Reductions

Zeeshan Ahmed

3SAT ∈ NP

$x_1, \ldots x_n$

x_i = T or F [guess in constant time] $\forall x_i$

Non deterministic machines → biased towards True, so even if 1 setting where x_i is true satisfies the overall condition, it will return true.

→ All the x_i are found in such order & then at the end we can check by substituting them into the equation if the overall expression is satisfied.

Polynomial time solution (in Non deterministic machine)

Guesses ⇒ Witness / certificate

NP hard ⇒ Problems as hard as any problem in NP

NP complete ⇒ NP hard ∩ NP

You can reduce all the problems of NP to a problem in NP hard

↳ misleading name

Reduction ⇒ You reduce problem A to problem B if you can convert the input of A to equivalent input to problem B.

- If A → B then B is as hard as A.

X ∈ NP complete

→ X ∈ NP
→ X ∈ NP hard ⇒ by reducing all the problems in NP to X

[OR you can reduce already known NP complete to X]

A ∈ NP B ∈ NP complete X ⇒ Given problem
If B → X then ∀ A → X since ∀ A → B

Super Mario Brothers reduction

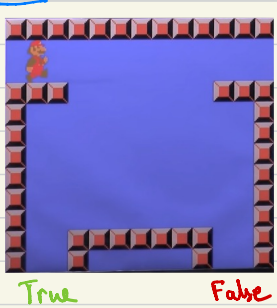
To prove that the game is NP hard, we need to reduce an NP complete problem to Mario. So we choose 3SAT.

Generalizations:

→ No time limit
→ The whole game is in 1 screen

Piecewise construction

Variable:

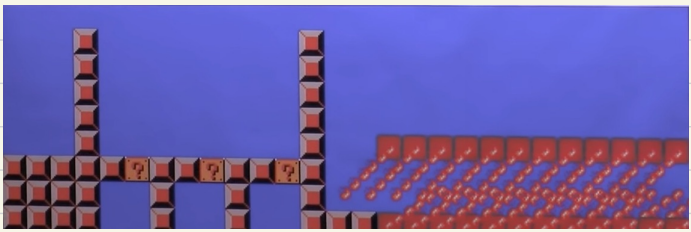


• n clauses for n variables

True exit leads to the clauses with X in them & False exit leads to $\bar{X}$ clauses

• After visiting all the clauses you get back to the next variable entry.

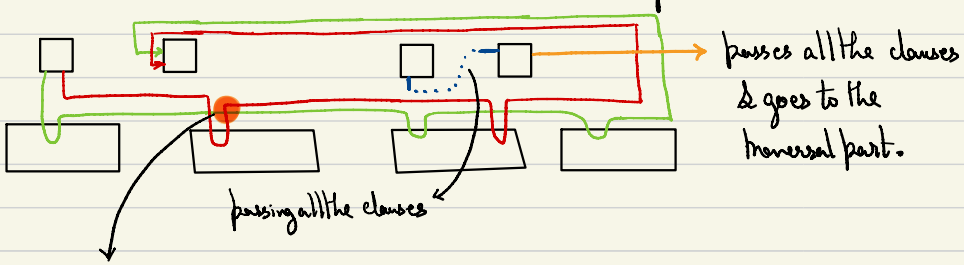
Clauses : • Needs to activate At least one of the stars (OR) & having more than 1 star gives nothing.



$X_1 \quad X_2 \quad X_3 \Rightarrow (X_1 \vee X_2 \vee X_3)$

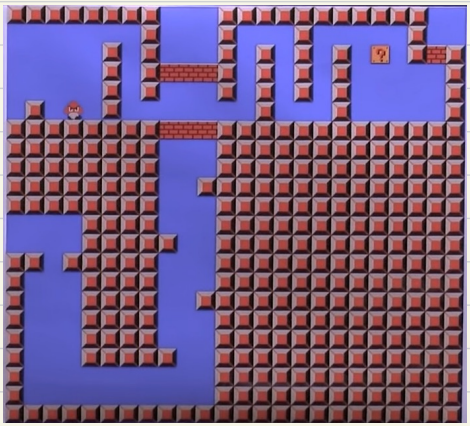
Traversal : After marking all the clauses it goes through all of them from the top.

The last variables Exit is connected to this traversal part.



