Gaming is a hard job, but someone has to do it!

Giovanni Viglietta

University of Pisa, Italy,
viglietta@gmail.com

Abstract. We establish some general schemes relating the computational complexity of a video game to the presence of certain common elements or mechanics, such as destroyable paths, collecting items, doors activated by switches or pressure plates, etc.. Then we apply such "metatheorems" to several video games published between 1980 and 1998, including Pac-Man, Tron, Lode Runner, Boulder Dash, Deflektor, Mindbender, Pipe Mania, Skweek, Prince of Persia, Lemmings, Doom, Puzzle Bobble 3, and Starcraft. We obtain both new results, and improvements or alternative proofs of previously known results.

1 Introduction

This work was inspired mainly by the recent papers on the computational complexity of video games by Forišek [4] and Cormode [2], along with the excellent surveys on the topic by Kendall et al. [6] and Demaine et al. [3], and may be regarded as their continuation on the same line of research.

Our purpose is to single out certain recurring features or mechanics in a video game that enable general reduction schemes from known hard problems to the games we are considering. To this end, in Section 2 we produce several *metatheorems* that will be applied in Section 3 to a wealth of famous commercial video games, in order to automatically establish their hardness with respect to certain computational complexity classes (with a couple of exceptions). Because most recent commercial games incorporate Turing-equivalent scripting languages that easily allow the design of undecidable puzzles as part of the gameplay, we will focus primarily on older, "scriptless" games. Our selection includes games published between 1980 and 1998, presented in alphabetical order for better reference. Due to space limitations, not every game is properly introduced, but our constructions should be promptly understood by any casual player.

Several open problems remain: Whenever only the hardness of a game is proved with respect to some complexity class, the obviously implied question is whether the game is also complete for that class. Different variants of each game may be studied, obtained for instance by further restricting the set of game elements used in our hardness proofs.

The reader is assumed to be familiar with general computational complexity theoretic concepts and classes: For an introduction, refer to [9].

2 Metatheorems

More often than not, games allow the player to control an *avatar*, either directly or indirectly. In some circumstances, an avatar may be identified within the game only through some sort of artifice or abstraction on the game mechanics. Throughout Section 2, we will stipulate that the player's actions involve controlling an avatar, and that the elements of the game may be freely arranged in a plane lattice, or a higher dimensional space.

2.1 Location traversal and single-use paths

A game is said to exhibit the *location traversal* feature if the level designer can somehow force the player's avatar to visit several specific game locations, arbitrarily connected together, in order to beat the level. Locations may be visited multiple times in any order, but the first one is usually fixed (starting location), and sometimes also the last one is (exit location). An example of location traversal feature is the *collecting items* feature discussed in [4]: A certain number of items are scattered across different locations, and the avatar's task is to collect them all.

The *single-use paths* feature is the existence of configurations of game elements that act as paths connecting two locations, which can be traversed by the avatar at most once.

Metatheorem 1. *Any game exhibiting both location traversal and single-use paths is* ***NP****-hard.*

Proof. We give a straightforward reduction from Hamiltonian Cycle, which is **NP**-complete even for 3-regular planar graphs. Construct a plane embedding of the given 3-regular graph (perhaps an orthogonal embedding, if needed) with an additional node u dangling from a distinguished node v. Nodes are locations that must be visited, and edges are implemented as single-use paths. The starting location is placed in v and, if an exit location is required, it is placed in u. □

As Section 3 testifies, Metatheorem 1 has a wide range of applications, and it tends to yield game levels that are more "playable" than those resulting from the somewhat analogous [4, Metathm. 2], which rely on a tight time limit to traverse a grid graph. Additionally, [4, Metathm. 2] is prone to design complications in games where the avatar moves at different speeds in different directions, for instance due to gravity effects.

2.2 Doors and pressure plates

A *door* is a game element that can be open or closed, and may be traversed by the avatar if and only if it is open. A door's status may be modified by certain actions of the avatar, such as activating a *pressure plate*. A pressure plate is a button that is pushed whenever the avatar steps on it, and its effect may be either the opening or the closure of a specific door. Each pressure plate is connected to just one door, and each door may be controlled by at most two pressure plates (one opens it, one closes it).

