Classifying Celeste

About Celeste

Celeste is a single-player 2D platformer developed by Maddy Thorson and Noel Berry. The game is about you being Madeline who is climbing to the peak of the mountain "Celeste". The goal of the game is to overcome obstacles on the way and make it to the end of the level.

The game consists 7 levels which consists of sub parts. In the each subpart you have to reach an exit point without taking damage. Taking damage resets you back to the starting point.

About the Player

Madeline is the main character whose actions are governed by our controls. She is restricted in **8 directions of motion** and primarily has 3 special moves other than basic left and right movement. those are:

Jump

Allows her to go over gaps.

She has the ability to jump to a certain height which can be then done again only when she hits the ground.

Dash

dashing gave her extra momentum to make to the platform.

When Madeline has the "charge" she is able to dash in any direction, this gives her extra momentum and ability to dash into "space blocks" (described later). The charge gets used up when she dashes and is restored when she hits the ground or passes through the space block.
(There are other objects which also recharge her, but those are not used in the proof)

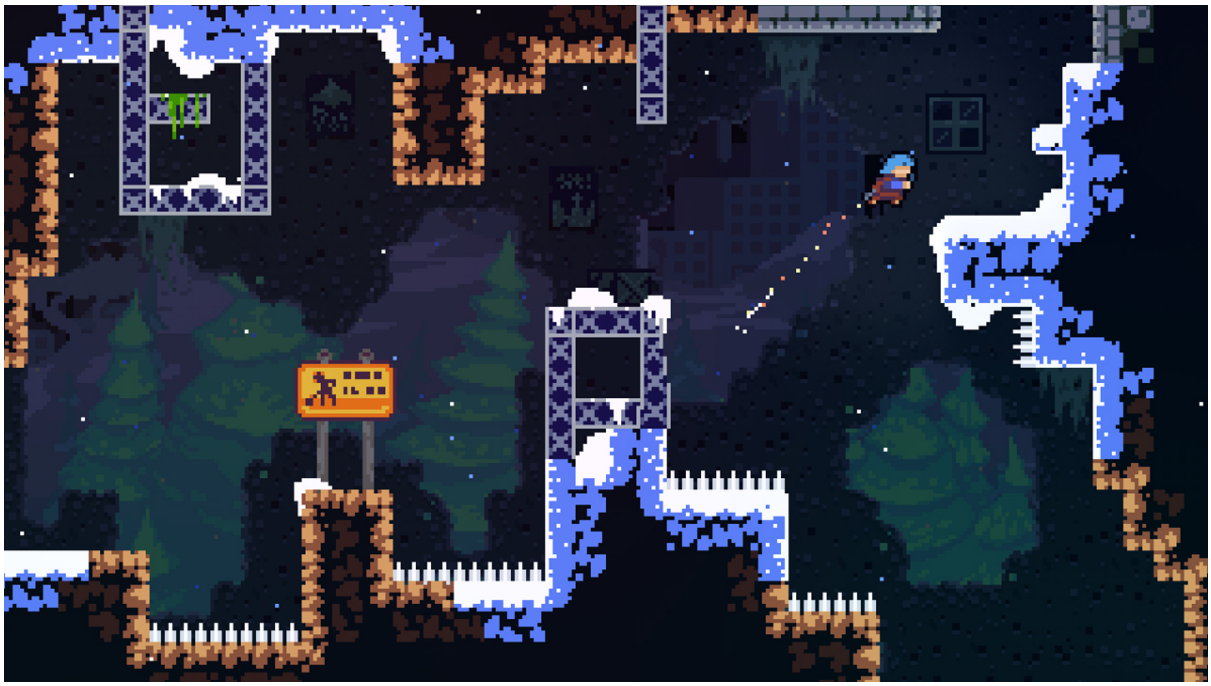
Grab

She is able to climb walls with this ability

Since she is a mountain climber, she has the ability to climb walls and wedges. But she has stamina, which limits the amount of time she can grab onto a wall before sliding down. This stamina is restored when she makes contact with the ground.

Level Implementation

Frames



bottom left corner is the entry point and the top right corner is the exit point.

What are frames?

Frames are areas of games which has a start and an endpoint. You have access to one frame at a time, using the entries and the exits you move from 1 frame to another. So Frames serve as the checkpoints in the game. Frames does not have a limit of size, since your screen can scroll.

The game has been split into such frames, which are puzzles on their own, which require planning and reflexes to reach the endpoint. There are multiple ways to solve these puzzles due to the versatility of the game and the mechanisms in it.

A graph of such frames connected together makes up a level of the game. To construct our proof, we will make frames which we will join together to make our level.

Objects

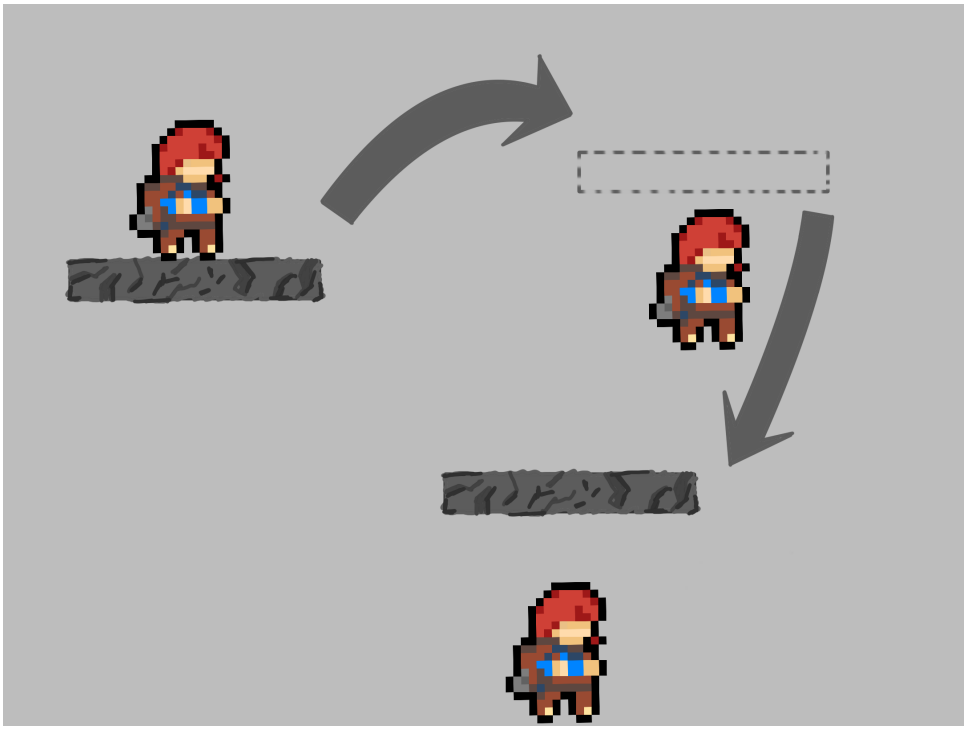
Revolving around the basic 3 operations the game has a lot of mechanisms that add to the fun and the difficulty of the game. These are added to the game as the player makes progress in the levels.

