LINKÖPINGS UNIVERSITET

# TDP005 Projekt: Objektorienterat system

# Dokumentmall

Författare

Daniel Persson, daniel.o.persson@liu.se
Emanuel Kinberger, emanuel.kinberger@liu.se
Klas Arvidsson, klas.arvidsson@liu.se

INSTITUTIONEN FÖR DATAVETENSKAP

Höstterminen 2016
Version 1.1

2016-09-05

# 1 Revisionshistorik

| Ver. | Revisionsbeskrivning | Datum |
|---|---|---|
| 1.1 | Modifierad för att stödja xelatex och unicode | 150603 |
| 1.0 | Skapad för studenter att använda som mall till kommande dokumentinlämningar | 140908 |

# 2 Möte

Under mötet så disskuterade vi varandras kod, vad som var bra och vad som kunde förbättras.
Vi förberede mötet genom att ladda upp projekten på gitlab och se till att alla beställargruppen hade tillgång till koden. För att underlätta mötet så såg vi till att vi och vår beställargrupp hade läst varandras kod så vi inte skulle behöva förklara koden på mötet.
Vi hade mötet den 8:de December 2016 klockan 13:00

# 3 Granskat projekt

Det spelet som vi granskade och gav feedback på heter “Arcaball-X”, det går ut på att förstöra alla brickor genom att styra en liten platta längst ner på spelet. Man ska studsa en boll på platan så den åker upp på brickorna, om bollen nuddar en bricka så går den sönder.

## 3.1 Klassuppdelning

“Arcaball-X” är uppdelat i flera olika filer, de som fanns när vi granskade projektet heter:

- GameEngine
- GameLoop
- GameMenu
- GameState
- Graphic
- ResourceManager

### 3.1.1 Alla klasser

De borde överväga att ännända “container.at(x)” istället för “[]-operatorn”, fördelen med “container.at(x)” är att den throwar ett error om indexet inte finns, vilket “[]-operatorn” inte gör. Detta kan framförallt vara bra under utveckling då man inte missar några undefined behaviors, men om de kontrollerar sizen på vectorn innan så är det ingen fara.

Konstruktorerna borde definieras i “.cc” filen och enbart deklareras i “.h” filen, just nu deklareras och definieras klasserna i “.h” filen.

### 3.1.2 GameEngine

Av vår förståelse så innehåller klassen GameEngine funktioner och variabler som hanterar hela spelet. Det är denna klassen som används från main filen, funktionen "startGame()" startas och den styr sedan hela spelet.
Variablerna "brickHeight" och "brickWidth" som finns i denna klassen tycker vi inte borde finnas här utan borde finnas i en brick klass istället då detta hör till brickorna. En stor fördel med detta är att man kan ha brickor med olika height och width vilket kan vara interessant på en bana.
Dom har även instanser av andra klasser sparade som medlemsvariabler i GameEngine, detta tycker vi är väldigt snyggt gjort då de kan komma åt exakt allt dom behöver från denna klassen.

I funktionen "GameEngine::startGame()" så hittade vi två saker som kan förbättras, på rad 34 så finns det en while-loop utan några måsvingar, det blir väldigt mycket tydligare om man har while och if-satser med måsvingar, speciellt om det ligger en helt if-sats i while-loopen vilket det i detta fallet gör.

Allt detta ligger även i en while-loop som har "while(!done!)" som argument, det blir mycket tydligare om man istället har "while(window.isOpen()" som argument då man enklare fattar att denna loopen kommer köras sålänge spelet är på.

I destruktorn så deletar dom även bara de första objektet av varje typ då dom inte använder en for-loop. De borde loopa över varje typ och deleta den typen för att undvika minnesläckor.