In our diagrams, a pressure plate that is labeled $+x$ (resp. $-x$) opens (resp. closes) the unique door labeled x, and a door is initially open (resp. closed) if its border is white (resp. black).

Metatheorem 2. *If a game features doors and pressure plates, and the avatar has to reach an exit location in order to win, then:*

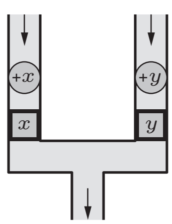
a) *Even if no door can be closed by a pressure plate, and if the game is non-planar, then it is **P**-hard.*

b) *Even if no door is controlled by two pressure plates, the game is **NP**-hard.*

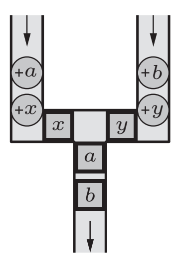
c) *If each door may be controlled by two pressure plates, then the game is **PSPACE**-hard.*

Proof. The reduction for part **(a)** is from Monotone Circuit Value, and the OR and AND gates are implemented as in Figures 1.a and 1.b. The starting location is connected to all true input literals, and the exit is located on the output.

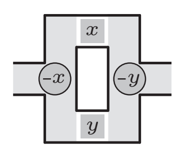
(1.a)



(1.b)

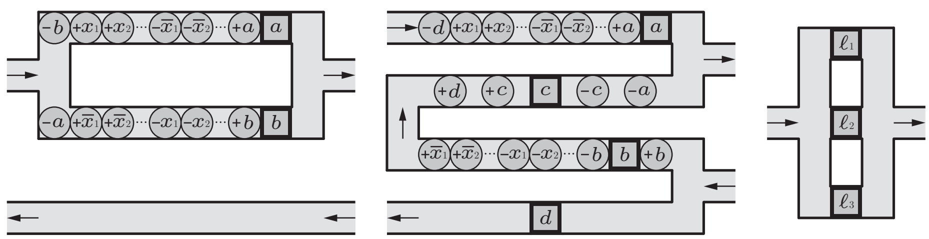


(1.c)



For part **(b)**, observe that we can implement single-use paths as shown in Figure 1.c. Since we can also enforce location traversal by blocking the exit with several closed doors, which may be opened via as many pressure plates positioned on every location, we may indeed invoke Metatheorem 1.

Finally, to prove **(c)**, we implement a straightforward reduction from Quantified Boolean Formula. Quantifier gadgets for $\exists x$ and $\forall x$ are depicted in Figures 2.a and 2.b (labels x_1, x_2, etc. correspond to different occurrences of literal x in the formula), while the clause gadget for $(\ell_1 \vee \ell_2 \vee \ell_3)$ is illustrated in Figure 2.c. The avatar starts on the upper left corner, and its "flow" is indicated by the arrows: First it traverses the upper parts of the quantifier gadgets, setting the value of each variable x by opening or closing the doors corresponding to all the occurrences of literals x and $\bar{x}$, then it traverses all the clause gadgets, and then it "turns back" and walks as much as it can through the lower parts of the quantifier gadgets, and repeats the above process with different variable assignments. When all the necessary combinations of truth assignments have been evaluated and the formula keeps being satisfied, the lower left corner of the construction (containing the exit) becomes accessible. $\square$



(2.a) (2.b) (2.c)

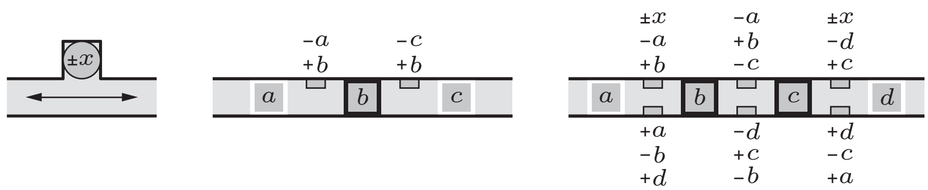
Observe that our Metatheorem 2.c is an improvement on [4, Metathm. 4], in that the *long fall* feature (and thus the concept of gravity) is not used, and it works with a more restrictive model of doors: In [4], arbitrarily many pressure plates can act on the same door, while we allow just two.

