


CPROG Rapport för Programmeringsprojektet

[Gruppnummer: 67]

[Gruppmedlemmar: Leonardo Gast - 760904-6474

Marcus Bergengren - 930805-5277]

1. Beskrivning

Spelet som vi har utvecklat heter “ Asteroid Kitty” och är ett arkadspel där spelaren kontrollerar en kattfigur (Kitty) som måste undvika att kollidera med asteroider. Spelaren styr Kitty med antingen tangentbordet eller musen. Asteroiderna faller från toppen av skärmen, och deras fallhastighet ökar över tid. Spelet är byggt med en objektorienterad ansats och använder SDL2 för grafik och ljud.

2. Instruktion för att bygga och testa

1. Se till att C++-kompilator och SDL2-biblioteket (inklusive SDL2_image, SDL2_ttf och SDL2_mixer) är installerade.
2. Kör make från roten av projektet för att bygga spelet.
3. Starta spelet genom att köra den exekverbara filen med ./build/debug/play

3. Krav på den Generella Delen(Spelmotorn)

- 3.1. [Ja] Programmet kodas i C++ och grafikbiblioteket SDL2 används.
Kommentar: Vi har använt C++ och SDL2 som specificerat. Koden för SDL2-initialisering och -hantering finns i System.cpp.
- 3.2. [Ja] Objektorienterad programmering används, dvs. programmet är uppdelat i klasser och använder av oo-tekniker som inkapsling, arv och polymorfism.
Kommentar: Spelet är uppdelat i klasser såsom GameEngine, Sprite, Kitty, och Asteroid.
- 3.3. [Ja] Tillämpningsprogrammeraren skyddas mot att använda värdesemantik för objekt av polymorfa klasser.
Kommentar: För att hantera objekt av polymorfa klasser använder vi smarta pekare, såsom std::shared_ptr och std::weak_ptr. Detta säkerställer en säker hantering av minne och resursläckage.
- 3.4. [Ja] Det finns en gemensam basklass för alla figurer(rörliga objekt), och denna basklass är förberedd för att vara en rotclass i en klasshierarki.
Kommentar: Sprite är basklassen för alla figurer, från vilken Kitty och Asteroid är härledda.

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

Kommentar: Alla datamedlemmar i våra klasser är privata. Detta följer principen om inkapsling och hjälper till att skydda objektets tillstånd från oönskad åtkomst eller modifiering.

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

Kommentar: Vi har varit noggranna med att säkerställa att allt dynamiskt allokerat minne frigörs korrekt. Genom att använda verktyg som "valgrind" har vi verifierat att vårt program inte innehåller minnesläckage.

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 är utformad för att effektivt hantera användarinput såsom tangentbords- och mushändelser, vilket gör det möjligt för spelaren att interagera med spelet på ett intuitivt sätt.

3.8. [Ja] Spelmotorn har stöd för kollisionsdetektering: dvs. det går att kolla om en Sprite har kolliderat med en annan Sprite.

Kommentar: Kollisionsdetektering är en kritisk del av spelet och har implementerats för att avgöra när Sprite-objekt, som Kitty och asteroiderna, kolliderar med varandra.

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 har säkerställt att spelet är kompilerbart och fungerar på våra operativsystem. Här är våra **c_cpp_properties.json** information:

```
{
  "configurations": [
    {
      "name": "Mac",
      "includePath": [
        "${workspaceFolder}/**",
        "${workspaceFolder}/include",
        "/usr/local/include/SDL2",
        "/usr/local/include",
        "${workspaceFolder}/include"
      ],
      "compilerPath": "/usr/local/bin/g++-13",
      "intelliSenseMode": "macos-clang-x64",
      "cStandard": "c11",
      "cppStandard": "c++17",
      "configurationProvider": "ms-vscode.makefile-tools"
    }
  ],
  "version": 4
}
```

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 erbjuder en rik och dynamisk värld där objekt som Kitty och asteroider interagerar och rör sig på olika sätt, vilket skapar en fängslande spelupplevelse.

- 4.2. [Ja] Det finns minst två olika typer av objekt, och det finns flera instanser av minst ett av dessa objekt.

Kommentar: Spelet innehåller flera objekttyper, inklusive spelkaraktären Kitty och flera instanser av asteroider. Detta bidrar till speldynamiken och utmaningen.

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

Kommentar: Kitty kan röra sig horisontellt med tangentbordet och fritt med musen. Asteroiderna har en programmerad fallrörelse; detta är en central del av spelupplevelsen och utmaningen.

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

Kommentar: Spelplanen är utformad för att ge spelaren en känsla av rymd och rörelse, med tillräckligt utrymme för att navigera och undvika faror.

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

Kommentar: Spelaren har full kontroll över Kitty genom antingen tangentbordet eller musen, vilket ger en responsiv och engagerande spelupplevelse.

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

Kommentar: När objekt möter varandra, till exempel när Kitty kolliderar med en asteroid, påverkas spelet och dess tillstånd på ett meningsfullt sätt, vilket är avgörande för spelupplevelsen.