## Beskrivning

Spelmotorn har byggts upp i Visual Studios Code med SDL2 där motorn är skapad för att användaren ska kunna göra sitt eget co-op-spel. Hierarkin och diverse data lagrade på motorklasserna finns där för att underlätta detta, exempelvis int-variablen “playerNumber”, som finns för att tilldela vilken av spelarna ett objekt tillhör (om denne så önskar). Exempelvis skulle en programmerare kunna använda sig av playerNumber samt dess change- och get-funktioner för att tilldela ett vapen, skada, eller poäng till den önskade spelaren.

Spelet som har skapats med hjälp av motorn har därför endast två spelare och inga fiender eftersom spelarna kör mot varandra. Det huvudsakliga målet med spelet är att det ska simulera en värld med två olika skepp i rymden som ska eliminera den andra där man ska träffa den andra spelaren med skott för att den ska ta skada. Den som först lyckas göra skada på den andra spelaren vinner. Varje spelare har ett varsitt rymdskepp som den kontrollerar via tangentbordet och det spelas därför via LAN, dvs via ett lokalt nätverk. Spelarna behöver befinna sig vid samma tangentbord för att kunna spela tillsammans och varje spelare har olika tangentbordsknappar som de kontrollerar. Spelare A har A och D för att kunna röra sig fram och tillbaka på skärmen och C för att kunna skjuta skott på spelare B. Spelare B har J och L för rörelse samt M för att kunna skjuta på spelare A. Motorn är uppbyggd för att kunna skapa fler spelare än två vilket gör att användaren tekniskt sett även kan ha fyra spelare som kör mot varandra men för detta spel har vi endast två spelare. Rymdskeppen i spelet kontrolleras endast via tangentbordet för att underlätta samspelet mellan spelarna eftersom det endast finns en mus för varje dator men flera knappar på ett tangentbord. När en spelare skjuter mot den andra spelaren tar den skada ifall denna träffas av skottet(bullet) och varje spelare har 10 liv(health). Varje skott som träffar en spelare tar ett liv vilket gör att varje spelare behöver träffas av 10 skott för att förlora. Om en spelare har blivit av med sina 10 liv försvinner denna och den andra spelaren vinner.

## Instruktion för att bygga och testa

I resursfilerna befinner sig “resources” mappen där alla components som programmet använder sig av är lagrade. För att förenkla processen av att byta ut de sprites och diverse visuella komponenter som programmeraren önskar visa upp använder programmet sig av Constants.h för att smidigt underlätta denna process. Vill programmeraren skapa en subklass till ett objekt (i detta fall döpt “Component”), behöver denne enbart ange en förenklad sökväg till bilden således denna finns i resursmappen. Ett exempel på detta är vore att initiera en komponent på detta vis:

*"images/exempel.png”*

**Spelet**

För att bygga ett spel med hjälp av motorn skapar man först en ny fil, i detta testfall är filen döpt efter spelet (SpaceDuel.cpp). I den finner man en main-funktion som är den funktionen som startar igång det byggda spelet. I vår SpaceDuel.cpp fil inkluderas spelunika klasser, som till exempel Ship.h, som är en subklass till den generella “Component”-klassen. Efter att alla önskade spelklasser byggts färdigt är det dags för programmeraren att lägga till dessa till en session av spelmotorn. För att göra det implementerar man ett namespace vid namnet GameEngine som låter en nå alla spelmotorklasser direkt i main. För att lägga till ett objekt till spelet kan man använda sig av engine, som tack vare dess add()-metod låter programmerare lägga till objekt av spelklasserna direkt in till spelet. När man är nöjd med vad som adderats startar man igång motorn med engine.run(), som skrivs näst längst ner i main-metoden, precis över main-metodens “return 0;”.

Då motorn är anspassad för att skapa multiplayer-spel kräver den att minst två spelare är initierade innan run() får köras. Man initierar spelare med hjälp av engine’s InitializePlayers()-funktion. I vårt fall initieras två instanser av spelklassen “Ship”, men eftersom InitializePlayers() tar en generell komponent som parameter är denna anpassad för alla nya sorters spelklasser som programmeraren önskar initiera.

För att spela SpaceDuel kan denna klass implementeras:

#include "Constants.h" *//gResPath-contains path to your resources.*

#include <SDL2/SDL.h>

#include "Session.h"

#include "Bullet.h"

#include "Ship.h"

#include "Background.h"

#include <SDL2/SDL\_image.h>

#include "System.h"

#include <string>

#include "iostream"

using **namespace** GameEngine;

**int** main(**int** argc, **char\*\*** argv) {

*//Skapar en bakgrund med angivet filnamn*

Background\* background = new Background("images/bg.png");

engine.add(background);

*//Skapar två spelare*

Ship\* playerOne = new Ship("images/ship.png", 1, 300, 400, 100, 100);

Ship\* playerTwo = new Ship("images/ship2.png", 2, 300, 50, 100, 100);

*//Initialiserar spelarna i spelet*

engine.InitializePlayers(playerOne);

engine.InitializePlayers(playerTwo);

*//Startar spelet. OBS! Kan endast göras när minst två spelare initialiserats.*

engine.run();

return 0;

}