To make a realistic level you need to deal with the cuts, for this we use another setup called Cross

Issues: Does not allow the crossing path to go through in other directions.



Unidirectional pass to prevent 1 path to lead to multiple openings.

Allowed Reductions

$A \xrightarrow[f(x)]{} B \xrightarrow[g(x)]{} C$ size of $f(x)$ can be at most $O(T(n))$ now

$T(n)$ $T'(m)$

So the complexity of overall conversion will be $T'(T(n))$

3SAT is NP complete $\Rightarrow$ Cook Levin Theorem

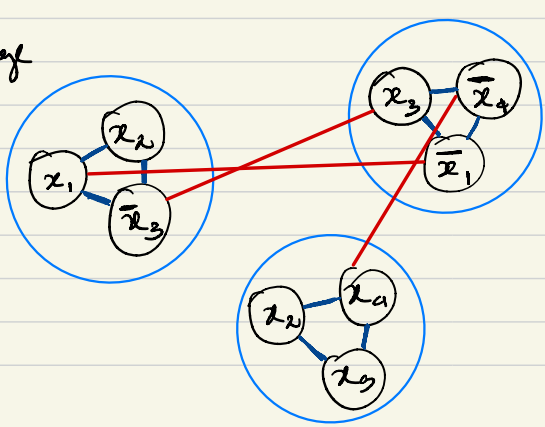
Independent set

- Given a graph, can we pick k vertices such that no 2 vertices have an edge b/w them.

- To prove that is NP hard we reduce 3SAT to K clique.

Given an expression: $(x_1 \vee x_2 \vee \bar{x}_3) \wedge (x_3 \vee \bar{x}_4 \vee \bar{x}_1) \wedge (x_2 \vee x_4 \vee x_3)$

Each variable: A vertex
Each clause: A clique
- Joining x & $\neg x$ by an edge



- We just have to pick one from each clique, if there are repetitions then it is fine.

- Picking 1 x_i does not mean the other x_i is already picked. But if any 1 of them is picked it is sufficient for it to become 1.

NPcomplete proof

- We need a witness & a verification algorithm to check if the graph has a k sized independent set.

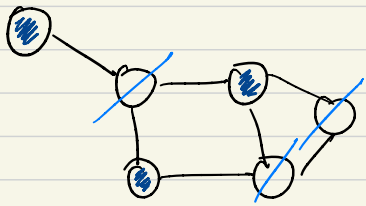
Witness: Set of vertices [k] which form the independent set.

Verification:
- Pick a vertex from the set & remove all its neighbours from the vertex.
- Repeat until the set is exhausted, if a vertex which is present in the set gets eliminated then its invalid.

G has k sized independent set $\Leftrightarrow$ $\bar{G}$ has n-k sized clique

Vertex Cover

- Set of vertices which contain atleast 1 end of each edge.
- Finding the minimum vertex cover is the NP hard problem.



- Given graph, does it have vertex cover of size k.

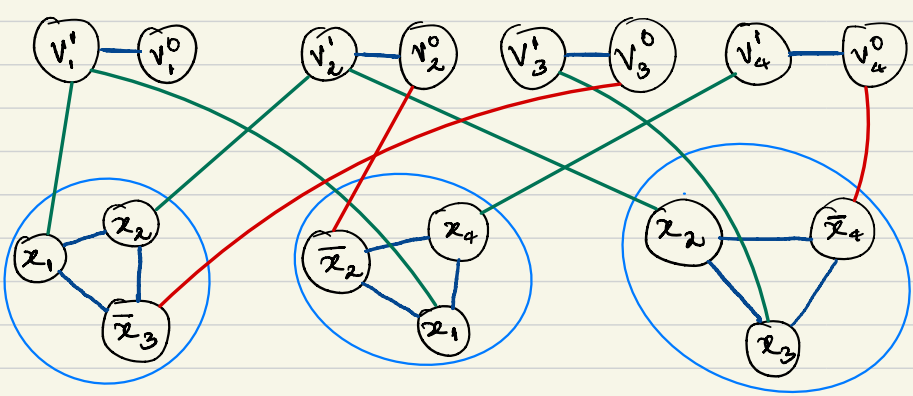
- Independent set → Vertex cover of k has n−k sized independent set.

