The Computational Complexity of Portal and Other 3D Video Games

Erik D. Demaine
MIT CSAIL, 32 Vassar Street, Cambridge, MA 02139, USA
edemaine@mit.edu

Joshua Lockhart[1]
Department of Computer Science, University College London, London, WC1E 6BT, UK
joshua.lockhart.14@ucl.ac.uk

Jayson Lynch
MIT CSAIL, 32 Vassar Street, Cambridge, MA 02139, USA
jaysonl@mit.edu

Abstract

We classify the computational complexity of the popular video games Portal and Portal 2. We isolate individual mechanics of the game and prove NP-hardness, PSPACE-completeness, or pseudo-polynomiality depending on the specific game mechanics allowed. One of our proofs generalizes to prove NP-hardness of many other video games such as Half-Life 2, Halo, Doom, Elder Scrolls, Fallout, Grand Theft Auto, Left 4 Dead, Mass Effect, Deus Ex, Metal Gear Solid, and Resident Evil. These results build on the established literature on the complexity of video games [1, 3, 7, 18].

2012 ACM Subject Classification Dummy classification

Keywords and phrases video games, hardness, motion planning, NP, PSPACE

Digital Object Identifier 10.4230/LIPIcs.FUN.2018.19

Related Version arXiv:1611.10319

1 Introduction

In Valve's critically acclaimed *Portal* franchise, the player guides *Chell* (the game's silent protagonist) through a "test facility" constructed by the mysterious fictional organization Aperture Science. Its unique game mechanic is the Portal Gun, which enables the player to place a pair of portals on certain surfaces within each test chamber. When the player's avatar jumps into one of the portals, she is instantly transported to the other. This mechanic, coupled with the fact that in-game items can be thrown through the portals, has allowed the developers to create a series of unique and challenging puzzles for the player to solve as they guide Chell to freedom. Indeed, the Portal series has proved extremely popular, and is estimated to have sold more than 22 million copies [2, 20].

We analyze the computational complexity of Portal following the recent surge of interest in complexity analysis of video games and puzzles. Examples of previous work in this area includes NP-completeness of Tetris [5], PSPACE-completeness of Lemmings [19] and

[1] Work started while author was at School of Electronics, Electrical Engineering and Computer Science, Queen's University, Belfast, BT7 1NN, UK

9th International Conference on Fun with Algorithms (FUN 2018).
Editors: Hiro Ito, Stefano Leonardi, Linda Pagli, and Giuseppe Prencipe; Article No. 19; pp. 19:1–19:21
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Super Mario Bros. [6], and hardness of many other classic video games [7, 18]. See also the surveys [4, 9, 11].

In this paper, we explore how different game elements contribute to the computational complexity of Portal 1 and Portal 2 (which we collectively refer to as *Portal*), with an emphasis on identifying gadgets and proof techniques that can be used in hardness results for other video games. We show that a generalized version of Portal with Emancipation Grills is weakly NP-hard (Section 4); Portal with turrets is NP-hard (Section 5); Portal with timed door buttons and doors is NP-hard (Section 6); Portal with High Energy Pellet launchers and catchers is NP-hard (Section 7); Portal with Cubes, Weighted Buttons, and Doors is PSPACE-complete (Section 8); and Portal with lasers, laser relays, and moving platforms is PSPACE-complete (Section 8).

Table 1 summarizes these results. The first column lists the primary game mechanics of Portal we are investigating. The second and third column note whether the long fall or Portal Gun mechanics are needed for the proof. Section 2 provides more details about what these models mean. The turret proof generalizes to many other video games, as described in Section 5.4.

Mechanics	Portals	Long fall	Complexity
Emancipation Grills, No Terminal Velocity	Yes	Yes	Weakly NP-comp. (§4)
Turrets	No	Yes	NP-hard (§5)
Timed Door Buttons and Doors	No	No	NP-hard (§6)
HEP Launcher and Catcher	Yes	No	NP-hard (§7)
Cubes, Weighted Buttons, Doors	No	No	PSPACE-comp. (§8)
Lasers, Relays, Moving Platforms	Yes	No	PSPACE-comp. (§9)
Gravity Beams, Cubes, Weighted Buttons, Doors	No	No	PSPACE-comp. (§9)

Table 1 Summary of new Portal complexity results

2 Definitions of Game Elements

Portal is a single-player *platform game*: a game with the goal of navigating the avatar from a start location to an end location of a series of stages, called *levels*. The gameplay in Portal involves walking, turning, jumping, crouching, pressing buttons, picking up objects, and creating portals. The locations and movement of the avatar and all in-game objects are discretized. For convenience we make a few assumptions about the game engine, which we feel preserve the essential character of the games under consideration, while abstracting away certain irrelevant implementation details in order to make complexity analysis more amenable:

- Positions and velocities are represented as triples of fixed-point numbers in Cartesian coordinates.[2] Each velocity vector is limited in magnitude by a terminal velocity v_{max}.
- Time is discretized and represented as a fixed-point number. Parameter δ defines the amount of time advanced during each simulation time step.
- At each time step, there is only a constant number of possible user inputs: button presses and the cursor position. The user is able to apply any of these inputs within a time step.

[2] The actual game uses floats in many instances. We claim that all our proofs work if we round the numbers involved, and only encode the problems in the significand.

- The cursor position is represented by two fixed-point numbers in spherical coordinates.
- At each time step, we update all objects' positions and velocities as follows:
  - Update velocities based on acceleration from user commands and from gravity: $\vec{v}_{t+1} = \vec{v}_t + \delta(\vec{a}_{input} + \vec{a}_\gamma)$ where $\vec{a}_\gamma = [0, 0, -\gamma]$ and g is a constant.
  - If a velocity vector $\vec{v}_{t+1}$ has magnitude $> v_{max}$, scale it down to have magnitude v_{max}.
  - Update positions according to these velocities: $\vec{p}_{t+1} = \vec{p}_t + \delta\vec{v}$.
  - Check for collisions by extruding the objects into a fourth temporal dimension by δ and checking for intersection of those objects.[3]
  - For the purposes of this paper, we define a collision model only between single moving objects and non-moving objects, as this is all we need in our proofs possibly involving collisions (Sections 4 and 7). We ignore details of more complex collisions as they are not relevant to our results.
  - For an inelastic collision between a moving object A and a non-moving object B, we calculate the first time $\delta' \leq \delta$ at which the objects would intersect, and move A instead to this position (scaling the velocity vector by δ' instead of δ). Then we project A's velocity vector onto the surface of B at the point of intersection.
  - For an elastic collision, we similarly calculate the first time of intersection and update the position of A, but update the velocity vector instead to its reflection off of the surface at the point of intersection.
  - If an object passes through a portal, its velocity vector is rotated by the rotation that brings the entering portal frame to the exiting portal frame.
- Portals from the portal gun and bullets from turrets are resolved instantaneously in a single time step by line-of-effect rather than any ballistic simulation.[4]

In Portal, a *level* is a description of the polygonal surfaces in 3D defining the geometry of the map, along with a simulation rate and a list of game elements with their locations and, if applicable, connections to each other. In general, we assume that the level can be specified succinctly as a collection of polygons whose coordinates may have polynomial precision, (and thus so can the player coordinates), and thus exponentially large values (ratios). This assumption matches the Valve Map Format (VMF) used to specify levels in Portal, Portal 2, and other Source games [16]. A realistic special case is where we aim for *pseudopolynomial* algorithms, that is, we assume that the coordinates of the polygons and player are assumed to have polynomial values/ratios (logarithmic precision), as when the levels are composed of explicit discrete blocks. This assumption matches the voxel-based P2C format sometimes used for community-created Portal 2 levels [15].

In this work, we consider the following decision problem, which asks whether a given level has a path from the given start location the end location.

