System design document for 017

Version: 0

Date: 2016-05-03

Author: Philip Tham, Anton Josefsson, Mikael Ragnhult, Stefan Fritzon

This version overrides all previous versions.

# 1 Introduction

## 1.1 Design goals

We want to design the system in a modular way, with each module as loosely coupled to each other as possible. In particular, we want the application to be extendable and modifiable due to system requirements changing, specifically due to game design changes.

## 1.2 Definitions, acronyms and abbreviations

# 2 System design

## 2.1 Overview

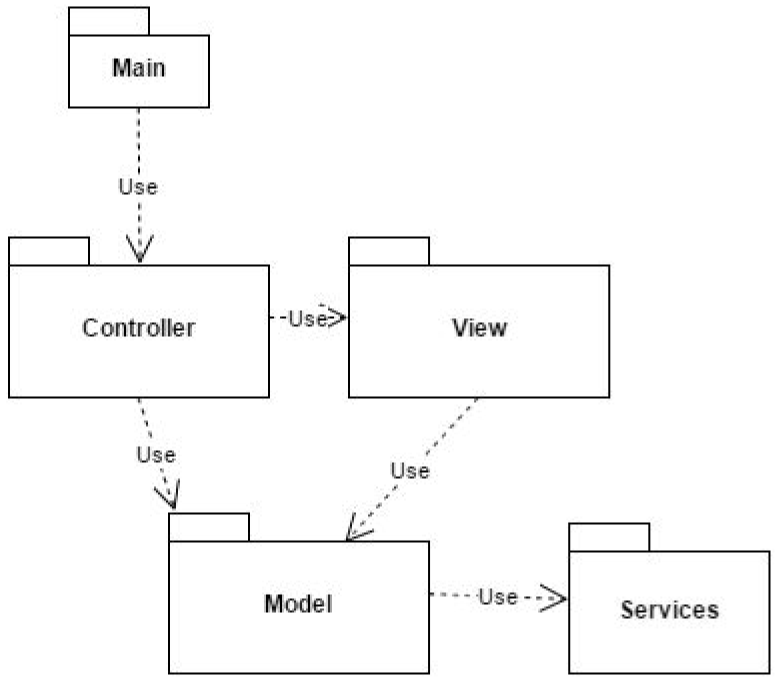
The application will use the MVC model to separate rendering, logic and input from each other. Generally, we want to use a pull strategy between the view and the model (i.e the renderer extracts necessary data from the model), and the controller should handle any user input and instruct the model to change as appropriate.

For necessary large changes in the model (such as transitions between levels) and in order to preserve modularity of certain services (such as audio, potential achievements) we will use the observer pattern, allowing the model to broadcast events to any potential interested parties without breaking the dependencies.

