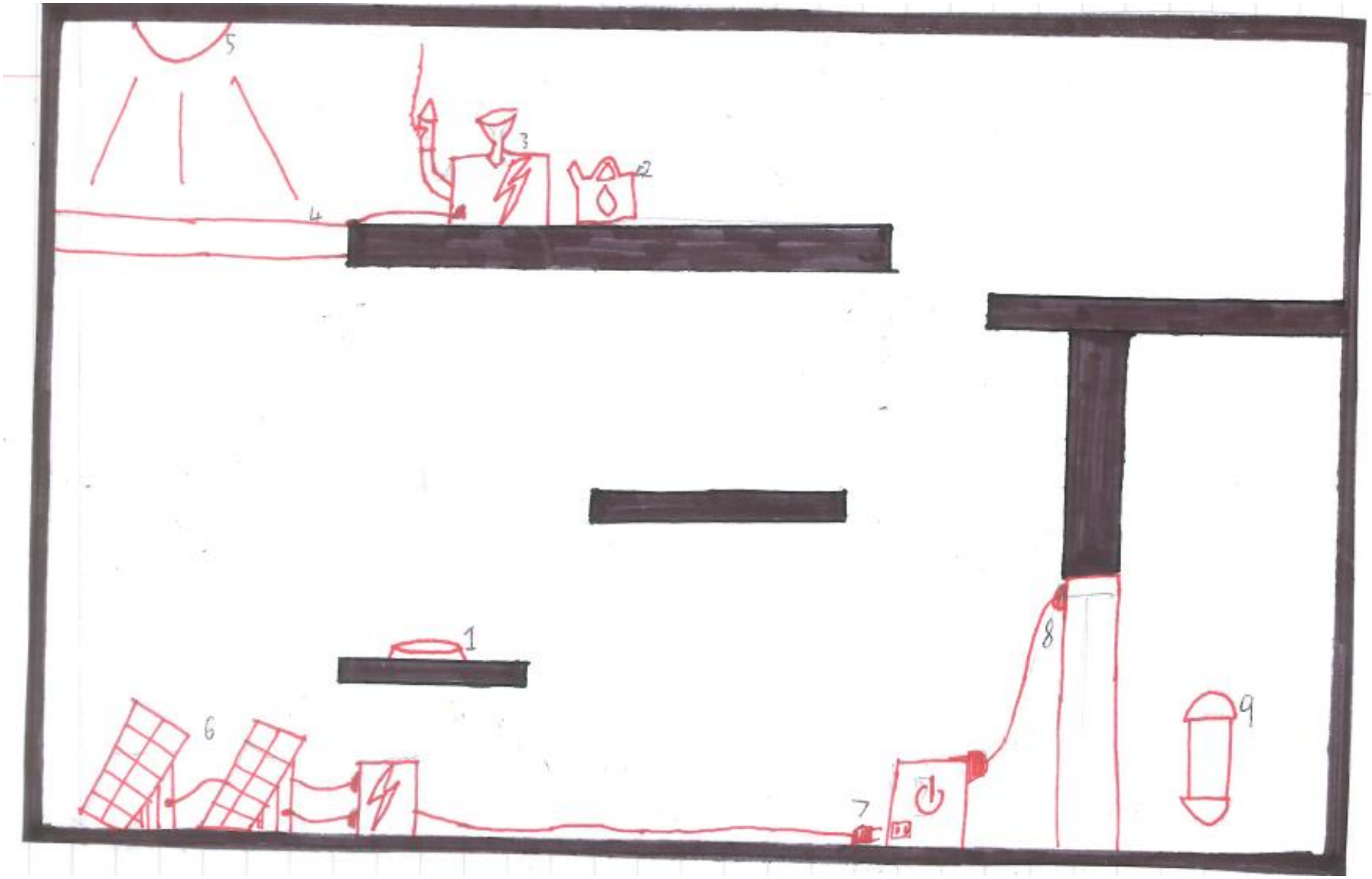
P1 Level Design -Mechanics and Layout

As this is the first map of the layout and mechanics, I will quickly explain it. All parts of the level can be broken down into two main aspects. The static level (shown in black) and the interactive level (shown in red). The static level acts as a platform, being solid and stationary, nor will it ever change from its current state.

On the other hand, anything red is either dynamic or interactive. This means that either interacting with them (by pressing spacebar) will cause them to change state (e.g. off to on) or when certain conditions are met (e.g. when object 4 is open) the object itself will change state as long as the conditions are met (e.g. when object 4 is open: enter powered state).

