DTT
APPS WEB GAMES

SINCE 2010

Jigsaw Puzzle

Documentation | 18-05-2022

Table of Contents

1. Get started quickly 3

2. Introduction 5

3. Set-Up 6

4. API 9

5. Known Limitations 12

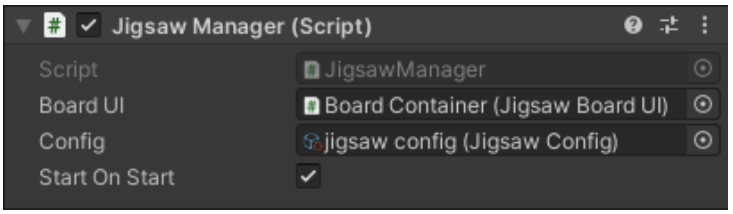
6. Support and feedback 13

1. Get started quickly

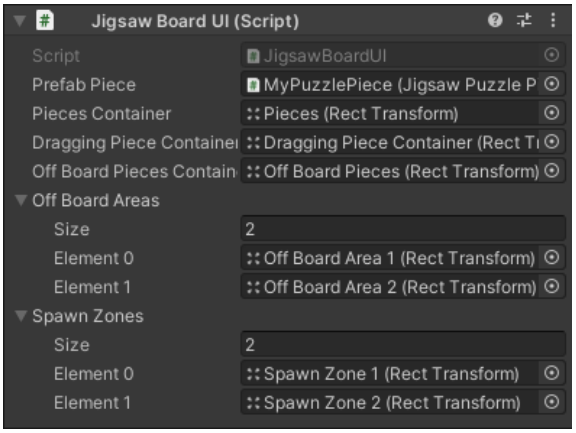
This is a summary to help you get started quickly. More details are provided below.

To get started quickly, follow these steps. For a more detailed setup, go to the Set-Up section of this document.

- First, add the **JigsawManager** component to a game object in your scene. This component requires a reference to the **JigsawBoardUI**.



- After creating the manager, create a canvas and add the **JigsawBoardUI** component to a game object somewhere in your scene. Drag the game object with the **JigsawBoardUI** component into the **Board UI** field of the **JigsawManager** component.

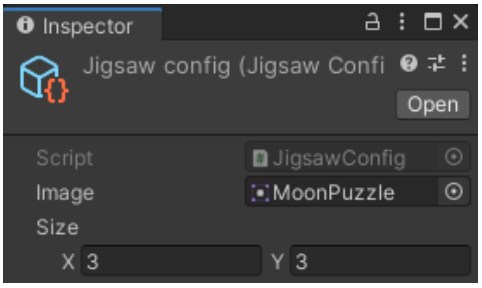


- After adding these components, create 3 separate game objects in your canvas. Make sure they have a rect transform component. Drag one of the objects into the **Pieces Container** field on your **JigsawBoardUI** component. The rect transform area of this object will serve as the size of your puzzle.
- Drag another one of these objects into the **Dragging Piece Container**. This object will simply serve as a holder for puzzle pieces when you are dragging them.
- Drag the last object into the **Off Board Pieces** field. This object will serve as a holder for the puzzle pieces that currently aren't on the puzzle board.
- After creating these objects, you can set the **Off Board Areas** and **Spawn Zones** fields on the **JigsawBoardUI**.

The rect transforms of the objects in the **Off Board Areas** will serve as the places you can drop your pieces on when they are not on the **Pieces Container**. The pieces will snap back to their last position when dragged outside the area.

The rect transforms of the objects in the **Spawn Zones** will serve as the area pieces will be randomly placed in when the game starts.

- Now you need to create a prefab for the puzzle piece. For now, you can simply use the **PuzzlePiece** prefab in the Demo/Prefabs folder, and drag it into the **Prefab Piece** field of the **JigsawBoardUI** component.
- Now go to the project tab in your unity editor and right click. Go to Create/DTT/MiniGame/Jigsaw/Config to create the **JigsawConfig** object. Here you can set the image and the size of the puzzle. Drag this object into the **Config** field on the **JigsawManager** component.



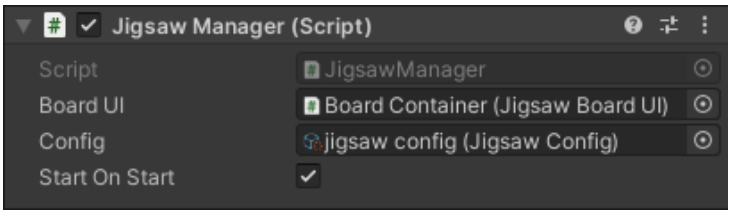
- After following these steps, you can start the game by calling the **StartGame** method from the **JigsawManager**, or by setting the **Start on Start** field on the **JigsawManager** component to true.