▶ Problem 1. PORTAL

Parameter: A set of allowed gameplay elements.

Input: A description of a Portal level using only allowed gameplay elements, and spatial coordinates specifying a start and end location.

Output: Whether there exists a path traversable by a Portal player from the start location to the end location.

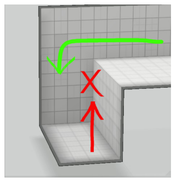
[3] This approach is precise, and should reasonably capture the relevant dynamics in the game, but computationally inefficient and likely not how collision detection is performed in practice.

[4] The end of Portal 2 gives a very large lower bound on the speed of effect of the portal gun.

3 Game Element Descriptions

The key game mechanic, the *Portal Gun*, creates a portal on the closest surface in a direct line from the player's avatar if the surface is of the appropriate type. We call surfaces that admit portals *portalable*. There are a variety of other gameplay elements which can be a part of a Portal level. Below we give descriptions and images of various game elements used in Portal 1 and 2.

1. A *long fall* is a drop in the level terrain that the avatar can jump down from without dying, but cannot jump up.



It's a long way down.

2. A *door* can be open or closed, and can be traversed by the player's avatar if and only if it is open. In Portal, many mechanics can act as doors, such as literal doors, laser fields, and moving platforms. On several occasions we will assume the door being used also blocks other objects in the game, such as High Energy Pellets or lasers, which is not generally true.

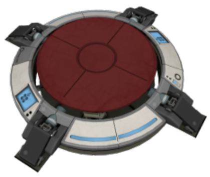
A Door in Portal 2

3. A *button* is an element which can be interacted with when the avatar is nearby to change the state of the level, e.g., a button to open or close a door.

4. A *timed button* will revert back to its previous state after a set period of time, reverting its associated change to the level too, e.g., a timed button which opens a door for 10 seconds, before closing it again.

Timed Button

5. A *weighted floor button* is a an element which changes the state of a level when one or more of a set of objects is placed on it. In Portal, the 1500 Megawatt Aperture Science Heavy Duty Super-Colliding Super Button is an example of a weighted floor button which activates when the avatar or a Weighted Storage Cube is placed on top of it. An activated weighted floor button can activate other mechanics such as doors, moving platforms, laser emitters, and gravitational beam emitters.

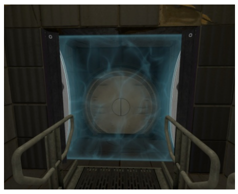


Heavy Duty Super-Colliding Super Button

6. *Blocks* can be picked up and moved by the avatar. The block can be set down and used as a platform, allowing the avatar to reach higher points in the level. While carrying a block, the avatar will not fit through small gaps, rendering some places inaccessible while doing so. In Portal, the Weighted Storage Cube is an example of a block that can be jumped on or used to activate weighted floor buttons. We will refer to Weighted Storage Cubes, Companion Cubes, etc. as simply *cubes*.

Weighted Storage Cube

7. A *Material Emancipation Grid*, also called an *Emancipation Grill* or *fizzler*, destroys some objects which attempt to pass 121 through it, such as cubes and turrets. When the avatar passes through an Emancipation Grid, all previously placed portals are removed from the map. Portals cannot be shot through an emancipation grid.

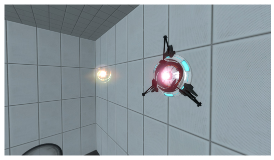


Emancipation Grid

8. The *Portal Gun* allows the player to place portals on portalable surfaces within their line of effect. Portals are orange or blue. If the player jumps into an orange (blue) portal, they are transported to the blue (orange) portal. Only one 122 orange portal and one blue portal may be placed on the level at any given time. Placing a new orange (blue) portal removes the previously placed orange (blue) portal from the level.

Portal Gun

9. A *High Energy Pellet* (HEP) is a spherical object which moves in a straight line until it encounters another object. HEPs move faster than the player avatar. If they collide with the player avatar, then the avatar is killed. If a HEP encounters a wall or another object, it will bounce off it 123 with equal angle of incidence and reflection. In Portal, some HEPs have a finite lifespan, which is reset when the HEP passes through a portal, and others have an unbounded lifespan. These unbounded HEPs are referred to as *Super High Energy Pellets*.

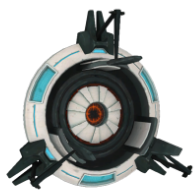


A HEP about to reach a HEP Collector

10. A *HEP Launcher* emits a HEP at an angle normal to the surface upon which it is placed. These are launched when the 124 HEP launcher is activated or when the previously emitted HEP has been destroyed.

HEP Launcher

11. A *HEP Catcher* is a device which is activated if it is ever hit 125 by a HEP. In Portal, this device can act as a button, and is commonly used to open doors or move platforms when activated.

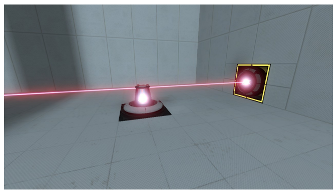


HEP Catcher

12. A *Laser Emitter* emits a *Thermal Discouragement Beam* at an angle normal to the surface upon which it is placed. The beam travels in a straight line until it is stopped by a wall or another object. The beam causes damage to the player avatar and will kill the avatar if they stay close to it for too long. We call the beam and its emitter a *laser*.

A Laser Emitter and Thermal Discouragement Beam.

13. A *Laser Relay* is an object which can activate other objects while a laser passes through it.
14. A *Laser Catcher* is an object which can activate other objects while a contacts it.



An active laser relay and laser catcher.

15. A *Moving Platform* is a solid polygon with an inactive and an active position. It begins in the inactive position and will move in a line at a constant velocity to the active position when activated. If it becomes deactivated it will move back to the inactive position with the opposite velocity.

A horizontal moving platform.

16. A *Turret* is an enemy which cannot move on its own. If the player's avatar is within the field of view of a turret, the turret will fire on the avatar. If the avatar is shot sufficiently many times within a short period of time, the avatar will die.

Turret from Portal 2

17. An *Excursion Funnel*, also called a *Gravitational Beam Emitter* emits a gravitational beam normal to the surface upon which it is placed. The gravitational beam is directed and will move small objects at a constant velocity in the prescribed direction. Importantly, it will carry Weighted Storage Cubes and the player avatar. Gravitational Beam Emitters can be switched on and off, as well as flipping the direction of the gravitational beam they emit.

A Gravity Beam and Excursion Funnel.

There are two main pieces of software for creating levels in Portal 2: the *Puzzle Maker* (also known as the *Puzzle Creator*), and the *Valve Hammer Editor* equipped with the *Portal 2 Authoring Tools*. Both of these tools are publicly available for players to create their own levels. The Puzzle Maker is a more restricted editor than Hammer, with the advantage of providing a more user-friendly editing experience. However, levels created in the Puzzle Maker must be coarsely discretized, with coarsely discretized object locations, and must be made of voxels. In particular, the Puzzle Maker uses the P2C file format while Hammer uses VMF, which restricts it to instances where the size of the level is polynomial in the size of the problem description. Furthermore, no HEP launchers or additional doors can be placed in Puzzle Maker levels. We will often comment on which of our reductions can be constructed with the additional Puzzle Maker restrictions (except, of course, the small level size and item count), but this distinction is not a primary focus of this work.

4 Portal with Emancipation Grills is Weakly NP-complete

In this section, we prove that Portal with portals and Emancipation Grills is weakly NP-hard by reduction from Subset Sum [8], which is defined like so.

