#WWDC18

A Tour of UICollectionView

Session 225

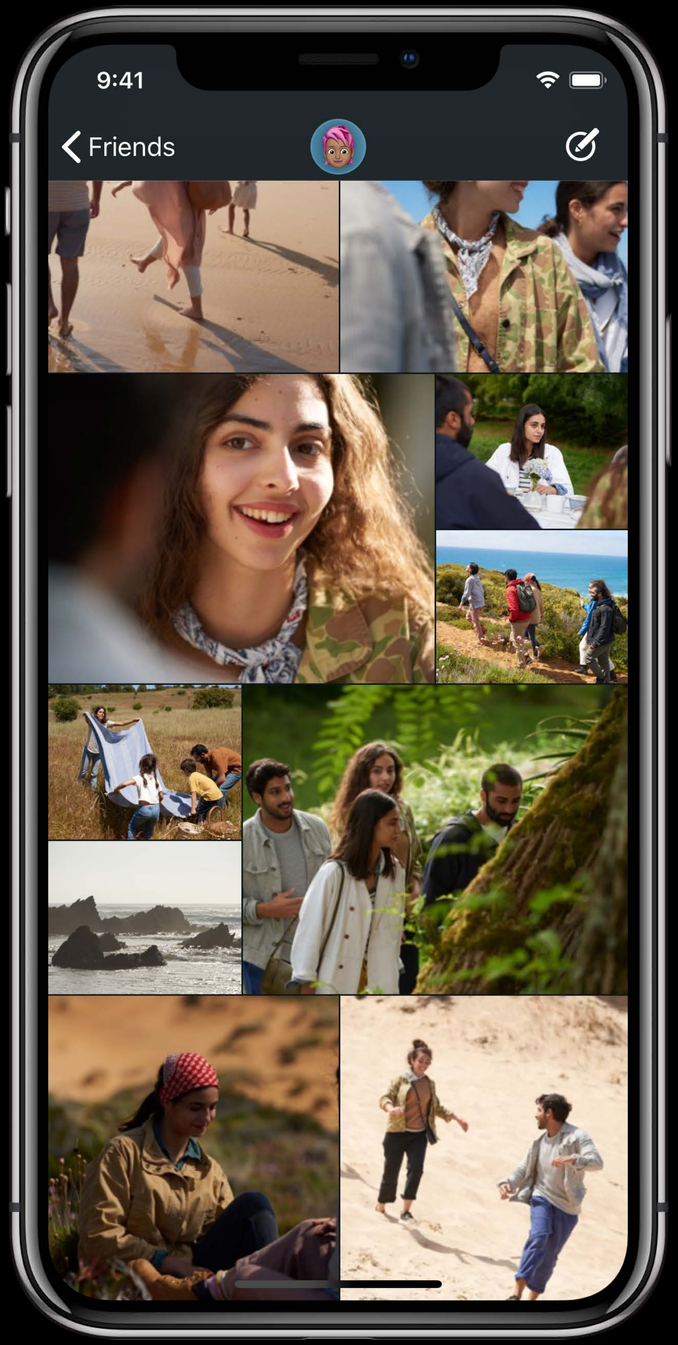
Steve Breen, UIKit Frameworks Engineer
Mohammed Jisrawi, UIKit Frameworks Engineer

Agenda

Build an app

- Layouts
- Updates
- Animations


9:41
Friends
Steve's Feed
Updated Jun 3, 2018
Mohammed's Feed
Updated Jun 2, 2018
Samir's Feed
Updated May 21, 2018
Priyanka's Feed
Updated May 20, 2018


9:41
Friends

Key UICollectionView Concepts

Layout

Data source

Delegate

UICollectionViewLayout

Visual arrangement of content

UICollectionViewLayoutAttributes

- Bounds, center, and frame…

Invalidation

Animate between layouts

UICollectionViewFlowLayout

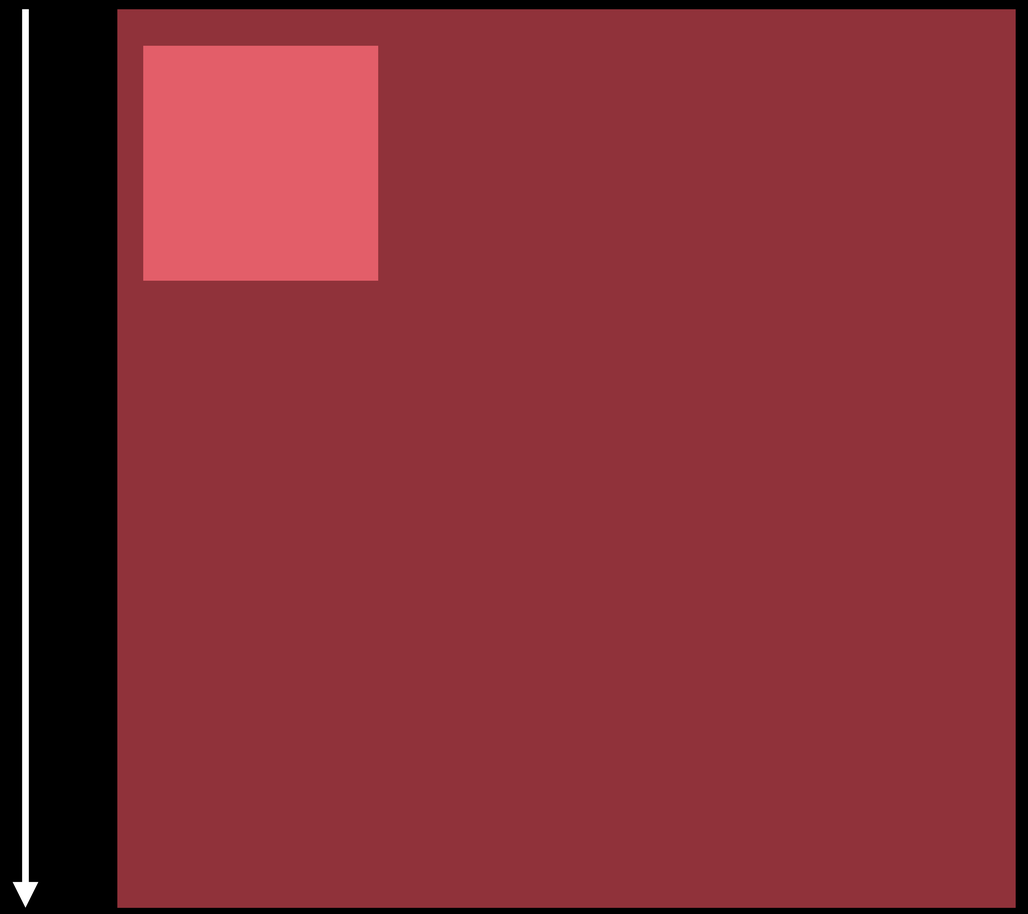
Concrete subclass of UICollectionViewLayout

UICollectionViewDelegateFlowLayout extends UICollectionViewDelegate

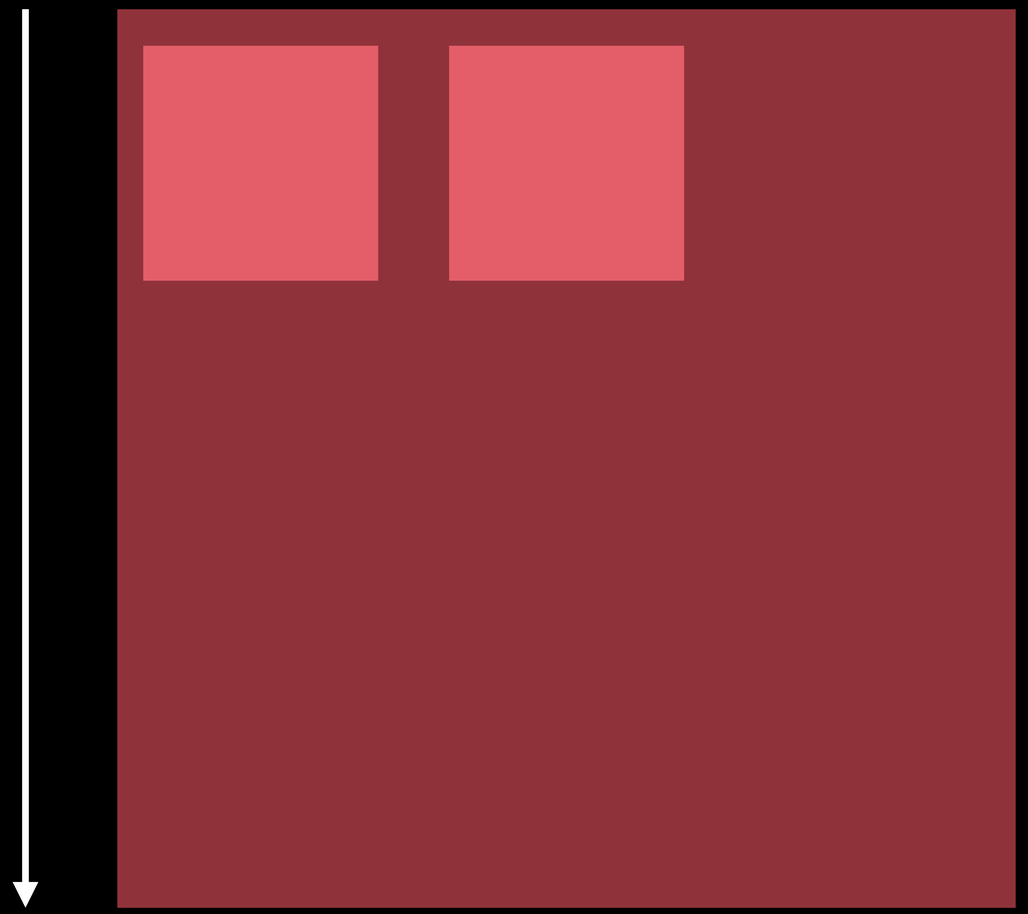
Line-based layout covers a wide range of designs

