Autista App – Documentation

Application Overview 3

Application Views 4

AppDelegate.m (h) 9

didFinishLaunchingWithOptions 9

applicationDidBecomeActive 9

fetchCurrentUser 9

managedObjectContext 9

managedObjectModel 10

persistentStoreCoordinator 10

primeCoreDataStoreWithData 10

Application’s First Run: 10

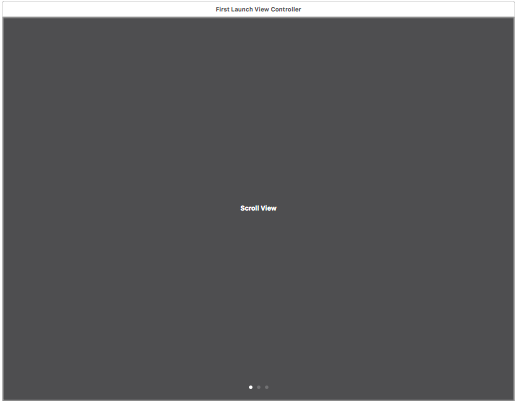
FirstLaunchViewController.m (h) 10

# Application Overview

# Application Views

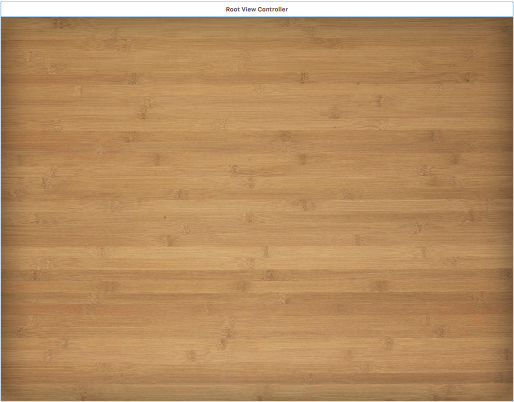
Autista application embodies nine views including:

1. **First Launch View:** this view will be the first entry to Autista application if this application is opening on a device for the first time. This view contains a scroll view element and a page control element.