## 2.2 Software decomposition

### 2.2.1 General

Our application is divided into the following packages

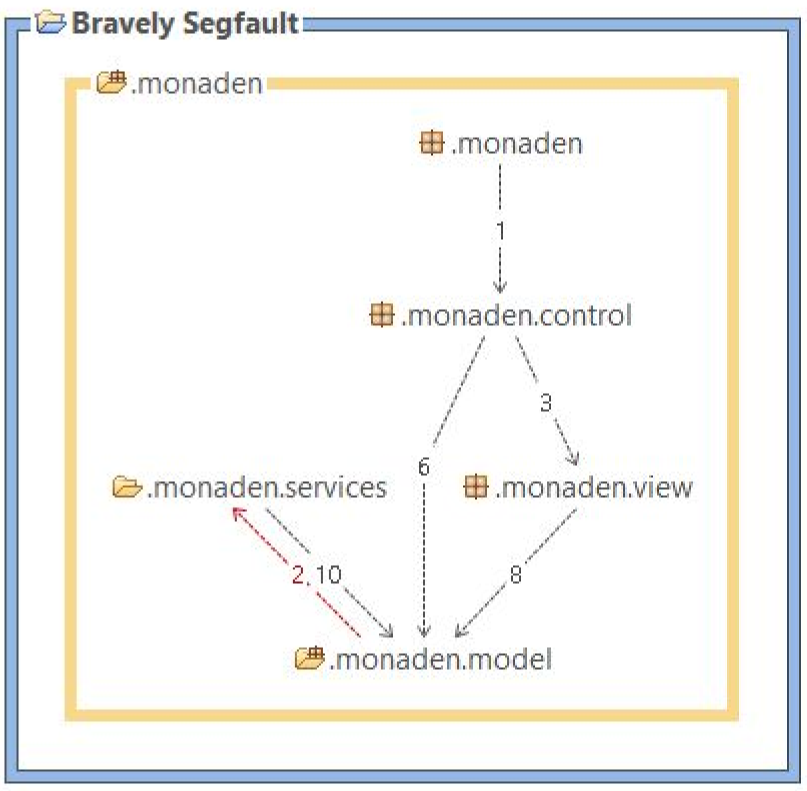


Main is the entry point of the program
Controller contains input handling
View contains rendering code
Model contains classes that both contain and modify data
Services contains functionality such as parsing and audio playing

### 2.2.2 Decomposition into subsystems

The system currently has two services, the tile parser and the level parser. The tile parser is responsible for parsing an xml-file containing a certain id and what graphic file the id represents, which is used when rendering the levels. The level parser is responsible for parsing xml files containing what objects a level consists of, what characters there are in the level, and what other levels you can move to from that level.

### 2.2.4 Dependency analysis



Dependencies were analyzed with the STAN tool. Apparently, there is an issue between services and model, probably due to the services and model having a pull strategy for the parsing of data. This should probably be fixed at a future date, but is likely not a big issue.

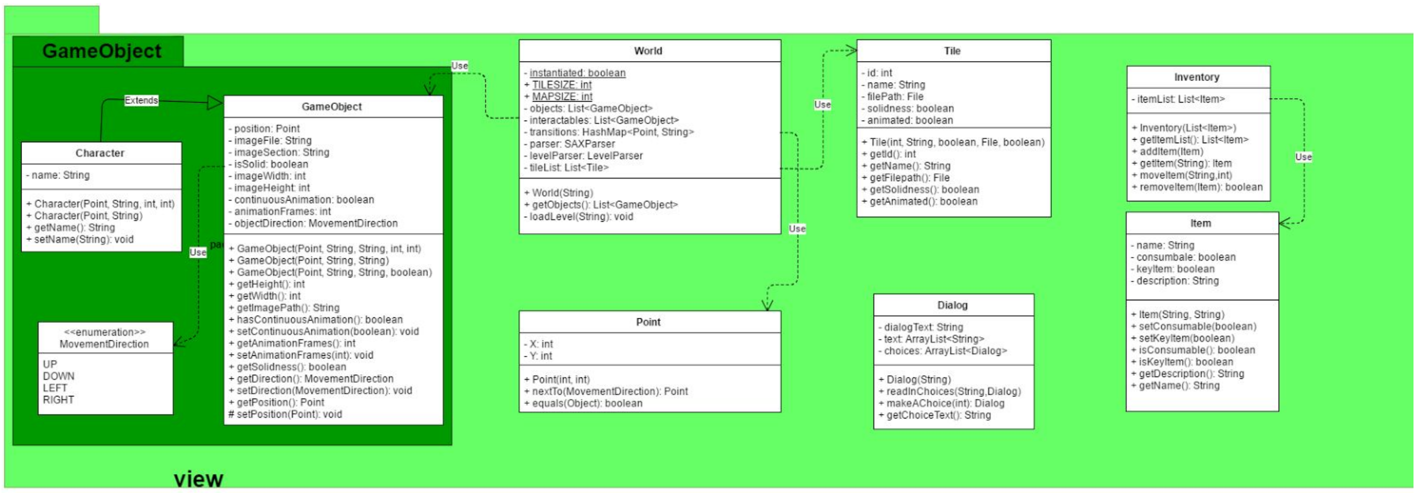
## 2.4 Persistent data management

The application will contain a service for saving and loading a game in progress. The details of these services have yet to be designed.