For our proof, we will mainly use 3 of the objects. They are:

- Unstable Platform
- Button door
- Space block

The purpose of these objects will be explained later.

Unstable platform

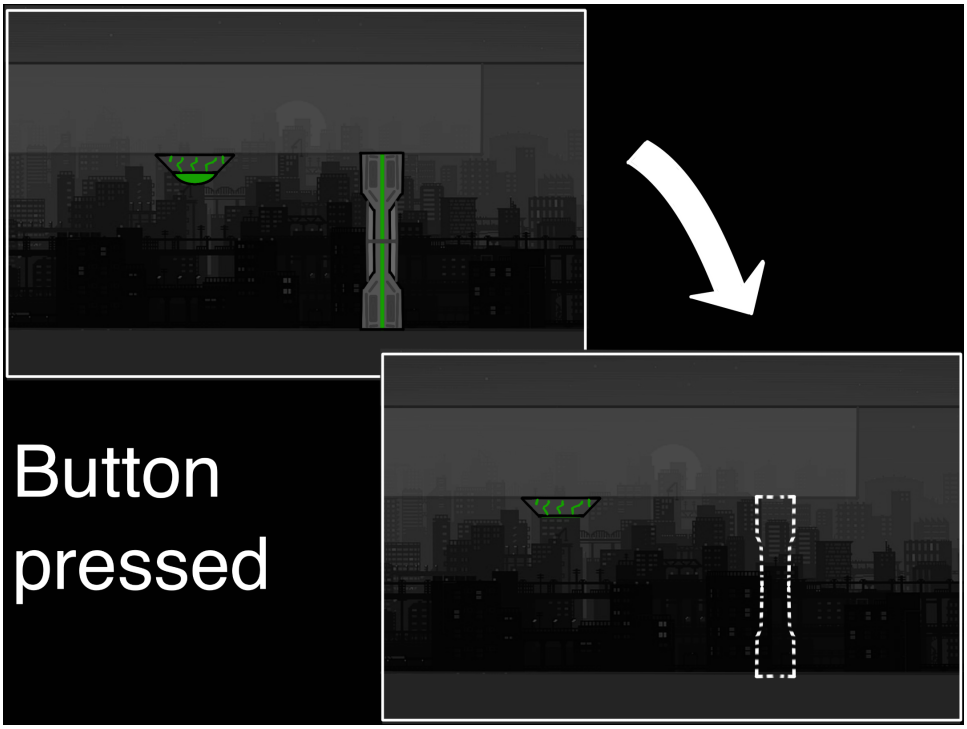


Forms back at the same position as before.

A stone platform that can float, this platform breaks when Madeline stands on this platform for more than a second. After the platform breaks, Madeline if not jumped will fall down.

This platform then forms back in the same place. But it can only be broken when stood upon and can not be broken from below, making it like a trap door if placed correctly.

Button door

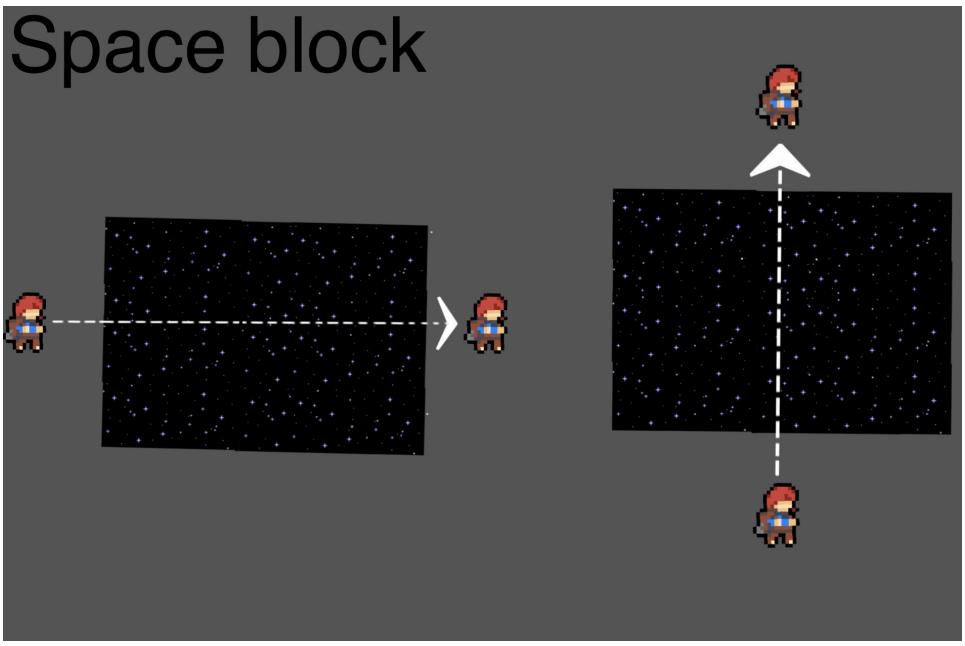


Multiple doors in a single frame are allowed. But one button per door.

A door that can only be unlocked using its specific button. This button must be placed in the same frame as the door, but it has the freedom to be placed anywhere in the frame.

Once the door is opened it can not be closed again.

Space Block



A block of celestial material which lets you float into and out in a straight line when you dash into it. Once Madeline dashes into the block, you can not stop her from reaching the other opening of the block in a straight line.

If the other side of the line is blocked with a wall, Madeline dies and respawns at the start of the frame.

Level put together

A level consists of many frames, but there is only 1 flag at the top of the level and there is only 1 initial start point of the level. For most of the levels, there is only 1 linear path from the start to end with a sequence of frames, but some other levels are more complex, which include sub-tasks and other detours.