2.3 Doors and switches

A *switch* is similar to a pressure plate, except that the player may choose whether to push it or not, whenever his avatar encounters one.

Games with switches are in general not harder than games with pressure plates, because a pressure plate can trivially simulate a switch, as Figure 3.a shows. However, since the converse statement is not as clear, we will allow a single switch to act on several doors, in contrast with pressure plates. A switch acting on k doors is called a *k-switch.*

Metatheorem 3. *If a game features doors and k-switches, and the avatar has to reach an exit location in order to win, then:*



(3.a) (3.b) (3.c)

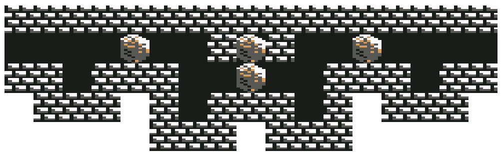
a) *If $k \geqslant 1$ and the game is non-planar, then it is* ***P****-hard.*
b) *If $k \geqslant 2$, then the game is* ***NP****-hard.*
c) *If $k \geqslant 3$, then the game is* ***PSPACE****-hard.*

Proof. The proof mirrors that of Metatheorem 2, with minor changes. For part **(a)** we merely use 1-switches as opposed to pressure plates, for part **(b)** we implement single-use paths as in Figure 3.b, and for part **(c)** we use the gadget in Figure 3.c to simulate a pressure plate for $\pm x$. □

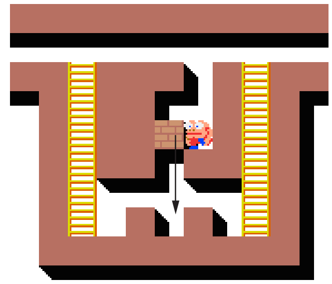
3 Applications and further results

Boulder Dash (First Star Software, 1984) is NP-hard. The game is similar to Sokoban, but with gravity. The avatar may push single boulders horizontally, excavate some special tiles, and must collect diamonds and avoid monsters. Gravity affects boulders and diamonds, but not the avatar or the monsters. A proof that "pushing blocks in gravity" is **NP**-hard is given in [5], based on a rather involved reduction scheme and several gadgets that may be adapted to work with the slightly different "physics" of Boulder Dash. Our much simpler proof relies on Metatheorem 1, via the presence of diamonds, that enforce location traversal, and the single-use path gadget of Figure 4.a. Notably, we do not use diggable tiles or enemies in our reduction, although we do require diamonds.

Deflektor (Vortex Software, 1987) is in L. This remarkable example of an "easy" commercial game features several mirrors that can be rotated by the player in order to reflect a laser ray around in 16 possible directions, collecting items and avoiding static mines, and without reflecting the laser back to the source for too long (which overheats the beam). The crucial fact is that the ray never needs to be reflected twice by the same mirror in order to reach some location, because it can be re-oriented to any direction already on its first reflection. Some tiles act as reflecting walls, some are opaque and absorb the ray, some special



(4.a)

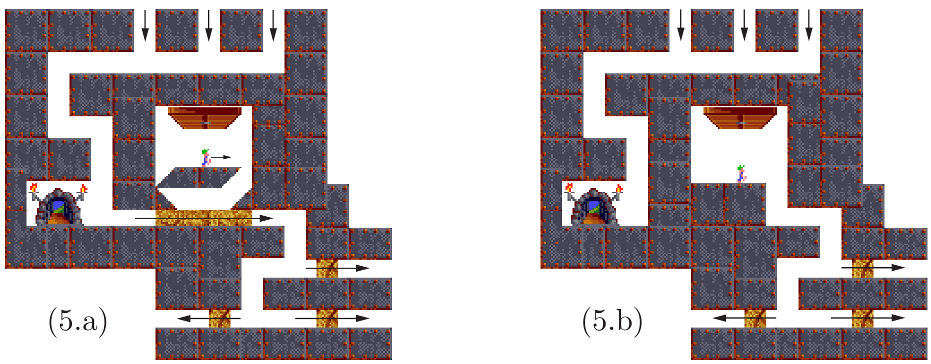


(4.b)

