CS4303 P2

150006108

October 2018

1 Overview

The aim of this practical was to create a Roguelike dungeon exploration game, where the dungeon consists of many procedurally generated floors containing treasure and enemies for the player to fight. New items can be found and used to help the player advance through the dungeon, until they either win or die.

2 Aim of the Game

The premise of my game is simple. Traverse 5 levels and reach the end whilst obtaining the highest score possible. Points are won by collecting treasure, advancing floors, and defeating enemies. The harder the enemy, the more points awarded. Points also allow the player to "level up" (more on that later).

When the player either dies or reaches the end of the fifth level, and end screen is displayed asking the player whether or not they want to start again.

3 The Dungeon

The dungeon consists of 5 floors, as previously mentioned. Each floor is a square. If you take the floor number to be n, the length of the sides of the square are $10n$ (floor 3 has a width of 30, floor 4 has a width of 40, etc). The dungeon level is represented by a 2D array of *Block* objects, objects that contain all the required information for that grid space within the level. This includes the *BlockType*, which is an enumerable representing whether a grid space is a *WALL*, *FLOOR*, *START* or *FINISH* block.

Each dungeon floor is procedurally generated using a "Random Walker". At the start, every grid space if filled with a *WALL* block. The walker will start in a random space, change this block to a *FLOOR* block, choose a random direction, and move in the chosen direction - provided it doesn't break any predefined conditions. This next block, if it isn't already, is then changed to a *FLOOR* type. The walker will continue drawing out a path until another preset condition is

met. For my game, this terminal condition was to have at least 50% of the total dungeon floor area traversable. Once this condition is met, the walker stops, and the dungeon level has been generated.

The random walker algorithm ensures that all points will be reachable, as the walker will only ever move one space at a time. This means a continuous path is drawn from where the walker starts to where the walker finishes.

I chose the 50% threshold as anything significantly bigger and the dungeon floor just looked like one empty room with a few patches in it, and anything smaller meant there was very little path branching, limiting the choices of the player in terms of exploration.

Once the level has been generated, the entrance and exit need to be specified. This means simply finding the top leftmost and bottom rightmost floor tiles, and assigning them as *START* and *FINISH* respectively.

Each floor will have two treasures. I originally wanted them to be located at the end of long, thin corridors, and if one is found, a treasure will be placed there, but a lot of the time this is not possible. A block at the end of a corridor would be represented as a block with only 1 neighbouring floor block (as the path is continuous, you will never have two blocks isolated with only each other as their neighbour). If such a block cannot be found, any treasures not yet placed are then placed at a random grid slot.

Enemies are also placed selectively. Some enemies may have a patrol path set, so they will require space to move. As such, enemies are only placed in spaces where all 8 neighbouring blocks are floor tiles also. This means if an enemy has a patrol path, it has the space to move freely.

The amount of enemies found on a floor increases the deeper you go. If the floor number is n, the amount of enemies you will find on the floor is $n - 1$. The difficulty of the enemies will also increase the deeper you go, but more on that later.