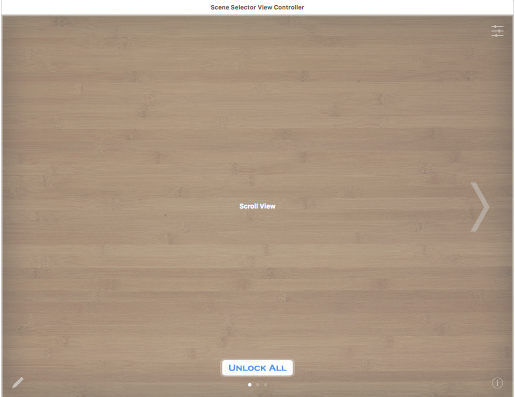


SceneSelectorViewController

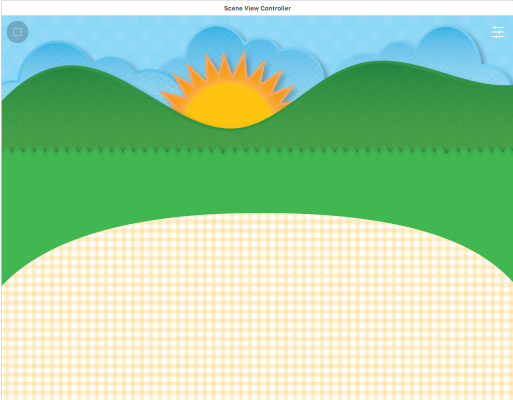
1. **Root View**



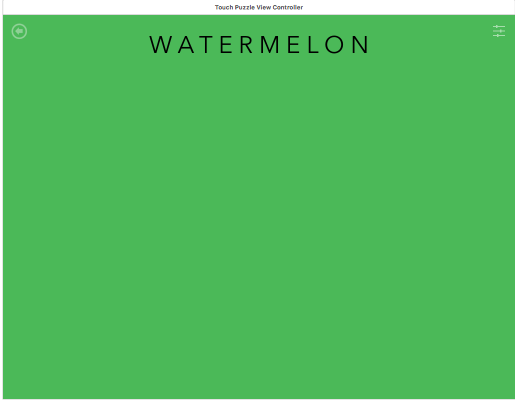
1. **Selector View**



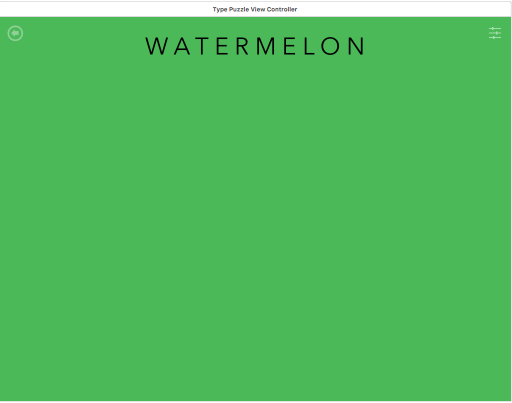
1. **Scene View**



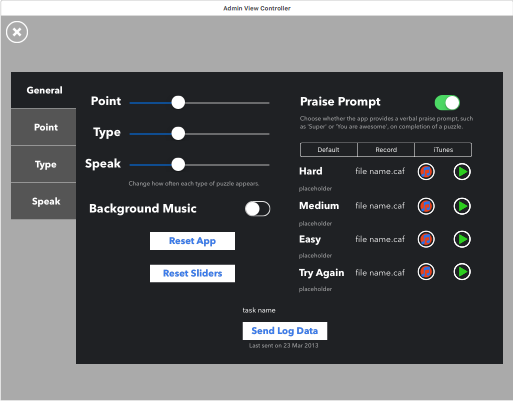
1. **Touch Puzzle View**



1. **Type Puzzle View**



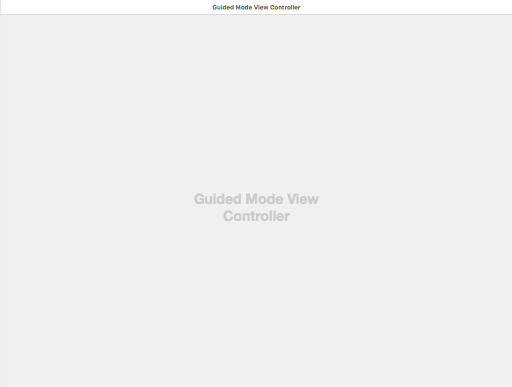
1. **Admin View**



1. **Say Puzzle View**



1. **Guided Mode View**



**GlobalPreferences.h**

The application starts from **main.m (h)** which points to **AppDelegate.m (h).**

## AppDelegate.m (h)

### didFinishLaunchingWithOptions

Should return YES if everything is gone through successfully

This method/constructor uses InstaBug SDK for In App bug reporting so the users can draw, comment, and report the bugs on different views of the app. (Line 40)

This method/constructor uses uses AutistaIAPHelper which is for In App Purchase which is set as default by Apple.

It gets the current app version from the bundle in Autista-Info.plist pList under /Supporting Files. The "Version" input corresponds to CFBundleShortVersionString in the info.plist file, and the "Build" input corresponds to CFBundleVersion.



Line 45: create object form GlobalPreferences.m to use preferencs such as sendData\_preference, backgroundMusicEnabled, sendAnonymousData, etc.

Line 47: restore the preferences in line 45 from the last saved on the device.

Line 49-61: checks if this is the first time that the app is loading, or it is a updated version of the app on the device, or the app is running normally.

Line 63: -> EventLogger 101-148, the event logger in case the admin mode selected, a piece is dragged, a piece is being dragged, a piece is being released, normal app operation, etc.

### applicationDidBecomeActive

Restart any tasks that were paused (or not yet started) while the application was inactive. If the application was previously in the background, optionally refresh the user interface.

Here we copy over changes that may have been made in the app Settings while we were in the background – this is mostly for user information.

### fetchCurrentUser

Fetches the information of the current user

### managedObjectContext

Returns the managed object context for the application. If the context doesn't already exist, it is created and bound to the persistent store coordinator for the application.

