Introduction

For this practical we were asked to make a variant of the classic "Roguelike" roleplaying game. I found my variant to be complete in terms of specification as well as run performantly. My game features textures to make it visually appealing and appropriate values to allow for a fun playing experience, which I narrowed down by testing and playing the game. I hope you enjoy it.

.

Player's Guide

The following compilation instructions, and the controls / rules that are explained to the player upon loading into and progressing through the game should provide a complete guide to playing the game.

For the sake of completeness I will include that information here as well.

Controls

W: move up
A: move left
S: move right
D: move down

E: open inventory

Mouse: left click used to level up

Objective

Defeat all monsters before advancing to the next level by walking onto the bright green tile in the corner of one of the rooms. There are five levels total. If the player defeats all five levels, they win.

The player's "score" and measure of how well they have done is the experience level they reached and the amount of gold they found before either dying or completing the game. 10 gold is gained when defeating a goblin and 100 is gained when the goblin has a chance of dropping a rare valuable treasure.

Stats Explanation

Health:	the players hit points remaining
Sword:	the amount of damage the player deals
Armour:	the amount of damage the player absorbs from attacks
Luck:	determines the likelihood of the player finding gold when killing a goblin, and the likelihood of the goblin hitting itself when blocking an attack.
Speed:	the maximum speed at which the player runs.

Pickups

Sword:	repairs sword
Armour:	repairs armour
Potion:	stored in inventory, can be used during combat to replenish health

Compilation

The “/src” folder containing all .pde source files as well as the textures folder for images is needed to be extracted to run the game.

Once extracted, to run my game open the “Roguelike.pde” and “Sprite.pde” source files in the Processing 4 editor and click run in the top left.

Design

World generation

My world is structured using a 2D integer array of tiles. The indices of the array store the X and Y location of each tile. The value of each tile ranges from 0 to 5.

0: black empty space tile
1: white room tile
2: green end tile
3: sword pickup tile
4: armour pickup tile
5: potion pickup tile

I found structuring and storing the map for my game this way to be intuitive and makes it easy to draw the whole level by using a nested for loop to iterate through the array.

In *makeRooms()* I initialise every value in this tile array to 0.

Rooms

The information for the location and dimensions of rooms is stored inside a 2D integer array of rectangles. The first index stores the room number, and the second index stores the location and size of that room and importantly, the index of the room it "points to" or connects to with a corridor.

In *makeRooms()* use random integers to generate starting x and y coordinates of a rooms as well as their widths and heights.

Next, I check that none of these rooms overlap with my *checkOverlap()* method.

Designing my rooms in this way doesn't clutter my tiles array and allows for varied and interesting level designs every time.

Corridors

I designed my own method for constructing corridors. In *makeCorridors()* I find the middle tile of a room and the middle tile of the room that is closest to it that isn't already pointing to it.

Then I iterate from the X location of the first middle tile to the X location of the second middle tile, setting each tile to 1 to mark it as a room tile. Next, I do the same with the Y locations.

This draws a line or "corridor" from one room to the next.

Layout Validation

Now I have constructed a level layout I check that each room is accessible in my *checkLayout()* method. This method iterates over the rooms in my array and stores the index of the room it points to in a set. It does this a few times and then the size of the set is checked to see if it matches the number of rooms. If it does then the layout is valid and all rooms are accessible, if it doesn't, a while loop is used to generate a new layout as many times as needed until it is valid.

Player

The players movement and steering works based from the KinematicArriveSketch function from studres.

Upon pressing the WASD keys, Booleans are set to true which are then checked in my draw function *drawPlayer()* to avoid polling the keyboard every frame. Next, depending on which booleans are true, a target vector will be created with a new X and Y location based off the player's location.

Then the player's integrate function is called with this new target to update their speed, location and orientation.

Collisions

I check collisions using the get(x,y) method which returns the value of the colour of the specified pixel. I make a copy of the players previous target and if their new chosen target's pixel is black then I return to the old target. This allows the player and monsters to stay within the white room tiles.

Monsters

Further dungeon levels increase their damage dealt, their health and both their idle and aggro speeds.

Idle mode

By default the monsters or "goblins" are green and move in random directions, if the player gets too close to them they change to aggro mode.

Aggro mode

If the monsters are aggravated they change to red and move faster and follow the player again using the same method from the KinematicArriveSketch function from studres. Their target is set to the player's X and Y location every pass.

Sometimes the monsters can appear to visibly shake when pathfinding which, when combined with their altered colour and speed, does give them the appearance of being "angry".

If they reach the player the game enters combat mode, if the player gets far enough away they return to idle mode.

