

CPROG Rapport för Programmeringsprojektet

[Gruppnummer: 03]

[Gruppmedlemmar: William de Try: 0411050677, Sadra YaghoobzadehTari: 0306026311, Isabelle Holloway 0208316349]

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

Projektet består av två tydligt separata delar: en generell spelmotor och ett specifikt spel som använder spelmotorn.

Spelmotorn är skriven i C++ och bygger på SDL. Den tillhandahåller funktionalitet för spelloop, rendering, inputhantering, sprites, kollisioner, text och ljud. Motorn är utformad så att flera spel av samma typ kan skapas utan att spelmotorns kod behöver modifieras.

För att demonstrera spelmotorn har ett arkadspel av typen Breakout implementerats som ett separat tillämpningsprogram.

2. Instruktion för att bygga och testa

Projektet byggs med hjälp av make och kräver att SDL samt biblioteken SDL_image och SDL_ttf är installerade på systemet.

Bygginstruktion

1. Placera projektet enligt den angivna filstrukturen.
2. Säkerställ att SDL, SDL_image och SDL_ttf är installerade.
3. Kör make i projektets rotkatalog.
4. Det färdiga körbara programmet skapas i build/debug/.
5. Kör kommandot ./build/debug/play för att starta spelet

Resurser

Alla resurser, ligger i resources/.

Sökvägar hanteras via den globala konstanten gResPath i Constants.h.

Test och spelinstruktion

Spelet styrs med tangentbord:

- Vänster och höger piltangent eller A/D för att flytta paddeln
- Mellanslag för att skicka ut bollen

Målet är att slå sönder alla bricks/brickor med bollen utan att förlora alla liv.

3. Krav på den Generella Delen(Spelmotorn)

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

Ja, Spelmotorn är implementerad i C++ och använder SDL tillsammans med SDL_image och SDL_ttf.

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, Programmet är uppdelat i klasser och använder arv, polymorfism och inkapsling, bland annat genom en gemensam Sprite-hierarki.

3.3. [Ja/Nej/Delvis] Tillämpningsprogrammeraren skyddas mot att använda värdesemantik för objekt av polymorfa klasser.

Kommentar:

Ja, Copy constructor och assignment operator är borttagna för klasser som används polymorft, exempelvis Sprite och dess subklasser.

3.4. [Ja/Nej/Delvis] Det finns en gemensam basklass för alla figurer(rörliga objekt), och denna basklass är förberedd för att vara en rotclass i en klasshierarki.

Kommentar:

Ja, Klassen Sprite fungerar som basklass för alla visuella objekt och är avsedd att vara rotclass i hierarkin.

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

Kommentar:

Ja, Datamedlemmar är privata eller skyddade och nås via publika medlemsfunktioner.

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, Dynamiskt allokerade objekt städas bort i destruktörer. Projektet är granskat för att undvika minnesläckor.

- 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, Tangentbord och mus hanteras via en egen InputManager som används av spelmotorn och spelet.

- 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, Enkel kollisionsdetektering) är implementerad.

- 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, Programmet är byggbart och körbart på Linux, MacOS och Windows med SDL och Makefile. En av oss i projektet satt på Linux, en satt på MacOS och en satt på Windows, det kompilerade och kördes på alla operativsystem.

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, Spelet innehåller paddel, boll och bricks med olika beteenden.

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 flera instanser av bricks samt separata objekt för paddel och boll.

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

Kommentar:

Ja, Paddeln och bollen rör sig kontinuerligt över spelplanen baserat på input.

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, Spelplanen motsvarar fönstrets storlek och ger tydlig rörelse i världen.

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

Kommentar:

Ja, Spelaren styr paddeln via tangentbordet.

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

Kommentar:

Ja, Bollen studsar mot paddeln och väggar, och bricks/brickor tas bort vid kollision.

