Table of Contents

[**User Guide** 2](#_Toc503358278)

[**Summary:** 2](#_Toc503358279)

[**Controls:** 2](#_Toc503358280)

[**Release Bugs:** 2](#_Toc503358281)

[**Implementation check list** 3](#_Toc503358282)

[**Adequate Rated Items (D to C)** 3](#_Toc503358283)

[**Good Rated Items (C to B)** 3](#_Toc503358284)

[**Excellent Rated Items (B to A)** 4](#_Toc503358285)

[**Extra Marks (Examples)** 5](#_Toc503358286)

[**Maintenance Guide** 5](#_Toc503358287)

[**Graphics/Visualization System:** 5](#_Toc503358288)

[**World Model System:** 5](#_Toc503358289)

[**Conclusion** 6](#_Toc503358290)

# **User Guide**

## **Summary:**

The demo is for a 2D RPG game where the player is spawned on a tiled map along with one enemy chicken. The all enemies have an alert zone. If the player is in a certain distance from the enemy, the enemy will start chasing the player until it collides, or the player manages to escape the alert zone. If the Enemy manages to collide with the player, they go into battle. At least one more enemy joins the battle as well. The battle is turn based, the player attacks first, and then the enemies until the battle finishes. The enemies have a chance to miss to make it a fair fight the player currently never misses. In order to win the game, the player must fight the enemies and defeat them. The enemies are always numbered in order so if an enemy dies the next enemy takes its number. If the player dies, they lose the game.

## **Controls:**

The player moves using the W, A, S, D keys or if an XBOX controller is plugged the A, B, Y and X keys on the controller can be used to move the player. When in battle the player currently can only select the attack option. To select they need to use the W and S keys for moving the selection box up or down the options and press space once the selection box is on top of the option they need to select. When attack is selected the enemy names and their health appear on the Hud. The player can select which enemy to attack as well.

## **Release Bugs:**

The player's movement is not capped at a specific speed so depending on the frames per second the player will move in different speeds. Due to lack of time a model tick on the player’s movement wasn’t implemented. A model tick was added to the movement of the selection box and making selections in order to prevent the selection to happen multiple times within a single click. As holding space down would result in selections being made every second because of the frame limitation by the model tick.

## **Implementation check list**

### **Adequate Rated Items (D to C)**

**Graphics**

* A ‘black boxed’ graphic system is in place **(Done)**
* Textures can be efficiently drawn to arbitrary positions on and partially off the screen (clipped) **(Done)**
* Animation is implemented and working correctly **(Done)**

**World State**

* A player entity exists **(Done)**
* Input is recognised and can be used to alter the world state e.g. move the player entity **(Done)**
* The Xbox controller is supported and ‘hot pluggable’**(Attempt)**

**Code Quality**

* Class interfaces are minimal and complete. Class function and member variable visibility is correct **(Attempt)**
* Code can be built and executed without compiler errors or warnings in debug and release
* Code is well commented **(Attempt)**
* There are no memory leaks
* There is good error handling throughout **(Attempt)**
* You have followed all the submission requirements e.g. made a video, submitted the correct files etc. **(Done)**

**Report**

* All requested sections have been attempted adequately and the report is professionally presented **(Attempt)**

### **Good Rated Items (C to B)**

**World State**

* A world model system is in place. It is separate from other code and black boxed**(Attempt)**
* There is a game loop handling input, world update and rendering **(Done)**
* Bounding rectangle collisions are detected **(Done)**
* There are multiple world entity types **(Done)**

**Code Quality**

* Good use of object oriented techniques e.g. polymorphism, member variable visibility **(Attempt)**
* Memory is only allocated / deallocated outside of the game loop
* Const is used correctly**(Attempt)**

**AI**

* Some AI routines are in place e.g. enemy entities move around the world following paths, use state machines etc. **(Attempt)**

**Report**

* This report would allow another programmer to work with your code systems **(Attempt)**