### 3.1.3 GameLoop

Klassen GameLoop tycker vi är snygg designad, det enda vi kunde hitta var på rad 87:
[Kod]

```
m_gameObjects[Balls][0]->upDate();
m_gameObjects[Pads][0]->move();
//draw all current objects
m_gameObjects[Balls][0]->move();
m_graphic->drawWindow(m_gameObjects[Balls][0]->draw());
m_graphic->drawWindow(m_gameObjects[Walls][0]->draw());
m_graphic->drawWindow(m_gameObjects[Walls][1]->draw());
m_graphic->drawWindow(m_gameObjects[Walls][2]->draw());
m_graphic->drawWindow(m_gameObjects[Pads][0]->draw());
```

Här målas inte alla gameObjects ut utan bara dom första, för att lösa detta måste de implementera en for-loop som loopar igenom alla gameObjects och målar ut dom på i den loopen istället.

### 3.1.4 GameMenu

Klassen GameMenu ärvs av klassen GameState, vilket är snyggt då de kan återanvände kod från den klassen.
Det enda vi hittade var hårdkodade värden för namn och poäng på rad 146:
[Kod]

```
std::string name{"Christer"};
std::string point{"12334"};
```

#### 3.1.5 GameState

GameState är bas-klassen för GameMenu, det enda den innehåller är en virtual funktion som returnerar en enum CurrentState.
Vi tycker inte att konstruktorna ska vara protected utan den borde vara public då det är standard i C++.
Vi tycker även det är tydligare att ha
[Kod]

```
GameState() = default;
```

istället för
[Kod]

```
GameState() {}
```

#### 3.1.6 Graphic

Av vår förståelse så hanterar Graphic klassen spelfönstret. Vi har inga kommentarer om denna klassen då vi tygger att designen av denna klassen är bra.

#### 3.1.7 ResourceManager

Precis som i GameState så tycker vi att konstruktorn och destruktorn ska vara default om man bara har tomma måsvingar efter dom då det blir tydligare vad de gör.
Vi tycker namnet är lite otydligt då när vi bara läste klassnamnet så lät det som att klassen hanterar pekare och andra resurser, ett bättre namn skulle kunna vara “loader” då det enda den gör är att ladda fonts och bilder.

### 3.2 Använding av const

Vi hittade flera ställen där de kan använda const på funktioner då dom inte ändrar några värden.

Funktionen “GameMenu::seeHighScore()” borde vara const då den inte ändrar några värden ( Om funktionen är tänkt att ändra några värden så är det ett dåligt namn på funktionen ). Detta är även sant för följande:

- Funktionen: “Graphic::isWindowOpen()”
- Funktionen: “Graphic::getEvent()”
- Funktionen: “ResourceManager::getTexture”
- Funktionen: “ResourceManger::getFont”

Följande parametrar borde vara const-referenser:

- Funktionen: “Graphic::setUpWindow()”, parametern “std::string name” borde vara en const-referens.
- Funktionen: “GameLoop::GameLoop()”, parametern “ent” är en referens men borde vara en const-referens.

### 3.3 Namngivning

Följande funktionsnamn och/eller variabelnamn tycker vi kan ändras för att bli tydligare:

- Funktionen “Graphic::setUpWindow()” borde heta “Graphic::setupWindow()”. Det finns t.ex “setupText()” och “setupMarkerVectors()” funktioner som följer den senare namnkonventionen.
- Funktionen: “GameEngine::setUpGraphic()” borde heta “GameEngine:setupGraphic()”.
- Funktionen: “GameMenu::seeHighScore” borde heta “GameMenu::seeHighscore()” eftersom Highscore är ett ord.

### 3.4 Minneshantering

Som vi skrev innan så finns det minnesläckor i koden just nu.
I destruktorn för klassen “GameEngine” så tas inte alla “gameObjects” bort, bara den första av varje typ vilket leder till minnesläckor.

### 3.5 Vårt projekt

Vi fick bra feedback på vår kod, gruppen som granskade vår kod tyckte att vår filuppdelning var väldigt förvirande då vi har flera olika mappar med olika verisioner av spelet. När vi väl visade vilken mapp de skulle kolla i så förstod de hur koden fungerade.
Dom hade lite problem med att förstå vår kod då själva spelet inte kompilerade, anledningen till att den inte kompilerade är för att när dom granskade koden höll vi på att göra om hela klassstrukturen vilket i skrivande stund nu är klart.