CPROG Rapport för Programmeringsprojektet

[Gruppnummer: 02]
[Gruppmedlemmar: Alexander Herder, 20030706-0236]

Skriv en kortfattad instruktion för hur programmeringsprojektet skall byggas och testas, vilka krav som måste vara uppfyllda, sökvägar till resursfiler(bildfiler/ljudfiler/typsnitt mm), samt vad en spelare förväntas göra i spelet, hur figurernas rörelser kontrolleras, mm.
Om avsteg gjorts från kraven på Filstruktur, så måste också detta motiveras och beskrivas i rapporten.

Fyll i 'check-listan', så att du visar att du tagit hänsyn till respektive krav, skriv också en kort kommentar om på vilket sätt du/gruppen anser att kravet tillgodosetts, och/eller var i koden kravet uppfylls.

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

Spelet är en 2D-shooter, lik det gamla spelet “Space Invaders”. Man spelar som ett rymdskepp som ska försvara jorden från fiende-aliens.

Spelloopen går såhär:

Spelaren spawnas i mitten av skärmen. Man rör sig med WASD (W = upp, A = vänster, S = ner, D = höger). Rör man på musen så följer rymdskeppets nos musens riktning. För att kunna skjuta så klickar man på musens vänsterpekare. Då skjuts en “bullet”-sprite ut från nosen av skeppet, och åker i den riktning musen pekade åt under tiden man sköt sin bullet.

Spelet börjar direkt med massvis av fiender som åker uppifrån och nedåt, i riktning mot jorden, som är placerad på botten av skärmen. Om en fiende träffar jorden så försvinner (dör) fienden men jorden tar skada. Ditt uppdrag som spelare är att skjuta fienderna så att så få som möjligt träffar jorden. **Spelet går alltså ut på att du som spelare, men även jorden, ska överleva så länge som möjligt.**

Om du skjuter mot jorden så tar jorden även skada av dina egna bullets.

Om du krockar in i en fiende så tar både du och fienden skada, din spelare “bouncear” även ifrån fienden en kort stund.

Om du krockar in i jorden så tar även du och jorden båda skada. Samma bounce-beteende finns när man krockar in i jorden också.

2. Instruktion för att bygga och testa

Programmeringsprojektet kompileras med “make” i terminalen, och körs med “./build/debug/play” efter kompileringen är gjord.

Datorn som kör spelet måste ha SDL3 installerat, och kompilatorn som bör användas är GNU, kompilerat med g++. Det är med den som spelet är testat.

Resursfilerna finns i ./resources/ - uppdelat på objekttyp (t.ex. ./resources/player/textures/). Hittills är de enda resurser som används bildtexturer, planer finns att lägga till ljud, vilket är varför det finns en “audio”-mapp i varje underkatalog.

3. Krav på den Generella Delen(Spelmotorn)

3.1. [Ja/Nej/Delvis] Programmet kodas i C++ och grafikbiblioteket SDL används.
Kommentar: Ja.

3.2. [Ja/Nej/Delvis] Objektorienterad programmering används, dvs. programmet är uppdelat i klasser och använder av oo-tekniker som inkapsling, arv och polymorfism.
Kommentar: Ja.

3.3. [Ja/Nej/Delvis] Tillämpningsprogrammeraren skyddas mot att använda värdesemantik för objekt av polymorfa klasser.
Kommentar: Ja, copy-konstruktorn och tilldelningsoperatorn för Sprite-klassen är båda deklarerade som delete och privata.

3.4. [Ja/Nej/Delvis] Det finns en gemensam basklass för alla figurer(rörliga objekt), och denna basklass är förberedd för att vara en rotklass i en klasshierarki.
Kommentar: Ja. Sprite-klassen är rotklassen för alla figurer och för alla rörliga objekt.

3.5. [Ja/Nej/Delvis] Inkapsling: datamedlemmar är privata, om inte ange skäl.
Kommentar: Ja. Alla datamedlemmar är privata.

3.6. [Ja/Nej/Delvis] Det finns inte något minnesläckage, dvs. jag har testat och försökt se till att dynamiskt allokerat minne städas bort.
Kommentar: Ja. Jag använder std::shared_ptr för automatisk minneshantering av objekt, samt ser till att SDL-resurser städas bort korrekt i GameEngine-destruktorn och SDL_Quit anropas vid avslut.

3.7. [Ja/Nej/Delvis] Spelmotorn kan ta emot input (tangentbordshändelser, mushändelser) och reagera på dem enligt tillämpningsprogrammets önskemål, eller vidarebefordra dem till tillämpningens objekt.
Kommentar: Ja. Spelarens Sprite kan röra sig runt hela “spelplanen” baserad på input + skjuta bullets baserad på input.

3.8. [Ja/Nej/Delvis] Spelmotorn har stöd för kollisionsdetektering: dvs. det går att kolla om en Sprite har kolliderat med en annan Sprite.
Kommentar: Ja. I basklassen Sprite finns det en implementation som använder SDL_HasRectInterSectionFloat() för att kontrollera kollisioner mellan två objekts omslutande rektanglar.

För mitt specifika spel har jag dock utnyttjat polymorfism och skapat en subklass, "RoundSprite", som alla rörliga objekt ärver från. Denna klass överskuggar collidedWith()-funktionen i Sprite-basklassen. Detta eftersom jag ville ha kollisionsdetektering baserat på cirklar istället för rektanglar eftersom cirklar är mer precisa och lämpliga för Earth, Player, Bullet och Enemy-klassernas texturer/ beteende. Cirkelbaserad kollisionsdetektering är rentav nödvändigt för att Earth-objektet inte ska få helt fel kollisionsbeteende.

3.9. [Ja/Nej/Delvis] Programmet är kompilerbart och körbart på en dator under både Mac, Linux och MS Windows (alltså inga plattformspecifika konstruktioner) med SDL och SDL_ttf, SDL_image.
Kommentar: Ja. Inga plattformspecifika konstruktioner.

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: Ja.

4.2. [Ja/Nej/Delvis] Det finns minst två olika typer av objekt, och det finns flera instanser av minst ett av dessa objekt.
Kommentar: Ja. Det finns 5 olika typer av objekt: Player, Earth, Enemy, Bullet och Background. Flera instanser finns hos Enemy och Bullet.

4.3. [Ja/Nej/Delvis] Figurerna kan röra sig över skärmen.
Kommentar: Ja. Player, Enemy och Bullet 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.
Kommentar: Ja, världen är 1920x1080 pixlar stor.

4.5. [Ja/Nej/Delvis] En spelare kan styra en figur, med tangentbordet eller med musen.
Kommentar: Ja. Spelaren styrs med WASD för att förflytta sig + vänster musklick för att skjuta.

4.6. [Ja/Nej/Delvis] Det händer olika saker när objekten möter varandra, de påverkar varandra på något sätt.
Kommentar: Ja. Exempelvis studsar spelaren bort vid kollision med fienden, och jorden tar skada vid kollision med både spelare och fiender.