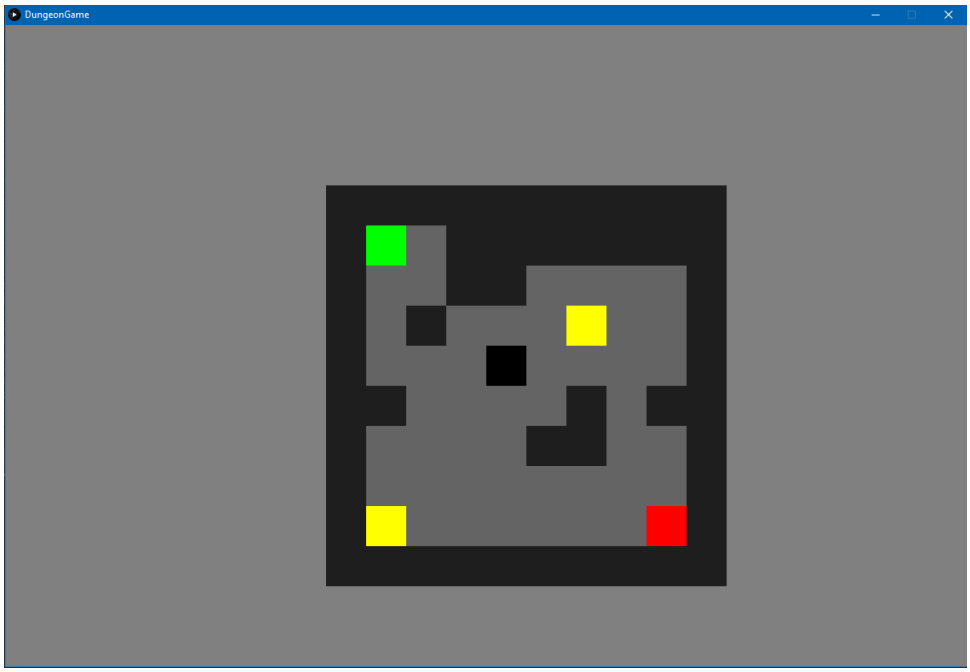

Figure 1: A picture of the first level. The green square represents the start, the red the finish, the yellow squares represent treasure and the black square is the player.

4 Character Movement

4.1 Player

The player is controlled by the conventional game movement keys "w" (up), "a" (left), "s" (down) and "d" (right). The player will spawn at the start tile of each dungeon floor. Using these controls, the player must navigate the black square to the red square.

Originally, I had it so it was possible for the player to see the entire floor map. Instead, I decided to "zoom in" the players perspective, to provide a sense of mystery and exploration. The player is able to see the entirety of the first level, but their borders quickly grow far outside the players window.

With this new view, I had to make it possible for the viewing window to always follow the player. To implement this, I essentially fixed the players position to the centre of the window, and whenever the player moves, it's the actual *map* that moves, rather than the player.

When a player attempts to move in a specified direction, the block in the re-

spective position is inspected. If it's found not to be a wall, the movement is valid, and the entire map is shifted in the opposite direction the player wished to move. This gives the illusion that it was the player itself that moved, rather than the world around it.

If a player moves over the top of a treasure tile, the treasure (if one is found) is added to the players inventory, and the yellow block is replaced with a plain floor tile. If the player moves over the finish tile, the level has been completed, and the next level is generated and displayed.

4.2 Enemies

Enemies are displayed as purple circles. Before sensing the presence of the player, the enemy can be in one of three states. It can either be stationary, patrolling up and down, or patrolling side to side.

When a player gets within an enemy's "sight radius", the enemy will begin to pursue the player. The sight radius is a function of the floor number. If the floor number is n, the radius of blocks an enemy can sense in is n blocks. This means the deeper you get in the dungeon, the harder it is to avoid the enemies.

When chasing the player, the enemy chooses the block it should move to using A* search. By treating the grid as a network with the blocks as nodes, the vertices being two neighbouring nodes always being the same, and using the euclidean distance between a node and the player as our heuristic, we can use A* search to allow the enemy to always move to the neighbouring block that is closest to the players current position. This pursuit will continue until either the enemy reaches the player, or until the player completes the level. Upon starting the pursuit, the enemy will gain a large speed boost, meaning being caught is almost inevitable.

5 Character Attributes

5.1 Player

The player has a pretty basic attribute set. This includes:

- HP - Health Points
- Level - The players experience level
- Str - Strength
- Dex - Dexterity
- Itl - Intelligence
- Armour - The players armour value

- Weapon - The players weapon damage

The game follows a very common model for Role-playing games. If the player completes certain actions, they gain experience. By collecting enough experience, they will increase their level, which in turn increases their base attributes. Upon levelling up, the amount of experience required to level up doubles. However, the player will also unlock the possibility to acquire new weapons and armour, which although rarer, are significantly stronger.

