# PIXELS TO PICTURES

A PROGRAMMING COURSE ON IMAGES WITH MATLAB

# Instructor Guide

## Module 9: Concatenation in MATLAB

**Prerequisite Domain Knowledge:** Matrices

**Expected Completion Time:** 60 minutes

**Table of Contents**



## Solving a Block Puzzle with Real Pictures

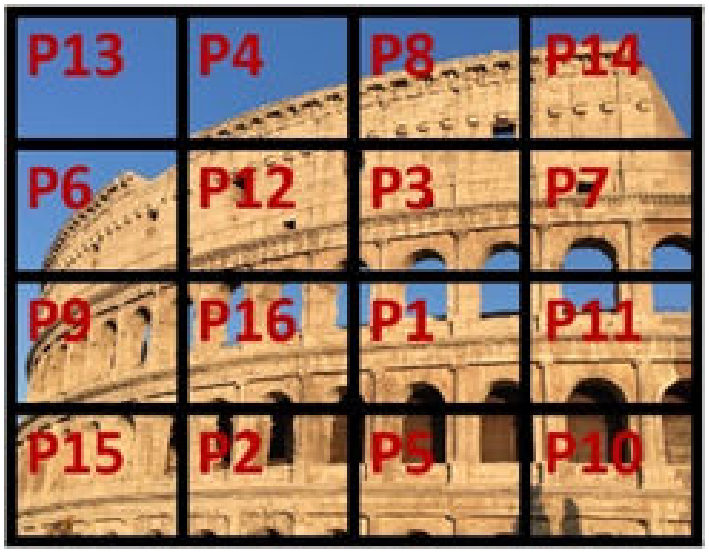
*Expected Duration: 10 minutes*

### Learning Objectives

An image can be created by linking several smaller images one after the other.

### Materials

- Printed copies of multiple images from the **Image Library** folder:
- londoneye.jpg, boat.jpg, store.jpg, church.jpg, colosseum.jpg, travi.jpg, etc.
- Cut images into 16 pieces. Number each piece but in a random order. The ‘P’ in each box stands for piece, so piece number 13, piece number 4, and so on.




- Each group gets their own picture puzzle.

## Steps

Divide the students in 4 – 5 groups and ask the groups to spread out.

Hand out one puzzle set to each group.

Students need to complete the image by solving the puzzle. Stress that this is not a competition, but rather a learning exercise.

They will end up creating a matrix with the small images - 4 rows and 4 columns.

Point out that each of the smaller pieces of the puzzle is an image by itself. So it is a matrix with rows and columns of pixels and has RGB values like any other image.


We were able to create a bigger image by linking another piece either next to it or under it.


Linking an image/matrix at the end of another image/matrix (like a chain) to create a bigger image/matrix is called **concatenation**.

# Matrix Concatenation in MATLAB

*Expected Duration: 20 minutes*

## Learning Objectives

- Rules of matrix concatenation in MATLAB

## Materials

- MATLAB®

## Steps

Tell students that before we start concatenating images with hundreds of rows of columns, let's find out the rules of matrix concatenation using smaller matrices. The same rules will then apply to images.

Instruct students to clean up the Command Window, type `>> clc, clear, close all` to clean up. Then create a new matrix:

```
>> A = [1 3; 7 5]

A =

    1     3
    7     5
```

Refresh their memory by pointing out the syntax that if you want a number ***next to*** another number you use ***space*** and if you want a number ***in the next line*** you use ***semi-colon***.