Combat

The game is paused when fighting and a new screen is drawn over the map. It shows the player and goblin sprites as well as their health bars.

The player is given three options to choose from, either attack, defend or use a potion to replenish their health.

When the player attacks or defends their sprite changes icon, which when combined with the transparent drawing of the grey background, gives an animation like effect to the combat.

If the player chooses to attack, their damage is randomised centred on their power attribute, and is subtracted from the monster's health. Then the monster attacks back. The amount of damage the player takes from this attack is lessened by their armour or defence attribute to be exact.

If they choose to defend and block with their shield, the power of the monster's attack is reduced, and a random integer is used to determine whether the goblin hits itself for some damage as well. The likelihood of this happening increases with the player's luck attribute.

When a goblin is killed, they drop 10 gold and 34 experience (as I wanted it to take 3 monsters to level up beyond the first two level ups). Additionally, they have a random chance of dropping valuable treasure worth 100 gold. The likelihood of this happening increases with the player's luck attribute.

Stats and Levelling

Player has a health, power, defence, speed and luck attribute.

The armour pickup replenishes defence, (lowers damage taken).

The sword pickup replenishes power (increases damage delt).

When the player receives enough experience, they can level up to increase both the current amount and max amount of one of these attributes.

I implemented this using the mouse, where the player clicks on the meter of the attribute they want to advance, I found this to be very quick and intuitive.

Speed determines the speed at which the player runs around the map, and luck determines the likelihood of lucky events occurring as mentioned earlier.

Items and Inventory

Potions can be stored to replenish health; they are visible in the player's inventory by holding E at any time. The inventory can store 16 potions. The potions can be used during combat to replenish health.

I use a *setSpecialTiles()* method to store item pickups in the centre of my rooms, by changing the value of tiles in my tiles 2D array, allowing for 3 swords, 3 armour and 4 potion pickups per dungeon level.

Specification Design Question

The specification states:

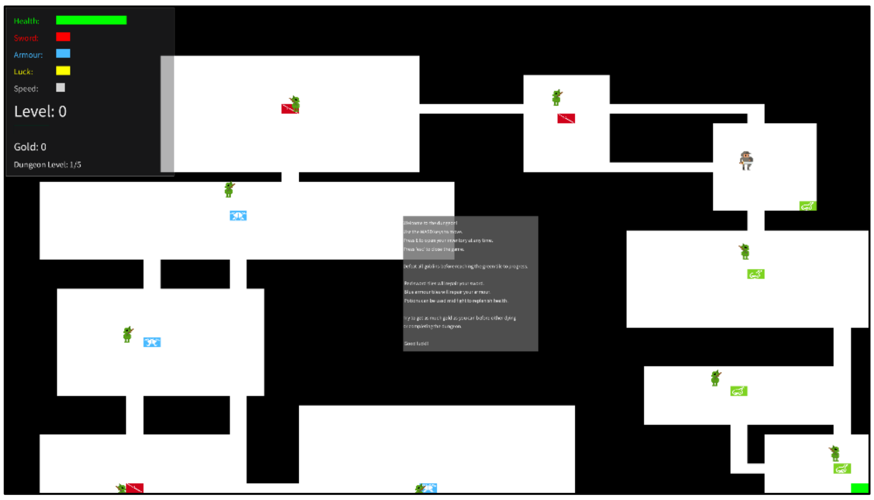
“For at least one element of your game, compare how different design choices affect the game for the players. Justify the choice you have made for the submitted version of your game.”

A design choice that affects the game for the players would be the levelling up system. Originally, I had thought about having a text or button-based system but I thought this would be clunky and cluttered due to the number of attributes. So instead I opted for my system where the player clicks the attribute they want to advance, I found this to be very straightforward and quick to understand and perform, especially for new players.

Another major design choice is how to display combat and other game elements. I chose to display combat and stats and popup messages in transparent boxes on top of the existing game. I found doing this to allow for a more cohesive experience and not confuse or distract the player which is what I think switching to other screens would do.

Screenshots

Intro



Health:
Sword:
Armour:
Luck:
Speed:
Level: 0
Gold: 0
Dungeon Level: 1/5

Welcome to the dungeon!
Use the WASD keys to move.
Press E to open your inventory at any time.
Press 'esc' to close the game.

Defeat all goblins before reaching the green tile to progress.

