt och kalkulera dess bana genom spelet genom trigonometri. Spelaren är kontrollerbar i formen att du kan bestämma dess hastighet och bestämma hur karaktären kan röra sig. Enkelt tillägg av en föreändring i spelmotorn kan göra att karaktären kan röra sig upp och ned istället för bara horisontellt.

2. Instruktion för att bygga och testa

Olika TextBoxes: som görs i spelet är generellt menade att visa på start- eller slutfas av spelet. Det finns även en möjlighet att lägga till spelarens HP(liv) och placera där man önskar. Man kan röra objekt(Boulder) vart man vill och bestämma dess hastighet, start och slutposition. En spelare kommer alltid vara kapabel till att skjuta ned dessa objekt. En spelare kan sikta och skjuta på ett objekt med en projektil av en viss hastighet.

3. Krav på den Generella Delen(Spelmotorn)

Programmet ska kodas i C++ och använda grafikbiblioteket SDL2:

Jag har använt c++ och SDL2.

Objektorienterad programmering ska användas: programmet ska vara uppdelat i klasser med användning av oo-tekniker som inkapsling, arv och polymorfism.

Spelmotorn bygger på de krav som ställs på klasshierarkier.

Tillämpningsprogrammeraren ska skyddas mot att göra svårupptäckta fel som att använda värdesemantik för objekt av polymorfa klasser.

Programmet stoppar tilldelning och kopiering av polymorfa klasser.

Det ska finnas en gemensam basklass (säg Sprite) för alla figurer, denna basklass ska vara förberedd för att vara en rotklass i en klasshierarki (om tillämpningsprogrammet önskar göra subklasser till den).

Alla objekt i programmet har en gemensam basklass dvs. ”Component”.

Inkapsling: datamedlemmar ska vara privata om inte väldigt speciella skäl föreligger.

Protected används i klassen projectile då både Shot och Boulder delar många av de trigonometriska uträkningar som krävs.

Det ska inte finnas något minnesläckage, du ska se till att dynamiskt allokerat minne städas bort.

Det finns inget minnesläckage såvitt jag känner till.

Spelmotorn ska kunna ta emot input (tangentbordshändelser, mushändelser) och reagera på dem enligt tillämpningsprogrammets önskemål, eller vidarebefordra dem till tillämpningens objekt.

Inputen kan enkelt ändras genom att ändra knapparna som get olika händelser i spelet via spelmotorn.

Enkel kollisionsdetektering: man ska kunna kolla om den omgivande rektangeln för en Sprite har kolliderat med den omgivande rektangeln för en annan Sprite.

Kollissionsdetektering existerer och kan ändras vid behov i spelmotorn.

Programmet ska gå att kompilera och vara körbart på en dator under både Linux och MS Windows (alltså inga plattformspecifika konstruktioner) med SDL2 och SDL2_ttf, SDL2_image och SDL2_mixer (*Om man vill programmera med andra bibliotek kan man få dispens från detta krav men det kräver en överenskommelse med kursansvarige*)

Programmet går att kompilera på windows, har inte prövat Linux.

4. Krav på den Specifika Delen(Spelet som använder sig av Spelmotorn)

4.1. [Ja/Nej/Delvis] Spelet simulerar en värld som innehåller olika typer av visuella objekt. Objekten har olika beteenden och rör sig i världen och agerar på olika sätt när de möter andra objekt.
Kommentar:

Spelet innehåller olika typer av visuella objekt. Spelet blir progressivt svårare med tiden.

4.2. [Ja/Nej/Delvis] Det finns minst två olika typer av objekt, och det finns flera instanser av minst ett av dessa objekt.

Det finns flera typer av objekt och mutipla instanser av dessa.

4.3. [Ja/Nej/Delvis] Figurerna kan röra sig över skärmen.

Figurerna kan röra sig över skärmen.

4.4. [Ja/Nej/Delvis] Världen (spelplanen) är tillräckligt stor för att den som spelar skall uppleva att figurerna förflyttar sig i världen.

Spelaren kan se figurerna röra sig.

4.5. [Ja/Nej/Delvis] En spelare kan styra en figur, med tangentbordet eller med musen.

Spelaren kan röra sig och skjuta ned olika objekt.

4.6. [Ja/Nej/Delvis] Det händer olika saker när objekten möter varandra, de påverkar varandra på något sätt.

Objekten interagerar beroende på vilket objekt som möter vilket objekt.