

Dinothawr

Level Design Guide



Agnes Heyer

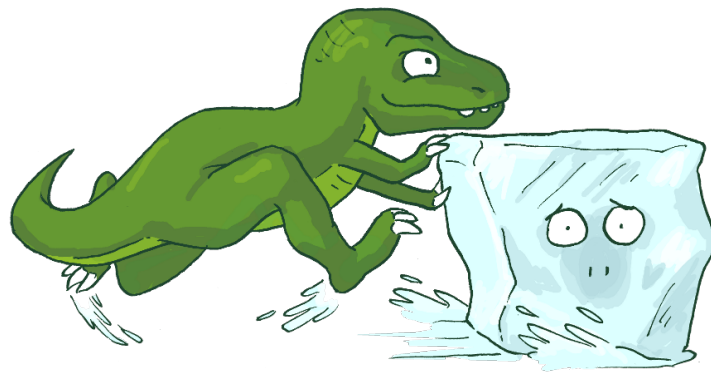
December 6, 2013

Contents

Introduction	2
Puzzle Design	3
Puzzle Elements	3
Sokoban-style Pushing	3
Navigating on Ice	3
Stopper Blocks	4
Gaps in the Ice	6
Difficulty Level	7
Uniqueness of Solutions	7
Predictability of Suitable Moves	8
Sequential Puzzle Stages	8
Shortcuts	9
Fairness	11
Level Creation	13
Design Tools	13
Graph Paper	13
Tiled	14
Level Files	14
Tile Layers	14
Layer properties	15
Level and Asset List	16
Chapter Attributes	16
Testing Your Level	16

Hi there!

This guide describes how to design and construct custom levels for the free-ware puzzle game **Dinothawr**. It describes the puzzle mechanics and some of the potential design challenges for those who wish to design their own Dinothawr levels.



Puzzle Design

This chapter is meant to help you create more difficult Dinohawr levels by describing some of the challenges and problems you may face when creating your levels.

Puzzle Elements

Blocks are either movable or immovable, and ground tiles are either slippery or non-slippery. These simple properties comprise the basis for Dinohawr's puzzle mechanics.

Having a small set of elements to work with makes it easier to get started, but it also makes the design of truly confounding levels more challenging. When you can't introduce complexity through the quantity of mechanics alone, a deeper understanding of the existing mechanics is required.

Sokoban-style Pushing

Block pushing is the simplest mechanic in Dinohawr. All this involves is moving blocks out of the way.

Sokoban-style levels are levels where you simply push blocks on a non-slippery surface. As blocks can only be moved one at a time, the challenge lies in moving the blocks in the right order to avoid congestion.

Navigating on Ice

When the player moves across a slippery surface, the player will slide until colliding with a block or reaching a non-slippery tile. This element introduces the main challenge in the Dinohawr puzzles, which is to overcome this limitation in the navigation.

Navigation levels are levels which mainly contain slippery ice. Such puzzles involve simply moving from one place to another across slippery tiles. The difficulty lies in choosing the correct movement direction for each move.



Level 2-1: A Sokoban-style level



Level 2-3: Navigating on slippery tiles

Stopper Blocks

Pushable blocks will, just like the player, slide across slippery surfaces. In some cases, blocks will slide until they hit the level border or other immovable rock tiles. This will restrict further movement of the block in one or more directions.

In order to prevent a block from getting trapped, stopper blocks may be used. Stopper blocks may be any kind of block, immobile or pushable, placed along the sliding path of another block.



Level 1-5: The stopper block

One way to obfuscate the presence of a stopper block is to use a dino block instead of an ice block. As dino blocks already have an inherent purpose, the dino block may be primarily thought of as a passive block whose only purpose is to travel along a path to the goal tile. By using the dino blocks actively, the number of ice blocks in the level may be reduced.

Sacrificial Blocks

An interesting example of the stopper block is found in puzzles where the player is expected to place "sacrificial" blocks on the borders, which means reducing the versatility of those blocks for the sake of redirecting other blocks. This can make a puzzle solution less obvious, as the player may not consider sacrificing blocks until other options have been exhausted.

Block Recycling

An ice block may serve a purpose in multiple locations within a level. This can add an extra layer of complexity, as the block will need to be moved into place to serve its first purpose before the player can easily recognize how to use it next. Recycling also reduces the number of ice blocks needed in a level, which means the possibility of alternative solutions is reduced as well.



