Project-237-TDD

Contents

Controls

The game is to be played using a keyboard and mouse. Below are the set controls for the game:

Key	Type	Action
W	Held	Move Up.
S	Held	Move Down
A	Held	Move Left
D	Held	Move Right
E	Pressed	Pick up / Use Item
P	Toggle	Pause/Un-pause Game
1-4	Pressed	Select Inventory Option
Esc	Pressed	Exit Menu
LMB	Click	Select Menu Option

Player

Movement

As you move around the snowy maze you place footprints as you go. This is used so that the player and AI know where the player has been already. If the AI finds the footsteps, they fade away. The sprite (shown right) is displayed on every tile the player exits, and the tile is rotated to show the correct direction of player travel.

Collision

As the player moves around, the program will look at the coordinates of both the player and the tiles the player is about to move into. The tile data is collected via passing the player's input direction through to the player class to get the coordinates of the tiles. The coordinates are passed back into the level class and checks the tiles for whether they are walkable or not (path or hedge). If the tile is not walkable, it will then check both the bounding boxes of the player and the tile to make sure they are not colliding. If not, the player is able to move closer to the tile, but as soon as the player and the tile are touching the movement stops. This is done by multiplying movement speed in the player's direction by zero. The player is still able to move in any other direction to move away from the tile.

Map Design

General

The map is created using some random generation. This works by finding the position inside the maze needed and then choosing one of the select "Map Blocks" in the map blocks file. On each play, the level is built by selecting each piece by random until at the desired size. Thus, every time you play the game, the map will look different. The desired size is determined by the level difficulty chosen.

Difficulty	Map size (Number of Map Blocks)
Easy	10x10
Medium	10x10
Hard	15x15

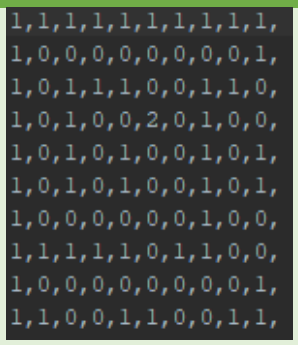
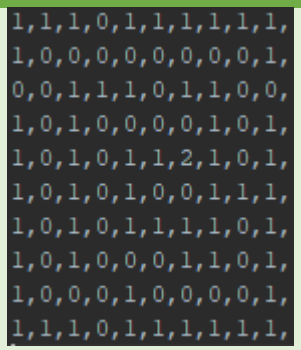
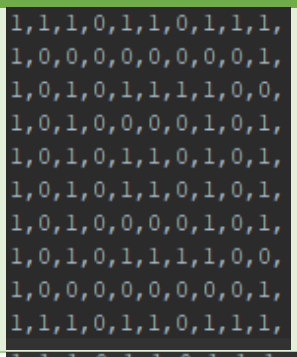
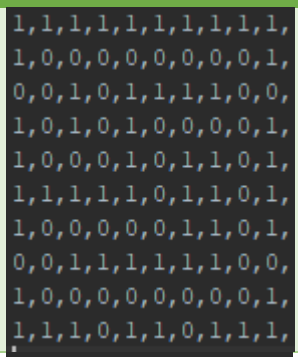
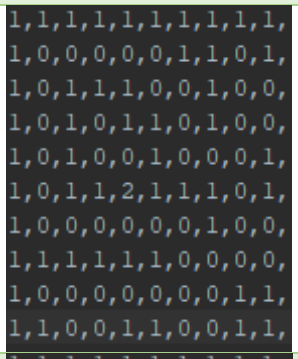
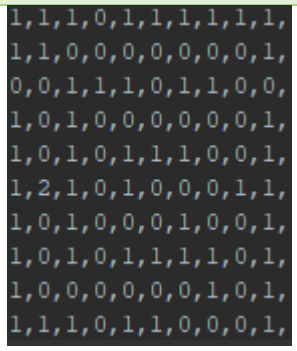
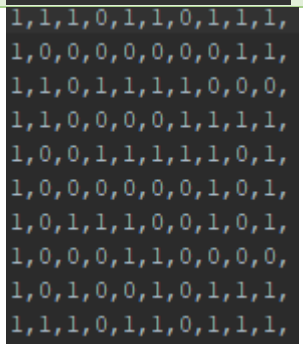
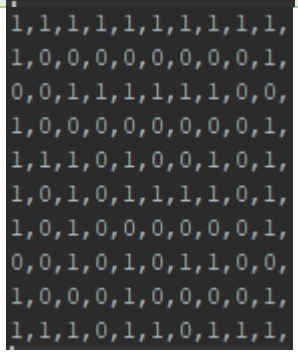
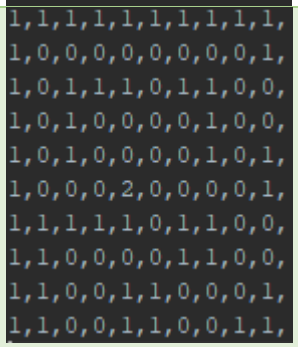
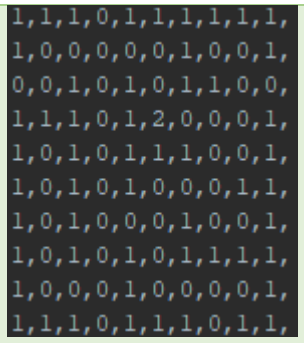
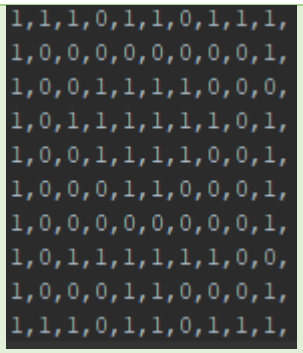
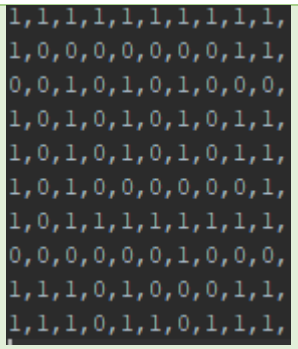
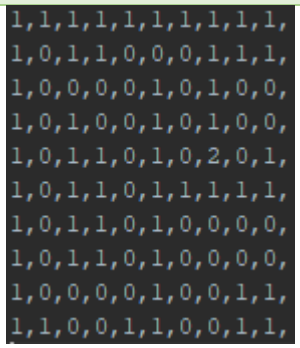
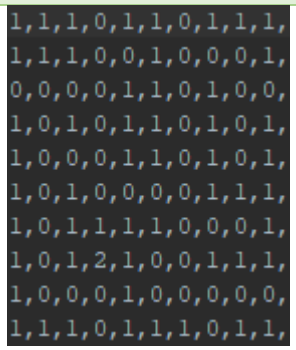
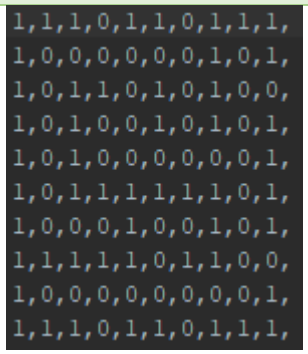
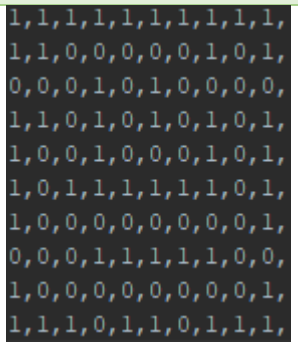
Tile Types