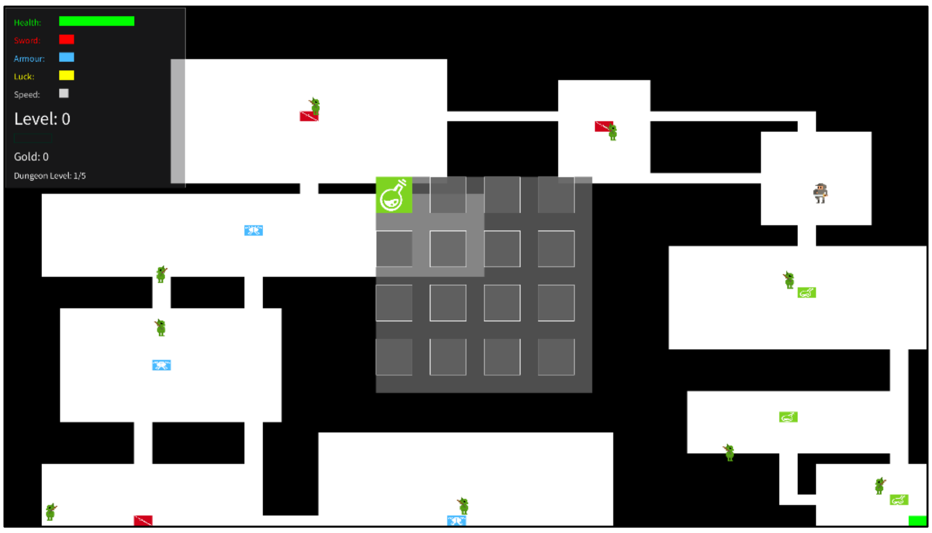
Red sword tiles will repair your sword.
Blue armour tiles will repair your armour.
Potions can be used mid fight to replenish health.

Try to get as much gold as you can before either dying or completing the dungeon.

Good luck!

Figure 1: Screenshot showing the level design, stats, as well as the tutorial popup when starting a game.