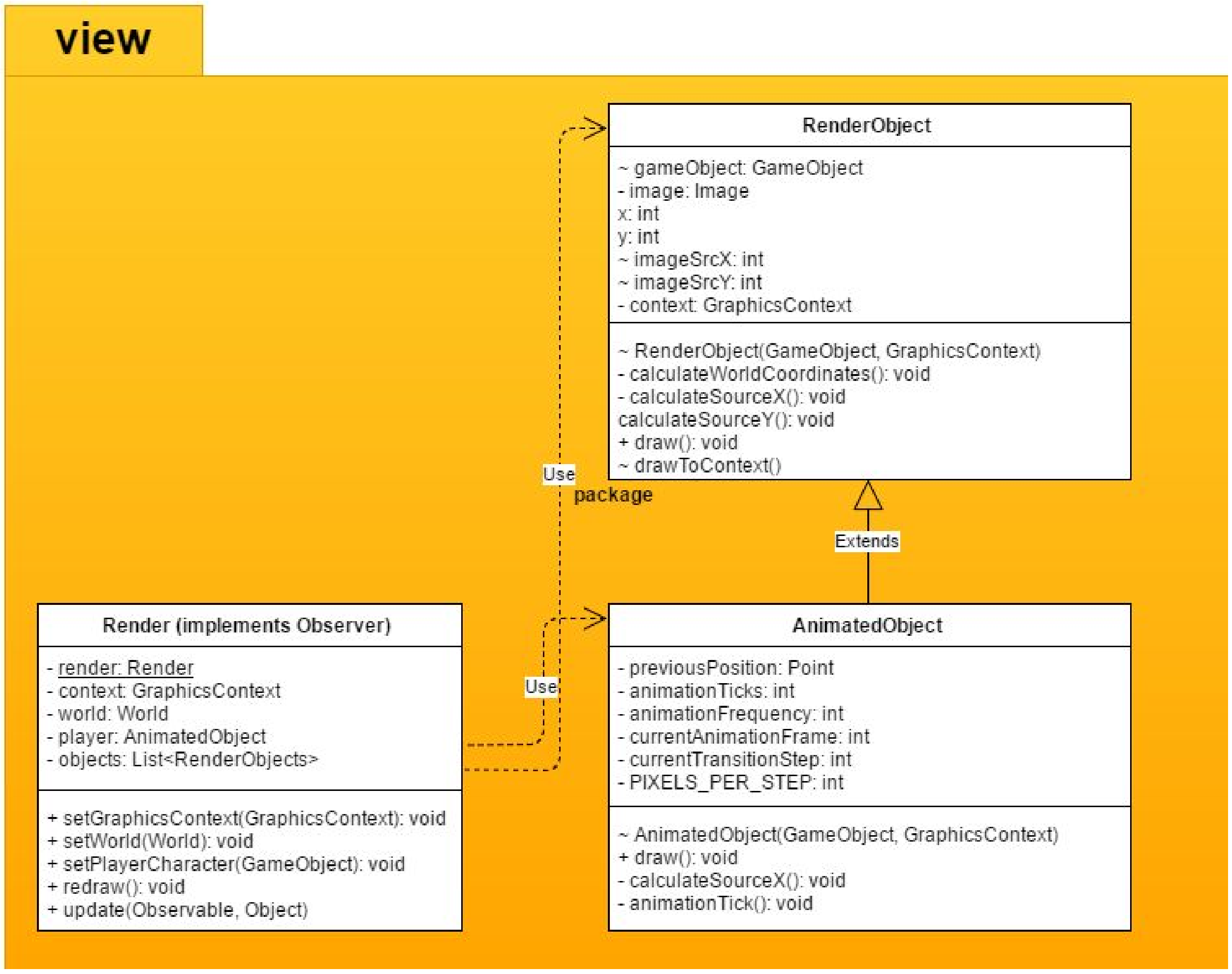
# 3 References

# APPENDIX

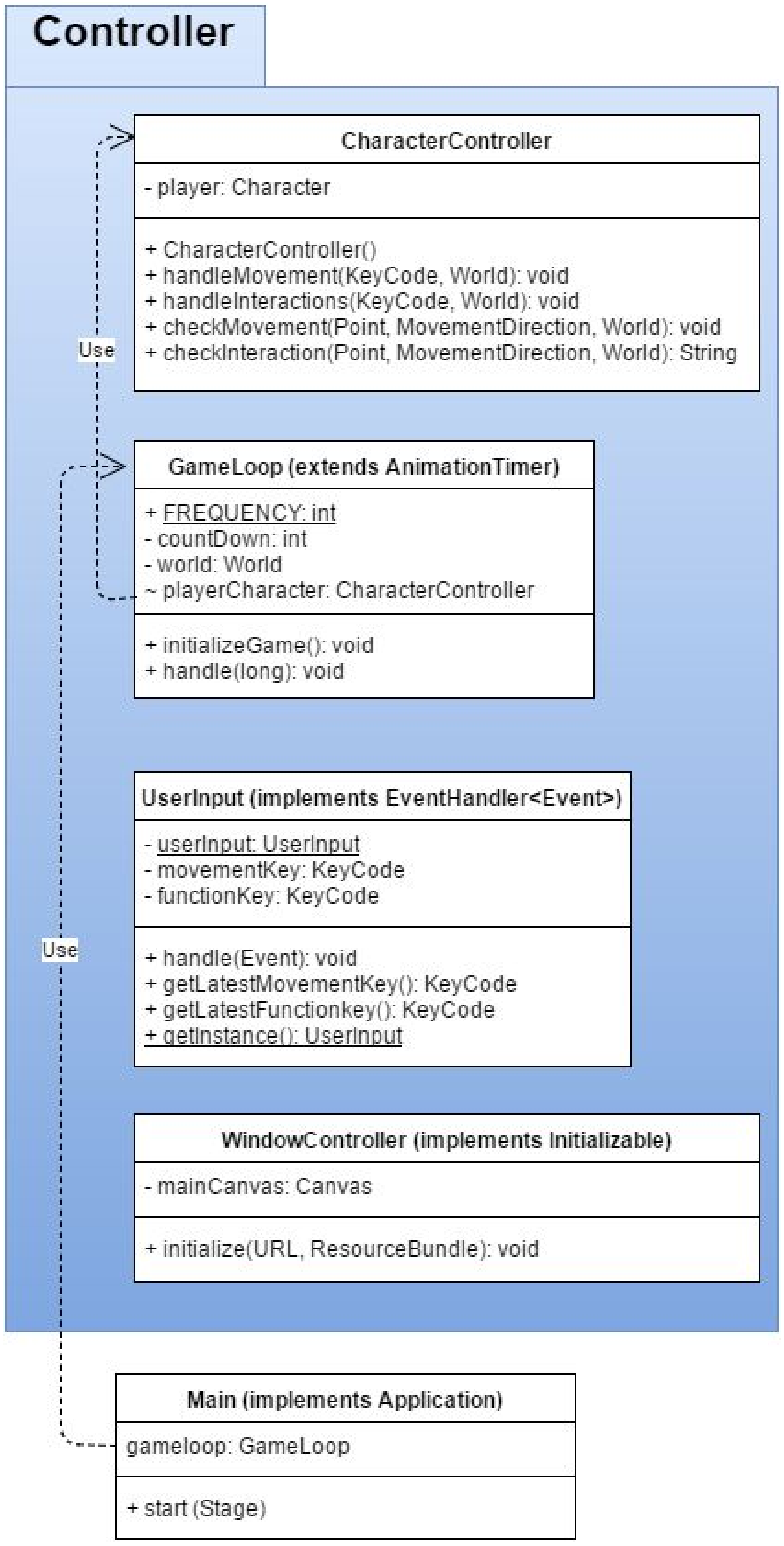
Model UML


GameObject
Extends
Use
Character
- name: String
+ Character(Point, String, int, int)
+ Character(Point, String)
+ getName(): String
+ setName(String): void
Use
<<enumeration>>
MovementDirection
UP
DOWN
LEFT
RIGHT
GameObject
- position: Point
- imageFile: String
- imageSection: String
- isSolid: boolean
- imageWidth: int
- imageHeight: int
- continuousAnimation: boolean
- animationFrames: int
- objectDirection: MovementDirection
+ GameObject(Point, String, String, int, int)
+ GameObject(Point, String, String)
+ GameObject(Point, String, String, boolean)
+ getHeight(): int
+ getWidth(): int
+ getImagePath(): String
+ hasContinuousAnimation(): boolean