Flow Layout

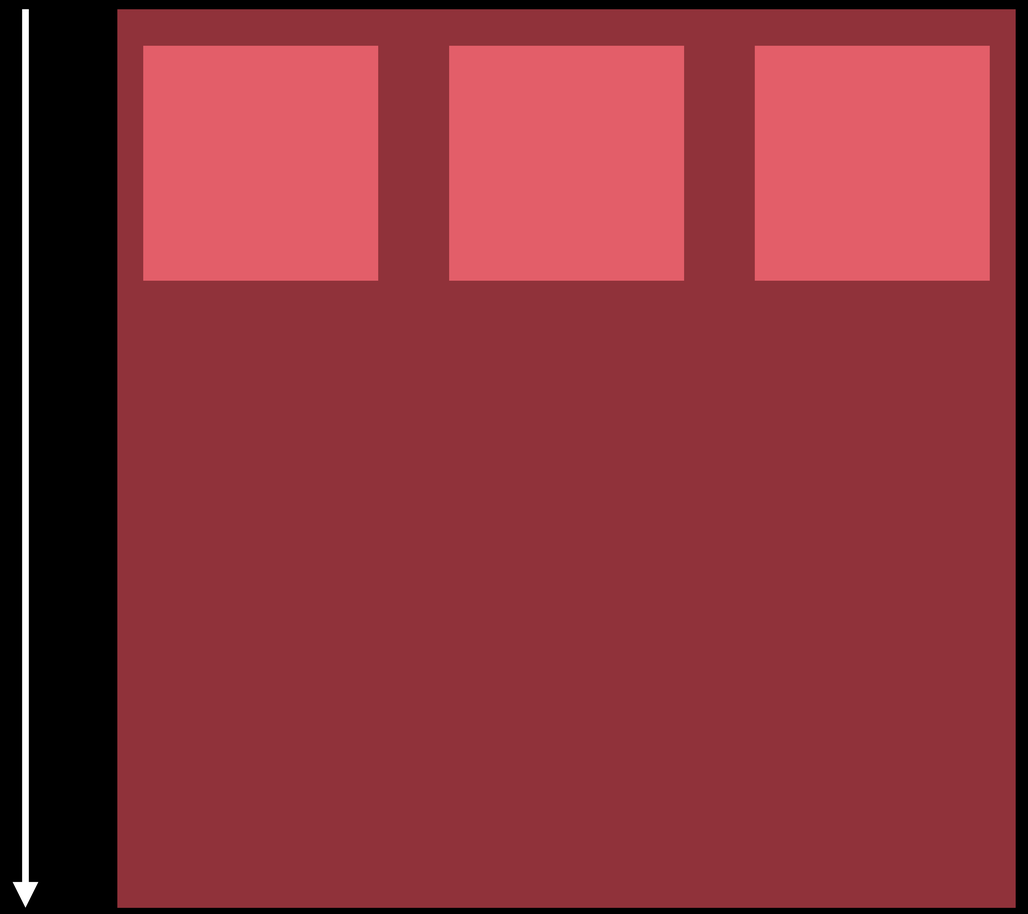
Flow Layout



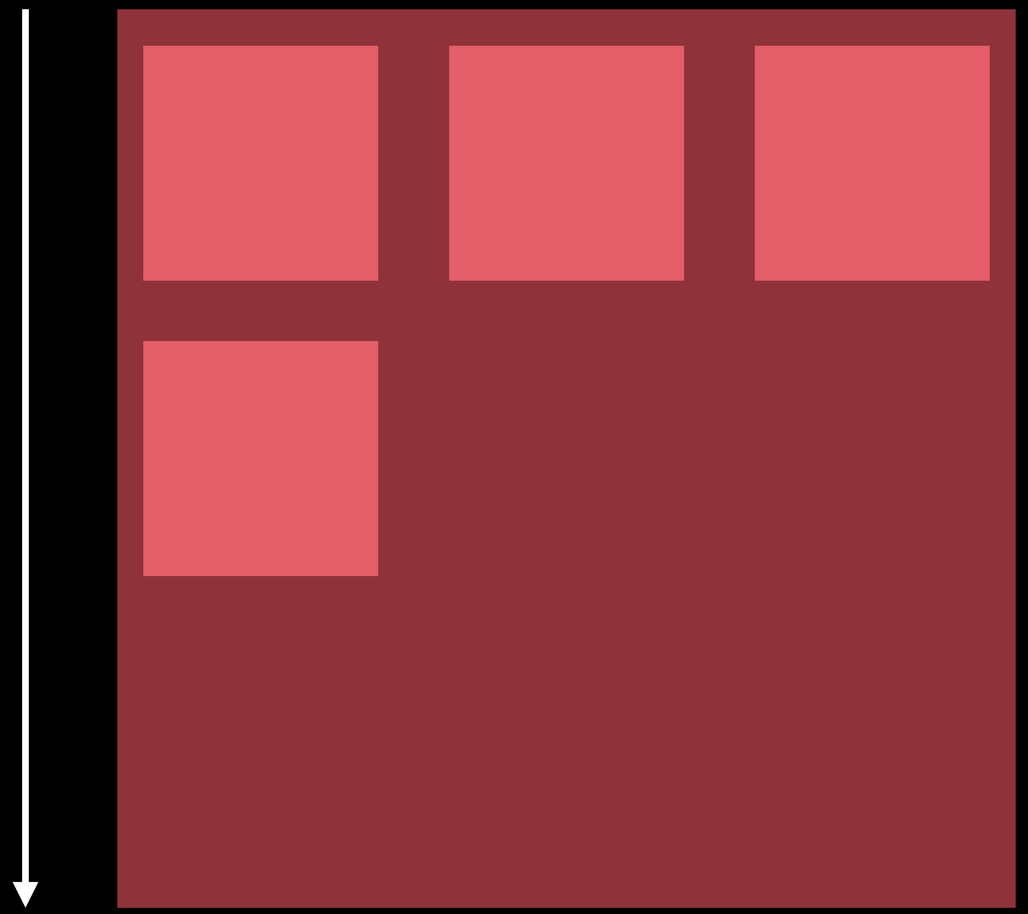
Flow Layout



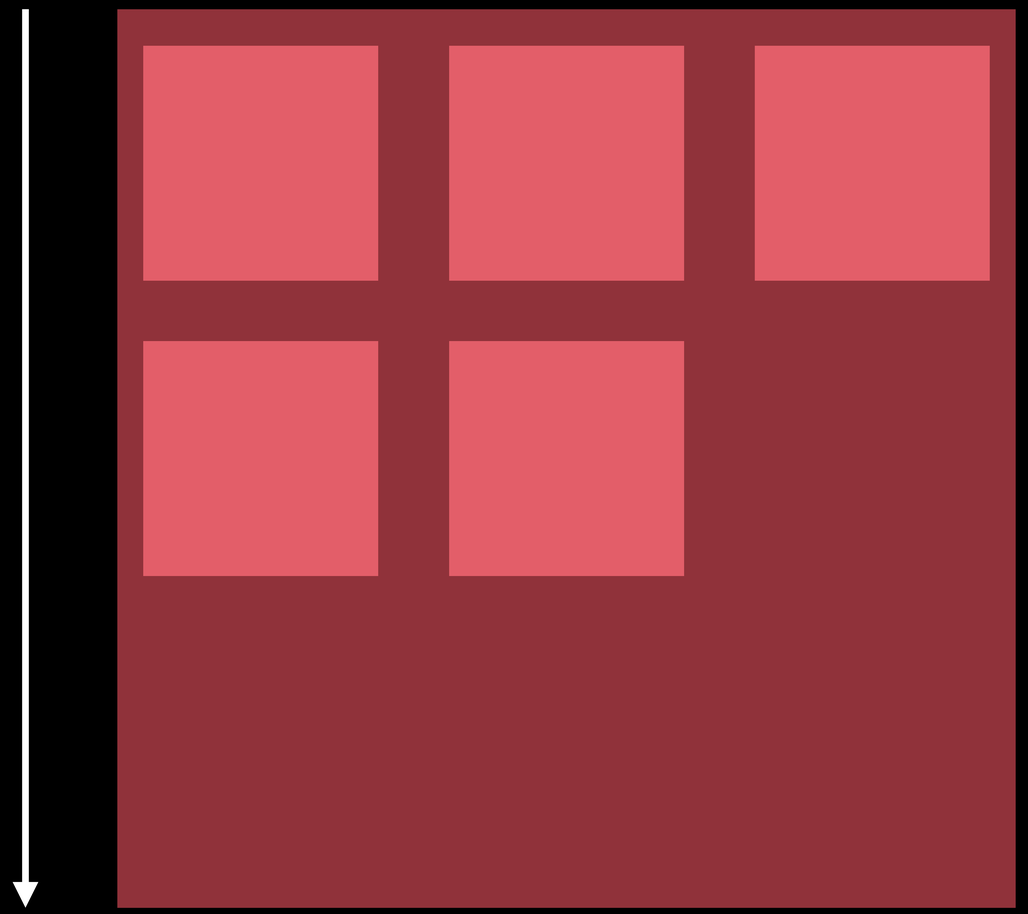
Flow Layout



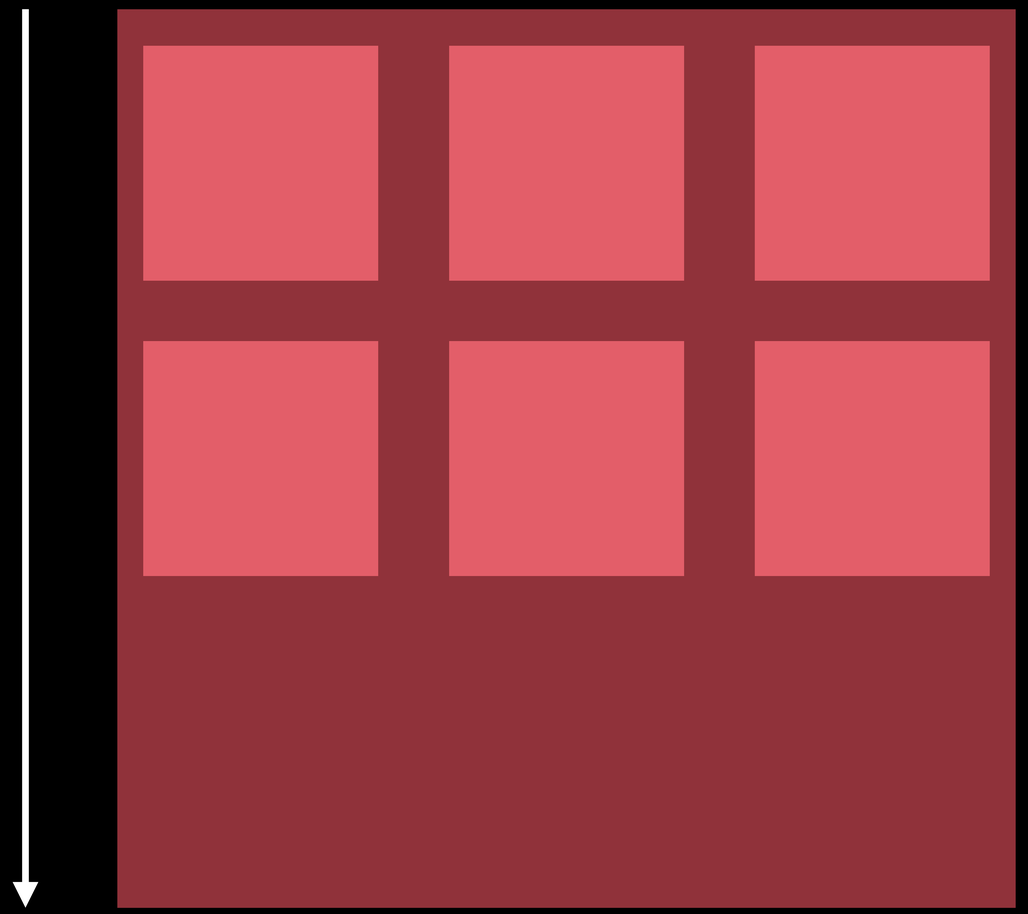
Flow Layout



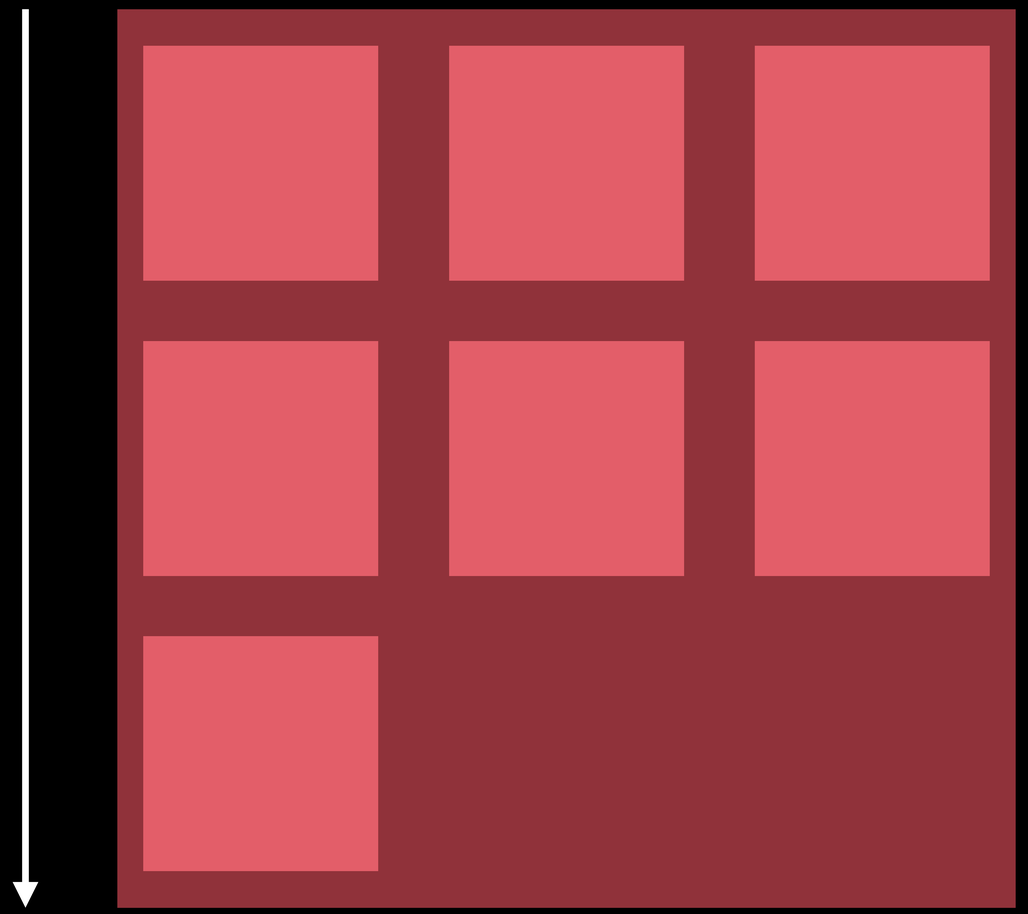
Flow Layout



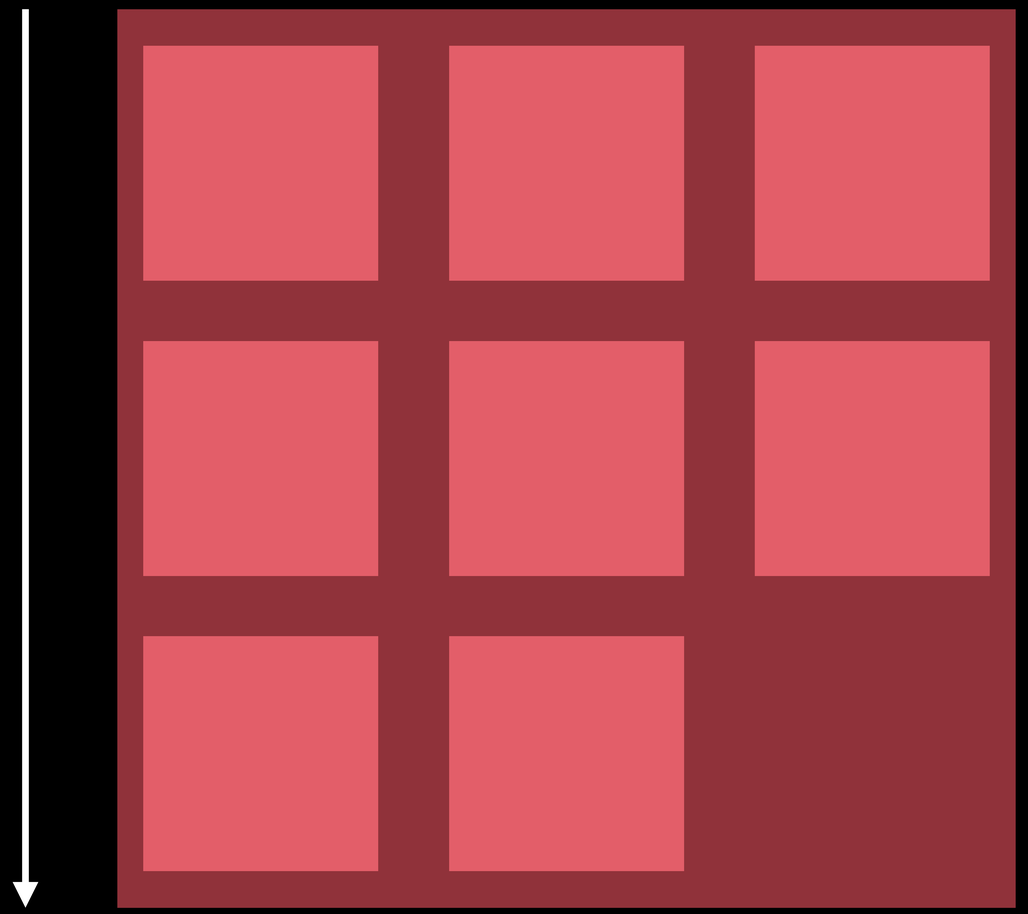
Flow Layout



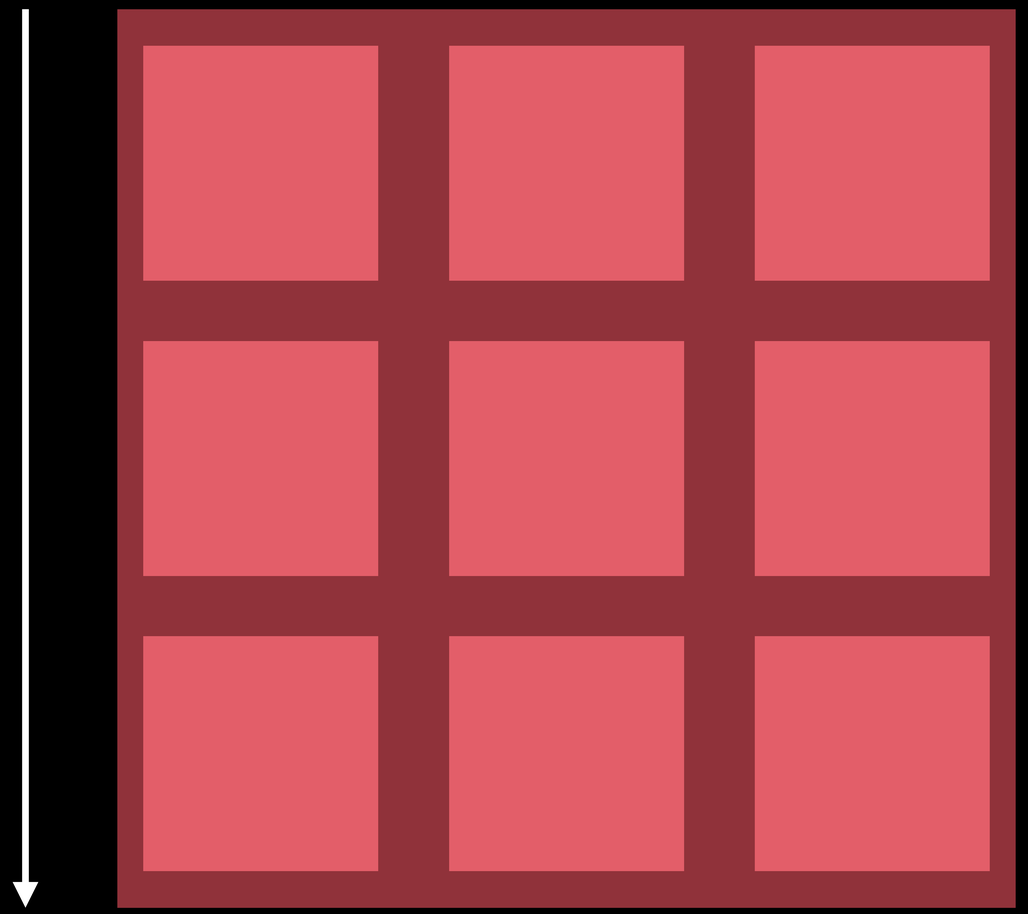