▶ Problem 2. Subset Sum

Input: A set of integers $A = \{a_1, a_2, \ldots, a_n\}$, and a target value t.
Output: Whether there exists a subset $\{s_1, s_2, \ldots, s_m\} \subseteq A$ such that

$$\sum_{i=1}^{m} s_i = t.$$

The reduction involves representing the integers in A as distances which are translated into the avatar's velocity. More explicitly, the input A will be constructed from long holes the avatar can fall down, and the target will be encoded in a distance the avatar must launch themselves after falling. For the next theorem, it is necessary to allow the terminal velocity v_{max} to be specified as input to the problem (so it can scale with the level size).

▶ **Theorem 3.** Portal *with portals, long fall, Emacipation Grills, and generalized terminal velocity is weakly NP-hard.*

Proof. Refer to Figure 1. The elements of A are represented by a series of wells, each of width c and depth $b \cdot a_i$ as measured from the ceiling directly above it. Here $a_i \in A$ is the number to be encoded, $b = 2 \cdot c \cdot n^2 \cdot t$ is a large number, c is a large constant expansion factor greater than the height of the avatar plus the height she can jump, n is the number of elements in A, and t is the target value of the Subset Sum instance. The bottom of each well is a portalable surface, and the ceiling above each well is also a portalable surface. Each well also has an Emancipation Grill a distance c from the ceiling. This construction allows the avatar to shoot a portal to the bottom of the well they are falling into, and to a ceiling tile of another well, selecting the next number.

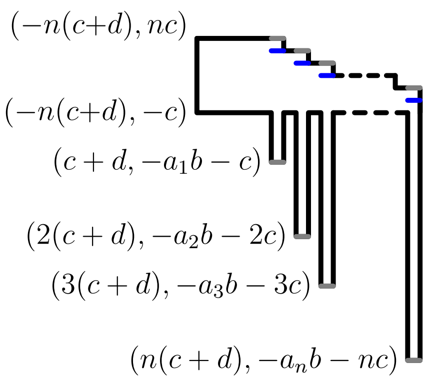


Figure 1 A cross-section of the element selection gadget, where $b = 2 \cdot c \cdot n^2 \cdot t$. Grey lines are portalable surfaces and blue lines are Emancipation Grills.

If the Subset Sum instance has a solution S, we can fall through the wells of depth $b \cdot a_i$ for each $a_i \in S$ in order, without touching any walls, for a total fall distance of $b \cdot t$. After such a fall, we reach a "target" velocity $v_t = g\sqrt{2bt}$.

We cannot allow the avatar to select the same element more than once. The Emancipation Grills below each portalable ceiling serve to remove the portal from the ceiling of the well into which the avatar is currently falling, and to prevent sending a portal up to that same ceiling tile. The stair-stepped ceiling allow the player to see the ceilings of all of the wells with index greater than the one they are currently at, but prevents them from seeing the portalable surface of the wells with a lower index. This construction ensures that the player can select each element only once using portals. The enforced order of choosing does not matter when solving Subset Sum.

We also need to prevent the avatar from moving horizontally from one well to another while falling. The avatar can move horizontally (via user input) up to a small fixed acceleration α_h. To successfully fall through one well of width c and depth at least b below the ground without hitting its side walls, the avatar's horizontal velocity v_h over vertical velocity v_v must be at most c/b. Also, after falling at least b, we must have vertical velocity $v_v \geq \sqrt{2b}$. The fall through the top part of the next well, of depth less than $(n+1)c$, will thus take $s \leq (n+1)c/v_v$ time. During this fall, the avatar can add at most $\alpha_h s \leq \alpha_h (n+1)c/v_v$ to horizontal velocity. Thus, during this fall, the avatar can travel horizontally by at most

$$\begin{aligned}
v_h s + \frac{1}{2}\alpha_h s^2 &\leq \frac{v_v c}{b}\frac{(n+1)c}{v_v} + \frac{1}{2}\alpha_h \left(\frac{(n+1)c}{v_v}\right)^2 \\
&= (n+1)\frac{c^2}{b} + \frac{\alpha_h (n+1)^2 c^2}{2v_v^2} \\
&\leq (n+1)\frac{c^2}{b} + \frac{\alpha_h (n+1)^2 c^2}{b} \\
&= \left(n+1+\alpha_h (n+1)^2\right)\frac{c^2}{b} \\
&= \left(n+1+\alpha_h (n+1)^2\right)\frac{c}{2n^2 t} \\
&= \frac{\alpha_h}{2t}c + O(1/n).
\end{aligned}$$

Setting d to be at least this value (and at least c), we prevent the player from reaching an adjacent well by horizontal travel.

We must also ensure that the player actually able to target the portable surfaces to select the elements of A. To do so, we set the time step δ to be less than $c/(10v_t)$ where v_t is the target velocity. This ensures that the player will have at least 9 time steps to target while falling c units, in particular while passing between the heights of each target surface for A and its emancipation grid.

The verification gadget (not drawn) involves two main pieces: a single portalable surface on a vertical wall ("launch point") and a $c \times c$ horizontal floor ("target platform") for the player to reach. We place the launch point so it can always be shot from the region above the wells. Relative to the launch point, the target platform is placed $g/2$ units below and at a horizontal distance of v_t in front, so that leaving the portalable surface with the target velocity v_t will cause the player to reach the target platform in 1 unit of time. The size of the target platform is much smaller than the difference ($\geq \sqrt{b} \geq n$) if the target value t differed by 1. If the player enters the final portal with horizontal velocity v_h and vertical velocity v_v, satisfying $v_h/v_v \leq c/b$ as proved above, then the avatar launches with horizontal velocity v_v and vertical velocity $v_h \leq v_v c/b$. This vertical velocity is insufficient to affect the landing position by as much as changing t by 1. Similarly, user input during the 1 unit of time has minimal effect on the horizontal velocity. ◀

All of the game elements needed for this construction can be placed in the Puzzle Maker. However, this reduction would not be constructible because maps in the Puzzle Maker appear to be specified in terms of voxels. Because Subset Sum is only weakly NP-hard [8], we need the values of the elements of A to be exponential in n. Thus we need to describe the map in terms of coordinates specifying the polygons making up the map, whereas the Puzzle Maker specifies each voxel in the map.

▶ **Theorem 4.** Portal *with portals, long fall, emancipation grills, and generalized terminal velocity can be solved in pseudopolynomial time.*

Proof. We construct a state-space graph of the Portal level. Each vertex represents a tuple comprised of the avatar's position vector within the level, the avatar's velocity vector (limited by the terminal velocity v_{max}), the avatar's orientation, the position vector of the blue portal, and the position vector of the orange portal. The vertices are connected with directed edges encoding the state transitions caused by user input. Finally, for each edge that would represent traversal through an emancipation grid, we replace it by an edge that maps to the same state of the avatar but with both portal locations removed. We can then search for a path from the initial game state to any of the winning game states in time polynomial in the size of the graph. ◀

5 Portal with Turrets is NP-hard

In this section we prove Portal with turrets is NP-hard, and show that our method can be generalized to prove that many 3D platform games with enemies are NP-hard. Although enemies in a game can provide interesting and complex interactions, we can pull out a few simple properties that will allow them to be used as gadgets to reduce solving a game from 3-SAT, defined like so.

▶ Problem 5. 3-SAT

Input: A 3-CNF boolean formula f.

Output: Whether there exists a satisfying assignment for f.

This proof follows the architecture laid out in [1]:

1. The enemy must be able to prevent the player from traversing a specific region of the map; call this the *blocked region*.
2. The player avatar must be able to enter an area of the map, which is path-disconnected from the blocked region, but from which the player can remove the enemy in the blocked region.
3. The level must contain long falls.