From 3SAT

v_i^0 & v_i^1 are 2 vertices representing variable x_i, they have an edge between them.

A 3 clique represents a clause.

$$(x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_2 \vee x_4 \vee x_1) \wedge (x_2 \vee \bar{x}_4 \vee x_3)$$



- Joining x_i with v_i^1
- Joining $\bar{x}_i$ with v_i^0

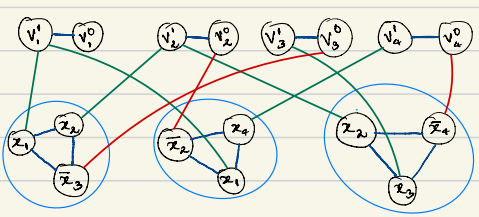
Conditions to form a vertex cover

- v_i^0 or v_i^1 has to be picked since there is an edge between them.
- If you pick v_i^0 then $\bar{x}_i$ has to be picked since there is an edge b/w v_i^1 & $\bar{x}_i$ which will be uncovered.
- If $x_i = 1$ pick v_i^1 & other vertices in the clause which has x_i.

Number of vertices picked = $2 \cdot$ clauses $+ 1 \cdot$ variables
(x_i) (v_i)

Solving 3SAT will solve min vertex cover of $k = 2m+n$ for that graph!

For the same graph, if we have a vertex cover of size $2m+n$ then we solve 3SAT.



- Edge b/w v_i^1 & v_i^0 is present, 1 of them has to be picked. Eliminates n vertices
- Among the 3 vertices in a clause clique we have to pick at least 2. Eliminates the rest.

Cook levin theorem

Checking the satisfiability of a boolean expression is NP complete.

SAT is NP

- Given an assignment we can verify if the assignment satisfies it or not.

SAT is NP hard

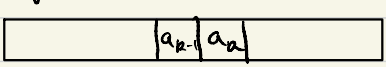
- We need to reduce all NP problems to 3 SAT to prove it.
- Any SAT can be reduced to 3SAT with boolean algebra.

Let A be some NP problem. A has a polynomial time verifier

$\exists c \; V(x,c) = 1 \quad x \in A \qquad \forall c \; V(x,c) = 0 \quad x \notin A$

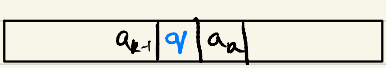
↳ polynomial in $|x|$

Every TM has a configuration, it is the union of states & the Tape



state → q $\qquad C = \Gamma \cup Q$

This can be written as



→ This contains the entire configuration of the given TM.

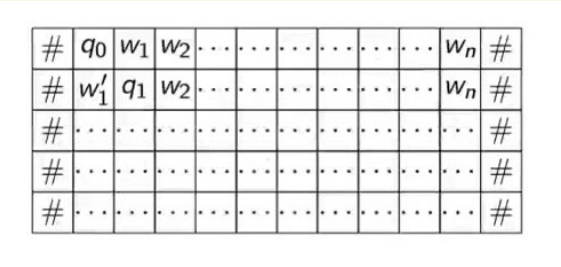
$m = T(n) \rightarrow$ running time [number of steps]

- After each step there can be a different configuration, all of them belong to C

So complete configuration $\in C^m$

This can also be proven that $T(n) > S(n)$ is a loop since we come back to pre existing state & loop.

Tableau → Sequence of configuration

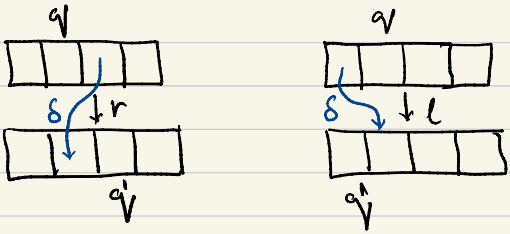


No loop → $S(n) \leq T(n)$

Validity of a Tableau → Sequence should be obtained from a valid transition table (δ)

$q_1' w_0 w_1' w_2 \xleftarrow{l} w_0 q_0 w_1 w_2 \xrightarrow{r} w_0 w_1' q_1 w_2$

by default the block after the state is the header



In general, it can be seen that the block depends upon the 3 blocks above it in the previous configuration.