+ setContinuousAnimation(boolean): void
+ getAnimationFrames(): int
+ setAnimationFrames(int): void
+ getSolidness(): boolean
+ getDirection(): MovementDirection
+ setDirection(MovementDirection): void
+ getPosition(): Point
# setPosition(Point): void
World
- instantiated: boolean
+ TILESIZE: int
+ MAPSIZE: int
- objects: List<GameObject>
- interactables: List<GameObject>
- transitions: HashMap<Point, String>
- parser: SAXParser
- levelParser: LevelParser
- tileList: List<Tile>
+ World(String)
+ getObjects(): List<GameObject>
- loadLevel(String): void
Use
Use
Tile
- id: int
- name: String
- filePath: File
- solidness: boolean
- animated: boolean
+ Tile(int, String, boolean, File, boolean)
+ getId(): int
+ getName(): String
+ getFilepath(): File
+ getSolidness(): boolean
+ getAnimated(): boolean
Point
- X: int
- Y: int
+ Point(int, int)
+ nextTo(MovementDirection): Point
+ equals(Object): boolean
Dialog
- dialogText: String
- text: ArrayList<String>
- choices: ArrayList<Dialog>
+ Dialog(String)
+ readInChoices(String,Dialog)
+ makeAChoice(int): Dialog
+ getChoiceText(): String
Inventory
- itemList: List<Item>
+ Inventory(List<Item>)
+ getItemList(): List<Item>
+ addItem(Item)
+ getItem(String): Item
+ moveItem(String,int)
+ removeItem(Item): boolean
Use
Item
- name: String
- consumbale: boolean
- keyItem: boolean
- description: String
+ Item(String, String)
+ setConsumable(boolean)
+ setKeyItem(boolean)
+ isConsumable(): boolean
+ isKeyItem(): boolean
+ getDescription(): String
+ getName(): String
view

