

Difference Puzzle Pack

User Guide

What is this pack ?

The difference puzzle pack lets you create difference puzzles by setting up 2 images and marking their differences.

The display and handling of the puzzle in the application is handled by this pack's code and it will present the player with the puzzles, one at the time.

The pack supports custom code integration by subclassing the main puzzle class and overriding certain key methods.

This product is created using our Orthello 2D framework and is setup using sprites. This way it is easy to customize and expand this puzzle pack to your desire.




How to start creating a puzzle-game ?

Best approach would be to take a copy of the demo scene as a base of operations and transform that into the puzzle scene that you would like.

Transforming meaning : adding and changing puzzles in the PuzzleGame object, changing images, adding sprites and additional functionality.

Because this puzzle game's functionality was build using our Orthello 2D framework it would be a good thing to build-up some knowledge about it, especially if you want to extend it with additional sprites and code.

Changing images or duplicating puzzles or setting difference area's is a simple task that does not need much Orthello knowledge.



The image shows a game interface for a 'Difference Puzzle Pack'. At the top, there are four small preview images: two of a red fish-like character and two of a landscape scene with trees and a sun. To the right is a directional pad with 'Back' text. Below these is a larger puzzle screen with a wooden background. It displays two identical-looking images of a girl's face with red pigtails. A green circle highlights a difference between the two images. Below the images is a black box with the number '5'. At the bottom of the puzzle screen, it says 'Find the differences to complete the puzzle.' Below the puzzle screen, there is a black box with white text that reads: 'Design your puzzle using sprites from our Orthello 2D framework. Great for quick setup and configuration!'

How is the demo scene setup?



OT : Orthello Main Object

Holds the 2 font atlases for the courier fonts used on the orthello text sprites.

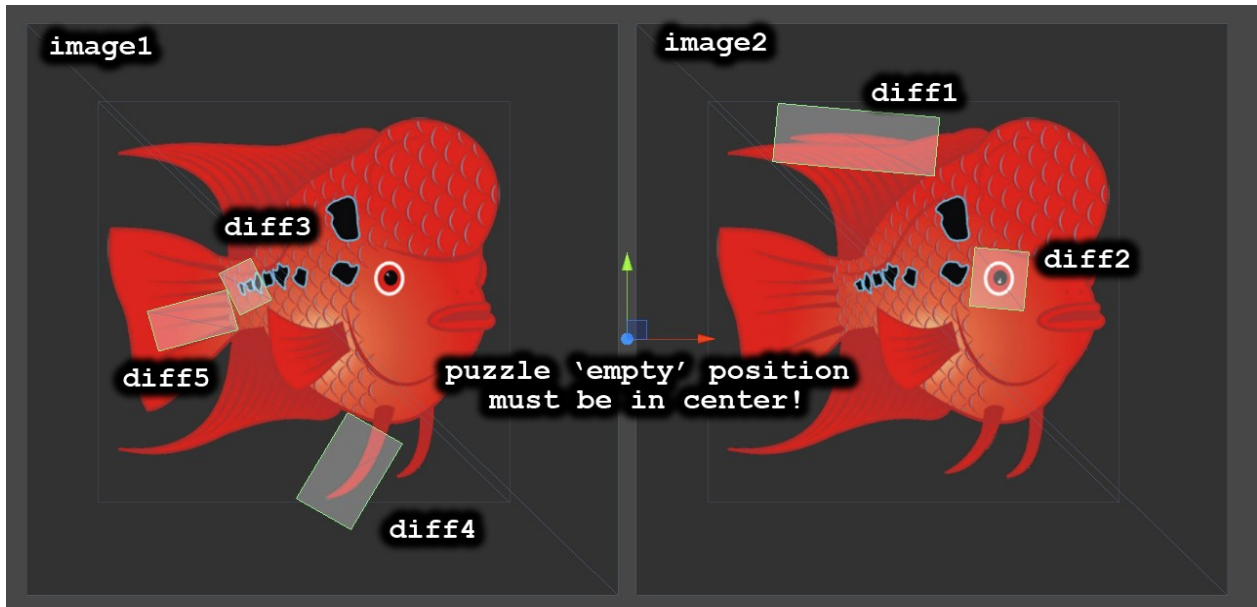
PuzzleGame : Puzzle game Structure

The PuzzleGame object that has the (subclassed) DemoPuzzleGame script attached, has some specific elements that are used in the puzzle's execution.

- **ImageOk**
Area found indicator sprite
- **labelDiffCount**
Display text sprite for difference
- **positionMoveIn**
Puzzles will be moving in from this empty's spot.
- **positionMoveOut**
Puzzles that are solved will be moving to this empty's spot
- **positionPuzzle**
The current puzzle will be resting in this empty's spot.
- **positionStore**
The puzzle's that are not being played at the time will be stored here while they wait to be moved in.
- **puzzles**
An 'empty' that contains the puzzles.

Scene puzzle structure?

Each puzzle's has a specific structure :



Make sure the puzzle's main empty (container) game object is located in the center.

On the left and right of the center are the 2 image sprites, named 'image1' and 'image2'

The difference area's are (blank/transparent) sprites that are named 'diff#' where # is an incrementing number starting at 1. This way you can set how many differences you have on your puzzle.

The 2 image's and the diff1 - diff(n) are the sprites that have to be present in each puzzle.

NOTE Because Orthello like to have its sprites having unique names, it will often add a '-{nr}' to each sprite. This will pose no problem as long as the names start with the ones describe above.

To change and add a puzzle?

When you investigate the puzzle structure, changing a puzzle would mean :

1. Setting **sprite.image** of the 2 puzzle images
(you can do this in the editor property inspector)
2. Deleting and copying diff area sprites and giving them the right name.

To add a new puzzle : just copy (Ctrl-D) an existing puzzle, naming it and following the steps above.

When running , the puzzles will be presented ordered by their name. Where you put them into your scene is of no importance as they will be hidden and moved automaticly.

Custom code integration?

To integrate your own code, the best way is done by subclassing the `DifferencePuzzleGame` class and overriding the `PuzzleSolved` (when 1 puzzle is solved) and `PuzzlesSolved` (when all puzzles are solved).

NOTE the `PuzzleSolved` method will return a float that specifies a wait time after that the next puzzle will be moved in.

Also by not calling the parent's update method (`base.Update`) from you own subclassed game class, you can pauze the game and determine when it should go on.

When you examine the Demo project and look at the `DemoGame`'s code, you see how we integrated the 'puzzle completed' dialog and the 'Wanna try again' dialog at the end.

Class properties

The **DifferencePuzzleGame Class** has the following properties and methods :

Editor

- **bool startInCenter**
The first puzzle will no be moved in but starts in center.
- **bool fadeFirst**
The first puzzle will also be faded in
- **OTEasing.EasyType moveEasing**
Easing style for moving puzzles in and out
- **float moveDuration**
How long to move in or out.
- **float fadeDuration**
How long to fade in or out.

Protected (so available in your subclass)

- **Vector2 positionMoveIn**
- **Vector2 positionMoveOut**
- **Vector2 positionMovePuzzle**
- **Vector2 positionMoveStore**
- **void Restart()**

Override

- **virtual void PuzzlesSolved();**
- **virtual float PuzzleSolved(DifferencePuzzle puzzle);**
(puzzle.nr/.name/.gameobject ...)

If you have any questions send us an email at info@wyrmtale.com

Wyrmtale Games.