Flow Layout



Flow Layout

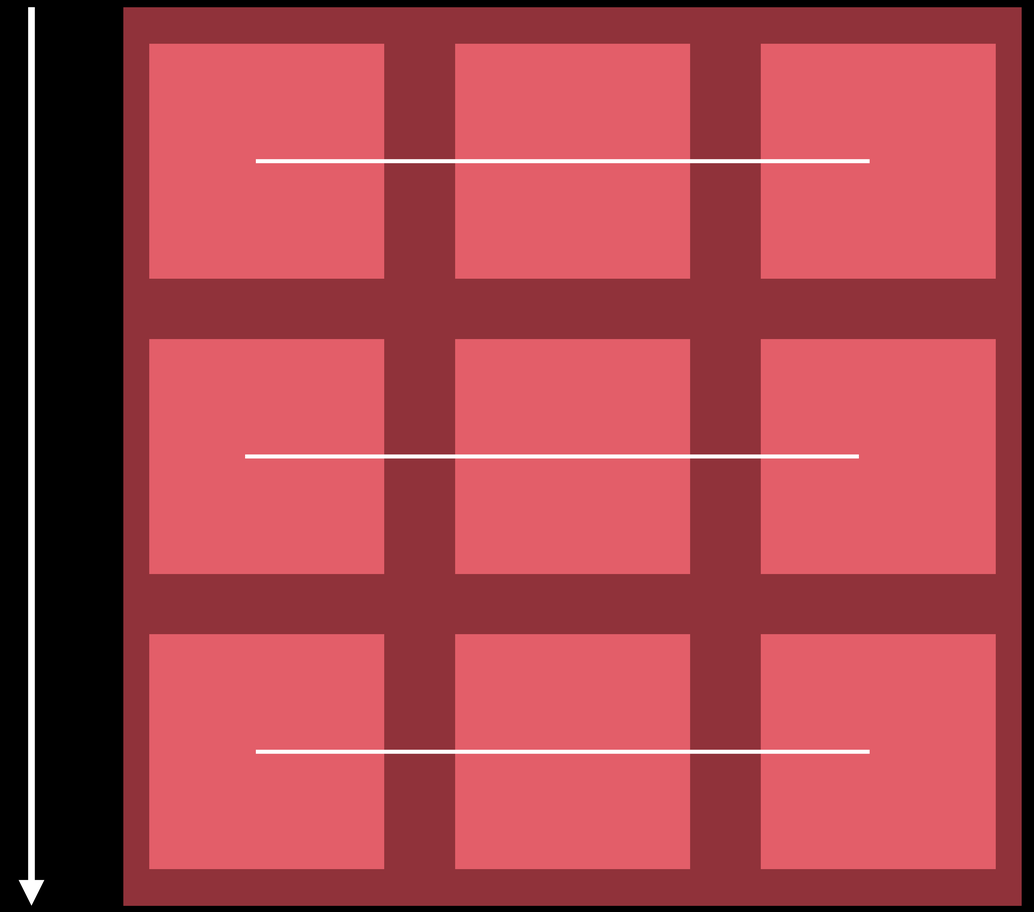


Flow Layout



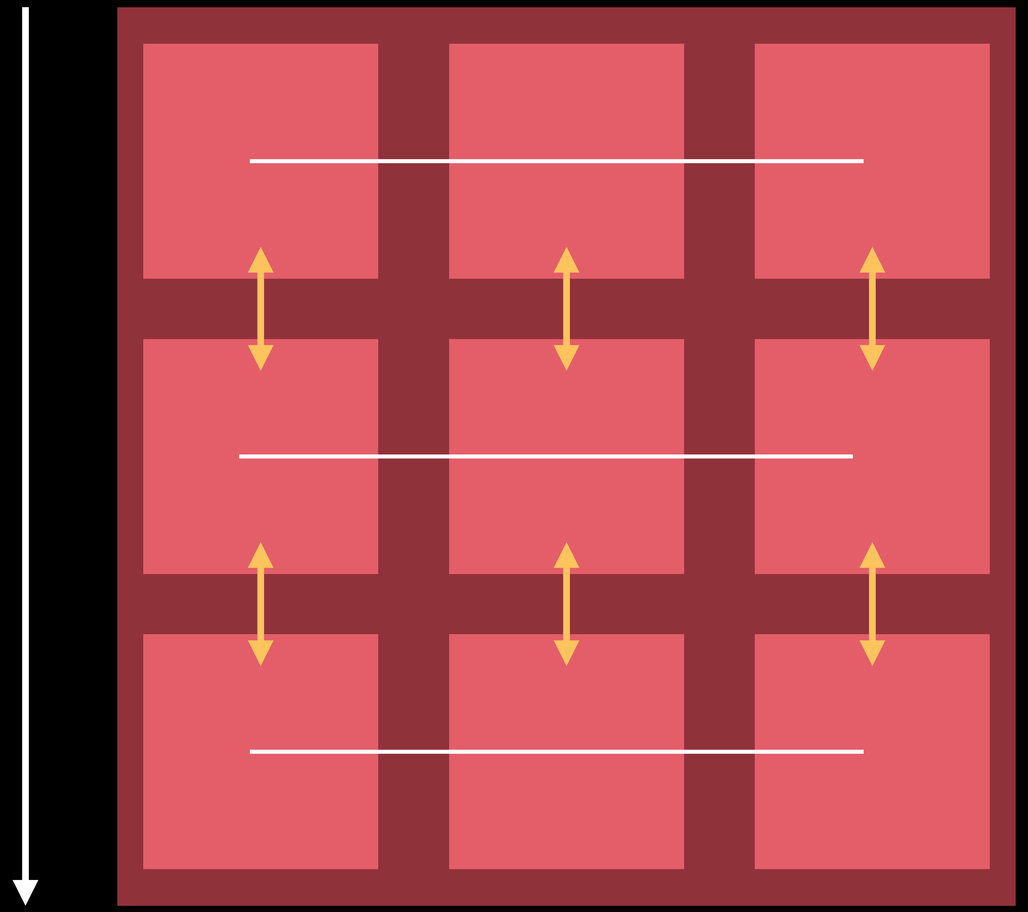
Flow Layout

Line Orientation



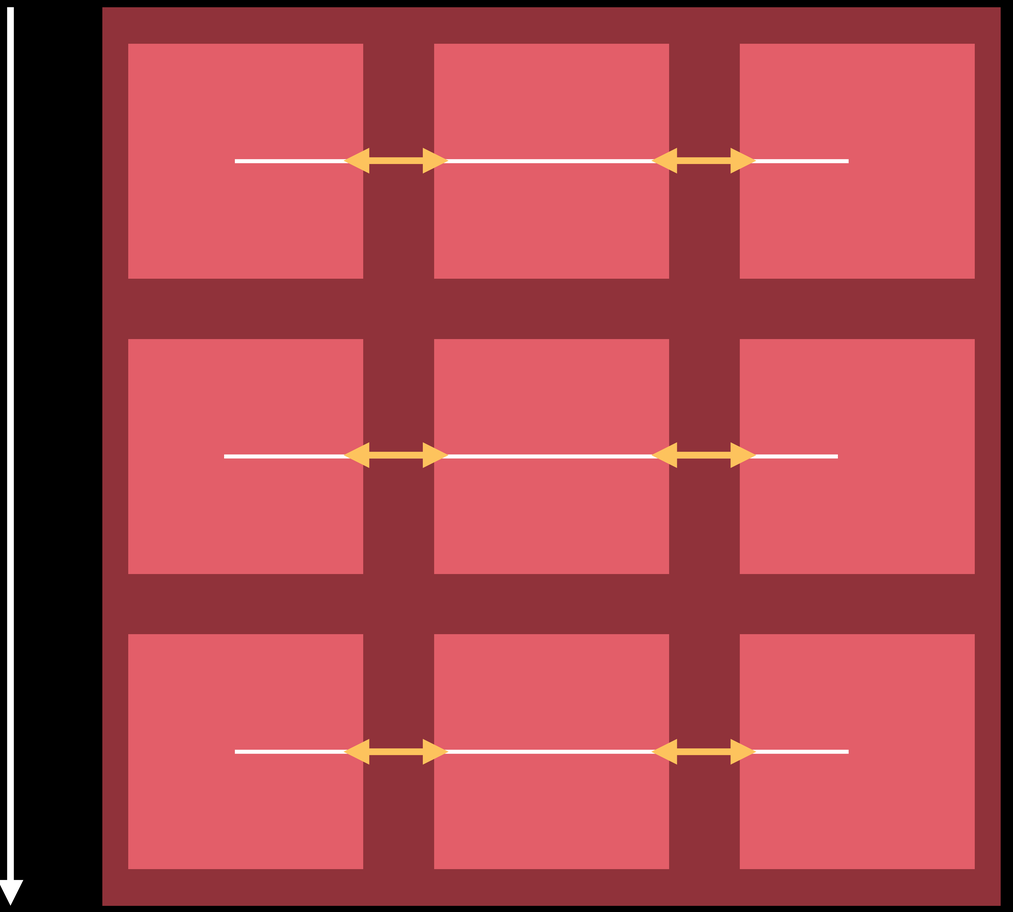
Flow Layout

Line Spacing

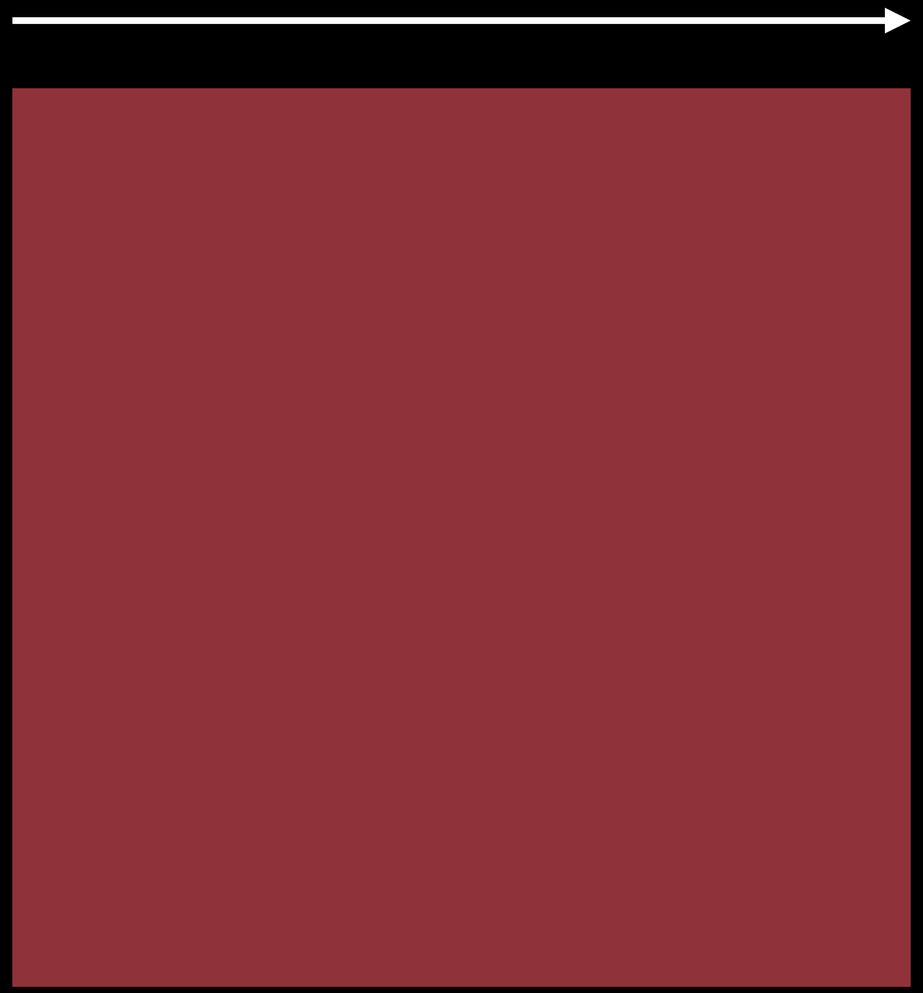


Flow Layout

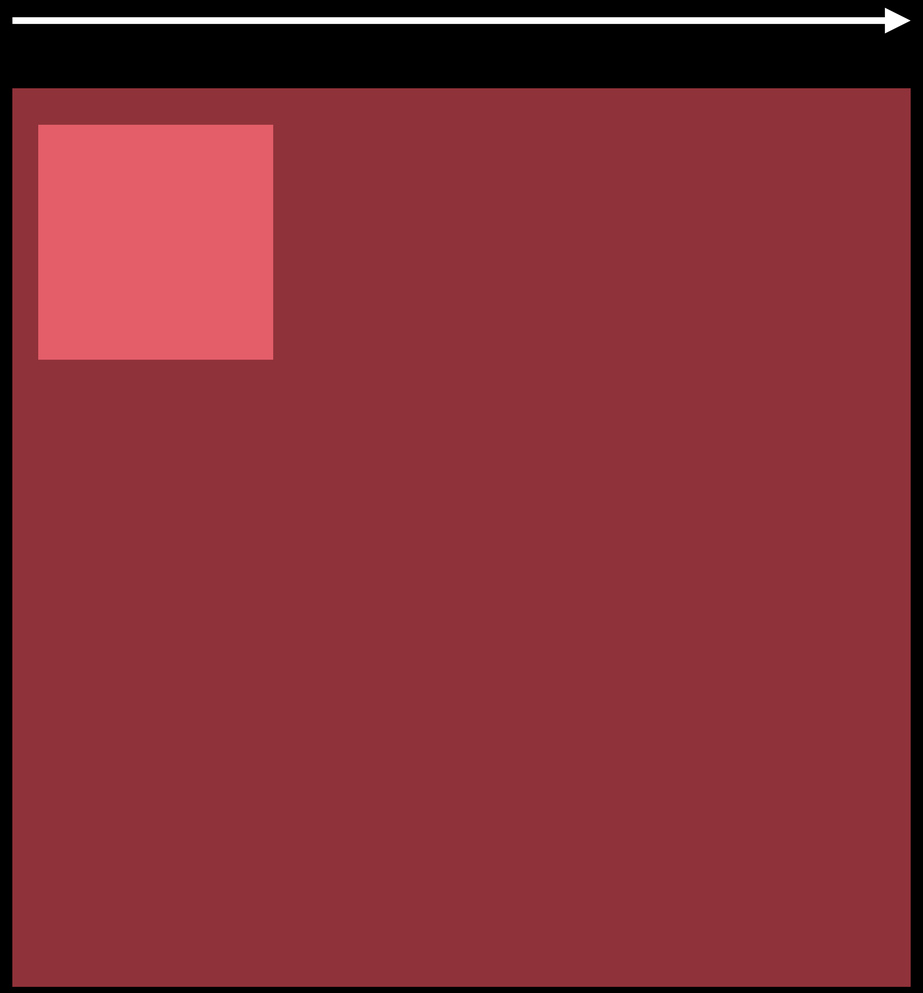
Inter-Item Spacing



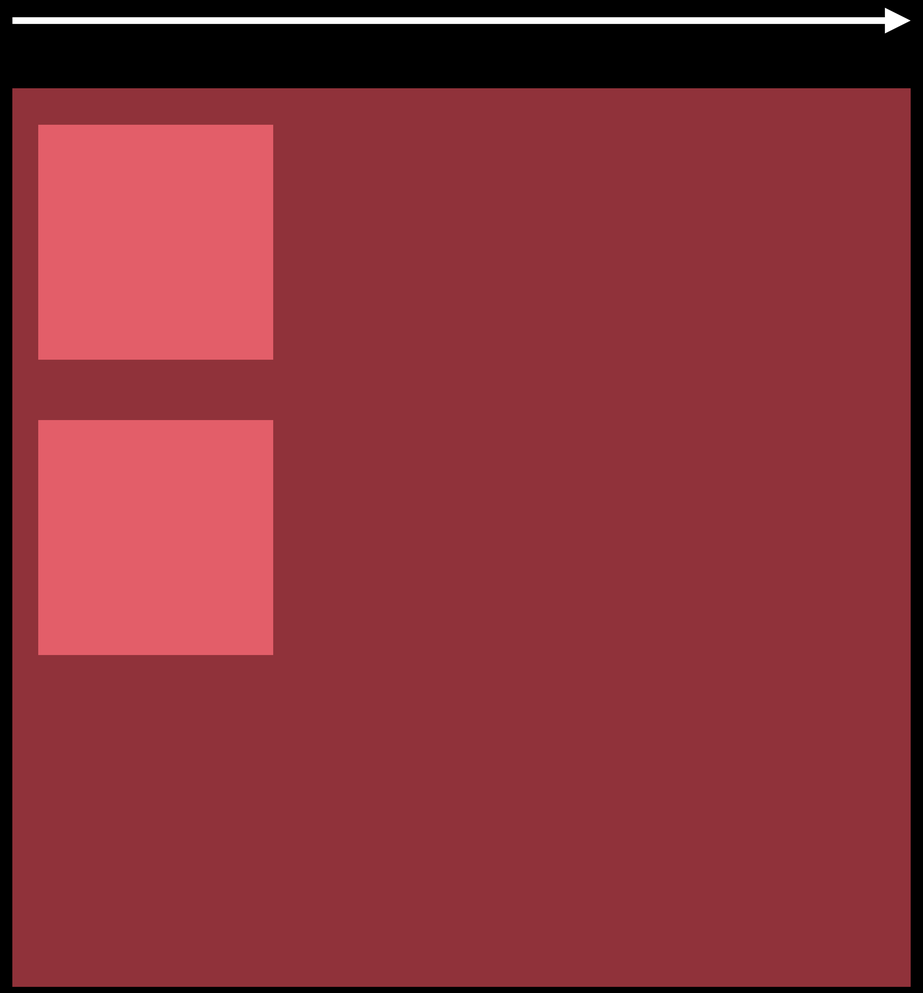
