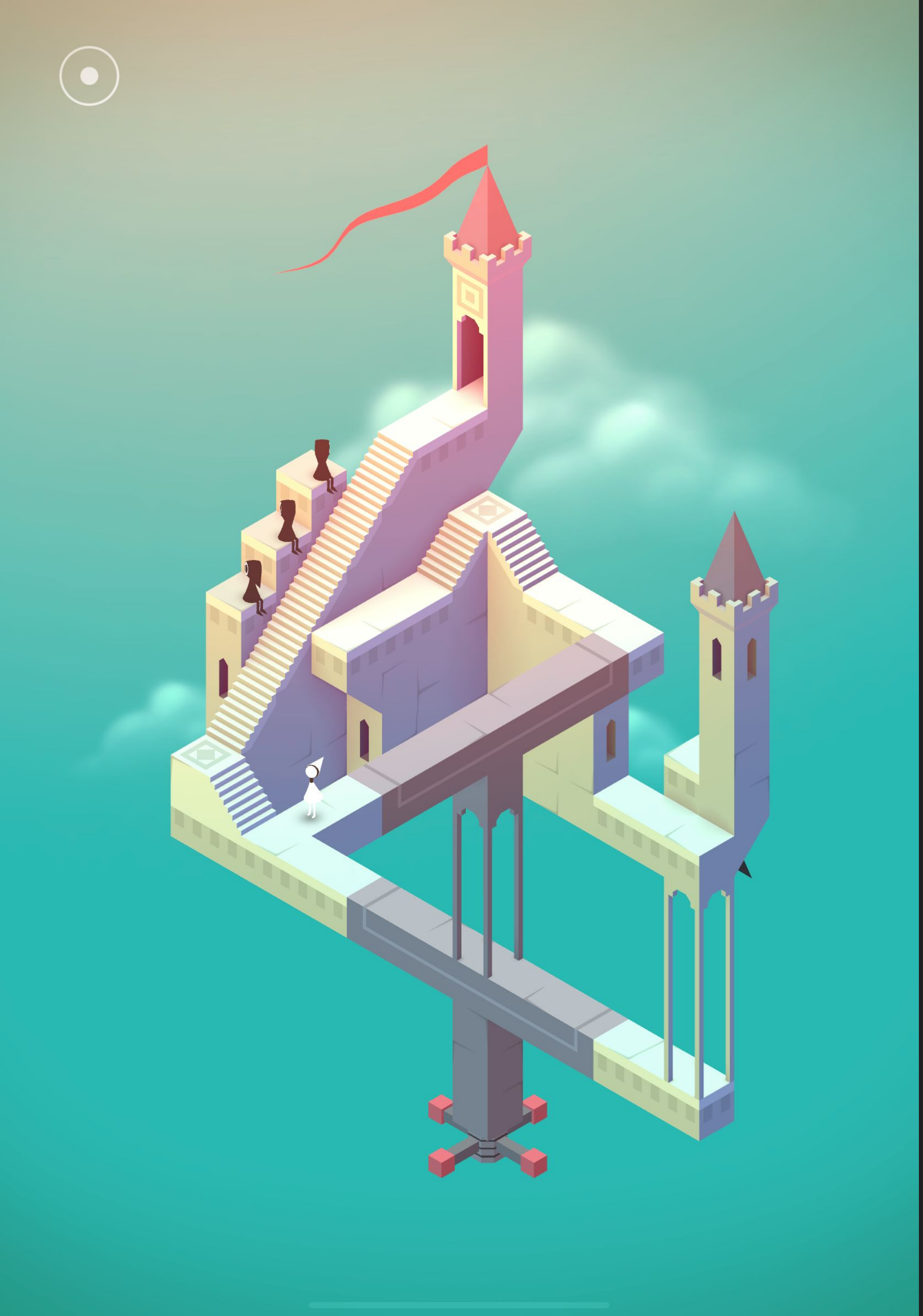
MONUMENT VALLEY IS PSPACE-COMPLETE

Vy Nguyen and Catherine Yeh



ABSTRACT

This project studies the computational complexity of **Monument Valley**, focusing on cases when the game is restricted to certain types of mechanisms. In particular, we show that Monument Valley puzzles restricted to reusable buttons and rotating blocks are **PSPACE-Complete**. Our complexity analysis is done through constructing motion planning gadgets as described by Demaine et al. in [1, 2].

