CPROG Rapport för Programmeringsprojektet

[Gruppnummer: 63]
[Gruppmedlemmar: Adam Lindroth, 19970722-5711]

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 går ut på att styra en gubbe och försöka undvika att bli träffad av lådor som studsar över skärmen. Gubben kan skjuta kulor som förstör lådorna. Om en kula träffar och förstör en låda ökar poängräknaren med 10. Om kulan missar minskar den med 1 (om man har mer än 0 poäng).

Gubben styrs med tangenterna ’A’ och ’D’
Skjuta gör man genom att klicka med vänstra musknappen. En kula kommer då skapas på gubbens huvud och färdas i den riktning musklicket skedde.

2. Instruktion för att bygga och testa

Spelet byggs via ’make’ på det sättet som vi fått lära oss på föreläsningarna. Makefile är bifogad.
Alla ytterligare resurser (bilder och typsnitt) ligger under respektive mapp i resources.
Jag har delat upp mina headerfiler i submappar. En för Sprite-klasshierarkin och en för min implementation.

3. Krav på den Generella Delen(Spelmotorn)

3.1. [Ja/Nej/Delvis] Programmet kodas i C++ och grafikbiblioteket SDL2 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. Samtliga används i Sprite-hierarkin.

3.3. [Ja/Nej/Delvis] Tillämpningsprogrammeraren skyddas mot att använda värdesemantik för objekt av polymorfa klasser.
Kommentar: Ja. Copy-konstruktorn och tilldelningsoperatorn är borttagna i samtliga basklasser.

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:
Om ni med rörlig menar föränderlig, så ja. Jag har delat upp Sprite-klassen i två subklasser. MovableSprite och ImmovableSprite. MovableSprite kan röra sig över skärmen medans ImmovableSprite inte kan det. Båda har dock en tick-metod som anropas och får dem att reagera med omvärlden.

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

3.6. [Ja/Nej/Delvis] Det finns inte något minnesläckage, dvs. jag har testat och sett till att dynamiskt allokerat minne städas bort.
Kommentar: Ja.

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
Sättet jag har implementerat detta på är att alla spelbara sprites har en metod tar emot en SDL_SCANCODE. I denna metod finns sedan ett switch-statement där man kan skriva hur spelaren ska reagera beroende på vilken knapp som tryckts ner.

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
Sättet jag implementerade detta på var att jag skapade en metod, collision(), i huvudloopen. Den är tänkt at anropas från sprite objektens tick-metod. Om kollision upptäcks returnerar metoden det objekt som anroparen kolliderat med. Anroparen kan sedan reagera utifrån typen på det andra objektet.

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 2 och SDL2_ttf, SDL2_image och SDL2_mixer.
Kommentar: Ja

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
Objekt av klassen Box ändrar riktning när de kolliderar med en annan box.
Bullet förstör en box vid kollidering.
Player dör när den kolliderar med en box.

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. Gäller för både Bullet och Box.

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

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.

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

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