Create another matrix B:

```
>> B = [4 1; 9 2]

B =

    4     1
    9     2
```

Now ask:

- *What do we do to concatenate two matrices next to each other?*
- Lead students to the point that this will be similar to placing two number next to each other by using **space.**

```
>> C = [A B]

C =

    1     3     4     1
    7     5     9     2
```

- *How might we concatenate one matrix under another?*
- Students should mention that it is the same method, only this time using **semi-colon**.

```
>> D = [A;B]

D =

     1     3
     7     5
     4     1
     9     2
```

That was simple but there are a couple of rules to keep in mind. Create a row vector R:

```
>> R = [1 5]

R =

     1     5
```

Try to concatenate A and R row-wise (next to each other)

```
>> X = [A R]
```

```
>> X = [ A R ]
Error using horzcat
Dimensions of matrices being concatenated are not consistent.
```

This gives an error because the result of the above would have two empty spaces in the matrix, which is not valid. A matrix needs to have the same number of row for every column so that it looks like a box.

| | | | |
|---|---|---|---|
| 1 | 3 | 1 | 5 |
| 7 | 5 | ✖ | ✖ |

However, we can concatenate A and R column-wise since they have the same number of columns:

```
>> X = [ A ; R ]
X =
     1     3
     7     5
     1     5
```

***So for row-wise concatenation, the matrices need to have the same number of rows, and for column-wise concatenation they need to have the same number of columns - otherwise the dimensions will not be consistent.***

Create a column vector C:

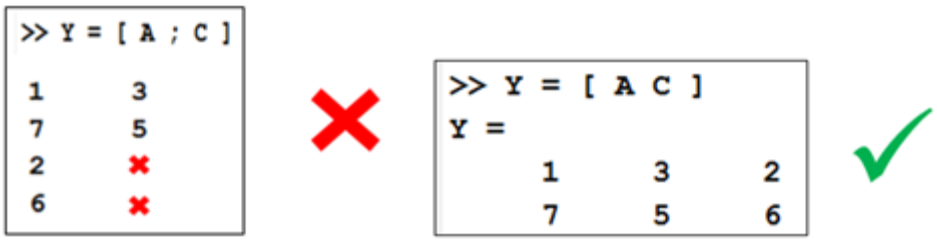
```
>> C = [2; 6]

C =

    2
    6
```

Ask the students what would be the right way to concatenate A and C matrices.

Answer:



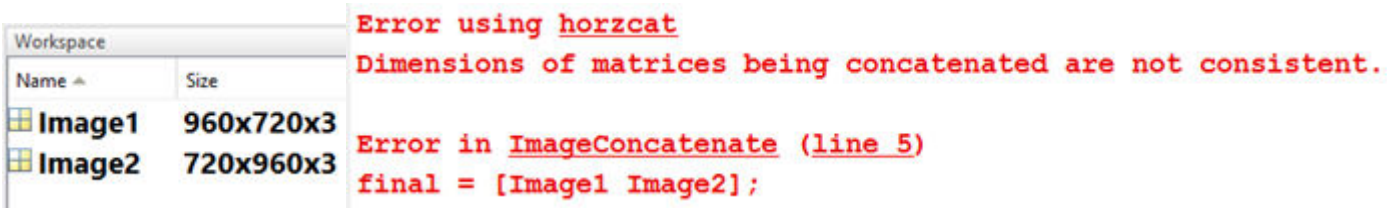

Create different sized matrices and practice some indexing with the students. Example: concatenating 3x3 matrix with the 3x2 matrix (row-wise), 2x3 matrix (column-wise), 2x2 matrix (dimensions don't match, so can't concatenate).

Ask the students to open the following script:

```
open 'ImageConcatenate.mlx'
```

This script first cleans up, then reads in two images from the **Image Library** folder and then tries to concatenate them.

Run the script. This should fail because the row numbers and column numbers of both the images are different sizes.



Resize both images to be the same size.