Flow Layout



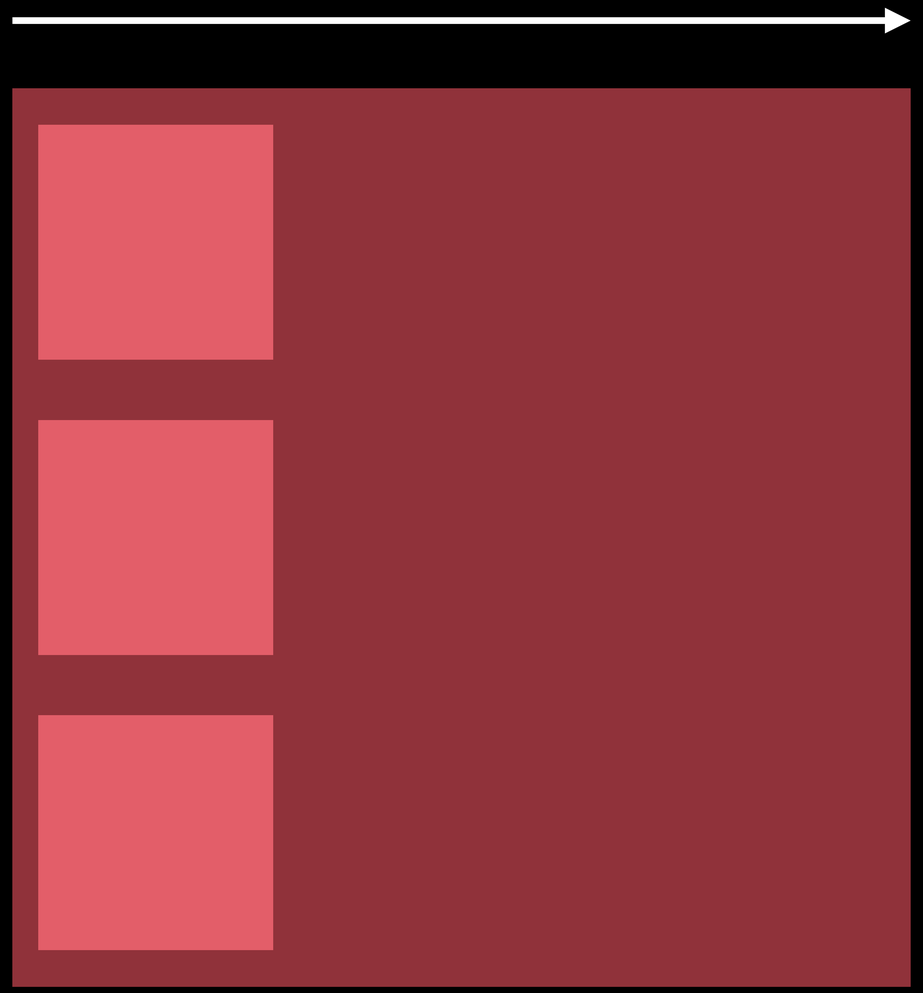
Flow Layout



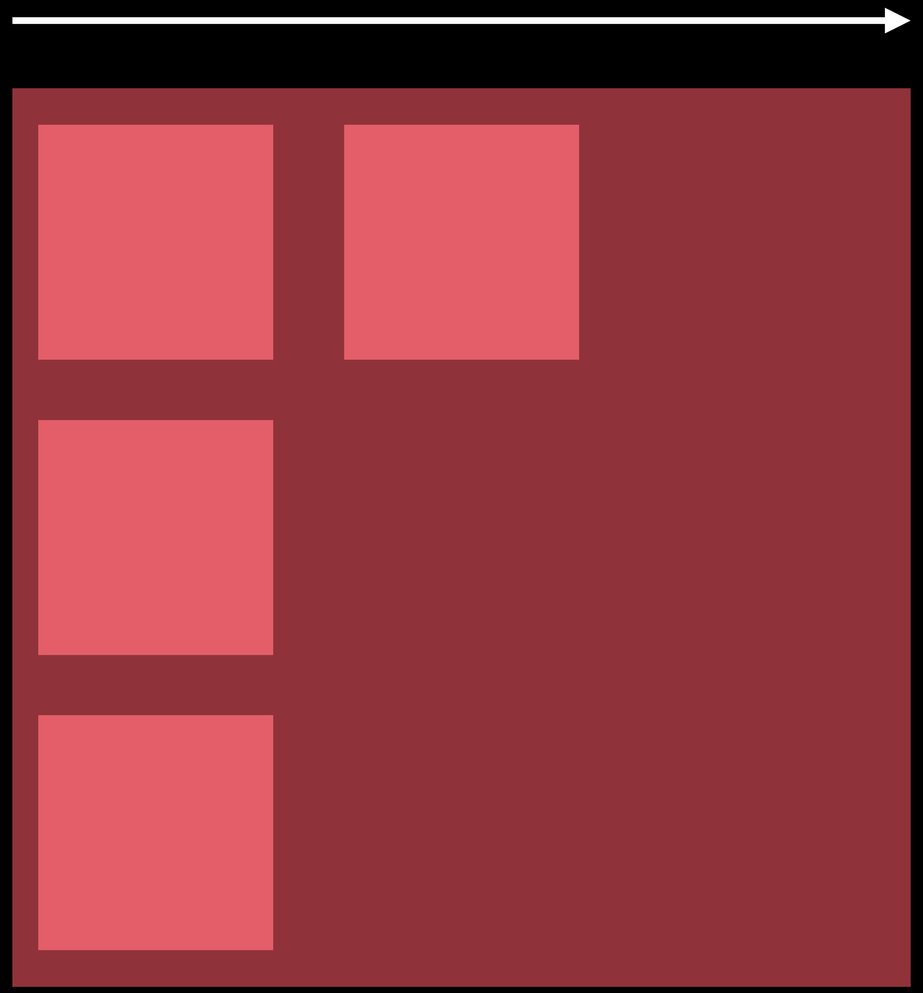
Flow Layout



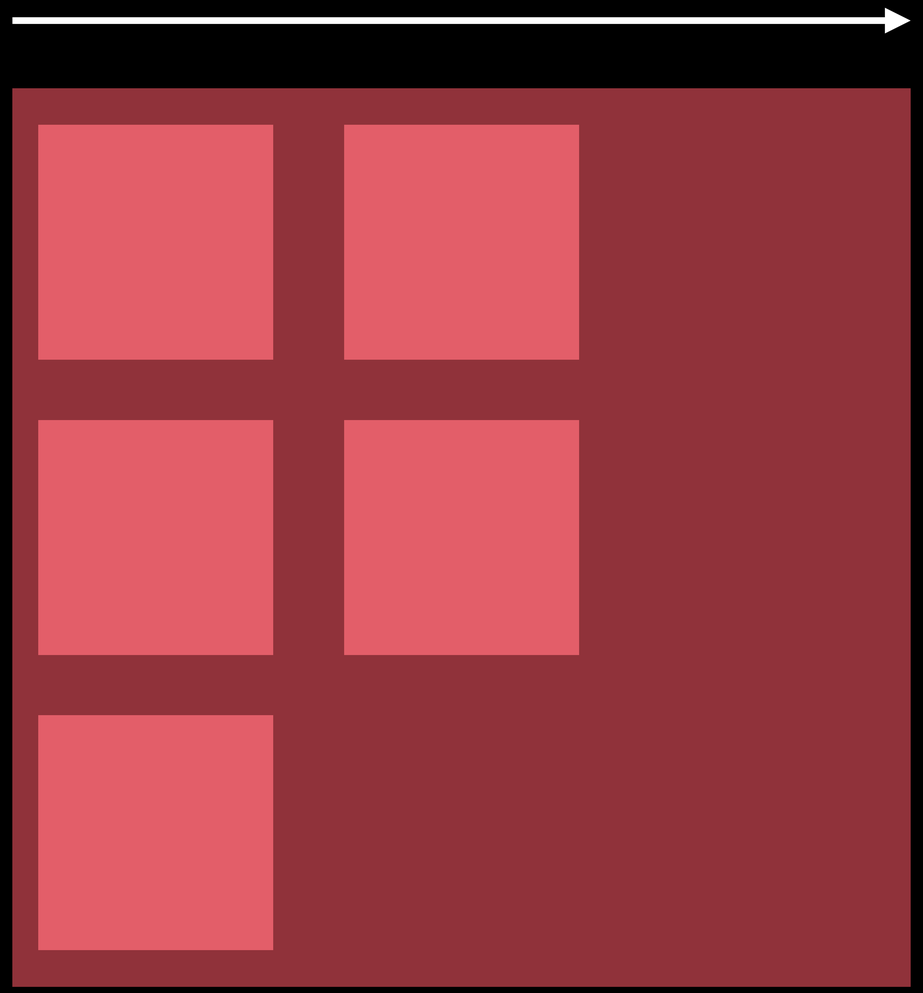
Flow Layout



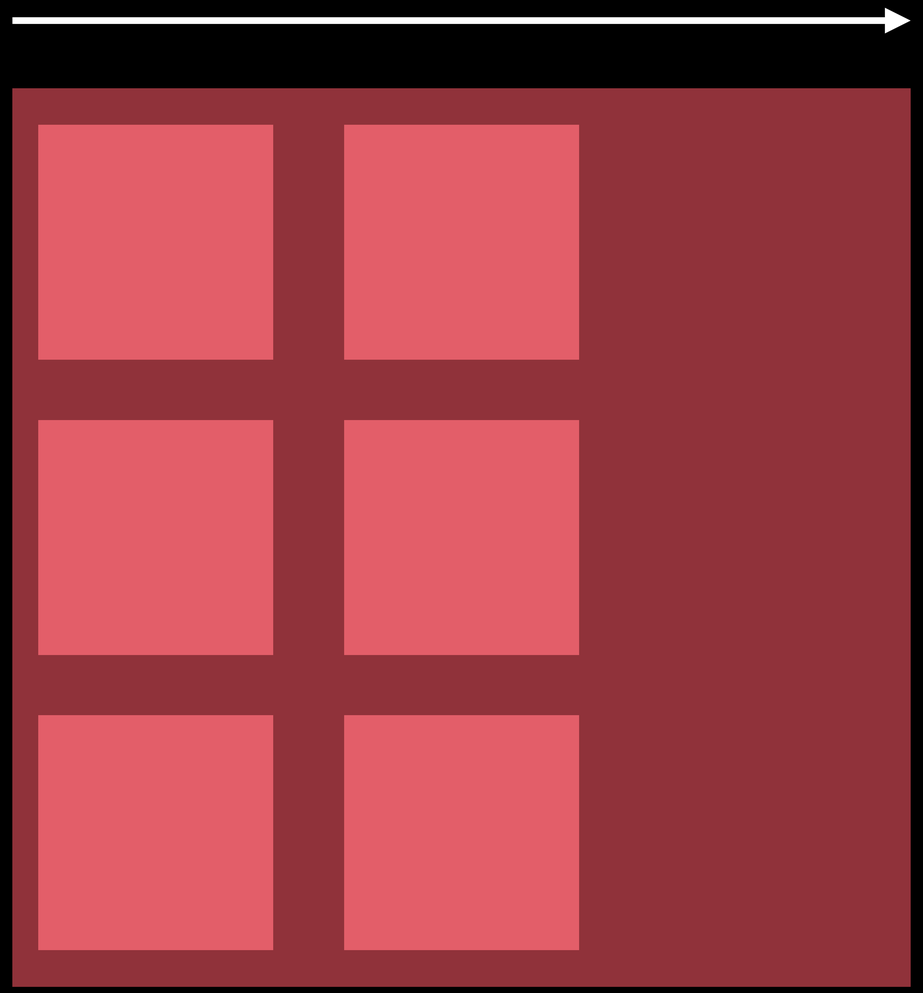
Flow Layout



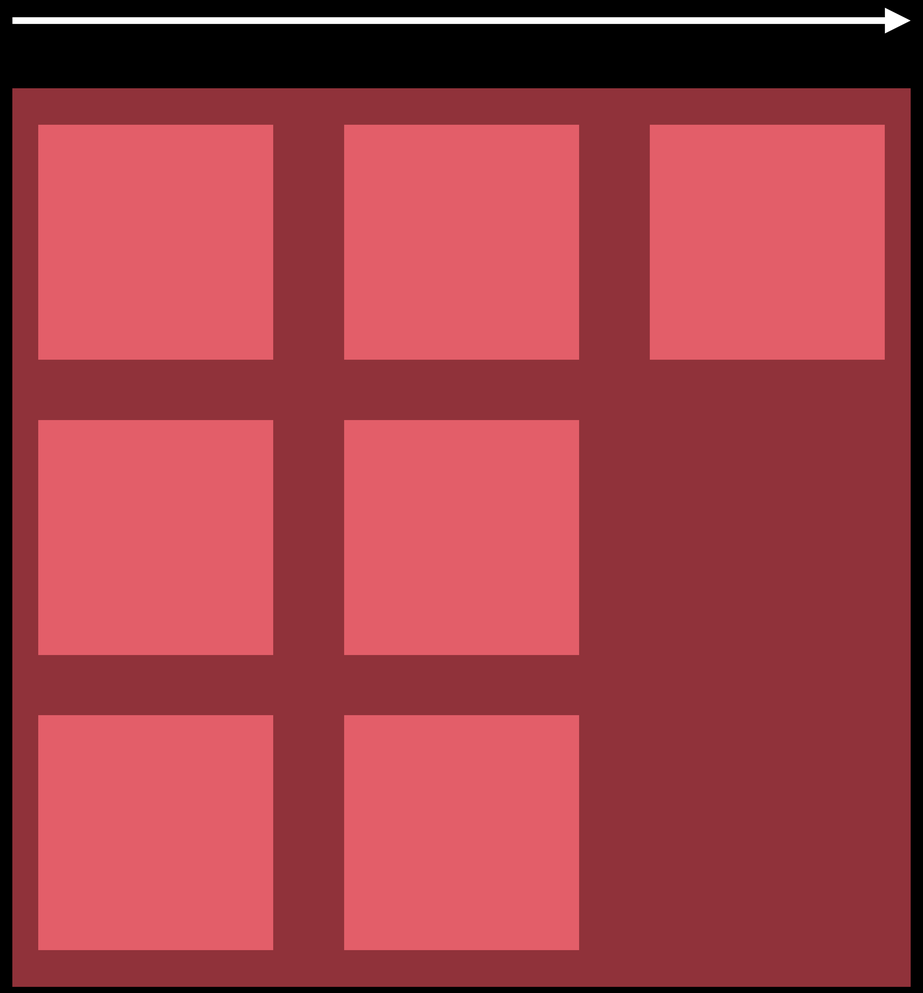
Flow Layout



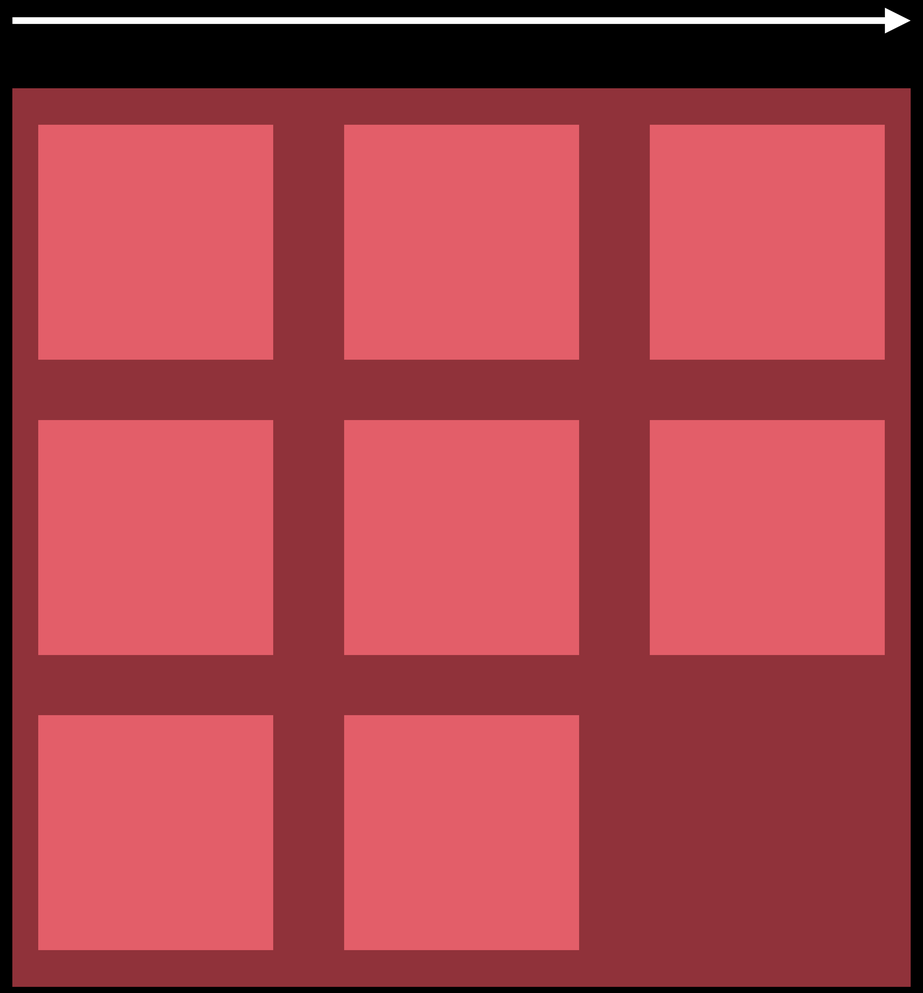
Flow Layout



