Requirements and Analysis Document for Achtung, die Klonen (Group 7)

Contents

Version: 2

Date: 2013-05-16

Authors:
Jakob Csörgei Gustavsson, Niklas Helmertz, Lucas Persson, Joel Torstensson

This version overrides all previous versions.

1 Introduction

Achtung, die Klonen is a game based of a classic game from 1995 called *Achtung, die Kurve*. It is a game where multiple players compete against each other by controlling their own snake in such a way that it doesn't collide into a wall or a part of another snake's, including oneselves, body.

1.1 Purpose of application

To be a version of *Achtung, die Kurve* for the desktop, with good performance and unique gameplay compared to similar products. It's intended to be used by a group of people gathered around a single computer for pure entertainment purposes.

1.2 General characteristics of application

A stand-alone desktop application with local multiplayer functionality and a graphical user interface made to work on Windows, Mac OS X and Linux.

The game has rounds wherein the game progresses at a constant rate in real-time. The player can control its snake by using keys defined by the player at the start of the game. Once there is only one player left in a round, that player has won that round. Points are awarded linearly from 0 to the amount of player minus one. 0 points are given to the player who dies first, while the player who survives the longest gets the most points. The rounds continue until one of the players has [10 * (amount of players -1)] points.

The style is fairly minimalistic, with pronounced colors and a lack of textures. (See GUI example in APPENDIX.)

1.3 Scope of application

* Local multiplayer gameplay.
* A number of power-ups.
* player defined players and controls.

Players will not be able to specify a specific power-up to use before the game starts, as seen in other versions of Achtung, die Kurve.

Due to lack of time we will not be implementing any kind of online multiplayer.

1.4 Objectives and success criteria of the project

Our goal is to create a fun, stable game which supports local multiplayer to some extent. We feel that power-ups are a must to provide variety to the game, which makes it an essential part of the finished product. You should also be able to configure a few select settings to customize the game experience to your liking.

Other features that we wish to add, but are not quintessential to the core gameplay, is support for different maps, statistics, persistent player accounts and teamplay; essentially in that order of priority. After the product has been "finished" according to our criteria, we will attempt to implement as many of the above features as possible.

1.5 Definitions, acronyms and abbreviations

Achtung, die Kurve - an online multiplayer game developed in 1995 wherein multiple players using the same computer compete by controlling their own snake so that it doesn't collide with anything. At start, the snakes are only a single dot, but as the head moves, a trail referred to as the body is left behind in every point that the snake visits.

Power-up - an object that can be picked up by a player and causes an effect on something in the game. For example a speed boost which makes the player move at a faster pace than normal.

Lightweight Java Game Library (LWJGL) - a game library for java that amongst other things gives developers the ability to utilize the GPU using OpenGL.

Open graphics Library (OpenGL) - a very common Graphics Library that's basically an API that has been implemented in most modern graphics hardware.

Graphics processing unit (GPU) - a hardware circuit specialized for graphics rendering.

Nifty-GUI - a library for easily constructing menus and graphical user interfaces for Java games.

2 Requirements

2.1 Functional requirements

The players should be able to:

1. Select the amount of players for playing, their colors, names and keys used to control them.
2. Select the map to be played on as well as how many points the limit should be for winning.
3. Start a new game. Start the first round.
4. Move
   a) Turn right
   b) Turn left
5. Pick up power-up.
6. End the current round and move to the next.
7. Exit from the on-going game. Takes the player to the main menu.
8. Exit the application.

2.2 Non-functional requirements

2.2.1 Usability

Any player should easily be able to learn the rules of the game and how the application works in a short amount of time. Documentation, with instructions, of some sort will be either be in the game or available separately.

2.2.2 Reliability
NA

2.2.3 Performance
It should be able to run smoothly on basically any somewhat modern computer. Since the graphical component of the game is fairly simple, it shouldn't be much of a problem. The limiting factor would be the OpenGL version supported in the GPU. Even older computers that can only just support it should be able to run it quite smoothly.