The map is designed through the use of “Map Blocks” which in turn are created using “Map tiles”. Each block is created by using a 10x10 grid of tiles. The positioning of the tiles is specified using a comma separated values (CSV) file. 3 types of tile are specified in the game:

Tile Type	Number assigned to tile	Texture
Snow	0	
Hedge	1	
Any Item	2	(Multiple Textures, shown in Items Section)

Block Types

Different types of Block are needed depending on the position they are in the maze. Each Block is given a sub-block type. These are “Corner”, “Middle”, “X Edge” and “Y Edge” blocks. This mostly effects the outer walls on each block type. This was done so that when placed together the outer walls is always defined and the player can’t escape the maze. Corner, Edge X and Edge Y block are also rotated to be in the correct orientation depending on what position they are in the maze.

Number	Corner	Middle	X Edge	Y Edge
1				
2				
3				
4				

5	1,1,1,1,1,1,1,1,1,1,1, 1,0,0,0,0,0,0,0,0,0,1, 1,1,1,1,1,1,1,1,1,0,0, 1,0,1,0,0,0,0,0,0,0,0, 1,0,0,0,1,1,1,1,1,1,1, 1,0,1,0,0,0,0,0,0,0,1, 1,0,1,0,0,1,1,1,0,0,0, 1,0,1,2,1,1,1,0,0,0,0, 1,0,0,0,1,1,0,0,1,1,1, 1,1,0,0,1,1,0,0,1,1,1,	1,1,1,0,1,1,1,1,1,1,1, 1,1,1,0,0,0,0,0,1,1,1, 0,0,0,0,1,1,0,1,0,0,0, 1,1,1,0,0,1,0,1,0,1,1, 1,0,1,1,0,1,0,0,0,1,1, 1,0,1,0,0,0,0,1,1,1,1, 1,0,1,0,1,0,1,1,2,1,1, 1,0,1,0,1,0,1,1,0,1,1, 1,0,0,0,0,0,0,0,0,1,1, 1,1,1,0,1,1,1,1,1,1,1,	1,1,1,0,1,1,0,1,1,1,1, 1,0,1,0,1,1,0,0,0,1,1, 1,0,1,0,1,1,0,1,0,0,0, 1,0,1,0,1,0,0,1,0,1,1, 1,0,1,0,0,0,0,1,0,1,1, 1,0,0,0,1,1,1,1,0,1,1, 1,0,1,0,1,0,0,1,0,1,1, 1,0,1,0,0,0,0,0,0,0,0, 1,1,1,0,1,1,0,1,1,1,1, 1,1,1,0,1,1,0,1,1,1,1,	1,1,1,1,1,1,1,1,1,1,1, 1,0,0,0,1,0,0,0,0,1,1, 0,0,1,0,1,1,0,1,0,0,1, 1,0,1,0,0,1,0,1,0,1,1, 1,0,1,0,0,1,1,1,0,1,1, 1,0,1,0,0,0,0,0,0,1,1, 1,0,1,1,1,0,1,1,0,1,1, 0,0,0,0,1,0,1,0,0,0,1, 1,0,0,0,1,0,0,0,1,1,1, 1,1,1,0,1,1,0,1,1,1,1,
6	1,1,1,1,1,1,1,1,1,1,1, 1,0,0,0,0,0,0,1,0,1,1, 1,0,1,1,1,0,0,1,0,0,1, 1,1,1,0,1,1,0,0,0,0,1, 1,2,0,0,0,1,1,1,0,1,1, 1,1,1,0,1,1,0,1,0,1,1, 1,0,1,0,0,0,0,1,0,0,1, 1,0,1,0,1,1,0,1,0,0,1, 1,0,0,0,0,0,0,0,0,0,1, 1,1,0,0,1,1,0,0,1,1,1,	1,1,1,0,1,1,0,1,1,1,1, 1,0,0,0,0,0,0,0,0,0,1, 0,0,1,1,1,1,1,1,1,0,0, 1,0,1,0,0,0,0,0,0,0,1, 1,0,1,1,1,1,1,0,1,1,1, 1,0,1,0,1,0,0,0,1,1,1, 1,0,1,0,0,0,1,2,0,0,1, 1,0,1,0,1,0,1,1,0,1,1, 1,0,0,0,1,0,0,0,0,0,1, 