

# CST8110: Introduction to Computing

## Lab 7: Designing a Simple Game

### Overview

Your textbook has five sample programs that explain C++ language constructs using a simple game theme. These sample programs are in the Word document I produced for Chapter 3. They are numbered 3-15, 3-17, 3-19, 3-21, 3-22 in this document.

All five programs assume that a player moves in 2-dimensional space; and the player's position is tracked through *int* variables for *x* and *y*.

You will use this design as a foundation for your game development.

### The Details

You will complete the three preliminary design steps:

- Problem Statement and Requirements (Assumptions)
- Algorithm
- Test Plan

And these three parts are completed *before* you begin any C++ coding.

#### Problem Statement and Requirements (assumptions)

This section will be very important. By working through the *Problem Statement*, you really define your game.

The modifications to the game will be of your own choosing, though I will give several constraints.

- There will only ever be a single player.
- The player will only be able to move one position *North*, *South*, *East* or *West* on a single move.
- The player will have an initial health-point level captured as an integer value. The health-point value will increase and decrease in some way (to be decided by your team). When the health-point reaches 0 (or possibly goes negative), the player dies and game play ends.
- You can assume that you will have access to a random number generator (which is important in gaming).

The problem statement section must be detailed enough to give clarity to changes you intend to make. There are several questions you must consider:

- How is the game won? Is it finding treasure or reaching some high value health-point?
- What causes the health-point value to increase (it can be more than one thing)?
- What causes the health-point value to decrease (it can be more than one thing)?
- How is the game lost (is there another way in which the game can be lost besides the expiry of health-point described above)? Is there a time limit? (Later, I will provide sample C++ code to show how to manage timing.)
- Are there boundaries to the area in which a player can move? If so, what are the consequences if the player encounters a boundary.
- Does the game use sound effects? (Later, I will provide sample C++ code that implements sound: .WAV files only, not MP3)

Before proceeding to your algorithm design, I would like to review your statement of assumptions to insure that you have clearly scoped the nature of the game. There is a risk that you might propose a development project that is too complex given the amount of time you'll have available.

#### Algorithm

Your algorithm is a set of detailed step-wise instructions. You will use a few basic instructions to achieve a problem solution, including: *GET*, *PUT*, *←*, *IF-ENDIF*, *IF-ELSE-ENDIF*, *IF-ELSE IF-ENDIF*, *WHILE-ENDWHILE*, Relational and Boolean operators. You can add *FOR*, *DO-WHILE* and *SWITCH-CASE* to your list of algorithmic tools

#### Test Plan

You will need to run through your algorithm with sample data several times to check for errors. Each sample run should test one or more of the following issues:

- *Out-of-order statements*: Does something happen during a test run that you where not expecting or violated an assumption?
- *Incorrect Steps*: Did some individual action do the wrong thing?
- *Missing Information*: Did you forget to get information from the user? Are you making an assumption that may not be valid?
- *Boundary Conditions*: Are input values supposed to be within a certain range? Did you check for that? How does a repeating (looping) section know when to stop?

#### Note:

You will not be writing C++ code at this stage. That will come after I've seen your problem statement (assumptions), your algorithm and your test plan.

### Submission Standards

- Cover page:
  - Course Name / Number (including Lab Section)
  - Lab Number
  - You and your partner's names
  - Date of Submission
- Problem Statement / Requirements (Assumptions)
- Test Plan
- Algorithm

### Submission Process

- The *Problem Statement (Assumptions)* should be shown to me as soon as possible for an informal review. This phase concerns the scope of your proposal.
- When all three components (*Problem Statement*, *Test Plan*, *Algorithm*) are complete, staple them together with the marking guide / evaluation sheet.
- Due Date Posted on Website.

### Evaluation Criteria

See the attached marking guide.

## **CST8110: Introduction to Computing Lab Exercise 7: Designing A Simple Game: Evaluation**

Your lab assignment has been evaluated using the following criteria:

- Problem Statement and Requirements (Your Assumptions) (3 marks)
  
- Test Plan (2 marks)
  
- Algorithm (4 marks)
  
- Format Organization (1 mark)

**Subtotal /10**

- Penalties (up to 3 marks deducted)
  
- Bonuses (up to 3 marks added)
  
- Adjustment for unequal effort by individuals (unlimited number of marks added or subtracted)

**Total /10**