Variables in SAT

There are $|C|$ variables for each position on the tableau

$$x_{i,j,s} = T \quad \text{if} \quad \text{Tableau}_{i,j} = s$$

Encoding all the necessary conditions for the tableau based on these variables.

Conditions:

- Starts at q_{start} at the beginning of the tape
- Each transition corresponds to the transition table
- Reaches q_{accept} somewhere.

$A \longrightarrow SAT$

Converting conditions to formula

$\forall\, i,j \qquad x_{i,j,s} = T \quad \exists\, s \in C$ [only 1 should satisfy]

$$\phi_{i,j} = \bigvee_{s \in C} \left(x_{i,j,s} \bigwedge_{\substack{s' \in C \\ s' \neq s}} \neg x_{i,j,s'} \right) \qquad \text{size} \rightarrow O(|C|^2)$$

That was for a single cell. we have it for all the cells

$$\bigwedge_{i,j \in T} \phi_{i,j} \longrightarrow \text{condition for variable to be valid}$$

→ Initial configuration

q_{start} → initial state $\qquad$ [input string] $\qquad w \rightarrow \langle w_0, w_1, \ldots w_n \rangle$

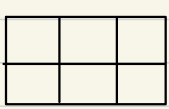
q_{start}	w_0	....	w_n

→ The required start

$$\phi_{start} = x_{0,0,q_s} \wedge x_{0,1,w_0} \cdots x_{0,n+1,w_n}$$

→ Valid transitions

- We check 3x2 windows → There are $(S(n)-2)(T(n)-1)$ 3x2 windows



Total possibilities → $|C|^6$

Valid windows based on the δ function $\subseteq C^6$

We check if every 2x3 window is valid.

General window → $\begin{matrix} x_{i,j} & x_{i+1,j} & x_{i+2,j} \\ x_{i,j+1} & x_{i+1,j+1} & x_{i+2,j+1} \end{matrix}$ → B_{ij}

Now suppose we have some valid window W_1 → $\begin{matrix} a & b & c \\ d & e & f \end{matrix}$

Suppose there are K valid windows

To check if B_{ij} matches W_1 we do

B_{ij1} → $x_{ij,a} \wedge x_{i+1,jb} \dots \wedge x_{i+2,j+1,f}$

Validity of B_{ij} → $\bigvee_{R=0}^{K} B_{ijR}$

Now we have to check validity of all the windows in the tableau

$$\bigwedge_{i,j} (B_{ij})$$

→ Accepting configuration

ϕ_{accept} → $\bigvee_{i,j} x_{i,j,q_{acc}}$ At least 1 state has to be q_{acc}.

→ Combining the constraints

$$\phi_{A,w} = \phi_{cell} \wedge \phi_{start} \wedge \phi_{window} \wedge \phi_{accept}$$

→ Polynomial time reduction

checking at all i, j, so the complexity will be $O(T(n)S(n))$

→ SAT to CNF3SAT

$$(l_1 \vee l_2 \vee l_3 \ldots l_n) \longrightarrow (l_1 \vee l_2 \vee z) \wedge (\bar{z} \vee l_3 \vee \ldots l_n)$$

→ perform it recursively

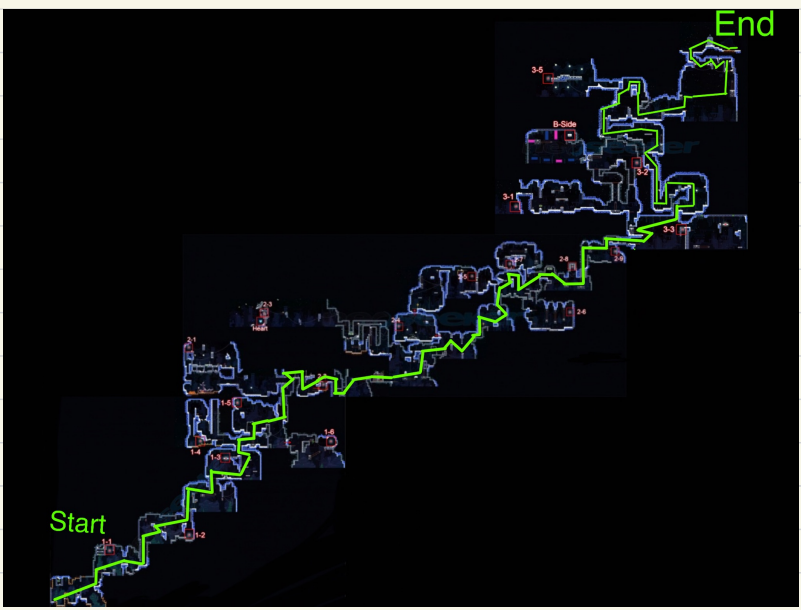
This is polynomial time, steps taken → $\frac{n}{2}$

Celeste

NP

Given a level, We can create a witness, that is the path leading from the start to the end.

