LINKÖPINGS UNIVERSITET

TDP005 Projekt: Objektorienterat system

Designspecifikation

Författare

Alexander Jonsson, alejo720@student.liu.se
Joakim Johansson, joajo229@student.liu.se

INSTITUTIONEN FÖR DATAVETENSKAP

Höstterminen 2016
Version 1.2

2016-12-17

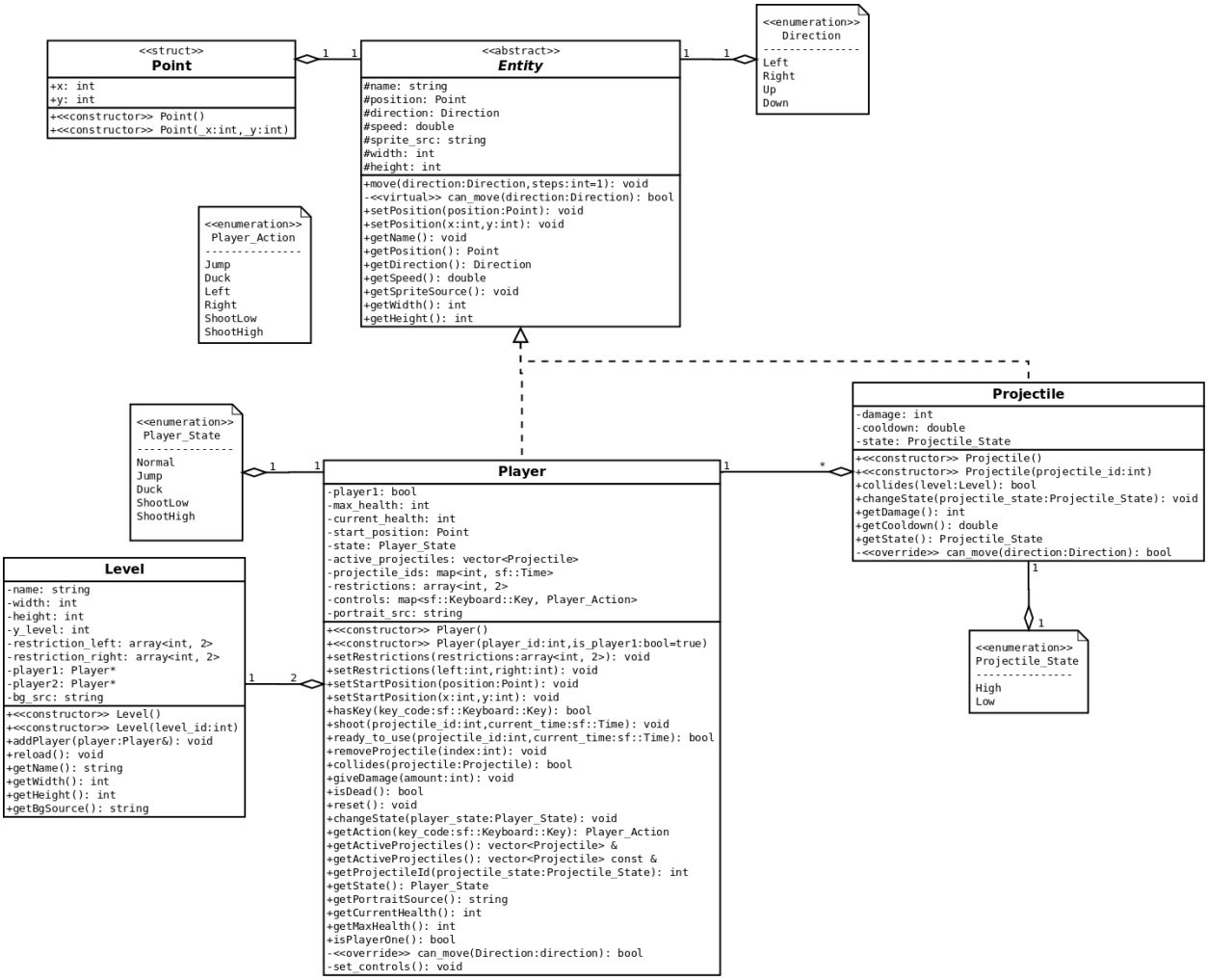
Innehåll

1 Revisionshistorik

Ver.	Revisionsbeskrivning	Datum
1.2	Uppdaterat klassdiagram och detaljbeskrivning, samt korrekturläst	2016-12-17
1.1	Lagt till detaljbeskrivning.	2016-11-30
1.0	Första utkast med klassdiagram, externa filer samt diskussion om valet av design.	2016-11-25

