The Neutral Area

My project is to promote GCSE science, this gives me three sciences each split into and further three different sections (e.g. P1, P2 and P3 for triple science). This means that there are 9 levels that will introduce the player to different themes.

While designing levels will solve the problem of the much needed obstacles, the next questions is how will the player get to these levels? The idea of a neutral area between the levels has been already been covered in the main concept section. It exists as an area unrelated to the 3 sciences to serve as a way to get between their own areas without ruining the balance between them.

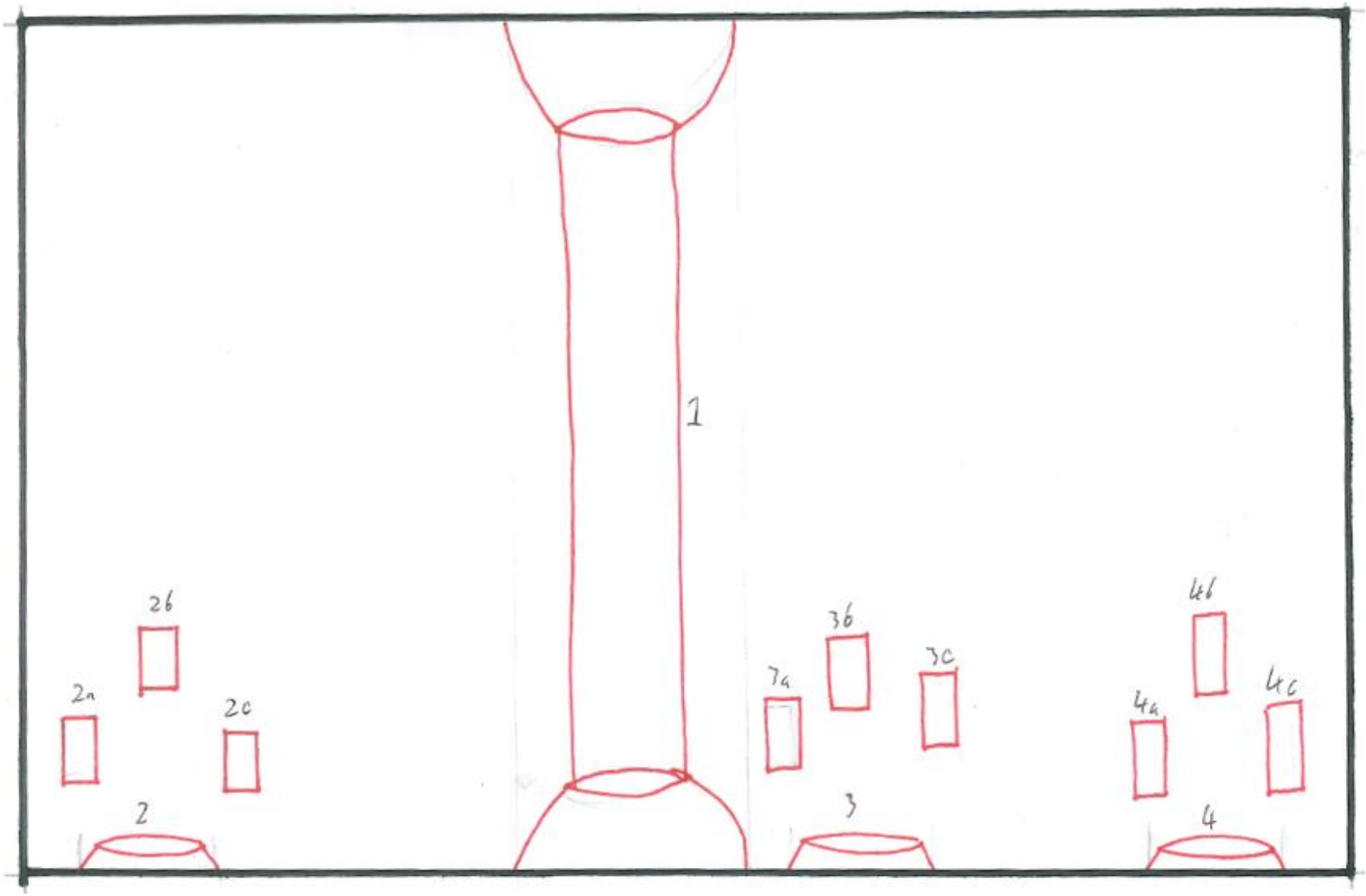
Mechanically, you will start in this area. As it is circular in design, the screen will loop if you walk to one edge (or you'll just come out of the other side, I'll wait to actual programming to decide what one would be a better choice). I keep on mentioning that this serves as the area to get to each level, it does this by having the entrances to the levels as part of it.

In this area there will be three teleporters. Each teleporter will glow a colour representing their science. When the player walks up and interacts with one of the teleporters, it will load up the lowest unlocked level instance (this will be explained in a moment) and close the neutral area. This action will effectively teleport the player from the neutral area to the level instance.

When a level is cleared (when the player touches a battery), that instance will be closed and the neutral area will once again be loaded, but in a different state. Like the nine science levels, the neutral area can be separated into interactive and static components. Unlike the science levels, there are a lot less of both.

The only interactive components are the teleporters. These work by loading up the lowest unlocked level (when a battery is collected in a level, the level becomes locked, preventing it from being accessed again). The dynamic components are certain visual aspects of the area; these will change depending on the stackable states that the player will unlock by collecting batteries (e.g. P1, P2, B1, B2, B3, C1 are all states that can be active at the same time), these will change certain dynamic areas to change visuals to show that these areas are completed.

When all nine states exist at the same time, the neutral area will actually be closed and will load up a different neutral area.