The parts of the interactive level have been ordered by number. I will now begin to explain dynamic or interactive object that how they contribute towards level progression:

Name: P1
Class: Stage
States: Incomplete, Complete
Can transfer states: no
Starting state: Incomplete
Algorithm(s):
If [P.Teleporter]: {on},
enter {Incomplete}.
While: {Incomplete},
[P1] will be assigned to main instance.
If [Battery]: {Win},
enter {complete}.
While: {complete},
[P1] will be deleted.


5
3
2
4
1
8
9
6
7

1. *The teleporter*: The playable character will start at the teleporter every level. This time you start of a platform.

Name: P1.Teleporter

Class: Stage object
States: Off, on
Can transfer states: no
Starting state: Off
Algorithm(s):
animation: teleporter
If [P1]: {Off};
Load [Player] in +0 spaces

2. *Fuel*: Jump on the platforms to reach this. Interact with object 2 to hold it.

Name: Fuel

Class: Hold Object
States: Idle, Hold
Can transfer states: Yes
Starting state: Idle
Algorithm(s):
While: {Idle}; animation: Idle.
If {interacting} occurs within +-1 spaces,
enter {hold}.
While: {hold}; animation: none,
Transfer {fuel} to player.
While: {hold},
If {interacting} occurs within +-1 spaces,
Enter {Idle},
Transfer {fuel} from player.

3. *Generator*: If interacted while player holds object 2, object 3 will enter a powered state for 5 seconds, and then enter a broken state.

Name: Generator

Class: Interactive Object
States: Unpowered, Powered
Can transfer states: Yes
Starting state: Unpowered
Algorithm(s):
While: {Unpowered}; animation: Unpowered,
If {interacting} occurs within +-1 spaces,
While [Player]: {fuel},
Transfer {fuel}.
If: {fuel}, enter {powered}.
While: {powered}; animation: powered
Wait 4 seconds; animation: broken.

4. *Powered platform*: When in an unpowered state, it will be horizontal. When object 3 is in a powered state, object 4 will enter a powered state. When is a powered state, object 4 it will swing down (via the right platform) to hang vertically.

Name: Powered platform

Class: Reactive Object
States: Unpowered, Powered
Can transfer states: no
Starting state: Unpowered
Algorithm(s):
While: {Unpowered}; animation: Unpowered,
mimic {[platform]}.
If [Generator]: {powered},
enter {powered}.
While: {powered}; animation: none.
Mimic {[none]}.

5. *High powered lamp*: Shines an intense light vertically down. When object 4 is in a powered state, object 5 will enter an unblocked state, causing it to change animation to extend the light further down.

Name: High Powered Lamp

Class: Reactive object
States: Off, on
Can transfer states: no
Starting state: Off
Algorithm(s):
While: {Off}; animation: blocked
If [Powered platform]: {powered},
enter {on}.
While: {on}; animation: unblocked

6. *Solar panels*: will enter a powered stated if object 5 is in an unblocked state. This will be represented by a change in animation by the box they are connected to.

Name: Solar Panel
Class: Reactive Object
States: Unpowered, Powered
Can transfer states: no
Starting state: Unpowered
Algorithm(s):
While: {Unpowered}; animation: blocked
If [High Powered Lamp]: {on},
enter {powered}.
While: {powered}; animation: powered

7. *Power Box*: When interacted with, will change state from unplugged to plugged. If object 6 is in a powered state and object 7 is in a plugged state, then object 7 will enter a powered state.

Name: Power Box
Class: Interactive Object
States: Unplugged, Plugged, Powered
Can transfer states: no
Starting state: Unplugged
Algorithm(s):
While: {Unplugged}; animation: Unplugged.
If {interacting} occurs within +-1 spaces,
enter {plugged}.
While: {plugged}; animation: plugged.
If: {plugged},
If [Solar panel]: {powered},
enter {powered}.
While: {powered}; animation: powered

8. *Gate*: If object 7 is in a powered state, then object 8 will change state from locked to unlocked. This changes its animation and its interaction, removing its ability to block users.

Name: Gate
Class: Reactive Object
States: Unpowered, Powered
Can transfer states: no
Starting state: Unpowered
Algorithm(s):
While: {Unpowered}; animation: Unpowered,
Mimic {[platform]}.
If [Box]: {powered},
enter {powered}.
While: {powered}; animation: none.
Mimic {[none]}

9. *Battery*: When the player touches object 9 the level instance enters a win state. This will end and lock the instance while opening the neutral area in a P1 clear state (This will change certain visuals of the neutral area).

Name: P1 Battery

Class: Stage Object
States: On, Off, Win
Can transfer states: no
Starting state: On
Algorithm(s):
While: {On}; animation: On.
If [Player] enters +-0 spaces,
enter {Off}.
While: {Off}; animation: Off.
If {Off},
Wait 5seconds
enter {Win}.