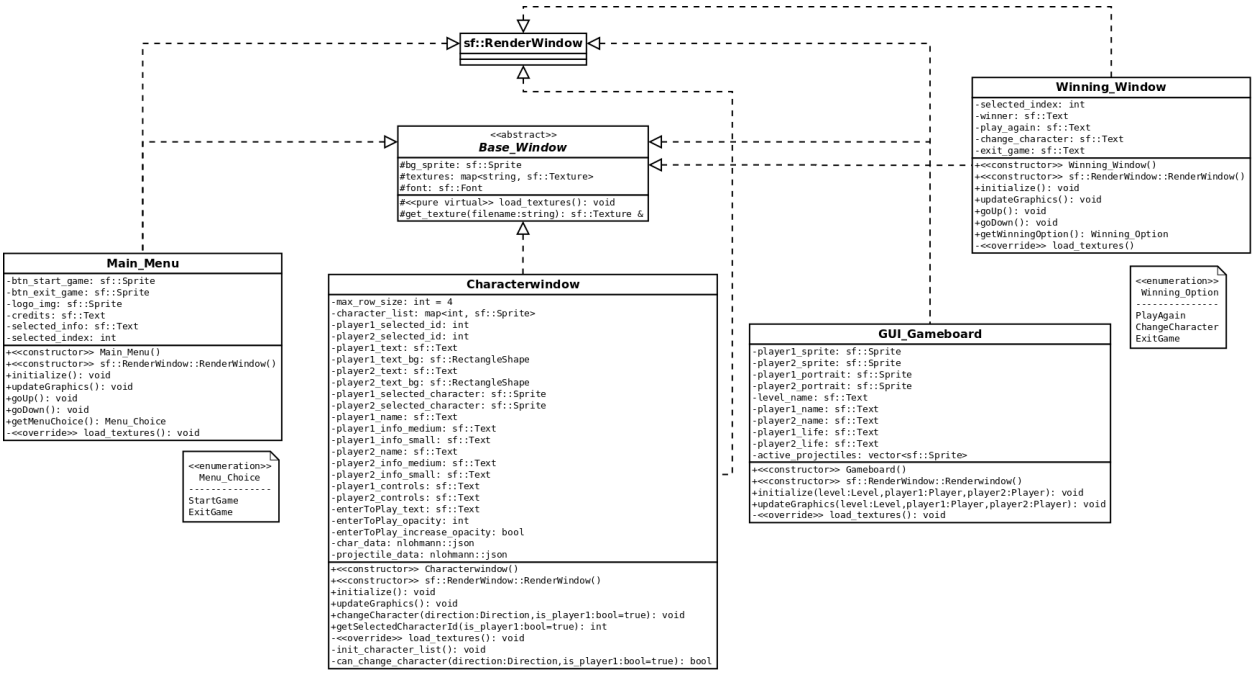
2 UML klassdiagram av systemet

Här visas ett UML klassdiagram för den logiska delen av systemet:



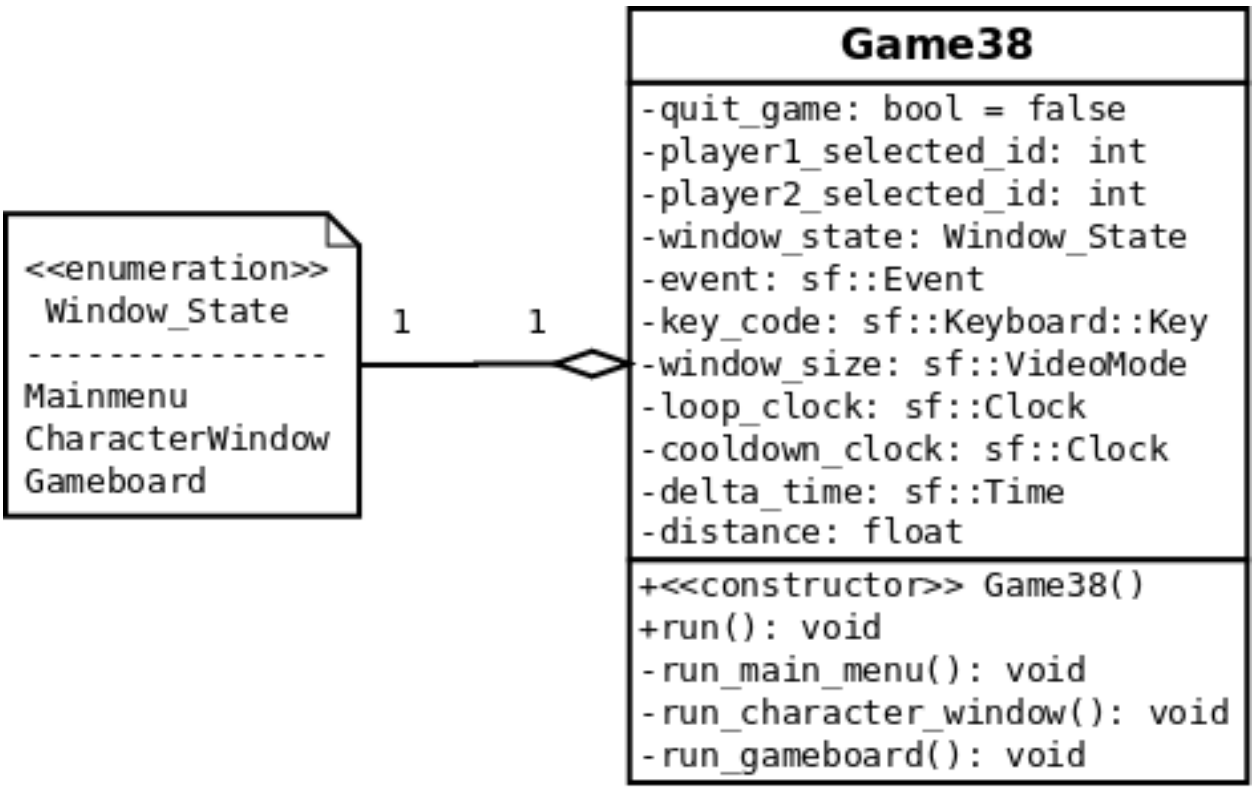
Figur 1: Klassdiagram: logisk del

Här visas ett UML klassdiagram för den grafiska delen av systemet:



Figur 2: Klassdiagram: grafisk del

Här visas ett UML klassdiagram för huvud klassen som representerar själva spelet:



Figur 3: Klassdiagram: spelklassen

3 Detaljbeskrivning

3.1 Player

- **Namn:**
  - Player (spelare)
- **Syfte:**
  - Syftet med klassen 'Player' är att ha en mall som representerar en spelare med olika egenskaper.
- **Relationer:**
  - Denna klass ärver från en 'Entity'-klass.
  - Klassen 'Level' hanterar två spelare (spelare 1 och 2).
  - Denna klass äger och hanterar 'Projectile'-objekt.
- **Konstruktor:**
  - Klassen har en konstruktor som tar in ett ID och om det är spelare 1 eller inte. Sedan används ID:t till att hämta korrekt data från JSON-filen innehållande karaktärdatan. Spelarens kontroller sätts även beroende på om det är spelare ett eller inte
- **Publika metoder:**
  - **hasKey(keycode)**: Kollar om den specificerade knappen är en knapp spelaren kan använda.
  - **shoot(projectile_id, current_time):** Skapar en projektil beroende på vilket ID och lägger in den i listan med spelarens projektiler. Samtidigt läggs nuvarande tid på i listan med spelarens projektil-id:n.

  readyToUse(projectile_id, current_time) Kollar om projektilen med den specificerade projektil-id:t kan användes beroende på hur långt tid som gått sedan senaste användningen av projektilen.