We further assume that the behavior of the enemies is local, meaning an interaction with one enemy will not effect the behavior of another enemy if they are sufficiently far away. In many games one must also be careful about ammo and any damage the player may incur while interacting with the gadget, because these quantities will scale with the number of literals. Here long falls serve only in the construction of one-way gadgets, and can of course be replaced by some equivalent game mechanic. Similarly, a 2D game with these elements and an appropriate crossover gadget should also be NP-hard. The following is a construction proving Portal with Turrets is NP-hard using this technique. Note that these gadgets can be constructed in the Portal 2 Puzzle Maker.

5.1 Literal

Each literal is encoded with a hallway with three turrents placed in a raised section, illustrated in Figure 2. The hallway must be traversed by the player, starting from "Traverse In", ending at "Traverse Out". If the turrets are active, they will kill the avatar before the avatar can cross the hallway or reach the turrets. The literal is true if the turrets are deactivated or removed, and false if they are active. The “Unlock In” and “Unlock Out” pathways allow for the player avatar to destroy the turrets from behind, deactivating them and counting as a true assignment of the literal.

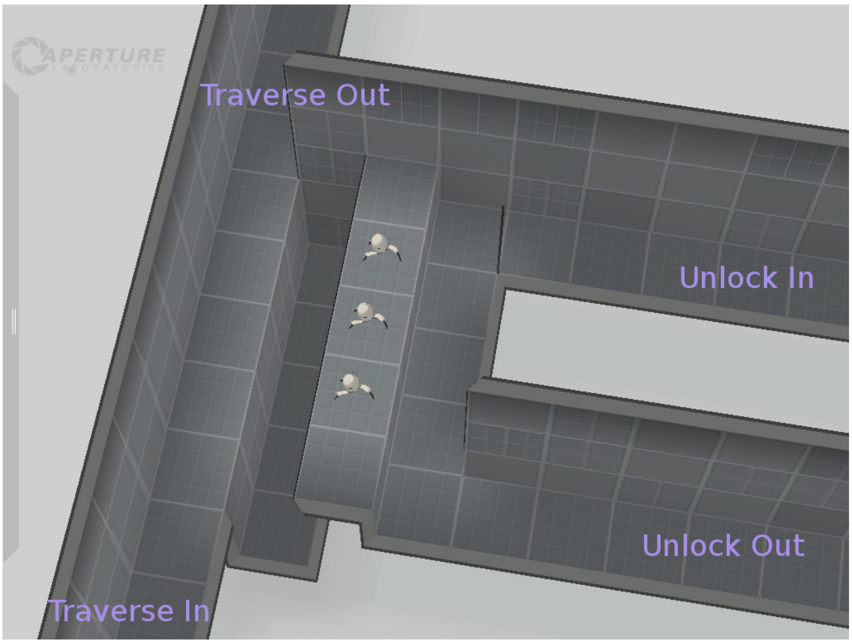


Figure 2 An example of a (currently) false literal constructed with Turrets. Labels added over the screenshot denote

5.2 Variable

The variable gadget consists of a hallway that splits into two separate paths. Each hallway starts and ends with a one-way gadget constructed with a long fall. This construction forces the avatar to commit to one of the two paths. The hallways connect the “Unlock In” and “Unlock Out” paths of the literals corresponding to a particular variable. Furthermore, one path connects all of the true literals, the other connects all of the false literals.

5.3 Clause Gadget

Each clause gadget is implemented with three hallways in parallel. A section of each hallway is the “Traverse In” through the “Traverse Out” corresponding to a literal. The avatar can progress from one end of the clause to the other if any of the literals is true (and thus passable). Furthermore, each of the clause gadgets is connected in series. Figures 3 and 4 illustrate a full clause gadget.

▶ **Theorem 6.** Portal *with Turrets and long falls is NP-hard.*

Proof. Given an instance of a 3SAT problem, we can translate it into a Portal with Turrets map using the above gadgets. This map is solvable if and only if the corresponding 3SAT problem is solvable. ◀

It is tempting to claim NP-completeness because disabling the turrets need only be performed once per turret and thus seems to have a monotonically changing state. However,

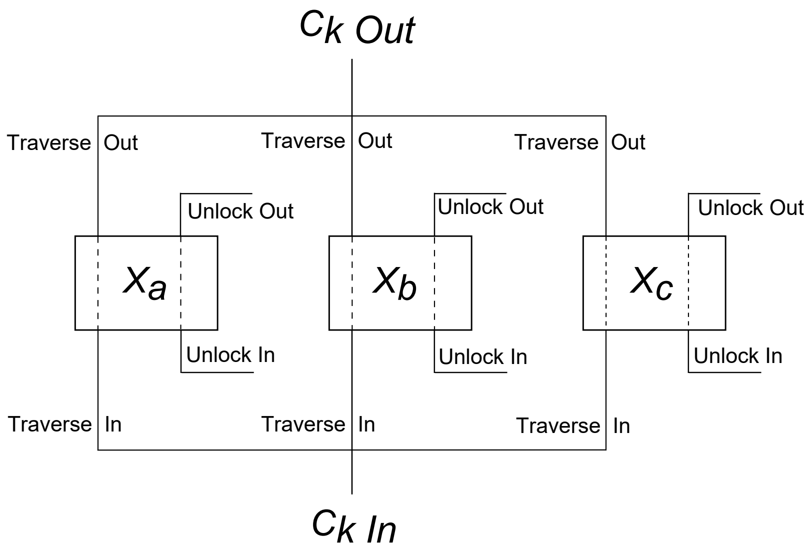


Figure 3 A diagram of clause C_k which contains variables x_a, x_b, and x_c.

the turrets themselves are physical objects that can be picked up and moved around. Their relocation add an exponential amount of state to the level. Further, if they can be jumped on top of or used to block the player in a constrained hallway, they may conceivably cause the level to be PSPACE-complete in the same way boxes can add significant complexity to a game.

5.4 Application to Other Games

While the framework we have presented is shown using the gameplay elements of Portal, similar elements to those we have used show up in other video games. Hence, our framework can be generalized to show hardness of other games. In this section we note several common features of games which would allow for an equivalent to the turret "guarding unit" in Portal. We list examples of notable games which fit the criteria. We give ideas how to use our framework to prove hardness results for these games, but it is important to note that game-specific implementation details will need to be taken into account for any hardness proof.

The first examples are games that include player controlled weapons with fixed positions, such as stationary turrets or gun emplacements. The immovable turrets should be placed at the unlock points of the literal gadget, so that they only allow the player to shoot the one desired blocking unit. Examples in contemporary video games include the Emplacement Gun in Half-Life 2, the Type-26 ASG in Half-Life, and the Anti-Infantry Stationary Guns in Halo 1 through 4.

Another set of examples are games which include a pair of ranged weapons, where one is more powerful than the other, but has shorter range. In place of the turrets in the Portal literal gadgets, we place an enemy unit equipped with the short range weapon, and give the player avatar the long range weapon. We place the blocked region such that it is in range and line of sight of the player while standing in the unlock region of the literal gadget.

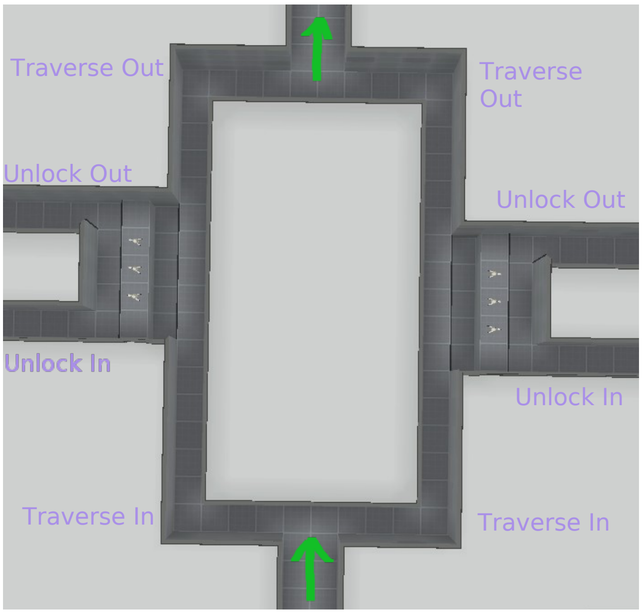


Figure 4 An example of a clause gadget with two literals.