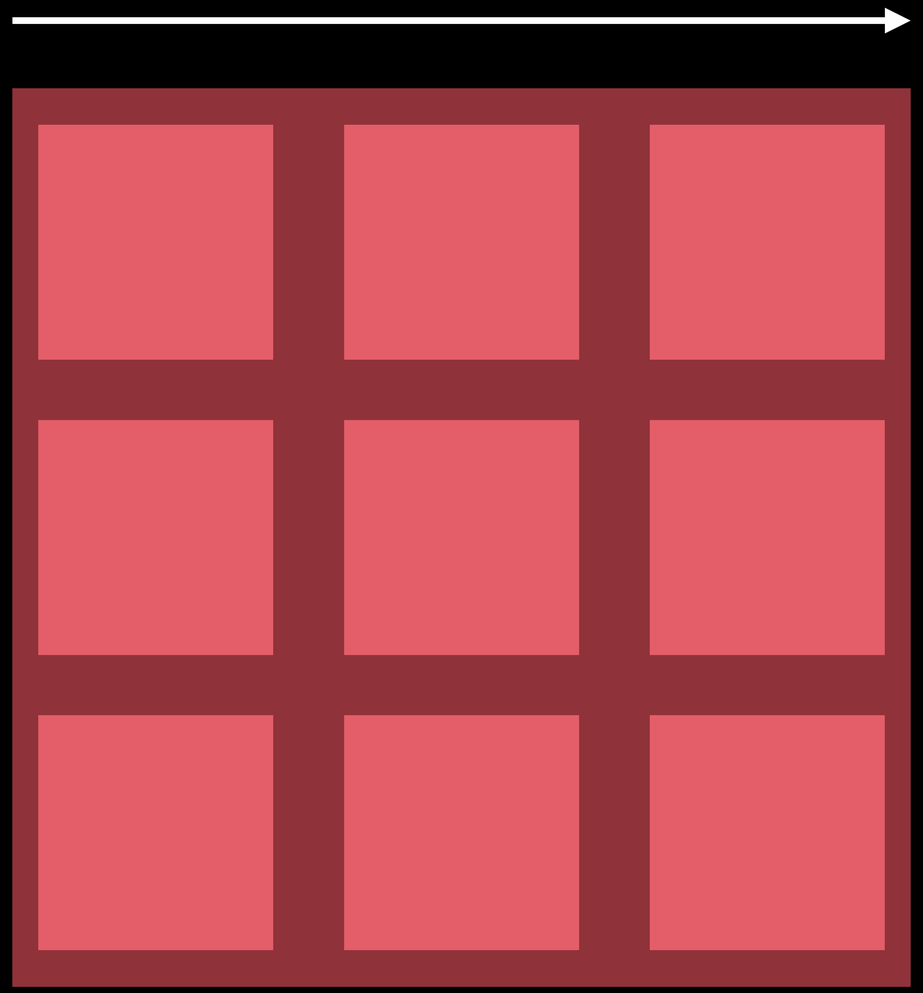
Flow Layout



Flow Layout

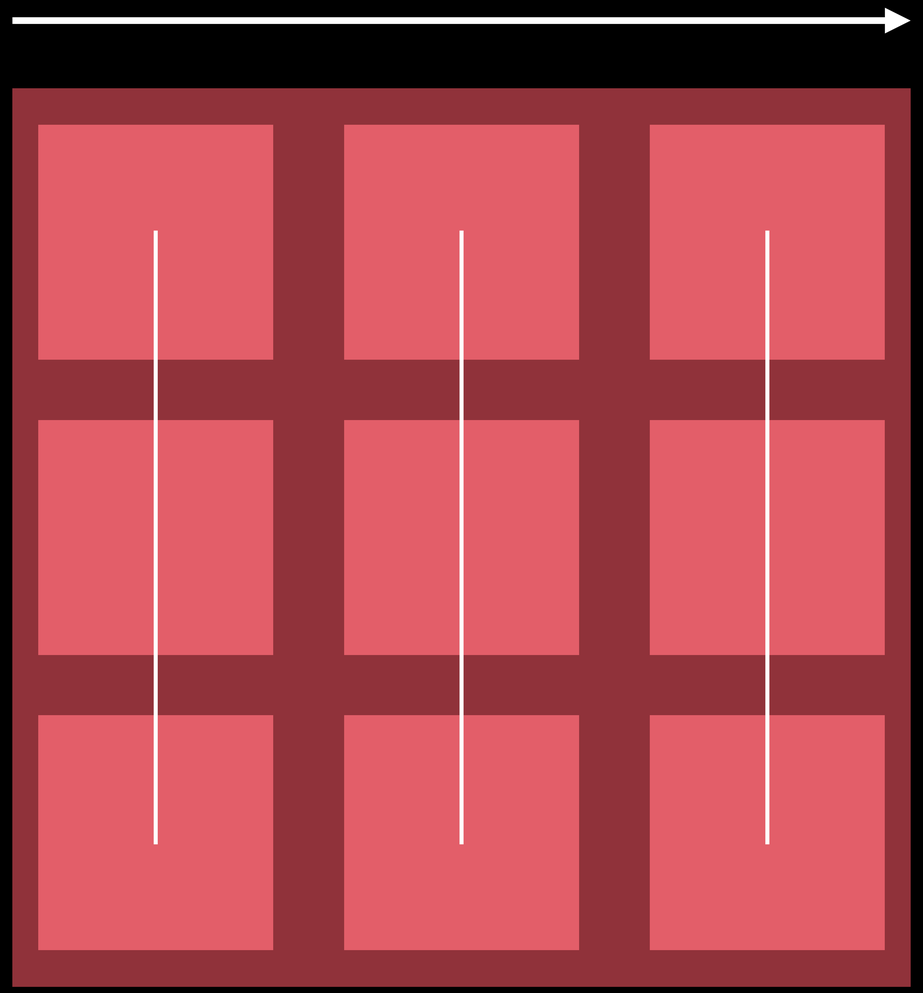


Flow Layout



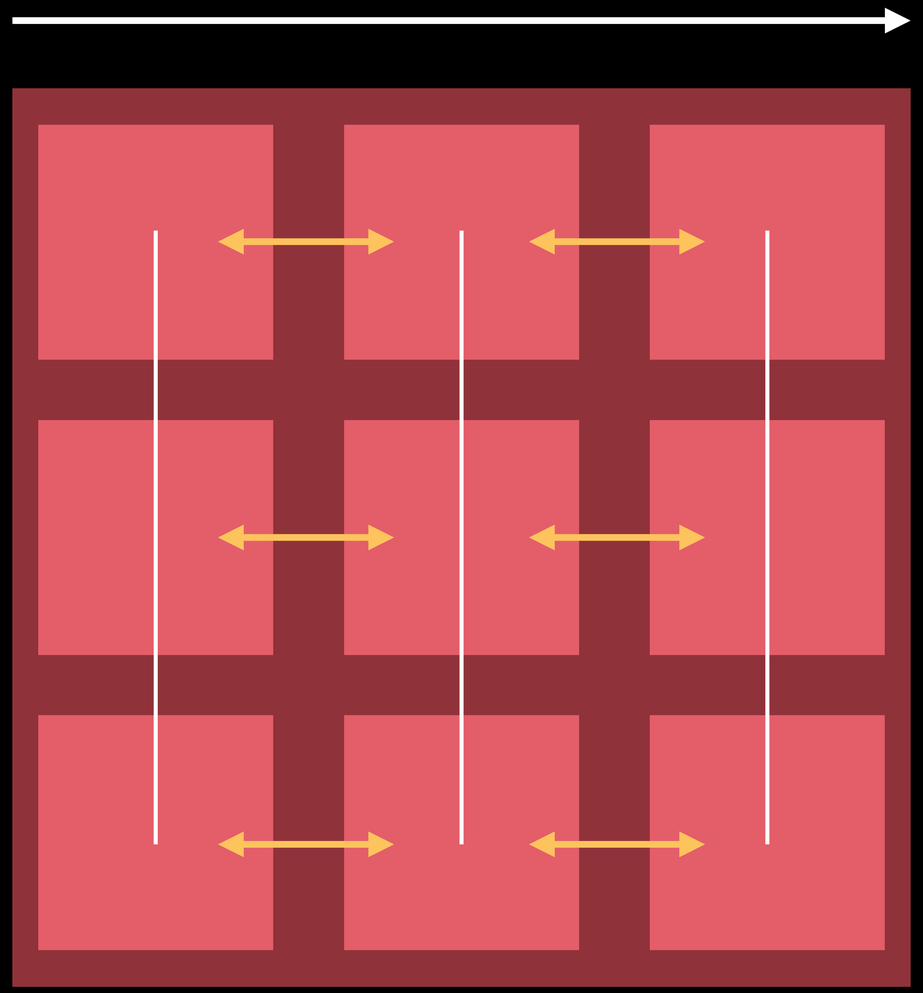
Flow Layout

Line Orientation



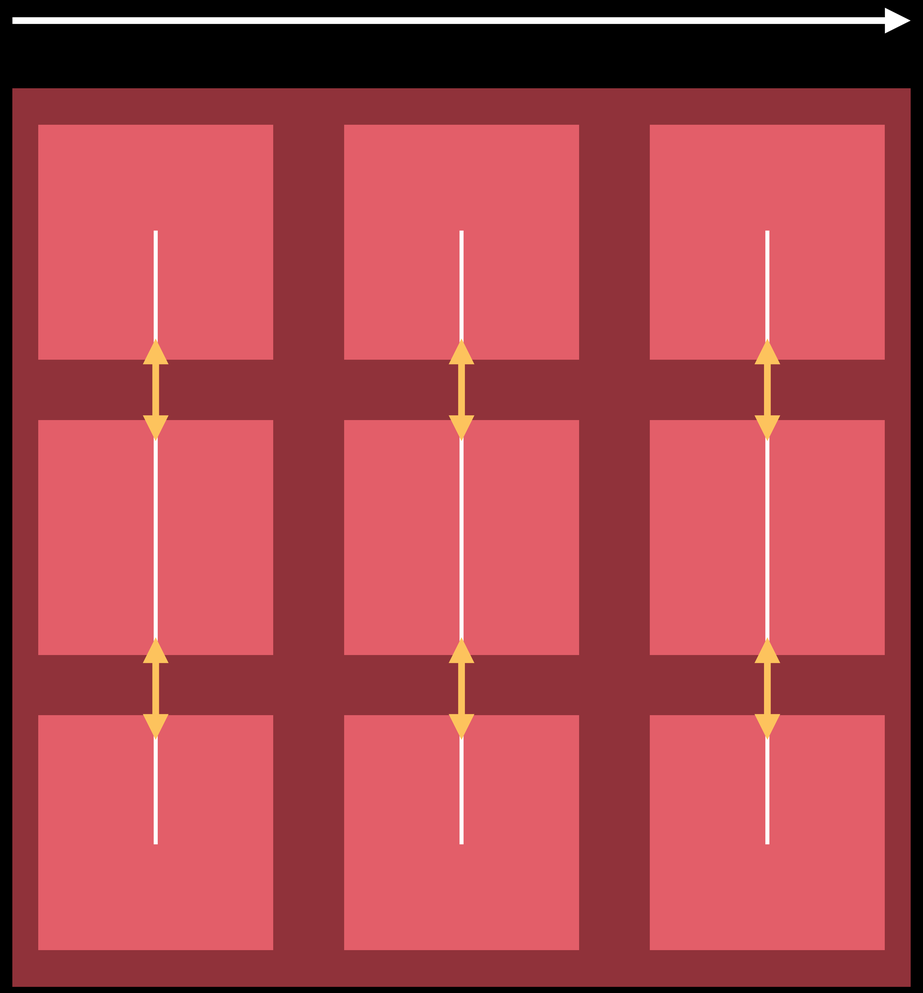
Flow Layout

Line Spacing



Flow Layout

Inter-Item Spacing



UICollectionViewDataSource

Provides content

Section and item counts

```