  removeProjectile(index) Tar bort en spelares projektil beroende på position i listan med aktiva projektiler
  - **collides(projectile):** Kollar om spelaren kolliderar med den specificerade projektilen.
  - **giveDamage(amount):** Minskar spelarens liv med den specificerade mängden.
  - **isDead():** Kollar om spelarens liv är noll.
  - **reset():** Flyttar t.ex. spelaren till spelarens startposition, ger spelaren fullt liv och tar bort alla ägda projektiler från spelaren.
  - **move(direction, steps)**: Flyttar spelaren ett specificerat antal steg åt den specificerade riktningen tills det inte är möjligt längre.
- **Variabler:**
  - **position:** Håller reda på spelarens nuvarande position.
  - **direction:** Håller reda på vilket håll spelaren är vänd åt.
  - **speed:** Håller reda på hur snabbt spelaren kan röra sig.
  - **sprite_src:** Filnamnet på spelarens textur.
  - **portrait_src:** Filnamnet på spelarens porträttbild.
  - **width:** Spelarens bred.
  - **height:** Spelarens höjd.
  - **name:** Namnet på karaktären spelaren har valt.
  - **player1:** Håller reda på om spelaren är spelare 1 eller 2.
  - **max_health:** Maximala mängden liv spelarens karaktär kan ha.
  - **health:** Nuvarande mängden liv spelaren har.
  - **start_position:** Position spelaren ska start på, i detta fall på spelplanen.
  - **state:** Håller reda på vilket tillstånd spelaren är i (Hoppa, ducka eller stående).
  - **active_projectiles:** En lista med projektiler spelaren skjuter just nu.
  - **projectile_ids:** En lista med ID:n på projektiler som spelaren kan använda.
  - **restrictions:** Definierar ett område spelaren kan röra sig inom, i detta fall på spelplanen.
  - **controls**: En tabell på vilka knappar som leder till vilken spelarhandling.

3.2 Level

- **Namn:**
  - Level (nivå)
- **Syfte:**
  - Syftet med klassen 'Level' är att ha en mall som representerar en spelplan med olika egenskaper.
- **Relationer:**
  - Denna klass äger två spelare: ('Player'-klasser) spelare 1 och 2.
- **Konstruktor:**
  - Klassen har en konstruktor som tar in ett ID. Sedan används ID:t till att hämta korrekt data från JSON-filen innehållande nivådatan.
- **Publika metoder:**
  - **addPlayer(player):** Lägger till en spelare till spelplanen och lagras i spelare 1 eller 2 beroende på vad den specificerade spelaren är.
  - **reload():** Återställer spelplanen och alla objekt på den.
- **Variabler:**
  - **name:** Namnet på nivån.
  - **width:** Bredden på nivån.
  - **height:** Höjden på nivån.
  - **y_level:** Bestämmer hur högt spelarna står ifrån botten av nivån.
  - **restriction_left:** Definierar vänstra planhalvan.
  - **restriction_right:** Definierar högra planhalvan.
  - **player1:** Spelare 1 på spelplanen.
  - **player2:** Spelare 2 på spelplanen.
  - **bg_src:** Filnamn på nivåns bakgrundsbild.

4 Diskussion av valet om design

4.1 Fördelar

• Klasser ärver andra klasser. Exempelvis ärver 'Player' och 'Projectile'-klasserna utav 'Entity'-klassen vilket leder till återanvändning av kod. Det gör det även lättare att skapa en liknande klass senare.

• De grafiska klasserna innehåller endast grafiska komponenter vilket leder till separation mellan det grafiska och logiska. Detta gör det möjligt att det grafiska eller logiska ska kunna användas till andra system.

• Att läsa objekts egenskaper från filer leder till mindre kod då vi slipper hårdkoda objektens egenskaper inuti koden.

4.2 Nackdelar

• Alla grafiska klasser som representerar ett fönster ärver både från en 'Base_Window' och 'sf::RenderWindow' klassen. Eftersom 'Base_Window' ska representera ett fönster skulle det ha varit med lämpligt att endast 'Base_Window' ärvde från 'sf::RenderWindow'. Detta skulle skapa en smidigare klassstruktur.

• I spelarobjektet lagras det internt om spelaren är spelare 1 eller 2. Detta begränsar spelarklassen ifall det skulle behövas fler än två spelare.