INTRODUCTION

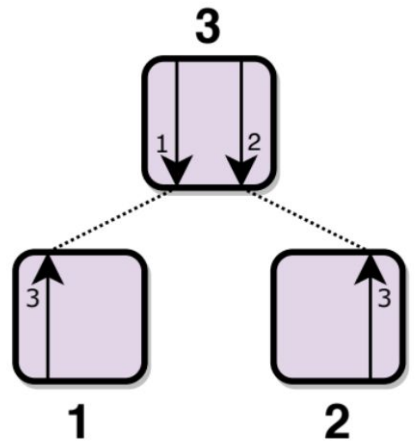
Monument Valley is a 3D puzzle game developed by Ustwo Games in 2014. The player leads Ida through a series of optical illusions and impossible mazes on her journey of discovery and forgiveness.

There are currently two iterations of Monument Valley:

- Monument Valley I
- Monument Valley II

PSPACE-COMPLETENESS

The **motion planning** framework is based on the idea of using “gadgets” in complexity proofs [1]. These gadgets usually consist of multiple states that change as the robot or player agent moves from one location to another.



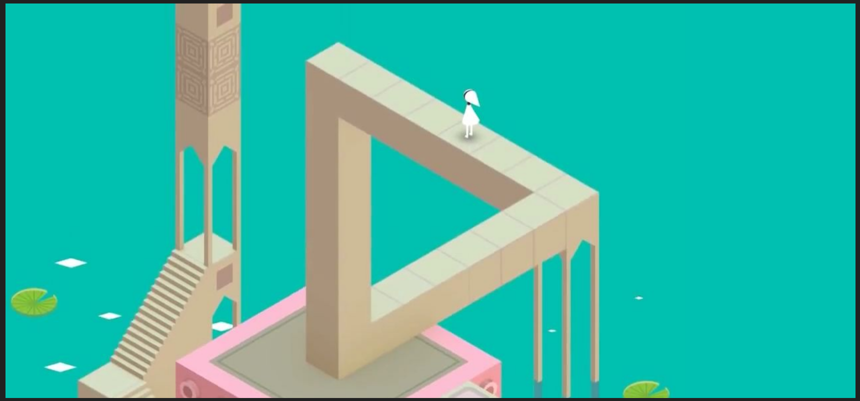
(a) The locking 2-toggle gadget. In the top state 3, you can traverse either tunnel going down, which blocks off the other tunnel until you reverse the initial traversal.

Demaine et al. [2] proved that any 1-player game that can be modeled with at least one *reversible, deterministic* gadget that has 2+ *interacting* tunnels (and the ability to walk between gadgets) is PSPACE-Complete. One class of such gadgets is the **locking 2-toggle** (see diagram on the left).

In this project, we use motion planning gadgets to prove that Monument Valley is PSPACE-Complete. To our knowledge, there isn’t any existing complexity research about this game.

