Part One: Product Specification - MVP

NP AL 创建者：Nick Patrikeos
由 Amanda Lu 最后更新于 3月 03, 2023 • 125 人已查看

1. Preamble and Problem

For many years, Penguin Pty Ltd (a small software business run by Atilla Brungs) has dominated the native application gaming market with hit games. However in recent years, advancements in web technologies mean that the new generation of consumers don't want to download and run a native application, but instead want to play games online. To adapt to the ever-changing market, Penguin Pty Ltd decided in 2021 to take users back to the 1980s and develop a new game called *Dungeonmania*, but with a modern twist - designed as a web application.

They hired two teams - one to complete the frontend and one to complete the backend. Together, the teams built an MVP which brought Penguin back to #1 on the charts - but now the users are wanting more! The hard working backend engineers, previous terms' COMP2511 students have all left. A lull in sales has left only budget for two people rather than the previous five to work on the backend. What's more, the previous engineers left a series of design issues in their implementation.

You and your partner have been hired and have inherited the existing codebase for the Dungeonmania game.

In this assignment, you will be working with your partner on the existing codebase in a series of tasks:

- Acclimatising yourselves to the codebase, analysing the existing code, including Design Patterns present and smells;
- Refactoring the code to improve the quality of the design, incorporating Design Patterns discussed in the course;
- Extending on the existing codebase to provide new functionality.

2. Product Specification (MVP)

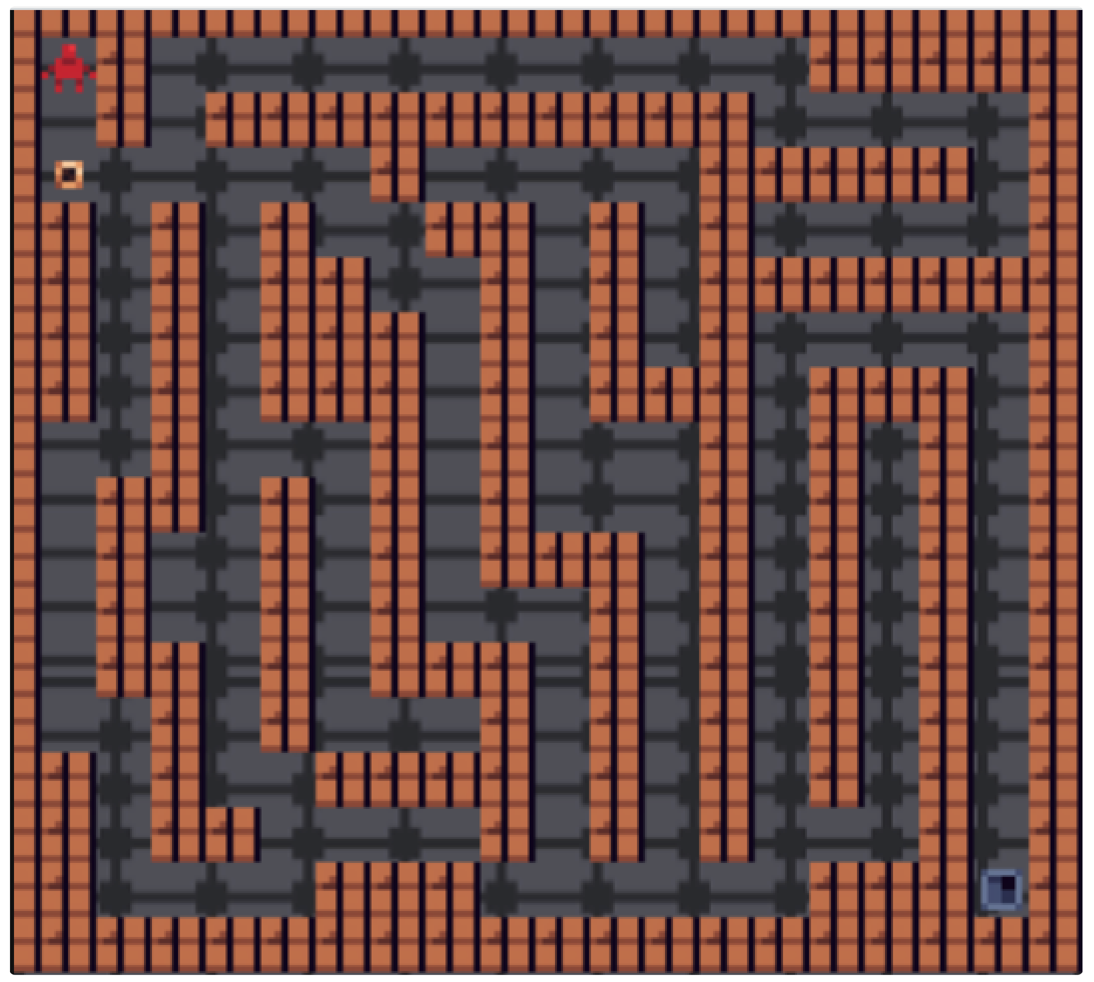
You have been given the product specification from the MVP version of Dungeonmania to help you understand the existing code and functionality it provides.