optional func numberOfSections(in collectionView: UICollectionView) -> Int

func collectionView(_ collectionView: UICollectionView, numberOfItemsInSection section: Int)
    -> Int

func collectionView(_ collectionView: UICollectionView, cellForItemAt indexPath: IndexPath)
    -> UICollectionViewCell
```

UICollectionViewDelegate

Optional

Extends `UIScrollViewDelegate`

Fine-grained control

- Highlighting
- Selection

View appearance events

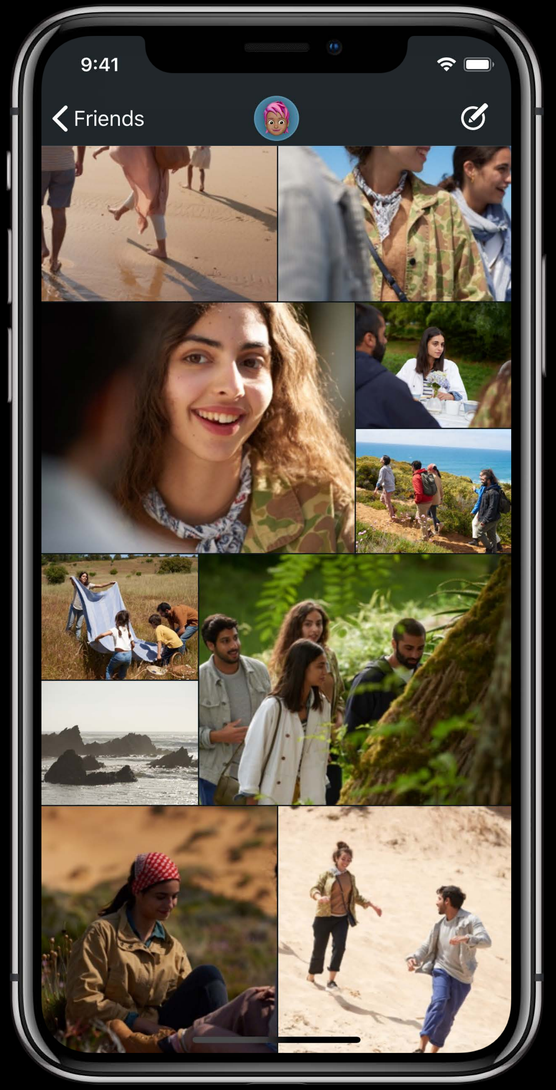
- `willDisplayItem`
- `didEndDisplayingItem`

Demo

Getting started with UICollectionView

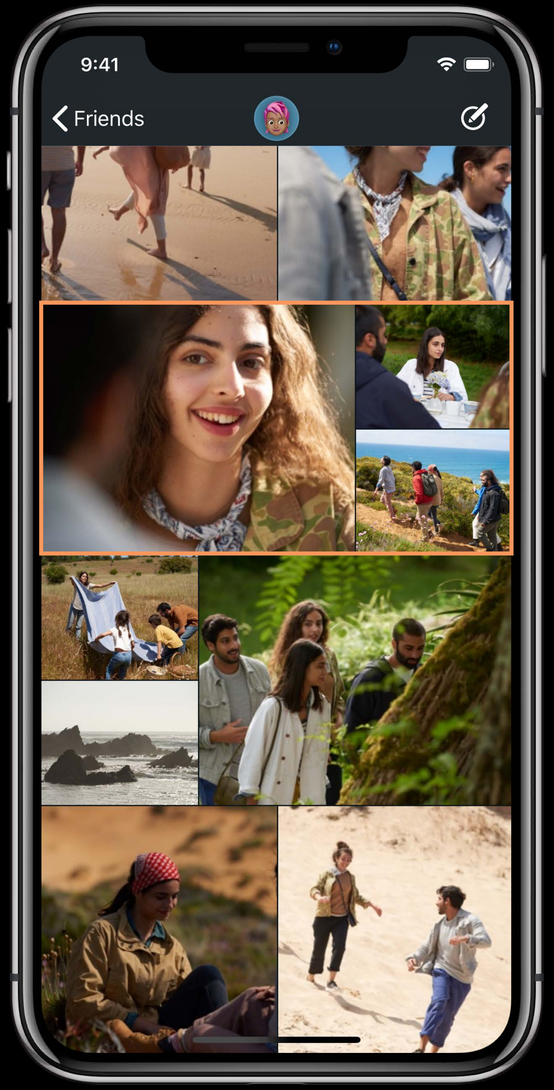
When to Go Custom?

Can we use UICollectionViewFlowLayout?



When to Go Custom?

Can we use UICollectionViewFlowLayout?



When to Go Custom?