Here is an example, this is the 1st level in the game, the frames have been arranged according to the order.

Bottom left being the start and the top right being the end.

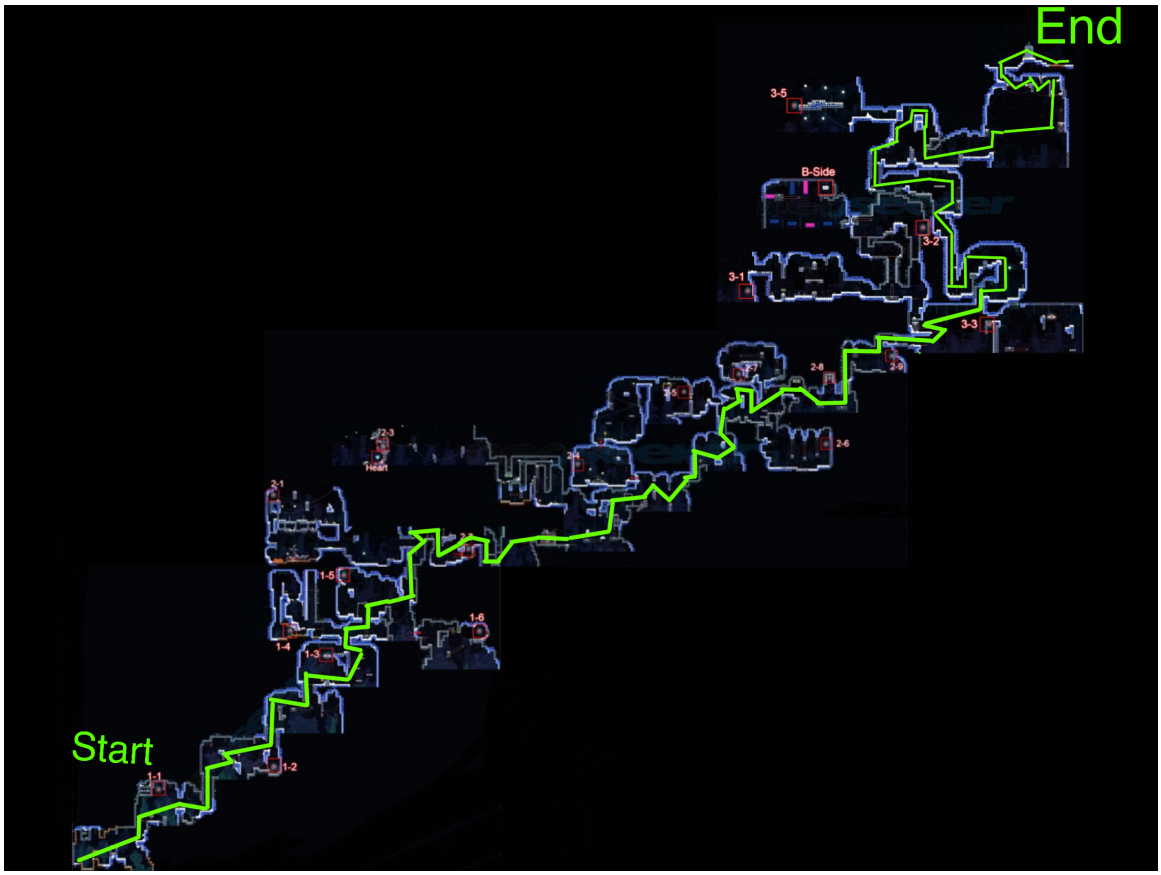
Complexity Classification

Now that the game has been well defined, we work on classification of the game. Classification has been discussed broadly under the Complexity zoo section. Now we adapt one of the methods.

Basic Observation

Given a level and a path, that is the moves required to reach the endpoint, You can verify if the path is correct just by applying those moves.

This clearly implies that the game is NP. For example, for the 1st level, we can map the path from start to end as seen below.



In this pattern we can always map from start to end.

Now since we have proven that Celeste is NP. We can consider it to be Polynomial or NP-Hard which will further make it NP-complete.

Due to many paths of the game which do not lead to the end, and certain mechanisms that lock us out which we will see in the future, We can not immediately tell that there exists a polynomial-time algorithm to solve the levels.

So we will try proving that this game is NP Hard.

NP-Hardness

To prove that Celeste is NP-hard, we have to reduce an already proven NP-hard problem to Celeste. Although there are many options, we will go with the classic 3SAT reduction.

3SAT reduction

Checking if a boolean expression in its 3 Conjunctive normal forms, has a satisfiability is the 3SAT problem. We need to construct a level where reaching the flag would be equivalent to solving a 3SAT problem.

We construct a generalized algorithm to construct a level on Celeste given the boolean expression. For that, we have to construct Frames for the corresponding requirements in 3SAT expression.

Constructing the Frames

3SAT formula is of the form $(x_1 \vee x_2 \vee \neg x_3) \wedge \ldots (x_i \vee x_j \vee x_k)$. We construct different frames for the following parts and then combine them together to obtain the level. The parts which define a 3CNF expression are:

- A Boolean Variable
- OR of variables
- AND of clauses
- Support Frames

What are support frames? Since the combination of the frames will not be straightforward, we need helper frames which will make the connection easier.

The Variable Frame

A Boolean Variable can take 2 values, True or False, and It might have multiple occurrences throughout the formula.

The verification of 3SAT is done by giving the satisfiable values to the variables, hence the values can not be changed in the middle of the

substitution.

For now, we need to take care of the binary and the irreversible nature of boolean variables. We do this with the help of an **Unstable Platform**.

Exits are covered by Unstable platforms making them one way traps.