All of this functionality on this Confluence page has been implemented in the monolith repository we have provided to you.

You do not need to implement it yourselves.

In Dungeon Mania you control a Player and have to complete various goals within a series of dungeons to complete the game!

The simplest form of such a puzzle is a maze, where the Player must find their way from the starting point to the exit.



More advanced puzzles may contain things like boulders that need to be pushed onto floor switches,

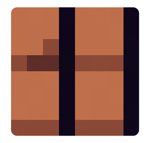
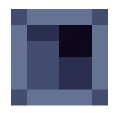
enemies that need to be fought with weapons, or collectables like potions and treasure.

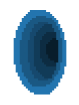
2.1 Player

The Player, can be moved up, down, left and right into cardinally adjacent squares, provided another entity doesn't stop them (e.g. a wall). The Player begins the game with a set amount of health and attack damage. The Player spawns at a set 'entry location' at the beginning of a game.

2.2 Static Entities

The game contains the following static entities.

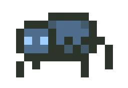
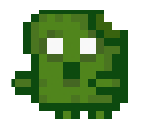
Entity	Image	Description
Wall		Blocks the movement of the Player, enemies and boulders.
Exit		If the Player goes through it, the puzzle may be complete.
Boulder		Acts like a wall in most cases. The only difference is that it can be pushed by the Player into cardinally adjacent squares. The Player is only strong enough to push **one** boulder at a time. When the player pushes a boulder, they move into the spot the boulder was previously in. Boulders can be pushed onto collectable entities.
Floor Switch		Switches behave like empty squares, so other entities can appear on top of them. When a boulder is pushed onto a floor switch, it is triggered. Pushing a boulder off the floor switch untriggers.
Door		Exists in conjunction with a single key that can open it. If the Player holds the key, they can open the door by moving through it. Once open, it remains open.

Portal		Teleports entities to a corresponding portal. The player must end up in a square cardinally adjacent to the corresponding portal. The square they teleport onto must also be within movement constraints - e.g. the player cannot teleport and end up on a wall. If all squares cardinally adjacent to the corresponding portal are walls, then the player should remain where they are.
Zombie Toast Spawner		Spawns zombie toasts in an open square cardinally adjacent to the spawner. The Player can destroy a zombie spawner if they have a weapon and are cardinally adjacent to the spawner. If all the cardinally adjacent cells to the spawner are walls, then the spawner will not spawn any zombies.

2.3 Moving Entities

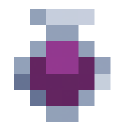
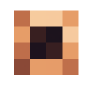
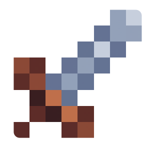
In addition to the Player, the game contains the following moving entities.

All enemy entities can be created as part of the initial dungeon. Each tick, all enemies move according to their respective behaviour.