2.2.4 Supportability
The application will support Windows, OS X and Linux without any modifications.

The implementation should allow for easily replacing the existing GUI and rendering components so that different GUI and/or rendering libraries can be used. Porting it to for example the Android platform should be possible without much modification to the non-graphics related code.

The implementation of the power-up system should be sufficiently modular so that a new power-up can be implemented with little or no modification to the existing code.

2.2.5 Implementation
Platform independence will be achieved using the Java Runtime Environment (JRE), which will have to be installed on all host computers running the game. The game will also use the LWJGL-library for rendering, which includes native components for Windows, Linux, Mac OS X and Solaris.

2.2.6 Packaging and installation
Downloadable archive containing a .jar file as well as scripts that can be used for running it on various platforms.

2.2.7 Legal
The included music and licences:
Author, name, link, licence
Fantastic Vamps, Borderline (Fantastic Vamps : 8-Bit Mix), http://ccmixter.org/files/vamps/8749, https://creativecommons.org/licenses/by/2.5/
ParagonX9, SM64 - The Alternate Route http://www.newgrounds.com/audio/listen/11245, https://creativecommons.org/licenses/by-nc-sa/3.0/

2.3 Application models
2.3.1 Use case model
See APPENDIX.

2.3.2 Use cases priority

1. Move
2. Turn
3. Collide (included here CollideWithSolid)
4. PlayerDeath
5. EndRound
6. CollideWithPowerUp
7. DoRound
8. SetPlayers
9. SetGameSettings

2.3.3 Domain model

See APPENDIX.

2.3.4 User interface

The first screen the player is presented with after starting the application contains all the tools needed to start a game, right away. If all use-cases are implemented you can choose:

- How many players you want in the game and which buttons each player should use
- Which map you want to play on
- How many points are required to win (by default this is set to 10 * (n-1) where n is the amount of players).

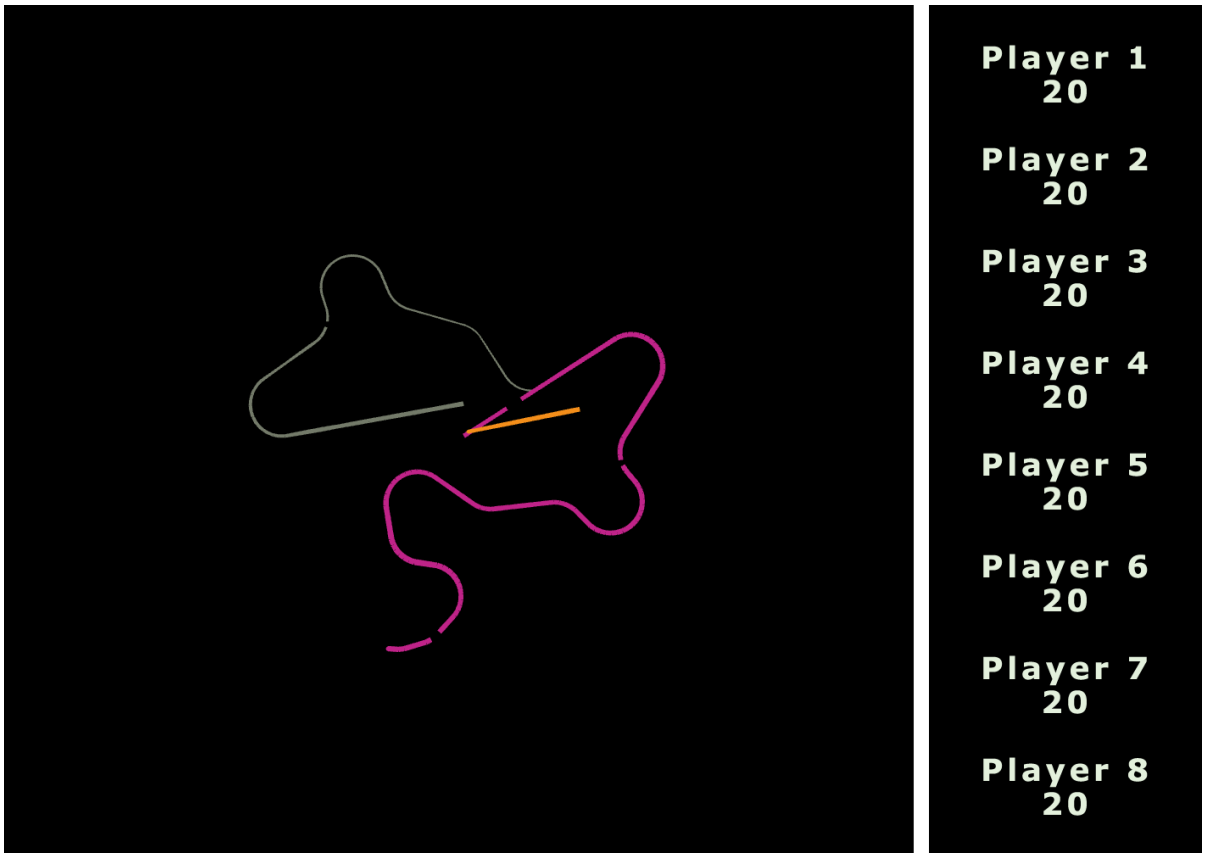
The game part consists of a plain black background where the players can move around freely. To the right a list of all the players with their respective score can be seen (see GUI in APPENDIX for an example).