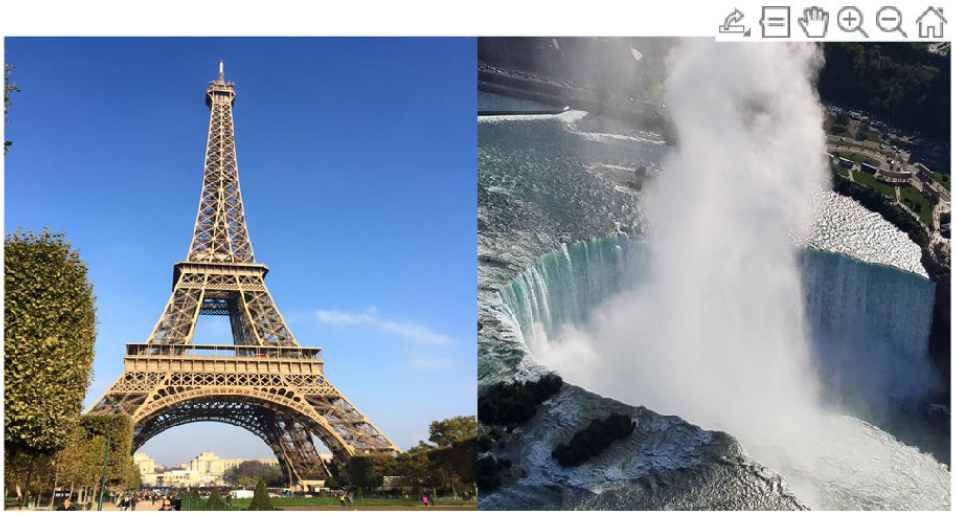
### Read in first image

```
Image1 = imread('eiffel.jpg');
Image1 = imresize(Image1,[500 500]);
```

### Read in second image

```
Image2 = imread('niagara.jpg');
Image2 = imresize(Image2,[500 500]);
```

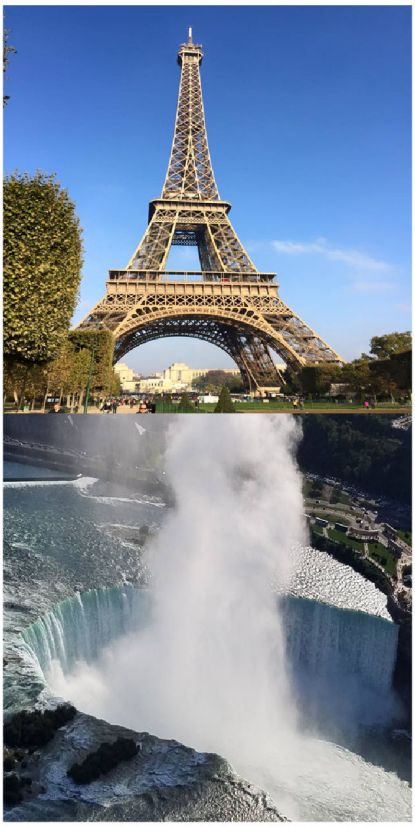
Run the script again to see the images placed next to each other as one big image.



Change line 8 as follows:

```
final = [Image1 ; Image2];
```

Run the script again to see one image under another as expected:



## Solving Block Puzzles in MATLAB

*Expected Duration: 30 minutes*

### Learning Objectives

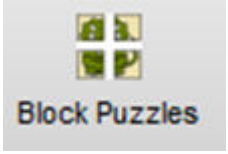
- Practice matrix concatenation

### Materials

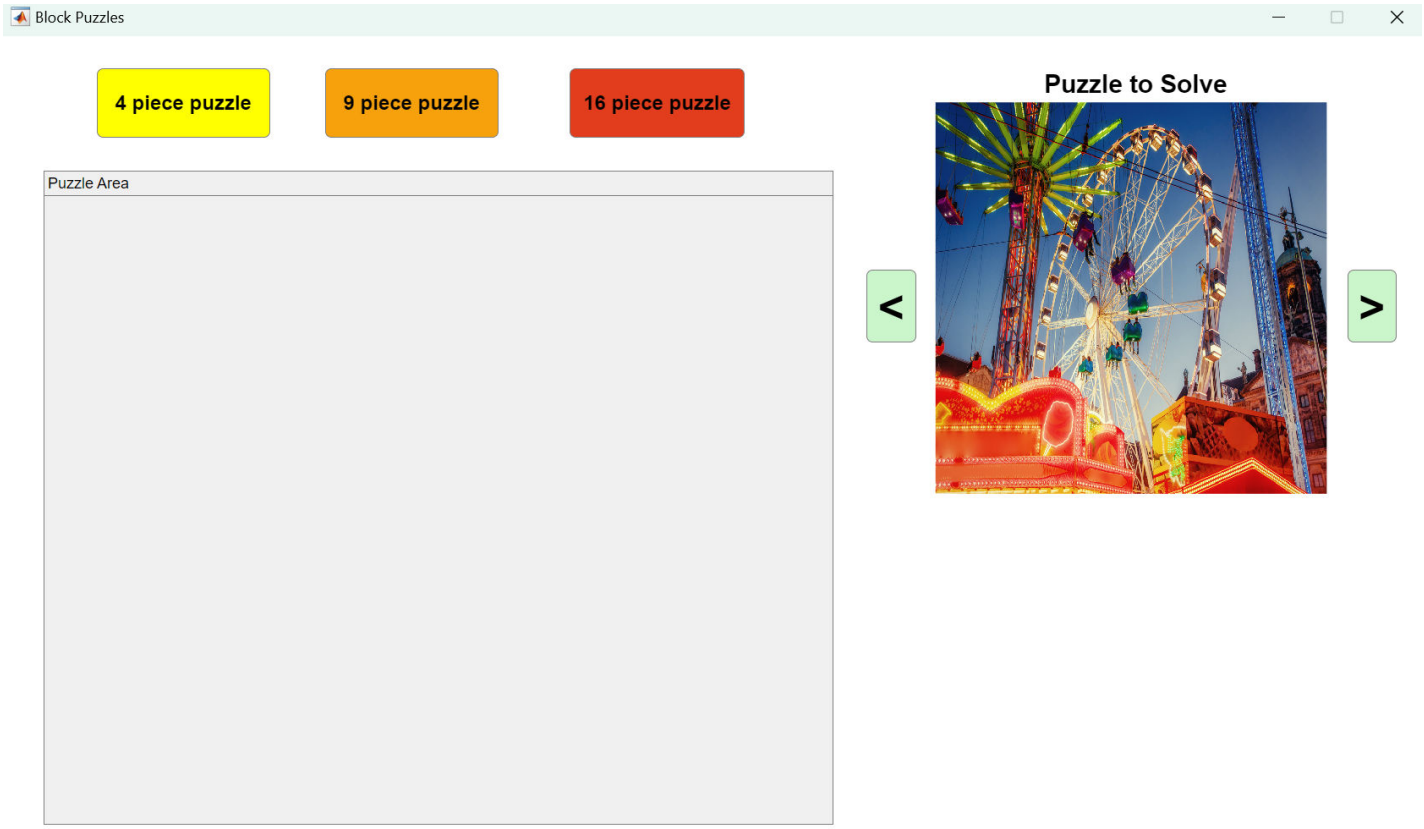
- MATLAB®

### Steps

Tell the students that they will now use their knowledge of concatenating arrays to solve a block puzzle using MATLAB.  Instruct the students to navigate the the **APPS** tab and click on the Block Puzzles App:




- This will open up the BlockPuzzles app:

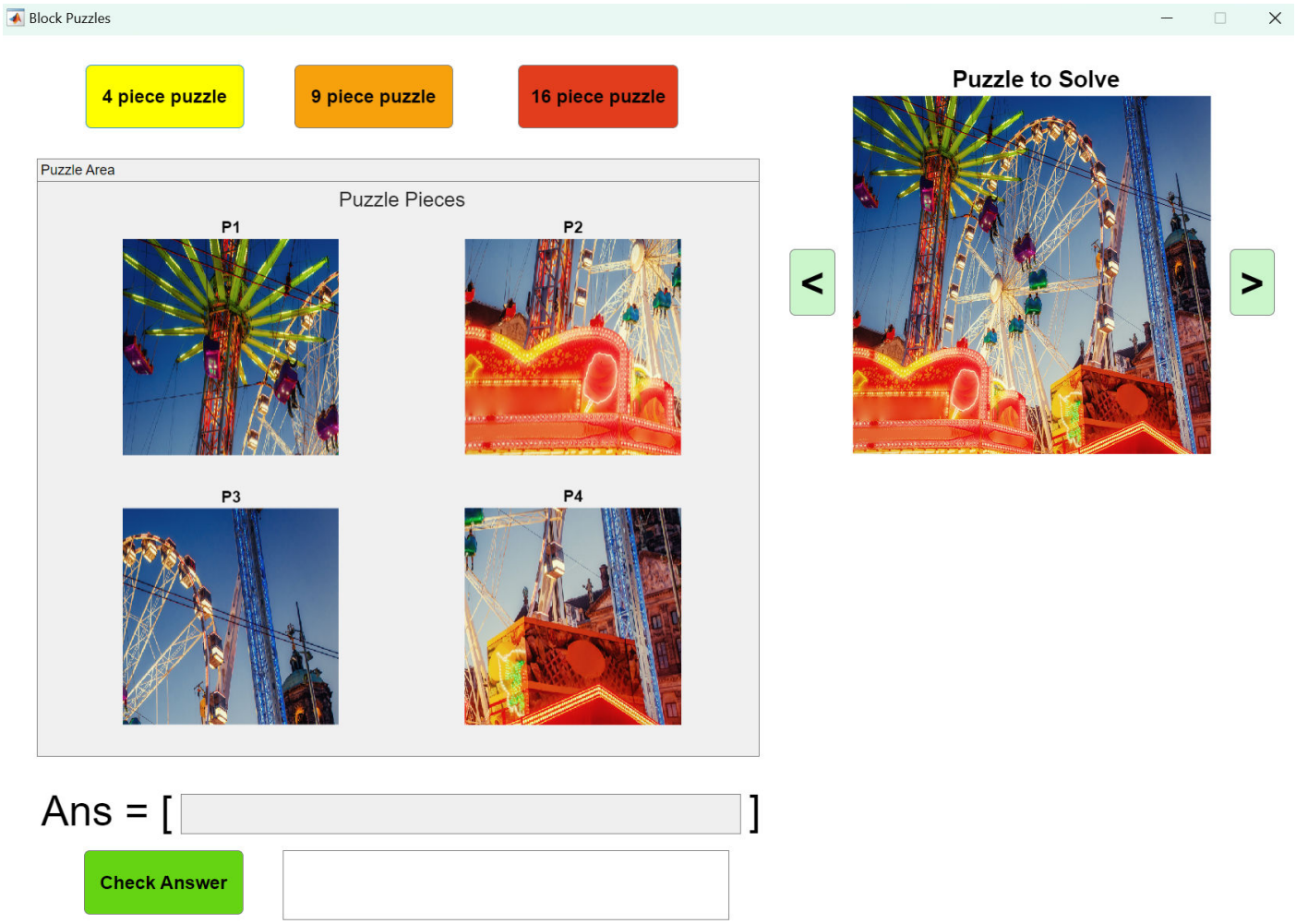



Tell the students that we will now solve some block puzzles in MATLAB and continue practicing concatenation.

Students can choose a puzzle image to solve by click on the green arrow buttons < > next to the image.