Level 4-1: A block recycling level

Gaps in the Ice

Gaps in the ice can serve two different purposes. The gaps can function either as stopper tiles, where a block is halted when it lands on the tile, or as regular stopper blocks when the stopper tile is used to hold another block. The stopper tile can allow a block to be moved in any direction after being stopped, provided that the player can reach it.

Using a lava tile as a stopper tile may be beneficial, as lava tiles are always present and may be seen by the player as nothing more than a goal tile. Using the lava tile this way may also reduce the need for other stopper tiles.



Level 1-3: Gaps as stopper tiles

Temporary Storage

One of the more interesting ways to use stopper tiles is to provide the player with temporary storage for pushable blocks. A pushable block may serve one purpose in one part of the level, after which it becomes a hindrance and must be moved out of the way. With the aid of the stopper tile, this block may be used once more at a later stage by allowing it to be returned along the same path.

If regular stopper blocks are used, the same redirection requires a maximum tile width of three (the first slide path, the return slide path and the stopper block used for the final 90° redirection). By requiring fewer tiles, the stopper tile solution can be more practical for a complex level.



Level 2-5: Gaps for stopper blocks

Difficulty Level

The difficulty level of a given puzzle depends on several factors, such as the difficulty and design of preceding levels, differences in the ways the players approach the puzzles, and the number of alternative steps which can lead to a solution.

Uniqueness of Solutions

Difficulty is to a large extent determined by the number of possible solutions to a puzzle level. Some levels will have a single solution, which makes it necessary for the player's solution to exactly match the designer's plan. Levels with only one solution can be the most difficult to solve, but can also be the

most difficult to make. A single-solution level requires that all alternative solutions are eliminated. This usually requires playtesting, with subsequent modifications to the level and follow-up testing.

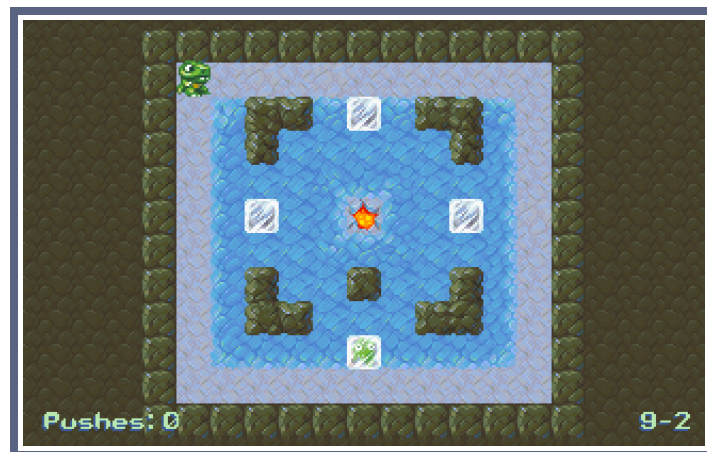
Multi-solution levels do not require the same level of testing and modification. In many cases, they may be sufficiently difficult, as long as the alternative solutions are not significantly less complex than the intended solution.

Although alternative solutions may reduce the difficulty level, they may also be advantageous in some cases, as they allow players to compare solutions and experiment to find alternatives to their initial solutions. Early levels may seem less discouraging if the players are given less rigid solutions until they reach a clearer understanding of the puzzle mechanics.

Predictability of Suitable Moves

If a level is designed so that there are several moves which seem beneficial at each point, it is more likely that the players will have to spend more time experimenting before they can clearly see the solution. The less predictable the consequences of a move is, the more time it will take the player to find the following most suitable move.

In level 9-2 shown below, it's difficult to see which block should be moved first.



Level 9-2: Several potential starting moves

Sequential Puzzle Stages

A relatively simple level can be complexified by adding additional steps which the player must complete in order to arrive at the initial configuration. As

the player is likely to focus on the core part of the design, this can make the correct starting move less obvious and increase the difficulty of the level.

However, added steps should be well integrated in the design. Otherwise, the level may be seen as two separate levels connected by a single move, which in effect is like two levels which have to be solved in succession. Without the ability to save progress in between, the player may consider the level more tedious.

