

# Nonogram

Documentation | 02-10-2022



# Table of Contents

1. Get started quickly	3
2. Introduction	4
3. Set-Up	5
Required steps	5
4. Editor	6
5. API	8
6. Known Limitations	10
7. Support and feedback	11

# 1. Get started quickly

To quickly set up a scene and configure a Nonogram game, use the steps below.

1. Open either the landscape or portrait demo scene depending on what you prefer. You can run the game from here and see the asset in action.
2. To change the levels, navigate in your projects tab to Packages/Minigame-Nonogram/Demo/Levels configs.
3. Here, you can select a Nonogram configuration and change the settings to your liking.

## 2. Introduction

**DTT Nonogram Minigame** is a grid-based game where the user can highlight tiles or not. The user needs to highlight the correct tiles based on the numbers on the sides of the columns and rows. When the nonogram is correctly completed, you will see it form an image from the tiles that were highlighted.

The numbers on the rows and columns indicate the amount of tiles that require to be highlighted on said row or column in sequence. If there are multiple numbers on one row or column, that means that there will be at least one white tile between each number. For example, a row with the numbers 3 2 3 will start with three adjacent tiles, followed by a white tile. Then, two adjacent tiles and another white tile. Lastly, there will be three adjacent tiles to finish the row.

			1	3						4		
			4	4	1	1	3		5	1	5	
			1	1	3	1	1	5	1	1	1	3
3	5					x	x					
1	5	x			x	x	x					
1	6	x			x	x						
	5	x	x	x	x							x
2	4	1			x					x		x
	2	1			x	x	x	x	x		x	x
	3					x	x	x	x	x	x	x
	5	1						x	x		x	x
	1		x	x		x	x	x	x	x	x	x
2	1	1			x	x	x	x		x		x

## 3. Set-Up

Down below is a basic procedure described on how to implement the asset to a working example.

### Required steps

The steps described below are important to get a working example of what the asset has to offer by generating a nonogram and setting up the main foundation of the asset.

**It is critical to set references through the inspector of components you use once the references are available to reference to.**

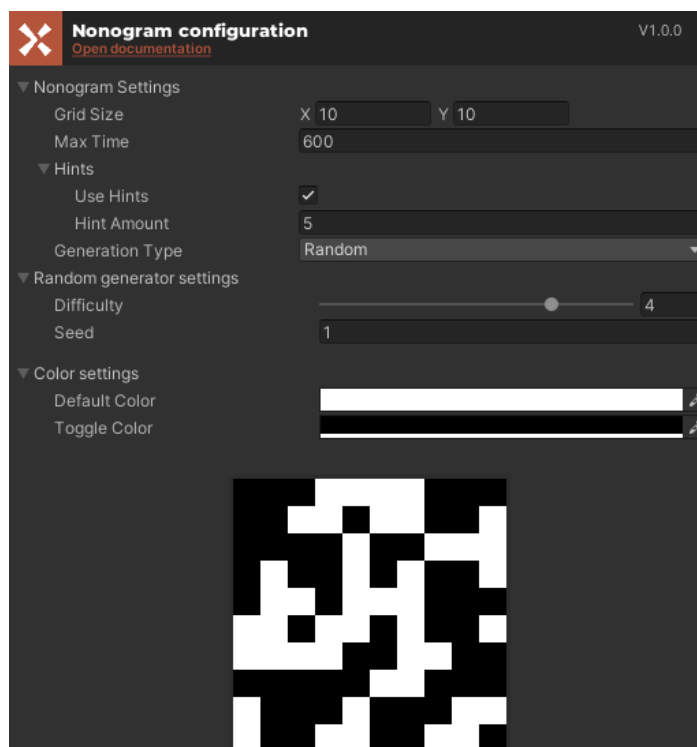
1. Create an object in the scene and attach the **NonogramManager** script to it.
2. Create a ui panel in the scene and attach a **GridRenderer** script to it.
3. Fill in the references on the **GridRenderer**.
  - a. **Manager:** Reference the **NonogramManager** here.
  - b. **Input Manager:** Reference the **InputManager** here.
  - c. **Color Manager:** Reference the **ColorManager** here.
  - d. **Grid Parent:** the area where the grid will be spawned in. Requires a **GridLayoutGroup** component.
  - e. **Number Parent Vertical/Horizontal:** parent transform of the numbers. Requires a vertical/horizontal layout group component.
  - f. Prefabs can be found at Packages/Minigame – Nonogram/Demo/Prefabs. The tile prefab needs a **TileBehaviour** script attached. The number prefab needs a **NumberDisplay** script attached.
4. Add **NonogramInputManager** to the scene and fill in the references.
  - a. **Grid renderer:** reference to the **NonogramGridRenderer**
5. Add **ColorManager** to the scene and fill in the references.
  - a. **Color button prefab:** The button used to select colors to draw in the nonogram. Default = NonogramMinigame/Demo/Prefabs/ColorButton
  - b. **Button parent:** the parent to the color buttons.
  - c. **Mark button:** Sets the selection to mark tiles with X.
  - d. **Clear mark button:** Sets the selection to clear any marks and use colors again.
  - e. **Input manager:** reference to the **InputManager**.
  - f. **manager:** reference to the **NonogramManager**

