# Final Project Log:

## Week 1-2 (18/01/23 – 01/02/23):

Aims:

* Create a demo of the game.
* Create a random nxn dungeon.
* Create an AI player that can navigate from the start to the end.

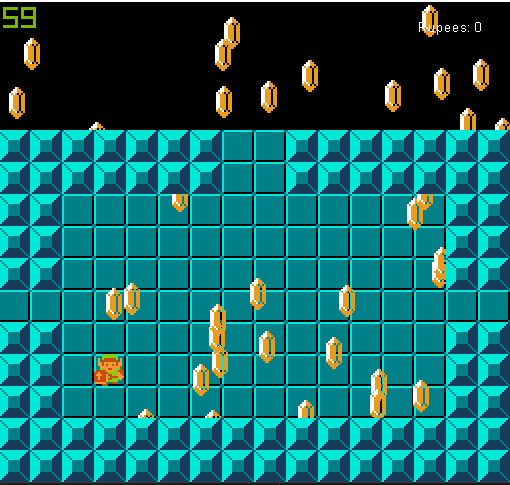
### Day 1 (20/01/23):

* Created room layout, moving player, and collectible rupees.
* Walls are Sprites that are added manually, and overlap with each other.
* 5 pixels per frame = 300 pixels per second?



Goal for next day: Turn room layout into a matrix. Add walls to list using this matrix.

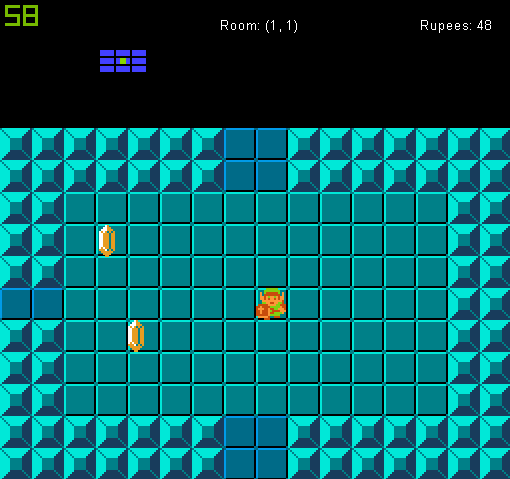
### Day 2 (21/01/23):



* Walls are now determined by 2D arrays.
* Rooms have 4 flags stating where their doors are.

Tomorrow: Transition between different rooms.

### Day 3 (22/01/23):



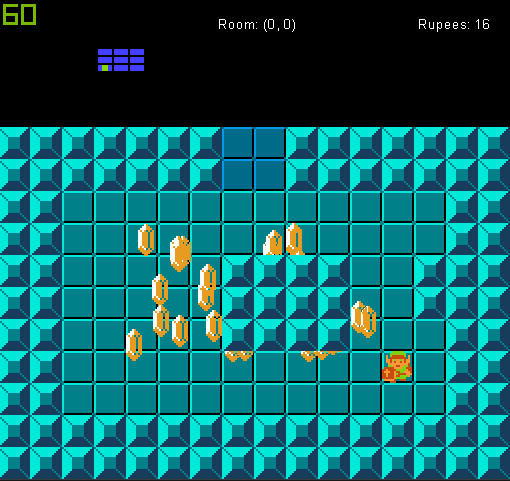
* Rooms are now stored in a matrix. Rooms are connected via 4 links that they have to neighbouring rooms. Inefficiencies from copied code removed.
* Minimap added, shows layout of dungeon, and link’s current room.
* Made a 3x3 maze with 2 branching paths.

Minimap dimensions:

* Link/triforce: 3x3 -> 9x9
* Rooms: 7x3 -> 14x6
* Room gap: 1 wide -> 2

Tomorrow: Start working on A\* to navigate to doors.

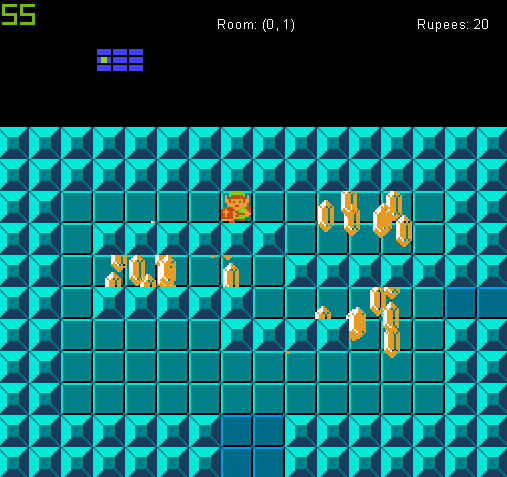
### Day 4 (23/01/23):



* Started working on A\* navigation.
* Arcade’s in-built algorithm was functional but failed to account for walls somehow and was too restricting.
* Began work on my own algorithm that I can tailor to my liking.
* Uses a grid of nodes rather than a list.
* Reaches goal, but not very well.
* Takes a very roundabout path due to no heuristic.
* Does not seem to care about walls, tries to ram through them.
* If it tries to go through 3 or more walls the game crashes.