The first level of the game as an example:



NP-complete

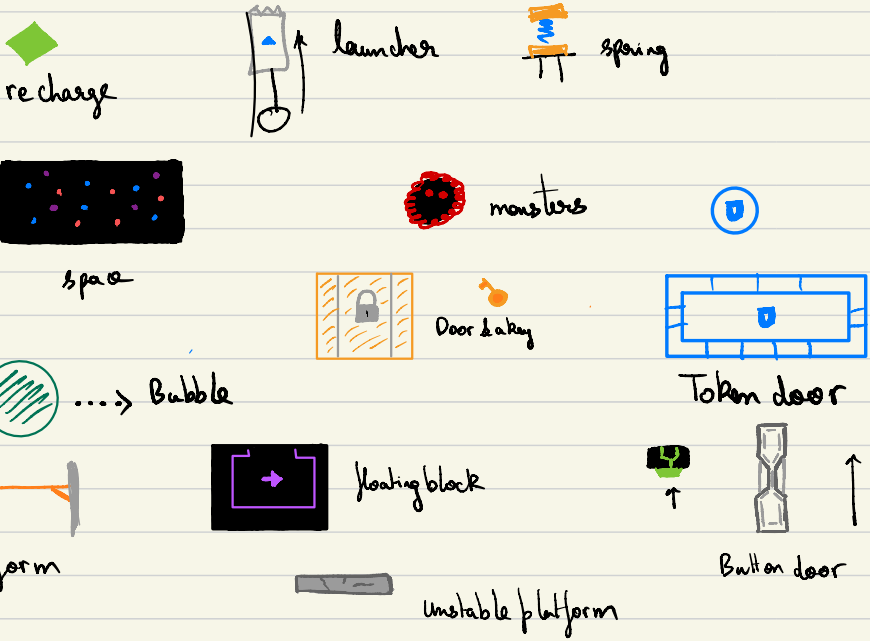
→ Reducing from 3SAT

Level has sequence of frames which are connected through tunnels.

Types of frames : • Frame for variable • Frame for clause

• Connecting frames

Tools to use



[more not included]

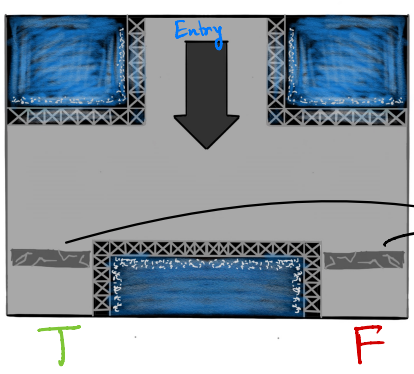
Frames

Variable frame

- It has an entry point which will be connected to a tunnel
- 2 1 way exits (traps) which represent the value set for the variable. (1 & 0).

Tools used → Unstable platform

Unstable platform → breaks when stood upon. Can't be broken from the bottom. making it 1 way.



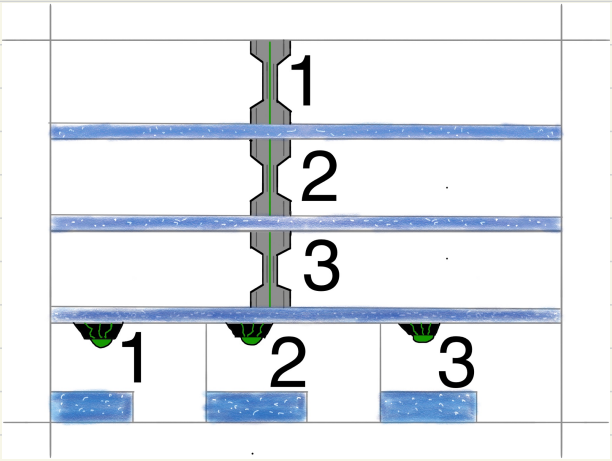
Unstable platform

Why unstable platform?