tiles act as teleporters, others as self-rotating *polarizators* that may be traversed by the ray only in one direction at a time. Deflektor can be reduced to the **L** problem Undirected Connectivity as follows: There are eight possible combined orientations of the polarizators, each yielding a *reachability graph* G_i, that may be computed in **L** by shooting the 16 possible rays from each mirror and extending them until each ray is absorbed or reaches a new mirror (which happens after a finite amount of reflections, because the available ray slopes are rational, and a ray that is never absorbed must have a "purely periodic" trajectory). $G^\star$ is the disjoint union of all the G_i's, in which the eight copies of the laser beam are connected to a common *starting node*. The final graph is obtained as the disjoint union of several copies of $G^\star$, one for each item to collect, in which the eight copies of the j-th item in the j-th copy of $G^\star$ are linked to the starting node of the $(j+1)$-th copy of $G^\star$. The eight copies of the last item in the last copy of $G^\star$ are connected to a common *final node*.

Doom (id Software, 1993) is PSPACE-hard. Application of Metatheorem 2.c is immediate: Pressure plates can be implemented via walkover lines and sector tags. A similar claim holds for most FPS games, adventure games, and dungeon crawls.

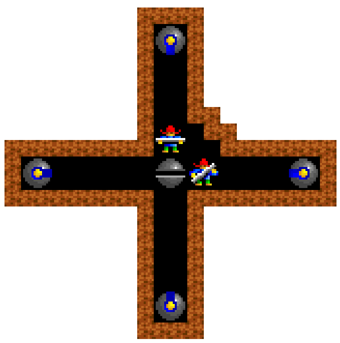
Lemmings (DMA Design, 1991) is NP-hard. This result was proved in [2] using only Digger skills, but we give two alternative constructions, based on Metatheorem 1, that use only Basher skills and only Miner skills, respectively. We model each of the n locations as in Figure 5.a, except for one *distinguished location*, depicted in Figure 5.b. Locations are suitably connected by paths, and the available skills are $2n-1$ Bashers. The Lemming in the distinguished location will be the "avatar," whose task is to visit every other location to free its mates from their cages. Once the avatar reaches a new location, it Bashes the ground below the cage, then proceeds to the right and picks one of the three paths to a different location, thus using another Bash. Lemmings breaking out of a cage are bound to walk leftward, exiting the level. The avatar remains stuck in a loop whenever it reaches an already visited location (except the distinguished location), unless it uses an extra Bash to pick a different path, in which case it will be short of Bashes later. Notice that we need double edges between locations (our paths are "oriented"!), hence we must implement crossings, which can be done as in [2]. A similar reduction works also if $2n-1$ Miners are available, instead of $2n-1$ Bashers. With a careful construction, we could even give a reduction from 3-SAT to levels with only one Lemming (and either only Bashers or only Miners).



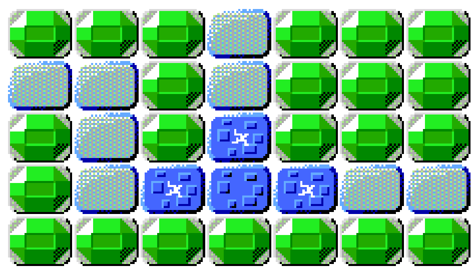
(5.a) (5.b)

Lode Runner (Brøderbund, 1983) is NP-hard. The avatar must collect gold pieces while avoiding enemies, and is able to dig holes into certain floor tiles, which regenerate after a few seconds. Both the avatar and the enemies may fall into such holes, and the avatar cannot jump. We apply Metatheorem 1: Location traversal is implied by collecting items, and a single-use path is illustrated in Figure 4.b. On the first traversal, the avatar can safely land on top of the enemy and dig a hole on the left. The AI will make the enemy fall in the hole, so the avatar may follow it, land on its top again, and proceed through a ladder, while the enemy remains forever trapped in the hole below. The avatar cannot attempt to traverse the gadget a second time without getting stuck in the hole where the enemy previously was.

