ATLS 4120/5120: Mobile Application Development
Week 2: Adaptive Layout

Adaptive Layout
The goal of adaptive layouts is to have the user interface look good and work well on all iOS devices in all orientations.
Interface Builder gives us the tools to do this using auto layout, constraints, and size classes.

Orientation
- iOS devices can be used in portrait or landscape mode.
- There are four defined iOS interface orientations values for UIInterfaceOrientation(look at doc)
- iPhone
  - Most apps support autorotation but not all
    - Movies can be watched in landscape only
    - Contacts can only be edited in portrait mode
    - Immersive games often support only 1 orientation
  - Upside down portrait is not usually supported
- iPad
  - Recommended that all apps support every orientation
- For iPhone apps, the base rule is that if autorotation enhances the user experience, you should add it to your application. For iPad apps, the rule is you should add autorotation unless you have a compelling reason not to.
- Interface orientation can be set for the whole application and that will be the default for all view controllers
- You can also set valid orientations for each view controller.
- When the user rotates the device the active view controller calls **supportedInterfaceOrientations()** to see if the new orientation is supported. If it is then the app's window and view will be rotated and resized to fit the new orientation.
- Xcode
  - Target | Deployment info

Auto Layout
https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/AutolayoutPG/index.html#//apple_ref/doc/uid/TP40010853-CH7-SW1
Auto Layout enables you to create a user interface that responds to different screen sizes, orientations, and localization.
Auto layout was first introduced in iOS6 and Xcode 4 and improved in Xcode 5 and 6
Auto layout also helps with internationalization as it can resize your labels based on the content

Constraints
https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/AutolayoutPG/AnatomyofaConstraint.html#//apple_ref/doc/uid/TP40010853-CH9-SW1
- Constraints express rules for the relationship between UI elements in your interface
- Auto Layout uses these rules to automatically create your user interface when your app runs
- Creating constraints
  - IB can add suggested or missing constraints based on your interface layout
  - Create your own constraints
  - You can change constraints

- For each element there must be enough constraints to determine its placement
  - horizontal position
  - vertical position
  - size: height, width
- Constraint properties
  - Attributes: leading and trailing are the default
  - Values: size or offset of constraint in points
  - Relation: type of relationship between elements
  - Priority: required, high, or low

https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/AutolayoutPG/WorkingwithConstraintsinInterfaceBuidler.html#//apple_ref/doc/uid/TP40010853-CH10-SW1

- Align: align edges or centers of elements to each other or within the container
- Pin: sets width or height spacing for an element or to a view
  - Leading space (from left for English)
  - Trailing space (from right for English)
  - Top
  - Bottom
- Constraints are color coded
  - Blue: constraints are valid
  - Orange: constraints are misplaced or ambiguous
    - Dotted orange box shows constraints with mismatched views
    - Need to update frames or update constraints
  - Red: constraints are conflicting
- Layout issues are listed in the document outline and provide suggested fixes
- You can clear, reset, change, or add missing constraints to resolve issues
- Once you think you have the right constraints, Update Frames to see what it will look like for different class sizes using Preview

Size classes
(slide)

- Size classes enable a storyboard to work with all iOS device screen sizes and orientations
- Build your interface as it will look on most devices and orientations
- Change the interface for different size classes
  - Size or position of a view
  - Add or uninstall a view or constraint
- When you change the size class any changes you make are only for that size class

daVinci

Let's try to get our da Vinci app to look good on different size screens.(change to universal if needed)
Move the UI controls to where you'd like them.
Select the View Controller and click the right Resolve Auto Layout Issues and under All Views in View Controller chose Add Missing Constraints.
This works for simple apps but if it doesn't work go back into the same menu and chose Clear Constraints.
Let's work on one UI control at a time.

Label
X position: align center to superview by using Align | Horizontally in Container
Y position: Pin top space to top margin

Update | Selected Views | Update frames
Don't need to specify width or height.
Many of the views that UIKit provides, including UILabel, are capable of having Auto Layout set their size based on their actual content. They do this by calculating their natural or intrinsic content size. At its intrinsic size, the label is just wide enough and tall enough to completely surround the text that it contains. When we run the application and click one of the buttons, the label's text will be set and its intrinsic content size will change. Try it.
Use Preview to see how the label looks on different size devices.

