CPROG Rapport för Programmeringsprojektet

[Gruppnummer: 16]
[Gruppmedlemmar: Jeremy Vaudan – 19990117-4533]

1. Beskrivning

Jag skapade endast ett spel för att testa att det går men det går att skapa andra också men då måste programmeraren skapa egna spelklasser som är subklasser av sprite.
I spelet jag gjort som kallas FirstGame så styrs spelaren via piltangenterna och kan skjuta skott via mellansteg/space. Fienderna skapas kontinuerligt över tid ovanför toppen av skärmen och rör sig sakta nedåt. Spelarens mål att döda fienderna utan att bli nuddad för att få poäng. Blir spelaren rörd av en fiende så förlorar hen och en stor text Game Over visas och spelaren tas bort. Målet med spelet är att få så mycket poäng som möjligt utan att dö.

2. Instruktion för att bygga och testa

Sökvägar

Resursfilerna ligger i resources mappen inom projektet och alla sprites använder sig av en constants.h fil för att hitta resources mappen. Allt programmeraren behöver göra när hen skapar ett objekt av en subklass av sprite, t.ex. spelaren, är att ange /images samt filnamnet på bilden. T.ex. /images/player.png.

Spelet

Spelet byggs och testas genom att skapa en ny fil som t.ex. heter FirstGame.cpp som innehåller en main metod. Det får endast finnas en main metod åt gången så om man vill göra fler spel får man flytta de andra spelen från src till exempelfiler. Inuti FirstGame.cpp så inkluderar man alla spelklasser som man vill använda sig av. T.e.x #include ”Player.h”. När man väl inkluderat alla spelklasser man vill använda så måste man använda sig av namespace MyGameEngine. Det är så man använder sig av spelmotorklasserna. Sedan inuti main så skapar man objekt av spelklasserna och sedan lägger till de i globalGameEngine via dess add metod. När alla objekt är inlagda i globalGameEngine så körs spelet genom att kalla globalGameEngine.run(). Detta måste vara sist i main precis ovanför return 0; Se bild nedan för exempel.

```cpp
#include "Label.h"
#include "Player.h"
#include "EnemyTwo.h"
#include "EnemySpawner.h"
#include "Bullet.h"
#include "Background.h"
#include "ScoreLabel.h"
#include <iostream>

using namespace MyGameEngine;

int main(int argc, char **argv)
{
    // Skapar och lägger till bakgrunden
    Background *bg1 = Background::getInstance("images/bg.png");
    globalGameEngine.add(bg1);

    // Skapar och lägger till ScoreLabel objekt.
    ScoreLabel *scoreLabel = ScoreLabel::getInstance(50, 50, 48, 48);
    globalGameEngine.add(scoreLabel);

    // Skapar och lägger till Player objekt.
    Player *player = Player::getInstance("images/player.png", 500, 700, 80, 6);
    globalGameEngine.add(player);

    // Skapar och lägger till en EnemySpawner
    EnemySpawner *eSpawner = EnemySpawner::getInstance("images/destroyer.png", 100, -100, 100, 5, 2, 50);
    globalGameEngine.add(eSpawner);

    // Kör session.
    globalGameEngine.run();
    return 0;
}
```

3. Krav på den Generella Delen(Spelmotorn)

3.1. [Ja/Nej/Delvis] Programmet kodas i C++ och grafikbiblioteket SDL2 används.
Kommentar: Ja, alla spelmotor klasser använder sig av SDL, inga spelklasser använder sig av SDL, utan de inkluderar spelmotorklasser istället. Allting är kodat i C++ efter min egen förmåga.

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, har delat upp så spelmotorn består av 3 klasser, GameEngine, System samt Sprite och sedan har Sprite subklasser för alla spelklasser som programmeraren ska ha gjort för att skapa sitt spel.

3.3. [Ja/Nej/Delvis] Tillämpningsprogrammeraren skyddas mot att använda värdesemantik för objekt av polymorfa klasser.
Kommentar: Ja, kopieringskonstruktorn samt tilldelningsoperatorn är förbjudna i Sprite klassen för att förhindra detta.

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, basklassen kallas Sprite och alla spelklasser är subklasser av sprite.

3.5. [Ja/Nej/Delvis] Inkapsling: datamedlemmar är privata, om inte ange skäl.
Kommentar: Alla datamedlemmar som kan vara private är det, vissa är protected i t.ex. Sprite eftersom subklasserna måste komma åt de variablerna.

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. Jag har försökt testa och när objekt tas bort så kallas deras destruktor där minnesläckage tas hand om. Alltså i destruktorn så rensar den surface och texture.

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, subklasser av sprite kan överlagra till exempel rightArrow om programmeraren vill att subklassen ska göra en viss sak när höger pil trycks ner. Detsamma gäller rightArrowUp vilket är om höger pilen släps. Det finns överlagring för alla pilar och mouseDown och mouseUp.

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, i Sprite klassen finns en metod checkKollision som kollar om två sprites har kolliderat. Kollideringen är att de två spritesens SDL_Rectangle jämförs.

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.

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 har en bakgrund, fienden har en viss sprite och spelaren har en annan sprite.

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 en spelare och flera fiender.

4.3. [Ja/Nej/Delvis] Figurerna kan röra sig över skärmen.
Kommentar: Ja, spelaren kan röra sig via piltangenterna och fienderna rör sig av sig själv över tid.

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, fienderna rör sig från toppen av skärmen sakta nedåt och spelaren kan röra sig runt över hela skärmen.

4.5. [Ja/Nej/Delvis] En spelare kan styra en figur, med tangentbordet eller med musen.
Kommentar: Ja, spelaren styrs via piltangenterna.

4.6. [Ja/Nej/Delvis] Det händer olika saker när objekten möter varandra, de påverkar varandra på något sätt.
Kommentar: Ja, när ett skott som skjuts från spelaren och en fiende möter varandra så förstörs båda. Möter fiende en spelare så blir det Game Over och båda fiende och spelaren förstörs.