1
2a
2b
2c
2
3a
3b
3c
3
4a
4b
4c
4

Name: Neutral area

Class: Stage
States: Start, On, Win
Can transfer states: no (Can hold multiple states)
Starting state: Start
Algorithm(s):
Game Start.
While: {Start}; animation: start.
[Neutral area] will be assigned to main instance.
If [Player]: {interacting},
Enter {On}.
While: {On}; animation: On.
If [P1 slot, P2 slot, P3 slot, B1 slot, B2 slot, B3 slot, C1 slot, C2 slot, C3 slot]: {On}
Enter {Win}
While: {Win}; animation: Win.

1) *Main pillar*: When all three batteries in a science are on, it will change animation to produce a stream of energy (the left, right and middle are all separate parts).

Name: Main pillar

Class: Reactive Object
States: Off, On
Can transfer states: no
Starting state: Off
Algorithm(s):
Game Start.
While: {Off}; animation: Off.
Left section
If [P3 Battery]: {Win}
Enter {On}.
While: {On}; animation: On.
Middle section
If [B3 Battery]: {Win}
Enter {On}.
While: {On}; animation: On.
Right section
If [C3 Battery]: {Win}
Enter {On}.
While: {On}; animation: On.
If [Left, Right, Middle]: {on},
Enter On.
While: {On}; animation: On.

2) *P.Teleporter*: When interacted with, will change to an on state. This will cause the physics

levels to load.

Name: P.Teleporter

Class: Interactive object
States: Off, on
Can transfer states: no
Starting state: Off
Algorithm(s):
While: {Off}; animation: teleporter
If {interacting} occurs within +-1 spaces,
enter {on}.

3) *P1 slot*: Will change animation to show a battery in the slot when the battery in level P1 has been collected.

Name: P1 slot

Class: Reactive Object
States: Off, On
Can transfer states: no
Starting state: Off
Algorithm(s):
Game Start.
While: {Off}; animation: Off.
If [P1 Battery]: {Win}
Enter {On}.
While: {On}; animation: On.

4) *P2 slot*: Will change animation to show a battery in the slot when the battery in level P2 has been collected.

Name: P2 slot

Class: Reactive Object
States: Off, On
Can transfer states: no
Starting state: Off
Algorithm(s):
Game Start.
While: {Off}; animation: Off.
If [P2 Battery]: {Win}
Enter {On}.
While: {On}; animation: On.

5) *P3 slot*: Will change animation to show a battery in the slot when the battery in level P3 has been collected.

Name: P3 slot
Class: Reactive Object
States: Off, On
Can transfer states: no
Starting state: Off
Algorithm(s):
Game Start.
While: {Off}; animation: Off.
If [P3 Battery]: {Win}
Enter {On}.
While: {On}; animation: On.

6) B.Teleporter: When interacted with, will change to an on state. This will cause the biology levels to load.

Name: B.Teleporter
Class: Interactive object
States: Off, On
Can transfer states:
Starting state: Off
Algorithm(s):
While: {Off}; animation: teleporter
If {interacting} occurs within +-1 spaces,
enter {on}.

7) *B1 slot*: Will change animation to show a battery in the slot when the battery in level B1 has been collected.

Name: B1 slot
Class: Reactive Object
States: Off, On
Can transfer states: no
Starting state: Off
Algorithm(s):
Game Start.
While: {Off}; animation: Off.
If [B1 Battery]: {Win}
Enter {On}.
While: {On}; animation: On.

8) *B2 slot*: Will change animation to show a battery in the slot when the battery in level B2 has been collected.

Name: B2 slot
Class: Reactive Object

States: Off, On
Can transfer states: no
Starting state: Off
Algorithm(s):
Game Start.
While: {Off}; animation: Off.
If [B2 Battery]: {Win}
Enter {On}.
While: {On}; animation: On.

9) *B3 slot*: Will change animation to show a battery in the slot when the battery in level B3 has been collected.

Name: B3 slot
Class: Reactive Object
States: Off, On
Can transfer states: no
Starting state: Off
Algorithm(s):
Game Start.
While: {Off}; animation: Off.
If [B3 Battery]: {Win}
Enter {On}.
While: {On}; animation: On.

9) C.Teleporter: When interacted with, will change to an on state. This will cause the chemistry levels to load.

Name: C.Teleporter
Class: Interactive object
States: Off, on
Can transfer states: no
Starting state: Off
Algorithm(s):
While: {Off}; animation: teleporter
If {interacting} occurs within +-1 spaces,
enter {on}.

10) *C1 slot*: Will change animation to show a battery in the slot when the battery in level C1 has been collected.

Name: C1 slot
Class: Reactive Object
States: Off, On
Can transfer states: no
Starting state: Off
Algorithm(s):

Game Start.
While: {Off}; animation: Off.
If [C1 Battery]: {Win}
Enter {On}.
While: {On}; animation: On.

11) *C2 slot*: Will change animation to show a battery in the slot when the battery in level C2 has been collected.

Name: C2 slot

Class: Reactive Object
States: Off, On
Can transfer states: no
Starting state: Off
Algorithm(s):
Game Start.
While: {Off}; animation: Off.
If [C2 Battery]: {Win}
Enter {On}.
While: {On}; animation: On.

12) *C3 slot*: Will change animation to show a battery in the slot when the battery in level C3 has been collected.

Name: C3 slot

Class: Reactive Object
States: Off, On
Can transfer states: no
Starting state: Off
Algorithm(s):
Game Start.
While: {Off}; animation: Off.
If [C3 Battery]: {Win}
Enter {On}.
While: {On}; animation: On.