Additionally, we place the player such that they are not in range of the enemy's weapon. Thus the player can kill the enemy from the unlock area. Suppose further that the blocked region is built in such a way that the player can only pass through it by moving within range of the enemy. One way of doing this would be to build it with tight turns. The result would be an equivalent implementation of the variable and clause gadgets from our Portal constructions. Note that a special case involves melee enemies. This construction applies to Doom, the Elder Scrolls III–V, Fallout 3 and 4, Grand Theft Auto 3–5, Left 4 Dead 1 and 2, the Mass Effect series, the Deus Ex series, the Metal Gear Solid series, the Resident Evil series, and many others. The complementary case occurs when the player has the short ranged, but more powerful weapon and the enemy has the weaker, long ranged weapon. Here the unlock region provides close proximity to the enemy unit but the locked region involves a significant region within line of sight and range of the enemy but is outside of the player's weapon's range. Although most games where this construction is applicable will also fall into the prior case, examples exist where the player has limited attacks, such as in the Spyro series.

A third case is where the environment impacts the effectiveness of attacks. For example, certain barriers might block projectile weapons but not magic spells. Skills that can shoot above or around barriers like this show up with Thunderstorm in Diablo II, Firestorm in Guild Wars, and Psi-storm in StarCraft. Another common effect is a location based bonus, for example the elevated-ground bonus in XCOM. Unfortunately these games lack a long-fall, and thus require the construction of a one-way gadget if one wishes to prove hardness.

While we have so far only covered NP-hardness, we conjecture that these games are significantly harder. Assuming simple AI and perfect information, many are likely PSPACE-complete; however, when all of the details are taken into consideration, EXPTIME or NEXPTIME seem more likely. Proving such results will require development of more sophisticated mathematical machinery.

6 Portal with Timed Door Buttons is NP-hard

We provide a new metatheorem related to Forisek's Metatheorem 2 [7] and Viglietta's Metatheorem 1 [18].

▶ Metatheorem 7. A platform game with doors controlled by timed switches is NP-hard.

Proof. We will prove hardness by reducing from finding Hamiltonian cycles in grid graphs [10]. Every vertex of the graph will be represented by a room with a timed switch in the middle. These rooms will be laid out in a grid with hallways in-between. The rooms are small in comparison to the hallways. In particular, the time it takes to press a timed button and travel across a room is δ and the time it takes to traverse a hallway is $\alpha > n \cdot \delta$ where n is the number of nodes in the graph. This property ensures the error from turning versus going straight through a room won't matter in comparison to traveling from node to node. All of the timed switches will be connected to a series of closed doors blocking the exit hallway connected to the start node. The timers will be set, such that the doors will close again after $(\alpha + \delta) \cdot (t + 1) + \varepsilon$ where ε is the time it takes to move from the switch at the start node through the open doors to the exit. The exit is thus only reachable if all of the timed switches are simultaneously active. Because we can make α much larger than ε, we can ensure that there is only time to visit every switch exactly once and then pass through before any of the doors revert. ◀

▶ **Corollary 8.** *A Portal level with only timed door buttons is NP-hard.*

A screenshot of an example map for Corollary 8 is given in Figure 5. Because the Portal 2 Workshop does not allow additional doors, the example uses collapsible stairs. We note that anything which will prevent the player from passing unless currently activated by a timed button will suffice. Moving platforms and Laser Fields are other examples. Unfortunately, the Puzzle Maker does not allow the timer length to be specified, which is a needed generalization for the reduction and available in the Hammer editor.

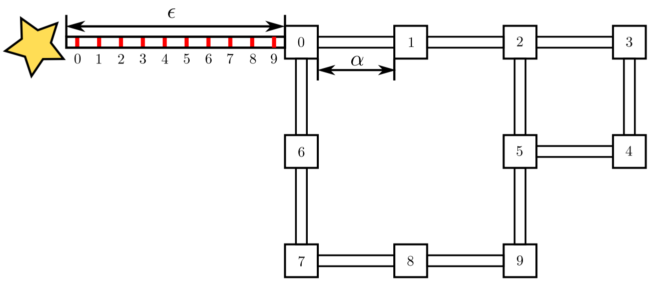


Figure 5 An example of a map forcing the player to find a Hamiltonian cycle in a grid graph.

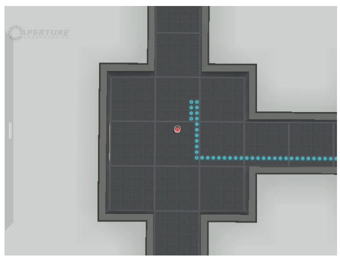


Figure 6 Close-up of a node in the grid graph.

7 Portal with High-Energy Pellets and Portals is NP-hard

In Portal, the High-Energy Pellet, HEP, is an object which moves in a straight line until it encounters another object. HEPs move faster than the player avatar and if they collide with the player avatar, the avatar is killed. If a HEP encounters another wall or object, it will bounce off of that object with equal angle of incidence and reflection. In Portal, some HEPs have a finite lifespan, which is reset when the HEP passes through a portal, and others have

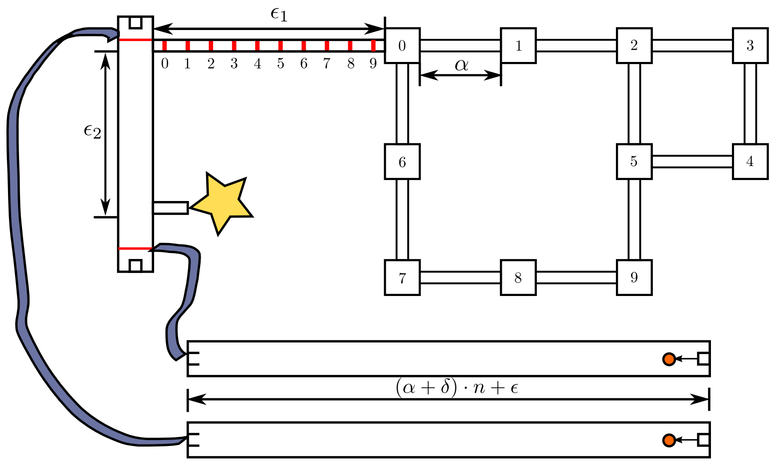


Figure 7 An example level for the HEP reduction. Not drawn to scale.

an unbounded lifespan. A HEP launcher emits a HEP normal to the surface it is placed upon. These are launched when the HEP launcher is activated or when the previous HEP emitted has been destroyed. A HEP catcher is another device that is activated if it is ever hit by a HEP. When activated this device can activate other objects, such as doors or moving platforms. HEP's are only seen in the first Portal game and are not present in the Portal 2 Puzzle Maker.

▶ **Theorem 9.** PORTAL *with Portals, High-Energy Pellets, HEP launchers, HEP catchers, and doors controlled by HEP catchers is NP-hard.*

Proof. We will reduce from finding Hamiltonian cycles in grid graphs [10]; refer to Figure 7. For this construction, we will need a gadget to ensure the avatar traverses every represented node, as well as a timing element. Each node in the graph will be represented by a room that contains a HEP launcher and a HEP catcher. They are positioned near the ceiling, each facing a portalable surface. The HEP catcher is connected to a closed door preventing the avatar from reaching the exit. The rooms are small in comparison to the hallways. In particular, the time it takes to shoot a portal, wait for it to enter the HEP Catcher, and travel across a room is δ and the time it takes to traverse a hallway is $\alpha > n \cdot \delta$ where n is the number of nodes in the graph. This property ensures the error from turning versus going straight through a room won't matter in comparison to traveling from node to node.