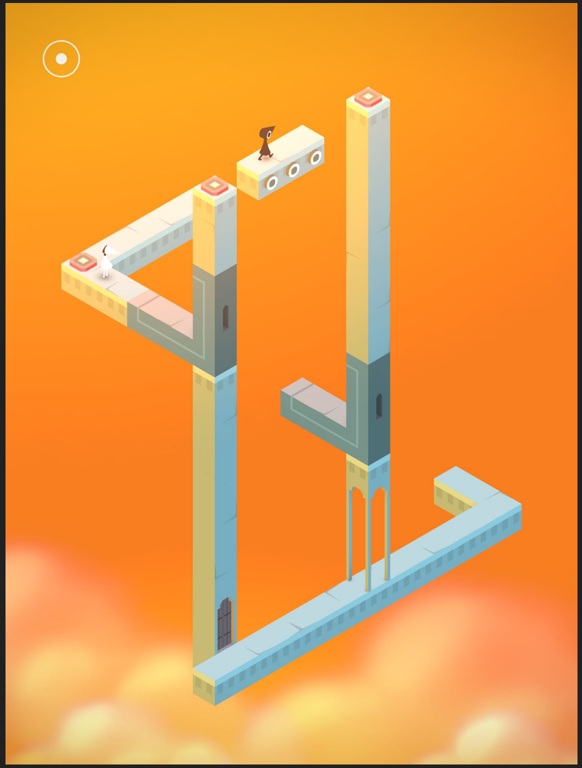
GAME MECHANISMS

We use two main mechanisms from the game to design our motion planning gadget: buttons and rotating blocks.



Buttons

Buttons are used in various ways to change the state of the game (i.e. rotate part of the path, activate hidden parts of the level, etc.). Some buttons can only be used once, while others are reusable. We will focus on reusable buttons as they allow us to switch between / go back to previous states.



Rotating Blocks

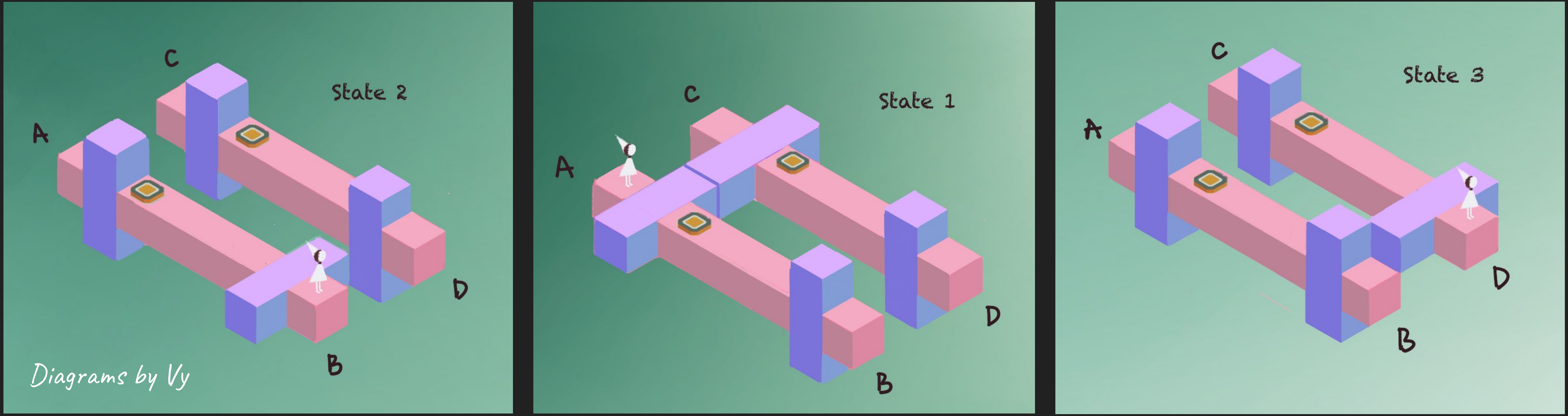
Rotating blocks appear in many forms throughout Monument Valley and are used to alter Ida's path. For example, they often reveal hidden stairs / ladders or alter the perspective of the game. These blocks are are usually controlled by another game mechanism, like a rotating dial or button.