Other than the players level, all other attributes only have an effect during combat. Obviously, if a player's HP reaches 0, the player has died, and it's game over. "Str", "Dex" and "Itl" act like dice rolls, with the player's value for each representing the maximum roll the player can get whilst in combat. A players armour value represents how much incoming damage is blocked (if an enemy attempts to deal 10 damage, but a player has leather armour equipped, the player will only take 7 damage, as the armour has blocked 3), and a player's weapon damage represents the modifier to the players strength roll (if a player rolls 5 on strength, but has a dagger equipped, they will end up dealing 10 damage total).

Figure 2: The players starting attribute values, as can be seen in the inventory screen.

5.2 Enemies

Enemies have a much simpler attribute set.

- HP - Health Points
- Str - Strength
- Dex - Dexterity

- ExpWorth - Amount of experience player gains after defeating the enemy

The first three work essentially the same as the player. However, if/when an enemy is defeated, the player will gain an amount of experience equal to the value of the enemies "ExpWorth".

There are 4 different enemies, each with different attribute values. Here they are in order of weakest to strongest.

- Rat - Weakest
- Goblin
- Knight
- Dragon - Strongest

If a floor contains n enemies, it will contain $n-1$ of one enemy strength, and 1 enemy of the tier above. Of course, when navigating the dungeon, all enemies look the same, so it's impossible to evade the stronger enemies. You just have to pray you're lucky.

6 Treasure

There are three types of treasure a player is able to pick up - Weapons, Armour and Spells. These can be acquired after the player moves onto a yellow treasure tile. There's a 25% chance of it being either one of the three, with a 25% chance of the player getting nothing at all. If a treasure is found, the treasure is placed within the player's inventory and they receive 100 experience points.

6.1 Armour

There are 5 different armours available for the player to find during the game. Below is a list of them in order from weakest to strongest and the level they're available to be found at.

- Cotton Shirt - lvl 1
- Leather Armour - lvl 1
- Chainmail Armour - lvl 2
- Steel Armour - lvl 3
- Dragonscale Armour - lvl 4

As mentioned previously, armour is used to reduce damage taken from an enemy's attack. The higher the level of armour, the more protection is offered.

6.2 Weapons

There are also 5 different types of weapon available for the player to find at varying points in the game. These can be seen below ordered from weakest to strongest.

- Wooden Ladle - lvl 1
- Dagger - lvl 1
- Short Sword - lvl 2
- Long Sword - lvl 3
- Fire Blade - lvl 4

Weapons are used as attack modifiers when a player is dealing damage to an enemy during combat.

6.3 Spells - Extension

Spells come in the form of scrolls, and are very powerful. There are two offensive spells and one defensive spell. All three can be seen below.

- Fireball Scroll - +10 itl modifier (off)
- Blizzard Scroll - +15 itl modifier (off)
- Flash Heal Scroll - +10 itl modifier (def)

Magic can only be used during combat. However, to add to their power, they don't cost any "action points", meaning if you wanted, you could cast every spell in your inventory in the same turn.

All spells are modifiers to an intelligence roll. They can be very useful for getting out of tight spots.

7 Inventory

The players inventory can be accessed at any point by pressing the "i" key. A new window will appear displaying all items currently owned by the player, as well as a summary of the players attributes (as seen in figure 2).

The inventory contains free sub-menus; Armour, Weapons and Spells. These can be navigated using the "a" and "d" keys. Different items in the sub-menu currently being visited can be selected by using the "w" and "s" keys.

To equip a new weapon or some new armour, simply navigate the inventory so the desired item is selected, then press "e". A green box should surround the

chosen item, and the player summary should be updated to show the newly equipped armour or weapon.

A player can also discard an item by pressing the "r" key whilst having the item they wish to discard selected.

There is no maximum capacity for the inventory, as the maximum amount of items a player will collect will not exceed 10.