Madeline has wall grab so we need a trap to prevent her from climbing back since it reforms.

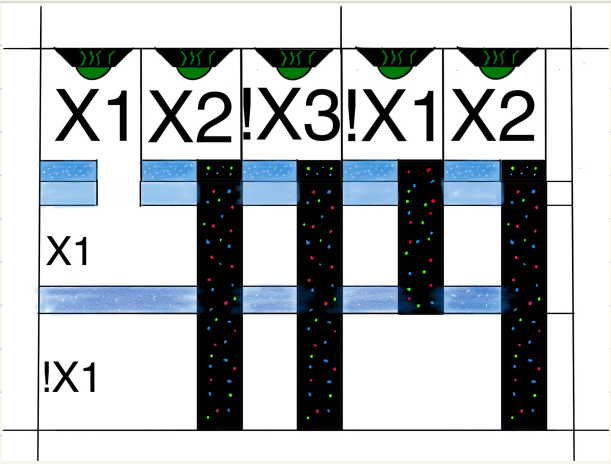
Clause Zone

- In total has 3 entries, each representing a variable. Has 3 separate chambers



- Refer full map to understand why opening one of the doors is sufficient.

Tunnel connection



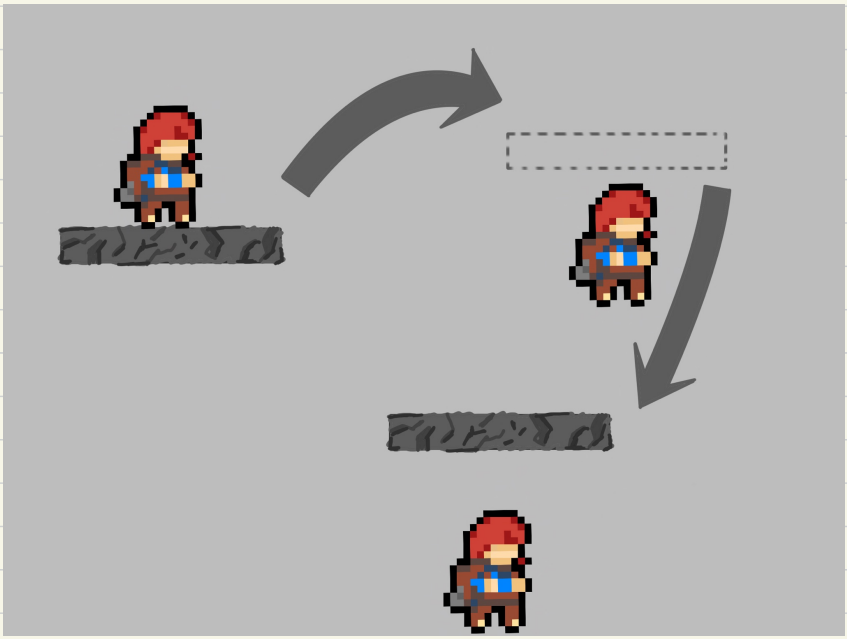
There are paths running below the buttons.

Each path is connected to the variable zone exits.

Creating 1 way trap

- Madeline has the ability to climb walls [till a certain extent], also she has the ability to dash in to the air.
- So to prevent the player from backtracking, we need to make traps.
- Unstable platform: It is a stone platform which breaks when stood upon for more than a second.

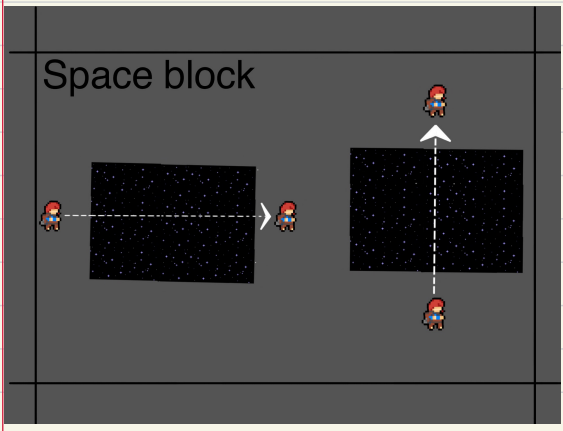
After Madeline falls through the platform reforms. and this platform can not be broken from the bottom.