6. Create a **NonogramConfig**: in assets right-click *Create > DTT > Minigame Nonogram > Nonogram Config*. Adjust the settings to your liking.
7. Attach the **LevelSelectHandler** script to the object with the **NonogramManager** script.
8. Fill in the **LevelSelectHandler** references.
  - a. **Level Database**: The default is found in the project tab under *Packages/Minigame-Nonogram/Demo/LevelDatabase*.  
A new one can be created under *Create > DTT > Minigame Base > Level Database*.
  - b. **Nono Data**: Put the **NonogramConfig** in this list.
9. Now go to File > Build settings. Add the open scene to the list.
10. In the project tab, navigate to *Packages/Minigame Base/Assets/Scene*. Add the DTT Level Select scene to the build index. Make sure to put this scene below the other scenes.

## 4. Editor

The DTT Nonogram Minigame allows the user to create a configuration file that can be easily passed to a **NonogramManager**.

An example configuration using random generation looks like the image below.



The properties in the configuration are defined and used as follows:

**Grid Size:** Sets the size of the nonogram.

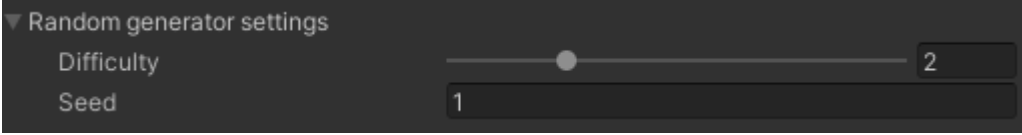
**Max Time:** Starts counting down when starting a nonogram. This value equals 3 stars. The further this number goes down the lower the score will be. Note: Once this number reaches 0, it does not end the level.

**Hints:** A hint marks a random incorrect tile.

- a. **Use hints:** Turns on hints for the current level.
- b. **Hint amount:** The amount of hints that may be used for the level.

**Generation type:** Defines whether to use **random** generation or an **image** to generate the nonogram. Hides the non-selected.

### Random generator settings:



The image shows a dark-themed settings panel titled 'Random generator settings'. It contains two controls: 'Difficulty', which is a horizontal slider with a dot in the middle and a numeric input field showing '2'; and 'Seed', which is a numeric input field showing '1'.

- c. **Difficulty:** The threshold for a tile to be marked. For each tile it rolls a number. If this number is higher than the difficulty level it will use this tile for the nonogram.
- d. **Seed:** Sets the seed for the random generation. If left on 0, it will pick a random seed.