1,1,1,0,1,1,1,1,1,1,1,	1,1,1,0,1,1,0,1,1,1,1, 1,0,0,0,1,0,0,0,0,1,1, 1,0,1,0,1,0,1,0,0,0,1, 1,0,1,0,1,0,1,0,0,1,1, 1,0,1,0,0,0,1,0,0,1,1, 1,0,1,1,0,1,1,1,1,1,1, 1,0,0,0,0,0,0,0,1,1,1, 1,1,1,1,1,1,1,1,0,0,0, 1,0,0,0,0,0,0,0,1,1,1, 1,1,1,0,1,1,0,1,1,1,1,	1,1,1,1,1,1,1,1,1,1,1, 1,1,1,1,1,1,1,0,1,0,1, 0,0,0,0,0,0,0,0,1,0,0, 1,1,1,1,0,1,0,1,0,1,1, 1,0,0,1,0,1,0,0,0,1,1, 1,0,0,1,0,1,0,0,0,1,1, 1,0,0,1,0,1,0,0,0,1,1, 0,0,0,0,0,0,1,0,1,0,0, 1,0,1,0,1,1,0,1,0,1,1, 1,1,1,0,1,1,0,1,1,1,1,
7	1,1,1,1,1,1,1,1,1,1,1, 1,0,0,0,0,1,0,0,0,1,1, 1,1,1,1,0,1,1,1,0,0,1, 1,0,1,0,0,1,0,1,0,0,1, 1,0,1,0,1,1,0,0,2,1,1, 1,0,1,0,0,0,0,1,0,1,1, 1,0,1,1,1,1,1,0,1,0,0, 1,0,0,0,0,0,0,0,0,0,0, 1,0,0,0,1,1,0,0,0,1,1, 1,1,0,0,1,1,0,0,1,1,1,	1,1,1,0,1,1,0,1,1,1,1, 1,0,0,0,0,0,0,0,1,1,1, 0,0,1,1,0,1,1,0,1,0,1, 1,0,1,1,0,0,0,0,0,0,1, 1,0,0,0,0,0,1,1,0,1,1, 1,0,1,0,1,2,0,0,0,1,1, 1,0,1,0,0,0,1,0,1,1,1, 1,0,1,1,1,1,0,1,0,1,1, 1,0,0,0,0,0,0,0,0,1,1, 1,1,1,0,1,1,1,1,1,1,1,	1,1,1,0,1,1,0,1,1,1,1, 1,0,0,0,1,0,0,1,0,1,1, 1,1,0,1,1,0,0,1,0,0,1, 1,0,0,1,0,0,0,0,0,0,1, 1,0,1,1,0,1,1,1,0,1,1, 1,0,1,0,0,1,0,1,0,1,1, 1,0,0,0,0,1,0,1,0,1,1, 1,0,1,1,1,1,0,1,0,0,1, 1,0,0,0,0,0,0,1,0,1,1, 1,1,1,0,1,1,0,1,1,1,1,	1,1,1,1,1,1,1,1,1,1,1, 1,0,1,0,0,0,0,0,0,0,1, 0,0,0,0,1,1,0,1,0,0,1, 1,1,1,0,1,1,0,1,0,1,1, 1,0,1,0,0,0,0,0,0,0,1, 1,0,0,0,1,1,1,1,1,1,1, 1,0,1,0,0,0,0,0,0,0,1, 0,0,1,1,0,1,0,1,0,0,1, 1,0,1,0,0,1,0,1,0,1,1, 1,1,1,0,1,1,0,1,1,1,1,
8	1,1,1,1,1,1,1,1,1,1,1, 1,0,0,0,0,1,0,1,1,1,1, 1,0,1,1,0,0,0,1,0,0,1, 1,0,1,0,0,1,0,0,0,0,0, 1,0,0,0,0,1,0,1,0,1,1, 1,0,1,1,0,1,0,0,0,1,1, 1,1,1,1,0,1,1,1,0,0,1, 1,1,1,0,0,0,0,1,1,0,1, 1,1,2,0,1,1,0,1,1,1,1, 1,1,0,0,1,1,0,0,1,1,1,	1,1,1,0,1,1,1,1,1,1,1, 1,0,0,0,0,1,0,0,1,1,1, 0,0,1,1,0,1,1,0,1,0,1, 1,0,1,0,0,1,0,0,0,0,0, 1,0,0,0,1,1,1,1,0,1,1, 1,0,1,2,0,0,0,0,0,1,1, 1,0,1,0,1,1,1,0,1,1,1, 1,0,1,0,1,0,0,0,1,1,1, 1,0,0,0,0,0,1,0,0,0,1, 1,1,1,0,1,1,1,0,1,1,1,	1,1,1,0,1,1,0,1,1,1,1, 1,0,0,0,0,0,0,1,1,1,1, 1,0,1,1,1,1,1,0,1,0,0, 1,0,0,1,0,0,0,1,0,1,1, 1,1,0,1,0,1,1,1,0,1,1, 1,1,0,1,0,0,0,0,0,1,1, 1,1,0,1,1,1,1,1,1,1,1, 1,0,0,0,0,0,1,0,0,0,1, 1,0,1,0,1,0,0,0,1,1,1, 1,1,1,0,1,1,0,1,1,1,1,	1,1,1,1,1,1,1,1,1,1,1, 1,0,0,0,0,0,0,0,0,0,1, 0,0,0,1,0,0,1,0,0,0,0, 1,0,1,1,0,0,1,1,0,1,1, 1,0,0,0,0,0,0,0,0,0,1, 1,1,1,1,1,0,1,1,1,0,1, 1,0,0,0,0,0,0,0,0,0,1, 0,0,1,1,1,1,1,1,0,0,1, 1,0,0,0,1,1,0,0,0,0,1, 1,1,1,0,1,1,0,1,1,1,1,