The timer will contain two elements. First, we will arrange for a hallway with two exits and a HEP launcher behind a door on one end. The hallway is long enough so it is impossible for the avatar to traverse the hallway when the door is open. Call this component the *time verifier*. In another area, we have a HEP launcher and a HEP catcher on opposite ends of a hallway that is inaccessible to the avatar. The catcher in this section will open the door in the time verifier. This construction ensures that the player can only pass through the time verifier if they enter it before a certain point after starting. To complete the proof, we set the timer equal to $(\alpha + \delta) \cdot n + \varepsilon_1 + \varepsilon_2$ where ε_1 is the minimum time needed for the avatar to traverse the hallway with doors, ε_2 is the minimum time needed for the avatar to traverse the time verifier, α is the minimum time it takes for the player to move to an adjacent room and change the trajectory of the HEP, and $n + 1$ is the number of HEP catchers in the level. Thus concludes our reduction from the Hamiltonian cycle problem in grid graphs. ◀

The HEP Catchers are only able to be activated once, so one may be tempted to claim this problem is in NP. This is not necessarily the case because navigating around HEP particles with more complicated trajectories might require long paths or wait times. The PSPACE-hardness of motion planning with periodic obstacles [14] suggests the natural class for this problem is actually PSPACE-complete.

8 Portal is PSPACE-complete

In this section we give a new metatheorem for games with doors and switches, in the same vein as the metatheorems in [7], [18], and [17]. We use this metatheorem to give proofs of PSPACE-completeness of Portal with various game elements, included here and in Section 9. All of the gadgets in this section can be created in the Portal 2 Puzzle Maker.

The proofs in this section revolve around constructing game mechanics which implement a switch: the construction can be in one of two states, and the state is controllable by the player. When the avatar is near the switch, it can be freely set to either state. Each state has a set of doors which are open and others which are closed when the switch is in that state. A switch is very similar to a button in that it controls whether doors are open or closed, and the player has the option of interacting with it. The key difference is that buttons can be pressed multiple times to open or close its associated doors, and cannot necessarily be 'unpressed' to undo the action. We show that a game with switches and doors is PSPACE-complete, using similar techniques to [17].

In what follows we will use the nondeterministic constraint logic framework [9], wherein the state of a nondeterministic machine is encoded by a graph called a *constraint graph*. The state is updated by changing the orientation of the edges in such a way that constraints stored on the vertices are satisfied.

Formally, an constraint graph is an undirected simple graph $G = (V, E)$ with an assignment of nonnegative integers to the edges $w : E \to \mathbb{Z}^+$, referred to as *weights*, and an assignment of integers to the vertices $c : V \to \mathbb{Z}$, referred to as *constraints*. Each edge has an orientation $p : E \to \{+1, -1\}$. A constraint graph is fully specified by the tuple $\mathcal{G} = (G, w, c, p)$. The edge orientation p induces a directed graph $D_{G,p}$. Let $v \in V$ be a vertex of G. Its *in-neighborhood*

$$N^-(v, p) = \{w \mid (v, w) \in A\}$$

is the set of vertices of $D_{G,p} = (V, A)$ with an arc oriented towards it. The constraint graph $\mathcal{G}$ is *valid* if, for all $y \in V$, $\sum_{x \in N^-(y,p)} w((x, y)) \geq c(x)$. The state of a constraint graph can be changed by selecting an edge and multiplying its orientation by -1, such that the resulting constraint graph is valid. We say that we have *flipped* the edge.

A vertex v in a constraint graph with three incident edges x, y, o can implement an AND gate by setting $c(v) = 2$, $w(x) = w(y) = 1$, and $w(o) = 2$. Clearly, the edge o can only point away from v if both x and y are pointing towards v. In a similar fashion, we can implement an OR gate by setting $w(v) = 2$, $w(x) = w(y) = w(o) = 2$. A constraint graph where all vertices are AND or OR vertices is called an *AND/OR constraint graph*. The following decision problem about constraint graphs is PSPACE-complete.

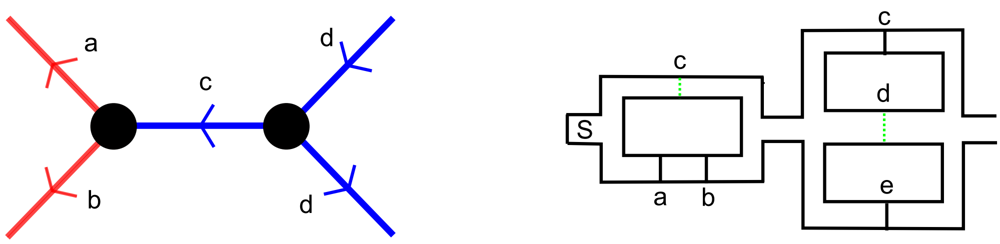
▶ Problem 10. NONDETERMINISTIC CONSTRAINT LOGIC

Input: An AND/OR constraint logic graph $\mathcal{G} = ((V, E), w, c, p)$, and a target edge $i, j \in E$.

Output: Whether there exists a constraint graph $\mathcal{G}' = ((V, E), w, c, p')$ such that $p'(\{i, j\}) = -p(\{i, j\})$, and which can be obtained from $\mathcal{G}$ by a sequence of valid edge flips.

▶ Metatheorem 11. Games with doors that can be controlled by a single switch and switches that can control at least six doors are PSPACE-complete.

Proof. We prove this by reduction from Nondeterministic Constraint Logic. The edges of the consistency graph are represented by a single switch whose state represents the edge orientation. Connected to each switch is a *consistency check gadget*. This gadget consists of a series of hallways that checks that the state of the two vertices adjacent to the simulated edge are in a valid configuration and thus that the update made to the graph was valid. Each edge switch is connected to doors in up to six consistency checks, two for itself and four for the adjacent edges. For an AND vertex, the weight-two edge is given by the door with the single hallway, and the weight one edges connect to the two doors in the other hallway. For an OR vertex we have a hallway that splits in three, each with one node. An example is given in Figure 8. Each switch thus connects to five doors. All of the edge gadgets, with their constraint checks, are connected together. This construction allows the player to change the direction of any edge they choose. However, to get back to the main hallway connecting the gadgets, the graph must be left in a valid state. Off the main hallway there is a final exit connected to the target location, but blocked by a door connected to the target edge. If the player is able to flip the edge by visiting the edge gadget, flip the switch which opens the exit door, and return through the graph consistency check, then the avatar can reach the target location. ◀



(a) Section of a constraint logic graph being simulated. Blue edges are weight 2 and red edges are weight 1.

(b) Gadget simulating edge *c* in the constraint logic graph. Green dotted lines are open doors.

Figure 8 Example of an edge gadget built from switches and doors.

