GADE6122-PART 1 Referencing

Neo Kgobe -ST10315059

Mighty Jarvis-ST10450679

Richard Mtoko

Table of Contents

[Making the Hero Game by using design and programming concepts 3](#_Toc207993554)

[Reference list 4](#_Toc207993555)

## Making the Hero Game by using design and programming concepts

The Hero Game project shows how to blend computer logic and structured design by making a turn-based game with C# Windows Forms. The project's main goal is to let a hero character move around a grid-based level, avoid anything in the route, and find a tile that leads to the exit. This simple yet flexible system shows how important object-orientated programming, modular programming, and control structures are. Farrell (2021) says that good programming requires breaking issues down into smaller parts and always using sequence, selection, and repetition structures. These concepts are the basis for the Hero Game, which also uses some of the broader software design principles that Nystrom (2014) talks about.

The project is based on the Level class. The map appears like a grid of tiles with two sides. A tile might stand for a wall, an empty area, a hero, or a path to escape away. This design illustrates Farrell's (2021) concept of "top-down modularisation", wherein the game is segmented into coherent components, each serving a distinct function. Nystrom (2014) stresses how important inheritance and polymorphism are for making code reusable and cutting down on duplication. Because of this, WallTile, EmptyTile, HeroTile, and ExitTile all come from the abstract Tile base class. By adding rendering to the Display property, the software makes sure that each tile is clear and can be changed.

Another method to explain how control systems work is to look at how movement works. The game informs you which way to move (up, down, left, and right), and those directions are utilised to change the coordinates. The system checks to see if the player's selected position is on the grid and if they can cross it when they try to move. The hero goes to the next level when they reach the exit. The hero doesn't move if there is a wall. Farrell (2021) calls this form of decision-making "selection logic", which means that the computer looks at specific conditions to decide what to do next. This style of thinking makes sure that players always perform things that are easy to understand.

Loops are an important part of how the game works. If the game requires the hero or exit to be in a random area, it keeps randomising until it finds a tile that isn't already occupied. Farrell (2021) talks about repetition structures, which show that people follow instructions until a certain condition is met. This is very much like what you mentioned. The game's main loop, which includes accepting input, moving the hero, and redrawing the grid, is similar to how all interactive apps work. The Hero Game is a great illustration of how to use basic reasoning and good programming. Nystrom (2014) talks about the State pattern, which says that the software works differently based on its state, such as "In Progress", "Complete", or "Game Over". You can keep track of how the game is progressing by using a GameEngine class. Separating the game engine from the user interface is another part of decoupling. This makes it simple to keep the code up to date and add new features like opponents or items in the future.

In short, the Hero Game is an interactive game that uses computer logic and principles from object-orientated design. According to Farrell's (2021) ideas on order, choice, and repetition, the project has a clear set of rules that control movement, choices, and loops. The project shows how little games may benefit from professional software design methods by applying Nystrom's (2014) ideas on patterns and modularity. All of these things work together to make sure that the Hero Game is both a great method to learn and technically sound.

## Reference list

Farrell, J. (2021). Programming Logic and Design, Comprehensive. 10th ed. Cengage Learning.

Nystrom, R. (2014). Game Programming Patterns. Genever Benning.