9	1,1,1,1,1,1,1,1,1,1, 1,0,0,0,0,0,0,0,0,1, 1,0,1,1,1,1,1,1,0,0, 1,0,1,0,0,0,0,0,0,0, 1,0,0,1,1,1,1,1,1,1, 1,1,0,1,1,1,1,1,1,1, 1,1,0,0,0,0,0,1,0,0, 1,1,0,1,1,1,0,1,0,0, 1,1,0,0,2,0,0,0,0,1, 1,1,0,0,1,1,0,0,1,1,	1,1,1,0,1,1,1,0,1,1, 1,1,1,0,1,0,1,0,1,1, 0,0,0,0,0,0,0,0,1,0, 1,1,1,1,1,1,1,0,1,0,0, 1,0,0,2,0,0,0,0,0,1, 1,0,1,1,1,0,1,1,0,1, 1,0,1,0,0,0,0,0,0,1, 1,0,1,0,1,1,1,1,0,1, 0,0,0,0,0,0,0,0,0,1, 1,1,1,0,1,1,1,0,1,1,	1,1,1,0,1,1,0,1,1,1, 1,0,0,0,0,1,0,0,0,1, 1,0,1,0,0,1,0,0,0,0, 1,0,1,0,0,0,0,1,1,1, 1,0,1,1,1,1,1,0,0,1, 1,0,0,0,0,1,0,0,0,1, 1,0,0,0,0,1,1,1,1,1, 1,0,1,1,0,1,0,0,0,0, 1,0,0,0,0,1,0,1,1,1, 1,1,1,0,1,1,0,1,1,1,	1,1,1,1,1,1,1,1,1,1, 1,0,0,0,1,1,1,1,1,1, 0,0,1,0,1,0,0,0,0,0, 1,1,1,0,0,0,0,1,0,1, 1,0,0,0,1,0,0,1,0,1, 1,0,1,0,1,0,0,1,0,1, 1,0,1,0,1,0,0,1,0,1, 0,0,1,0,0,0,0,1,0,0, 1,0,1,0,0,0,0,0,0,1, 1,1,1,0,1,1,0,1,1,1,

Map Connections

To ensure all tiles can connect seamlessly without doing it through the code directly, the tiles are designed so that certain positions will always link. On every tile of its type, some edges in the CSV file are purposely made to be a walkable tile. Below an example of each block type is shown. The highlighted digits (tile) specify which locations are always exits. You'll notice that some tiles have more forced exits than others, this is because when the piece is rotated the coordinates that need to be free get rearranged as well.

The specified exits are on the 4th column of every block and the 3rd row of every block. No matter what the rotation, so long as its not the very edge of the map that position will be free to pass to the next block.

Corner Piece

1,1,1,1,1,1,1,1,1,1,
1,0,0,0,0,0,0,0,0,1,
1,0,1,1,1,0,0,1,1,0,
1,0,1,0,0,2,0,1,0,0,
1,0,1,0,1,0,0,1,0,1,
1,0,1,0,1,0,0,1,0,1,
1,0,0,0,0,0,0,1,0,0,
1,1,1,1,1,0,1,1,0,0,
1,0,0,0,0,0,0,0,0,1,
1,1,0,0,1,1,0,0,1,1,

Middle Piece

1,1,1,0,1,1,1,1,1,1,
1,0,0,0,0,0,0,0,0,1,
0,0,1,1,1,0,1,1,0,0,
1,0,1,0,0,0,0,1,0,1,
1,0,1,0,1,1,2,1,0,1,
1,0,1,0,1,0,0,1,1,1,
1,0,1,0,1,1,1,1,0,1,
1,0,1,0,0,0,1,1,0,1,
1,0,0,0,1,0,0,0,0,1,
1,1,1,0,1,1,1,1,1,1,

X Edge

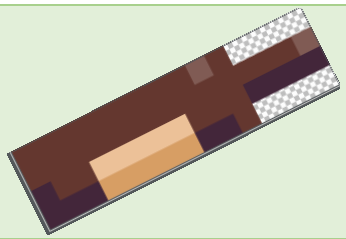
```
1,1,1,0,1,1,0,1,1,1,
1,0,0,0,0,0,0,0,0,1,
1,0,1,0,1,1,1,1,0,0,
1,0,1,0,0,0,0,1,0,1,
1,0,1,0,1,1,0,1,0,1,
1,0,1,0,1,1,0,1,0,1,
1,0,1,0,0,0,0,1,0,1,
1,0,1,0,1,1,1,1,0,0,
1,0,0,0,0,0,0,0,0,1,
1,1,1,0,1,1,0,1,1,1,
```

