CPROG Rapport för Programmeringsprojektet

[Gruppnummer: 78]
[Gruppmedlemmar: Victor Lejon 200210315552]

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

Programmet består av huvudklassen ”GameEngine” som sköter allt som har att göra med rendering och fönster samt för att kalla på uppdaterings funktionerna för alla objekt. GameEngine tar också hand om events såsom knapptryckningar. Klassen ”Sprite” är en abstrakt klass (purely virtual) och ska användas av programmeraren för att skapa objekten som ska renderas ut i fönstret. Det finns också en ”TextureLoader” klass med en statisk metod som ser till att ladda in texture filerna för varje sprite när de instanseras.

Alla resursfiler finns i mappen resources, då jag bara använder bilder ligger alla i mappen images. Det ska finnas sex stycken bmp filer. SDL biblioteket importeras med ’#include SDL2/SDL_image.h’.

2. Instruktion för att bygga och testa

Spelet byggs med makefilen och kommandot ”make”.

För att starta spelet kör man applikationen play som ligger i build/debug. Det är en simpel version av atari breakout. Bollen börjar röra sig när spelaren trycker på ”M” knappen och man använder sig av ”A” och ”D” för att röra sin platform höger respektive vänster. Målet är att slå sönder alla brickor på planen. Inget särskilt händer när man vinner eller förlorar, då får man starta om spelet själv.

Om man vill bygga en egen applikation/spel med spelmotorn skapar man en instans av GameEngine, kör funktionen start som skapar fönster/renderare, sedan skapar man en loop som kallar på medlemsfunktionerna event_handler, update så länge spelet är på. I loopen kan man självklart lägga till fler funktioner som sköter de objekten man skapat i sin implementation. Update funktionen kommer uppdatera alla sprites tillstånd genom att ta hand om alla kollisioner samt köra den abstrakta metoden update som ska implementeras i varje sprite.

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

3.3. [Ja/Nej/Delvis] Tillämpningsprogrammeraren skyddas mot att använda värdesemantik för objekt av polymorfa klasser.
Kommentar: Ja (hoppas jag inte missat något)

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

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 (tangentbordhändelser)

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

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 (testat på windows och 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: 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

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