Tomorrow: Try and fix A\*. Check if neighbours are correct, walls are not added etc.

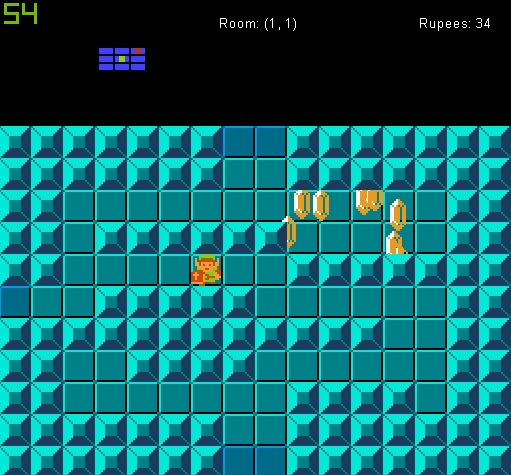
### Day 5 (25/01/23):



* Fixed A\* algorithm, it now always goes to the correct location, taking the fastest path (this was due to not removing nodes from open set), and without going through walls (due to misaligning the graph creation).
* Link will now restart A\* again when travelling to a new room, meaning he will soon be able to fully navigate between rooms.
* Split room and A\* related functions and classes into other files to structure the code. Will likely do this with the player in the future as well.

Tomorrow: Make Link locate a door to walk towards to initiate A\* without me specifying his goal.

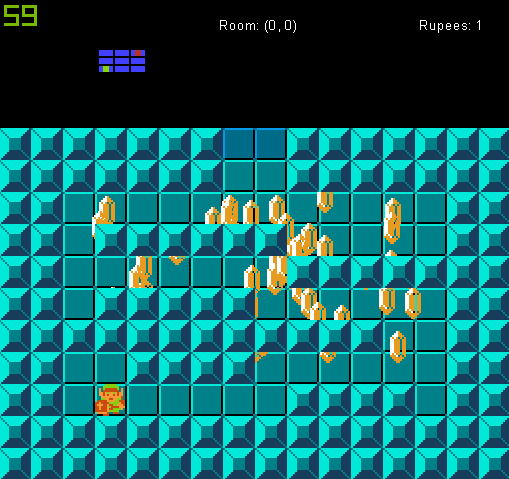
### Day 6 (26/01/23):



* A\* now restarts automatically when entering a new room.
* Will target a random door in the room that isn’t the one Link just came from. Stops when at a dead end.
* Added a marker to show where the final room should be.
* Made a GitHub repository to backup my files.

Tomorrow: Implement a depth-first search to reach the goal.

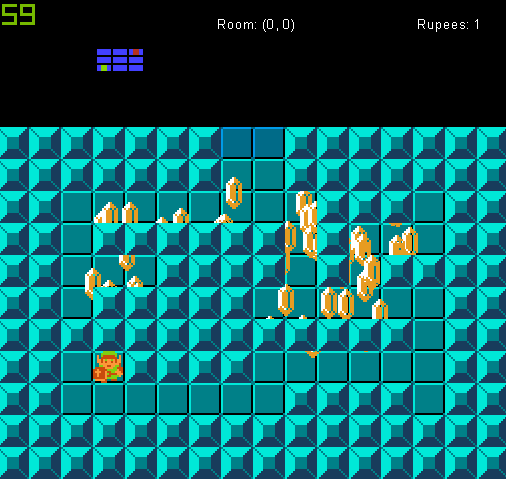
### Day 7 (29/01/23):



* Moved much all of Link’s AI into a player class to organise the code better, and keep all of the functionality self-contained.
* Began preparing for Link to choose between his own objective.
* Currently does not start moving automatically as the way A\* worked previously no longer matters.
* Began implementing the depth-first search.

Tomorrow: Continue with depth-first search. Hopefully help guide it to the end, backtracking may be difficult.

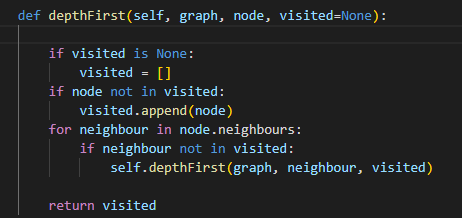
### Day 8 (30/01/23):



* Transferred all map related code to the Map class, tidying up the main file even further.
* Now that the map itself is an object consisting of rooms, it can be traversed using a depth-first search more easily.

Tomorrow: Implement depth-firs search using the Map class.

### Day 9 (31/01/23):



* Implemented depth-first search that returns a list of rooms in the order they should be visited by Link.
* There are no repeats, meaning that the path Link needs to take to backtrack is not accounted for.

Tomorrow: Find a way to include the backtracked path, and/or use the depth-first search to determine Link’s next move.