**Other**

* Some sound effects are in place

### **Excellent Rated Items (B to A)**

**Graphics**

* Interpolation is used to smooth entity movement

**World State**

* The player entity can shoot projectiles (or equivalent functionality)
* Explosion and bullet management
* Game play is independent of platform capabilities (i.e. uses a model tick approach) **(Done)**
* Game cycling e.g. detection of win / lose conditions and restarting the game **(Attempt)**
* There is a scoring system with the score shown on screen **(Attempt)**

**AI**

* Several different enemies with differing behaviours **(Attempt)**

**Report**

* This report has insightful and balanced reflection **(Attempt)**

### **Extra Marks (Examples)**

* Mapping of world space on to screen space
* Other graphics techniques have been implemented e.g. background scrolling, blending modes etc.
* Level data is loaded from a file **(Attempt)**
* A difficulty level
* More advanced C++ e.g. use of namespaces, STL, C++ 11 and further patterns **(Attempt)**
* A\* algorithm
* ‘Intelligent’ enemy behaviour
* There are sound effects for collisions, explosions and firing
* Additional black box systems have been implemented e.g. for AI, Sound **(Attempt)**
* HUD features beyond simple text e.g. health bars, mini maps etc. **(Done)**
* Other features, please list below:
  + Greedy Best-first search algorithm for 4-directional pathfinding. **(Done)**
  + Diagonal path finding implemented using Vectors to find the heuristic. **(Done)**
  + Black boxed Pathfinding system. **(Done)**
  + UI implementation. **(Done)**
  + Loading map details from an XML File. **(Done)**

# **Maintenance Guide**

## **Graphics/Visualization System:**

The Graphics system is black boxed and easy to maintain. This system is not linked directly to the game demo, so it could be easily reused to develop a different type of game. I can state only one possible disadvantages to having a black boxed system. And that is passing data from the black-boxed system to the other systems and vise versa. The fact that most things are encapsulated make it slightly more troublesome to get access to code in the graphics system. The classes in the system are the Graphics class and the Sprite class. The way the system works is The graphics stores a map of sprites with unique names as keys. And each key matches an entity name key. This allows an entity to draw itself by passing that key to the graphics system. After the graphics system obtains the key and the other details needed to draw the sprite, the sprite blit function is called to use all the data and draw the sprite for that entity.

## **World Model System:**

The world model is the most game dependent system, its purpose is to put together all the other systems in the engine in order to build a game. Currently the World Model does all the UI along with Gameplay, Collision detection and Input. The system itself isn't very reusable but its non-game dependent components are. For example, the UI can easily be reused to make a completely different game, along with the collision detection and the key input. Unlike the Graphics system which is black boxed/separate from the Game Demo and all the other classes in the engine, the World Model intertwines with the Game Demo. The problem with that is that debugging is a lot harder because there is a lot more code to go through to get to the problem and any change in the world model impacts everything else in the world. Which makes it a lot harder to program since any change could cause a bug somewhere else.

# **Conclusion**

There are many things that could be improved in the engine and the game demo, starting with error handling, black-boxing the rest of the systems, improving the code quality and efficiency and anything else that makes the code harder to maintain. After dealing with everything mentioned above I would add a sound system, an AI system, more depth to the gameplay(Adding intelligent enemy behavior, bosses, interpolation for the path following, villages, NPCs, Quests, Dungeons, etc.), maybe separate the game engine from the game so that it can be used to make different genres of 2D games. Throughout the development of the game I’ve learnt that pseudo coding and diagrams could make a big difference in the development speed and making the code easier to maintain and bug free, and that time is never enough when making a game because there’s always more to add and improve. At this stage I believe the game demo I’ve created lacks a lot of basic things that most games have like a start menu with options, story, solid goal for the player to pursue, engagement, etc. Despite that I believe I’ve set the stepping stones to build a unique RPG game that in time could become a success.