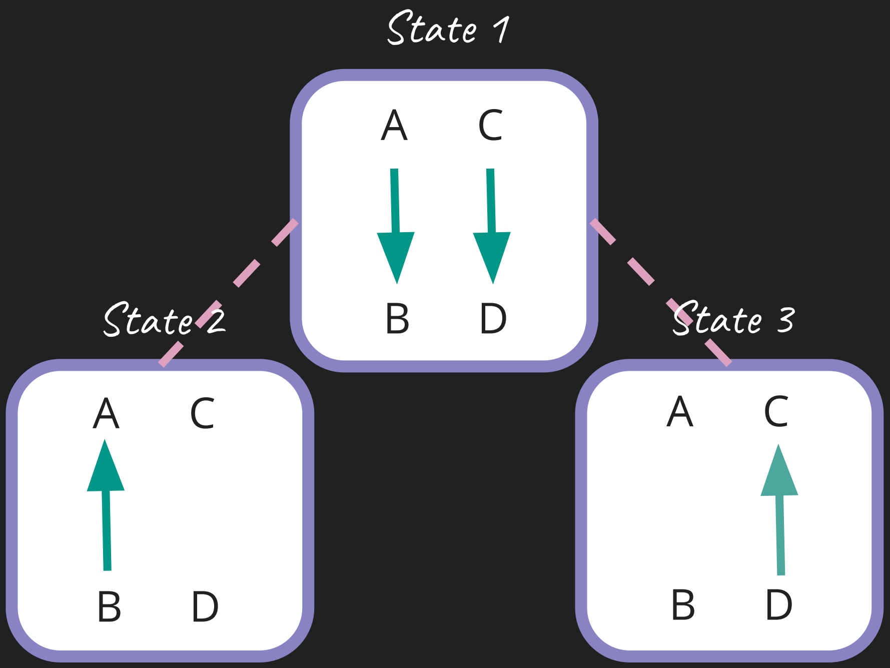
GADGET DESIGN

Our gadget (on next slide) is an example of the **locking 2-toggle**. There are three states and four locations (A, B, C, and D) that Ida can move between using two interacting, parallel tunnels.

- When Ida is in *State 1*, she can either walk from A → B or C → D.
- If Ida moves to B from State 1, the gadget changes to *State 2*, and the C → D tunnel becomes blocked. Ida can only walk from B → A in State 2.
- Similarly, if Ida moves to D while in State 1, the gadget changes to *State 3*, and the A → B tunnel becomes blocked. Ida can only walk from D → C in State 3.

We create the locking effect using **reusable buttons** and **rotating blocks**.





OUR GADGET (LOCKING 2-TOGGLE)

Originally (in state 1), Ida can choose to go from A → B or C → D. B and D are blocked by the 2-tall rotating "wall" blocks. A and C have reusable buttons that rotate the blocks 45 degrees when they are stepped on.

A → B traversal (shown above):

1) Ida enters at A. When she presses the button, the "wall" blocks at A, B and C rotate. Now A and C are blocked (in addition to D), while B becomes unblocked.
2) Now Ida can go from A → B (changing the gadget to state 2). C and D are blocked, so the only thing Ida can do is to walk in the reverse direction from B → A (and essentially reset the game state to state 1).

Similarly, if Ida traverses C → D, this changes the gadget to state 3. And D → C resets the game state to state 2. When the button at C is pressed, the blocks at A, C, and D rotate.

OUR GADGET (CONTINUED)

We have created a gadget that is:

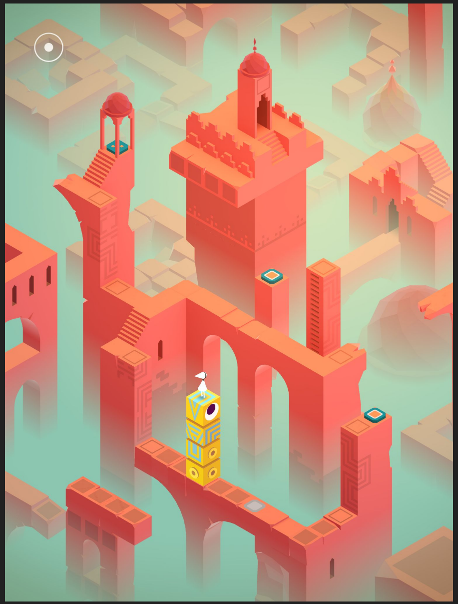
1) Deterministic: Every time Ida traverses a path at a particular state, she will end up at the same final state. If she starts at A at *state 1* and traverse to B, then Ida will get to *state 2*, and vice versa. The same logic applies for C/D traversal in *state 1* and *state 3*.

2) Reversible: A/B traversal in either direction requires Ida to press the reusable button at A. A → B causes the 3 wall blocks at A, B, and C to rotate 90 degrees. Traversing back B → A will cause the 3 blocks to rotate again, thus resetting the state to before, and with Ida back at where she was. The same apply to C/D traversal, but with blocks A, C, and D.

