

CPROG Rapport för Programmeringsprojektet

[Gruppnummer: 24]

[Gruppmedlemmar: Mattias Liska (040810-5476), Isak Flores (030829-3331)]

1. Beskrivning

När spelet startas kommer det att finnas två knappar i mitten av skärmen. Med muspekaren tryck på EXIT för att stänga spelet (alternativt ESC) eller START för att köra spelet. Din spelare kommer att finnas i mitten av fönstret när spelet startas. Rör på spelaren med W A S D tangenterna. Om spelaren kolliderar med en vägg går det inte att fortsätta i den riktningen men spelaren kan dock röra sig längs väggen. Använd piltangenterna för att skjuta en kula i pilens riktning. Fienderna kommer att laddas in kontinuerligt från sidorna av fönstret och röra sig mot spelarens position. Om en fiende kolliderar med spelaren förlorar du och spelfiguren försvinner. När en kula träffar en fiende försvinner de och spelaren får ett poäng som syns på poängräknaren i det vänstra hörnet. När spelaren kommer upp i 10 poäng vinner de. Om en spelare skjuter på en vägg kommer väggen progressivt att gå sönder, om spelaren skjuter väggen fem gånger försvinner den. Om spelaren dör eller vinner kan de starta om spelet genom att trycka på R.

2. Instruktion för att bygga och testa

För att kunna köra programmet måste du ha installerat c++, g++, SDL3 samt make. Öppna projektet i valfri IDE eller kommandotolken, om du är i en IDE öppna en terminal och navigera till roten av projektmappen. Beroende på vilket operativsystem som används kan du behöva ändra filerna i vscode mappen och make filen. För att kunna köra programmet måste det först kompileras i terminalen med kommandot 'make'. Efter kompilering går programmet att köras med kommandot './build/debug/play'. Om det inte skulle fungera kan du behöva lägga till citattecken, använd istället kommandot './build/debug/play'

3. Krav på den Generella Delen(Spelmotorn)

3.1. [Ja] Programmet kodas i C++ och grafikbiblioteket SDL används.

Kommentar: Vi har enbart använt C++ i samband med SDL, SDL Image samt SDL TTF.

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 använt inkapsling, arv samt polymorfism i projektet. Klasserna för spelmotorn och spelen överlappar ej för att underlätta för tillämpningsprogrammeraren. Sprites som används i spelet utgår alla från en basklass som alla sprites ärver från samt alla konstruktörer för sprites kan antingen ta några argument eller använda sig av default konstruktorn.

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

Kommentar: I basklassen för alla sprites ser vi till att radera copy-konstruktorn samt

tilldelning operationer med kommandot 'delete' för att tillämpningsprogrammeraren inte av misstag ska kunna kopiera sprites objekt av något slag.

- 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: Alla våra sprites ärver från en specifik basklass, Sprites, och har några metoder som spelprogrammeraren kan använda sig av. Dessutom har vi andra spelmotorklasser, Label samt Button, som både ärver från Sprites.
- 3.5. [Ja] Inkapsling: datamedlemmar är privata, om inte ange skäl.
Kommentar: Alla medlemmar som bör vara privata eller som andra objekt inte bör ha åtkomst till är privata.
- 3.6. [Ja] Det finns inte något minnesläckage, dvs. jag har testat och försökt se till att dynamiskt allokerat minne städas bort.
Kommentar: Allt dynamiskt allokerat minne städas bort av programmet under körning, vid avstängningen eller när programmet kraschar.
- 3.7. [Ja] Spelmotorn kan ta emot input (tangentbordshändelser, mushändelser) och reagera på dem enligt tillämpningsprogrammets önskemål, eller vidarebefordra dem till tillämpningens objekt.
Kommentar: Spelmotorn tar emot input genom tangentbordshändelser och mushändelser för att röra på spelaren och interagera med spelet och skickar input vidare till de klasser som använder sig av input.
- 3.8. [Ja] Spelmotorn har stöd för kollisionsdetektering: dvs. det går att kolla om en Sprite har kolliderat med en annan Sprite.
Kommentar: Det finns stöd för kollisionsdetektering mellan olika instanser av objekt som är sprites.
- 3.9. [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: Vi har ej haft möjligheten att testa programmet på något annat operativsystem än Windows. Programmet bör fortfarande fungera då det finns ingen operativsystemspecifik kod, dock man må behöva ändra i make-filen beroende på vilket operativsystem man använder (gällande linker-flags odyl.)

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. Objektet har olika beteenden och rör sig i världen och agerar på olika sätt när de möter andra objekt.

Kommentar: Spelet består primärt av en spelfigur samt många olika fiendefigurer som spelaren måste besegra. Spelaren kan röra sig fritt på skärmen förutom när de går in i en vägg eller nuddar en fiende vilket gör att de dör. Fiende reagerar dessutom på spelarens skott och dör om de nuddar dem.

- 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 primärt fyra olika typer av instanser. En spelare, flera fiende som kontinuerligt laddas in, skott som spelaren kan avfira samt väggar.

- 4.3. [Ja] Figurerna kan röra sig över skärmen.

Kommentar: Spelaren kan röra sin figur med hjälp av W A S D tangenterna och röra sig fritt över X samt Y axeln.

- 4.4. [Ja] Världen (spelplanen) är tillräckligt stor för att den som spelar skall uppleva att figurerna förflyttar sig i världen.

Kommentar: Ja, spelaren kan tydligt se att både spelfiguren samt fienderna rör sig på skärmen.

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

Kommentar: Spelaren kan navigera den enkla menyn med musen samt styra på spelfiguren med tangentbordet.

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

Kommentar: Ja, nästan varenda kombination av objektkollision har en unik påverkan på de instanserna av spelobjekten. Exempelvis, en spelare dör när de interagerar med en fiende, väggar kan gå sönder om spelaren skjuter på dem, fiender kan inte gå igenom väggar utan måste gå runt dem.