Requirements and Analysis Document for GTFA

Samuel Utbult, Anton Levholm, John Andersson, Marcus Eliasson

26 May 2016

Contents

1 Introduction

1.1 Purpose of application

The purpose of the application is to be a game where the player is chased by police cars. The player should avoid the police as long as possible, as longer time survived results in higher score. There should also be obstacles that all cars should avoid.

1.2 General characteristics of application

The game is a two-dimensional game where the world is seen from above. The player is presented a small section of the world in a selected instance. The player should be located in the center of this section. The environment is presented as triangles with a single color each. The cars are represented as sprites. The player controls their car with the arrow keys.

1.3 Scope of application

The game can simultaneously be played by a single player and has no support for multiple players, neither online nor offline. The police cars are controlled by artificial intelligence. The application saves the player's score when a playthrough is lost and the application has the functionality to show the highest scores registred on the computer.

1.4 Objectives and success criteria of the project

- The player should be able to drive their car around the map while not being able to drive through the obstacles that has been placed on the map.
- The game should contain police cars that will chase the player. These should not be able to drive through obstacles neither.

2 Requirements

2.1 Functional requirements

- The player will drive a car around
- A number of police cars are chasing the player
- All the cars will be represented as sprites
- The environment will be represented as single colored triangles
- No cars should be able to drive through environmental objects that are solid (ie. fences, houses, streetlamps, etc)

- The longer time the game runs, the higher score the player acquires
- The game will end when a police car touches the player
- The current score will be shown during the duration of the game

2.2 Non-functional requirements

2.2.1 Usability

A regular desktop computer user shoud be able to play the game with their keyboard. The application should communicate the state of the game in a clean and simple way and eliminate all ambiguity. The text labels will only be avaliable in english.

2.2.2 Performance

The game should not process a large amount of heavy computations. As with every game, every frame should be fast to process to keep the number of frames per second as high as possible. This is a simple two-dimensional game without any complex calculations. Therefore, this will probably not be an issue.

2.2.3 Supportability

The game logic should be separated from the input and graphics framework as much as possible, as this will achieve portability between libraries and therefore platforms. The libraries should be fully replaceable and not affect any game logic. The game logic sections should also have extensive unit tests.

2.2.4 Implementation

The game will be implemented in Java and Java Swing will be used for graphics and input management. However, the Java IO-environment should be easily replaceable with other graphics libraries, such as Android's IO-environment.

2.2.5 Packaging and installation

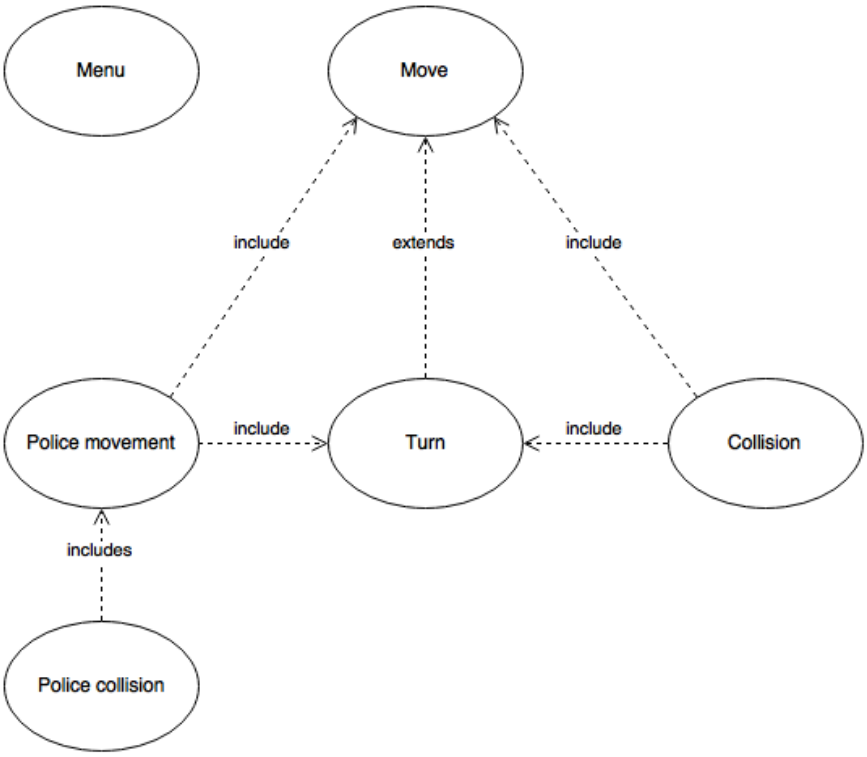
The game will be packaged in a single JAR-file together with the necessary dependencies and resources, to achive great portability.