### Color settings:

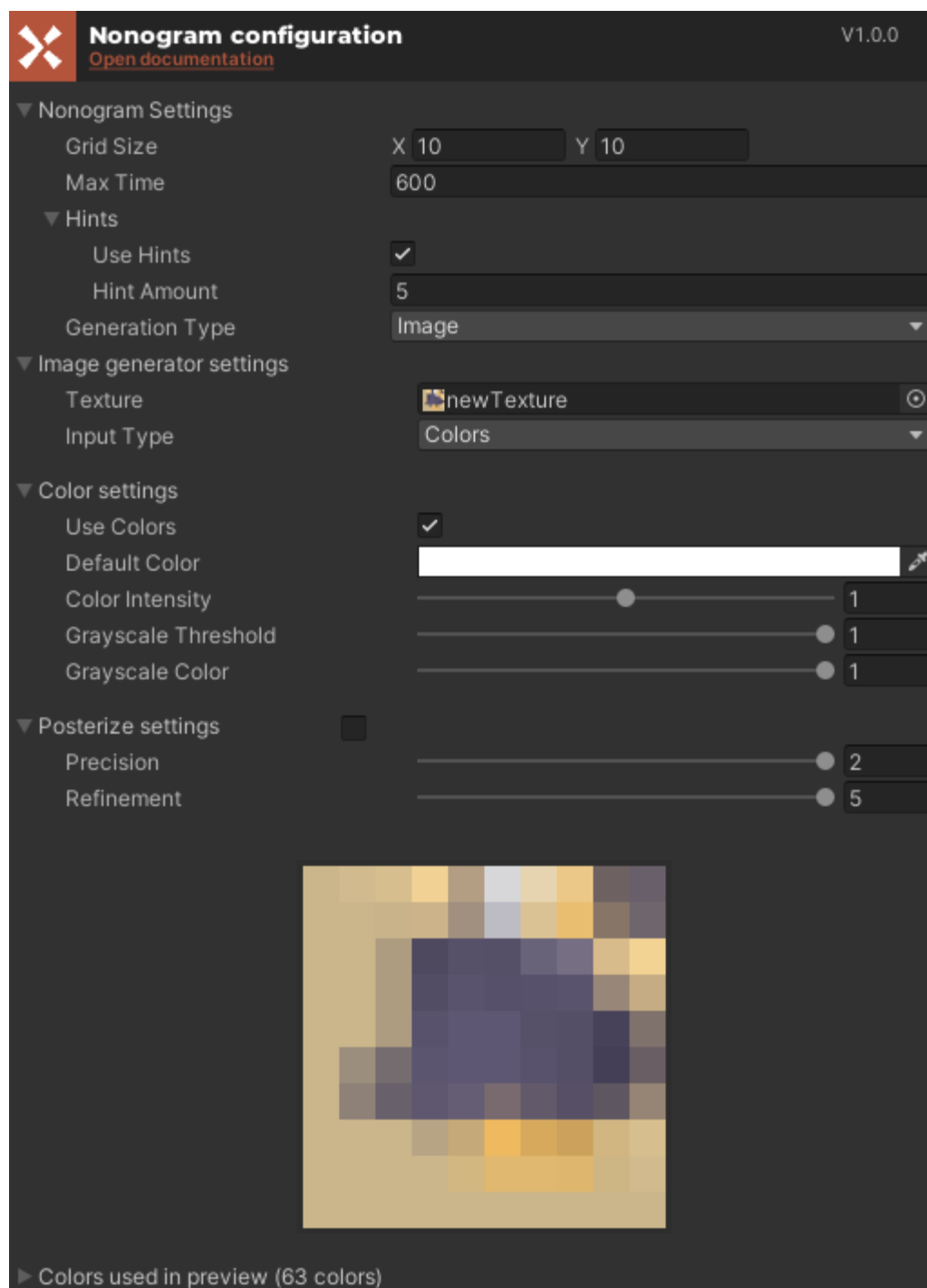


The image shows a dark-themed settings panel titled 'Color settings'. It contains two color selection controls: 'Default Color', represented by a white color swatch, and 'Toggle Color', represented by a black color swatch. Both swatches have a small icon to their right for opening a color picker.

- e. **Default Color:** The nonogram will use this color as default color. Starting with the whole grid this color.
- f. **Toggle Color:** This color will be used on the tiles that have been clicked.

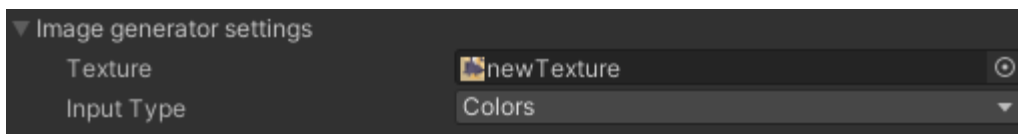


An example configuration when using an image looks like the image below.