Frame description

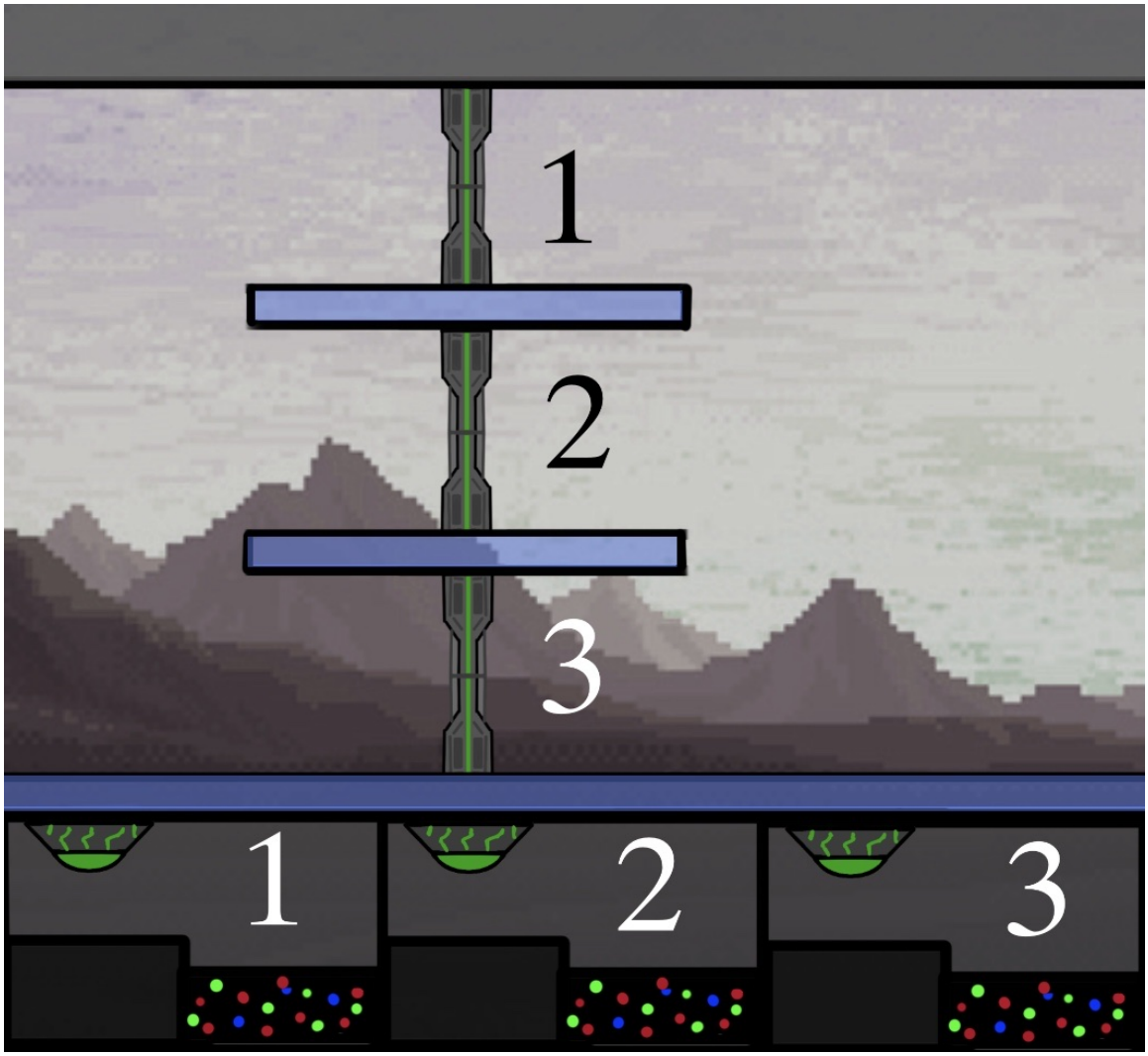
Madeline falls from the top on the platform, that is the only entry to the Frame. The Frame has 2 exits on the sides of the floor, each leading to a tunnel.

These exits have an Unstable platform covering them, these have to be broken before the exit can be used. Why the unstable platform?

The unstable platform makes Madeline seal her choice. Once the path is taken, there is no way to access this frame again other than restarting since the platform will reform blocking the entry.

OR Frame

Each Clause has 3 variables, out of which even if 1 were true the Clause would be true. To implement that in our frame, we use **The Button Door**.



The colorful dots represent space block and the hits are reachable by Madeline.

Frame description

An OR Clause consists of 3 Parallel Button Doors, the buttons are accessed through the variable tunnels. For now, do not worry about how the tunnels are connected.

The main idea is that even if 1 door opens it is sufficient for Madeline to pass through the region, making it an OR gate.

Madeline presses the buttons according to the values she took for the variables, these will unlock the doors, if the variables made a clause true, the clause would have at least 1 door open.

AND Frame

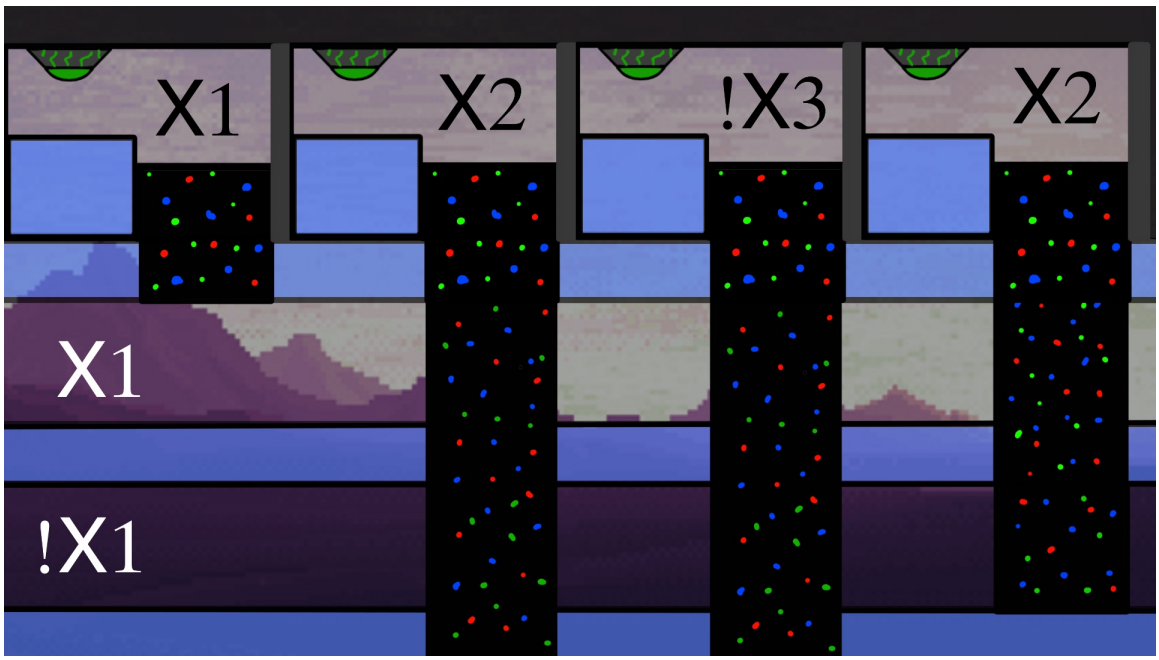
Although there is no separate frame for the AND of clauses, we construct AND of the clauses by Putting the OR frames in a sequence. This will be equivalent to the AND of the clauses since to reach the other side at least one of the door should be open for each of the clauses.