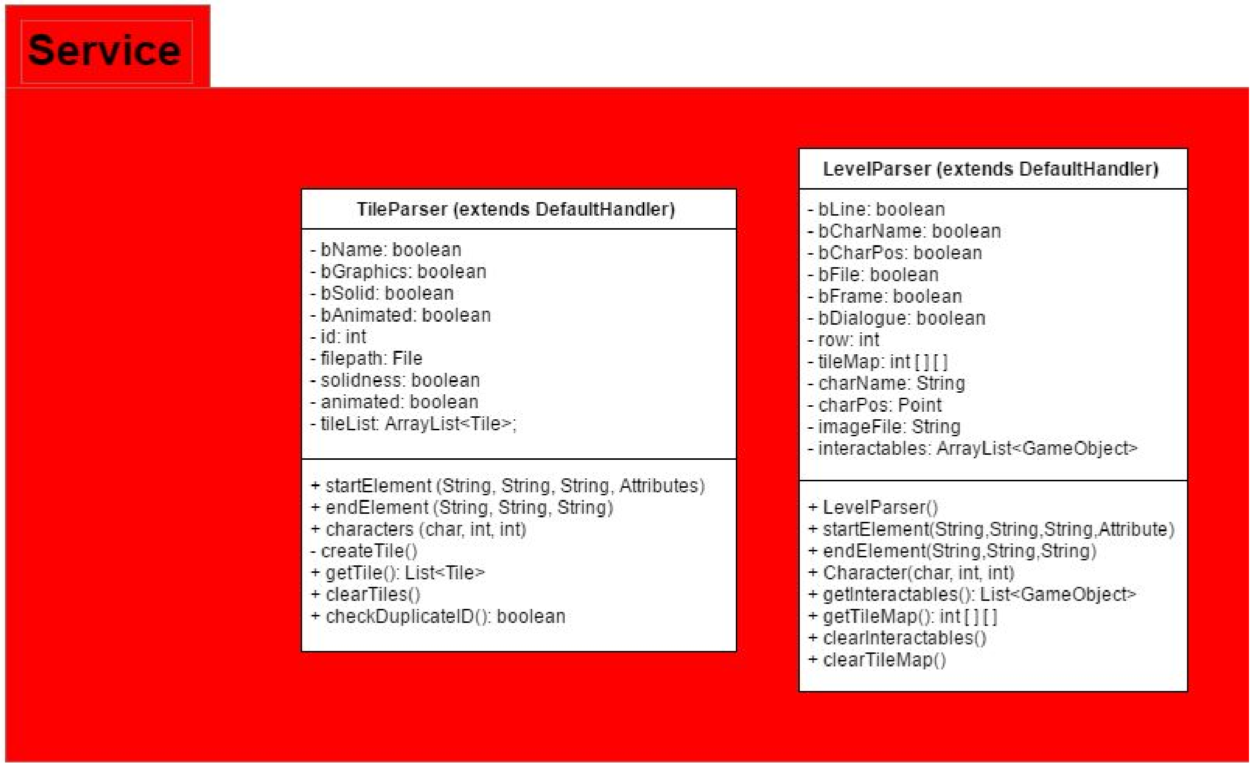
View UML


view
RenderObject
~ gameObject: GameObject
- image: Image
x: int
y: int
~ imageSrcX: int
~ imageSrcY: int
- context: GraphicsContext
~ RenderObject(GameObject, GraphicsContext)
- calculateWorldCoordinates(): void
- calculateSourceX(): void
calculateSourceY(): void
+ draw(): void
~ drawToContext()
Use
package
Extends
Render (implements Observer)
- render: Render
- context: GraphicsContext
- world: World
- player: AnimatedObject
- objects: List<RenderObjects>
+ setGraphicsContext(GraphicsContext): void
+ setWorld(World): void
+ setPlayerCharacter(GameObject): void
+ redraw(): void
+ update(Observable, Object)
Use
AnimatedObject
- previousPosition: Point
- animationTicks: int
- animationFrequency: int
- currentAnimationFrame: int
- currentTransitionStep: int
- PIXELS_PER_STEP: int
~ AnimatedObject(GameObject, GraphicsContext)
+ draw(): void
- calculateSourceX(): void
- animationTick(): void

Controller UML


Controller
CharacterController
- player: Character
+ CharacterController()
+ handleMovement(KeyCode, World): void
+ handleInteractions(KeyCode, World): void
+ checkMovement(Point, MovementDirection, World): void
+ checkInteraction(Point, MovementDirection, World): String
Use
GameLoop (extends AnimationTimer)
+ FREQUENCY: int
- countDown: int
- world: World
~ playerCharacter: CharacterController
+ initializeGame(): void
+ handle(long): void
UserInput (implements EventHandler<Event>)
- userInput: UserInput
- movementKey: KeyCode
- functionKey: KeyCode
+ handle(Event): void
+ getLatestMovementKey(): KeyCode
+ getLatestFunctionkey(): KeyCode
+ getInstance(): UserInput
Use
WindowController (implements Initializable)
- mainCanvas: Canvas
+ initialize(URL, ResourceBundle): void
Main (implements Application)
gameloop: GameLoop
+ start (Stage)

Service UML


Service
TileParser (extends DefaultHandler)
- bName: boolean
- bGraphics: boolean
- bSolid: boolean
- bAnimated: boolean
- id: int
- filepath: File
- solidness: boolean
- animated: boolean
- tileList: ArrayList<Tile>;
+ startElement (String, String, String, Attributes)
+ endElement (String, String, String)
+ characters (char, int, int)
- createTile()
+ getTile(): List<Tile>
+ clearTiles()
+ checkDuplicateID(): boolean
LevelParser (extends DefaultHandler)
- bLine: boolean
- bCharName: boolean
- bCharPos: boolean
- bFile: boolean
- bFrame: boolean
- bDialogue: boolean
- row: int
- tileMap: int [ ] [ ]
- charName: String
- charPos: Point
- imageFile: String
- interactables: ArrayList<GameObject>
+ LevelParser()
+ startElement(String,String,String,Attribute)
+ endElement(String,String,String)
+ Character(char, int, int)
+ getInteractables(): List<GameObject>
+ getTileMap(): int [ ] [ ]
+ clearInteractables()
+ clearTileMap()