# Requirements and Analysis Document for “Frank the Tank”

**Table of Contents**

1 Introduction
- 1.1 Purpose of application
- 1.2 General characteristics of application
- 1.3 Scope of application
- 1.4 Objectives and success criteria of the project
- 1.5 Definitions, acronyms and abbreviations

2 Requirements
- 2.1 Functional requirements
- 2.2 Non-functional requirements
  - 2.2.1 Usability
  - 2.2.2 Reliability
  - 2.2.3 Performance
  - 2.2.4 Supportability
  - 2.2.5 Implementation
  - 2.2.6 Packaging and installation
  - 2.2.7 Legal
- 2.3 Application models
  - 2.3.1 Use case model
  - 2.3.2 Use cases priority
  - 2.3.3 Domain model
  - 2.3.4 User interface
- 2.4 References

Appendix I
GUI
Domain model
Use case texts

**Version** 1.1
**Date** 2012-05-20
**Authors** Johan Brook, Jesper Persson, John Barbero Unenge, Chris Nordqvist

This version overrides all previous versions.

# 1 Introduction

## 1.1 Purpose of application

The project's goal is to create a two-dimensional, wave-based, survival shoot-em-up-game. The player controls a character, "Frank", who is fighting through waves of enemies whose numbers are getting higher and higher for each new wave. The (never ending) goal of the game is to survive the longest time possible.

## 1.2 General characteristics of application

The application will be a multi-platform (OS X, Windows and Linux) desktop application, controlled with a keyboard, with rich interface design.

When the player starts the game, the character begins on the first wave and he has a set of weapons to use in order to fight the enemies, as well as a special "Portal Gun", which is used for creating portals. Two portals may be created, and with them it will be possible to teleport the character or any other entity to another physical location in the game world. If the enemies get too close to the character, they will start giving damage, and the character's health will incrementally drop. When the health of the character goes down to zero, the game session ends (game over).

Subtle sound effects will be used where appropriate.

## 1.3 Scope of application

The application will not feature multiplaying in the software prototype.

## 1.4 Objectives and success criteria of the project

1. A player should be able to start a new game, move around in the game world, fight off at least two waves of enemies, and use the different weapons in the inventory (including the Portal Gun to teleport entities).
2. The game's performance should not prevent normal gameplay, i.e. no lag or glitches should be present in the final prototype when running the software on modern hardware on all specified platforms.

## 1.5 Definitions, acronyms and abbreviations

- Entity, an object in the game world.
- Character, the entity the player controls in the game.
- Enemy, the entity the character is suppose to fight.
- Spawn, a game entity is created in the game world.
- HUD, "Heads Up Display", interface elements which show status information.
- Wave, a new group of enemies spawned after the player successfully fight off the previous group.
- Java, the programming language used for creating the application.
- FPS - frames per second, the number of times per second that the game is updated. Higher FPS - less lag.

# 2 Requirements

## 2.1 Functional requirements

The player should be able to:

1. Start a new game.
2. Move around in the game world.
    a. The character can collide with objects.
    b. The character can move in eight different directions (north, north-east, east, south-east, south, south-west, west, north-west).
3. Fire a weapon in his arsenal.
    a. If it is a projectile weapon, and the projectile hits an enemy, the enemy should lose health. If the current weapon is out of ammo, the weapon can't be used.
    b. If it is the Portal Gun, two portals should be placed in the game world, and they should let entities pass through either of the portals and appear on the other side.
4. Exit the game.

## 2.2 Non-functional requirements

### 2.2.1 Usability

Players should be able to get started very quickly – the application focuses on simplicity and casual gaming. It should be clear for the player when the characters health drops down to zero and the game is over, and the same for when a new wave spawns.

There should be a setting for controlling the sound effects and volume.

### 2.2.2 Reliability

N/A.

### 2.2.3 Performance

1. The application should never drop below 30 FPS.
2. Any actions initiated by a player should get a seemingly immediate graphical response.

### 2.2.4 Supportability

The applications architecture should be extensible and easy to build upon. Things such as different level configurations and weapons should be easily implemented, with little effort. There should be a clear separation between models and views, i.e. sprites and visual representations of entities should be changeable and don't concern the model.

There should be automated tests verifying possible use cases, actions and effects in the gameplay.

### 2.2.5 Implementation

The application will require Java and the Java Runtime Environment in order to run properly.

### 2.2.6 Packaging and installation

### 2.2.7 Legal

N/A.

## 2.3 Application models

### 2.3.1 Use case model

See Appendx I for UML diagram and a list of Use Cases.

### 2.3.2 Use cases priority

1. *RunGame*
2. *MovePlayer*
3. *FireWeapon*
4. *ExitGame*
5. *FirePortalGun*

### 2.3.3 Domain model

See Appendix I.

### 2.3.4 User interface

Text to motivate a picture.