Support Frames

Now the Above frames must be connected, for that, we use our support frames that will be constructed as per the requirements.

The Tunnel

To connect the Variable exit to the Buttons of the OR Clause, we use a Tunnel frame.



x1 Tunnel only has access to button which have x1 as their variable in their clause.

Frame description

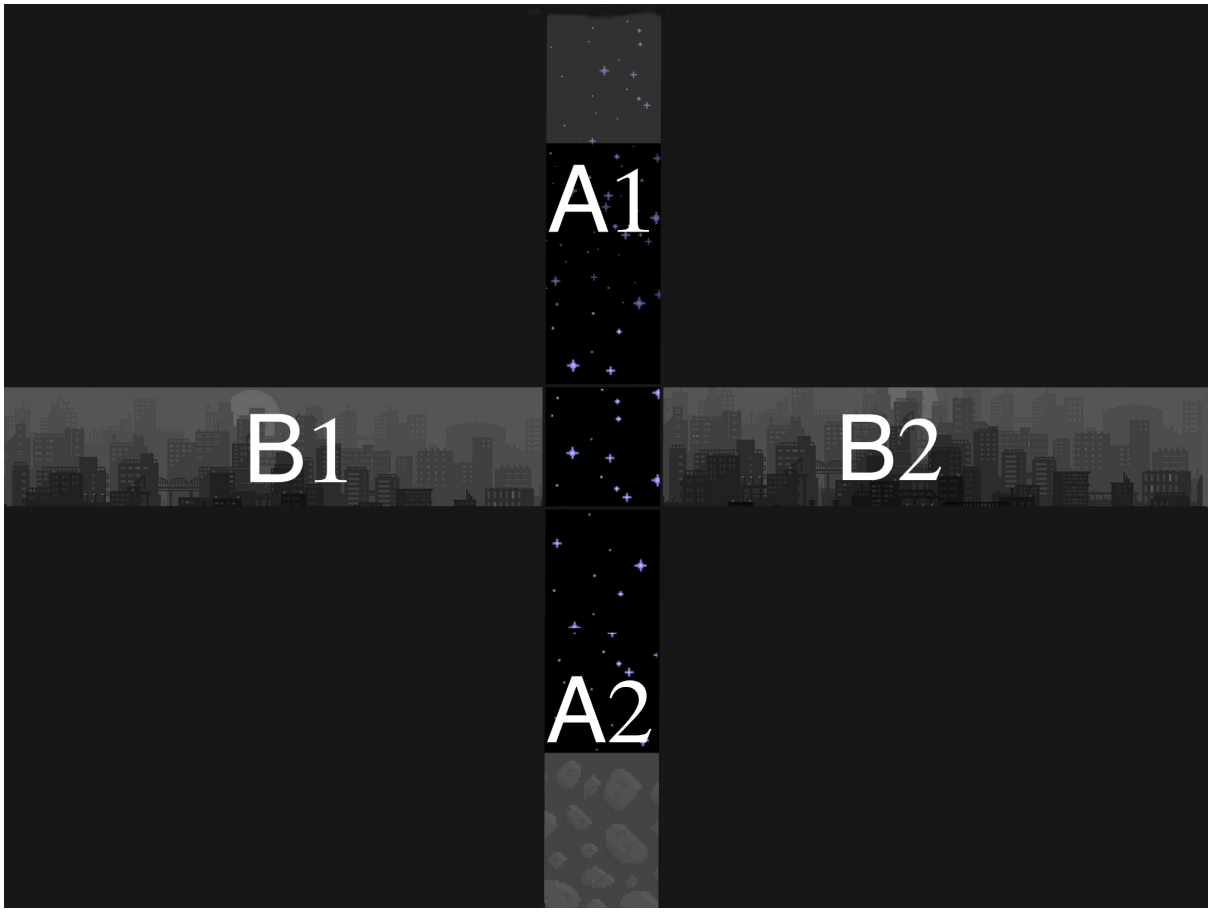
Below the buttons, there are Tunnels leading from the exits of the variables. The variables have access to the buttons they can set true according to the boolean expression.

For example, if x_1 is chosen to be true, then Madeline gets access to the x_1 tunnel and $\neg x_1$if she had chosen false. x_1 tunnel has access to buttons that open a door to the OR clause having x_1.

How do we block the variables from accessing the other doors? For that, we have constructed the **Crossing Frame**. The space blocks that are displayed in the diagram are used in a specific manner described in the Crossing Frame.

Crossing Frame

Since the game is 2D, you can not avoid paths from crossing each other during the construction of such a level. We can make sure that the intersection of the paths happens only in the form of a cross.



Frame description

Suppose we want Madeline to go from A1 to A2 or B1 to B2 or the other direction. But she shouldn't be able to go from an A to a B or vice versa.

The Space block as described before teleports the player from 1 end to the other in a straight line without any interference. Encountering a wall will kill Madeline and she will respawn at the start of the frame.

So we put the space blocks in the intersection of the paths in such a way that there are no straight lines connecting the opening side of A to B.

Note: It might seem like you can draw a line from A to B but remember that Madeline can only move in 8 directions, so the lines can be parallel or 45 degrees inclined with the axis. So no such line will exist.

This means that the only way she can travel through the space block is in a straight line parallel to the axis, hence she can not access A from B or vice versa.