3) Interacting k-tunnel: When Ida traverses A → B and steps on the button at A, the wall block at C rotates, thus blocking C → D. Similarly, when Ida traverses C → D and steps on button at C, A → B will be blocked.

FUTURE WORK

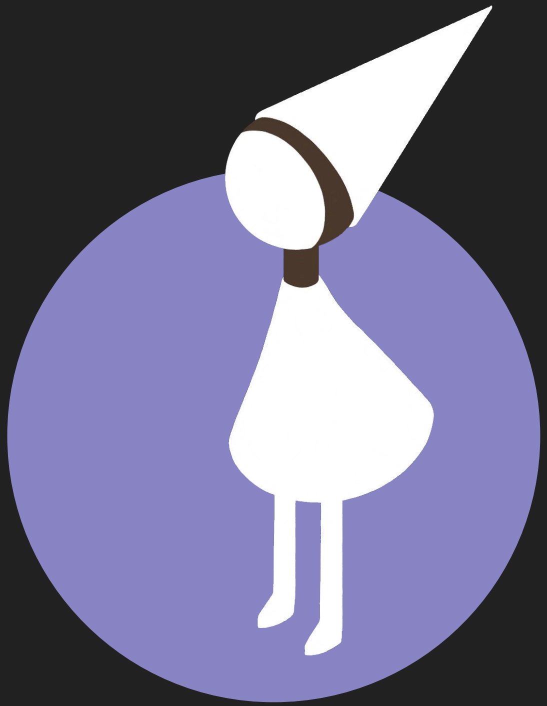
The successful construction of a locking 2-toggle concludes our proof that Monument Valley is PSPACE-Complete.



Future work involving motion planning and Monument Valley could try to model the perspective and more advanced visual effects found in the game with various gadgets. There are also several other game mechanisms that could be studied, such as the rotating dial, totem pole, crow people, and moving blocks. Monument Valley II may also have additional mechanisms to explore (this project focused on Monument Valley I: A Quest for Forgiveness).

REFERENCES

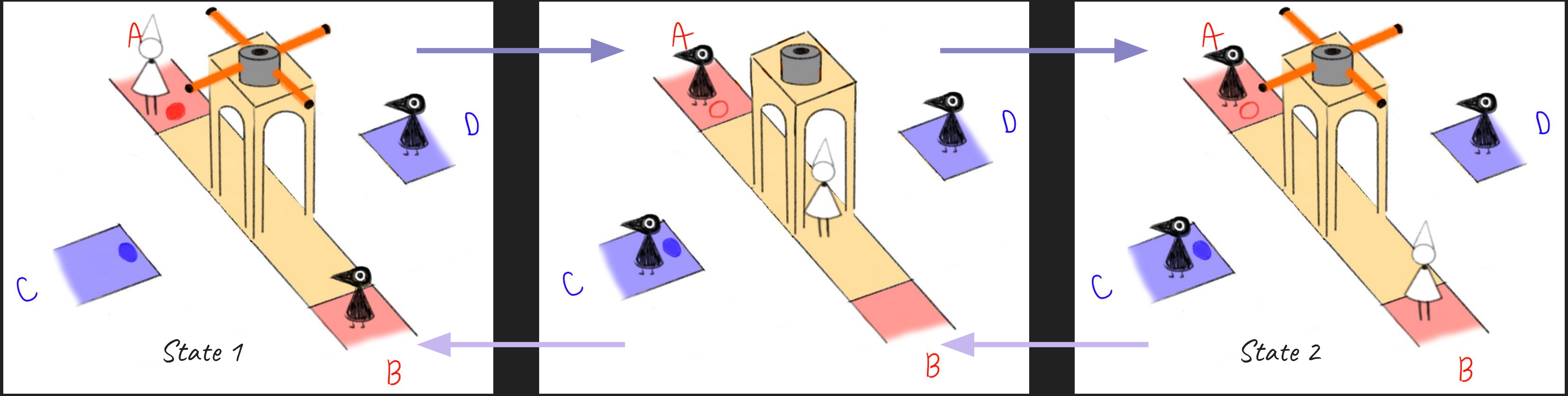
1. E. D. Demaine, I. Grosof, J. Lynch, and M. Rudoy, "Computational Complexity of Motion Planning of a Robot Through Simple Gadgets," *FUN 2018*, June 2018.
2. E. D. Demaine, D. H. Hendrickson, and J. Lynch, "A General Theory of Motion Planning Complexity: Characterizing Which Gadgets Make Games Hard," *arXiv:1812.03592*, Dec 2018.
3. "Monument Valley (video game)." *Wikipedia*. https://en.wikipedia.org/wiki/Monument_Valley_(video_game).
4. "MONUMENT VALLEY Walkthrough All Levels 1-10 + Ending | iOS, Android." *Youtube,* uploaded by CraniumCode, 11 May 2015, https://youtu.be/zchEMcZtI3M.