### managedObjectModel

Returns the managed object model for the application. If the model doesn't already exist, it is created from the application's model.

### persistentStoreCoordinator

Returns the persistent store coordinator for the application. If the coordinator doesn't already exist, it is created and the application's store added to it.

### primeCoreDataStoreWithData

# Application’s First Run

When the application starts running for the first time, the **AppDelegate** (Lines 49-61) will force the app to launch the First Launch View and the starting point for the code will be **FirstLaunchViewController.m (h)**.

The application’s first run classes are located in the First Launch folder.

## FirstLaunchViewController.m (h)

The data (text and logo) will be loaded from a pList (e.g. XML format) created as **FirstLaunchInfoTextNew.plist** in /Resources.

Number of pages for the first launch view will be automatically decided based on the number of elements in the pList.

Changes the size of the scroll view to fit the height of the device and the width of total number of pages (calculated from the number of items in pList) multiplied by the width of the device.

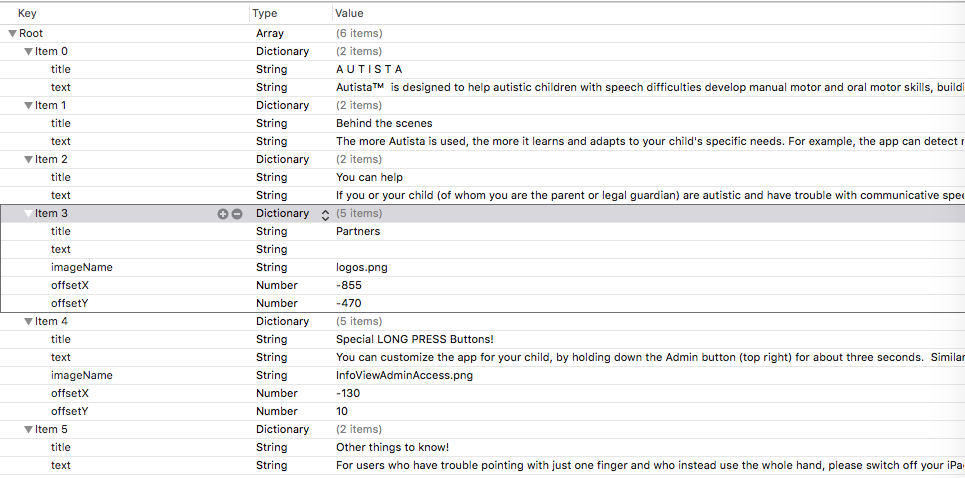


Figure - PList of information for application's first launch

Lines 58 – 135 in this class creates frames with the contents of the key-value pairs read from the pList and associates actions such as page controller action to the view. Depending the index of each item in pList, this class uses three subclasses (frames) for loading the contents including **InfoViewPageOne**, **InfoViewPageThree**, and **InfoView**. The **InfoViewPageOne** is used to show the normal text information to the user; the **InfoViewPageThree** will show text information and logos of partners/supporters in the app; the **InfoView** loads the last item in the pList and it will contain a button (Get Started) to dismiss the introductory part of the application. The Main View of the application will be appeared after this dismissal.

# Puzzle Views

## TypePuzzleViewController.m (h)

After the initiation of preferences, keyboard’s data, background, font, and puzzle pieces in **ViewDidLoad()**, code switches to **initializePuzzleState()**.

In **initializePuzzleState()**, a placeholder (the background transparent image of the selected puzzle) is added to the view and the soft keyboard slides out by calling **slideInKeyboard()**. It then takes the title of the current puzzle in uppercases and reads the characters in the current title and creates enabled buttons for the pop up keyboard for these characters. The rest of the keys in the keyboard will be loaded as blank. In this method, then a method of **playObjectTitleSound()** is called and then the soft keyboard slides in by calling **slideInKeyboard().** **playObjectTitleSound()** plays the title sound for the selected puzzle, e.g. says (plays) MUG for mug.

**slideInKeyboard()** is the next step in this the **TypePuzzleViewController.m.**