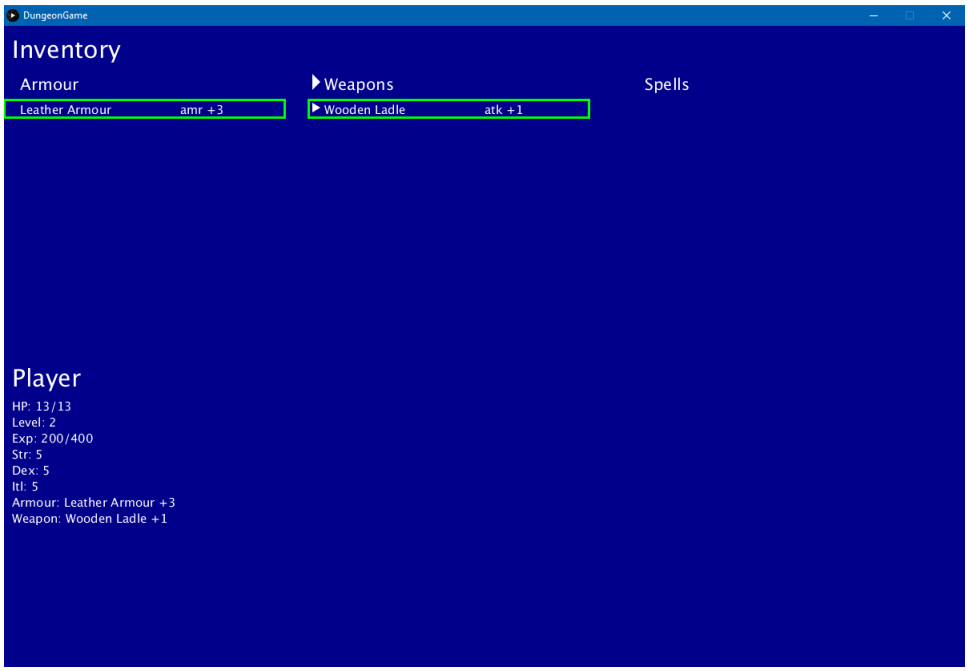

Figure 3: A players inventory with some leather armour and a wooden ladle equipped.

8 Combat

Once an enemy has caught the player, combat occurs. An enemy does not "catch" the player until the centre coordinate of the enemy is located inside the player's grid position.

A player has three options during combat.

- Attack
- Dodge

- Cast

8.1 Attack

When a player chooses to attack, a random number is generated between 1 and the players strength attribute inclusively. The attack modifier of the players weapon (if they have one equipped) is then added to this random number.

The enemy will then do a dexterity roll (same as the strength random number generation). If this dexterity value is greater than the incoming damage, the enemy has successfully dodged the players attack.

However, if the enemy fails the roll, damage is dealt. This means the enemy loses an amount of HP equivalent to the damage dealt.

8.2 Dodge

A random dexterity multiplier is generated between 1.1 and 2.0 (only to 1 d.p.). When the enemy attacks, the players dexterity roll will be multiplied by this modifier, greatly increasing the players chance of dodging the incoming attack.

8.3 Cast

This is where the player is able to use the magical scrolls they have collected on their adventures. The players inventory is opened, and the player is now able to select what spell they wish to cast by pressing "e" with the desired spell selected. If a player changes their mind, they can exit the spell selection screen by pressing "r".

Upon selection, an intelligence roll occurs. Whatever value contained within the scroll is then added to this dice roll, and the spell is subsequently cast. If it's one of the offensive spells, this sum is the amount of damage dealt to the enemy. If it's the healing spell, that's how much you will be healed for.

A player is able to cast a spell without ending their turn. This is extremely powerful, as it allows them to quickly save themselves if they are on the edge of defeat. To counter this, scrolls can only be used once. Once a spell has been cast, it is removed from the players inventory, to stop a player from casting infinite fireballs.

8.4 Enemy

If the enemy is still alive after your turn, they get to retaliate. This is essentially the same event flow as when a player attacks, except with the roles of attacker and victim swapped.

If an enemies attack lands, damage is dealt. The amount varies on the level of armour the player has equipped. As explained earlier, each point of armour negates a point of damage being dealt.