Shortcuts

Ostensibly, a complex level with many blocks is more difficult to solve than a level with fewer blocks. In practice, the opposite is often the case. When there are many blocks in a level, the player will be given more opportunities to construct alternative solutions, or shortcuts. In order to remove these shortcuts and create more difficult levels, the designer must try to focus less on the desired solution and instead attempt to think like the players.

Border Shortcuts

Unrestricted borders can be problematic in levels where a walkable border surrounds an otherwise slippery level. Ice blocks can be intentionally pushed out onto this border as sacrificial blocks. The player can then use these blocks to create buffer blocks which will prevent a dino block from getting trapped at the level edges.



Level 9-3: (a) Using a border shortcut

The player may easily position the ice blocks where needed due to the lack of slippery tiles at the edges. In addition, multiple ice blocks stacked

at the edges can serve as the desired offset required to reach the goal tile through redirection. Thus, the player can circumvent any obstacles placed in the center of the level.

A simple, but likely inadequate, solution is to move the goal tile further away from the edges. However, this requires that there is enough available space in the level. If the puzzle contains block paths leading into the goal tile from multiple directions, it may be difficult to move the goal tile while simultaneously preserving the connections to each path.

A solution to the border shortcut problem which is less likely to break the existing puzzle is to place rock tiles along the edges of the slippery area to prevent circumvention. L-shaped rock formations with single-block protrusions can be used to trap blocks, but note that they can also trap the player if placed in all four corners. In the example below, a lava tile serving as a stopper tile will prevent the player from getting trapped.



Level 9-3: (b) Corner traps

Gap Tile Shortcuts

Ice gap tiles are multi-function tiles, which can function as either stopper tiles as is or stopper blocks if filled. This added complexity can make it challenging to account for every new possibility in a stopper tile puzzle. For instance, a stopper tile intended only for redirection or temporary storage can be used to hold a stopper block, which may offset or otherwise change the block paths available to the player.

Also keep in mind that the lava tile counts as a stopper tile, which means it could be used to provide shortcuts if the designer is not conscious of its placement.

In level 9-5 shown below, alternative solutions were difficult to eliminate, as the intended solution only involved using the stopper tiles for redirection and not as holding tiles for stopper blocks.



Level 9-5: Potential for gaps as shortcuts

Fairness

There are ways to design levels which could be seen as unfairly difficult. A fair level could be described as one which immediately and clearly provides the player with all the information required to solve the level. For instance, a design method could be considered flawed if it encourages the player to open the level in a level editor in order to gain this information.

Redundant Blocks

Redundant blocks are ice blocks which will not be used in the solution of the level. This should be avoided, as it discourages the player from solving the level through observation. It should be noted that even if all blocks are intended for use, alternative solutions to a level may render some blocks redundant. If such solutions are discovered, the simplest approach is to eliminate the unused blocks, although it may not be necessary if the alternative solution is sufficiently obscure compared to the intended solution.

Immovable blocks may be exempted from the redundancy rule, as they are never an active part of a solution.

Goal Tile Visibility

The goal tile should always be visible so that the player can focus on solving the puzzle right away. The process of discovery won't be a part of subsequent attempts once completed, which means that it can't be considered a useful puzzle element.

Multi-screen Levels

Levels which take up more than one screen would be more difficult to solve for reasons similar to the previous point. The added difficulty in solving such a puzzle would not be based on the puzzle design itself.

Trapping the Player

Occasionally, puzzles may contain traps from which the only escape is a level reset. For instance, the player may become trapped sliding along the edges of a slippery level if the wrong move is made. Getting stuck in this way would quickly make it obvious to the player that the level has become unsolvable. Should it happen at a later stage in a more complex level, a trap might become an annoyance as the player's progress would be lost when resetting the level.

In level 8-3 shown below, the initial level configuration contains a trap at the level border. The player may easily break this trap by pushing one of the ice blocks out to the side.



Level 8-3: Use ice blocks to prevent getting trapped

Level Creation

The goal of the design process is to produce the contents of a level file. If you already have designs in mind, this chapter should be all you need.