Entity	Image	Description
Spider		Spiders spawn at random locations in the dungeon from the beginning of the game. When the spider spawns, they immediately move the 1 square upwards (towards the top of the screen) and then begin 'circling' their spawn spot (see a visual example below). › Spider Movement Figure 1 Spiders are able to traverse through walls, doors, switches, portals, exits (which have no effect), but not boulders, in which case it will reverse direction (see a visual example below). › Spider Movement Figure 2 Spiders spawn in a square that is less than or equal to a radius of 20 (via Manhattan distance) around the player's current position. If there is no available space, a spider is not spawned. Spiders cannot spawn on boulders, or in the same square as the player/enemies. If a spider is stuck between two boulders in its movement path, it should remain still.
Zombie Toast		Zombies spawn at zombie spawners and move in random directions. Zombies are limited by the same movement constraints as the Player, except portals have no effect on them.
Mercenary		Mercenaries do not spawn; they are only present if created as part of the dungeon. They constantly move towards the Player, stopping only if they cannot move any closer (they are able to move around walls). Mercenaries are limited by the same movement constraints as the Player. All mercenaries are considered hostile, unless the Player can bribe them with a certain amount of gold; in which case they become allies. Mercenaries must be within a certain radius of the player in order to be bribed, which is formed by the diagonally and cardinally adjacent cells in a "square" fashion, akin to the blast radius for bombs. As an ally, once it reaches the Player it simply follows the Player around, occupying the square the player was previously in.

2.4 Collectable Entities

Entity	Image	Description

Treasure		Can be picked up by the Player.
Key		Can be picked up by the player when they move into the square containing it. The Player can carry only one key at a time, and only one door has a lock that fits the key. Keys disappear once used in any context i.e. opening a door, building an item. If a key is used before opening its door, its corresponding door may be locked forever.
Invincibility Potion		When a Player picks up an Invincibility potion, they may consume it at any time. Any battles that occur when the Player has the effects of the potion end immediately after the first round, with the Player immediately winning and taking no damage. Zombies and mercenaries will run away from the player when the player is invincible. Movement of spiders remains unaffected. The effects of the potion only last for a limited time.
Invisibility Potion		When a player picks up an invisibility potion, they may consume it at any time and they immediately become invisible and can move past all other entities undetected. This means that mercenaries will no longer follow the player and will now move randomly when the player is invisible. Battles do not occur when a player is under the influence of an invisibility potion.
Wood		Can be picked up by the player.
Arrows		Can be picked up by the player.
Bomb		Can be collected by the player. When used it is removed from the inventory it is placed on the map at the player's location. When a bomb is cardinally adjacent to an active switch, it destroys all entities in diagonally and cardinally adjacent cells, except for the player, forming a "square" blast radius. The bomb should detonate when it is placed next to an already active switch, or placed next to an inactive switch that then becomes active. The bomb explodes on the same tick it becomes cardinally adjacent to an active switch. A bomb cannot be picked up once it has been used.
Sword		A standard melee weapon. Swords can be collected by the Player and used in battles, increasing the amount of damage they deal by an additive factor. Each sword has a specific durability that dictates the number of battles it can be used before it deteriorates and is no longer usable.

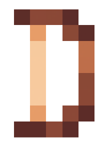
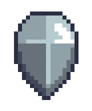
It is possible for a player to use another potion while the effects of an existing potion are still lasting (can be of the same or a different type of potion). In this case, the effects are not registered immediately but are instead 'queued' and will take place the tick following the previous potion wearing of. For example:

- On tick 0 the Player consumes an invinsibility potion that lasts for 5 ticks and becomes invisible to enemies moving that tick
- On tick 3 they use an invincibility potion
- At the end of tick 4 (after all enemy movements) the player becomes visible again and becomes invincible.

2.5 Buildable Entities

Some entities can be built using a 'recipe' by the player, where entities are combined to form more complex and useful entities. Once a buildable item has been constructed, it is stored in a player's inventory. For all buildable entities, once the item is constructed the materials used in that construction have been consumed and disappear from the player's inventory.

constructed the materials used in that construction have been consumed and disappear from the player's inventory.

Entity	Image	Description
Bow		Can be crafted with 1 wood + 3 arrows. The bow has a durability which deteriorates after a certain number of battles. Bows give the Player double damage in each round, to simulate being able to attack an enemy at range (it can't actually attack an enemy at range).
Shield		Can be crafted with 2 wood + (1 treasure OR 1 key). Shields decrease the effect of enemy attacks. Each shield has a specific durability that dictates the number of battles it can be used before it deteriorates.

2.6 Battles

A battle takes place when the Player and an enemy are in the same cell at any point within a single tick. The conditions for a battle occurring are the same regardless of whether the player moves onto the same tile as the enemy, or vice versa.