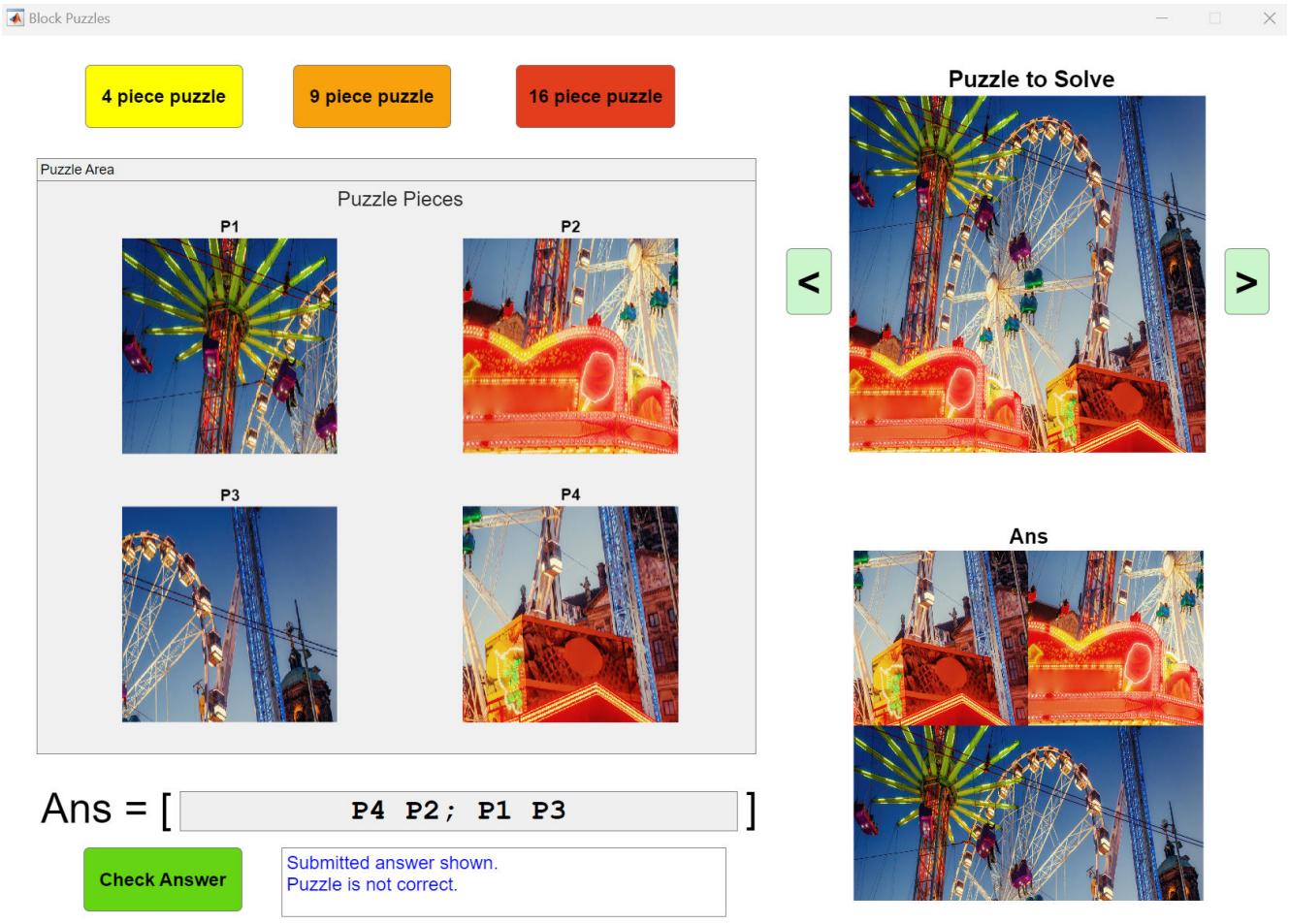
They can select either a 4 piece, 9 piece or 16 piece puzzle:


Demonstrate one puzzle completion for the students. Click on 4-piece puzzle as an example:

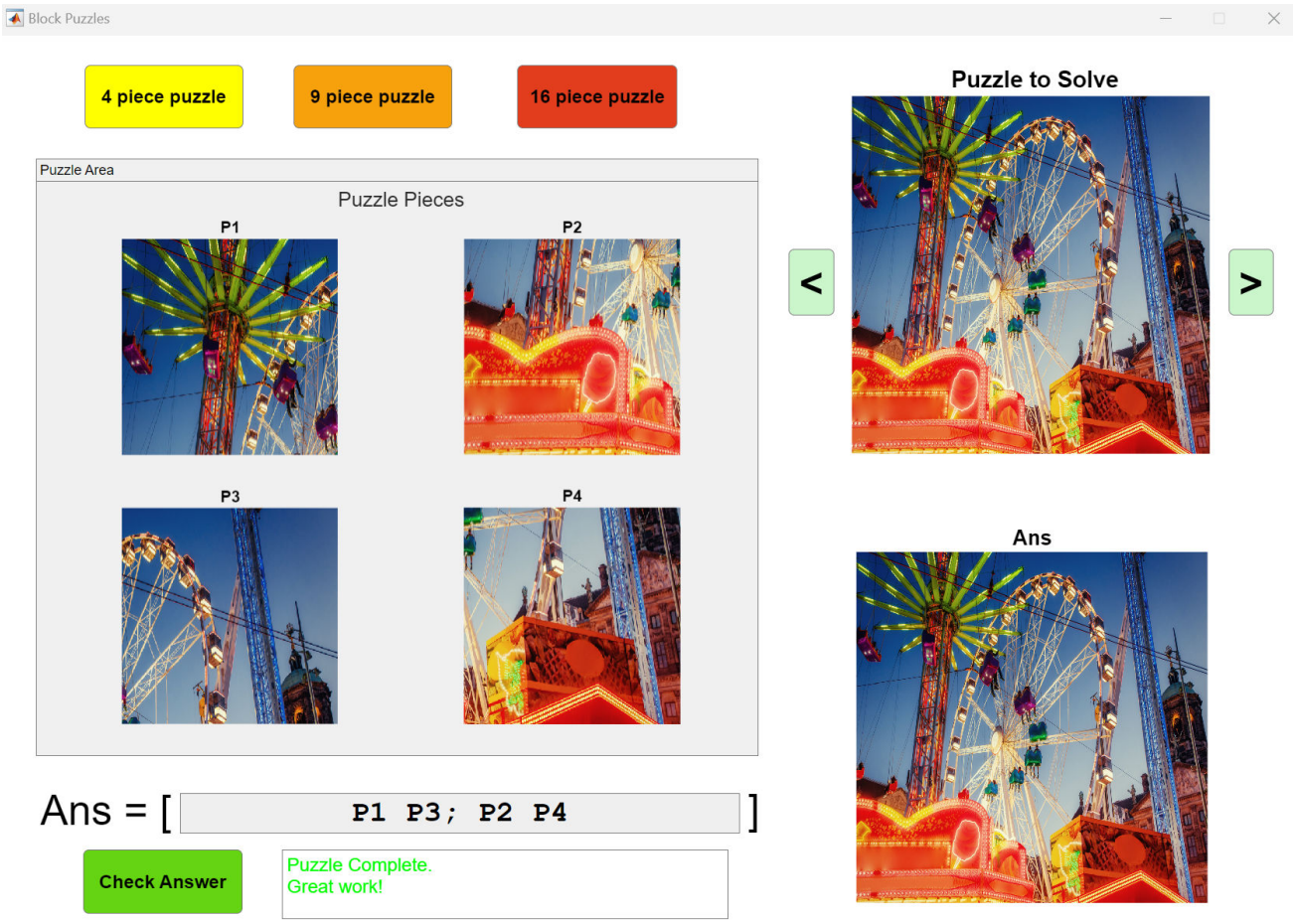



In the "Ans" box, start trying out the solution of the puzzle inside the square brackets. Remind them the square brackets are the correct syntax when concatenating matrices.

Click the green "Check Answer" box shows the progress of puzzle solution.