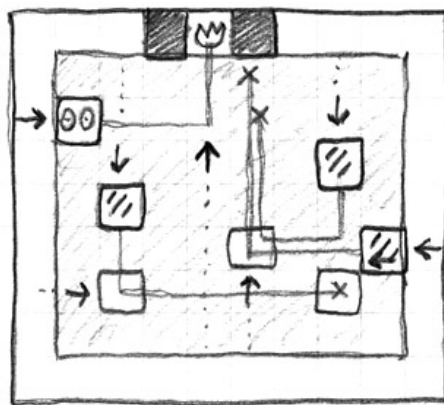
If you simply want to know how to construct level files, you can skip straight to the **Tiled** section.

Design Tools

Designing a level entirely in the level editor may be inconvenient, as the design process often involves making corrections along the way. Hence, you may need to use a combination of tools.

Graph Paper

One of the biggest benefits of the simple Dinothawr mechanics is that each level contains only a few symbolic entities, which makes Dinothawr puzzles suitable for paper design.



A paper design example

Ruled paper is suitable for less complex puzzles. For more complex puzzles, it might be a good idea to start out on paper, and then finish the design by trying it out in-game and combining modifications made on paper with modifications made in the level editor.

Tiled

To create and edit the level files, we used **Tiled**, an open source level editor. The easiest way to start creating your own level files is to find an existing level which is similar to the one you have in mind, and then open it in Tiled and modify it.

Note Make sure you select the plain **XML** format instead of Base64 in the drop-down menu found in **Preferences** → **General** → **Saving and Loading**. This will give you a human-readable level file when you save your `.tmx` file.

Level Files

The level files are XML files which describe the type of tiles used and the properties of the tile layers. These files can be edited by hand (for minor modifications) or in the level editor.

Tile Layers

The levels are divided into four separate layers based on tile type. Placing a block in a layer not intended for that type of block will lead to unwanted behavior, such as rocks being pushable and ice blocks being immovable. **The number of dino blocks must match the number of lava tiles.**

Blocks

Pushable blocks go here. Any tile placed in this layer will be pushable.

The Blocks layer must contain at least one dino block

Tile set: pushblocks

FloorTop

Immobile blocks go here. Any tile placed in this layer will be immovable.

Tile set: rocks

Floor

Tiles from the slippery set will be slippery. Other tiles will behave as regular ground tiles. **The Floor layer must contain at least one lava tile.**

Tile set: slippery, misc (for lava tile)

Bottom

Background tiles with no other purpose.

Tile set: ground

Layer properties

The **Floor** layer contains the level properties. You can edit these properties in the XML files directly, or in Tiled by right-clicking on the layer in the layer list and select Layer Properties.

The most useful properties are player starting position and facing direction. Player starting position is simply the distance in tiles from the top left corner tile. A player sprite offset, specified in pixels, is useful when the player sprite is larger than the tile size.

Default values are used if the property is not set.

start_x

Player start position in horizontal direction.

Default: 0

start_y

Player start position in vertical direction.

Default: 0

start_facing

The direction the player should face when the level loads.

Default: right

player_offset_x

Player sprite offset in horizontal direction.

Default: 0

player_offset_y

Player sprite offset in vertical direction. This is set to -1 for the current dino sprite.

Default: 0

player_sprite

The player sprite to use. **Must be set.**

Level and Asset List

The `.game` file is an XML file which contains a list of assets and levels for Dinothawr. Adding a level to the list is as simple as adding a map tag with the name of your level as source attribute.

The example below shows how to create a chapter containing the three level files `level1.tmx`, `level2.tmx` and `level3.tmx`, where at least one of the levels has to be completed.

```
<chapter name="My chapter" minimum_clear="1">
  <map source="level1.tmx"/>
  <map source="level2.tmx"/>
  <map source="level3.tmx"/>
</chapter>
```

Chapter Attributes

name

The chapter name attribute is not used in-game, but makes the level list easier to read.

minimum_clear

The number of levels which must be completed before the player can progress to the next chapter. Optional.

Default: 0

Testing Your Level

In order to test a level, it has to be listed in the `.game` file when Dinothawr is launched. When a level is reset, it is freshly loaded, meaning that you can select a level in Dinothawr, modify that level in Tiled, save the changes, and simply reset the level in Dinothawr in order to see the modifications.

Note The menu preview of the level will not be updated until you restart the game.

Thanks for reading!

That's it! Now you should know everything you need to make a Dinohawr level of your own. Have fun!