Buttons
It makes sense to always have these buttons aligned vertically.
Stack views provide a way to layout a series of views horizontally or vertically. By configuring a few simple properties such as alignment, distribution, and spacing, you can define how the contained views adjust themselves to the available space.

Click on one button and then shift click to select the second.
Editor | Embed in | Stack View or click the Stack icon on the bottom right, left most button.
Now we need to position the stack view so make sure it's selected the whole time now using the document outline.
I'd like to have it aligned with our title.
X position: Control click and drag from the stack view to the label. This tells it you want to connect these two views. Add Align leading constraint.
Y position: Control click and drag from the stack view to the label. Add vertical spacing – top space constraint. Show how to go in the size inspector and change the value.
Width: Control click and drag from the stack view to the label. Add equal widths constraint.
Now you have all the constraints you should need. If you have a red arrow at the top of your document outline click on it to see what your missing.
If you see orange lines in your view select the stack view and Resolve Auto Layout Issues | Selected View | Update Frames
Now it will show you where the buttons will be based on the constraints.
Use Preview to see how the buttons look on different size devices.

Image
X position: align center to superview by using Align | Horizontally in Container
Y position: Control click on the image and drag to the stack view to add a vertical top space constraint(20)
I want to make sure the image's aspect ratio never changes.
Pin | Aspect Ratio (220:299)
(all above are required)
Now if you look in preview it looks ok, but the image isn't scaling larger on the iPad. It could even be bigger on the iPhone.
If I pin the width and height to the size of the image then it will be cut off on the iPhone.
Pin width = 440
Pin height = 598
I really want these to be this width and height or smaller, so change both of these to "Less Than Or Equal".
I don't want them larger because that's the size of the image and scaling up would look bad. The best thing to do is find a high resolution image and then scale down.

Now to make sure the image is as big as space allows I want it to scale up to the superview width if it can.
Control click from the image and drag to the right and chose Equal Widths. Make this constraint High Priority(750) because we still want width <=440 to be required. Change the constant to -20 if you want a border on the right and left.
Then I pinned the image to the bottom superview with a constant of 10 so that it never went past the bottom margin.

Landscape
If you rotate(command arrow) a smaller iPhone to landscape the buttons disappear because it's running out of room.
If you pin the stack view height(30) then you lose the top label.
I left the height constraint high priority but changed the stack view top constraint to the label to high priority instead so if there's not enough room it won't force this.

Size classes
If you can't get the layout to work on all sizes and orientations, or you want a different layout, you can specify a different layout for a certain size class.
I'm going to change the layout for iPhones (other than plus) in landscape.
Change the size class to w Compact h Compact
Notice the bottom bar turns blue meaning we are only editing for this size class.
Move the views around so the label is still centered but the buttons are on the left and the image on the right instead of being stacked.
Stack view:
Select the stack view (document outline) and in the size inspector for any constraint double click on it, click on the + next to installed and chose our current size class, Compact width | Compact Height.
This adds a row for this size class. Now uncheck installed for this size class. Notice the constraint is still installed for other size classes.
Do this for all the stack view constraints except the top space to label bottom.
Add constraints
Y position: Top space to label bottom(50)
X position: Leading space to superview (-60)
Height: I pinned the height to 90 and since the stack view is already set to distribution: equal spacing, this spreads out the buttons vertically.
Notice that the new constraints are only installed for the wC hC size class.
Update frame(just for the stack view) to see where it will be placed.

Image:
Repeat what we did above to remove the center X constraint, equal width to superview, and bottom margin to superview.
X position: Trailing space to superview (60)
Y position: Top space to label bottom(20) and Bottom space to bottom layout guide(>= 20) to ensure there's a border.
Width: pin to 180 but make it high(not required) priority so it will make it bigger if it can.

This should make the layout in two columns for size class compact | compact and not change the others.