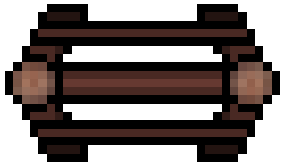
Y Edge

```
1,1,1,1,1,1,1,1,1,1,
1,0,0,0,0,0,0,0,0,1,
0,0,1,0,1,1,1,1,0,0,
1,0,1,0,1,0,0,0,0,1,
1,0,0,0,1,0,1,1,0,1,
1,1,1,1,1,0,1,1,0,1,
1,0,0,0,0,0,1,1,0,1,
0,0,1,1,1,1,1,1,0,0,
1,0,0,0,0,0,0,0,0,1,
1,1,1,0,1,1,0,1,1,1,
```

Items

There is a total of 4 items in the game that the player can find, use to aid themselves and defeat Jack. The item and their usage in the game is listed in the table below:

Item Name	Texture	Use
Bear trap	 (open) (Closed)	Place on the ground and lure Jack over it to hurt and stun him for a brief period of time. Does 3 damage.
Torch		Extends the Line of sight radius for a period of time, making it easier to navigate the maze, find more items, and avoid Jack.
Bottle		Thrown in the direction of movement until it collides with something. If it collides with Jack, do 1 damage.

Barrier	 (Active)  (Broken)	Place to make a tile not walkable. Deteriorates over 10 seconds.
Lore		Rare item that unlocks a piece of lore in the lore menu when collected. Different difficulties allow for different lore to be found.

Enemy

The enemy (Jack) is controlled by the AI manager. By checking what state AI is in, the enemy knows where to move and how fast to move. Jack has a total of 15HP that must be completely destroyed in order to win the game.

AI

The AI is separated to 5 states. In each state the enemy behaves differently. Some of the states are also affected by the difficulty chosen.

Roaming

AI goes in a random direction around the map until it either finds footprints or sees the player.

Searching

if the AI finds footprints, then the enemy follows the path of travel of these prints. This occurs until it sees the player or the trail runs into a dead end. On medium difficulty, if close enough to the player, then the searching will be overridden and A* pathfinding will be used to find the player. On hard difficulty A* is always used and on easy difficulty A* is never used. The search changing is done within a set radius to mimic the idea of Jack hearing your footsteps.

Chasing

When the player is visible to the enemy the AI tells it to move to the last known position of the player. It keeps doing this until the player can successfully get far away enough to not be seen.

Paused

Used to stop the enemy from moving for a short while when stunned by the player. This is done by using an item against Jack.

Dead

If the enemy does not have any HP remaining then the enemy is dead and the game is won.

Menus

Main Menu

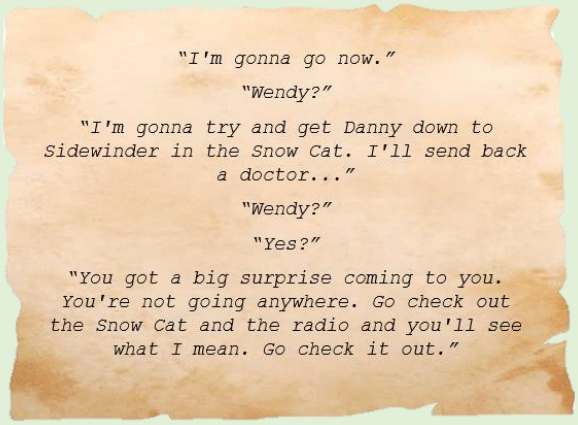
The game loads into the main menu with six game objects acting as buttons. Each button detects whether the curser is hovering over them and changes sprite to indicate to the player what is highlighted. When clicked, a scene ID is passed through the update function to either load a new scene, load a sprite or exit the game. The first three buttons will load the level class (actual game) and pass a difficulty through to load the correct settings. The fourth button will reveal to the player how to play the game and what the instructions are. While this sprite is loaded, clicking anywhere on the screen will remove it. The fifth button will change the scene to the lore menu, and the final button will exit the game.

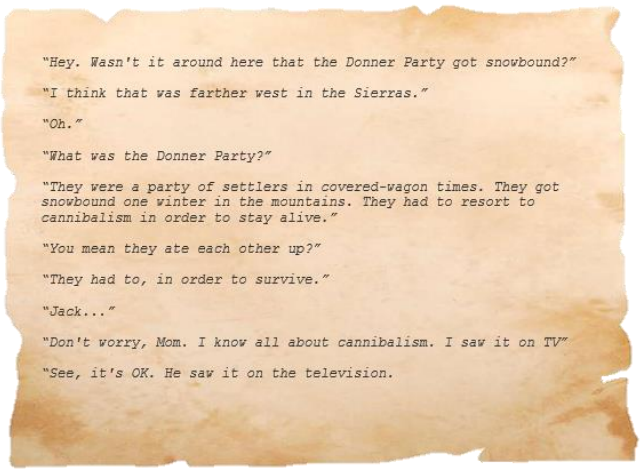
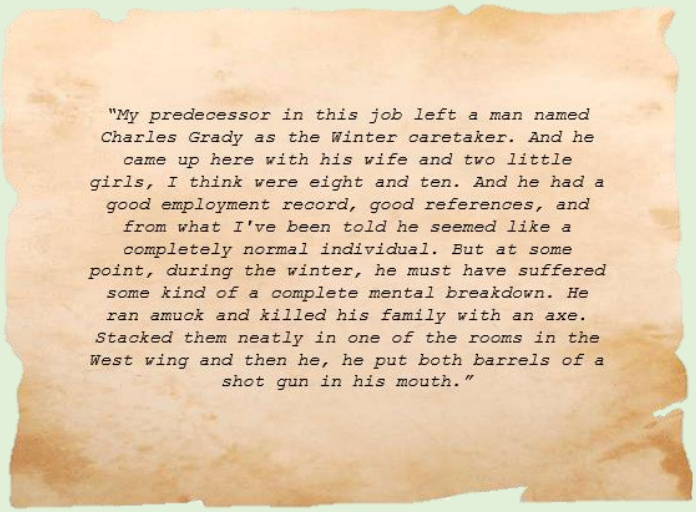
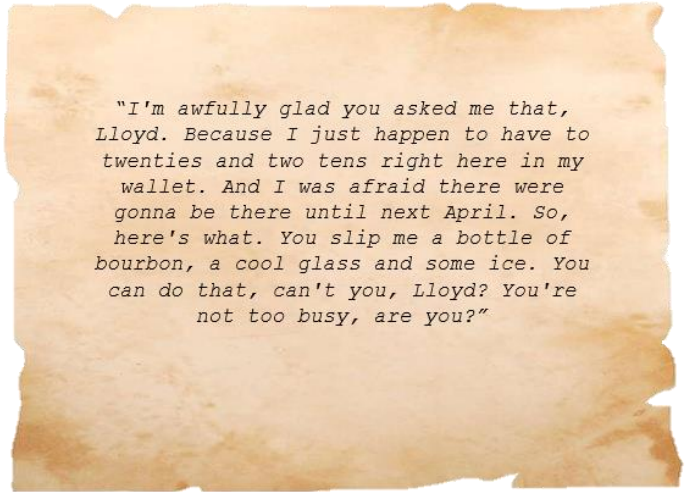
When the scene changes, the main menu is deleted and returned to a nullptr.

