3: Pentominoes

Due: Noon September 24

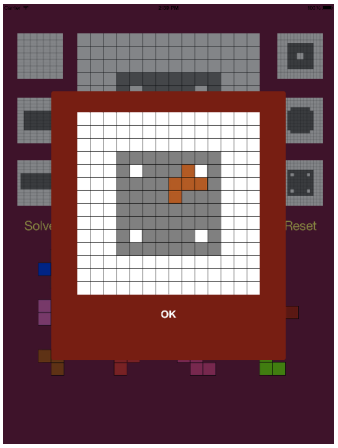
Description

In this assignment you will finish the Pentominoes game by adding gesture handling for moving, rotating and flipping the pieces, device rotation and also adding hints.

Walk Through

This walk-through will only describe the overall architecture/requirements of your solution.

1. Add three gesture recognizers to each playing piece: single tap, double tap, and a pan. A single tap rotates a piece 90 degrees clockwise. A double tap flips a piece over, giving its mirror image. A pan gesture moves the piece. The single tap recognizer needs the double tap recognizer to fail before it should accept a gesture. The single and double tap gestures should only be recognized when the playing piece is on the game board. The pan gesture should always be recognized.
2. Add support for rotation. Most of this is already handled for you by Auto Layout and (hopefully) the way you place the playing pieces in their initial position. However, you need to account for playing pieces that can either be on the game board or in their initial position. Pieces on the game board should remain there. Pieces in their initial position might need to be repositioned as necessary due to the size change.
3. The movement of pieces should always feel natural and give a good user experience. In particular:
   3.1. Pieces should never “jump” from one position to another. They should be animated.
   3.2. The size of a piece should be scaled up slightly while the piece is being moved.
   3.3. If a piece is moved to a position off the board it should return to its “home” position on the main view with any transforms removed.
   3.4. When a piece is placed on the board it should “snap” to the grid, lining up horizontally and vertically. This should be an animated action that happens when the gesture ends. Note that you may also need to snap a piece after a rotation.
4. Add a hints button that modally presents a simple scene (view controller) giving hints about the solution to the current board. The button should be disabled for the blank board. Otherwise, the first time the button is pressed the presented view should show the current puzzle with one piece in its correct position (disregarding any pieces the user might currently have on the board). Each successive hint (tapping the hint button again) will add a piece to the solution. I.e., the third time the user asks for a hint three pieces will be displayed in their correct position. If a user changes the game board then hints reset so the next time the hint button is pressed only one piece will be shown initially.



Testing

Be sure to test your app thoroughly. Ensure that rotations and flips all perform in a consistent and uniform manner, and that the Solve and Reset actions still work properly, regardless of where pieces may currently be. All interaction with the game should provide a good user experience. Device rotations should not reset the game, etc.

Hints

1. When moving a piece onto the board you should add it as a subview of the board. Then you can use its frame to snap it into place. Use the gesture recognizer's state to check for the end of the gesture (`UIGestureRecognizerState.Ended`). This type of gesture is *continuous*, meaning that the action method is fired every time a touch event (that is part of the gesture) is detected. The action method for the pan gesture should check for the state of the gesture (began, changed, ended, cancelled) and perform the appropriate actions.
2. To provide a consistent animation of the pieces (rotations & flips), you might find it necessary to keep track of the current rotation and whether the piece has been flipped or not.
3. Rotations may not appear to be centered on the piece. Don't worry about this.
4. Rotations should always appear to be clockwise and flips should mostly appear to flip along the vertical access. I say "mostly" because you will notice a situation in which a rotated piece will animate a flip in an awkward manner. No simple way to remedy this - iOS determines how it animates from an initial to final state.
5. You do not have to check for and/or prevent pieces from overlapping when placed on the board.
6. Your modally presented view controller will need an instance variable referring to the instance of the model. Set this up in `prepareForSegue`. You do not want to create multiple instances of the model. It should be a singleton class shared by the viewControllers
7. A segue creates a new instance of a view controller each time it is called. When the view controller is dismissed, this instance is deallocated. Do not expect any persistence between instances of the presented view controllers.
8. Use any of the three methods given in class for dismissing a view controller.

Troubleshooting

1. Tracking when gestures are recognized and what state they are in can be useful.
2. If a view is not responding to touch events be sure that its (and all its superviews') `isUserInteractionEnabled` property is set to `true` (The property defaults to `false` for programmatically created UIImageViews) and be sure that it is within the bounds of its superview.
3. A gesture recognizer must be attached to a view. Don't forget to add them to appropriate views after you create them. Each recognizer can only be attached to a single view so you will need many recognizers (but they can all use the same target & action).

Submission

Your submission should be pushed on the master branch. Be sure to verify that your project builds and remove all cruft.