2. Introduction

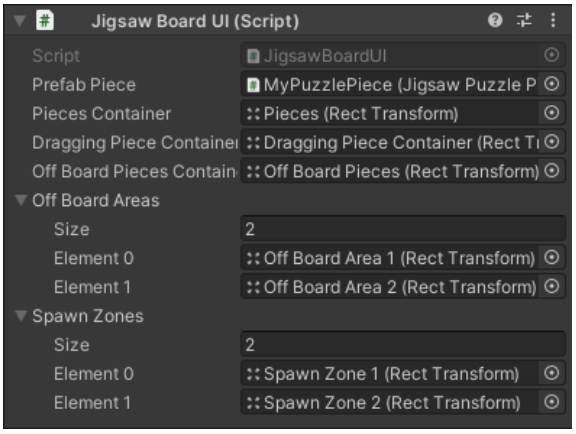
DTT Jigsaw is a Unity asset that allows you to easily implement a jigsaw puzzle into your project. The asset allows you to customize the playing field and difficulty of the puzzle. Using a Scriptable Object you can change the difficulty of the game, allowing you to have different difficulty settings per level. Furthermore, you can use any sprite in your project as the puzzle. All this allows for an easy and fast way to implement a simple jigsaw puzzle into your project.

3. Set-Up

1. Start off by adding the **JigsawManager** to a gameobject in your scene. This component manages the game loop, and allows you to start, pause and continue the game. You can also subscribe to the start and finish events of the game. It can be found in the Runtime folder of the package.



2. After creating the manager, add a canvas to your scene for the necessary UI elements. No specific settings are needed for the canvas. Add the **JigsawBoardUI** component to a gameobject in your scene. drag the gameobject with this component into the **Board UI** field on the **JigsawManager** component. The **JigsawBoardUI** handles creating and moving the UI elements of the puzzle in the correct places. You can use it to subscribe to events that are invoked when the user puts a puzzle piece on the board. The **JigsawBoardUI** can be found in the Runtime/UI folder of the package.



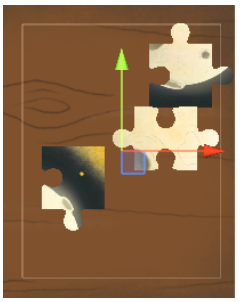
3. The **JigsawBoardUI** will require you to create some objects with a **Rect Transform** component.

   The first one is the **Pieces Container**. The rect transform of this object will serve as the area the puzzle pieces will fill. The completed puzzle will fit in this area. This doesn't affect the amount of pieces in the puzzle.

Second is the **Dragging Piece Container**. This object will simply be used to put the puzzle pieces in while they are being dragged. The parent of the puzzle pieces will be this transform while they are being dragged.

The third one is the **Off Board Pieces** rect transform, which will be the object the puzzle pieces are put in if they are dragged off the **Pieces Container** field. So when placed off the board, the parent of the puzzle pieces will be this transform.

Lastly, there are 2 rect transform arrays. The first one is the **Off Board Areas**. The rect transforms in this array will serve as the areas the puzzle pieces can be dragged too when outside the **Pieces Container**.



The second is the **Spawn Zones**. These rect transforms will be the areas the puzzle pieces can randomly be placed in when the game starts.

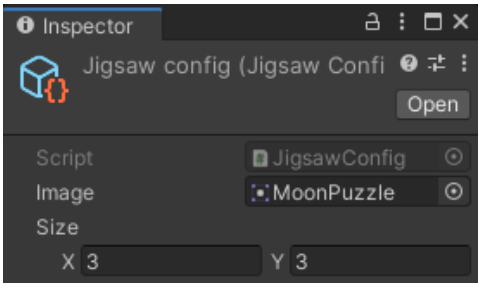
NOTE: Both the arrays need at least 1 rect transform to work.

4. After setting up the manager and jigsaw board, you can assign a prefab with a **JigsawPuzzlePieceUI** component into the **Prefab Piece** field on the **JigsawBoardUI** component. The script can be found in the Runtime/UI folder.

   The prefab will need a material with the Unlit/PuzzlePieceOptimized shader on it. To change the alpha of the puzzle piece, you need to use the **Transparency** field on the material with the shader.