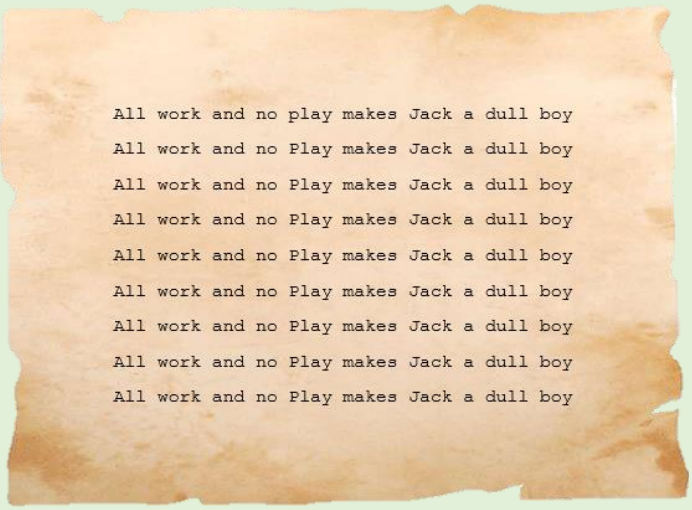
Lore Menu

The lore menu acts similar to the main menu. After the main menu has been deleted and the lore menu has been made, six game objects are loaded in, acting as buttons. The first five buttons represent the lore that the player has already picked up in game. If the player has never picked up the lore, the "locked" sprite is loaded in and the player cannot interact with it. If unlocked, the buttons will act similarly to the main menu buttons and become highlighted when the curser hovers over them. Clicking on a button will reveal a game object to the side of the screen that has the lore attached to it. Clicking anywhere on the screen will hide the game object.

The sixth button is a button that returns the player back to the main menu. When clicked, the lore menu will be deleted and returned back to a nullptr, and the menu scene will be restored.