2.4 References

Popular flash version: http://jesper.nu/spel/achtung-die-kurve
Wikipedia Article: http://en.wikipedia.org/wiki/Achtung,_die_Kurve!

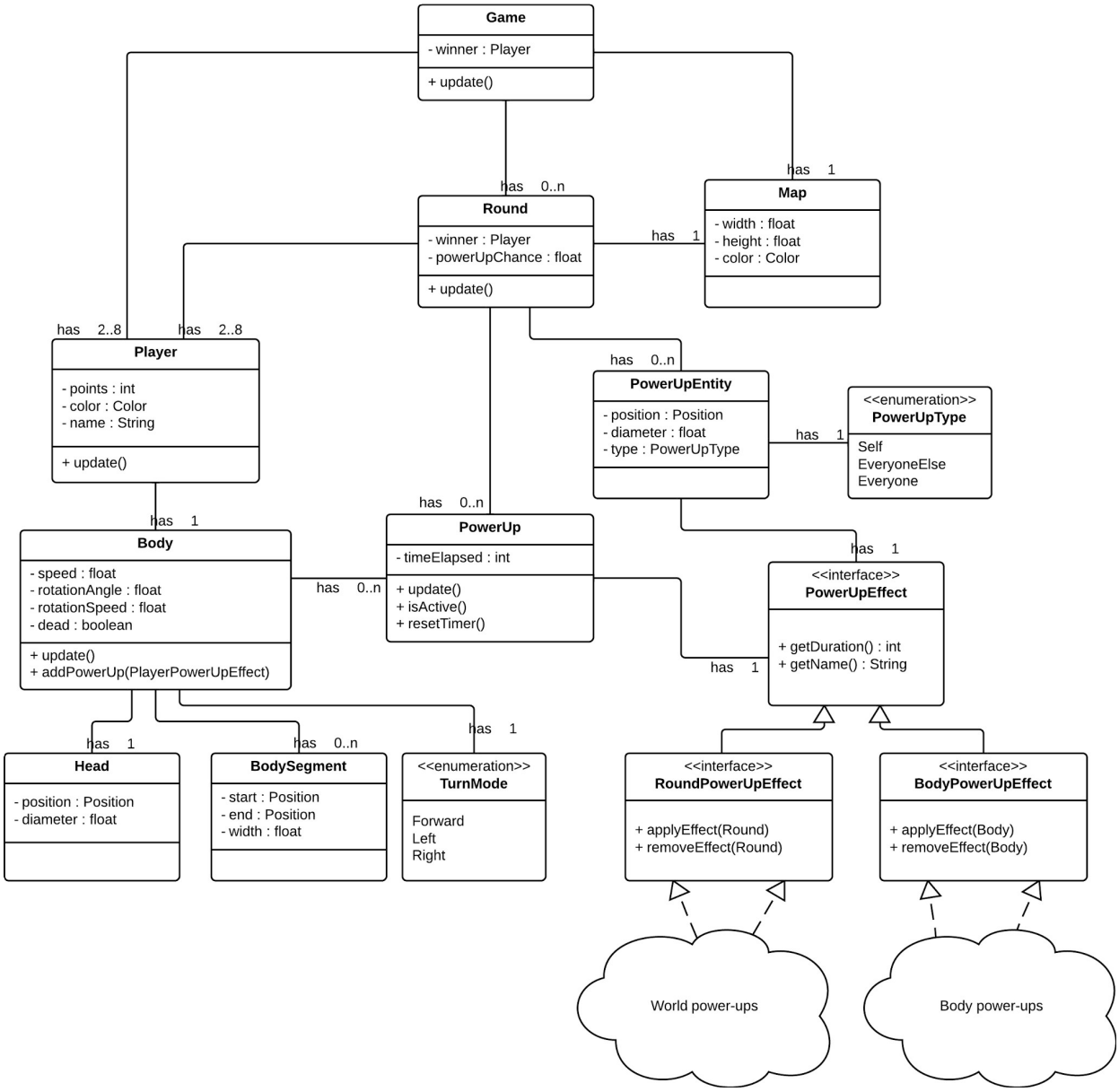
APPENDIX

GUI

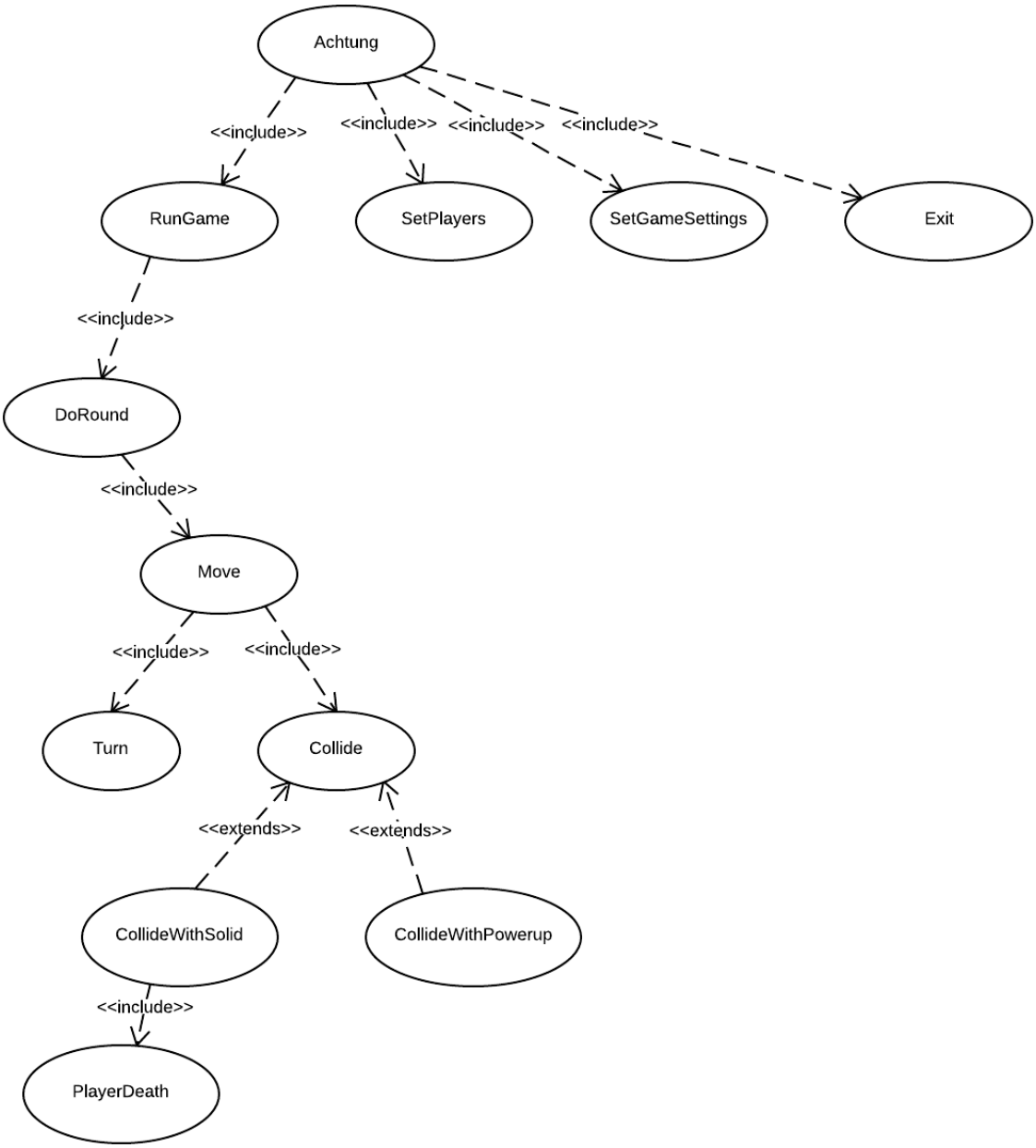


Simplest possible GUI mockup that fills the requirements.

Domain model



Use Cases



Use case texts

Use Case: Collide (Abstract)

Summary: A player collides with something

Priority: high

Extends: -

Includes: CollideWithSolid, CollideWithPowerup

Participators: One of the players

Normal flow of events

	Actor	System
1		Responds with the appropriate action depending on what has been collided into (defined in included UCs)

Use Case: CollideWithPowerup

Summary: A player collides with a powerup.

Priority: medium

Extends: -

Includes:

Participators: One of the players

Normal flow of events

	Actor	System
1		Powerup disappears.
2		The effect of the powerup is applied to the player.

Use Case: CollideWithSolid

Summary: A player collides with something solid (such as a player's body or a wall)

