CPROG Rapport för Programmeringsprojektet

[Gruppnummer: 03]
[Gruppmedlemmar: Mårten Engberg 921212-3757, Erik Gustafsson 990902-4573, Arvin Soltanpour 930102-0799]

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

Vi har skapat ett bibliotek ämnat för att göra simpla plattformare i stil med Mario och Donkey Kong. Detta bibliotek består av följande klasser:
-GameEngine: själva kärnan för biblioteket.
-System: Sköter en del av bakgrundsarbetet.
-Sprite: Basklass för objekt i spelvärlden.
-Player: Subklass till Sprite som tillhandahåller spelarkaraktärer.
-Enemy: Subklass till Sprite som tillhandahåller fiender.
-Ground: Subklass till Sprite som tillhandahåller plattformar som karaktärer kan stå på.
-Goal: Subklass till Sprite som tillhandahåller målet på en given nivå.

2. Instruktion för att bygga och testa

Projektet nyttjar make och kompileras som ett vanligt make-projekt, i vårat fall i VS-Code. Respath-variabeln ska fungera utan att behöva modifieras, i alla fall på alla Windows-datorer vi testat det på. Det kan dock vara så att man på Mac eller Linux behöver använda en plattformsspecifik make-fil. Själva spelet fungerar så att man använder piltangenterna för att styra sin karaktär. Uppåtpil får karaktären att hoppa. Kolliderar man med en fiende så förlorar man omedelbart medan vinst uppnås genom att kollidera med myntet. För att nå myntet hoppar man mellan plattformarna. För att karaktären ska stanna på en plattform behöver mer än hälften av karaktärens bredd befinna sig över den.

3. Krav på den Generella Delen(Spelmotorn)

3.1. [Ja] Programmet kodas i C++ och grafikbiblioteket SDL2 används.
Kommentar: Hela projektet är kodat i C++ och SDL2 används flitigt.

3.2. [Ja] Objektorienterad programmering används, dvs. programmet är uppdelat i klasser och använder av oo-tekniker som inkapsling, arv och polymorfism.
Kommentar: Vi har försökt använda oo så mycket som det känts rimligt med projektets storlek. En punkt där vi antagligen kunnat göra mer klassuppdelning är med vår implementation av gravitation. Det sköts för närvarande av Sprite-klassen, men hade kunnat vara sin egen klass om vi haft tid att lösa det.

3.3. [Ja] Tillämpningsprogrammeraren skyddas mot att använda värdesemantik för objekt av polymorfa klasser.
Kommentar: Assignment-operatorn och copy-konstruktorerna är förbjudna i Sprite-hierarkin och vi nyttjar getInstance-metoder för att instansiera sådana klasser. Konstruktorerna är inte heller publika.

3.4. [Ja] Det finns en gemensam basklass för alla figurer(rörliga objekt), och denna basklass är förberedd för att vara en rotklass i en klasshierarki.
Kommentar: Klassen Sprite är vår basklass för figurer. Biblioteket har också fyra inbyggda subklasser i form av Player, Ground, Goal och Enemy med utrymme att skapa fler vid behov.

3.5. [Delvis] Inkapsling: datamedlemmar är privata, om inte ange skäl.
Kommentar: Det finns ett fåtal undantag. Ett par av datastrukturerna med sprites i GameEngine behöver vara publika med vår implementation av kollisionshantering då det sker i varje sprite enskilt som måste testa sin kollision mot alla andra. En alternativ lösning på detta hade varit att sköta kollisionshanteringen i själva spelmotorn i stället vilket hade undvikit detta.

3.6. [Ja] Det finns inte något minnesläckage, dvs. jag har testat och sett till att dynamiskt allokerat minne städas bort.
Kommentar: Vi har testat att alla destruktorer kallas när ett objekt tas bort från spelmotorn.

3.7. [Ja] Spelmotorn kan ta emot input (tangentbordshändelser, mushändelser) och reagera på dem enligt tillämpningsprogrammets önskemål, eller vidarebefordra dem till tillämpningens objekt.
Kommentar: Exempelvis använder vi detta i vår tillämpning för att styra spelarkaraktären med piltangenterna.

3.8. [Ja] Spelmotorn har stöd för kollisionsdetektering: dvs. det går att kolla om en Sprite har kolliderat med en annan Sprite.
Kommentar: Kollisionsdetekteringen används både för win/lose-conditions beroende på vilken kollision som sker samt för gravitation.

3.9. [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: Vi använder till vår vetskap inget plattformsspecifikt för Windows i vårt projekt, men vi har bara tillgång till den plattformen och kan ej verifiera det. Det enda potentiella problemet kan vara att make-filen kan vara plattformsspecifik.

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

4.1. [Ja] 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: Spelaren kan hoppa upp på och stå på plattformarna, dör när de kolliderar med fienderna och vinner när de kolliderar med myntet. Fienderna rör sig enligt en förutbestämd bana baserat på plattformen de står på.

4.2. [Ja] Det finns minst två olika typer av objekt, och det finns flera instanser av minst ett av dessa objekt.
Kommentar: Vi har fyra typer av objekt – spelare, mål, fiender och plattformar med multipla av de två senare i vår tillämpning.

4.3. [Ja] Figurerna kan röra sig över skärmen.
Kommentar: Fiender och Spelaren har rörelser där fiendernas är förbestämt och spelaren är kontrollerad via tangentbordet.

4.4. [Ja] Världen (spelplanen) är tillräckligt stor för att den som spelar skall uppleva att figurerna förflyttar sig i världen.
Kommentar: Vi har utrymme för flera plattformar och möjligheten att hoppa mellan dem.

4.5. [Ja] En spelare kan styra en figur, med tangentbordet eller med musen.
Kommentar: Spelarkaraktären styrs med piltangenterna.

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

Kommentar: Spelaren kan hoppa upp på och stå på plattformarna, dör när de kolliderar med fienderna och vinner när de kolliderar med myntet.