Lore Number	Texture
1	

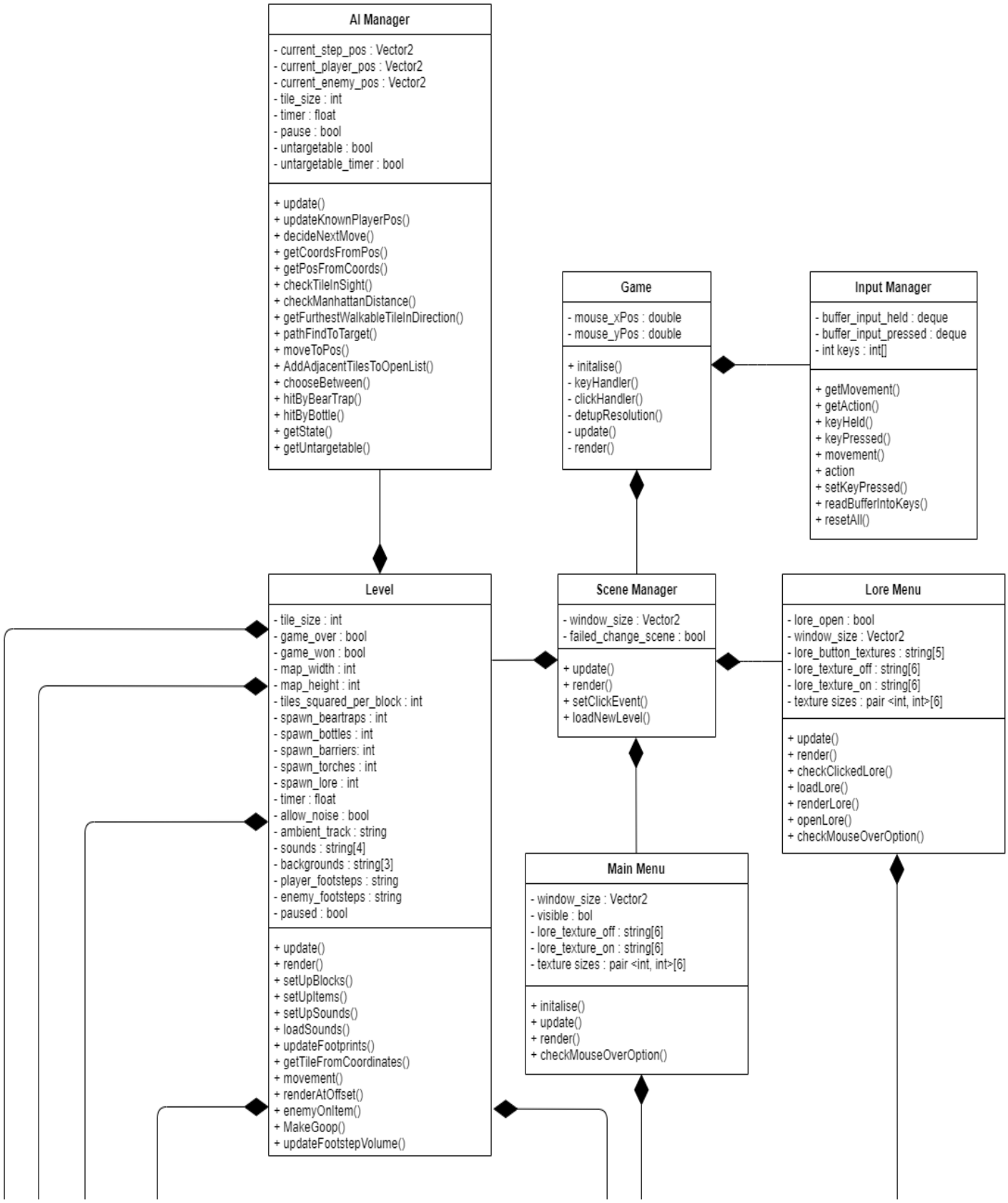
2	 "Hey. Wasn't it around here that the Donner Party got snowbound?" "I think that was farther west in the Sierras." "Oh." "What was the Donner Party?" "They were a party of settlers in covered-wagon times. They got snowbound one winter in the mountains. They had to resort to cannibalism in order to stay alive." "You mean they ate each other up?" "They had to, in order to survive." "Jack..." "Don't worry, Mom. I know all about cannibalism. I saw it on TV" "See, it's OK. He saw it on the television.
3	 "My predecessor in this job left a man named Charles Grady as the Winter caretaker. And he came up here with his wife and two little girls, I think were eight and ten. And he had a good employment record, good references, and from what I've been told he seemed like a completely normal individual. But at some point, during the winter, he must have suffered some kind of a complete mental breakdown. He ran amuck and killed his family with an axe. Stacked them neatly in one of the rooms in the West wing and then he, he put both barrels of a shot gun in his mouth."
4	 "I'm awfully glad you asked me that, Lloyd. Because I just happen to have to twenties and two tens right here in my wallet. And I was afraid there were gonna be there until next April. So, here's what. You slip me a bottle of bourbon, a cool glass and some ice. You can do that, can't you, Lloyd? You're not too busy, are you?"