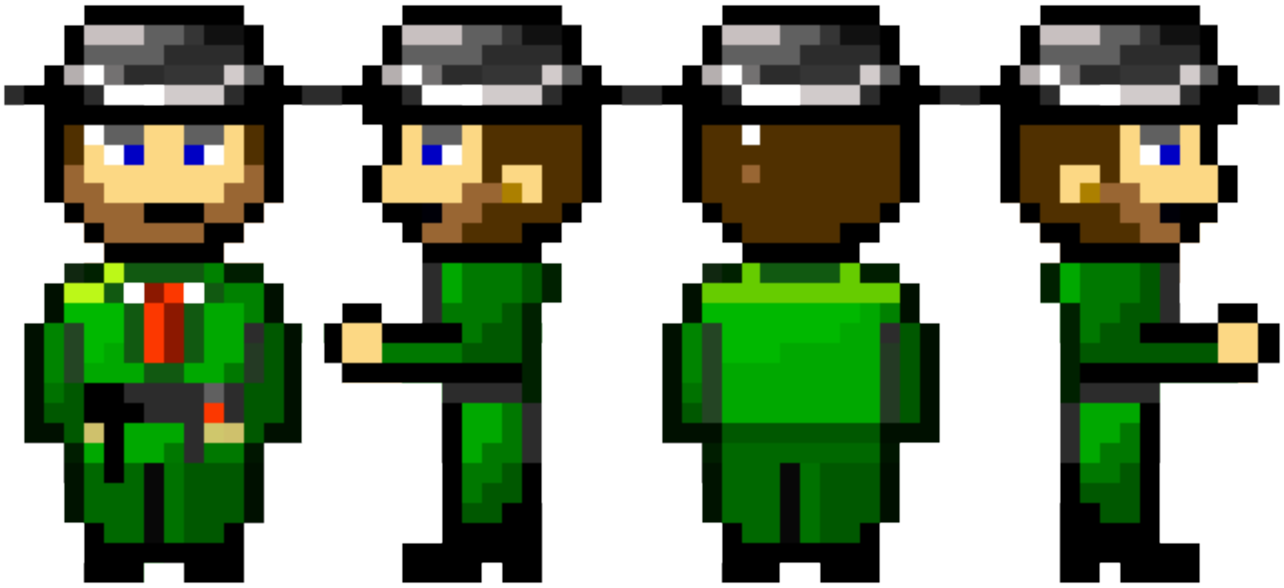
# Appendix I

# GUI

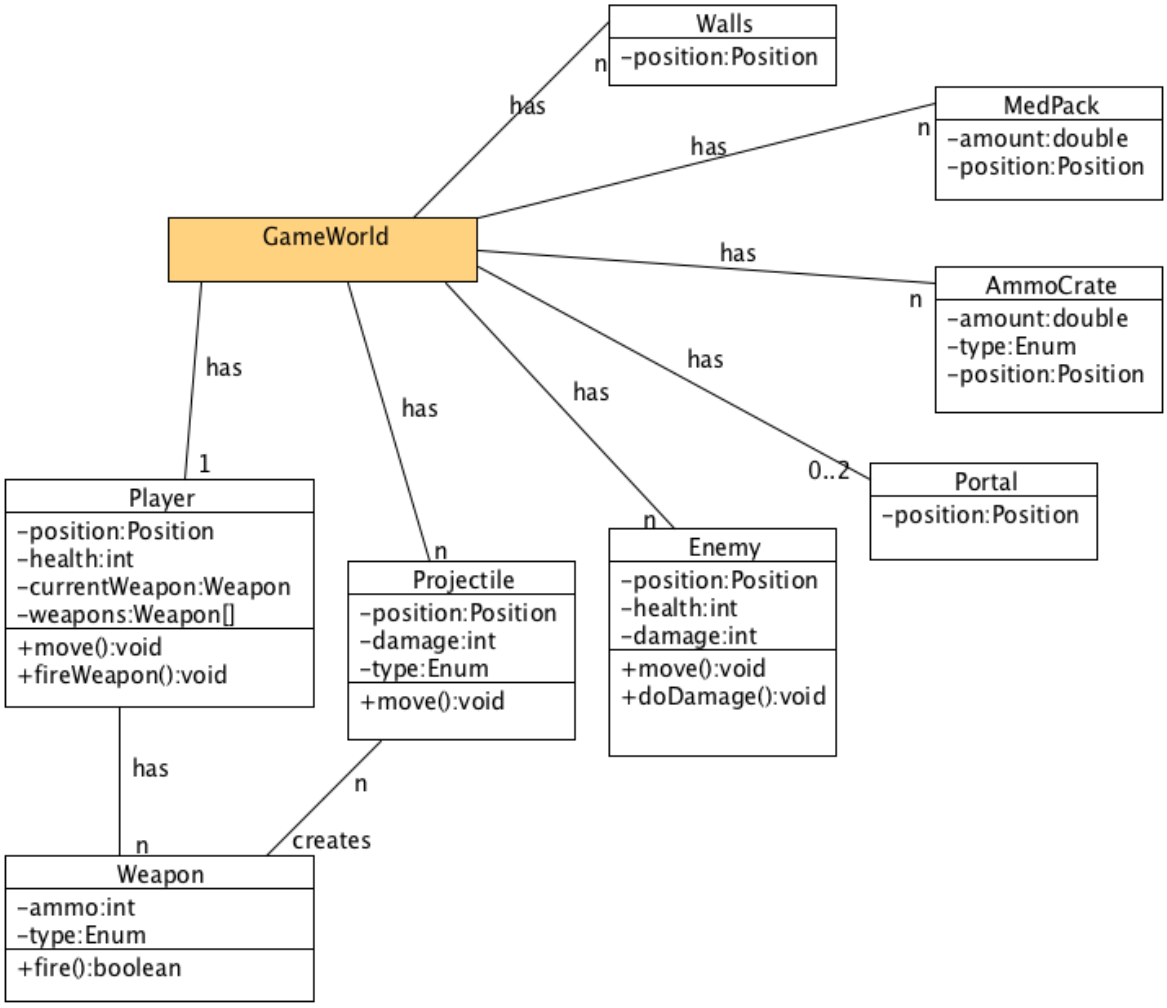
Draft of main character.

