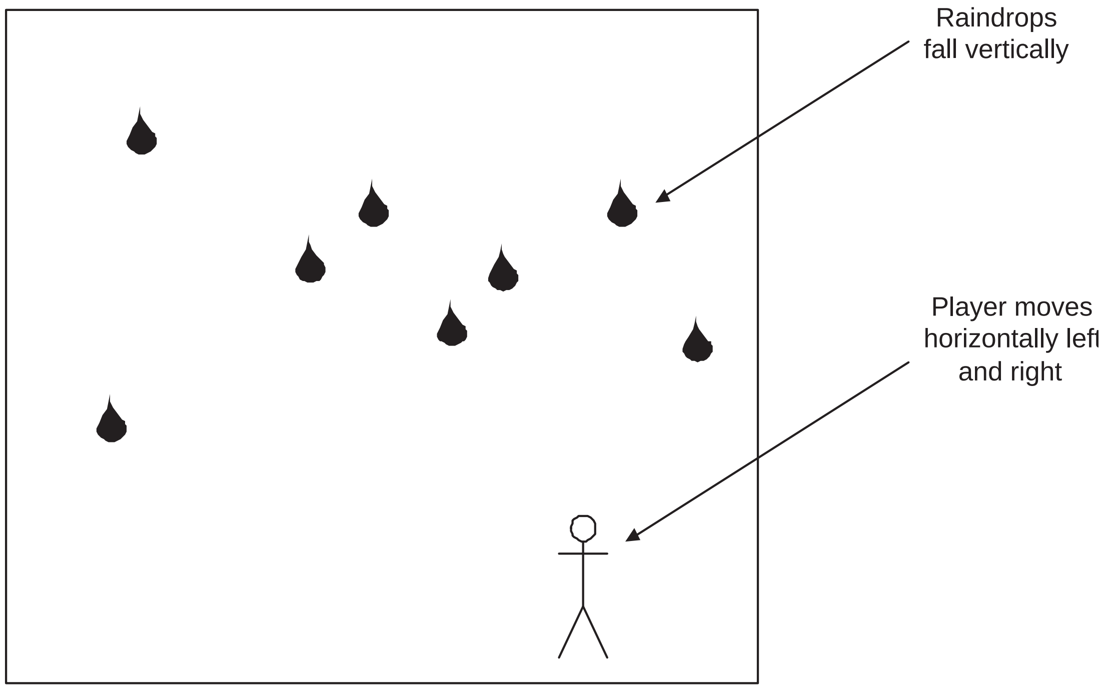
UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG
School of Electrical and Information Engineering
ELEN7045 - SD Methodologies, Analysis and Design

Individual Assignment I

Software Design and Modifiability: The Dodge Game

1 Introduction



The figure above depicts a screen shot from a game called “Dodge”. Dodge consists of:

- A single player who can only move left and right and is controlled via the keyboard.
- Raindrops which appear in random positions in the top quarter of the screen and fall vertically downwards. The number of raindrops on the screen at any one time remains constant — whenever a raindrop disappears off the bottom of the screen, a new one appears at the top of the screen.

The objective of the game is to move the player so that he or she does not get wet, that is, the player must dodge the raindrops. If a raindrop touches the player then the game is over.

2 Deliverables

This is an individual assignment — each student is required to work on their own.

2.1 Executable and Code

Provide a working implementation (executable) of the Dodge game along with the complete source code. Your source code needs to be well commented and/or easily understandable.

Your solution must be object-oriented but the choice of OO language and graphics library is up to you. Bear in mind the following constraints:

- your solution must be able to run on the Windows platform,
- you may not use an existing game framework.

If you are an experienced developer then take this opportunity to learn a new language. If your development skills are not that strong then choose an OO language that you are familiar with.

You should use *simple* graphics for the game elements. *The aim of this assignment is not to produce a game with sophisticated functionality but rather to create a well thought out design for the game which will easily support certain changes in requirements (see below)*.

2.2 Documentation

Produce, and submit, a *hard copy* of a short report (no more than 5 pages, excluding title page, abstract and references) which:

1. presents and critiques your design, and
2. discusses your game's *modifiability*.

In presenting your design clearly identify the key objects and classes involved, their responsibilities, and how they collaborate with each other to provide the game's functionality. You are also required to examine your design critically and motivate the design choices that you have made.

Modifiability is a quality attribute of a software system which refers to the ease with which the existing design can be modified to meet new requirements. The ideal situation is to meet new requirements with new code — not by re-writing existing code. However, often this ideal cannot be achieved and we try to localise code changes to as few classes or modules as possible. It is important to realise that a design cannot support all changes equally and some changes will require an extensive re-write of the code.

A good way of evaluating the modifiability of a software system is to consider scenarios for future change. In this game, for example, the following requirement changes might occur:

1. A new game element is introduced. This is an umbrella which falls in a similar manner to the rain drops. However, if the player "catches" the umbrella, he or she receives an extra life.
2. The keys which control the player need to be configurable.
3. The game must change from a one player version to a two player version. Both players must simultaneously dodge raindrops. The game ends when one of the players is hit.
4. Scoring is introduced and the top three highest scores need to be saved.
5. A slightly different version of the game is needed in which the raindrops become hailstones and the player becomes a dog. The dog moves left and right in order to avoid being hit by the hailstones.

6. The game is converted to 3D with the player being able to walk in three dimensions.
7. A different graphics library must be used in place of the existing one.
8. The game must run on Linux.

Which of these futures does your design naturally support and which does it not? Justify your answers by discussing how your application's code will have to be modified (note, you are not required to actually implement any of these scenarios). Also list and discuss any additional change scenarios which are supported by your design.

3 Deadline and Submissions

The deadlines for all the course deliverables are available on the course homepage.

All submissions must be in strict accordance with the guidelines contained in the *School's Blue Book* and the rules contained in the *School's Red Book*. Please note the minor changes with regard to the late submission policy in Section 3.1.

3.1 Late Submission Penalty Policy

Submission are due at the start of the lecture on the deadline day. You may submit the following day to the front desk before the close of business (approximately 4pm) but you will be penalised by 15 percentage points. No submissions after this will be accepted.

3.2 Plagiarism

Refer to the *School's Blue Book* for an explanation of what plagiarism is and how to avoid it.

All instances of plagiarism from either the internet or within the class will be severely dealt with. No two students may have identical or overly similar reports. No two students may have identical or overly similar source code.