Sequence of Frames

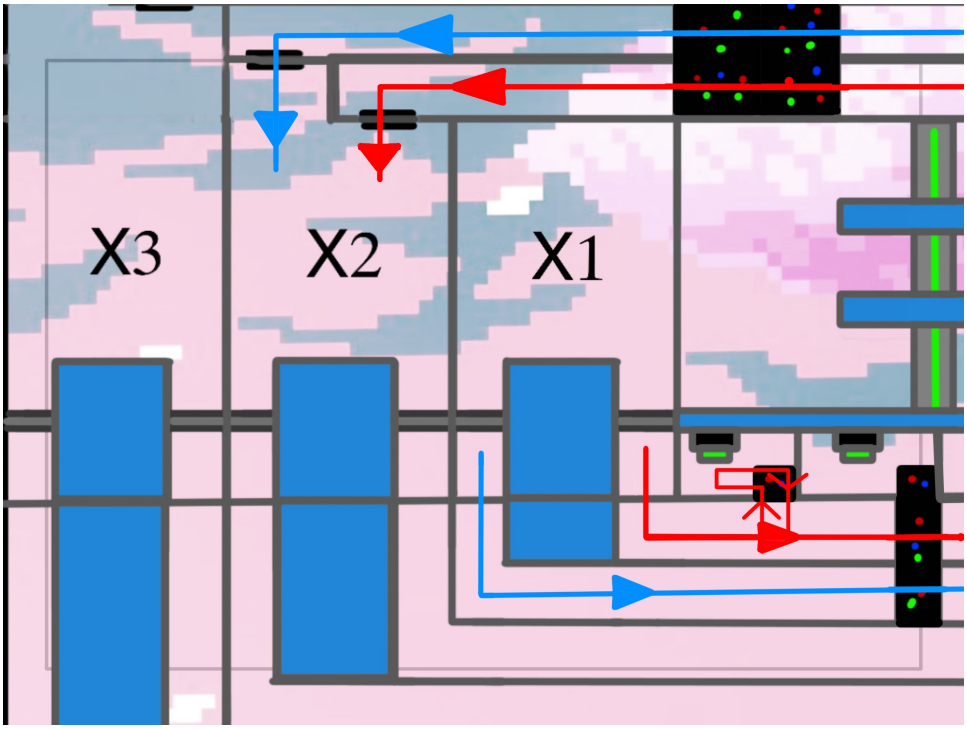
Now that all the frames have been constructed, we decide the sequence of the frames.

Let `n` be the number of variables in the Boolean expression distributed into `k` clauses.

Variable order

We will assume the order of the variables to be $x_1, x_2, x_3 \dots x_n$. We select values for these variables in the same order. So the starting position will be in the x_1 frame since we pick its value first.

Transition between variables

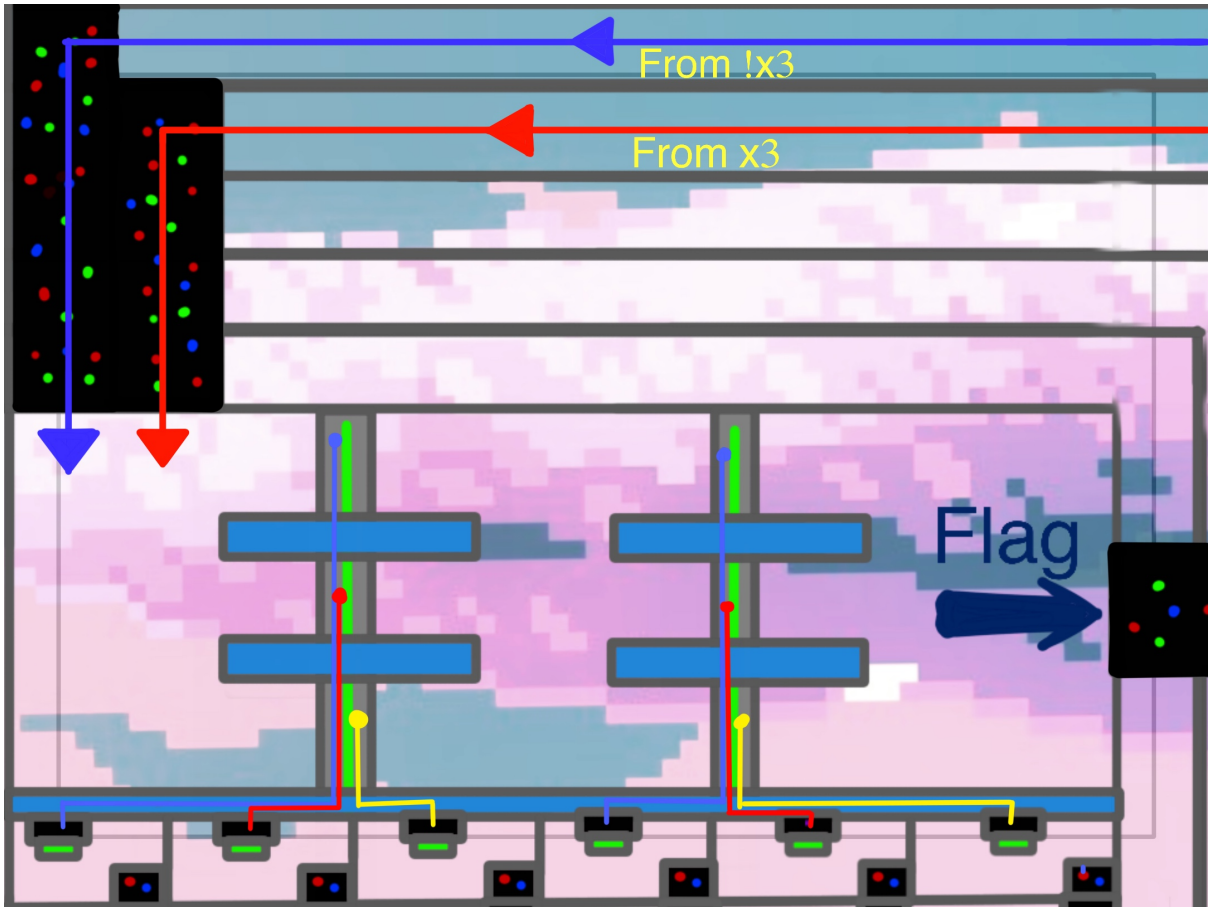


Transition from x1 to x2 after completing the path.have

From the variable frame of x_1, we choose its value and go to the respective tunnel. In the tunnel, we press all the accessible buttons, After all the buttons, we reach the end of the tunnel. Now since the values of x_1 have been already picked and substituted, we have to pick a value for the next variable hence we must go to the x_2 variable frame.

In such order we pick the value of all the variables, click the buttons which open the OR clause doors, and continue until we reach the end of the x_n variable. Since we ran out of variables, where do we go now?

Final passage



Example case where n = 3

At the end of the x_n passage, we should have an entrance to the Final passage containing the OR clauses. Madeline after choosing all the variables will now have to reach the flag.