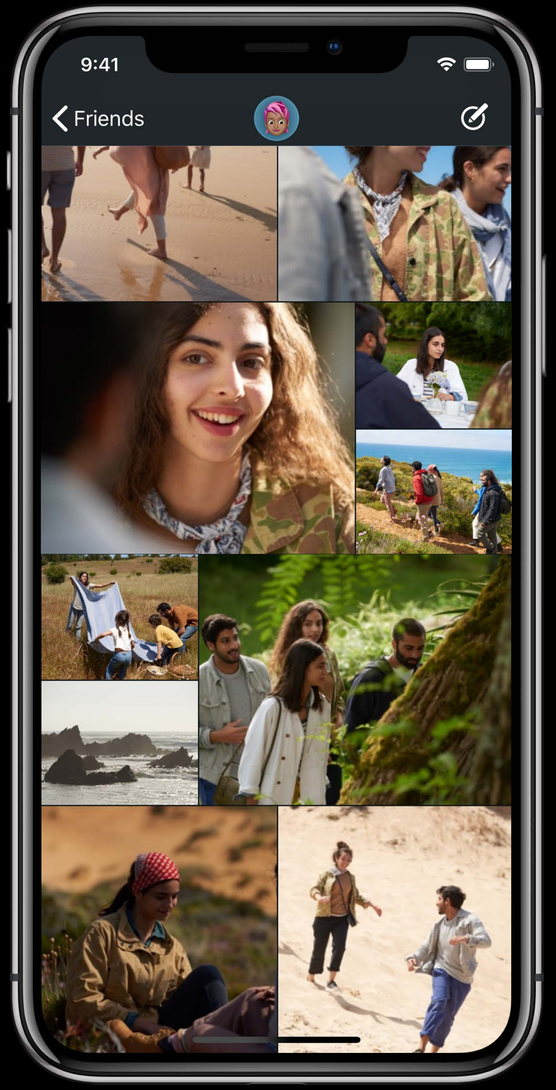
Can we use UICollectionViewFlowLayout?

When to Go Custom?

When to Go Custom?

When to Go Custom?

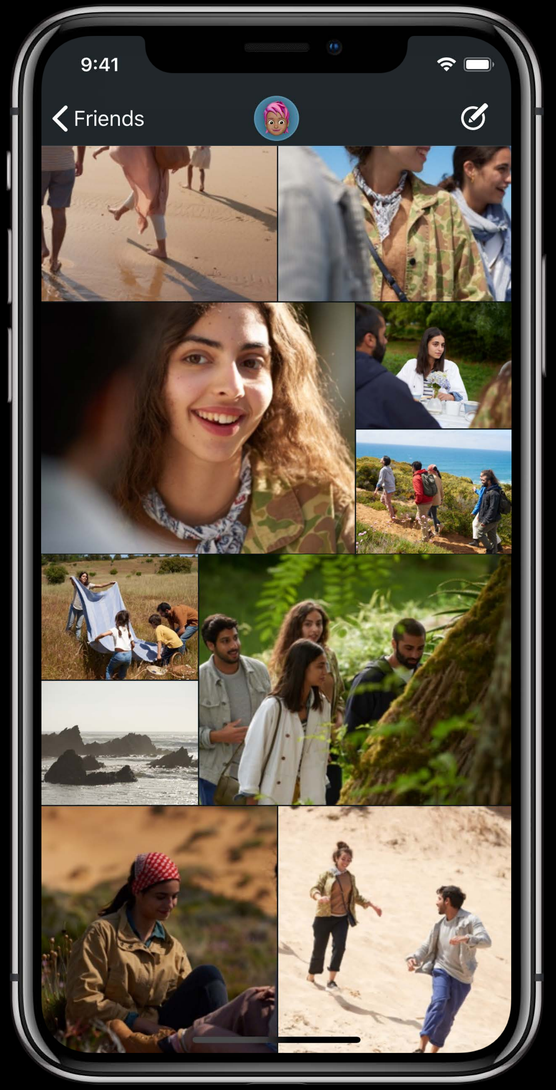
Our fancy layout is not line-based



When to Go Custom?

Our fancy layout is not line-based

Start with flow before going custom!



Creating a Custom UICollectionViewLayout

Not complicated!

Four basic methods

...and one additional method to consider...

Providing Content Size

```
open var collectionViewContentSize: CGSize { get }
```

Size of bounds which contains all items

Needed for `UIScrollView.contentSize`

Providing Layout Attributes

```
func layoutAttributesForElements(in rect: CGRect) -> [UICollectionViewLayoutAttributes]?

func layoutAttributesForItem(at indexPath: IndexPath) -> UICollectionViewLayoutAttributes?
```

Query by geometric region

Query by IndexPath

Performance matters

Preparing the Layout

```
func prepare()
```

Called for every `invalidateLayout`

Cache `UICollectionViewLayoutAttributes`

Compute `collectionViewContentSize`

Handling Bounds Changes in Your Custom Layout

```
func shouldInvalidateLayout(forBoundsChange newBounds: CGRect) -> Bool
```

Called for every bounds change

- Size change
- Origin change

Called during scrolling 😳

Default implementation returns `false`

Demo

Our fancy custom UICollectionView Layout

Our Totally Cool Update Animation

1. Reload last item
2. Move last -> first
3. Delete 3rd item

Our Totally Cool Update Animation

1. Reload last item
2. Move last -> first
3. Delete 3rd item

Debugging Update Exceptions 😐

```
*** Terminating app due to uncaught exception
'NSInternalInconsistencyException', reason: 'attempt to
perform a delete and a move from the same index path
(<NSIndexPath: 0xc000000000000016> {length = 2, path =
0 – 3})' ****
```

Debugging Update Exceptions 😐

```
*** Terminating app due to uncaught exception
'NSInternalInconsistencyException', reason: 'attempt to
perform a delete and a move from the same index path
(<NSIndexPath: 0xc000000000000016> {length = 2, path =
0 - 3})' ****
```

Debugging Update Exceptions 😐

```
*** Terminating app due to uncaught exception
'NSInternalInconsistencyException', reason: 'attempt to
perform a delete and a move from the same index path
(<NSIndexPath: 0xc000000000000016> {length = 2, path =
0 - 3})' ****
```

Wait … delete? 🙄

performBatchUpdates()

```
func performBatchUpdates(_ updates: (() -> Void)?, completion: ((Bool) -> Void)? = nil)
```

Animate updates together

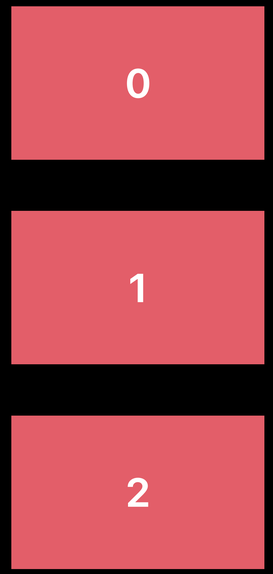
Perform data source updates and collection view updates in updates closure

Collection view updates ordering does not matter…

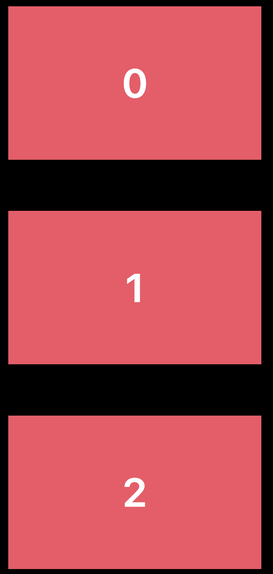
…data source updates ordering does matter

Data Source Updates: Order Matters

Delete 0 → Insert 1

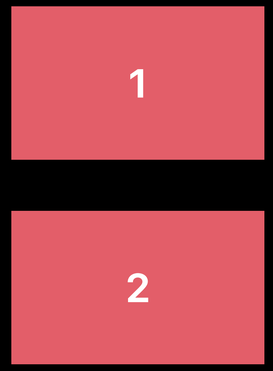


Insert 1 → Delete 0

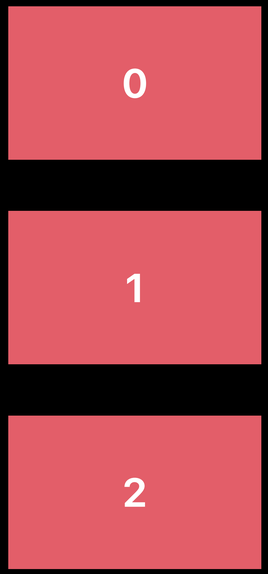


Data Source Updates: Order Matters

Delete 0 → Insert 1



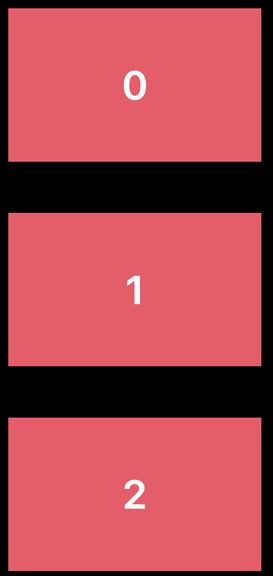
Insert 1 → Delete 0



Data Source Updates: Order Matters

Delete 0 → Insert 1

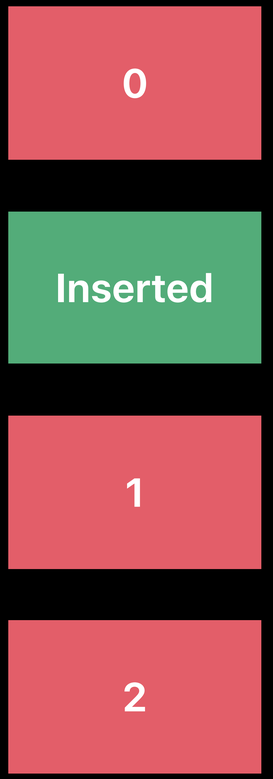
Insert 1 → Delete 0



Data Source Updates: Order Matters

Delete 0 → Insert 1

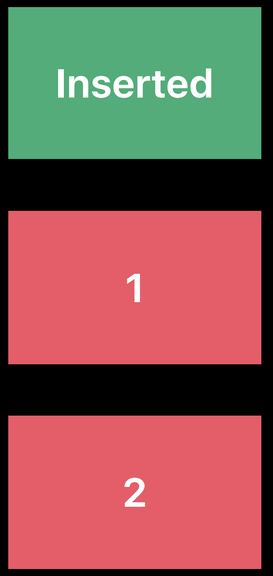
Insert 1 → Delete 0



Data Source Updates: Order Matters

Delete 0 → Insert 1

Insert 1 → Delete 0



```
// Both of These Collection View Updates Are Equivalent!

let insertedIndexPath = IndexPath(item:0, section:0)
let deletedIndexPath = IndexPath(item:1, section:0)

let updates1 = {
    collectionView.insertItems(at: insertedIndexPaths)
    collectionView.deleteItems(at: deletedIndexPaths)
}
collectionView.performBatchUpdates(updates1)

let updates2 = {
   collectionView.deleteItems(at: deletedIndexPaths)
   collectionView.insertItems(at: insertedIndexPaths)
}
collectionView.performBatchUpdates(updates2)
```

Collection View Updates Coalescing

Action	Notes	Index Path Semantics
Delete	Descending IndexPath order	Before updates
Insert	Ascending IndexPath order	After updates
Move		From: Before updates To: After updates
Reload	Decompose: Delete and inserts	Before updates

Update Combinations That Don’t Make Sense

Exceptions will result from:

- Move and Delete the same location
- Move and Insert to the same location
- Move more than 1 location to the same location
- Referencing an invalid IndexPath

Applying Data Source Updates

Decompose Move into Delete and Insert updates

Combine all Delete and Insert updates

Process Delete updates first, in descending order

Process Insert updates last, in ascending order

What About reloadData()?

No updates required

Resync data source to collection view

Not animated

“The Sledgehammer”

Demo

Saving our totally cool update animation

Call to Action

- Build custom layouts
- Update animations
- Reload data

More Information

https://developer.apple.com/wwdc18/225

UIKit and Collection View Lab	Technology Lab 11	Friday 9:00AM


WWDC18