APPENDIX: ROTATING DIAL GADGET

This was a preliminary 3-state, 4-location, locking 2-toggle gadget we designed featuring the rotating dial and stationary crow people. It works analogously to our final, simpler 2-toggle with the buttons controlling multiple rotating blocks, except now the crows (standing on blocks behind the buttons) are the ones blocking Ida's path (when the blocks are rotated upright) and the A → B and C → D tunnels are crossing rather than parallel. The dial can rotate the path to connect either A and B or C and D. However, the dial cannot be used when Ida is on that section of the path.

Pictured below are *State 1* and *State 2* (where Ida moves between A and B). Ida can traverse C/D in a similar way, moving between *State 1* and *State 3* instead.

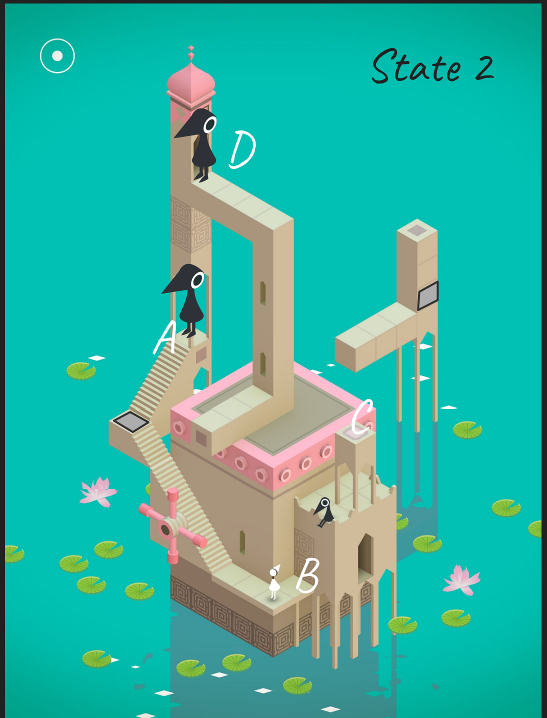


APPENDIX: PERSPECTIVE GADGET

This is an in-progress 3-state, 4-location, locking 2-toggle gadget that utilizes perspective/optical illusions (specifically, the penrose triangle). It works analogously to our other gadgets, where crow people/wall blocks can block certain paths and are controlled by buttons. The A → B and C → D tunnels are also crossing in this case, and the path fragment in the middle can be rotated by the player.

These are direct screenshots from the game, but modifications would likely need to be made (i.e. to ensure that there could feasibly be rotating blocks in the desired locations). It is also possible that a simpler perspective gadget could be created. Other notes:

- The button between A and B controls the orientation of the angled path segment on the right so we don't need another crowd at C in *state 2*. It might be possible to replicate this effect in *state 3* (and remove the crow at D), if the button between C and D were moved.
- If the buttons controlled the rotating dial on the stairs instead of the player, the crow at B could potentially be placed directly on the stairs, instead of on a rotating block at B.
- We also may have to figure out a way to ensure that Ida cannot go directly from A → C or vice versa, which may involve placing pieces of the path on different levels, or more carefully placing each crow/wall block.



THE END