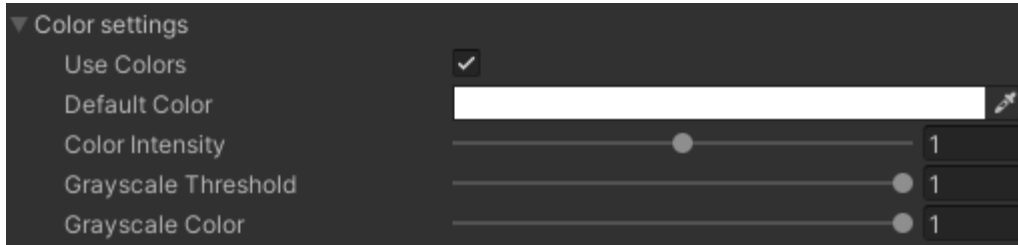
1. **Grid Size:** Sets the size of the nonogram.
2. **Max Time:** Starts counting down when starting a nonogram. This value equals 3 stars. The further this number goes down the lower the score will be. Note: Once this number reaches 0, it does not end the level.

3. **Hints:** A hint marks a random incorrect tile.
  - a. **Use hints:** Turns on hints for the current level.
  - b. **Hint amount:** The amount of hints that may be used for the level.
4. **Generation type:** Defines whether to use **random** generation or an **image** to generate the nonogram. Hides the non-selected.
5. **Image:**



- a. **Texture:** The image used to generate the nonogram.
- b. **Input type:** Defines whether to use a toggle or color selection as input type.
  - i. **Toggle:** The nonogram will be generated in black and white tiles. The user can only toggle the tiles between black and white.
  - ii. **Colors:** The nonogram will be generated in color. The user can select the colors and fill in the nonogram.

#### 6. Color settings:



- a. **Use Colors:** Will use colors when enabled. Otherwise it will use grayscale.
- b. **Default color:** The nonogram will start with this color and also use it as background.
- c. **Color intensity:** Multiplies the color output of the nonogram.
- d. **Grayscale Threshold:** When the grayscale of a tile falls below this value. It will use the default color for that tile instead.
- e. **Grayscale Color:** Used to make the nonogram fade into grayscale.

7. **Posterize settings:** Enables posterize.



- a. **Precision:** The base value to change the posterize effect.
- b. **Refinement:** The secondary value to adjust the posterize effect for more precision.

## 5. API

### Nonogram Manager

The nonogram manager manages the state of the nonogram, starts, stops, pauses, resumes and completes it. It also handles the connection between the levels and level selection.

Property Name	Type	Description
LastConfig	NonogramConfig	The config to load the level with.
GridHandler	GridHandler	Reference to the GridHandler.
InputType	InputType	The input type used for playing the nonogram.
IsPaused	bool	Indicates whether the game is paused.
IsGameActive	bool	indicates whether the game is not paused.
CurrentHintScore	int	The amount of hints that may be used.
CurrentInputSelection	IInputSelection	The input selection used for the current level.

Event Name	Argument(s)	Description
Finished	-	Gets called when the nonogram has been completed.
Started	-	Gets called when a level has started.
PauseGame	-	Gets called when the game pauses.
ResumeGame	-	Gets called when the game resumes.
Finish	NonogramData	Gives the nonogram results to level selection.
HuntUsed	int	Gets called when a hint is used. Holds the amount of hints left.

Method name	Return Type	Parameters	Description
Pause	void	-	Pauses the game.
Continue	void	-	Resumes the game.
ForceFinish	void	-	Forces the game to end.
Restart	void	-	Restarts the current game.
StartGame	void	NonogramConfig : config	Starts the game with the given configuration.
TryUseHint	void	-	Uses a hint when the hint count is above 0.
ValidateNonogram	bool	Tiles[] : tiles, int : elapsedTime	Checks if the nonogram is correct or not.
FinishLevel	void	-	Finishes the game and gives the results to the level selection.
NumberGenToDict	void	-	Generates all the possible inputSelections and adds them to the dictionary.

## Grid Handler

The Grid Handler generates a NonogramData. This data is later used to render the nonogram.