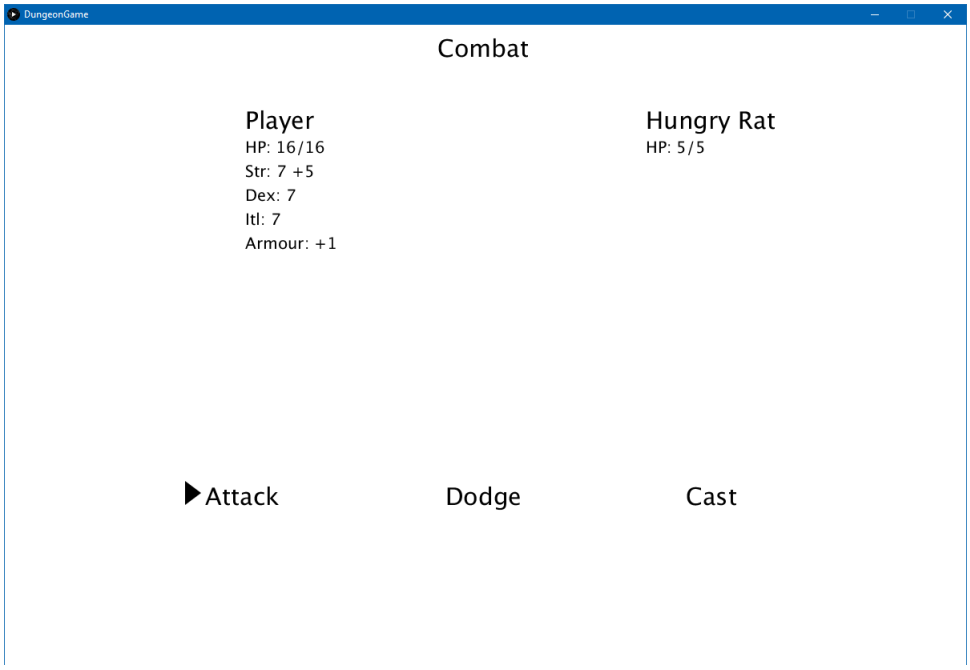

Figure 4: The combat screen. Here we can see the player battling a "Hungry Rat".

9 Evaluation and Conclusion

In conclusion, I feel I have met the full specification and even extended it. I have made a game that procedurally generates levels containing treasure and moving enemies that are able to make intelligent decisions on map navigation. The player is able to navigate around the level using player input. For an extension, the view of the player is zoomed in to hide the full level and provide a greater feeling of exploration.

The player is able to collect treasures of multiple types, all with different uses. Armour and Weapons are able to modify combat rolls. Magic spells - another extension - provide a quick escape for players in tough situations, but are limited to one use per scroll. The player is able to view, equip and discard these items via the inventory, as well as view their current attributes.

Upon contact with an enemy, combat begins. Combat does not end until one combatant is dead. The player is able to attack, dodge or cast a spell. There are multiple types of enemies to fight, with the harder types being introduced the deeper you get into the dungeon.

Collecting treasures, killing enemies and exiting levels grants the player experience points, which allow them to level up and increase their base attributes, as well as unlock the possibility of finding more powerful items.

When one dungeon level is completed, a new floor is built. This new floor is larger, containing more enemies that are harder and can see for a further distance, making them harder to evade. All these factors mean each floor is more difficult than its predecessor.

When five floors are completed, the player has won. The total amount of experience the player has gained over their adventure is shown, and they are asked if they want to play again.

If a player is killed by an enemy, a game over screen is displayed, and the player is asked if they want to play again.

Overall, I am very happy with the end product. It is a solid game with simple but effective mechanics that allow for quick and enjoyable gameplay.

10 How To Run and Controls

To play the game, simply run the .jar file using the command:

java -jar p2.jar

10.1 Controls

- “w” - Move up/Move selection up
- “a” - Move left/Move selection left
- “s” - Move down/Move selection down
- “d” - Move right/Move selection right
- “e” - Make selection/Equip item
- “r” - Discard item/Go back/Start new game
- “i” - Toggle inventory