Mindbender (Magic Bytes, 1989) is NL-hard. The fantasy-themed sequel of Deflektor, with a wizard shooting a ray of light, some static dwarves holding orientable mirrors, and several new game elements. The full game is arguably **PSPACE**-complete but, remarkably, even the subgame that is supposed to be isomorphic to Deflektor is in fact **NL**-complete, thus harder than Deflektor. The crucial difference is that polarizators in Mindbender are manually orientable by the player, hence the gadget in Figure 6.a allows rays coming from the left and from below (out of the "teleporters") to be reflected either rightward or upward, at any time. Such a gadget can model a node in a straightforward reduction from the **NL**-complete problem Directed Connectivity: Indeed, we may safely assume that each node has at most two incoming and two outgoing edges, while teleporters allow to model even non-planar graphs.



(6.a)



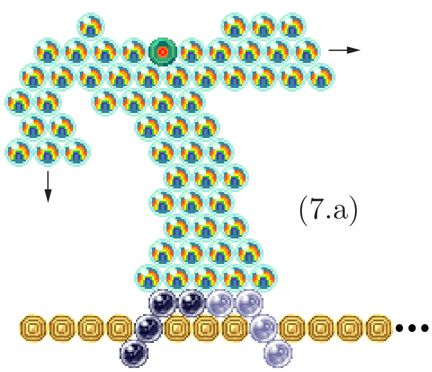
(6.b)

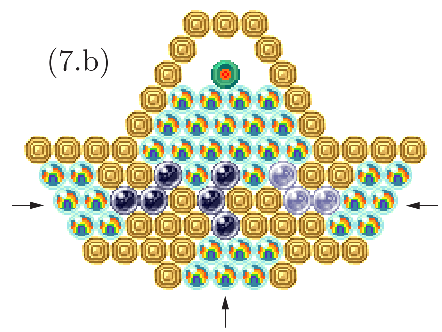
Pac-Man (Namco, 1980) is NP-hard. The decision problem is whether a level can be completed without losing lives. We assume full configurability of the amount of ghosts and ghost houses, speeds, and the durations of Chase, Scatter, and Frightened modes (see [1] for definitions). One simple way of applying Metatheorem 1 is to put a power pill in each of the n locations (except the starting location, which contains two power pills, and the "exit location," which contains just a normal pill), and connect locations with pairs of parallel *directed* edges. An edge from u to v is a path, longer than $4n+6$ tiles, containing a ghost (of any color) spawning $n+2$ tiles away from u. The game alternates between Chase and Scatter mode fast enough, so that each ghost reverses direction after covering only one tile. Frightened mode lasts long enough to allow the avatar to cover $2n+3$ tiles into any path, and ghosts are so slow during Frightened mode that they can cover at most one tile. Since ghosts reverse direction if and only if the game mode changes, the result is that each ghost "patrols" a portion of its own edge (u,v), and can be eaten (and safely crossed) by the avatar only immediately after consumption of the power pill in u. Hence, single-use paths are enforced by the presence of only one power pill in each location (except the starting and exit locations).

Pipe Mania (The Assembly Line, 1989) is NP-complete. Not to be confused with KPlumber, with a similar theme but much different mechanics, in this puzzle game a long-enough pipe has to be constructed out of several pieces, randomly presented in a queue, starting and ending in two given locations. Since the player can keep constructing on the same tile until he gets the piece that he wants, he may indeed shape the pipe as he pleases. Some obstacles are also present in each level, and we may use them to model the boundaries of locations and paths, and apply Metatheorem 1. Paths are necessarily single-use, and to enforce location traversal, we build locations as large squares and set the goal pipe length to twice the sum of their areas. Indeed, we may cover the interior of each square with cross-shaped pipes, so that each tile contributes twice to the total pipe length. Moreover, if the squares are large enough, the total length of all paths becomes negligible compared to the area of a square.