2.2.6 Legal

The only legal issue arises around the game sprites. The sprites has been aquired through a website containing free sprites. This brings up the question of what license under these images were published. However, this is only an educational project, and therefore, this should not be any problem.

2.3 Application models

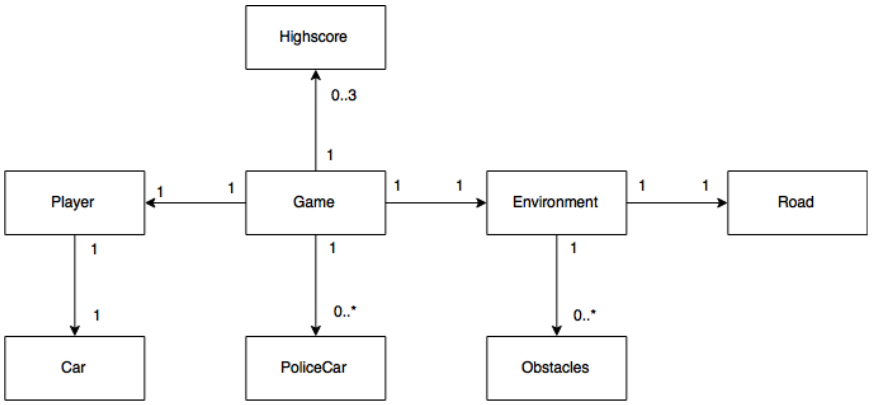
2.3.1 Use cases



These are the use cases of the application. They are presented in the selected priority. For more information about the use cases, see the appendix.

1. Move
2. Turn
3. Collision with environment
4. Police movement
5. Police collision
6. Menu

2.3.2 Domain model



2.3.3 User interface

The application will use a standard desktop graphical interface following standard conventions. The user interface will be resizeable and it will adjust for different screen sizes.

Appendix

Use cases

Move

Summary: This is how the cars move in the game. Either as an effect of pressing, or releasing, a key or as an effect of the game still running.

Priority: High.

Extends:

Includes: Turn.

Participators: The player

Normal flow of events
A simple accelerating movement.

Table 1:

Actor	System
The player press the up key	The car accelerates forward if standing still or already going forward and brakes if going backwards.
The player press the down key	The car brakes if the car goes forward and the car accelerates backward if it is standing still or already going backwards.
A car has velocity	The car moves forward
Subcase: The car goes into a solid object	The car stops
A police car collides with the player	The game is over
The game is still running	Increase the player's score
The player presses left or right while car is moving	Use case: Turn

Turn

Summary: When the player presses the left or right arrow key the car will change its direction.

Priority: High.

Extends: Move.

Includes:

Participators: The player.

Normal flow of events

Table 2:

Actor	System
Player press left arrow key when moving forward.	The car turns to the left.
Player press left key without moving forward.	The car stands still.
Player press right key when moving forward.	The car turns to the right.
Player press right key without moving forward.	The car stands still.

Collision

Summary: When the player hits an object in the environment.

Priority: High.

Extends:

Includes: Move, Turn

Participators: The player.

Normal flow of events

Table 3:

Actor	System
The player hits an object.	The players car stops.

Police movement

Summary: How the police follows the player using computation.

Priority: High.

Extends:

Includes: Move, Turn

Participators:

Normal flow of events

Table 4:

Actor	System
The game starts.	
(Loop begin)	Police calculates the players position.
	Police calculates direction towards player.
	Police use included use-cases Move and Turn to drive towards player.
(Loop ends)	

Police collision

Summary: How the police acts when encountered with an object in the environment.

Priority: High.

Extends:

Includes: Police movement.

Participators:

Normal flow of events

Table 5:

Actor	System
Police encounters an object.	Policecar reverse for a set time.
	Use case: Police Movement.

Menu

Summary: What happens when the player interacts with the menu.

Priority: Medium.

Extends:

Includes:

Participators: The player.

Normal flow of events

Table 6:

Actor	System
Player presses down key	Menuindex moves down to next menuitem.
Player presses up key	Menuindex moves up to previous menuitem.
Player presses space key	Check menuindex.
	Update the view with information held by the index.