5 Externa filer

5.1 Mappstruktur

```
resources
    img
        level
            *_portrait.png
            *.png
        character
            *_portrait.png
            *.png
        projectile
            *.png
    data
        levels.json
        characters.json
        projectiles.json
        *.json
    music
        *.mp3
    sfx
        *.wav
*.cc
*.h
settings.config
```

I 'root'-mappen ligger alla 'C++'-filer, konfigurationsfiler och en mapp som innehåller alla statiska filer som spelet ska ladda information från. Exempel på denna information är bilder, musik, ljudfiler och data som krävs för de olika objekten. '*_portrait'-bilderna är de bilder som ska visas vid val av karaktär. Alla data filer som ska användas är i 'JSON'-format vilket kräver att vi laddar ner en modul för att hantera 'JSON'-filer i C++.

5.2 Nivåer

Detta är ett exempel på hur en 'JSON'-fil innehållandes de olika nivåernas data skulle kunna se ut:

```
[
    {
        "id": 1,
        "name": "Avalanche Leftovers",
        "width": 800,
        "height": 640,
        "y_level": 100,
        "restriction_left": [20, 350],
        "restriction_right": [400, 780],
        "bg_src": "avalanche_leftovers.png",
        "portrait_src": "avalanche_leftovers_portrait.png",
        "music": "icy_wind.wav"
    },
    {
        "id": 2,
        "name": "Firelands",
        "width": 1024,
        "height": 1280,
        "y_level": 20,
        "restriction_left": [100, 200],
        "restriction_right": [824, 924],
        "bg_src": "firelands.png",
        "portrait_src": "firelands_portrait.png",
        "music": "scorching_battle.wav"
    }
]
```

5.3 Karaktärerer

Detta är ett exempel på hur en 'JSON'-fil innehållandes de olika karaktärernas data skulle kunna se ut:

```
[
        {
                "id": 1,
                "name": "Crazy guy",
                "width": 190,
                "height": 220,
                "max_health": 1000,
                "speed": 1.2,
                "projectiles": [1, 2],
                "sprite_src": "crazy_guy.png",
                "portrait_src": "crazy_guy_portrait.png"
        },
        {
                "id": 2,
                "name": "Less crazy guy",
                "width": 190,
                "height": 220,
                "max_health": 750,
                "speed": 1.5,
                "projectiles": [3, 4],
                "sprite_src": "less_crazy_guy.png",
                "portrait_src": "less_crazy_guy_portrait.png"
        }
]
```

5.4 Projektiler

Detta är ett exempel på hur en 'JSON'-fil innehållandes de olika projektilernas data skulle kunna se ut:

```
[
        {
                "id": 1,
                "name": "Football",
                "state": "low",
                "width": 40,
                "height": 40,
                "damage": 25,
                "speed": 3.0,
                "sprite_src": "football.png",
                "cooldown": 0.3
        },
        {
                "id": 2,
                "name": "Mountain Dew",
                "state": "high",
                "width": 40,
                "height": 40,
                "damage": 100,
                "speed": 1.0,
                "sprite_src": "mountain_dew.png",
                "cooldown": 1
        },
        {
                "id": 3,
                "name": "BOMB",
                "state": "high",
                "width": 40,
                "height": 40,
                "damage": 450,
                "speed": 0.3,
                "sprite_src": "bomb.png",
                "cooldown": 5
        },
        {
                "id": 4,
                "name": "Nut",
                "state": "low",
                "width": 40,
                "height": 40,
                "damage": 7,
                "speed": 2,
                "sprite_src": "nut.png",
                "cooldown": 0.1
        }
]
```

5.5 Konfigurationsfil

Detta är exempel på hur strukturen i konfigurationsfilen skulle kunna se ut:

```
player1_jump 'w'
player1_left 'a'
player1_duck 's'
player1_right 'd'
player1_low_shoot 'r'
player1_high_shoot 't'
player2_jump 'up-arrow'
player2_left 'left-arrow'
player2_duck 'down-arrow'
player2_right 'right-arrow'
player2_low_shoot 'o'
player2_high_shoot 'p'
```