▶ **Theorem 12.** Portal *with any subset of long falls, portals, Weighted Storage Cubes, doors, Heavy Duty Super Buttons, lasers, laser relays, gravity beams, turrets, timed buttons, and moving platforms is in PSPACE.*

Proof. Portal levels do not increase in size and the walls and floors have a fixed geometry. Assuming all velocities are polynomially bounded, all gameplay elements have a polynomial amount of state which describes them. For example the position and velocity of the avatar or a HEP; whether a door is open or closed; and the time on a button timer. The number of gameplay elements remains bounded while playing. Most gameplay elements cannot be added while playing, and items like the HEP launcher and cube suppliers only produce another copy when the prior one has been destroyed. We only need a polynomial amount of space to describe the state of a game of Portal at any given point in time. Thus one can nondeterministically search the state space for any solutions to the PORTAL problem, putting it in NPSPACE. Thus by Savitch's Theorem [13] the problem is in PSPACE. ◀

▶ **Theorem 13.** PORTAL *with Weighted Storage Cubes, doors, and Heavy Duty Super Buttons is PSPACE-complete.*

Proof. We will construct switches and doors out of doors, Weighted Storage Cubes, and Heavy Duty Super Buttons. Then, we invoke Metatheorem 11 to complete the proof. A switch is constructed out of a room with a single cube and two buttons as in Figure 9. Which of the buttons being pressed by the cube dictates the state of the switch. Each button is connected to the corresponding doors which should open when the switch is in that state. To ensure the switch is always in a valid state, we put an additional door in the only entrance to the room. This door is only open if at least one of the two buttons is depressed. Furthermore, this construction prevents the cube from being removed from the room to be used elsewhere. As long as there are no extra cubes in the level, the room must be left in exactly one of the two valid switch states for the avatar to exit the room. We now apply our doors and simulated switches as in Metatheorem 11 completing the hardness proof. Theorem 12 implies inclusion in PSPACE.

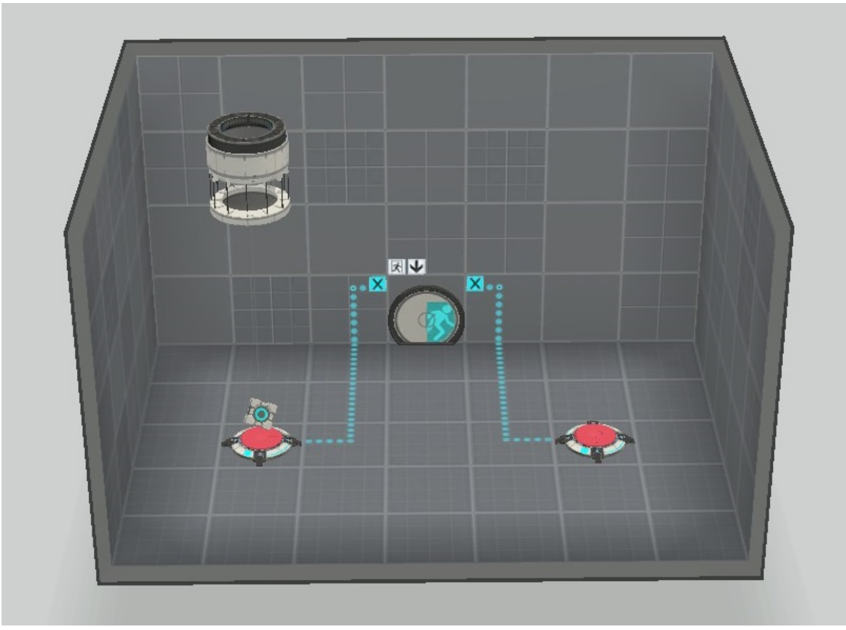


Figure 9 An example of a single switch implemented with cubes, doors, and buttons. The door will only open if at least one of the buttons is pressed.

◀

9 Additional Applications of NCL Construction

In this section we use Theorem 11 to prove additional results about Portal.

▶ **Theorem 14.** PORTAL *with lasers, relays, portals, and moving platforms is PSPACE-complete.*

Proof. We will construct doors and switches out of lasers, relays, and moving platforms allowing us to use Metatheorem 11. In Portal 2, the avatar is not able to cross through an active laser. Because lasers can be blocked by the moving platforms game element, a door can be constructed by placing a moving platform and laser at one end of a small hallway. If the moving platform is in front of the laser, the gadget is in the unlocked state. If the moving platform is to the side, then the player cannot pass through the hallway and it is in the locked state. Moving platforms can be controlled by laser relays and will switch position based on whether the laser relay is active. Lasers can be directed to selectively activate laser relays with portals, so we have a mechanism to lock or unlock the doors.

As it stands, once a new portal is created the previously opened door will revert to its previous state. To prove PSPACE-hardness, we need to make these changes persist. To do so, we introduce a memory latch gadget, shown in Figures 10 and 11. When the relay in this gadget is activated for a sufficiently long period of time, the platform will move out of the way and the laser will keep the relay active. If the relay has been blocked for enough time, the platform moves back and blocks the laser. Thus, the state of the gadget persists.

Figure 10 A memory latch in the off state.

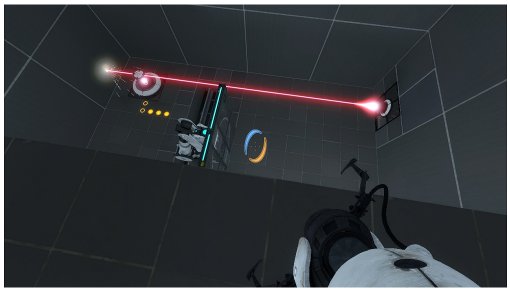


Figure 11 A memory latch in the on state.

The last construction is the switch, which we build out of two groups of lasers, moving platforms, and laser relays, as well as a memory latch. The player has the ability to change the state of the memory latch. We interpret the state of the memory latch as the state of the switch. When active, one of the relays in the latch moves a platform out of the way of one of the lasers, activating the corresponding relays and opening the set of doors to which they are connected. Another relay in the latch moves the second moving platform into the path of the second laser, deactivating its corresponding laser relays and the doors they control. Likewise, deactivating the memory latch causes both moving platforms to revert to their original positions, blocking the first laser and letting the second through. We have now successfully constructed doors and switches, so by Metatheorem 11 and Theorem 12, PSPACE-completeness follows. ◀

Note that in the proof of the preceding theorem, laser catchers could be used in place of laser relays, although the relays have the convenient property that they each need only be connected to a single moving platform. It is also possible that the proof could be adapted to use a single Reflection Cube instead of portals. Additional care would be required with respect to the construction of the door, and it would need to be the case that lasers from multiple directions blocked the avatar. Emancipation Grills or long falls with the moving platforms would simplify this particular door construction.

The game elements in the following corollary are a superset of those used in Theorem 13, so this result follows trivially. However, we prove it by using a construction similar to that in Theorem 14, as we feel that the gadgets involved are interesting. We also note that the

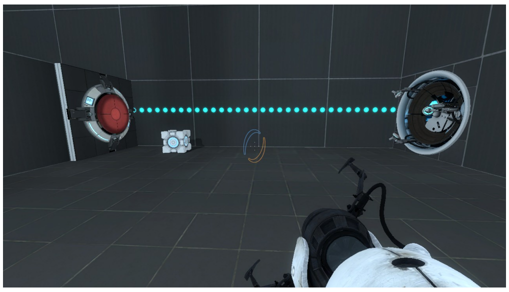


Figure 12 A memory latch in the off state.

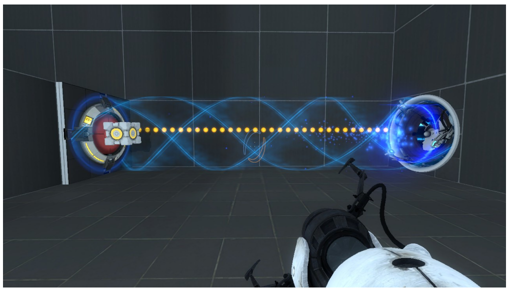


Figure 13 A memory latch in the on state.

proof only uses Heavy Duty Super Buttons placed on vertical surfaces, whereas Theorem 13 relies on their placement on the floor.