The path to the flag lies on the other side of the passage. If at least one door from each Clause is open only then will she be able to reach to the flag, else she won't be able to complete the level.

This was the AND of all the OR clauses, leading to a normal form with 3 variables in each clause.

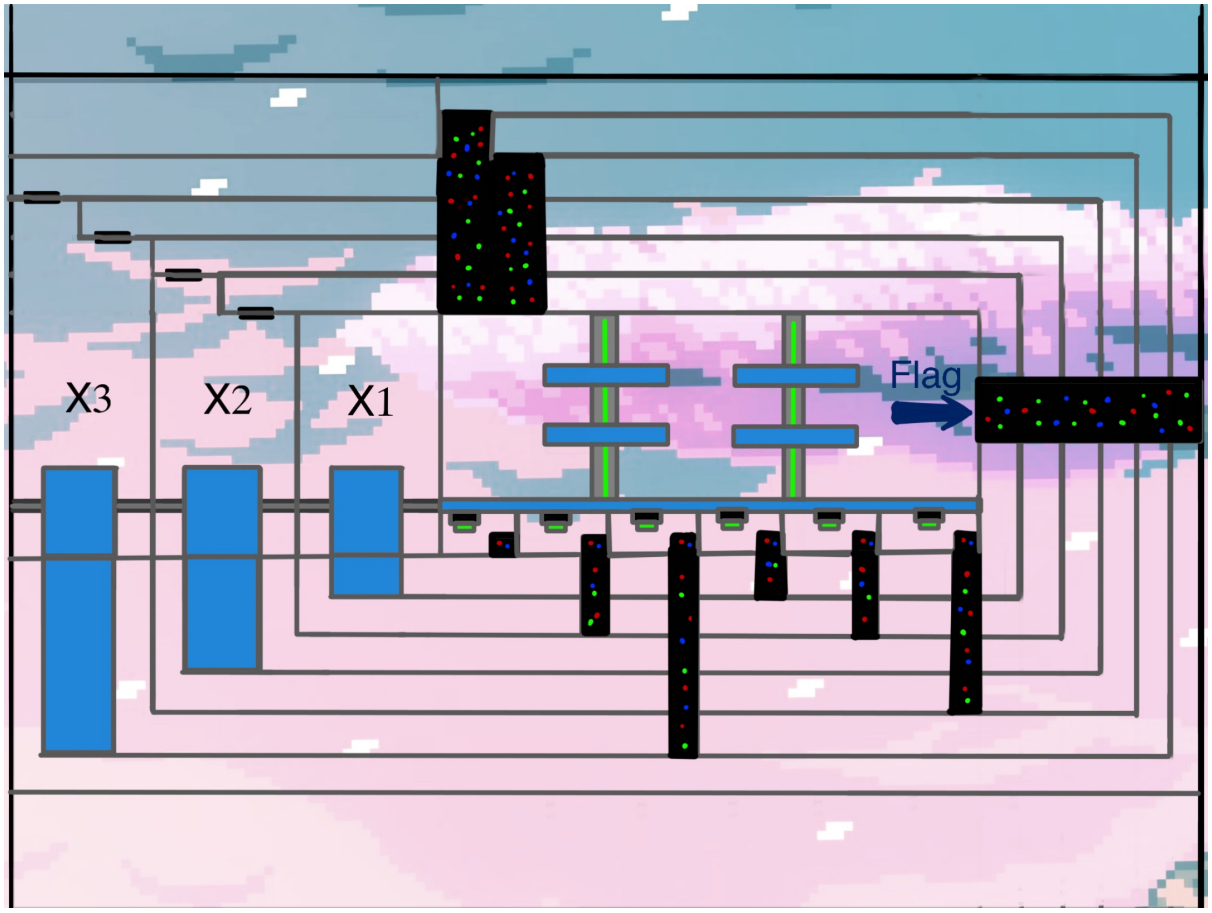
Why not put the flag at the end?

The flag is always at the top of the level. So we need to redirect the player from the end of the final passage to the flag. Since there are other Paths that come between it, we use crossing frames.

Final Level

Now that all the separate parts have been explained, we put together our final level. The Boolean expression for which the level has been implemented is:

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$$

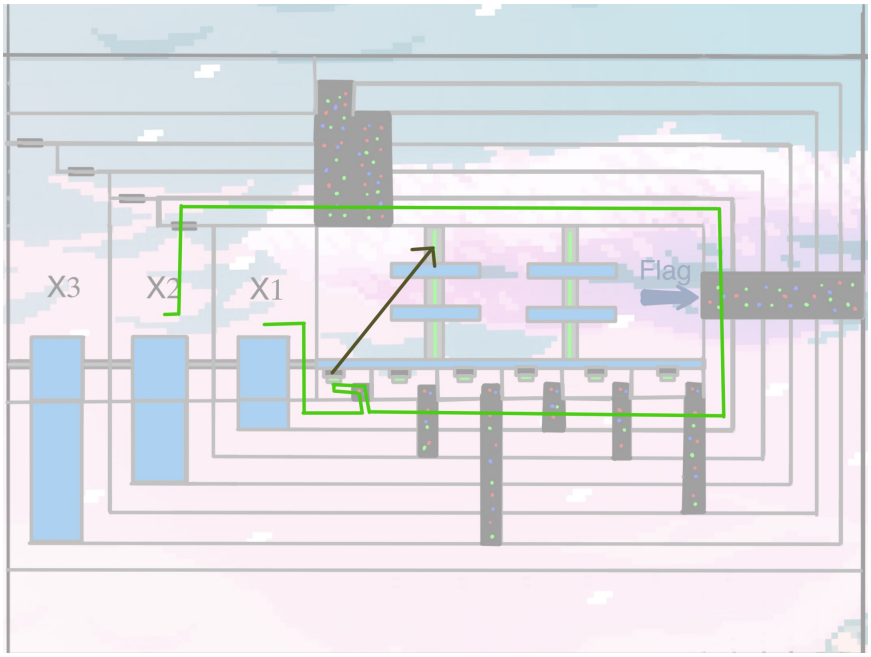


Example Subsitution

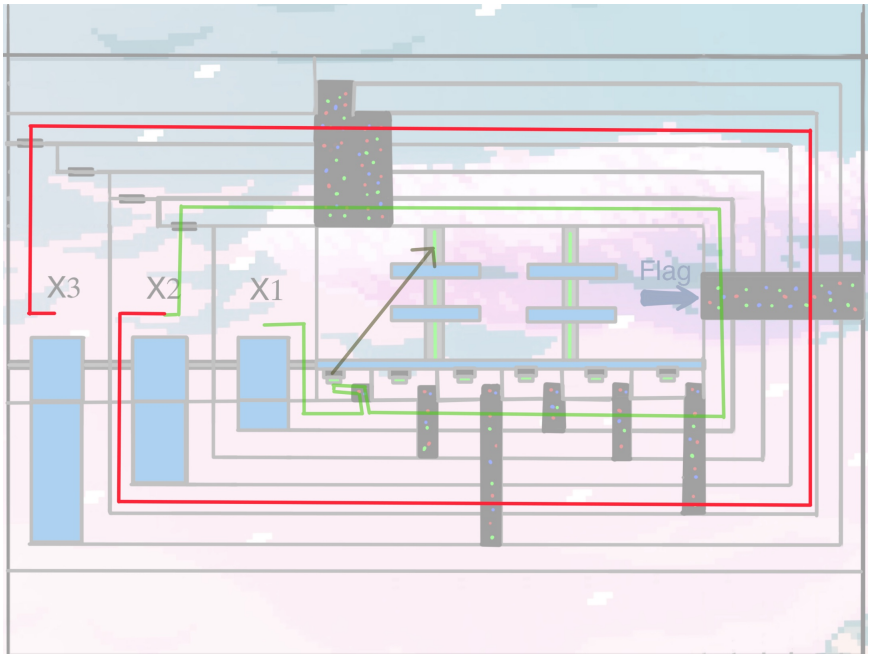
Let the substitution of the variables be:

- $x_1 = 1$
- $x_2 = 0$
- $x_3 = 1$

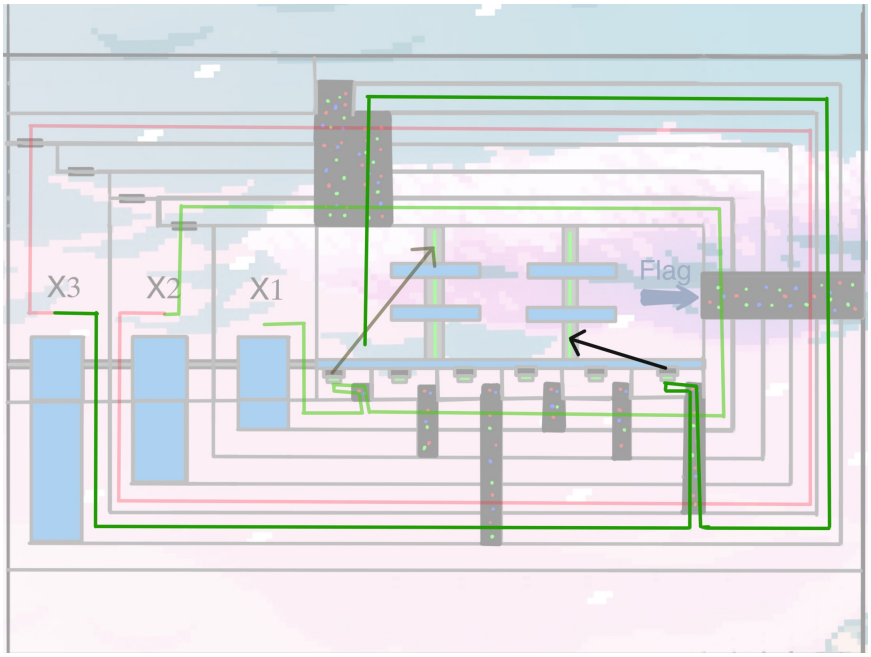
We start with the x_1 frame, take the true tunnel, and activate the door. After which we end up with a tunnel with an exit leading to x_2 frame.



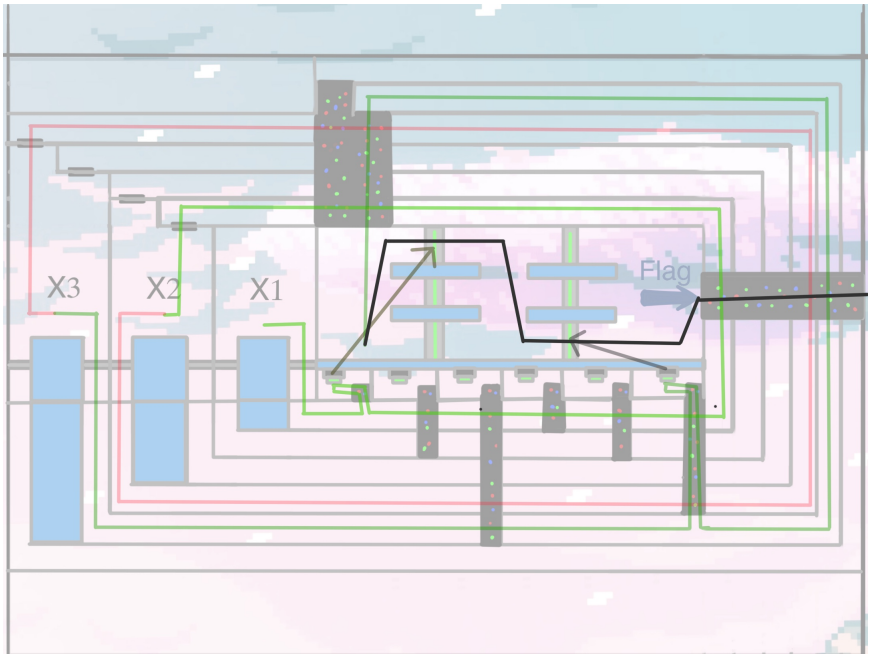
Now we are in the x_2 frame, we take the false tunnel, but since there is no x_2 in the expression, there is no door that can be opened from this tunnel. So we just continue and end up in the x_3 frame.



In the x_3 frame, we repeat the same procedure. We open a door in the 2nd OR clause and end up at the end of the tunnel which leads to a space block. This space block when used will take Madeline to the beginning of the Final passage.



Now one door of each clause has opened, Madeline can pass the final passage and make it to the flag.



Since Madeline was able to reach the flag. The level could be completed, hence the expression as expected is satisfied with the given values.

Conclusion

This proves that Celeste is at least as hard as 3SAT, making it NP-Hard. In conclusion, the game is both NP and NP-Hard, making it an NP-Complete puzzle.