Priority: high

Extends: -

Includes: PlayerDeath

Participators: One of the players

Normal flow of events

	Actor	System
1		The player dies (PlayerDeath is caused)

Use Case: PlayerDeath

Summary: A player dies.

Priority: high

Extends: -

Includes: EndRound

Participators: One of the players

Normal flow of events

	Actor	System
1		The player's snake is frozen in its current state and the player can no longer control it.
2		One point is awarded to every remaining player.

Alternate flow

Flow 2.1 EndRound (If there is only one player left after death)

	Actor	System
2.1		The round ends (use case EndRound is caused)

Use Case: EndRound

Summary: The round ends.

Priority: high

Extends: -

Includes:

Participators: All the players.

Normal flow of events

	Actor	System
1		Text is shown displaying who won the round.
2		A keyboard prompt is shown (e.g: “Press space to continue”).
3	Any player presses the requested key.	
4		A new round is started.

Alternate flow

Flow 2.1 If the win condition set prior to game start is reached (e.g. a player reaches 40 points)

	Actor	System
2.1		Text is shown displaying who won the game.
2.2		Further stats are shown (?, such as gametime etc). Prompt to start a new game with the same settings or return to main menu is displayed.
2.3	Player selects an	

	option.	
2.4		The appropriate action is taken.

Use Case: Move

Summary: A player moves.

Priority: high

Extends:

Includes: Collide

Participators: One of the players

Normal flow of events

	Actor	System
1		Extends the player's snake by a given constant in the direction the snake is heading.

Alternate flow

Flow 2.1 Turn

	Actor	System
2.1	Indicates in what direction the snake will turn by holding down the player's left or right key.	
2.2		Extends the player's snake by a given constant in the direction the snake is heading.

Exceptional flow

Collide

	Actor	System
3.1	The snake moves in a direction that makes it collide with another object.	
3.2		The appropriate collision action is taken. (See Collide)