▶ **Corollary 15.** PORTAL *with gravity beams, cubes, Heavy Duty Super Buttons, and long fall is PSPACE-complete.*

Proof. When active, a gravity beam causes objects which fit inside its diameter to be pushed or pulled in line with the gravity beam emitter. Objects in the gravity beam ignore the normal pull of gravity, and thus float along their course. We construct a simple door by placing a gravity beam so that it can carry the player avatar across a pit large enough that the avatar would otherwise be unable to traverse. We hook the gravity beam emitter up to a button allowing it to be turned on and off, unlocking and locking the door.

If we wish to only use buttons placed on vertical surfaces, we are now faced with the problem of making changes to doors persist once the avatar stops holding a cube next to the button. To solve this problem, we construct a memory latch as in Theorem 14. If a weighted cube button is placed in the path of a gravity beam, a weighted cube caught in the beam can depress the button as in Figure 13. A cube on the floor near a gravity beam, as in Figure 12 will be picked up by the beam. Weighted cube buttons can activate and deactivate the same mechanics as laser catchers, including gravity beam emitters. Figures 12 and 13 demonstrate a memory latch in the off and on positions, respectively.We also note that gravity beams are blocked by moving platforms, just like lasers. At this point, we have the properties we need from the laser, laser catcher, and moving platform. We also note that the player can pick up and remove cubes from the beam, meaning that portals are not needed.

◀

10 Conclusion

In this paper we proved a number of hardness results about the video game Portal. In Sections 4 through 7 we have identified several game elements that, when accounted for, give Portal sufficient flexibility so as to encode instances of NP-hard problems. Furthermore, in Section 8 we gave a new metatheorem and use it to prove that certain additional game elements, such as lasers, relays and moving platforms, make the game PSPACE-complete. The unique game mechanics of Portal provided us with a beautiful and unique playground in which to implement the gadgets involved in the hardness proofs. Indeed, our work shows how clause, literal, and variable gadgets inspired by the work of Aloupis et al. [1] can be implemented in a 3D video game. While our results about Portal itself will be of interest to game and puzzle enthusiasts, what we consider most interesting are the techniques we utilized to obtain them. Adding new, simple gadgets to this collection of abstractions gives us powerful new tools with which to attack future problems. In Section 5.4 we identified several other video games that our techniques can be generalized to. We also believe the decomposition of games into individual mechanics will be an important tactic for understanding games of increasing complexity. Metatheorems 7 and 11 are new metatheorems for platform games. We hope that our work is useful as a stepping stone towards more metatheorems of this type. Additionally, we hope the study of motion planning in environments with dynamic topologies leads to new insights in this area.

10.1 Open Questions

This work leads to many open questions to pursue in future research. In Portal, we leave many hardness gaps and a number of mechanics unexplored. We are particularly curious about Portal with only portals, and Portal with only cubes. The removal of Emancipation Fields from our proofs would be very satisfying. The other major introduction in Portal 2 that we have not covered is co-op mode. If the players are free to communicate and have perfect information of the map, this feature should not add to the complexity of the game. However, the game seems designed with limited communication in mind and thus an imperfect-information model seems reasonable. Although perfect-information team games tend to reduce down to one- or two-player games, it has been shown that when the players have imperfect information the problem can become significantly harder. In particular, a cooperative game with imperfect information can be 2EXPTIME-complete [12].

More than the results themselves, one would hope to use these techniques to show hardness for other problems. Many other games use movable blocks, timed door buttons, and stationary turrets and may have hardness results that immediately follow. Some techniques like encoding numbers in velocities might be transferable. It would be good to generalize some of these into metatheorems which cover a larger variety of games.

Acknowledgments

All raster figures are screenshots from Valve's Portal or Portal 2, either using Portal 2's Puzzle Maker or by way of the Portal Unofficial Wiki (http://theportalwiki.com/).

References

1 Greg Aloupis, Erik D. Demaine, Alan Guo, and Giovanni Viglietta. Classic Nintendo games are (NP-)hard. In *Proceedings of the 7th International Conference on Fun with Algorithms (FUN 2014)*, Lipari Island, Italy, July 1–3 2014.

2 Eric Caoili. Portal 2 has sold over 4m copies. http://www.gamasutra.com/view/news/169967/Portal_2_has_sold_over_4M_copies.php. Accessed: 2015-08-21.

3 G. Cormode. The hardness of the Lemmings game, or oh no, more NP-completeness proofs. In *Proceedings of Third International Conference on Fun with Algorithms*, pages 65–76, 2004. URL: ../papers/cormodelemmings.pdf.

4 Erik D. Demaine and Robert A. Hearn. Playing games with algorithms: Algorithmic combinatorial game theory. In Michael H. Albert and Richard J. Nowakowski, editors, *Games of No Chance 3*, volume 56 of *Mathematical Sciences Research Institute Publications*, pages 3–56. Cambridge University Press, 2009.

5 Erik D. Demaine, Susan Hohenberger, and David Liben-Nowell. Tetris is hard, even to approximate. In *Proceedings of the 9th International Computing and Combinatorics Conference (COCOON 2003)*, pages 351–363, Big Sky, Montana, July 25–28 2003.

6 Erik D. Demaine, Giovanni Viglietta, and Aaron Williams. Super Mario Bros. is harder/easier than we thought. In *Proceedings of the 8th International Conference on Fun with Algorithms*, La Maddalena, Italy, June 2016.

7 Michal Forisek. Computational complexity of two-dimensional platform games. In *Proceedings International Conference on Fun with Algorithms (FUN 2010)*, pages 214–227, 2010.

8 Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

9 Robert A. Hearn and Erik D. Demaine. *Games, Puzzles, and Computation*. A. K. Peters, Ltd., Natick, MA, USA, 2009.

10 Alon Itai, Christos H. Papadimitriou, and Jayme Luiz Szwarcfiter. Hamilton paths in grid graphs. *SIAM Journal on Computing*, 11(4):676–686, 1982.

11 Graham Kendall, Andrew J. Parkes, and Kristian Spoerer. A survey of NP-complete puzzles. *ICGA Journal*, 31(1):13–34, 2008. URL: http://dblp.uni-trier.de/db/journals/icga/icga31.html#KendallPS08.

12 Gary Peterson, John Reif, and Salman Azhar. Lower bounds for multiplayer noncooperative games of incomplete information. *Computers & Mathematics with Applications*, 41(7):957–992, 2001.

13 Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177 – 192, 1970. URL: http://www.sciencedirect.com/science/article/pii/S002200007080006X, `doi:http://dx.doi.org/10.1016/S0022-0000(70)80006-X`.

14 Klaus Sutner and Wolfgang Maass. Motion planning among time dependent obstacles. *Acta Informatica*, 26(1-2):93–122, 1988.

15 Valve Developer Community. P2C. https://developer.valvesoftware.com/wiki/P2C, 2013.

16 Valve Developer Community. Valve map format. https://developer.valvesoftware.com/wiki/VMF_documentation, 2016.

17 Tom C. van der Zanden and Hans L. Bodlaender. Pspace-completeness of bloxorz and of games with 2-buttons. In Th. Vangelis Paschos and Peter Widmayer, editors, *Algorithms and Complexity: 9th International Conference, CIAC 2015, Paris, France, May 20-22, 2015. Proceedings*, pages 403–415. Springer International Publishing, Cham, 2015. URL: http://dx.doi.org/10.1007/978-3-319-18173-8_30, `doi:10.1007/978-3-319-18173-8_30`.

18 Giovanni Viglietta. Gaming is a hard job, but someone has to do it! *Theory of Computing Systems*, 54(4):595–621, 2014.

19 Giovanni Viglietta. Lemmings is PSPACE-complete. *Theoretical Computer Science*, 586:120–134, 2015.

20 Wesley Yin-Poole. Portal sells nearly four million. http://www.eurogamer.net/articles/2011-04-20-portal-sells-nearly-four-million. Accessed: 2015-08-21.