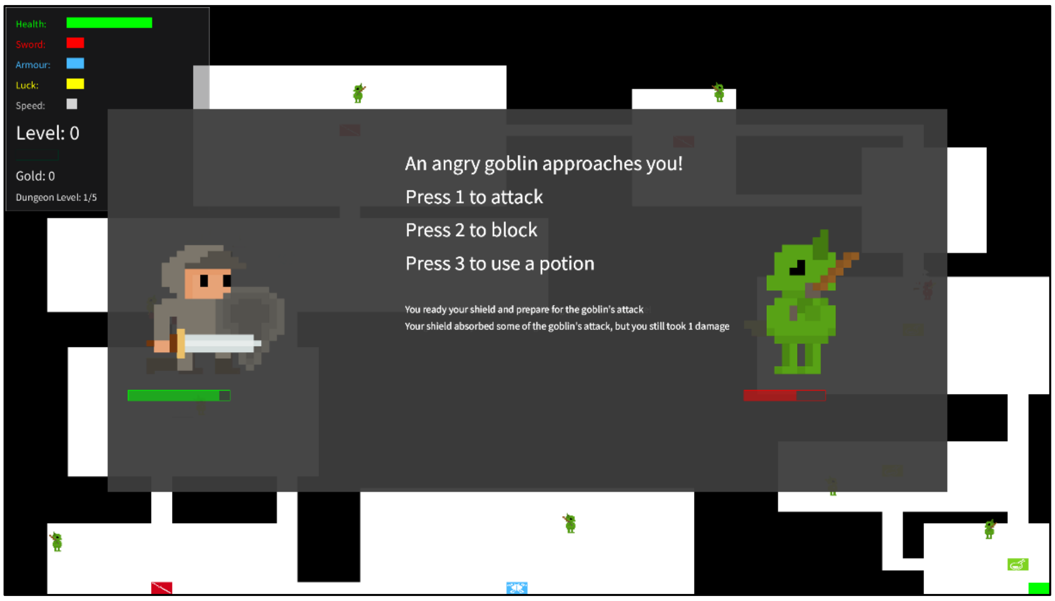
Inventory



Health:
Sword:
Armour:
Luck:
Speed:
Level: 0
Gold: 0
Dungeon Level: 1/5

Figure 2: Screenshot showing the open inventory with a potion inside

Combat

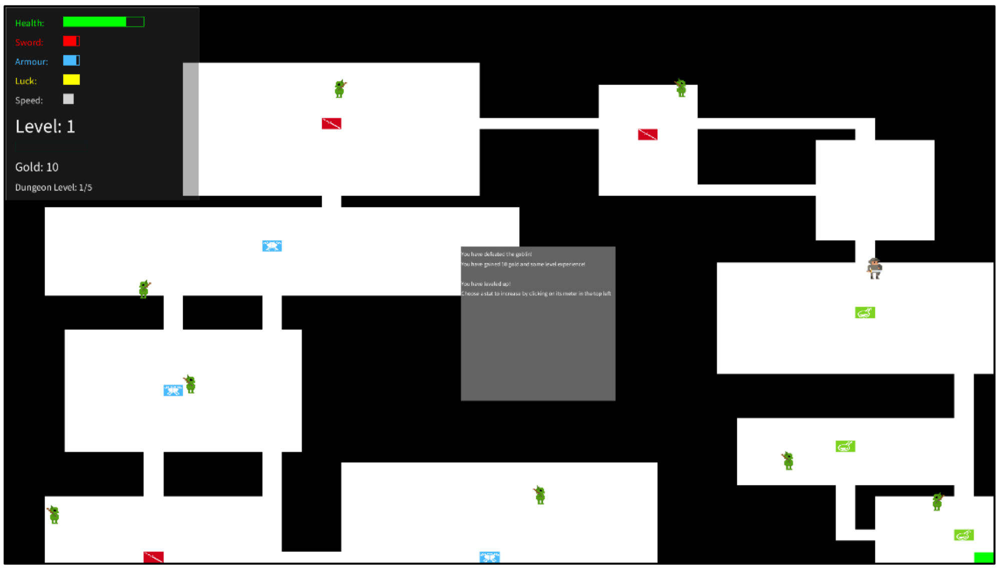


An angry goblin approaches you!
Press 1 to attack
Press 2 to block
Press 3 to use a potion

You ready your shield and prepare for the goblin's attack
Your shield absorbed some of the goblin's attack, but you still took 1 damage

Figure 3: Screenshot showing the combat pop

Level up

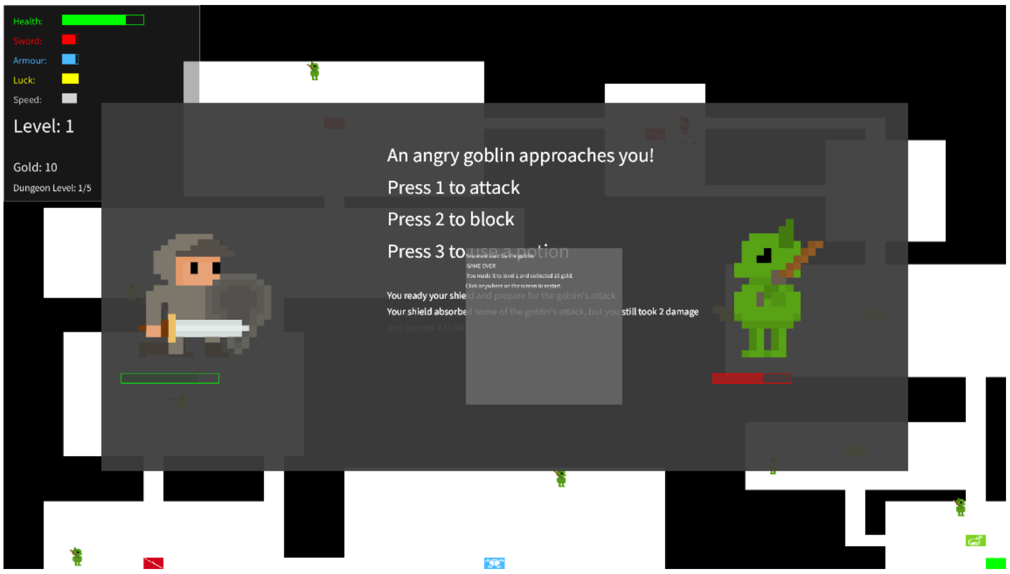


You have defeated the goblin!
You have gained 10 gold and some level experience!

You have leveled up!
Choose a stat to increase by clicking on its meter in the top left

Figure 4: Screenshot showing the level up popup

Death screen



An angry goblin approaches you!
Press 1 to attack
Press 2 to block
Press 3 to use a potion

You were slain by the goblin
GAME OVER
You made it to level 1 and collected 10 gold.
Click anywhere on the screen to restart.

You ready your shield and prepare for the goblin's attack
Your shield absorbed some of the goblin's attack, but you still took 2 damage
and caused it to hit

Figure 5: Screenshot showing the game over popup when the player's health reaches 0 and they have died.

Conclusion

I found my variant of the "Roguelike" game to be complete and meet the basic requirements from the specification. I also added textures and spent time polishing the UI of the game for a satisfying and fun to play experience.

Had I had more time I could have gone on to adding different weapons or armours etc. and having these stored in the inventory, so It isn't just for potions. I also would've added an inventory management system in this case.

Additionally, I would have liked to add different types of enemies, but the increasing level difficulty making the goblins harder is a good substitute.