Prince of Persia (Brøderbund, 1989) is PSPACE-complete. This was proved in [4], but the rather involved construction may be replaced by a somewhat simpler one given by Metatheorem 2.c, which in addition does not rely on gravity, long falls, or on doors that can be opened by more than one pressure plate. In order to prevent the prince from avoiding a pressure plate by jumping past it, we put it on an elevated tile, that has to be climbed in order to be traversed. We can even do without vertical walls (as in [4]), because they can be substituted with unopenable doors.

Puzzle Bobble 3 (Taito, 1996) is NP-complete. In this Tetris-like puzzle game, levels are made of several colored bubbles, stacked in a hexagonal distribution. The player controls a cannon at the bottom of the screen, which can shoot new bubbles of random colors in any direction. Bubbles attach to each other and, whenever at least three monochromatic bubbles form a connected set as a result of a shot, they pop. (Monochromatic triplets may indeed be present in the initial level configuration, and they pop only when hit by a new bubble of the same color.) Some *anchors* hold the whole stack together and, as soon as a bubble is not in the same "connected component" with an anchor, it falls out of the screen and is eliminated. Apart from colored bubbles, there are *stone blocks* that cannot be popped (but may fall if not held up by an anchor), and *rainbow bubbles* that turn the same color of any bubble that pops next to them, and can later be popped like normal bubbles. The goal is to clear all anchors, and an anchor is cleared when all the surrounding bubbles are gone. Our reduction is from Planar 3-SAT: several variable gadgets (Figure 7.a) are stacked on top of each other on the far left of the construction, while the clause gadgets (Figure 7.b) are on the right, far above the player's cannon. To separate *variable layers* from each other and from the clause gadgets, we put long *shields* of stone blocks, extending from each variable gadget to the far right of the construction. The last shield (i.e., the one in the top layer) also extends all around the whole construction, on the right, top and left sides, preventing bubbles shot by the player from bouncing on the sides of the screen. Variables and clauses are connected via carefully shaped *fuses* made of rainbow bubbles, forking and bending as in Figure 7.a. Initially, only the bottom variable gadget





is exposed, and the player may choose whether to pop the black or the white bubbles, which correspond to different truth values. Popping one of the two sets (say, the black one) causes three rainbow bubbles to turn black and pop immediately after. This triggers a chain reaction, in which at least three new rainbow bubbles turn black and pop at each step, consuming the fuse and eventually reaching the clause gadgets. At this point, a thin colored *wire* is reached (see Figure 7.b), which pops if and only if it is black (its color tells whether the corresponding literal in the clause is positive or negative). If it pops, the reaction propagates inside the clause gadget, eliminating the anchor. Notice that the reaction can never "backfire" from the clause gadget and consume fuses corresponding to different variables, because each wire is connected to only two rainbow bubbles of its attached fuse. When the fuse of the first variable has been consumed, the remaining part of the variable layer falls (because the anchor in Figure 7.a is eliminated), including the shield. The second variable layer is then exposed, and the process continues until all fuses have been consumed, and all shields have fallen. What remains are the "unsatisfied" clause gadgets, whose wires are now impossible to reach, due to the *sheaths* made of stone blocks. This proves **NP**-hardness. Completeness holds under the assumption that the player can always choose the color of his next bubble, which is not far from true in most cases, since bubbles can be either discarded by making them bounce back to the bottom of the screen, or can be stacked somewhere (if done properly, not more than two bubbles per color need to be stacked at once).