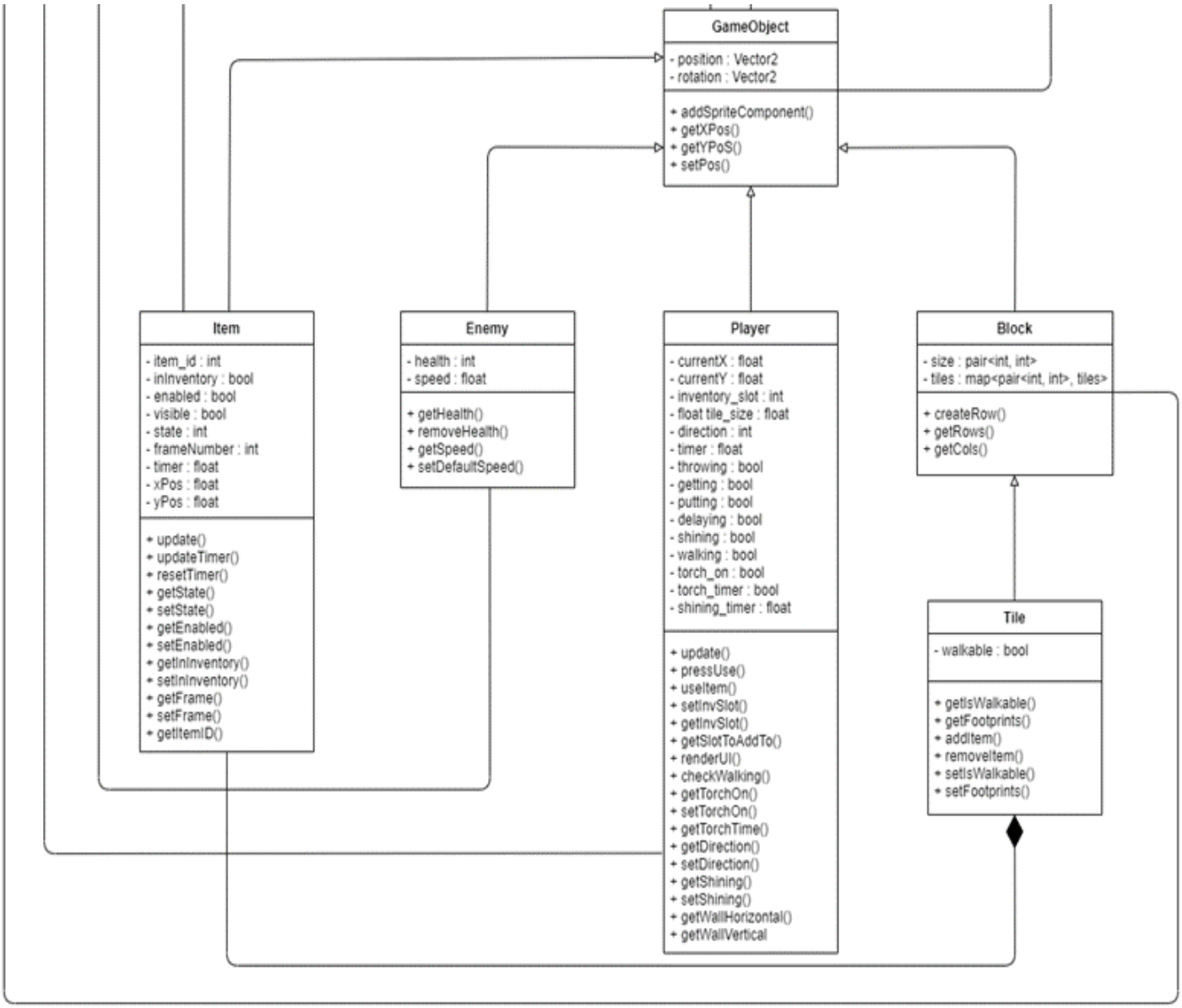
5	

Highlighted Button	Not Highlighted Button	Locked Button
Easy	Easy	n/a
Medium	Medium	n/a
Hard	Hard	n/a
Lore	Lore	n/a

		n/a
		n/a
		n/a
		
		
		
		
		

Class Diagram


AI Manager
- current_step_pos : Vector2
- current_player_pos : Vector2
- current_enemy_pos : Vector2
- tile_size : int
- timer : float
- pause : bool
- untargetable : bool
- untargetable_timer : bool
+ update()
+ updateKnownPlayerPos()
+ decideNextMove()
+ getCoordsFromPos()
+ getPosFromCoords()
+ checkTileInSight()
+ checkManhattanDistance()
+ getFurthestWalkableTileInDirection()
+ pathFindToTarget()
+ moveToPos()
+ AddAdjacentTilesToOpenList()
+ chooseBetween()
+ hitByBearTrap()
+ hitByBottle()
+ getState()
+ getUntargetable()
Game
- mouse_xPos : double
- mouse_yPos : double
+ initalise()
- keyHandler()
- clickHandler()
- detupResolution()
- update()
- render()
Input Manager
- buffer_input_held : deque
- buffer_input_pressed : deque
- int keys : int[]
+ getMovement()
+ getAction()
+ keyHeld()
+ keyPressed()
+ movement()
+ action
+ setKeyPressed()
+ readBufferIntoKeys()
+ resetAll()
Level
- tile_size : int
- game_over : bool
- game_won : bool
- map_width : int
- map_height : int
- tiles_squared_per_block : int
- spawn_beartraps : int
- spawn_bottles : int
- spawn_barriers: int
- spawn_torches : int
- spawn_lore : int
- timer : float
- allow_noise : bool
- ambient_track : string
- sounds : string[4]
- backgrounds : string[3]
- player_footsteps : string
- enemy_footsteps : string
- paused : bool
+ update()
+ render()
+ setUpBlocks()
+ setUpItems()
+ setUpSounds()
+ loadSounds()
+ updateFootprints()
+ getTileFromCoordinates()
+ movement()
+ renderAtOffset()
+ enemyOnItem()
+ MakeGoop()
+ updateFootstepVolume()
Scene Manager
- window_size : Vector2
- failed_change_scene : bool
+ update()
+ render()
+ setClickEvent()
+ loadNewLevel()
Lore Menu
- lore_open : bool
- window_size : Vector2
- lore_button_textures : string[5]
- lore_texture_off : string[6]
- lore_texture_on : string[6]
- texture sizes : pair <int, int>[6]
+ update()
+ render()
+ checkClickedLore()
+ loadLore()
+ renderLore()
+ openLore()
+ checkMouseOverOption()
Main Menu
- window_size : Vector2
- visible : bol
- lore_texture_off : string[6]
- lore_texture_on : string[6]
- texture sizes : pair <int, int>[6]
+ initalise()
+ update()
+ render()
+ checkMouseOverOption()


GameObject
- position : Vector2
- rotation : Vector2
+ addSpriteComponent()
+ getXPos()
+ getYPoS()
+ setPos()
Item
- item_id : int
- inInventory : bool
- enabled : bool
- visible : bool
- state : int
- frameNumber : int
- timer : float
- xPos : float
- yPos : float
+ update()
+ updateTimer()
+ resetTimer()
+ getState()
+ setState()
+ getEnabled()
+ setEnabled()
+ getInInventory()
+ setInInventory()
+ getFrame()
+ setFrame()
+ getItemID()
Enemy
- health : int
- speed : float
+ getHealth()
+ removeHealth()
+ getSpeed()
+ setDefaultSpeed()
Player
- currentX : float
- currentY : float
- inventory_slot : int
- float tile_size : float
- direction : int
- timer : float
- throwing : bool
- getting : bool
- putting : bool
- delaying : bool
- shining : bool
- walking : bool
- torch_on : bool
- torch_timer : bool
- shining_timer : float
+ update()
+ pressUse()
+ useItem()
+ setInvSlot()
+ getInvSlot()
+ getSlotToAddTo()
+ renderUI()
+ checkWalking()
+ getTorchOn()
+ setTorchOn()
+ getTorchTime()
+ getDirection()
+ setDirection()
+ getShining()
+ setShining()
+ getWallHorizontal()
+ getWallVertical
Block
- size : pair<int, int>
- tiles : map<pair<int, int>, tiles>
+ createRow()
+ getRows()
+ getCols()
Tile
- walkable : bool
+ getIsWalkable()
+ getFootprints()
+ addItem()
+ removeItem()
+ setIsWalkable()
+ setFootprints()