## Week 3-4 (01/02/23 – 15/02/23):

Aims:

* Implement depth-first search fully, keeping track of room history.
* Implement parametrised dungeon generation.

### Day 10 (02/02/23):

Idea:

When a branch is reached, add room to a stack?

Choose one door .

Straight line to next room. Add to stack.

Another branch is reached, at this to stack.

Choose door.

Dead end. Pop stack, return to last room.

Check other door.

Straight line, add to stack.

Dead end, pop from stack and return to last room.

Return again, now all doors have been checked. Pop from stack.

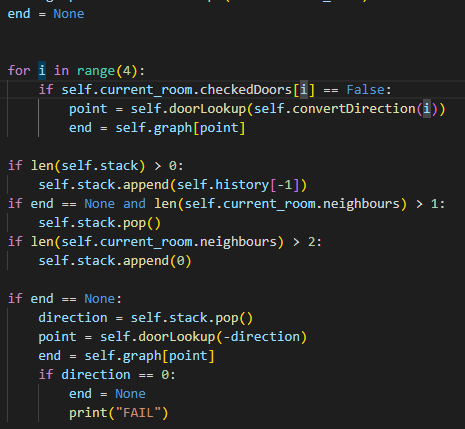
Pop stack again, go through straight room to Original branch.

Can now continue with other door.

If a branch is found, start adding every room to the stack. Add a room to the stack if it has more than 2 doors, or if the stack already has something in it.

Keep track of which of a room’s doors have been explored. When backtracking to a previous room, this can be used to look at the other path.

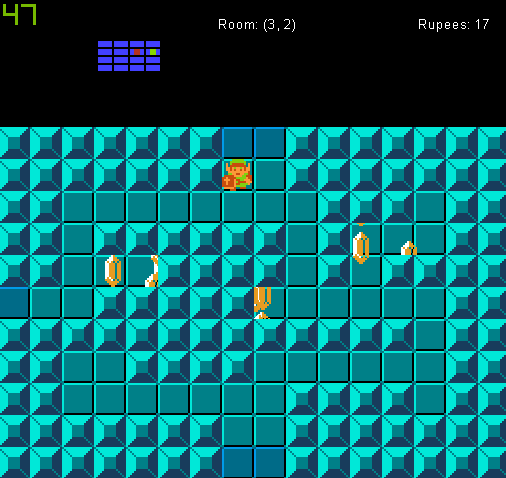
### Day 11 (03/02/23):



* Depth-first search now implemented and working.
* Link will choose a path when reaching a branch, then once he reaches a dead end will return and check the other path.
* If all paths are exhausted he will return to original branch and stop. This shouldn’t happen normally as he should fine the exit before this ever happens, but may be a problem later.
* Still needs testing with bigger mazes that have branches off of branches.

Tomorrow: Start working on maze generation. Will need to use it to create bigger mazes in order to fully test depth-first search. <https://www.algosome.com/articles/maze-generation-depth-first.html>

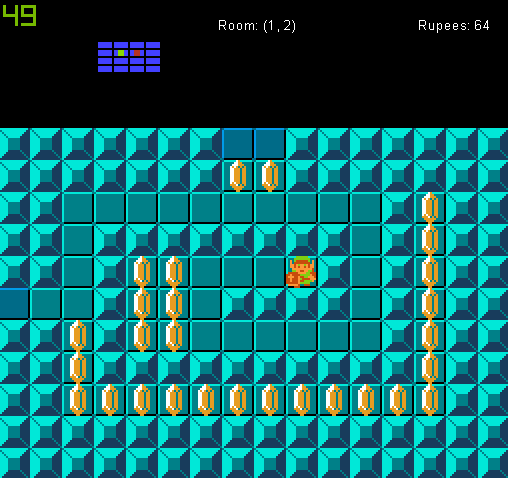
### Day 12 (06/02/23):



* Implemented depth-first maze generation.
* Creates fully traversable mazes, with any size.
* No longer need to define room connections manually in any way.
* Depth-first traversal still works, but breaks when a dead end is reached early on before a branch. Likely need to add to just check first room again? Shouldn’t be able to reach a dead end without already reaching a branch.
* Depth-first traversal could break when multiple paths are added. Will definitely need to implement A\* as well.

Tomorrow: Start making the rooms unique from each other. Have their own rupee list, and own floor plan. Make several floor templates that the rooms will pick from.

### Day 13 (07/02/23):



* There are now 14 predefined templates for rooms to use, all stored in a list for easy access. Currently, they are randomly chosen between to make each room feel special.
* Rupee lists now belong to individual rooms rather than the game itself. This means that each room can have its own amount, and they will disappear when leaving the room. They will not reappear when returning to the room if they have been collected. Currently, every floor tile spawns a rupee.
* Enemy list has been made, yet to be implemented.
* Much obsolete code has been deleted.

Tomorrow: Iron out how rupees are spawned. Begin working on targeting rupees as their own objectives.