Event Name	Argument(s)	Description
NonogramDataGenerated	NonogramData	Gets called when done generating the nonogram.

Method name	Return Type	Parameters	Description
GenerateNonogramData	NonogramData	NonogramConfig : config	Generates a nonogram based on the given configuration.
UseLastData	void	NonogramData : data	Gets used instead of GenerateNonogramData if the game has already generated this nonogram.
DistinctColors	IEnumerable<Color>	Color[] : colorsToFilter	Filters the given color array and returns it with no duplicates.

## Grid Renderer

The Grid Renderer renders the nonogram into ui. It uses the NonogramData generated by the grid handler to render the grid and numbers.

Property Name	Type	Description
_manager	NonogramManager	Reference to the nonogram manager.
_inputManager	InputManager	Reference to the input manager
_colorManager	ColorManager	Reference to the color manager.
_timer	Timer	Reference to the Timer.
_tilePrefab	TileBehaviour	The prefab used to spawn in the tiles on the grid.
_gridParent	RectTransform	The parent to the grid. The tiles will be spawned as a child to this transform.
_numberDisplayPrefab	RectTransform	Prefab used to display the numbers on the side of the nonogram.
_numberParentVertical	Transform	Parent to the numbers on the vertical axis.
_numberParentHorizontal	Transform	Parent to the numbers on the horizontal axis.
_container	RectTransform	Parent of the entire grid.
_scaler	GridScaler	Reference to the grid scaler.



Method name	Return Type	Parameters	Description
RenderGrid	void	NonogramData : data	Clears the previous render and renders in a new nonogram.
SpawnTile	void	Color : correctColor	Spawns in a tile and applies the settings to it.
ValidateNonogram	void	-	Calls the Validate function on the NonogramManager.
GiveHint	void	int : i	Gets called from an event. Calls MarkRandomIncorrectTile when the event is invoked. This marks a random incorrect tile that hasn't been marked by this function yet.
SetTilesToCorrectColor	void	-	Sets all the tiles to their correct state.
SetCorrectOverTime	IEnumerator	-	Sets all the tiles to their correct state overtime.

## Interface Nonogram Generator

An interface to create generation types with. **ImageGenerator** and **RandomGenerator** are examples for this. They both generate a nonogram with certain settings.

Method name	Return Type	Parameters	Description
Generate	Color[]	NonogramConfig : config	Generates a nonogram grid and its numbers for the sides. This generation uses the settings of the given configuration.

## Interface Input Selection

An interface for default functions based on the current InputType.

**InputSelectionColor** and **InputSelectionToggle** are examples of this. They both execute the same functions, but have a different output.

Method name	Return Type	Parameters	Description
GetNonogramSideNumbers	NumberContainer	Color[] : row, Color : defaultColor, Color : toggleColor	Gets the numbers which are displayed on the sides of the nonogram.
CheckTileStatus	bool	Tile : tile	checks if the given tile is in the correct status.

## 6. Known Limitations

- Low grid size can result in a poor projection of the image. Low grid size means fewer tiles. Lower grid size will come at the cost of the resolution of the image.
- Random generation has no color mode. This is because there would be too many colors to make an actual nonogram.

## 7. Support and feedback

If you have any questions regarding the use of this asset, we are happy to help you out.

Always feel free to contact us at:

[unity-support@d-tt.nl](mailto:unity-support@d-tt.nl)

*(We typically respond within 1-2 business days)*

We are actively developing this asset, with many future updates and extensions already planned. We are eager to include feedback from our users in future updates, be they 'quality of life' improvements, new features, bug fixes or anything else that can help you improve your experience with this asset. You can reach us at the email above.

Reviews and ratings are very much appreciated as they help us raise awareness and to improve our assets.

### **DTT stands for Doing Things Together**

DTT is an app, web and game development agency based in the centre of Amsterdam. Established in 2010, DTT has over a decade of experience in mobile, game, and web based technology.

Our game department primarily works in Unity where we put significant emphasis on the development of internal packages, allowing us to efficiently reuse code between projects. To support the Unity community, we are publishing a selection of our internal packages on the Asset Store, including this one.

More information about DTT (including our clients, projects and vacancies) can be found here:

<https://www.d-tt.nl/en/>