   In the Demo folder under Prefabs, there is a prefab called **PuzzlePiece**, which has all the required components. You can use this as an example, or as a quick way to implement the puzzle pieces.

5. The final thing you need to create is the **JigsawConfig** object. This object can be created by going to the project tab of Unity. After right clicking, go to Create/DTT/MiniGames/Jigsaw/Config. This will create an object on which you can set the grid size and image for your puzzle. After you fill in these fields, you can drag the object into the **Config** field on the **JigsawManager** component.



6. Now you can start the game by either setting the **Start On Start** field on the **JigsawManager** to true, or by calling the **StartGame** method from the **JigsawManager** at a later moment.

4. API

JigsawManager

The **JigsawManager** component handles the starting, pausing and continuing of the game.

- **Continue** allows you to resume the game. It resumes the timer and makes the puzzle pieces interactable.
- **ForceFinish** allows you to instantly end the game. This will invoke the **Finish** event.
- **Pause** allows you to pause the game. It stops the timer and makes the puzzle pieces non interactable.
- **StartGame** starts the game using the given **JigsawConfig** in the parameter, or the config in the **Config** field on the manager if no parameter is given.

There are also some variables you can use to view the state of the game:

- **IsGameActive** is a boolean which is true when the game has started. It remains true when the game is paused.
- **IsPaused** is a boolean which is true when the game has been paused.
- **TimeElapsed** is the amount of time that has elapsed since the start of the game. The value stops increasing when the game pauses or finishes.

These events can be subscribed to:

- **Finish** is invoked when the game has ended. It passes a **JigsawResult** as a parameter. This object contains the end results of the game.
- **Started** is invoked when the game has started.

JigsawBoardUI

The **JigsawBoardUI** component handles the creation of the jigsaw UI elements. It also handles the placement of the puzzle pieces, and puts them in the correct containers.

- **CleanBoard** destroys all puzzle pieces on the board.
- **SetBoardInteractable** takes in a bool parameter that turns the puzzle pieces intractability on or off.

These events can be subscribed to:

- **PieceDroppedOnBoard** is invoked when a puzzle piece has been dropped on a grid position on the board.
- **PieceMisplaced** is invoked when a puzzle piece is dropped in an incorrect spot.

JigsawPuzzlePieceUI

The **JigsawPuzzlePieceUI** component handles the dragging and appearance of a puzzle piece.

These variables can be accessed:

- **MoveHandle** is a reference to the move handler of this object. It handles the pickup, drag and drop events of the puzzle piece.
- **Piece** is a reference to the **JigsawPuzzlePiece** that is assigned to this UI object. It holds data about the correct grid position the puzzle piece should be in.
- **RectTransform** is a reference to the rect transform of the object.

These events can be subscribed to:

- **Drop** is invoked when the puzzle piece is dropped by the user.
- **PickUp** is invoked when the puzzle piece is picked up by the user.

MoveHandle

The **MoveHandle** component makes it possible to drag an UI object.

- **Drag** is invoked while the user is dragging the object.
- **PointerDown** is invoked when the object has been clicked.
- **PointerUp** is invoked when the object has been released,

JigsawConfig

The **JigsawConfig** is a scriptable object that is used to specify the size and image for the jigsaw puzzle. It can be created in the project tab by right clicking and going to Create/DTT/MiniGames/Jigsaw/Config.

- **Image** is the image used for the puzzle.
- **Size** specifies the amount of puzzle pieces in the width and height of the board. So a width and height of 3 would mean the image is split into 9 puzzle pieces.

JigsawResult

The **JigsawResult** class holds the result data of a jigsaw game.

- **timeTaken** is the time it took to finish the game.
- **amountOfMisplacedPieces** is the amount of times the user placed a piece in the wrong position on the board

JigsawPieceOptimized Shader

The **JigsawPieceOptimized** shader is used to create puzzle pieces out of the given image in the **JigsawConfig** object.

- **Color** specifies the tint of the puzzle piece.
- **Connectors** specifies the length/depth of the connectors.
- **Transparency** specifies the alpha of the puzzle piece.

5. Known Limitations

- **Puzzle Piece Spawning**: Puzzle pieces can only be spawned in random positions using the Spawn Zones on the **JigsawBoardUI**.
- **Puzzle Piece Shader:** Because of the shader, the color field on the **JigsawPuzzlePieceUI** can't be used to change the alpha or color of the piece. Instead, the **Transparency** and **Color** field on the **PuzzlePieceOptimized** shader need to be used for the same effect.

6. 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

(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/