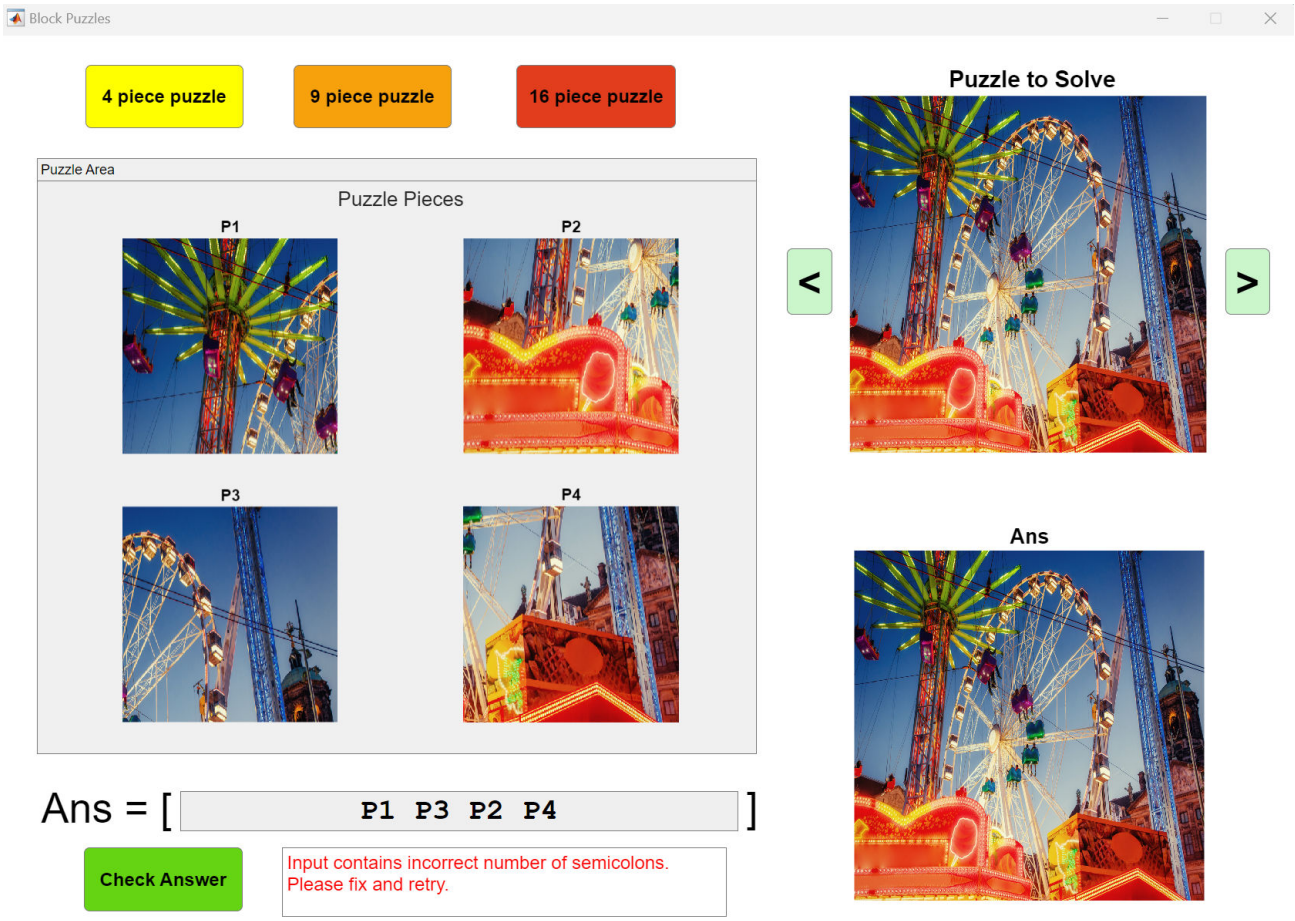



Remind the students that space concatenates the image row-wise and semi-colon concatenates the image column-wise.




Students can change the puzzle image and the number of puzzle pieces any time they desire. The app generates a new combination of the puzzle pieces each time.

Error messages below the "Ans" edit box should help guide the students on their syntax.




Give the students 15-20 minutes to solve different puzzles. At the end of this time, ask:

- *What was most interesting about solving the block puzzles with MATLAB?*
- Accept all reasonable answers.
- *What was most challenging about solving the block puzzles with MATLAB?*
- Accept all reasonable answers.
- *What were the differences between solving the block puzzles with MATLAB and a physical puzzle?*
- Accept all reasonable answers. Students may also give reasons as to which method was more successful and preferred.