Crossing frame

Since Celeste is a 2D game, paths from different frames can end up crossing each other.

We need to allow the player to cross the paths without the access of other paths. We use the space block.



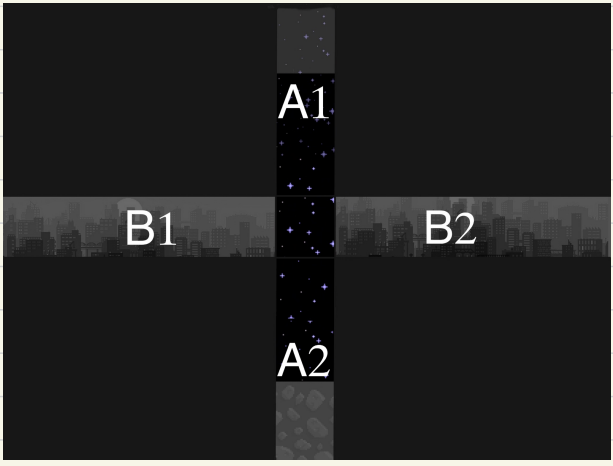
The space block moves Madeline from 1 end to the other in a straight line.

The player has no control when they are getting moved.

If a wall is encountered player dies.

The cross frame allows the player to travel from A1 to A2 or B1 to B2.

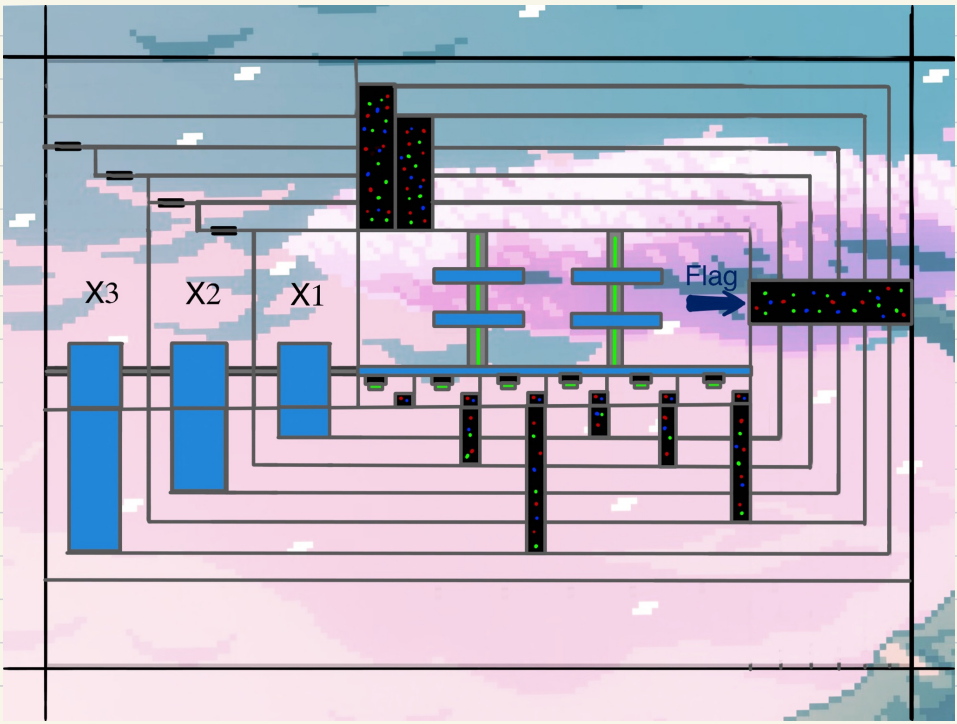
But due to the thickness of the tunnels, player can't access other tunnel.



Sequence of frames

$x_1 \rightarrow$ tunnel $\rightarrow x_2 \rightarrow \ldots$ tunnel $\rightarrow x_n \rightarrow$ tunnel $\rightarrow$

Start of the clause frame sequence $\rightarrow$ End flag



Start position : X1

Winning criteria : Needs to reach the flag.