Skweek (Loriciels, 1989) is NP-complete. Each level has blue tiles that the player's avatar has to paint pink by walking on them, while avoiding enemies. Some tiles are made of ice and do not have to be painted, the avatar slides on them and is unable to change direction until it reaches a different type of tile, or its slide is blocked by a wall. Some tiles fall apart when the avatar steps on them, opening a hole in the ground that becomes a deadly spot. Several bonuses randomly appear, including an exit door and teddy bears of several colors, which let the player immediately skip the level when collected. Our decision problem is whether a given level can be completed without losing lives, assuming that no exit doors or teddy bears ever appear. The presence of breakable tiles yields an immediate application of Metatheorem 1: Figure 6.b shows how a location is constructed. Proving completeness is tedious, due to the large amount of different power-ups and enemies, but it is straightforward.

Starcraft (Blizzard Entertainment, 1998) is NP-hard. The natural class for RTS games would be of course **EXP**, but a simple proof of **NP**-hardness can be given via a variation of Metatheorem 1, which applies, with minor changes, to most RTSs. Suppose the two players have bases on different *islands*, player B has a strong ground army but no income and no way to reach player A, while player A has no units and needs exactly x resources to train an army and barely defeat B. Player A starts with just enough resources to train a *worker*. In yet another unreachable island, there are n locations, each of which has a *main building* of A (to which workers must bring the resources they collect) and x/n resources. There is also a worker in each location, but it is "trapped" behind a resource patch, and cannot reach the main building. On each path connecting two locations, there is a *turret* (or other static defence) of B, positioned in such a way that a lone worker traversing the path is bound to be killed, but if two workers traverse it, exactly one survives.

B hopes for a draw, while A has only one strategy: Train a worker at some location, collect the resources, thus setting the second worker free, traverse a path with both workers to reach another location, and repeat. A cannot waste resources into training more than one worker, and can win if and only if the (planar) graph of locations has a Hamiltonian path.

Tron (Bally Midway, 1982) is NP-hard. One subgame is a "light cycle" race on a plane grid between the player and several opponents, in which the external walls and the trail of each light cycle are deadly obstacles. This game becomes **PSPACE**-complete if played on abstract graphs [8] whereas, for the standard plane grid version, a modification of Metatheorem 1 can be applied. We construct a configuration in which each opponent is trapped in a large rectangle, made of its own light cycle's trail. Rectangles are then arranged close together in the plane, in such a way that the thin interstices between them constitute a grid graph G on n nodes, scaled by a factor k, in which the player's light cycle is bound to run. The amount of free space in each rectangle is $kn - 2$, and the player starts from a given node v, so that he can win if and only if G has a Hamiltonian path from v (under the safe assumption that G has no degree-one nodes), which is an **NP**-complete problem even for grid graphs, as proved in [7]. Similarly to Pipe Mania, this construction is feasible for every fixed G, provided that the rectangles are large enough.

References

1. `http://home.comcast.net/~jpittman2/pacman/pacmandossier.html`.
2. G. Cormode. The hardness of the Lemmings game, or Oh no, more NP-completeness proofs. In *Proceedings of FUN'04*, 65–76, 2004.
3. E. D. Demaine and R. A. Hearn. Playing games with algorithms: Algorithmic combinatorial game theory. In *Games of No Chance 3*, edited by M. H. Albert and R. J. Nowakowski, MSRI Publications, 56:3–56, 2009.
4. M. Forišek. Computational complexity of two-dimensional platform games. In *Proceedings of FUN'10*, 214–226, 2010.
5. E. Friedman. Pushing blocks in gravity is NP-hard. Manuscript, `http://www2.stetson.edu/~efriedma/papers/gravity.pdf`, 2002.
6. G. Kendall, A. Parkes, and K. Spoerer. A survey of NP-complete puzzles. *International Computer Games Association Journal*, 31:13–34, 2008.
7. A. Itai, C. H. Papadimitriou, and J. L. Szwarcfiter. Hamilton paths in grid graphs. *SIAM Journal on Computing*, 11:676–686, 1982.
8. T. Miltzow. Tron, a combinatorial game on abstract graphs. In *Proceedings of FUN'12*, to appear.
9. C. H. Papadimitriou. *Computational complexity*. Addison-Wesley Publishing Company, Inc., 1994.