A 'round' of a battle occurs as follows:

```
Player Health = Player Health - (Enemy Attack Damage / 10)
Enemy Health = Enemy Health - (Player Attack Damage / 5)
```

Damage will be applied simultaneously to the player and enemy in each round.

If the Player's health is <= 0, then the Player dies, is removed from the game and the game is over. If the enemy's health is <= 0, then the enemy dies and is removed from the game. If after the above 'round', neither the Player nor the enemy is dead, the round repeats until either the Player or enemy is dead.

In battles, allies provide an attack and defence bonus to the player.

2.6.1 Weapons in Battle

An example of a bow, sword and shield being used in battle is as follows:

```
player health = 10
player base attack damage = 5
bow attack damage = 2
sword attack damage = 1
shield defence = 2
enemy health = 10
enemy attack damage = 5

Battle occurs:
- Round 1   enemy health    = 10 - ((2 * (5 + 1)) / 5)  = 7.6
            player health   = 10 - ((5 - 2) / 10)       = 9.7
- Round 2   enemy health    = 7.6 - ((2 * (5 + 1)) / 5) = 5.2
            player health   = 9.7 - ((5 - 2) / 10)      = 9.4
- Round 3   ...
```

All additive/reductive bonuses from weapons are processed before multiplicative bonuses.

2.7 Goals

In addition to its layout, each dungeon also has a goal that defines what must be achieved by the player for the dungeon to be considered complete. Basic goals are:

- Getting to an exit;
- Having a boulder on all floor switches;
- Collecting a certain number of treasure items (or more);

Goals are only evaluated after the first tick. If getting to an exit is one of a conjunction of conditions, it must be done last. For example, if the condition is to have a boulder on all floor switches AND get to an exit, the player must put the boulder on the switches THEN get to the exit.

3.7.2 Complex Goals

More complex goals can be built by logically composing goals. For example:

- Collecting a certain number of treasure AND getting to an exit
- Collecting a certain number of treasure OR having a boulder on all floor switches
- Getting to an exit AND (destroying all enemies OR collecting all treasure)

All compound goals are binary (they contain two and only two subgoals).

If getting to an exit is one of a conjunction of conditions, it must be done last. For example, if the condition is to collect 3 treasure AND get to an exit, the player must collect at least 3 treasures THEN get to the exit. It is possible for a subgoal to become un-achieved, for example if the dungeon goal is `boulders AND exit` and all boulders are pushed onto switches, then the boulders subgoal becomes complete. However, if a boulder is then moved off a switch, the boulders subgoal is no longer complete.

2.8 Winning & Losing

The game is won when all the goals are achieved. The game is lost when the player dies and is removed from the map.

2.9 Advanced Movement

The movement of mercenaries follows a Djikstra's algorithm to take the shortest path towards the player.

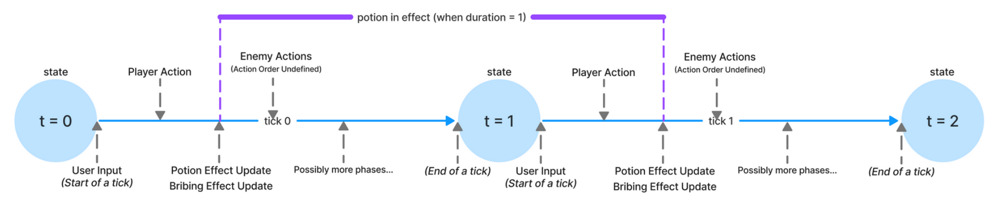
› You can view pseudocode for the algorithm here.

All references to **radii** distances are **Manhattan Distances** unless otherwise specified.

Manhattan distance [Explained]

2.10 Tick Definition

A tick is a transition from one state to a new state. A tick always starts with user input (i.e. in one tick, the player always does action first, then enemies/spawners). Then the game world changes in the tick and ends when another user input is needed. So "tick n" is the transition from the n-th state to the (n+1)-th state. There can be multiple developer-defined phases within one tick deciding the order of changes to the game world. Here is one possible example on phases sequence to help you understand.



You can view a UML diagram of the implementation of this MVP specification that's provided in the monolith on this Confluence page: MVP UML Diagram

成为第一个添加回复的用户