**1:1**

**1:1000**



## Domain model


Walls
-position:Position
MedPack
-amount:double
-position:Position
GameWorld
AmmoCrate
-amount:double
-type:Enum
-position:Position
Portal
-position:Position
Player
-position:Position
-health:int
-currentWeapon:Weapon
-weapons:Weapon[]
+move():void
+fireWeapon():void
Projectile
-position:Position
-damage:int
-type:Enum
+move():void
Enemy
-position:Position
-health:int
-damage:int
+move():void
+doDamage():void
Weapon
-ammo:int
-type:Enum
+fire():boolean
has
n
1
0..2
creates

# Use case texts

## Use case: *MovePlayer*

**Short description**
When the player moves in one direction of eight (north, south, east, west, north east, north west, south east, south west).

**Priority (high, mid, low)**
High

**Extends or Includes (other use case)**
N/A

**Participating actors**

- Player (P)
- Other Collidable Objects (O)

**Normal flow of events**

| **Player (=P)** | **System** |
|---|---|
| P presses any directional key(s) | |
| | Player character moves in given direction. |
| P releases the given key(s) | |
| | Player stops moving. |

**Alternate flow: special tile**

- If the player collides with an ammo tile, ammo is added to player's current ammo reserve.
- If the player collides with a med pack, hit points is added to the player's current health
- If the player collides with a portal it turns up at the other portal (given that there is another one).

**Exeptional flow**

- Player collides with a collidable object (O) and cannot continue.

## Use case: *RunGame*

**Short description**
Starts a new game

**Priority (high, mid, low)**
High

**Extends or Includes (other use case)**
Includes UC: MovePlayer, UC: FireWeapon, UC: SelectWeapon

**Participating actors**

- Player

**Normal flow of events**

| **Player (P)** | **System** |
|---|---|
| P starts the game | |
| | A new game is started |

## Use case: *ExitGame*

**Short description**
Quit to desktop.

**Priority (high, mid, low)**
High

**Extends or Includes (other use case)**
N/A.

**Participating actors**

- Player

**Normal flow of events**

| **Player (P)** | **System** |
|---|---|
| P presses exit | |
| | System quits |

## Use case: *FireWeapon*

**Short description**
When the player fires a weapon.

**Priority (high, mid, low)**
High

**Extends or Includes (other use case)**
N/A.

**Participating actors**

- Player (P)

**Normal flow of events**

| **Player (=P)** | **System** |
|---|---|
| P presses action key | |
| | The currently selected weapon fires and a projectile spawns. It continues to move until it collides with a collidable object, or until its range runs out. If collision with zombie the zombie looses health and the projectile disappears. |
| P releases the given key(s) | |
| | Player stops firing selected weapon. |

**Alternate flow: special tile**

- If the projectile collides with a portal it turns up at the other portal (given that there is another one) and its direction continues.

**Exceptional flow**

- Weapon can't be fired if player is out of ammo.
- If the projectile hits and destroys the last enemy, a new wave (round) of enemies is created.

## Use case: *FirePortalGun*

**Short description**
When the player fires the portal gun.

**Priority (high, mid, low)**
Mid

**Extends or Includes (other use case)**
Extends UC: FireWeapon

**Participating actors**

- Player (P)

**Normal flow of events**

| **Player (=P)** | **System** |
|---|---|
| P presses and releases action key with portal gun chosen | |
| | The portal gun fires a single portal of type A or type B, depending on which type is selected from the weapon menu. |

**Exceptional flow**

- If player shoots a portal on another active portal, nothing happens.
- If player shoots portal facing a wall (or other solid object), the player will be bumped away one tile and the portal will take the space.